Školicí materiál

NÁVRHÁŘ OBJEKTŮ K2

Verze K2 ori.04

OBSAH

1. ÚVOD 7
1.1. CÍL ŠKOLENÍ 7
1.2. PŘEDPOKLADY 7
2. SEZNÁMENÍ S NÁVRHÁŘEM OBJEKTŮ 8
2.1. ZALOŽENÍ A NASTAVENÍ NOVÉ DEFINICE ROZŠÍŘENÍ 8
2.1.1. SOUBOR S DEFINICÍ ROZŠÍŘENÍ 9
2.2. NASTAVENÍ NÁVRHÁŘE OBJEKTŮ 9
2.2.1. ZÁKLADNÍ INFORMACE O ROZŠÍŘENÍ 10
2.2.2. JMENNÝ PROSTOR 11
2.2.3. ZPĚTNÁ KOMPATIBILITA 12
2.2.4. ARCHIVACE 13
2.3. SPUŠTĚNÍ NÁVRHÁŘE OBJEKTŮ 14
2.3.1. POPIS ZÁLOŽEK ZÁKLADNÍHO FORMULÁŘE 14
3. ZÁLOŽKA „MODULY“ 15
3.1. POPIS SLOUPCŮ SEZNAMU ZÁKLADNÍHO FORMULÁŘE 15
3.2. MOŽNOSTI ROZŠÍŘENÍ 17
4. VYTVÁŘÍME DATOVÝ MODUL 19
4.1. MODUL 19
4.1.1. ZÁKLADNÍ NASTAVENÍ 20
4.1.2. CÍLOVÁ PLATFORMA 20
4.1.3. TABULKA 20
4.1.4. DATOVÝ MODUL 21
4.1.5. FORMULÁŘ 22
4.2. POLE 22
4.2.1. SKALÁR 24
4.2.2. CIZÍ KLÍČ 36
4.2.3. PŘÍMÁ VAZBA 47
4.2.4. RYCHLÁ ZMĚNA TYPU POLE – FUNKCE „PŘEMĚNIT NA“ 50
4.2.5. POLOŽKY 50
4.2.6. NEVLASTNĚNÉ POLOŽKY 56
4.2.7. VÝČET HODNOT 62
4.2.8. VÝSTUP V DATABÁZI 64
4.3. ROZDÍLY V NASTAVENÍ POLÍ DLE PŘEDKA DATOVÉHO MODULU 64
4.3.1. NASTAVENÍ POLÍ U PŘEDKA TYPU TTYPEDATAM 64
4.4. KLÍČE 65
4.4.1. ZÁKLADNÍ ÚDAJE 67
4.4.2. VLASTNOSTI 67
4.4.3. SLOŽKY KLÍČE (SEGMENTY) 69
4.5. ZDĚDĚNÉ PROPERTY 71
4.5.1. VLASTNOSTI PŘEDKA TBASEDATAM 73
4.5.2. VLASTNOSTI PŘEDKA TTYPEDATAM 82
4.5.3. VLASTNOSTI PŘEDKA TCHILDDATAM 91
4.5.4. VLASTNOSTI PŘEDKA TCUSTOMDATAM 92
4.5.5. VLASTNOSTI PŘEDKA TCUSTOMLISTDATAM 93
4.5.6. VLASTNOSTI PŘEDKA TCUSTOMDOCUMENTDATAM 94
4.5.7. VLASTNOSTI PŘEDKA TONERECORDDM 108
4.5.8. VLASTNOSTI PŘEDKA TMTBASEDM 120
4.6. VLASTNÍ PROPERTY 120

4.6.1. JEDNODUCHÁ PROPERTY (VLASTNOST) 121
4.6.2. PROPERTY TDATAM 126
4.6.3. FILTROVACÍ PARAMETR 131
4.7. AKCE 140
4.7.1. ZÁKLADNÍ NASTAVENÍ 142
4.7.2. ZÁLOŽKA „HINT“ 144
4.7.3. ZÁLOŽKA „EXECUTE“ 144
4.7.4. ZÁLOŽKA „UPDATE“ 144
4.8. METODY 147
4.8.1. METODA „BEFORESAVE“ 148
4.8.2. METODA „AFTERSAVE“ 149
4.8.3. METODA „PRAVONAREC“ 149
4.8.4. METODA „GETVALUEBYPARAMS“ 151
4.8.5. METODA „DEFINELINKS“ 155
4.8.6. METODA DEFINEFIELDS 155
4.8.7. METODA „AFTERENDEDIT“ 155
4.8.8. METODA NO_BEFOREPREVIEW 155
4.8.9. METODA „CHECKFIELDS“ 156
4.8.10. METODA „DEFAULTVALUES“ 156
4.8.11. METODA „ACCESSDENIED“ 158
4.8.12. METODA „ENABLEDFIELD“ 160
4.8.13. METODA „RECORDISHIDDEN“ 161
4.8.14. METODA „LOADCHILDDATA“ PŘEDKA CHILDDATAM 161
4.8.15. METODA „INITCHILDDATA“ PŘEDKA CHILDDATAM 162
4.9. VLASTNÍ METODY 162
4.9.1. DEFINICE METODY 162
4.9.2. IMPLEMENTACE METODY 166
4.10. REGISTROVANÉ FUNKCE 172
4.10.1. DOSTUPNÉ REGISTROVANÉ FUNKCE 173
4.10.2. POUŽITÍ REGISTROVANÝCH FUNKCÍ 176
4.10.3. ZÁLOŽKA „SDÍLENÉ PARAMETRY“ 179
4.10.4. ZÁLOŽKA „KÓD PŘED“ 180
4.10.5. ZÁLOŽKA „KÓD PO“ 181
4.11. ZÁVISLOSTI MODULŮ 184
4.12. ZÁVISLOSTI TABULEK 185
4.13. ZDROJOVÝ KÓD 188
4.14. API 189
4.15. ZÁVISLOSTI JEDNOTEK 191
4.16. RYCHLÉ HLEDÁNÍ 192
4.16.1. RYCHLÉ HLEDÁNÍ NA VLASTNÍM MODULU 194
4.16.2. RYCHLÉ HLEDÁNÍ NA STANDARDNÍM MODULU 196
4.17. NÁHLED 196
4.18. EDITOR 197
4.18.1. POUŽITÍ Č. 1 - GENEROVÁNÍ HLAVIČEK V NO A NÁSLEDNÁ IMPLEMENTACE V EDITORU 199
4.18.2. POUŽITÍ Č. 2 - GENEROVÁNÍ HLAVIČEK A JEJICH IMPLEMENTACE PŘÍMO V EDITORU 201
4.18.3. LADĚNÍ SKRIPTU POMOCÍ EDITORU 203
4.19. NÁSTROJE 203

4.19.1. ZPĚTNÉ ODKAZY ... 203
4.19.2. OBNOVENÍ OBSAHU TABULKY ... 204
4.20. MERLIN ... 204
5. ROZŠIŘUJEME STANDARDNÍ DATOVÝ MODUL ... 208
5.1. POLOŽKY VLASTNĚNÉ ... 208
5.2. POLOŽKY NEVLASTNĚNÉ ... 208
5.3. FYZICKÁ POLE ... 208
5.4. POČÍTANÁ POLE ... 208
5.5. REGISTROVANÉ FUNKCE ... 208
6. SKRIPTOVÁ TŘÍDA ... 209
6.1. NÁHLED ... 209
6.2. MODUL ... 210
6.3. POLE ... 211
6.3.1. VLOŽENÝ OBJEKT ... 216
6.4. ZDĚDĚNÉ PROPERTY ... 217
6.5. AKCE ... 217
6.5.1. ZÁKLADNÍ NASTAVENÍ ... 218
6.5.2. ZÁLOŽKA „HINT“ ... 219
6.5.3. ZÁLOŽKA „EXECUTE“ ... 219
6.5.4. ZÁLOŽKA „UPDATE“ ... 220
6.6. VLASTNÍ METODY ... 220
6.7. ZÁVISLOSTI MODULŮ ... 220
6.8. ZÁVISLOSTI TABULEK ... 220
6.9. ZDROJOVÝ KÓD ... 220
6.10. API ... 220
6.11. ZÁVISLOSTI JEDNOTEK ... 220
7. ROZŠÍŘENÍ KNIHY ... 232
7.1. MODUL ... 232
7.2. POLE ... 233
7.3. APLIKACE ÚPRAV – DEPLOY ... 234
8. ZÁLOŽKA „TABULKY“ ... 237
8.1. PŘEPNUTÍ DO REŽIMU ROZŠÍŘENÍ POMOCÍ NO ... 237
8.2. ROZŠÍŘENÍ TABULEK POMOCÍ NO ... 238
8.2.1. ZÁKLADNÍ INFORMACE TABULKY ... 239
8.2.2. POLE ... 240
8.2.3. INDEXY ... 242
9. ZÁLOŽKA „TYPY“ ... 250
9.1. POUŽITÍ VÝČTOVÉHO TYPU V DATOVÉM MODULU ... 252
10. ZÁLOŽKA „PRÁVA“ ... 256
11. ZÁLOŽKA „JEDNOTKY“ ... 259
11.1. SEZNAM DOSTUPNÝCH JEDNOTEK ... 259
11.2. NOVÁ JEDNOTKA ... 260
11.2.1. ZÁVISLOSTI JEDNOTEK ... 262
11.2.2. ZÁVISLOSTI MODULŮ ... 262
11.2.3. ZÁVISLOSTI TABULEK ... 263

11.2.4. ZDROJOVÝ KÓD 264
12. ZÁLOŽKA „ZÁVISÍ NA“ 265
13. GLOBÁLNÍ FUNKCE NÁVRHÁŘE OBJEKTŮ 272
13.1. „DEPLOY“ – APLIKOVÁNÍ ÚPRAV DO K2 272
13.2. „NÁSTROJE“ 273
13.2.1. KONTROLA 273
13.2.2. API 273
13.2.3. PŘEČÍSLOVÁNÍ 273
13.2.4. SPRÁVA KNIH 274
13.2.5. LADĚNÍ 274
13.3. „BALÍČEK“ 278
13.4. „ROZŠÍŘENÍ“ 278
13.4.1. INSTALACE ROZŠÍŘENÍ 279
13.4.2. AKTIVACE / DEAKTIVACE ROZŠÍŘENÍ 283
13.5. „NASTAVENÍ“ 284
13.6. IMPORT 284
13.7. „ZAVŘÍT“ 289
14. AKTUALIZACE ROZŠÍŘENÍ PO REINSTALACI 290
14.1. KONVERZE STRUKTURY ROZŠÍŘENÍ DO NOVÉHO FORMÁTU 291
14.2. MOŽNOSTI PARAMETRU EXTENSIONS 291

Autor: Michal Kaplík
Datum vydání: 10. 02. 2023
Vydavatel: K2 atmitec s.r.o.

1. ÚVOD

Tento dokument slouží jako podpůrný materiál při zaškolování uživatele pro práci s návrhářem objektů v informačním systému K2. Uživateli doporučujeme tento materiál během školení doplňovat a rozšiřovat o vlastní poznámky.

1.1. CÍL ŠKOLENÍ

Po absolvování školení, by měl být posluchač schopen pracovat s návrhářem objektů v informačním systému K2. Čímž se rozumí konfigurace, správa, vytváření a údržba nových struktur a také rozšiřování již existujících struktur informačního systému K2. Strukturami se rozumí datové moduly, skriptové třídy a jednotky.

1.2. PŘEDPOKLADY

Předpokládá se, že posluchač je administrátor nebo programátor informačního systému K2, který rozumí jeho strukturám a principům. Výhodou je znalost K2 skriptu, kterého se využívá k implementaci případné logiky jednotlivých struktur vytvořených pomocí návrháře objektů.

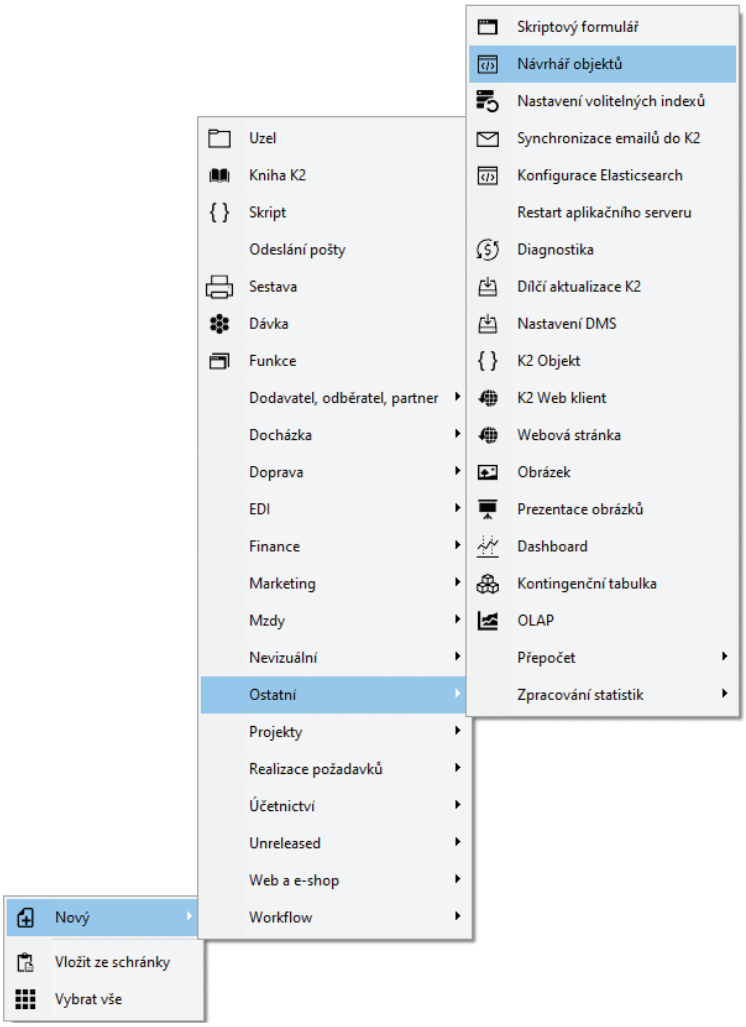
2. SEZNÁMENÍ S NÁVRHÁŘEM OBJEKTŮ

Návrhář objektů, dále jen NO, je nástroj, jehož cílem je zrychlit a usnadnit vývoj i údržbu nových součástí pro informační systém K2 s využitím nově vytvořených tříd a knihovních funkcí.

2.1. ZALOŽENÍ A NASTAVENÍ NOVÉ DEFINICE ROZŠÍŘENÍ

V informačním systému K2 může existovat libovolný počet definic rozšíření pomocí NO. Každá taková definice má jasně definován prostor pro vytváření svých struktur pomocí rozsahu číslování, prefixů v názvosloví a je kompletně uložena v samostatném definičním souboru, který je uložen ve formátu ***„xml“***.

Založení nové definice nebo otevření již existující, provedete následujícím způsobem. Do plochy K2 přidáte nového zástupce ***„Návrhář objektů“***, který je k dispozici v sekci ***„Ostatní“*** viz Obrázek 1 - Vložení návrháře objektů do plochy K2.

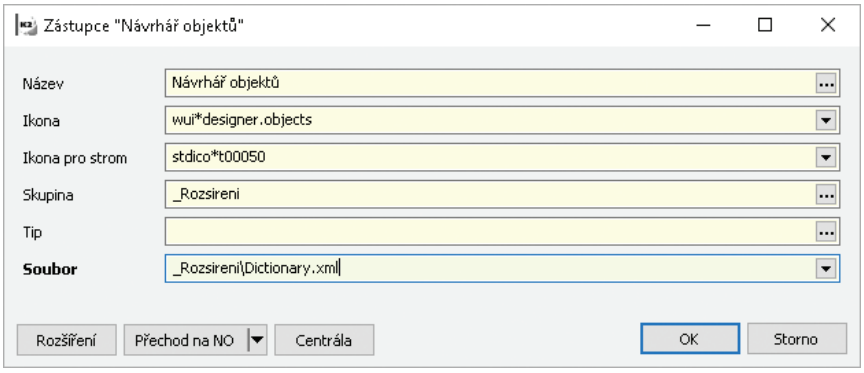


Obrázek 1 - Vložení návrháře objektů do plochy K2

Po vložení zástupce do plochy K2, se zobrazí nastavení NO, které bude poplatné tomuto zástupci. To znamená, která definice změn rozšíření se bude otevírat pod tímto tlačítkem. Ve formuláři, který je vidět na obrázku Obrázek 2 - Nastavení definice návrháře objektů, jsou k dispozici vstupní pole – ***„Název“***, ***„Ikona“***, ***„Ikona pro strom“***, ***„Skupina“*** a ***„Tip“***, která slouží pouze k popisu přidaného zástupce na ploše K2.

2.1.1. SOUBOR S DEFINICÍ ROZŠÍŘENÍ

Důležitým nastavením je pole ***„Soubor“***. Zde je uložena cesta do adresáře K2 s uloženou definicí pro návrhář objektů. Tedy ***„xml“*** soubor, který bude sloužit návrháři objektů jako vstup. Pokud nevyplníme žádný údaj, automaticky se použije cesta do uživatelského adresáře – ***„CONF\ATTACH\Uxxx\CustomModule“*** přihlášeného uživatele. V tomto umístění se vytvoří soubor s názvem ***„Dictionary.xml“***. V případě existující definice, otevřeme, pomocí tlačítka v poli ***„Soubor“***, dialog pro výběr souboru a vybereme.



Obrázek 2 - Nastavení definice návrháře objektů

Po stisknutí tlačítka OK přidáme zástupce pro návrhář objektů s Vaší definicí na plochu K2 a automaticky se otevírá nastavení určené pro Vaši definici, viz kapitola 2.2. Nastavení návrháře objektů.

Rozšíření na NO a Centrála

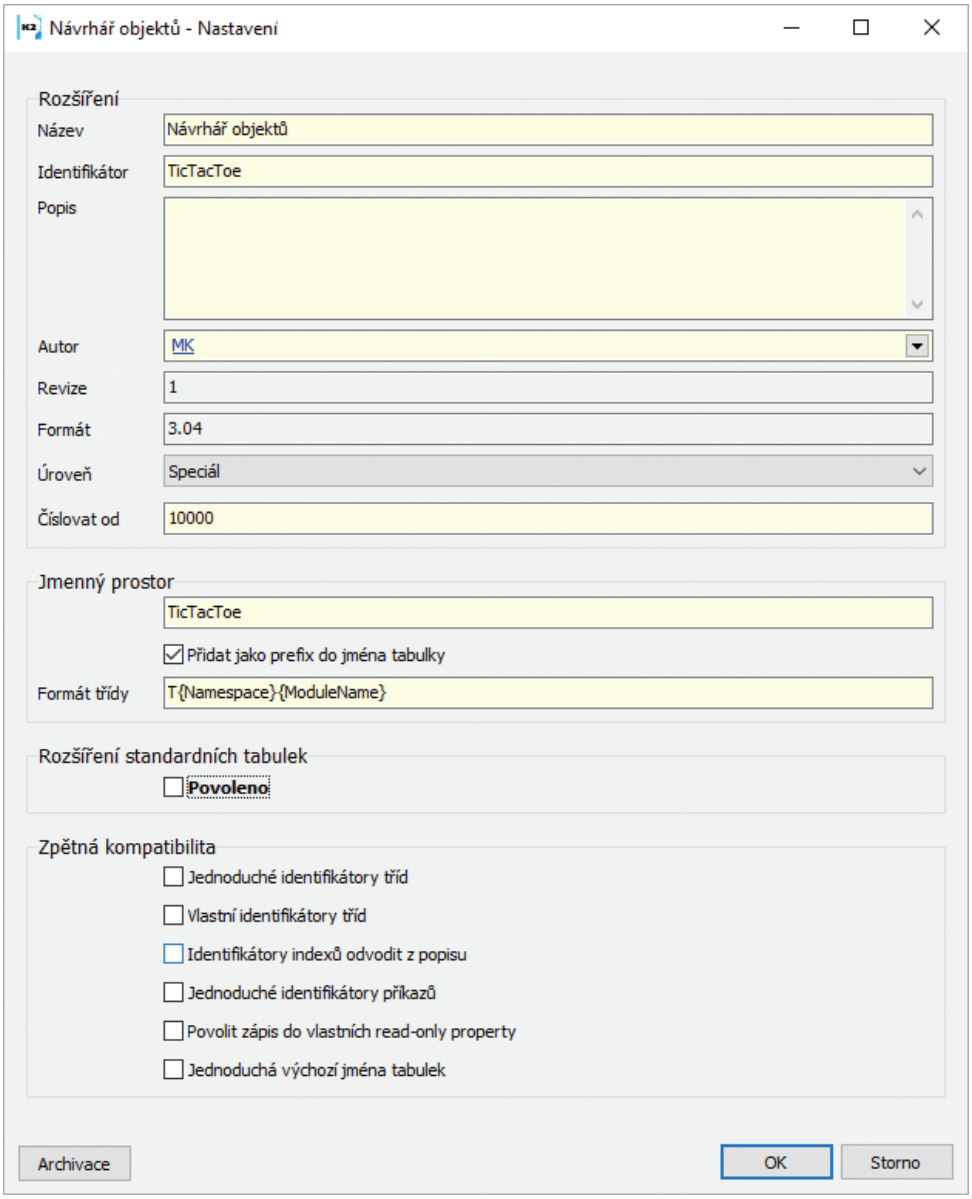
Tyto tlačítka slouží pro přechod speciálních polí z datového modulu ***„Rozšíření souborových modulů“*** přímo do NO.

TIP: Pro lepší údržbu všech rozšířených v dané K2, se doporučuje vytvořit si složku v rootu K2 např. ***„_Rozsireni“*** a do ní vkládat všechny soubory *.xml z NO (podtržítko před názvem složky je z důvodu, aby daná složka byla na začátku složek). Lépe se udržuje na jednom místě, než když jsou soubory *.xml z NO rozmístěné různě po složkách

2.2. NASTAVENÍ NÁVRHÁŘE OBJEKTŮ

Důležitou částí návrháře objektů je nastavení. Doporučujeme při každém založení nové definice, tuto sekci upravit dle potřeb. Změny lze provádět i později, ale vzhledem k objektům jako jsou univerzální formuláře, skripty, sestavy apod., které již mohou používat Vámi definované struktury, není tato operace doporučována.

Formulář pro nastavení se zobrazí automaticky po založení nové definice rozšíření návrháře objektů nebo po stisknutí tlačítka Nastavení, které můžete vidět na Obrázek 5 - Návrhář objektů. Po spuštění se zobrazí formulář Obrázek 3 - Nastavení návrháře objektů.



Obrázek 3 - Nastavení návrháře objektů

2.2.1. ZÁKLADNÍ INFORMACE O ROZŠÍŘENÍ

V sekci ***„Rozšíření“*** se nacházejí parametry ***„Název“***, ***„Identifikátor“***, apod., které slouží k základnímu nastavení, které si v následující části popíšeme.

Název
Slouží ke krátkému názvu pro definici rozšíření. Z názvu by mělo jít jednoduše poznat, k jakému účelu bude definice sloužit. Název se pak dále používá například v názvu instalačního balíčku, který si budeme popisovat dále v textu.

Identifikátor
Slouží k identifikaci definice rozšíření spíše na programové úrovni. Používá se například v názvu skriptu pro aplikační rozhraní definice rozšíření. Tento skript generuje návrhář objektů.

POZNÁMKA: Identifikátor musí být v K2 unikátní. V případě, že nebude, nepůjde projekt použít.

Popis
Slouží k detailnějšímu popisu definice rozšíření.

Autor
Zde může autor uvést své jméno, případně email.

Revize
Tuto hodnotu si návrhář objektů řídí sám. Slouží k verzování úprav, které jsou postupně do definice rozšíření aplikovány. Vždy, když je provedena akce ***„Deploy"*** nebo je vytvořen instalační balíček, navýší se tento údaj o jedničku. O operaci ***„Deploy"*** se více dozvíte v kapitole 13.1. „Deploy" – aplikování úprav do K2. O vytváření instalačních balíčků v kapitole 13.3. „Balíček".

Formát
Informace o verzi struktury rozšíření Návrháře objektů. Od verze K2 ori.04 mají struktury verzi 3.04.

Úroveň
Informace o úrovni rozšíření NO. Implicitně je nastaveno na ***„Speciál"***.

Nastavení číslování – „Číslovat od"
Důležitým nastavením je hodnota parametru ***„Číslovat od"***. Vyjadřuje počátek číslování jednotlivých struktur vytvořených v návrháři objektů. Pro každou definici by měl být nadefinován nový rozsah. Tedy, v případě, že vytváříme dvě různé množiny struktur, kdy každá má svou vlastní definici v ***„xml"*** souboru, pak u první např. nastavíme 10000 a u druhé začneme číslovat od 11000. Tento údaj je potřeba nastavit před vytvářením všech struktur v rámci jedné definice rozšíření. V případě, že tento údaj změníte, nedochází k přečíslování existujících objektů v dané definici. Každá nová struktura, pak bude číslována od nového rozsahu. Pro přečíslování všech existujících struktur dle nového rozsahu, je potřeba použít speciální funkci ***„Přečíslování"***, která je popsána dále v textu.

POZNÁMKA: Číslování je definováno v rozsahu <10000, 20000), tedy 10000 včetně a výše do 19999. Výše jsou číslovány standardní položkové datové moduly a další objekty, které jsou dodávány s instalací K2. V případě, že je překročen rozsah, návrhář objektů uživatele upozorní, v jakém rozsahu musí být číslování.

TIP: Z praxe se doporučuje číslovat různý projekty v NO vždy po 100. např. 10000, 10100, 10200. Otázkou ale někdy je, kolik objektů plánujeme pomocí NO vytvořit.

2.2.2. JMENNÝ PROSTOR

Další sekcí nastavení je definice tzv. jmenného prostoru. Tato hodnota je pak použita u všech struktur vytvořených v rámci dané definice rozšíření jako prefix názvu objektu. Např. vytvoříme nový datový modul, který nazveme ***„Test"***, jmenný prostor máme nastaven na hodnotu ***„Demo"***. Pak identifikátorem objektu ***„Test"*** bude ***„TDemoTest"***. Obecně by se dalo popsat výrazem ***"T{Jmenný prostor}{Název objektu}"***. Vzniklý identifikátor se používá všude, kde pracujeme s vytvářenými objekty, např. ve skriptu, při vkládání odkazu na plochu K2 apod.

POZNÁMKA: Pokud tento údaj nevyplníme, dosadí se výchozí hodnota ***„Custom"***, tedy všechny struktury budou začínat klíčovým slovem ***„TCustom"***.
Volbou ***„Přidat jako prefix do jména tabulky"*** nastavíte mechanismus, který v případě, že se vytváří pro nový objekt i databázová tabulka, pak i tato tabulka bude mít v názvu tento jmenný prostor jako prefix.

Formát třídy
Výše zmíněnou logiku, která definuje názvy tříd modulů vytvořených v Návrháři objektů lze ovlivnit nastavením ***„Formát třídy“***. Ve výchozím nastavení je vyplněno ***„T{Namespace}{ModuleName}“*** což již bylo popsáno. Pokud bychom chtěli použít jinou konvenci, můžeme zde upravit.

2.2.3. ZPĚTNÁ KOMPATIBILITA

Slouží k zachování zpětné kompatibility pro struktury, které byly vytvořeny ve starších verzích návrháře objektů. V případě nové definice není potřeba tyto volby měnit.

Jednoduché identifikátory tříd, Vlastní identifikátory tříd
Ve starších verzích návrháře objektů nebyla kontrola na názvy položkových modulů. V případě, že vzniklo více položkových modulů se stejným identifikátorem, docházelo k problémům. V novější verzi návrhář objektů přidává do identifikátoru položkových modulů název nadřízeného modulu, aby byla zajištěna unikátnost identifikátoru. V případě, že přecházíte ze staré definice rozšíření návrháře objektů na novější, návrhář Vám tyto názvy změní, což způsobí nefunkčnost formulářů, skriptů, sestav, obecně speciálních úprav, které moduly používají. Pak je potřeba nastavit volby pro zpětnou kompatibilitu.

Identifikátory indexů odvodit z popisu
Konstanty pro indexy se ve starších verzích odvozovaly z jejich popisu. V novějších verzích se konstanta tvoří již z identifikátoru. V případě zachování zpětné kompatibility je možné zapnout původní chování, které navíc ke konstantě z identifikátoru vygeneruje i konstantu z popisu. Tedy každý index je pak identifikovatelný dvěma konstantami.

Jednoduché identifikátory příkazů
Ve starších verzích K2 došlo ke změně logiky tvorby identifikátorů pro příkazy. Aby bylo možné překlenout tento stav, kdy přestanou fungovat vlastní příkazy na formulářích, je možné využít této volby.

Úplné identifikátory pro příkazy se tvoří ve tvaru ***„{identifikátor_příkazu}_T{NameSpace}{identifikátor datového modulu}“***. V případě potřeby jednoduché varianty, je možné tento parametr zapnout. Úplný identifikátor pak bude v následném tvaru ***„{identifikátor_příkazu}“***.

POZNÁMKA: Identifikátory se využívají například při vkládání příkazu do formuláře. V případě, že následně změníme nastavení tvorby identifikátoru, příkazy na formulářích přestanou fungovat, protože je nebude možné identifikovat.

POZNÁMKA: Pokud je to možné, je doporučeno opravit starou variantu identifikátorů v místech kde jsou použity. Nový způsob je zaveden z důvodu silnějších kontrol na duplicity a s tím spojené případné přeslechy, chyby apod.

Povolit zápis do vlastních read-only tabulek
Daná property slouží k nastavení zpětné kompatibility pro vytvoření nové property. Používá se pouze dočasně při přechodu na novější verzi. Může způsobit memory leaky a přepis paměti.

Jednoduchá výchozí jména tabulek
Pokud je property zatržená, tak se nové tabulky budou pojmenovávat **jmenný prostor_** (pokud je zatržena volba „Přidat jako prefix do jména tabulky“) **+ identifikátor modulu**. Pokud daná property není zatržená, vytváří se tabulka následujícím vzorcem, **jmenný prostor_** (pokud je zatržena volba ***„Přidat jako prefix do jména tabulky“***) **+ všechny identifikátory nadřízených modulů**.

POZNÁMKA: Daná property se týká především pojmenování položkových modulů. Jako příklad uvedu, mám hlavičkový modul pojmenovaný MyDM, k němu vytvořím položkový modul MyChild. Takto bude vypadat rozdíl mezí tím, zda mám volbu „Jednoduchá výchozí jména tabulek" či nikoliv.

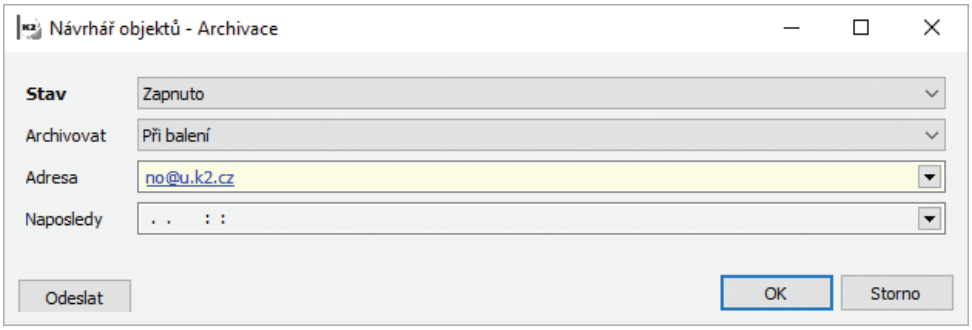
1. Zatržená volba „Jednoduchá výchozí jména tabulek" + jmenný prostor jako prefix tabulek (NO)
   a. Tabulka hlavičky: NO_MyDM
   b. Tabulka položek: NO_MyChild

2. Nezatržená volba „Jednoduchá výchozí jména tabulek" + jmenný prostor jako prefix tabulek (NO)
   a. Tabulka hlavičky: NO_MyDM
   b. Tabulka položek: NO_MyDMMyChild

POZNÁMKA: Pokud je již nainstalovaný projekt se zatrženou fajfkou z dřívějších verze (před 2020.08.106238 nebo 2021.04.106238), musí se případné přejmenování tabulek provést ručně. Odtržení property nebude reagovat na přejmenování tabulek.

2.2.4. ARCHIVACE

Tato sekce slouží k odesílání definice rozšíření na definovaný mail, a to za účelem archivace nebo také zjednodušení podpory v případě problémů s konkrétním rozšířením. Funkce archivace je ve výchozím nastavení vypnutý. Pokud nastavíme ***„Stav"*** na hodnotu ***„Zapnuto"***, zpřístupní se nám nastavení, viz Obrázek 4 - Nastavení archivace definice rozšíření.



Obrázek 4 - Nastavení archivace definice rozšíření

Archivovat
Tímto nastavením ovlivníme, kdy bude archivace probíhat. Možnosti jsou tři.

Při balení – vždy, když se spustí vytváření balíčku pomocí funkce 13.3. „Balíček"

Při nasazení – vždy, když se provede akce 13.1. „Deploy" – aplikování úprav do K2

Ručně – pouze při ruční akci pomocí tlačítka Odeslat, které je dostupné na Obrázek 4 - Nastavení archivace definice rozšíření

Adresa
Nastavení pro cílovou emailovou adresu.

Naposledy
Informace, kdy byla definice rozšíření naposled archivována.

Odeslat
Tlačítko k vynucení odeslání definice rozšíření.

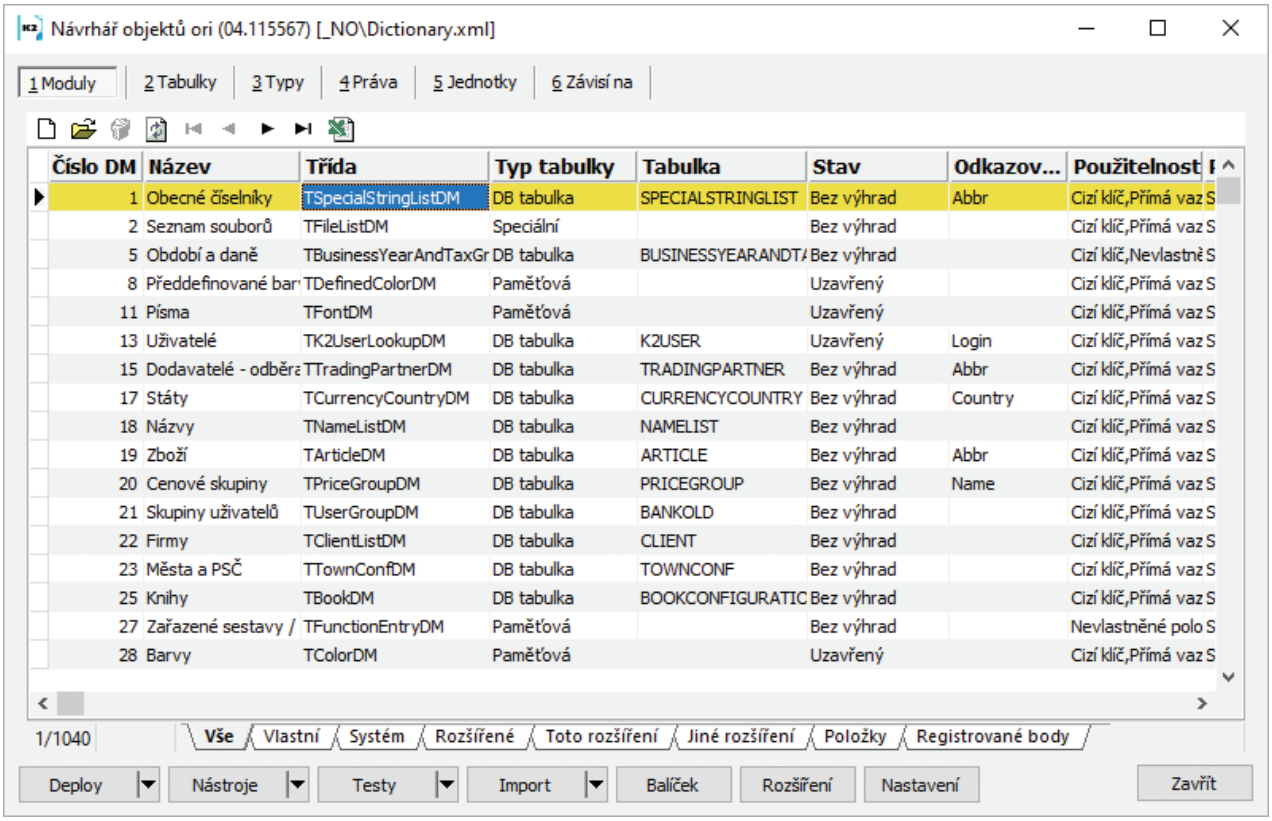
POZNÁMKA: Mailová adresa je přednastavená na hodnotu *no@u.k2.cz*. Tato adresa slouží ke sběru definic rozšíření, které se využívají k testování úprav návrháře objektů. Samozřejmě odeslaná definice rozšíření se nikde dále nešíří a je použita pouze k účelům testování.

2.3. SPUŠTĚNÍ NÁVRHÁŘE OBJEKTŮ

Po spuštění vytvořeného zástupce se zobrazí návrhář objektů. V záhlaví formuláře můžete vidět jméno a cestu k souboru k definici, pro kterou je návrhář spuštěn viz Obrázek 5 - Návrhář objektů. V následující části si popíšeme základní zobrazení formuláře pro návrhář objektů, kdy jednotlivé funkce a jejich použití bude detailně popsáno v samostatných kapitolách.

2.3.1. POPIS ZÁLOŽEK ZÁKLADNÍHO FORMULÁŘE

Na úvodní obrazovce návrháře objektů je vidět, že formulář je rozdělen na pět hlavních záložek. První záložka *„1 Moduly“* obsahuje definované struktury aktuálního rozšíření. Druhá záložka *„2 Tabulky“* obsahuje tabulky, které lze rozšířit o speciální EX_ pole a indexy. Třetí záložka *„3 Typy“* obsahuje námi definované výčtové typy. Čtvrtá záložka *„4 Práva“* obsahuje vlastní vytvořené práva. Pátá záložka *„5 Jednotky“* obsahuje seznam a správu dostupných skriptových jednotek. Poslední záložka *„6 Závisí na“* obsahuje seznam a správu dostupných balíčků definic rozšíření, které jsou připojené k aktuálně otevřené definici rozšíření. V textu se postupně budeme věnovat obsahu jednotlivých záložek.



Obrázek 5 - Návrhář objektů

3. ZÁLOŽKA „MODULY“

V první kapitole popisující hlavní části návrháře objektů, se budeme zabývat sekcí ***„Moduly“***. Ve spodní části formuláře najdeme záložky, kde ve výchozím zobrazení existuje záložka ***„Vše“***, kde jsou všechny datové moduly, které jsou dodávány instalací K2 a navíc datové moduly definované všemi možnými definicemi rozšíření návrhářů objektů.

POZNÁMKA: Struktury vytvořené v definici, pro kterou je aktuálně otevřen návrhář objektů, jsou pro přehlednost zobrazeny v seznamu jako první. V případě, že existují další definice rozšíření návrháře objektů, jsou struktury z těchto jiných definic, zobrazeny dle setřídění, ve výchozím nastavení, dle jejich identifikačního čísla.

Další záložky pak definují podmnožiny struktur ze záložky ***„Vše“***.

Záložka *„Vlastní“*
Seznam všech struktur, které jsou nově definovány v aktuální definici návrháře objektů.

Záložka *„Systém“*
Zde jsou všechny struktury, které jsou dodávány instalací K2.

Záložka *„Rozšířené“*
Seznam standardních struktur, které jsou dodávány instalací K2 a v aktuální definici návrháře objektů jsou modifikovány, např. rozšířeny o položkové moduly.

Záložka *„Toto rozšíření“*
Seznam všech struktur, jak vlastních tak standardních, které jsou rozšířeny v aktuální definici návrháře objektů.

Záložka *„Jiná rozšíření“*
Seznam všech struktur, jak vlastních tak standardních, které nejsou rozšířeny v aktuální definici návrháře objektů.

Záložka *„Položky“*
Seznam struktur, které jsou dodávány instalací K2, a jedná se o položkové moduly, které jsou programátorem IS K2 označeny speciálním příznakem, který umožňuje tyto položky rozšiřovat o nevlastněné položky.

Záložka *„Registrované body“*
Seznam struktur, ve kterých jsou implementovány registrované funkce pomocí Návrháře objektů.

3.1. POPIS SLOUPCŮ SEZNAMU ZÁKLADNÍHO FORMULÁŘE

V seznamu na Obrázek 5 - Návrhář objektů můžeme vidět několik sloupců, které definují nejdůležitější informace o jednotlivých strukturách dostupných v návrháři objektů. Následující tabulka tyto sloupce popisuje.

NÁZEV SLOUPCE	POPIS
Číslo DM	Identifikátor vytvořené struktury
Název	Název vytvořené struktury
Třída	Textový identifikátor struktury
Typ tabulky	Určuje typ tabulky
Tabulka	Název databázové tabulky
Stav	Bez výhrad, uzavřený, chyba, ...
Odkazové pole	Pole, určené jako výstupní pro odkazovou dynamickou vazbu
Použitelnost	Cizí klíč, přímá vazba
Původ	Určuje původ – tovární, jiná definice NO
Jednotka	Delphi jednotka
Autor	Autor
Dostupnost na AS	Určuje dostupnost na aplikačním serveru
Složitost	Přibližný počet operací k definování DM
Formulář(e)	Registrace DM pro formulář
Instance	Počet vytvořených instanci DM pro NO
Vytvoření	Čas vytvoření instance pro DM
Prodlení	Čas načtení metadat do NO

Některé výše popsané sloupce se opakují i v popisu formulářů pro zadávání nových objektů, kde budou popsány detailněji. Ty, které se nacházejí pouze v seznamu, popíšeme v následující části.

Stav
Určuje, v jakém stavu se datový modul nachází a zároveň i možnosti jeho dalšího rozšiřování. V případě, že nebyl detekován žádný problém, a zároveň je možné datový modul rozšířit, např. o další položky, vlastněné či nevlastněné, je ve stavu ***„Bez výhrad"***.

V případě, že je vše v pořádku, ale datový modul není možné rozšířit, je ve stavu ***„Uzavřený"***. Ve většině případů se jedná o paměťové datové moduly a položkové datové moduly, které nejsou označeny k dalšímu rozšiřování.

Dalším stavem pak je ***„Problém"*** a ***„Chyba"***. Oba tyto stavy značí nějaký problém s datovým modulem. V případě chyby se pravděpodobně nepodařilo vytvořit instanci datového modulu. V případě problému může jít o nekorektní stav definice klíčů, polí apod. Důležité je, pokud se vyskytne jakýkoliv problém v definici rozšíření objektů, tedy v uživatelských modulech, uživatel je na každý problém upozorněn při ukládání změn, případně při akci ***„Deploy"***.

Odkazové pole
Určuje, které pole bude použito jako výstupní při vytvoření odkazové dynamické vazby. Více se k tomuto tématu dozvíte v kapitole 4.2.2.2. Vazba – Odkazové pole.

Použitelnost
Tato informace říká, jakým způsobem je možné použít datový modul. V případě, že je uvedeno ***„Cizí klíč"***, můžeme vytvořit vazbu přes primární klíč do tohoto datového modulu z jiného modulu. V případě, že nabývá hodnoty ***„Přímá vazba"***, je možné vytvořit přímou vazbu do tohoto datového modulu z jiného modulu. Více o těchto typech vazeb bude popsáno v kapitole 4.2. Pole.

Jednotka
Delphi jednotka, ve které je třída deklarována. Určeno pro interní použití.

Složitost
Přibližný počet operací, které je potřeba udělat pro nadefinování modulu. Každá operace (nastavení podstatných property, přidání pole, indexu, registrované funkce, akce/commandu, metody) je ohodnocena nějakou složitostí a ty se sečtou. Určeno pro interní použití.

Formulář
Jedná se o historickou informaci, kdy existovala povinnost registrovat k DM nějaký formulář. Určeno pro interní použití.

Instance
Počet instancí datových modulů, které byly vytvořeny během vytvoření instance datového modulu během metadat pro návrhář objektů. Určeno pro interní použití.

Vytvoření
Kolik ms zabere vytvoření instance datového modulu. Určeno pro interní použití.

Prodlení
Kolik ms zabere načtení metadat do návrháře objektů. Určeno pro interní použití.

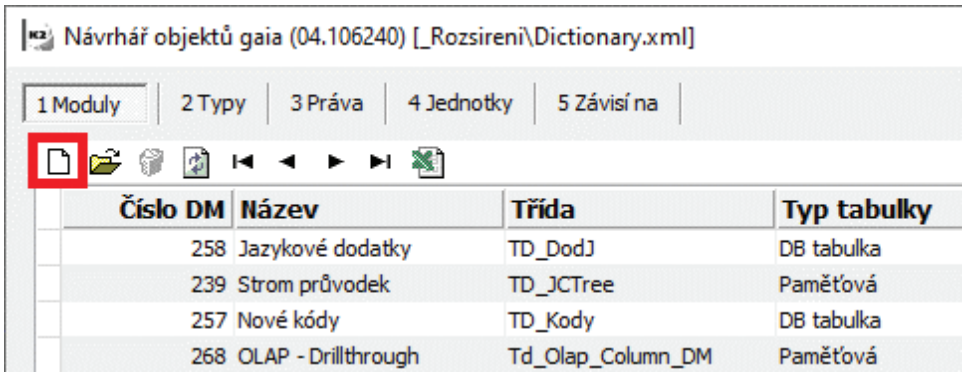
TIP: V seznamech datových modulů, je možné prostým psaním textu odfiltrovat záznamy, které neobsahují psané (hledané) slovo. Můžeme si takto usnadnit práci při prohledávání velkého množství záznamů.

Již máme popsáno nastavení, spuštění a vzhled základního formuláře pro návrhář objektů. V následujících kapitolách se budeme dále věnovat tvorbě nových a úpravě stávajících struktur, které jsou dostupné v návrháři objektů. Také funkcím, které návrhář objektů nabízí.

3.2. MOŽNOSTI ROZŠÍŘENÍ

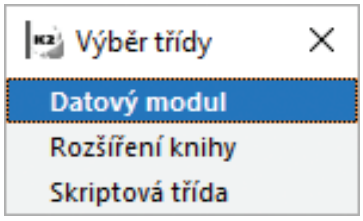
Pomocí návrháře objektů můžeme Informační systém K2 rozšířit o několik typů nových struktur. V této kapitole si popíšeme, které struktury to jsou. Následné kapitoly pak budou popisovat již samotnou práci na rozšiřování jednotlivých typů.

Nový objekt vložíme do návrháře objektů tak, že stiskneme tlačítko Nový záznam nebo stiskneme klávesu Insert viz Obrázek 6 - Vytvoření nové struktury.



Obrázek 6 - Vytvoření nové struktury

Zobrazí se formulář, viz Obrázek 7 - Výběr nového typu objektu, pomocí kterého si vyberete, které objekty potřebujete vytvářet. Jak je vidět na obrázku, k dispozici máme ***„Datový modul“***, ***„Rozšíření knihy“*** a ***„Skriptovou třídu“***. V následujících podkapitolách si popíšeme jednotlivé typy včetně jejich použití a vytvoření v návrháři objektů.

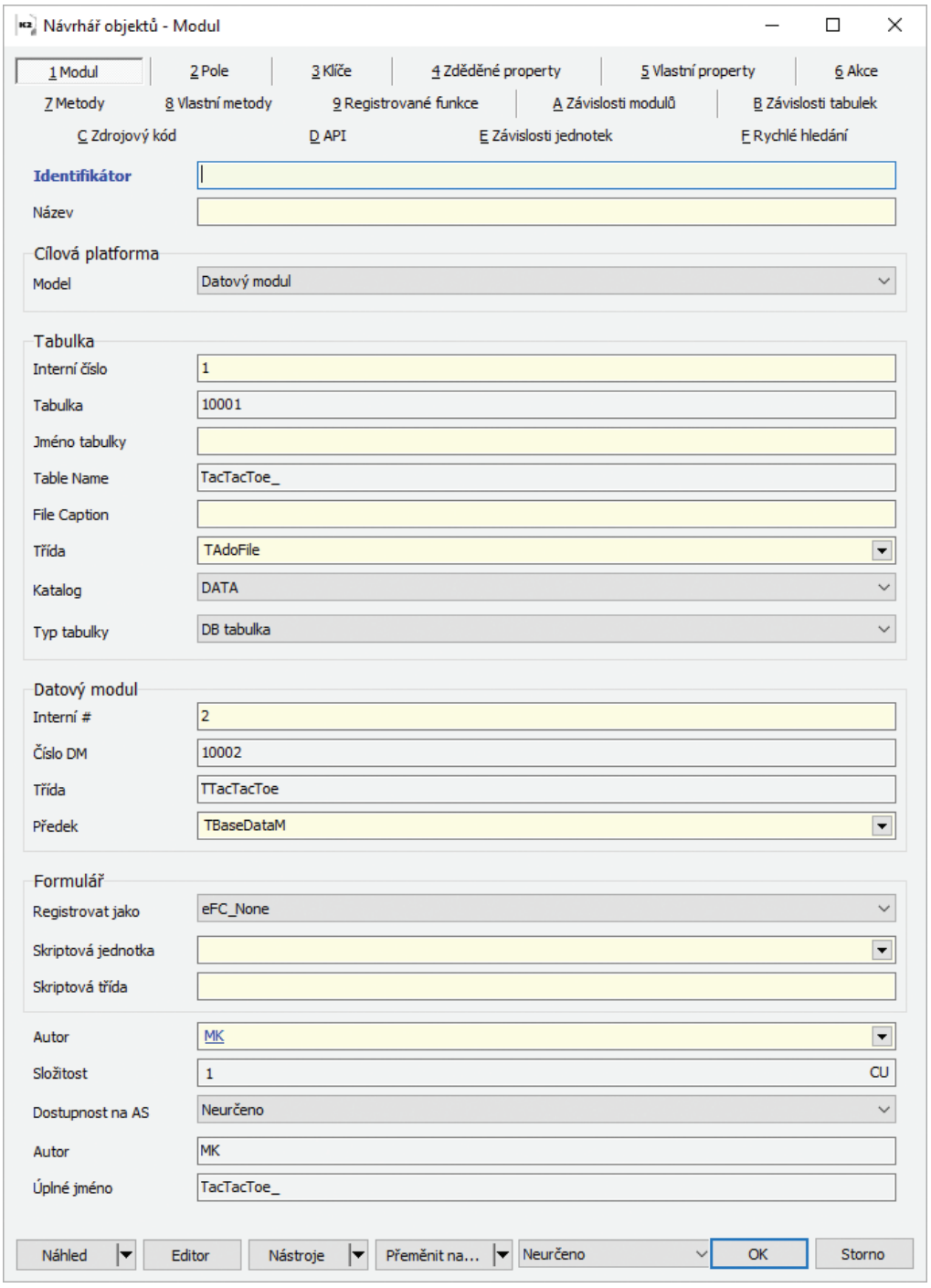


Obrázek 7 - Výběr nového typu objektu

POZNÁMKA: V případě potřeby editovat existující objekty, a to jak datové moduly, tak pole, klíče apod., platí, že do editace otevřete záznam dvojitým kliknutím myší na záznam, stiskem tlačítka Upravit záznam, zmáčknutím klávesy Enter nebo klávesovou zkratkou F5.

4. VYTVÁŘÍME DATOVÝ MODUL

Jak je uvedeno v kapitole výše, v záložce ***„Modul“*** stiskneme tlačítko Nový a z nabídky vybereme variantu ***„Datový modul“***, viz Obrázek 7 - Výběr nového typu objektu. Zobrazí se formulář, viz Obrázek 8 - Nový datový modul. Formulář je rozdělen do několika záložek, které si popíšeme.



Obrázek 8 - Nový datový modul

4.1. MODUL

První záložkou je ***„Modul“***. Jedná se o základní informace a nastavení o vytvářeném datovém modulu. V následujícím textu si popíšeme jednotlivé části, viz Obrázek 8 - Nový datový modul.

4.1.1. ZÁKLADNÍ NASTAVENÍ

Identifikátor
Slouží k jednoznačné identifikaci nového datového modulu. K názvu, který uvedete, se jako prefix přidá ***„T“*** + jmenný prostor. Tedy v našem obrázku je jmenný prostor ***„TicTacToe“***, nový datový modul nazveme ***„DemoModul“***, pak celý identifikátor našeho modulu bude ***„TTicTacToeDemoModul“***. Tato informace je dostupná v dolní části tohoto formuláře, v sekci ***„Datový modul“***, pole ***„Třída“***.

POZNÁMKA: Celý identifikátor, který je popsán v předchozím odstavci, se pak používá vždy, když s daným modulem pracujeme. Např. v designéru univerzálních formulářů, ve skriptu, v zástupcích na ploše K2 apod.

Název
Slouží k definici názvu nového datového modulu. Text, který uvedeme, se pak zobrazuje jako textová (uživatelsky přívětivá) varianta. Zobrazuje se pak např. v záhlaví otevřeného datového modulu, v seznamu datových modulů apod.

Autor
Do tohoto pole se uvádí autor datového modulu. Není povinná informace.

Složitost
Udává počet kroků, které musí návrhář provést, než sestaví datový modul v případě jeho použití.

Dostupnost na AS
Nabývá hodnot ***„Neurčeno“***, ***„Nedostupné“***, ***„Pouze pro čtení“*** nebo ***„Pro čtení i zápis“***. Dle jednotlivých variant vyjadřuje, zda je daná struktura dostupná i pomocí aplikačního serveru, případně, zda je dostupná pouze pro čtení nebo i pro zápis.

Úplné jméno
V tomto poli je uvedeno úplné jméno tabulky.

4.1.2. CÍLOVÁ PLATFORMA

Model
Určuje, jaký typ nového objektu právě přidáváme. Protože jsme zvolili ve vstupním formuláři, že potřebujeme vytvořit nový ***„Datový modul“***, je toto pole přednastaveno na tuto hodnotu. Ve formuláři není možné provést změnu na jiný typ.

4.1.3. TABULKA

Interní číslo
Je číslo identifikující tabulku daného datového modulu v rámci aktuální definice rozšíření v návrháři objektů. Hodnota je předvyplněná automaticky dle množství předchozích objektů. Doporučujeme číslo neměnit a nechat na návrháři objektů, jaké číslo bude novému objektu přiděleno.

Tabulka
Číselný identifikátor tabulky, který je unikátní napříč celou K2. Číslo je složeno z definovaného rozsahu v nastavení aktuální definice rozšíření návrháře objektů a z identifikátoru popsaného v předchozím bodě. Tento údaj se vyplňuje automaticky a přímo v tomto poli ho nelze měnit.

POZNÁMKA: Záměrně není uváděno, že se jedná o databázovou tabulku, protože v této definici je možné ***„tabulku“*** nastavit jako databázový pohled – ***„view“*** nebo se může jednat o paměťovou tabulku, která fyzicky není v databázi uložena.

Jméno tabulky
Určuje název tabulky pro nový datový modul. Název musí být unikátní napříč všemi definicemi. Doporučujeme tedy používat k názvu tabulky i prefix, který je možné zapnout v nastavení dané definice rozšíření návrháře objektů viz kapitola 2.2. Nastavení návrháře objektů.

Úplné jméno (Table Name)
Ukazuje úplné jméno tabulky, tedy v závislosti na nastavení, obsahuje jmenný prostor dané definice + „_" + jméno tabulky definované v předchozím bodě, nebo pouze název bez prefixu. Např. ***„TicTacToe_DemoModul"*** nebo pouze ***„DemoModul"*** v závislosti na nastavení.

File Caption
Jedná se o krátký popis tabulky. Na funkci tabulky nemá vliv. Slouží pouze k lepšímu pochopení použití tabulky, například při prohlížení tabulek databáze přes nástroj ***„Správa databází"*** v K2.

Třída
Definuje, o jaký typ tabulky se jedná. Standardně může nabývat hodnot ***„TAdoFile"*** nebo ***„TMemFile"***. Kde ***„TAdoFile"*** definuje standardní databázovou tabulku nebo databázový pohled, který se při akci ***„Deploy"*** vytvoří v databázích aktuální K2. Typ ***„TMemFile"*** určuje, že se jedná o paměťový datový modul, tedy neexistuje fyzická databázová tabulka.

Katalog
Jedná se o nastavení, které definuje, do jaké databáze se nová tabulka vytvoří. Tato vlastnost má smysl pouze pro tabulky s definovanou třídou ***„TAdoFile"***. Hodnota ***„CONF"*** definuje vytvoření tabulky v databázi, která je shodná s katalogem v k2.ini souboru a je společná pro všechny mandanty dané K2. Hodnota ***„DATA"*** definuje vytvoření tabulky v databázích jednotlivých mandantů, Tedy ve všech mandantech se při akci ***„Deploy"*** vytvoří databázová tabulka, která je zde definována.

Typ tabulky
Zpřesňuje typ tabulky. Tedy jak už bylo zmíněno, zda se jedná o databázovou tabulku, pohled nebo jde o paměťové úložiště, které v databázi fyzicky neexistuje. Pro databázovou tabulku použijte hodnotu ***„DB tabulka"***, pro databázový pohled použijte ***„DB pohled"*** a pro paměťovou tabulku pak ***„Paměťová tabulka"***.

4.1.4. DATOVÝ MODUL

Interní #
Stejně jako u tabulky, definované číslo identifikuje datový modul v rámci aktuální definice rozšíření v návrháři objektů. Hodnota je předvyplněná automaticky dle množství předchozích objektů. Doporučujeme číslo neměnit a nechat na návrháři objektů, jaké číslo bude novému objektu přiděleno.

Číslo DM
Číselný identifikátor datového modulu, který je unikátní napříč celou K2. Číslo je složeno z definovaného rozsahu v nastavení aktuální definice rozšíření návrháře objektů a z identifikátoru popsaného v předchozím bodě. Tento údaj se vyplňuje automaticky a přímo v tomto poli ho nelze měnit.

Třída
Ukazuje úplný textový identifikátor datového modulu, tedy v závislosti na nastavení, obsahuje jmenný prostor dané definice + „_" + jméno definované v prvním poli tohoto formuláře – ***„Identifikátor"***. Např. ***„TTicTacToeDemoModul"*** nebo pouze ***„DemoModul"*** v závislosti na nastavení. V sekci nastavení mění toto chování část ***„Zpětná kompatibilita"*** popsaná v kapitole 2.2. Nastavení návrháře objektů.

Předek
Určuje třídu, ze které bude datový modul odvozen. Výchozí hodnotou je ***„TBaseDataM“***. Dále jsou podporovány hodnoty ***„TOneRecordDM“***, ***„TTypeDataM“***, ***„TChildDataM“***, ***„TCustomDataM“***, ***„TCustomListDataM“***, ***„TCustomDocumentDataM“*** a ***„TMTBase“***. Každý z předků má různé chování a vyžaduje jejich nastavení na záložce ***"Zděděné property"***. Více si k vlastnostem předků v kapitole 4.5. Zděděné property.

4.1.5. FORMULÁŘ

Tato sekce je definována čistě pro tvorbu standardních šedých skriptových formulářů. Standardní šedé formuláře se nedoporučuje již používat. V budoucnu se plánuje jejich pozastavení.

Registrovat jako
Tato vlastnost určuje konstantu, pod kterou bude zaregistrován skriptový formulář. Výchozí hodnota je ***„eFC_None“***, kterou doporučujeme ponechat.

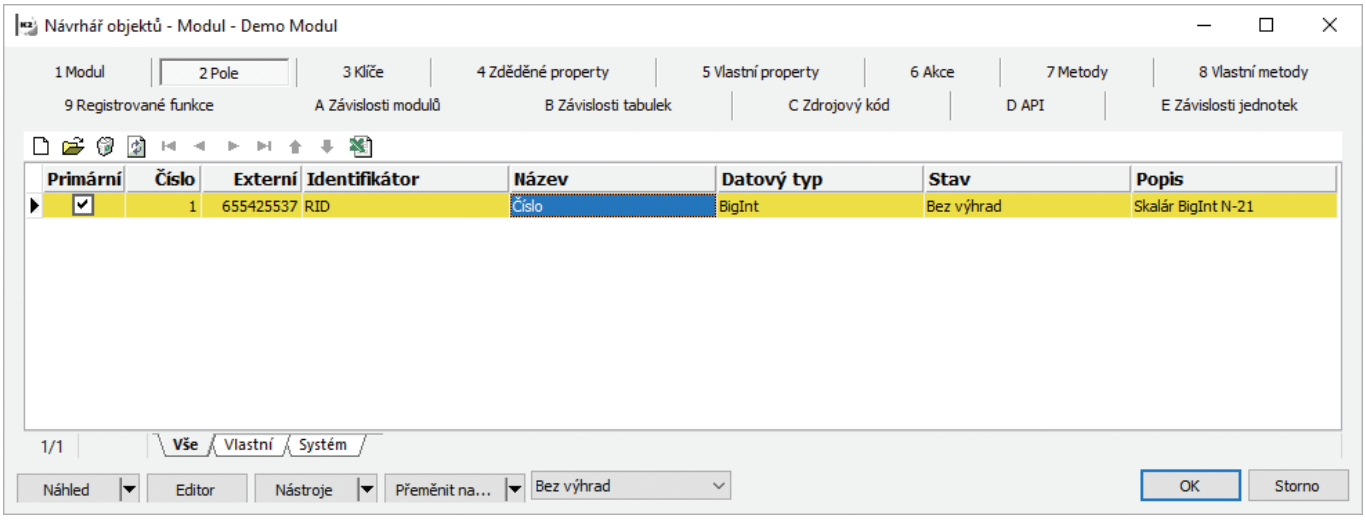
Skriptová jednotka
Zde definujeme skriptovou jednotku, soubor s příponou ***„.pas“***, ve kterém je ***„nakreslený“*** formulář. Tedy obsahuje skriptovou třídu, která popisuje daný formulář. Skriptová třída je pak definována následující vlastností.

Skriptová třída
Určuje třídu ve skriptové jednotce, definované v předchozí vlastnosti, která popisuje skriptový formulář.

4.2. POLE

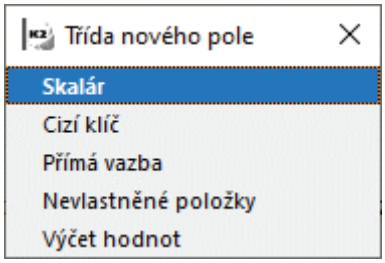
Další záložkou je ***„Pole“***. Tato sekce slouží k definici všech polí (sloupců) datového modulu. Tedy zde je možné do datového modulu vložit fyzická skalární pole, počítaná, pole odkazující do jiných modulů (vazby) a pole, která definují vlastněné i nevlastněné položky.

Formulář obsahuje seznam polí a lištu tlačítek s funkcemi pro práci se seznamem. Nové pole vložíme do seznamu pomocí ikony prázdného listu v liště tlačítek (v toolbaru) nebo pomocí stisku tlačítka Insert viz Obrázek 9 - Seznam polí datového modulu.

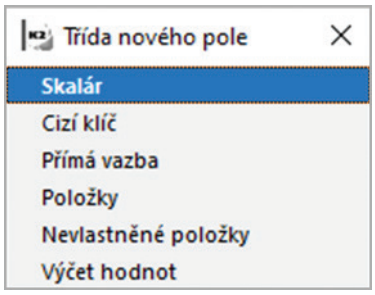


Obrázek 9 - Seznam polí datového modulu

Po stisku tlačítka nebo klávesy **Insert** se zobrazí formulář, který je vidět na Obrázek 10 - Výběr typu nového pole v datovém modulu nebo na Obrázek 11 - Výběr typu nového pole v datovém modulu 2 v závislosti na tom, zda pro daný datový modul existuje pole definující primární klíč či nikoli. V případě, že neexistuje primární klíč, zobrazí se první varianta, kde je k dispozici pět možností – ***„Skalár“***, ***„Cizí klíč“***, ***„Přímá vazba“***, ***„Nevlastněné položky“*** a ***„Výčet hodnot“***. V případě existujícího primárního k-líče se navíc zobrazí možnost vložit ***„Položky“***. Je to z toho důvodu, že položková pole vyžadují primární klíč k provázání. V následující části si popíšeme jednotlivé typy polí.

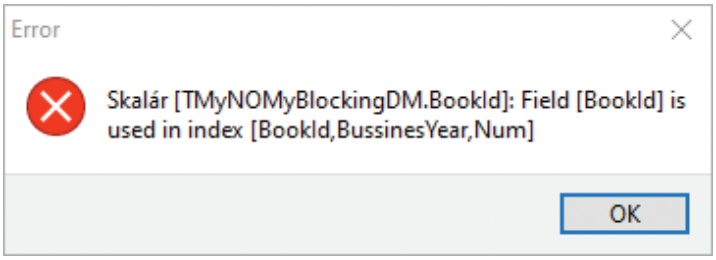


Obrázek 10 - Výběr typu nového pole v datovém modulu



Obrázek 11 - Výběr typu nového pole v datovém modulu 2

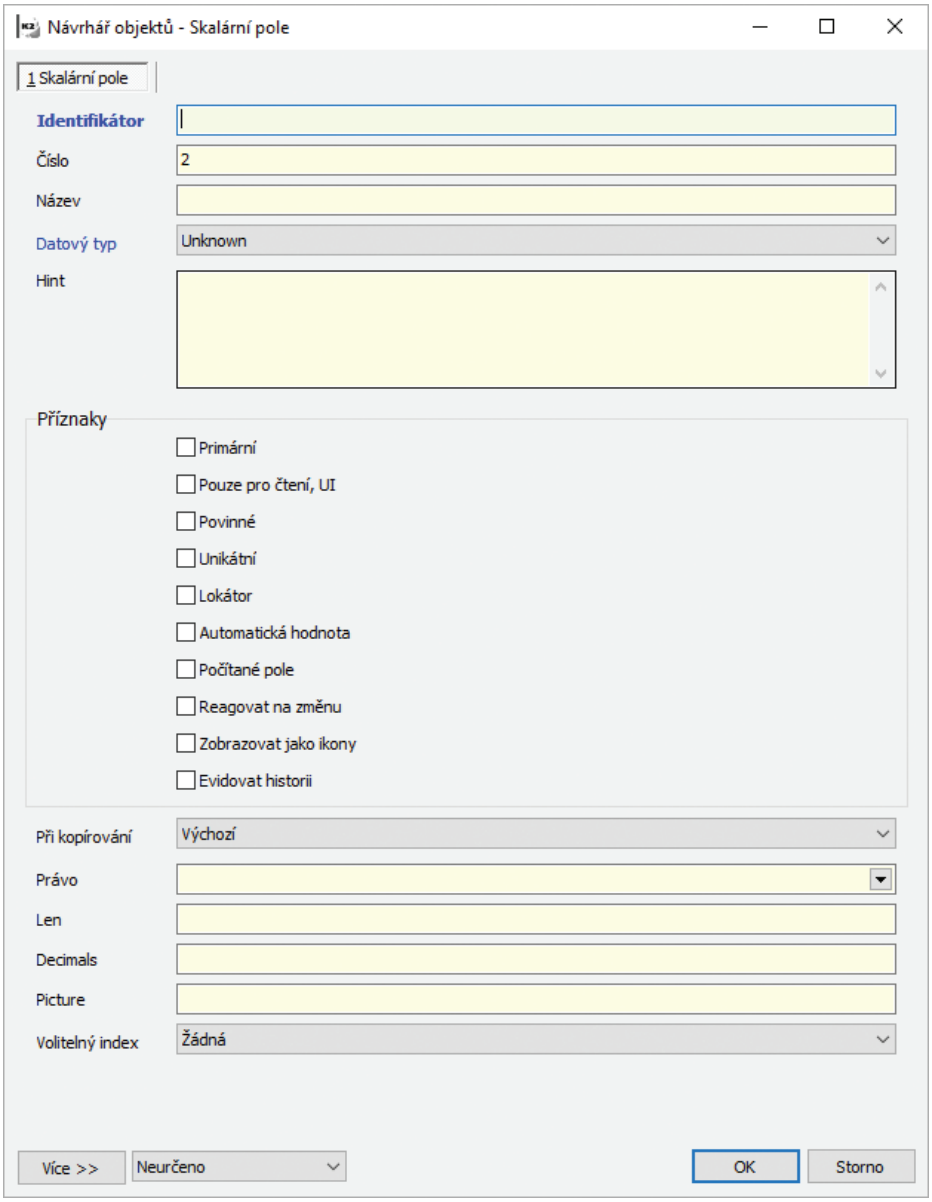
Pokud na vytvořené pole existuje odkaz/reference v jiném objektu (indexy, zděděné property atd.) a uživatel se jej pokusí smazat, tak jej NO zastaví s upozorněním, že je na dané pole někde použitý odkaz (viz Obrázek 12 - Hlášení při smazání pole o existenci odkazu) – zde hlášení říká, že pole BookId je použito v indexu.



Obrázek 12 - Hlášení při smazání pole o existenci odkazu

4.2.1. SKALÁR

Po výběru typu pole „Skalár" se zobrazí formulář Obrázek 13 - Nové pole datového modulu typu "Skalár". Jednotlivé části si nyní detailně popíšeme.



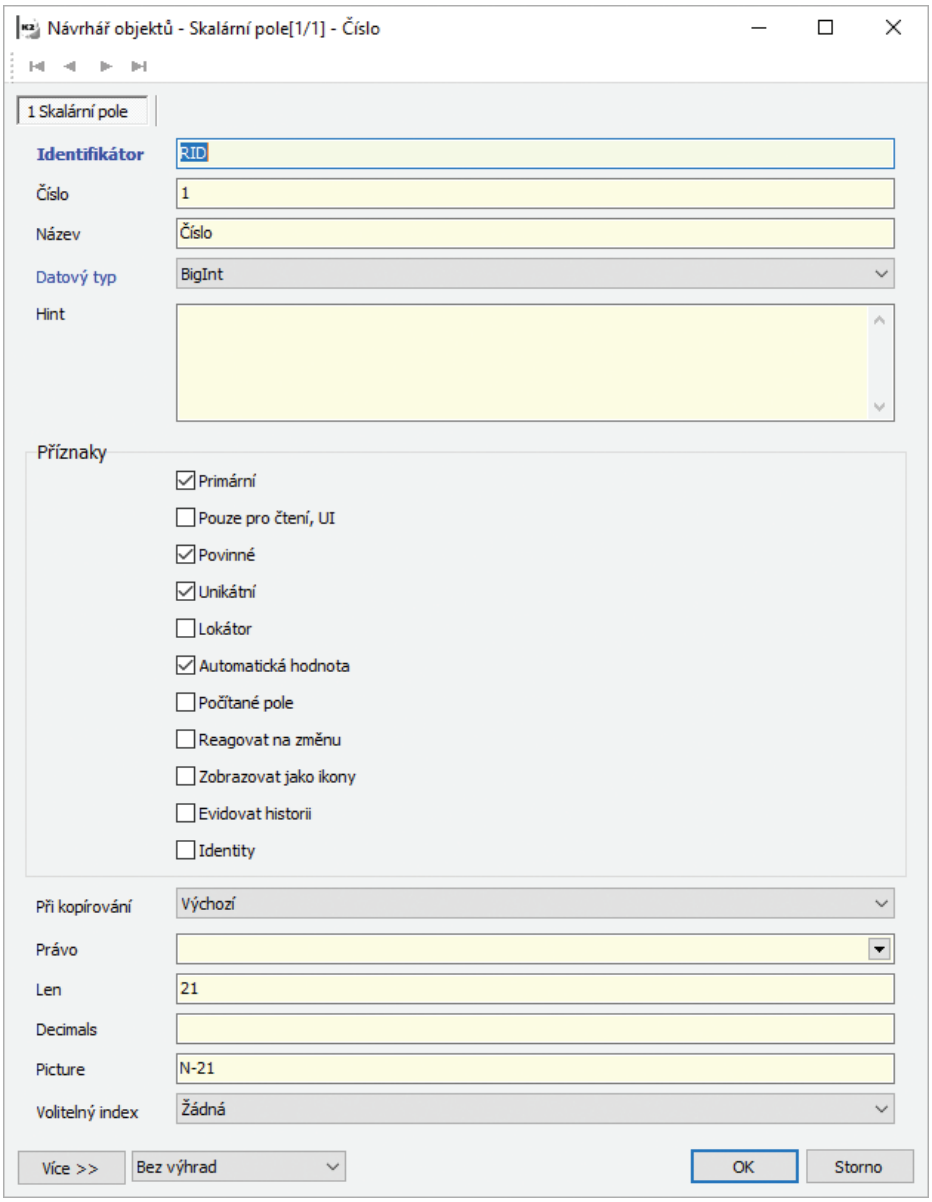
Obrázek 13 - Nové pole datového modulu typu "Skalár"

4.2.1.1. Základní nastavení

Identifikátor
Slouží k jednoznačné identifikaci pole v rámci datového modulu. V případě, že se jedná o datový modul, pro který existuje databázová tabulka, pak se toto pole v databázové tabulce bude jmenovat dle tohoto identifikátoru.

POZNÁMKA: Každý datový modul musí obsahovat primární klíč. V K2 je zvyklostí takováto pole definovat pomocí tzv. RID pole. V případě, že tyto konvence dodržíte a vložíte do identifikátoru hodnotu ***„RID“***, doplní návrhář objektů zbytek informací do zobrazeného formuláře dle obvyklé definice primárního klíče, viz Obrázek 14 - Definice primárního klíče v novém datovém modulu. Tedy doplní do pole ***„Název“*** obvyklou hodnotu ***„Číslo“***, jako datový typ definuje ***„BigInt“***, nastaví příznaky ***„Primární“***, ***„Povinné“***, ***„Unikátní“*** a ***„Automatická hodnota“***. Návrhář objektů se v tomto případě postará i o definici databázových klíčů. Více k jednotlivým vlastnostem polí a klíčů bude dále v textu.

POZNÁMKA: Stejně jako se automaticky doplní údaje při vyplnění identifikátoru s textem ***„RID“***, existuje několik dalších typů polí, která jsou obvyklá, a návrhář objektů takováto identifikuje a vyplní za nás. Jsou to například: ***„Abbr/Zkratka“***, ***„Description/Popis“*** a ***„Memo/Text“***. Uvést můžete buď anglickou, nebo českou variantu.



Obrázek 14 - Definice primárního klíče v novém datovém modulu

Číslo
Je jednoznačný číselný identifikátor pole v rámci datového modulu. Tento údaj je doplněn automaticky při zakládání nového pole. Uživatel nemusí tuto hodnotu měnit.

Název
Definuje název pole, tedy hodnotu, která se bude zobrazovat ve formulářích v případě použití tohoto pole.

Datový typ
Určuje, jakého datového typu vytvářené pole bude. Tedy v případě primárního klíče je například předvolena hodnota „BigInt", což je velké celé číslo, v případě textového pole bude vybrána hodnota „WideString". Více o nejpoužívanějších datových typech polí je popsáno v následující tabulce Tabulka 1 - Nejpoužívanější datové typy polí v NO.

Tabulka 1 - Nejpoužívanější datové typy polí v NO

DATOVÝ TYP POLE	POPIS
Unknown	Výchozí hodnota – značí nevybraný typ. Nelze použít, typ musí být vybrán.
AnsiString	Textová hodnota
Byte	Celé číslo z intervalu 0 až 255
Short	Celé číslo z intervalu -32 768 až 32 767
Long	Celé číslo z intervalu - 2 147 483 648 až 2 147 483 647
UShort	Celé číslo z intervalu 0 až 65 535
Real	Desetinné číslo z intervalu 2.9 x 10^–39 ... 1.7 x 10^38
Decimal	Desetinné číslo s možností definice počtu desetinných míst – maximálně čtyři, fyzicky je uloženo jako celé číslo
Time	Časový údaj
Date	Datumový údaj (neobsahuje časovou složku)
DateTime	Datum a čas
BigInt	Celé číslo z intervalu - 9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
Currency	Desetinné číslo z intervalu – 9 223 372 036 854 77.5808 ... 9 223 372 036 854 77.5807
GUID	Identifikace typu - Universally unique identifier
WideString	Textová hodnota s podporou Unicode
CLOB	Řetězec neomezené délky – poznámky, dokumentace
BLOB	Binární data – soubory, obrázky
XML	XML formát
Bool	Binární pole – ano/ne, true/false
TimeStamp	Časové razítko

Hint
Popis pole, který se bude zobrazovat jako nápověda v případě, že uživatel najede myší nad dané pole ve formuláři.

Při kopírování
Pomocí tohoto nastavení můžete změnit chování pole v případě, že dochází ke kopii záznamu. Možnosti, jak se má pole chovat při kopii jsou následující.

Výchozí
V tomto případě, použije návrhář objektů jednu z níže uvedených variant, kterou nelze blíže specifikovat. Rozhoduje se na základě dalších nastavení, která jsou v K2 k dispozici. Pokud potřebujete mít přesně definováno chování pole při kopii, doporučuji vybrat variantu, která je požadována.

Nedělat nic
Při tomto nastavení se s polem nic dít nebude. Tato vlastnost je vhodná použití, za předpokladu, že plníte pole při kopii nějakou vlastní logikou. Pak se nestane, že po doplnění je hodnota přepsána standardním mechanismem. Tedy v případě, že plníte pole vlastní implementací, použijte tuto hodnotu.

Kopírovat
Při tomto nastavení se přebere hodnota z kopírovaného záznamu.

Vyprázdnit
Při tomto nastavení se pole vyčistí. Tedy v poli nebude nastavena žádná hodnota.

RID
Tato varianta je zde pouze pro speciální případy. Nedoporučujeme tuto variantu používat. Je určena zkušeným administrátorům, kteří jsou dobře obeznámeni s implementací jádra K2.

Právo
Slouží k omezení přístupu uživatele k hodnotám pole dle existujícího práva K2. Hodnotu vybíráme ze seznamu práv. V případě, že uživatel nemá právo přiděleno, zobrazí se mu v tomto poli text "Nepovoleno", v případě sloupců pak hvězdičky jako zástupný symbol.

POZNÁMKA: Lze použít i vlastní práva vytvořená přes NO. K vlastním právům bude více řečeno v kapitole 10. Záložka „Práva".

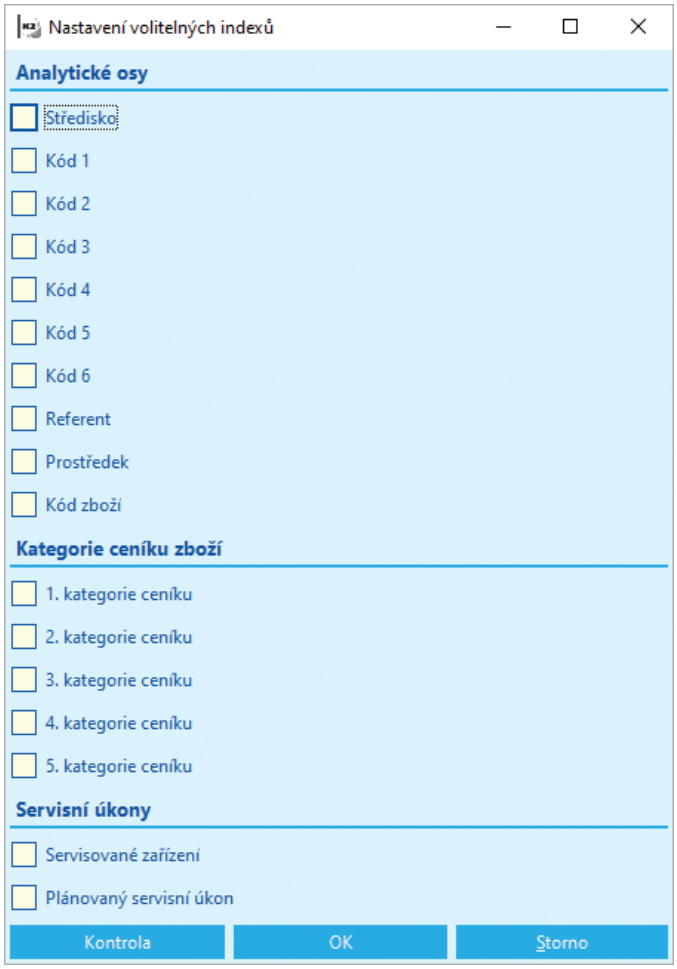
Len
Určuje velikost pole, tedy počet znaků, které je možné do pole vložit. Nastavení této proměnné má smysl pouze u textových typů polí, kde je potřeba specifikovat, jak dlouhý řetězec může pole pojmout.

Decimals
Určuje počet desetinných míst pole v případě, že se jedná o reální (desetinné) číslo.

Picture
Nastavuje formátování hodnoty při zobrazení ve formuláři. Např.: ***„N-10"*** – číslo o deseti znacích, ***„S20"*** – text o délce 20 znaků nebo ***„N-15´2"*** – reálné číslo o délce patnáct znaků, kde se do celkové délky započítává i znaménko, desetinná čárka, mezery apod.

Volitelný index
V případě, že definované pole obsahuje odkaz do některého z číselníku K2, kterému říkáme analytická osa, jedná se o ***„Středisko"***, ***„Kód 1"***, ***„Kód 2"***, ***„Referent"***, ***„Prostředek"*** a ***„Kód zboží"***, můžeme nastavit vlastnost ***„Volitelný index"*** na hodnotu dle typu analytické osy a K2 vytvoří databázový index, který nám zrychlí práci s tímto polem v případě čtení záznamů. Jednotlivé hodnoty vlastnosti ***„Volitelný index"*** jsou dle názvu vypovídající, nemusíme je tedy popisovat.

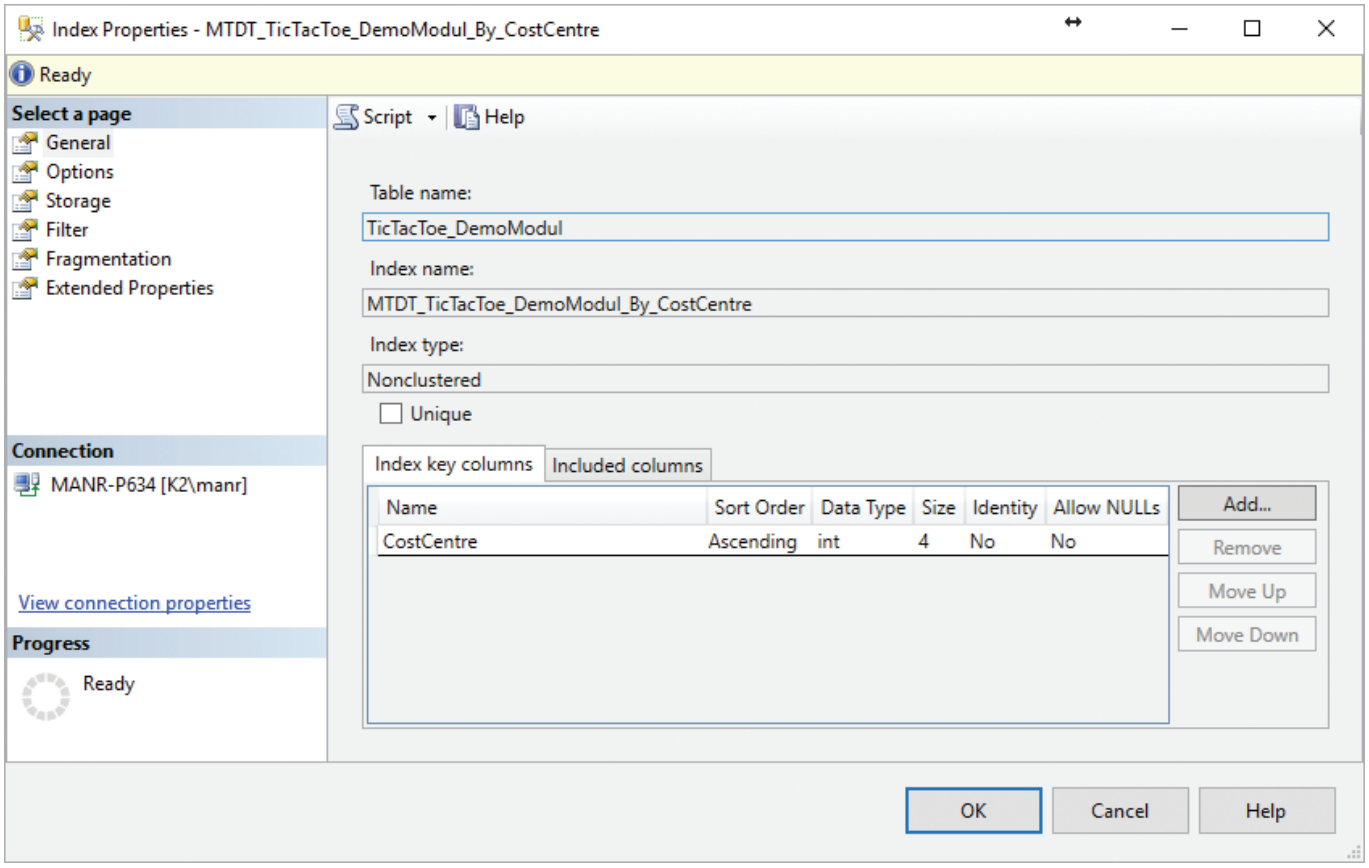
Aby se volitelné indexy vytvářely nad danými poli, nestačí takto pole označit v návrháři objektů. Je potřeba mít tento mechanismus povolený v K2 pro konkrétní analytické osy. K2 pak dle tohoto nastavení indexy vytváří a používá. Zapnutí indexování analytických os provedeme v ***„Nastavení volitelných indexů"***, které je dostupné v sekci ***„Správce/Analytické osy"***, viz Obrázek 15 - Nastavení volitelných indexů v K2.



Obrázek 15 - Nastavení volitelných indexů v K2

Pokud máme povolený volitelný index na analytické ose v K2 a v návrháři objektů na poli nastavíme, že se jedná o analytickou osu, pak nám K2 při akci „Deploy", při instalaci balíčku nebo případně při opětovném zapnutí volitelných indexů ve správě, vytvoří v databázové tabulce index nad tímto polem, viz Obrázek 16 - Vytvořený volitelný index v MSSQL. Tyto indexy vždy začínají prefixem „MTDT".

POZNÁMKA: Vytvořený index není evidován v seznamu klíčů na záložce „Klíče" datového modulu.



Obrázek 16 - Vytvořený volitelný index v MSSQL

4.2.1.2. Příznaky

Pro každé pole je možné nastavit vlastnosti, které definují jeho chování. Následující výčet jednotlivé vlastnosti popisuje.

Primární
Tato vlastnost udává, zda je dané pole určeno jako primární klíč. Takovýto příznak může být nastaven pouze na jednom poli v datovém modulu.

POZNÁMKA: Pokud je pole označeno jako ***„Primární“*** je pro něj automaticky vytvořen index a klíč, který je označen jako ***„unikátní“*** a ***„primární“***. Více pak v podkapitole zabývající se databázovými klíči a indexy.

Pouze pro čtení, UI
Takto označené pole slouží pouze pro čtení. Formuláře na tuto vlastnost reagují tak, že je pole nastaveno pouze pro čtení, tedy nejde změnit.

Povinné
V případě, že má pole nastaven tento příznak, znamená to, že je nutné vždy vyplnit hodnotou. Pokud tak neučiníme, K2 nám nedovolí záznam uložit. Formuláře na tuto vlastnost reagují validací vstupů a upozorňují uživatele, že nemá vyplněno povinné pole.

Unikátní
Takto označené pole musí obsahovat vždy unikátní hodnotu v rámci celého datového modulu. Pokud tomu tak není, uživatel je o tomto informován chybou, ke které dojde při ukládání záznamů, které porušují unikátnost.

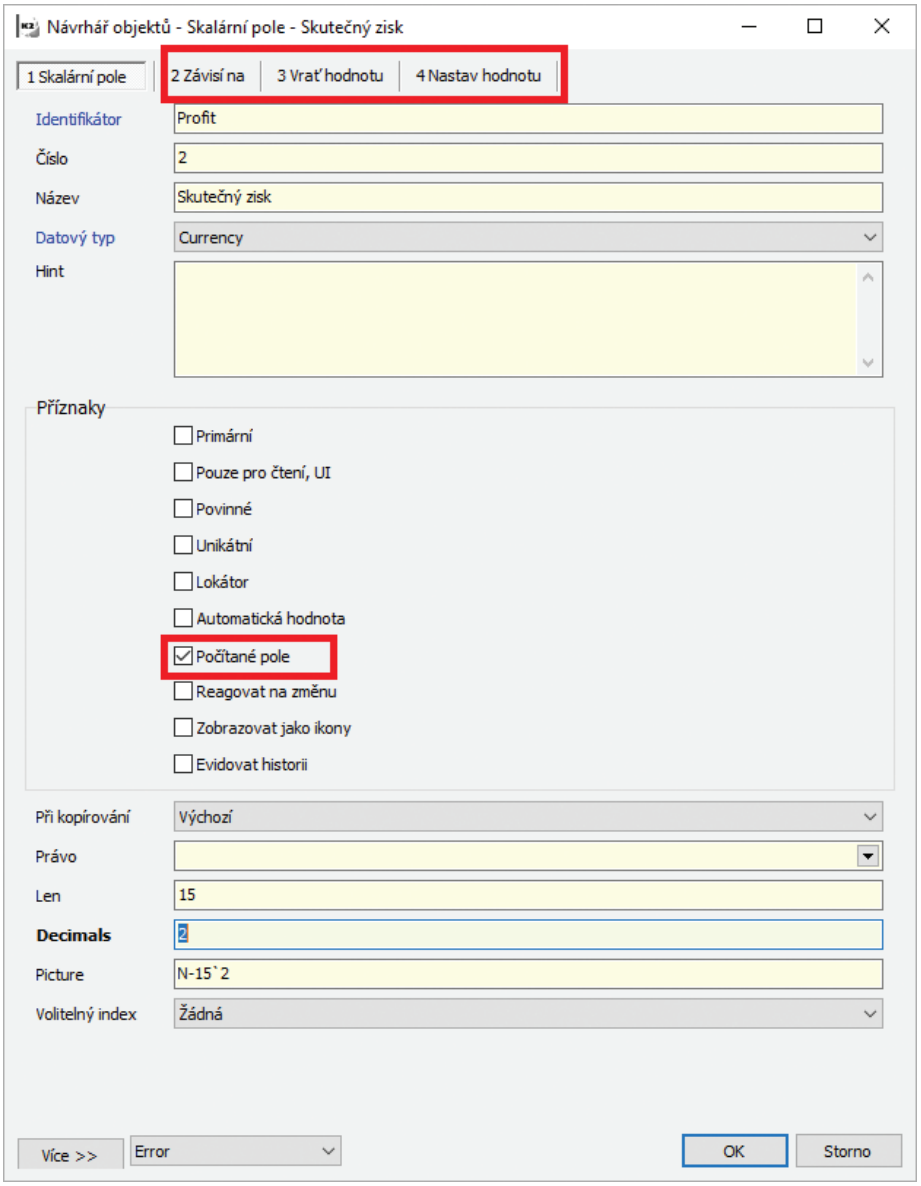
POZNÁMKA: Pokud je pole označeno jako ***„Unikátní“*** je pro něj automaticky vytvořen index, který je označen jako ***„unikátní“***. Více pak v podkapitole zabývající se databázovými klíči a indexy.

Lokátor
Takto označené pole lze použít k rychlejšímu vyhledávání v datovém modulu ve stavu ***„Kniha“***. Pomocí této vlastnosti zařadíme pole do seznamu lokátoru, který se objevuje nad datovým modulem ve formuláři.

POZNÁMKA: Pokud je pole označeno jako ***„Lokátor“*** je pro něj automaticky vytvořen index, který je označen jako ***„unikátní“*** a má stav ***„lokátor“***. Pole je zahrnuto v klíči, kde prvním segmentem je takto označené pole a v případě, že se nejedná o pole s unikátními hodnotami, je připojen druhý segment, který obsahuje pole z primárního klíče (pole je označeno jako ***„Primární“***). Více pak v podkapitole zabývající se databázovými klíči a indexy.

Automatická hodnota
Do takto nastaveného pole se při vytváření nového záznamu automaticky doplňuje hodnota dle algoritmu pro daný typ. Např. pole označená jako primární by měla mít vždy automaticky nastaveno, že se mají automaticky vyplňovat. Tento mechanismus zajistí unikátnost při plnění tohoto pole, tudíž se o to nemusí starat uživatel ani programátor.

Počítané pole
V případě nastavení vlastnosti pole jako ***„Počítané“***, objeví se na formuláři, v záhlaví, další tři záložky – ***„Závisí na“***, ***„Vrať hodnotu“*** a ***„Nastav hodnotu“*** viz Obrázek 17 - Definice počítaného pole datového modulu. Tyto záložky slouží k definici, jakým způsobem se pole bude počítat. V případě, že potřebujete implementovat chování pole při čtení jeho hodnoty, pracujete v záložce ***„Vrať hodnotu“***. V případě, že potřebujete řídit i nastavení hodnoty počítaného pole, což je méně častá varianta, pak je potřeba implementovat logiku v záložce ***„Nastav hodnotu“***.

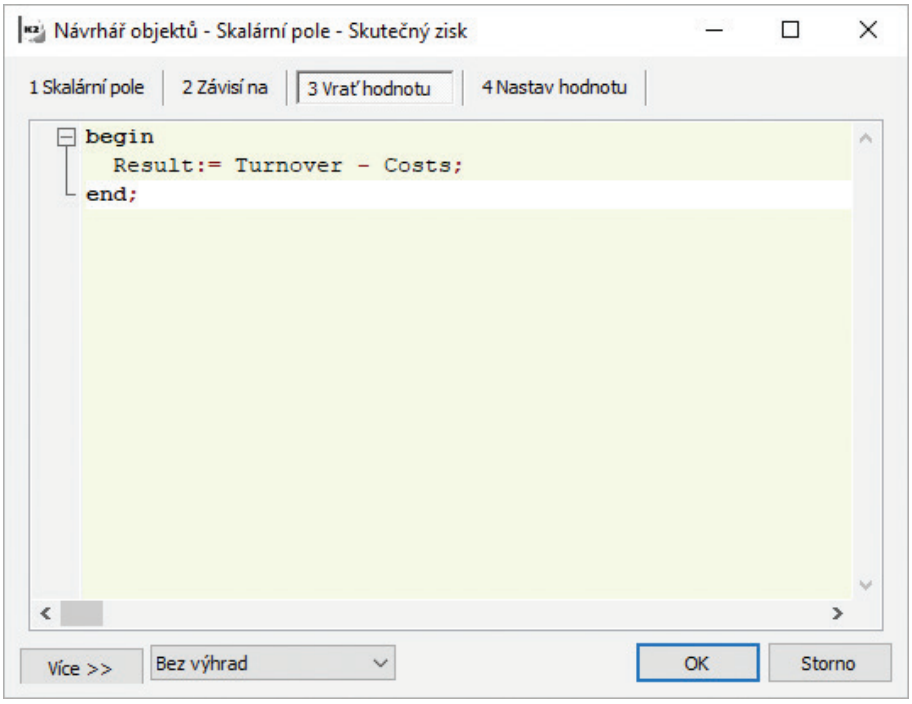


Obrázek 17 - Definice počítaného pole datového modulu

Implementace logiky ***„počítání“*** pole je na úrovni K2 skriptu. Jako záložku ***„Vrať hodnotu“*** si můžete představit funkci, která vrací hodnotu vypočítaného pole. V případě záložky ***„Nastav hodnotu“*** se jedná o proceduru, kde vstupním parametrem je hodnota, která je do pole vkládána. Vaším úkolem je implementovat tělo těchto funkcí a procedur. Návrhář objekt automaticky generuje definici, samotnou implementaci zajišťuje programátor, uživatel návrháře objektů. Pokud v jednotlivých záložkách nebude nic definováno, pak nebude návrhářem objektů vygenerována ani definice procedury či funkce.

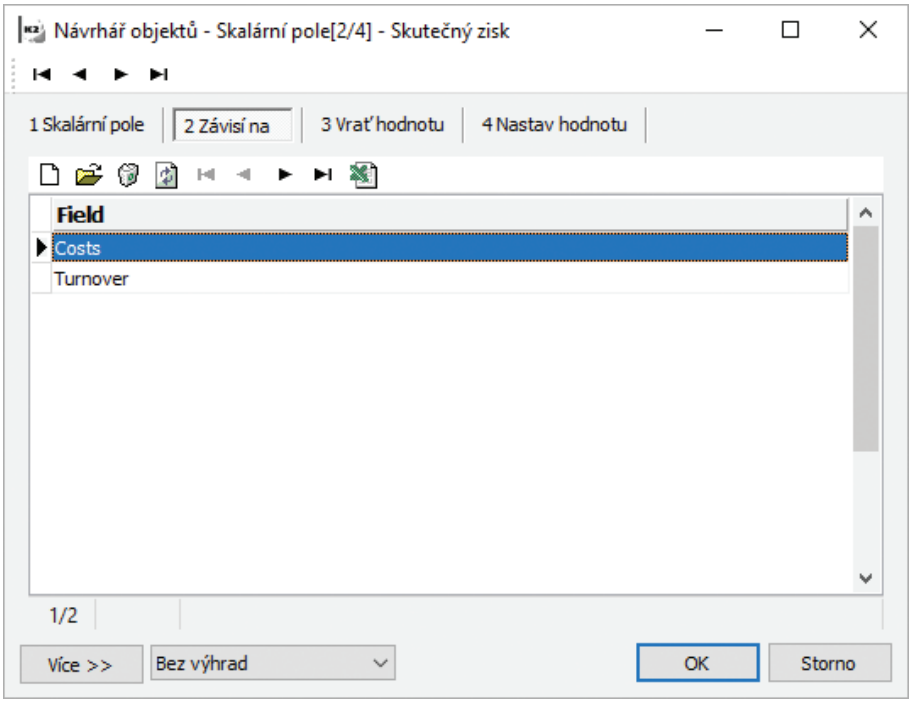
Ve formuláři je k dispozici také záložka ***„Závisí na“***. Zde můžeme definovat, na kterých polích je právě definované počítané pole závislé. Což jinými slovy znamená, že pokud je v závislém poli změněna hodnota, pak se automaticky spouští výpočet i pro definované počítané pole. Definice provedete vložením polí do seznamu podle toho, která považujete za závislá.

PŘÍKLAD: Nyní si ukážeme implementaci funkce na výpočet hodnoty včetně závislostí. Jak je vidět na Obrázek 18 - Implementace funkce pro počítané pole datového modulu, je potřeba definovat tělo funkce pro výpočet hodnoty. Návrhář sám vygeneruje klíčová slova na začátku „begin“ a na konci „end“. My doplníme samotný výkonný kód funkce. Např. na příkladu je vidět, že do návratové proměnné funkce „Result“ je vložen rozdíl polí „Turnover“ a „Costs“, která jsou definována ve stejném datovém modulu jako počítané pole. Tedy příklad ukazuje výpočet zisku.



Obrázek 18 - Implementace funkce pro počítané pole datového modulu

Protože máme výpočet prováděn na základě výsledků dvou polí, která existují v našem modulu, je žádoucí, aby se výpočet aktualizoval vždy, když je hodnota některého z polí ve výpočtu změněna. Závislost nadefinujeme na záložce ***„Závisí na“*** jak je vidět na Obrázek 19 - Definice závislostí počítaných polí datového modulu.

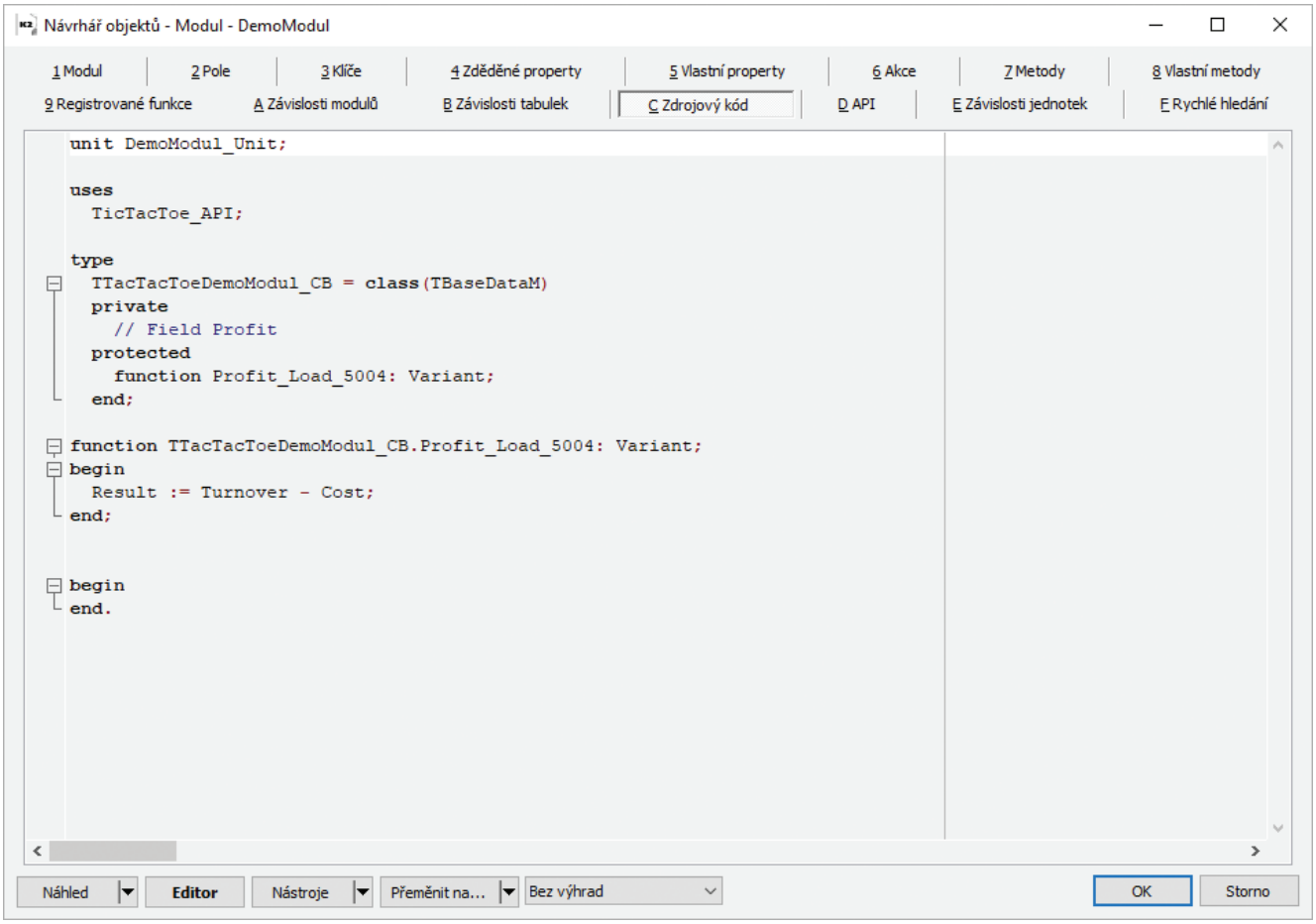


Obrázek 19 - Definice závislostí počítaných polí datového modulu

V takto definovaném modulu, pak definované počítané pole ***„Profit“*** funguje tak, že v případě, že změníme hodnotu polí ***„Costs“*** nebo ***„Turnover“*** nebo v případě, že uložíme záznam, je vyvolána akce přepočtu pole, tedy v počítaném poli se aktualizuje rozdíl ***„Costs“*** a ***„Turnover“*** a výsledek se zobrazí v poli ***„Profit“***.

POZNÁMKA: Kompletní funkci, včetně celého skriptu pro daný datový modul, pak můžeme vidět na záložce ***„Zdrojový kód“***, viz Obrázek 20 - Kompletní funkce pro počítané pole datového modulu, více o této záložce se dozvíte v podkapitole Zdrojový kód.

POZNÁMKA: Návrhář objektů nabízí i jinou možnost pro implementaci skriptových částí datových modulů, jako u výše zmíněného počítaného pole, dále pak pole zobrazeno jako ikona, ale také u vlastností a metod návrháře objektů, se kterými se seznámíme v dalším textu. Možností je spustit v návrháři objektů editor skriptu, ve kterém můžeme tyto metody generovat, včetně ***„begin“*** a ***„end;“*** a dokonce přímo implementovat těla těchto konstrukcí. Po uzavření editoru se nám implementace samozřejmě zobrazí v částech návrháře, kde patří dle druhu. Více si o editoru povíme v kapitole 4.18. Editor.



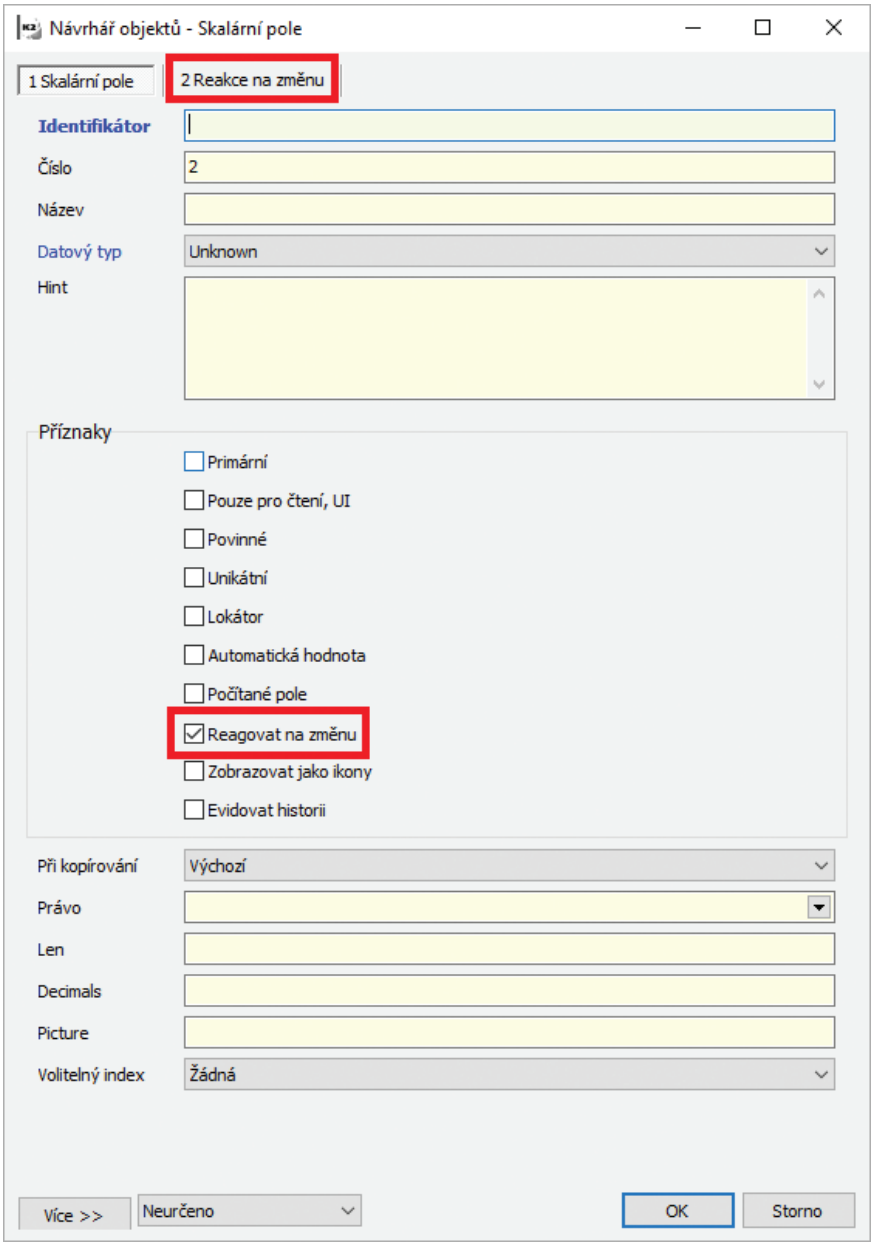
Obrázek 20 - Kompletní funkce pro počítané pole datového modulu

POZNÁMKA: Ve skriptech, které jsou implementovány v návrháři objektů, nelze používat proměnné ***„AktDM“*** a ***„AktForm“***. Tyto proměnné nejsou nikdy nastaveny, jejich hodnota je vždy ***„nil“***.

Reagovat na změnu

V případě, že potřebujeme reagovat na změnu hodnoty pole, využijeme vlastnosti ***„Reagovat na změnu“***. Po jejím nastavení se objeví na formuláři v záhlaví další záložka - ***„Reakce na změnu“*** viz Obrázek 21 - Definice reakce na změnu na poli datového modulu. Zde je třeba implementovat kód, který se provádí při změně pole. Implementaci provedeme podobně jako v předchozím případu. Návrhář nám opět vygeneroval klíčová slova ***„begin“*** a ***„end;“***. Pro tuto vlastnost návrhář generuje definici procedury, která má dva vstupní parametry, kdy první parametr obsahuje původní hodnotu v aktuálním poli předtím, než došlo k její změně. Druhý parametr definuje číslo bitu. Tuto informaci programátor, v případě definice v návrháři objektů, nevyužije. Nebudeme se jí tedy zabývat.

POZNÁMKA: Jak již bylo zmíněno, k implementaci skriptových částí návrháře objektů je doporučeno používání editoru skriptu, který je integrován v návrháři objektů viz kapitola 4.18. Editor.

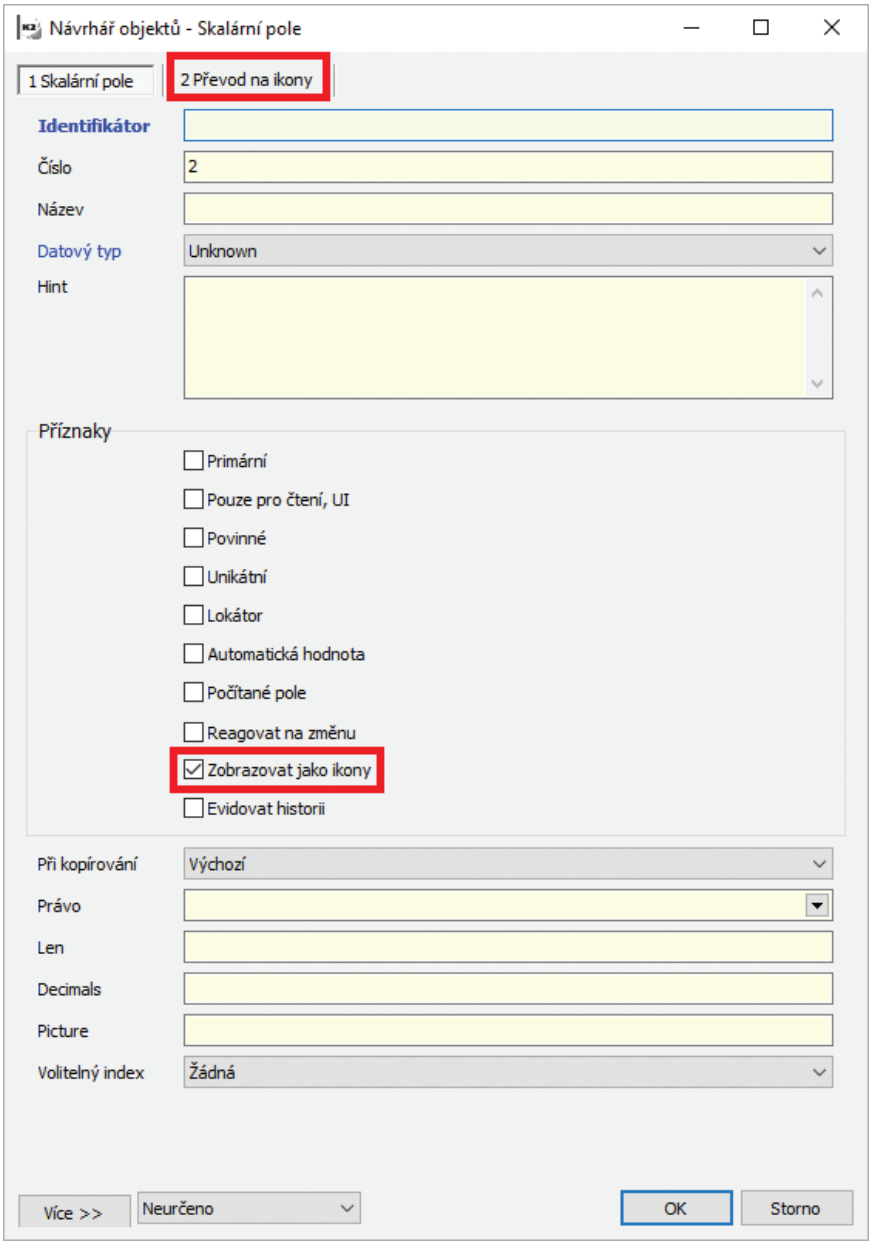


Obrázek 21 - Definice reakce na změnu na poli datového modulu

Zobrazovat jako ikony

V případě nastavení vlastnosti pole ***„Zobrazovat jako ikony“***, objeví se na formuláři, v záhlaví, další záložka - ***„Převod na ikony“*** viz Obrázek 22 - Definice zobrazovat jako ikony na poli datového modulu. Tuto vlastnost použijeme v případě, že chceme, kromě klasického zobrazení hodnoty pole, zobrazovat pole jako obrázek. Tedy pokud vložíme pole do seznamu nebo formuláře, můžeme jeho hodnotu zobrazit jako obrázek. Logiku, jaký obrázek zobrazíme, řešíme pomocí záložky ***„Převod na ikony“***. Návrhář objektů vygeneruje funkci, která má jako vstupní parametr proměnnou typu ***„TDataField“***, která je jakýmsi popisovačem (deskriptorem) daného pole. Návratovým typem funkce je pak řetězec, který představuje identifikátor obrázku. Implementaci provedeme podobně jako v předchozím případu, kdy mezi klíčové slova ***„begin“*** a ***„end;“*** zapíše kód.

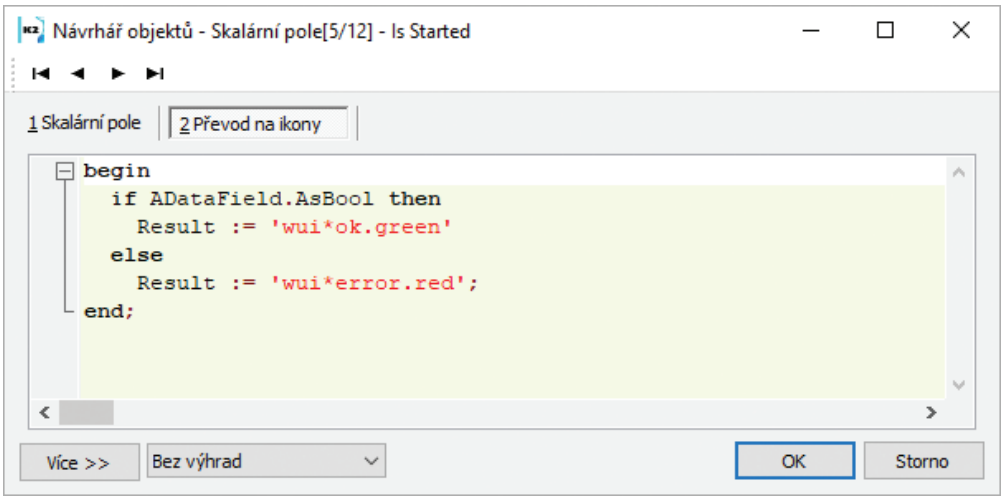
POZNÁMKA: Jak již bylo zmíněno, k implementaci skriptových částí návrháře objektů je doporučeno používání editoru skriptu, který je integrován v návrháři objektů viz kapitola 4.18. Editor.



Obrázek 22 - Definice zobrazovat jako ikony na poli datového modulu

PŘÍKLAD: Jako ukázku nadefinujeme pole s názvem ***„IsStarted“***, které je typu ***„Bool“***, tedy nabývá hodnot ***„ano/ne“***. Zatrhneme vlastnost ***„Zobrazovat jako ikony“*** a v záložce ***„Převod na ikony“*** vložíme K2 skript, pomocí kterého toto pole zobrazíme jako obrázek. Logika je následující: Pokud je hodnota pole ***„true (ano)“***, pak funkce vrátí hodnotu, která je identifikátorem obrázku, např.: ***„wui*ok.green“***, v případě hodnoty ***„false (ne)“*** funkce vrátí hodnotu ***„wui*error.red“*** viz Obrázek 23 - Implementace převodu hodnoty pole na ikony v poli pro datový modul.

POZNÁMKA: Identifikátor obrázku je složen z adresáře ***„wui“***, který se nachází v ***„Conf\Img“***, dále jméno souboru bez přípony. Oddělovačem adresáře od souboru je symbol hvězdičky. Např. za identifikátorem ***„wui*error.red“*** se nachází vektorový obrázek, který se nachází v adresáři se jménem souboru ***„K2\Conf\Img\wui\error.red.svg“*** a vypadá takto .



Obrázek 23 - Implementace převodu hodnoty pole na ikony v poli pro datový modul

Evidovat historii
V případě, že potřebujeme sledovat historií hodnot na daném poli. Použijeme vlastnost ***„Evidovat historii“***. K2 se pak automaticky postará o záznam těchto změn.

POZNÁMKA: Datový modul musí obsahovat primární klíč, který musí být celočíselného typu (např. BigInt, Long, Short, ...) a být jedinečný. Pokud ponese název ***„RID“***, uživatel nemusí nic vyplňovat navíc, pouze zatrhne fajfku na poli, kde chce evidovat historii pole ***„Evidovat historii pole“***. Pokud primární klíč neponese název pole ***„RID“***, ale např. ***„ID“***, tak uživatel musí navíc na záložce ***„Zděděné property“*** toto pole vložit do property ***„HistoryFieldNumber“***.

Full Text
Tato vlastnost je přístupná pouze pro textová pole, tedy ***„AnsiString“***, ***„WideString“*** apod. V případě, že zatrhneme, získáváme pro toto pole podporu fulltextového vyhledávání, které je realizováno na databázové úrovni. Tato vlastnost umožňuje uživateli nad tímto polem efektivněji a rychleji vyhledávat.

Identity
Tato vlastnost je přístupná pouze pro pole, které jsou celočíselného datového typu. Slouží k definici pole, které se na úrovni SQL serveru automaticky navyšuje o ***„1“*** v případě ukládání záznamů. Tuto vlastnost může mít nastaveno pouze jedno pole v datovém modulu. V případě, že ho takto nastavíme, automaticky se nám nastaví i další vlastnosti a to ***„Unikátní“***, ***„Automatická hodnota“*** a ***„Pouze pro čtení, UI“***.

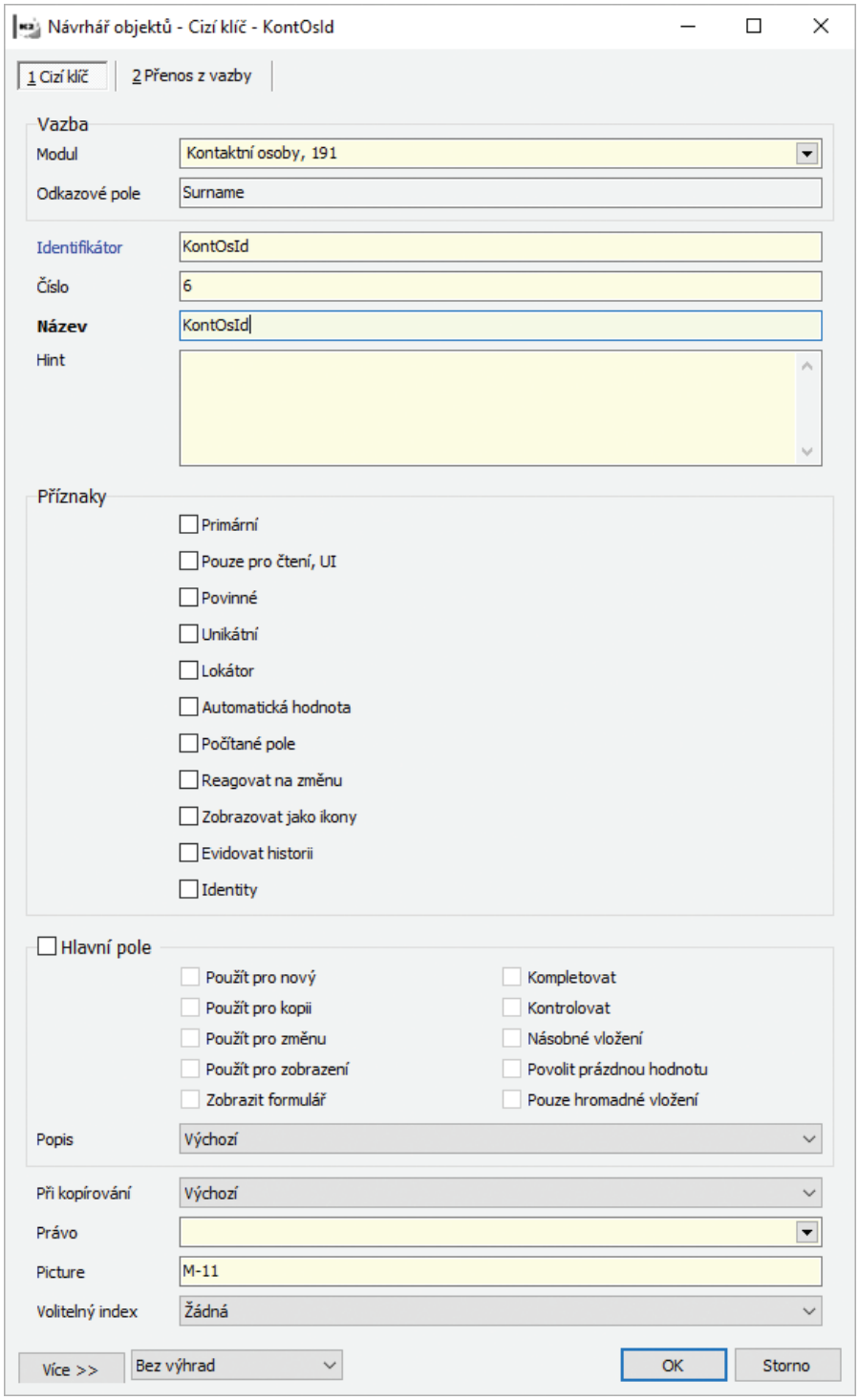
POZNÁMKA: V případě MS SQL serveru se jedná o implementaci s využitím vlastnosti ***„Identity“***, v případě Oracle se jedná o využití vlastnosti ***„Sequence“***. Obecně se této vlastnosti říká ***„AutoIncrement“***.

POZNÁMKA: Vlastnost ***„Identity“*** lze na poli nastavit i v případě, že se jedná o paměťový datový modul. Zde se samozřejmě nevyužije algoritmu SQL serveru, ale inkrement pole se provádí v rámci logiky K2 v paměti.

4.2.2. CIZÍ KLÍČ

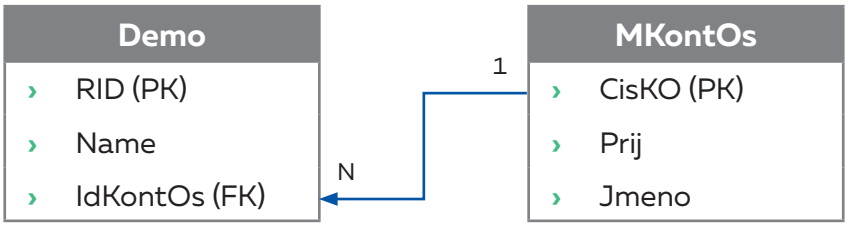
Vložení nového pole typu ***„cizí klíč“*** slouží k vytvoření vazby do jiného datového modulu. Tedy vzniká nám cizí klíč, který je navázán na primární klíč z datového modulu, do kterého vytváříme vazbu.

Po výběru typu pole ***„cizí klíč“*** se zobrazí formulář Obrázek 24 - Definice pole jako cizí klíč v datovém modulu. Jednotlivé části si nyní popíšeme.



Obrázek 24 - Definice pole jako cizí klíč v datovém modulu

Na Obrázek 25 - Schéma vazby pomocí cizího klíče je demonstrován princip vazby formou cizího klíče.



Obrázek 25 - Schéma vazby pomocí cizího klíče

4.2.2.1. Vazba – modul

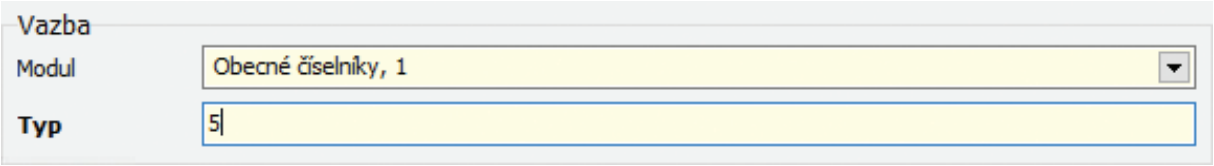
Protože pole typu ***„cizí klíč“*** je odkaz do jiného datového modulu, musíme nejprve definovat, do jakého datového modulu tato vazba bude směřovat.

Otevřeme rozbalovací nabídku s názvem ***„Modul“*** a vybereme ze seznamu potřebný datový modul, například ***„Kontaktní osoby“***. Návrhář objektů se již o zbytek postará sám, tedy nastaví správný datový typ pro pole a prováže s druhým modulem přes primární klíč.

4.2.2.2. Vazba – Odkazové pole

V tomto pole je ***„read only“*** tedy pouze pro čtení. Slouží nám jako odkazové pole, které lze použít např. ve funkci LinkReferenceStr1().

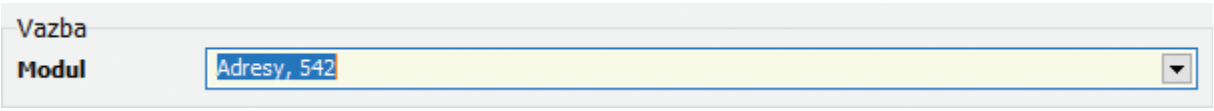
V případě, že vybereme cílový datový modul, který je typovým číselníkem, například ***„Spestr“*** nebo ***„CodeType“***, přibude nám pod modulem ještě jedna volba ***„Typ“***, pomocí které říkáme, který typ z typového číselníku bude figurovat ve vazbě, viz Obrázek 26 - Definice vazby do typového číselníku v datovém modulu.



Obrázek 26 - Definice vazby do typového číselníku v datovém modulu

POZNÁMKA: Jak již bylo uvedeno výše, v seznamech typu seznam datových modulů, je možné, prostým psaním textu, odfiltrovat záznamy, které neobsahují psané (hledané) slovo. Můžeme si takto usnadnit práci při prohledávání velkého množství údajů.

POZNÁMKA: Vzhledem k tomu, že adresy v K2 jsou trochu specifické, doporučuje se pro cizí klíč do adres výhradně používat modul Adresy (TContactPersonAddressItemDM) s číslem 542, viz Obrázek 27 - Použití vazby do adres.



Obrázek 27 - Použití vazby do adres

4.2.2.3. Hlavní pole

Jednou z vlastností, kterou můžeme na cizím klíči (vazbě) nastavit, je tzv. ***„Hlavní pole“***. V případě, že vazbu takto označíme, získáváme funkcionalitu, která zajišťuje uživatelsky přijatelnější práci s tímto polem v rámci datového modulu. A to tak, že v případě, že budeme například vkládat či editovat záznamy (záleží na nastavení) v aktuálním modulu, nejprve se zobrazí výběr záznamu z tohoto cizího klíče, který se následně vloží do pole datového modulu a až poté se může zobrazit formulář s ostatními vstupy. Opět záleží na nastavení funkce ***„Hlavni pole“***.

Následující příznaky upravují chování funkce ***„Hlavní pole“***. Vždy musí být vyplněn alespoň jeden z příznaku, aby tato vlastnost mohla fungovat.

Použít pro nový
Pokud nastavíme příznak ***„Použít pro nový“***, pak se výběr záznamu ze seznamu, do kterého směřuje cizí klíč, nabídne vždy, když vkládáme nový záznam do datového modulu, ve kterém je cizí klíč definován.

Použít pro kopii
Pokud nastavíme příznak ***„Použít pro kopii“***, pak se výběr záznamu ze seznamu, do kterého směřuje cizí klíč, nabídne vždy, když vkládáme nový záznam do datového modulu, ve kterém je cizí klíč definován, formou kopií existujícího.

Použít pro změnu
Pokud nastavíme příznak ***„Použít pro změnu“***, pak se výběr záznamu ze seznamu, do kterého směřuje cizí klíč, nabídne vždy, když editujeme záznam v datovém modulu, ve kterém je cizí klíč definován.

Použít pro zobrazení
V případě, že je tato vlastnost hlavního pole nastavena, pak se při otevření detailu záznamu, otevírá seznam záznamů, do kterého vazba směřuje a pozice v seznamu je nastavena na záznam z pole v datovém modulu.

Vlastnost má smysl využít pouze v případě, že datový modul obsahuje pouze jedno pole typu ***„cizí klíč“***.

Zobrazit formulář
Tímto příznakem nastavujeme, zda se má po výběru záznamu pro cizí klíč, zobrazit formulář, který je nadefinován pro aktuální datový modul. Tedy nejprve vybereme záznam pro cizí klíč z formuláře, který se nám automaticky otevřel, a následně se zobrazí již klasický formulář, ve kterém doplníme další údaje záznamu v aktuálním datovém modulu.

V případě, že příznak nenastavíme, zobrazí se pouze formulář pro výběr záznamu pro cizí klíč a po výběru se záznam vloží. Takovéto chování lze použít v případě, že datový modul obsahuje pouze cizí klíč. Tedy slouží k jednoduchému seznamu.

Kompletovat a kontrolovat
Po výběru hodnoty se z hlavního pole volá funkce jádra ***„IntreralCompleteFileds/CheckFields“***, která slouží ke kontrolám, které může programátor implementovat. Rozdíl je v tom, že při kompletovat se funkce volá pouze v případě výběru hodnoty, nikdy poté. U kontrolovat se funkce volá v různých situacích. Kromě výběru hodnoty, také například při ukládání.

Násobné vložení
Slouží k násobnému vložení, pomocí označených hvězdiček.

Povolit prázdnou hodnotu
Je povolena prázdná hodnota.

Pouze hromadní vložení
Lze vkládat pouze hromadně.

Popis
Pomocí tohoto nastavení hlavního pole, můžeme ovlivnit chování textu v záhlaví formuláře, který se zobrazí pro výběru hodnoty z vazby. Následuje výčet variant s jejich popisem.

„výchozí“ – výchozí hodnota. Použije se popis definovaný na cizím klíči v aktuálním datovém modulu

„z pole“ – použije se hodnota z pole, které je zobrazováno z definované vazby

„z vazby“ – použije se hodnota z vazby, která může být jiná než na poli, které je definováno ve vazbě

„z datového modulu“ – použije se popis z datového modulu, do kterého směřuje vazba

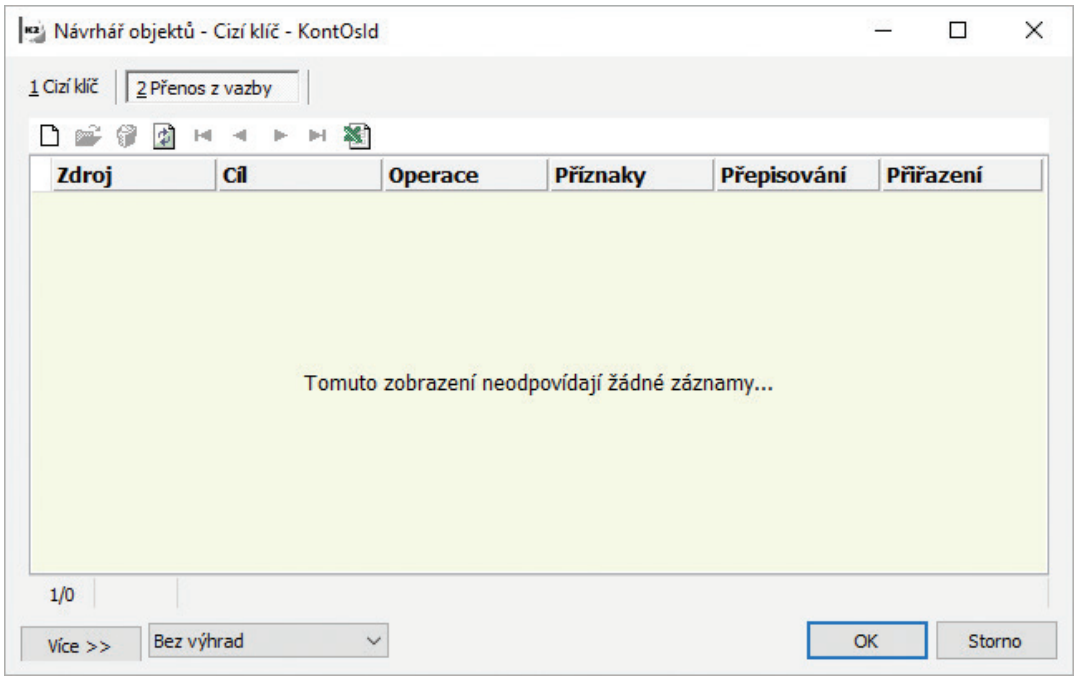
PŘÍKLAD: Představme si, že vytvoříme datový modul ***„DemoTest“***, který bude mít pole typu ***„cizí klíč“*** do datového modulu ***„Zboží“***. Na tomto cizím klíči nastavíme vlastnost ***„Hlavní pole“*** s použitím pro nové záznamy a dále s nastavením „Zobrazit formulář“. Následně promítneme definici do K2 (dle kapitoly 13.1. „Deploy“ – aplikování úprav do K2) a otevřeme si datový modul ***„DemoTest“***. Pomocí tlačítka Insert vložíme nový záznam. Nejprve se nám zobrazí seznam karet zboží, kde vybereme jednu hodnotu, která se vloží do nadefinovaného cizího klíče a formulář pro zboží se uzavírá. Nakonec se otevře již nadefinovaný formulář pro náš datový modul ***„DemoTest“***.

POZNÁMKA: V jednom datovém modulu může být pouze jedna vazba označená tímto příznakem.

Ostatní parametry vytvářeného pole jsou shodné s parametry popsanými v předchozí podkapitole 4.2.1. Skalár.

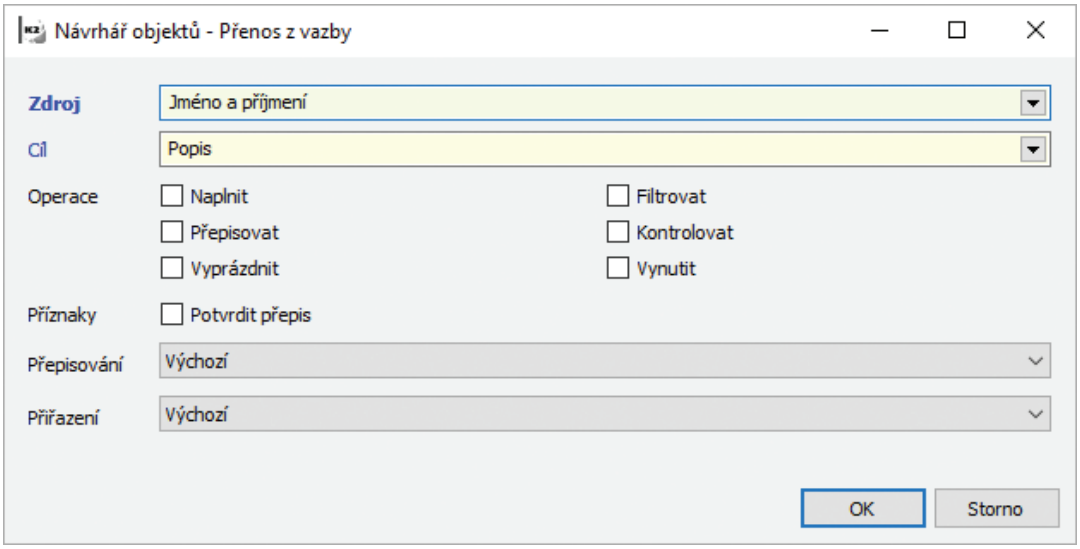
4.2.2.4. Přenos z vazby

Pomocí této záložky lze nastavit přenos hodnot z polí z modulu z definované vazby do polí aktuálního datového modulu. Přepnutím na záložku ***„Přenos z vazby“*** máme k dispozici seznam, ve kterém můžeme nadefinovat jednotlivé přenosy z vazby do našeho modulu, viz Obrázek 28 - Záložka "Přenos z vazby" na cizím klíči.



Obrázek 28 - Záložka "Přenos z vazby" na cizím klíči

Tlačítkem Nový vyvoláme nabídku pro vložení nového přenosu. Zobrazí se formulář, viz Obrázek 29 - Záznam v přenosu z vazby v cizím klíči.



Obrázek 29 - Záznam v přenosu z vazby v cizím klíči

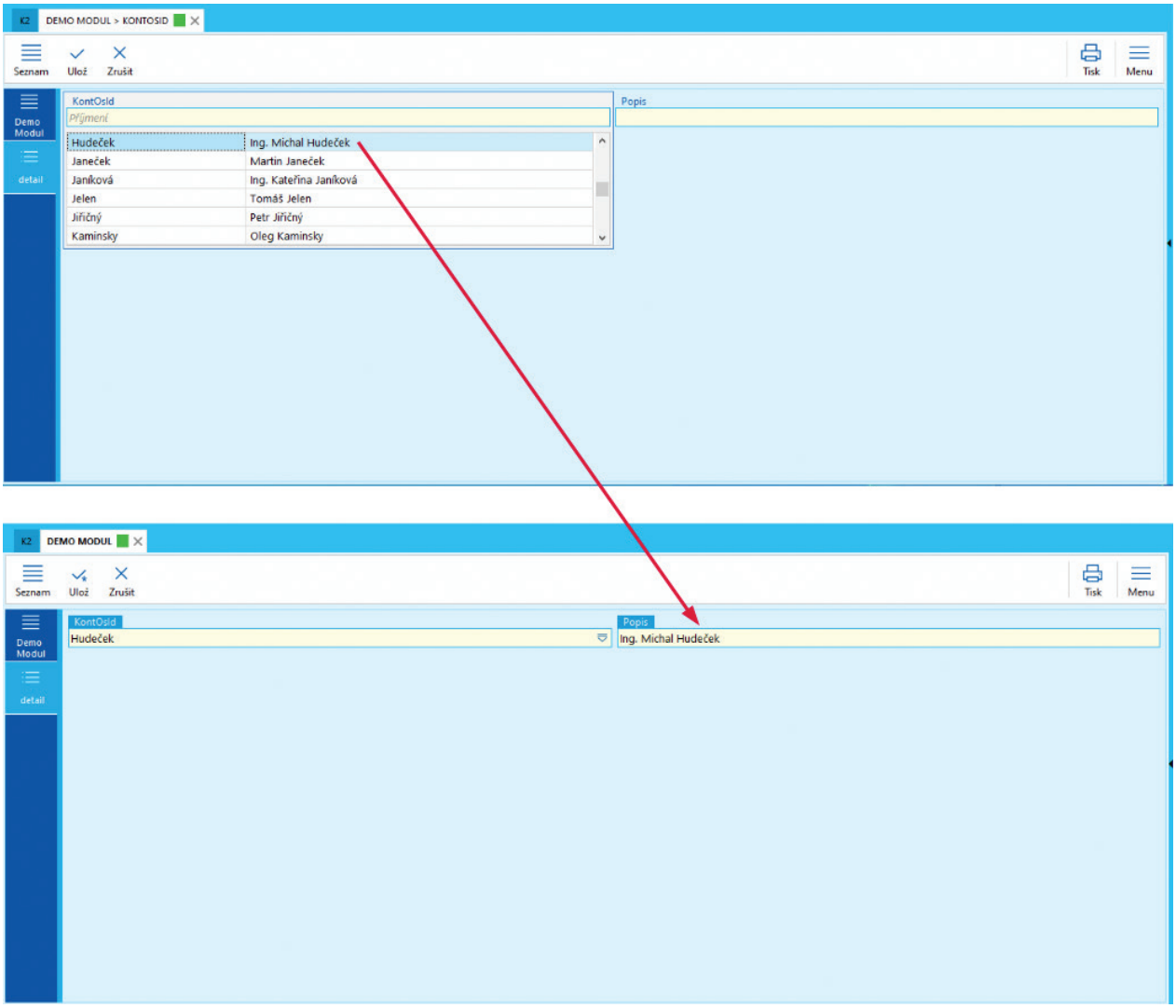
Na formuláři máme k dispozici vlastnosti, kterými definujeme chování přenosu z vazby. Následuje výčet všech včetně jejich popisu použití.

Zdroj
Zde vybíráme pole z datového modulu, do kterého směřuje vazba, a ze kterého chceme přenést hodnotu.

Cíl
Zde vybíráme pole z datového modulu, ve kterém máme definovanou vazbu (cizí klíč), a do kterého se přenese hodnota z vazby.

PŘÍKLAD: Nadefinujeme si vazbu do datového modulu ***„Kontaktní osoby“***. Z tohoto modulu přeneseme hodnotu z pole ***„SurenameForenameCalc“***, které obsahuje ***„Příjmení a jméno“*** z kontaktní osoby. Tuto hodnotu bychom chtěli přenést do pole ***„Popis“***, které máme v našem datovém modulu.

Na Obrázek 1 - Vložení návrháře objektů do plochy K2 tedy vybereme ve ***„Zdroj“*** pole ***„SurenameForenameCalc“***. Jako cíl vybereme pole ***„Popis“***. Uložíme a po provedení aplikace změn do K2 (***„Deploy“***) můžeme vyzkoušet, že při výběru kontaktní osoby se nám do pole ***„Popis“*** doplní jeho ***„Příjmení a jméno“*** viz Obrázek 30 - Příklad na přenos z vazby.



Obrázek 30 - Příklad na přenos z vazby

Operace
Při přenosu lze nastavit několik operací, které se mají provést. Varianty jsou následující.

POZNÁMKA: Pokud nevybereme žádnou operaci, pole vybrané v poli ***„Cíl“*** se naplní pouze poprvé, doporučuje se tedy při použití ***„přenos s vazby“*** použít i ***„Operace“***.

Naplnit
V případě, že je hodnota vazby vybrána, přenese se tato hodnota i do cílového pole, pokud cílové pole je neprázdné.

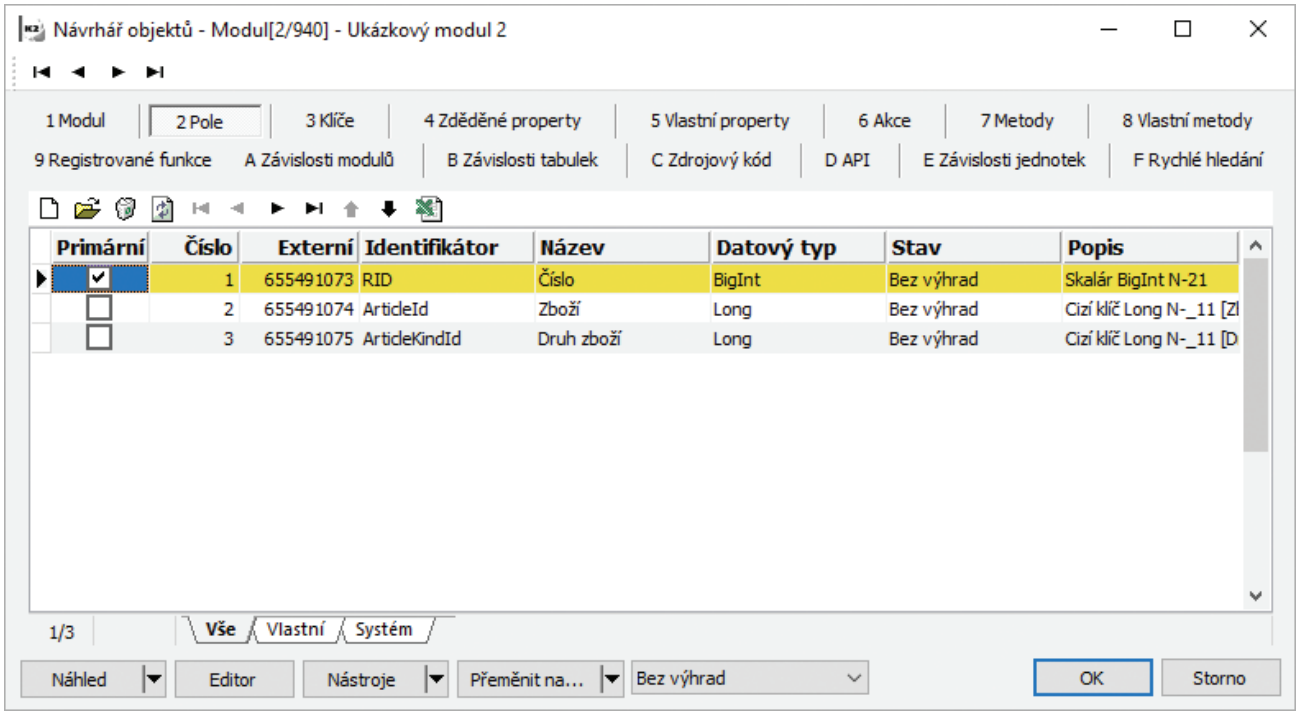
Přepisovat
Pokud nastavíme tuto operaci, pak se bude hodnota přepisovat v cílovém poli. Přesnější definici přepisování můžeme ještě doladit v nastavení ***„Přepisování“***, která je popsána níže.

Vyprázdnit
V případě, že je smazána hodnota vazby, je smazána i hodnota přenesená z vazby.

Filtrovat
Pomocí tohoto příznaku můžeme nastavit filtrování vazby, ze které přenášíme informace. Vazba je filtrována na základě vstupních dat z pole, které je definováno v nastavení cíle přenosu - ***„Cíl“*** a to tak, že se porovná hodnota pole z definice zdroje - ***„Zdroj“*** s hodnotou v poli z definice ***„Cíl“***.

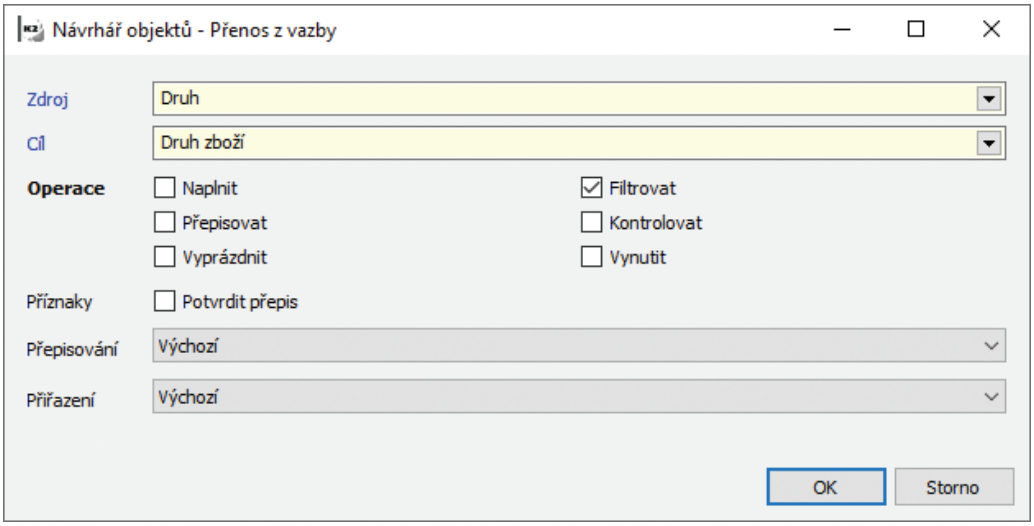
PŘÍKLAD: Jako příklad vytvoříme modul, ve kterém bude možné vybrat druh zboží z číselníku druhů zboží, a na základě této informace budeme filtrovat vazbu do seznamu zboží. Tedy budeme mít k dispozici na výběr pouze ty karty zboží, které budou odpovídat vybranému druhu zboží.

Založíme nový datový modul, který nazveme ***„ExampleModule2“*** - ***„Ukázkový modul 2“***. Vložíme klíčové pole ***„RID“***, dále pole ***„ArticleId“*** jako cizí klíč do modulu ***„Zboží“*** a ***„ArticleKindId“*** jako cizí klíč do modulu ***„Druh zboží“***, viz Obrázek 31 - Přenos z vazby - Filtrovat - Příklad datového modulu.



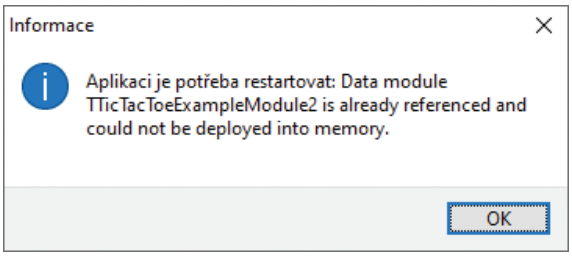
Obrázek 31 - Přenos z vazby - Filtrovat - Příklad datového modulu

Na cizím klíči ***„ArticleId“*** nadefinujeme na záložce ***„Přenos z vazby“*** záznam, kde nastavíme jako ***„Zdroj“*** pole ***„Druh“*** z modulu ***„Zboží“*** a jako ***„Cíl“*** zvolíme pole našeho modulu ***„Druh zboží“***. Hodnotu ***„Operace“*** nastavíme na ***„Filtrovat“***, viz Obrázek 32 - Přenos z vazby - Filtrovat - Příklad nastavení přenosu.

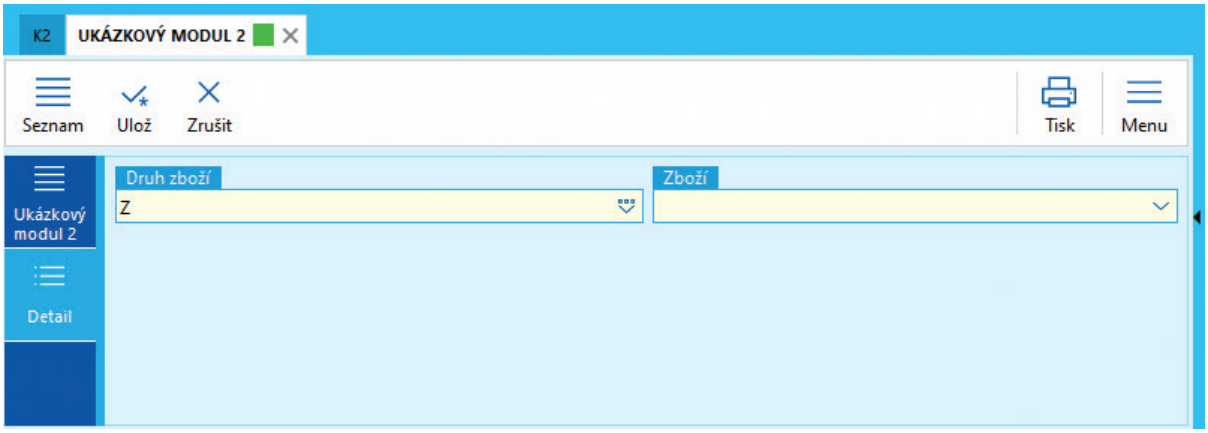


Obrázek 32 - Přenos z vazby - Filtrovat - Příklad nastavení přenosu

Po uložení, provedení operace ***„Deploy“*** a případném restartu K2 (pokud K2 sama vyhodnotí, že je zapotřebí restart, viz Obrázek 33 - Hlášení po "Deploy", že má být proveden restart K2) můžeme otevřít nový datový modul a zkusit vložit nový záznam. Nejprve vyplníme hodnotu ve vazbě do modulu ***„Druh zboží“***. Vybereme například ***„Z“*** – „Zboží“, Obrázek 34 - Přenos z vazby – Filtrovat - použití modulu.

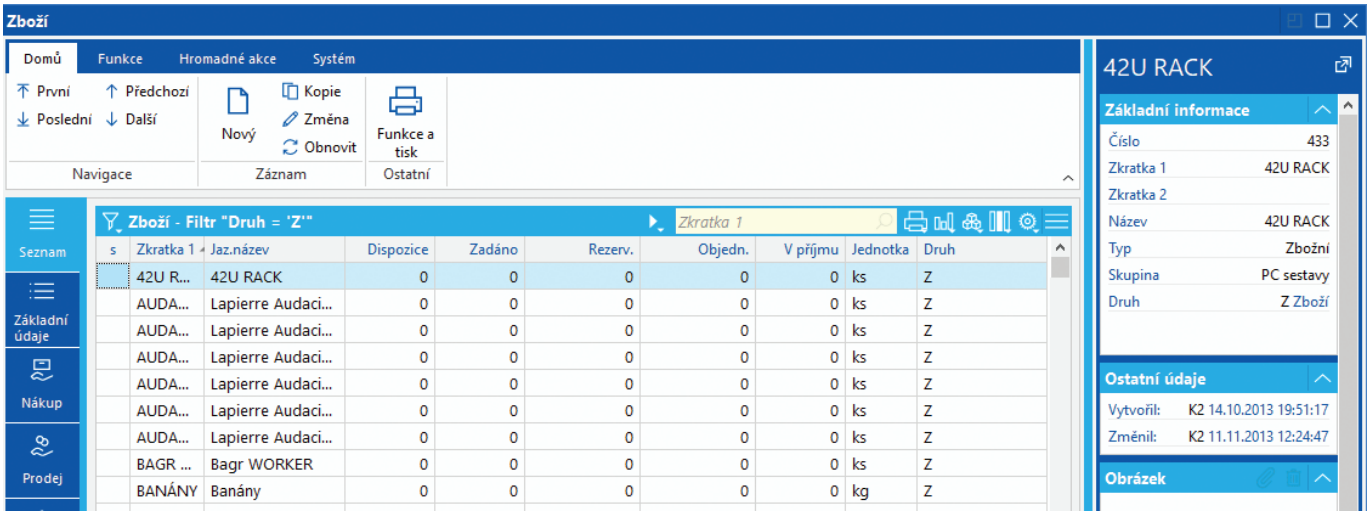


Obrázek 33 - Hlášení po "Deploy", že má být proveden restart K2



Obrázek 34 - Přenos z vazby – Filtrovat - použití modulu

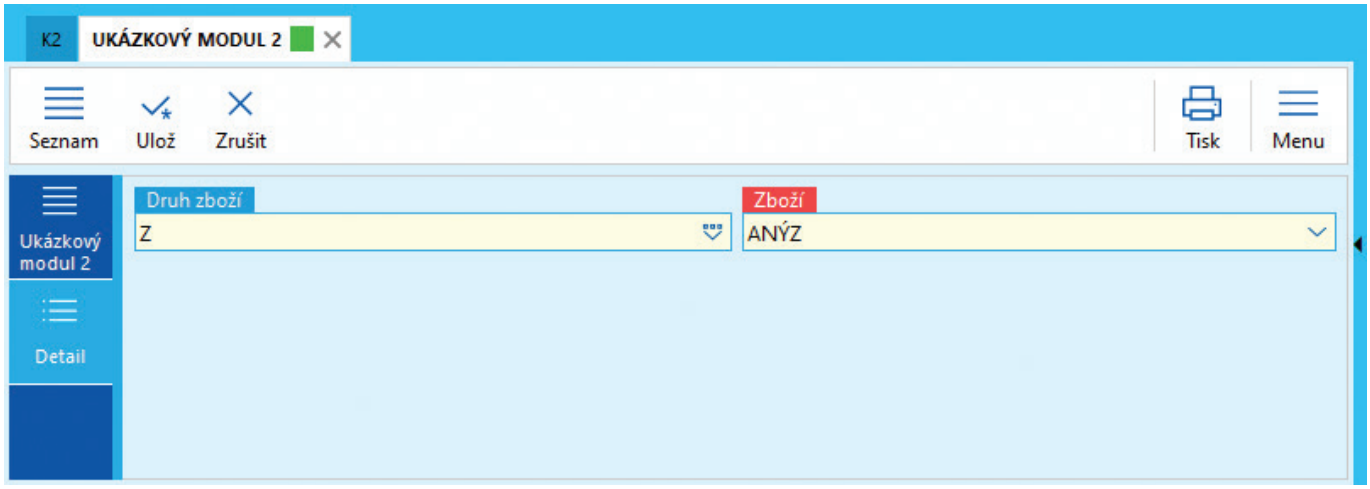
Po otevření vazby do modulu ***„Zboží“*** můžeme vidět, že se zde nachází pouze záznamy, které mají druh zboží typu ***„Z“***. Tuto informaci můžeme vidět například v modrém pruhu nad záhlavím tabulky, viz Obrázek 35 - Přenos z vazby – Filtrovat - filtrovaná vazba.



Obrázek 35 - Přenos z vazby – Filtrovat - filtrovaná vazba

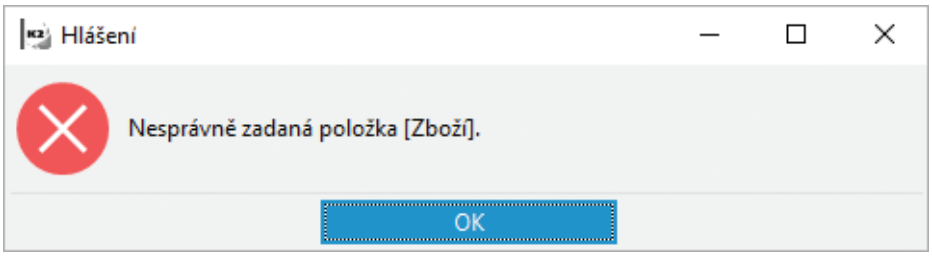
Kontrolovat
Nastavením operace ***„Kontrolovat“*** vynutíme kontrolu přenesené vazby. Pokud nebude odpovídat, pak je tato skutečnost zobrazena ve formuláři v podobě validační chyby, v případě, že bychom zkusili uložit záznam, pak by se zobrazilo chybové hlášení o nesprávně zadané hodnotě.

PŘÍKLAD: Využijeme předchozího příkladu na filtrování vazby. Pokud bychom v přenosu vazby zrušili ***„Filtrovat“*** a místo toho nastavili ***„Kontrolovat“***, pak v případě, že vybereme hodnotu zboží, která nebude mít stejnou hodnotu druhu zboží, jakou máme vybránu v poli ***„Druh zboží“***, se na vazbě do modulu ***„Zboží“*** zobrazí validační chyba, viz Obrázek 36 - Přenos z vazby – Kontrolovat.



Obrázek 36 - Přenos z vazby – Kontrolovat

Při pokusu o uložení pak dojde k chybovému hlášení, viz Obrázek 37 - Přenos z vazby – Kontrolovat - hlášení.



Obrázek 37 - Přenos z vazby – Kontrolovat - hlášení

Vynutit
V případě, že potřebujeme zajistit, aby byla použita vždy hodnota přenesená z vazby, nastavíme tuto operaci. Při ukládání záznamu a za předpokladu, že je hodnota v cílovém poli prázdná, se hodnota znovu z vazby přenese. Pokud potřebujeme zajistit přenos vždy, musíme navíc nastavit příznak ***„Přepisovat“***.

POZNÁMKA: Při vytvoření nového záznamu ***„Přenos z vazby“*** se automaticky označí 3 příznaky ***„Naplnit“***, ***„Přepisovat“*** a ***„Vyprázdnit“***. Tyto příznaky se automaticky označí, když nebude žádný příznak zatržen (po zmáčknutí OK). Tzn. vždy musí být označen aspoň jeden příznak.

Příznaky – Potvrdit přepis
Pokud máme nastaveno přepisování cílové hodnoty, pak můžeme nastavením tohoto příznaku vynutit chování, kdy v případě, že cílové pole obsahuje hodnotu, vždy se zobrazí dotaz, zda opravdu s přepisem souhlasíme.

POZNÁMKA: Tato vlastnost funguje pouze za předpokladu, že je zároveň nastaven příznak *„Přepisovat“*.

Přepisování
Operaci přepisování můžeme upřesnit následujícím nastavením.

Výchozí
Jako výchozí hodnota je nastavena *„Při editaci přepsat“*.

Je-li zdroj neprázdný
Cílové pole je přepsáno pouze za předpokladu, že je zdrojové pole neprázdné.

Při editaci přepsat
Cílové pole je přepsáno vždy.

Přepsat i při uložení
Cílové pole je přepsáno při operaci uložení. Tedy pokud bychom hodnotu z vazby přepsali, pak se stejně použije znovu hodnota z vazby.

Přiřazení
Tato vlastnost nastavuje, jakým způsobem se přiřadí hodnota z vazby do našeho cílového pole. Možnosti jsou následující.

Výchozí
Jako výchozí hodnota je nastavena *„ControlValue“*.

Value
Toto nastavení definuje jednoduchý přenos ***„čistě“*** hodnoty pole. To znamená, že se hodnota zapíše do pole a dále se nic neděje.

ControlValue
V případě nastavení *„ControlValue“* je logika jiná. Nastavení hodnoty proběhne přes funkci jádra K2, která se nazývá *„AsControlValue“*, která způsobí vyvolání dalších akcí nadefinovaných na poli.

PŘÍKLAD: Máme vytvořeno pole na datovém modulu, na kterém máme definovánu logiku na změnu jeho hodnoty. Tedy máme implementovánu funkci ***„Reagovat na změnu“***. V případě, že do tohoto pole vložíme hodnotu, spustí se tento kód jako reakce na změnu. To ale platí pouze za předpokladu, že hodnotu vkládáme přímo do tohoto pole ve formuláři. Pokud bychom hodnotu nastavili přenosem z vazby formou *„Value“*, kód by nebyl vyvolán. V takovém případě nastavíme přenos na hodnotu *„ControlValue“* a tím máme zajištěno, že po nastavení hodnoty, budou vyvolány i další funkce na tomto poli, v našem případě se jedná o ***„Reakce na změnu“***.

POZNÁMKA: Výchozí hodnotou pro *„Přiřazení“* je *„ControlValue“*.

4.2.3. PŘÍMÁ VAZBA

Vložení nového pole typu ***„přímá vazba“*** slouží také k vytvoření vazby do jiného datového modulu. Oproti vazbě typu cizí klíč je zde, ale zásadní rozdíl. S výběrem datového modulu, do kterého směřuje vazba, vybíráme i pole z vázaného datového modulu, přes které se propojení uskuteční.

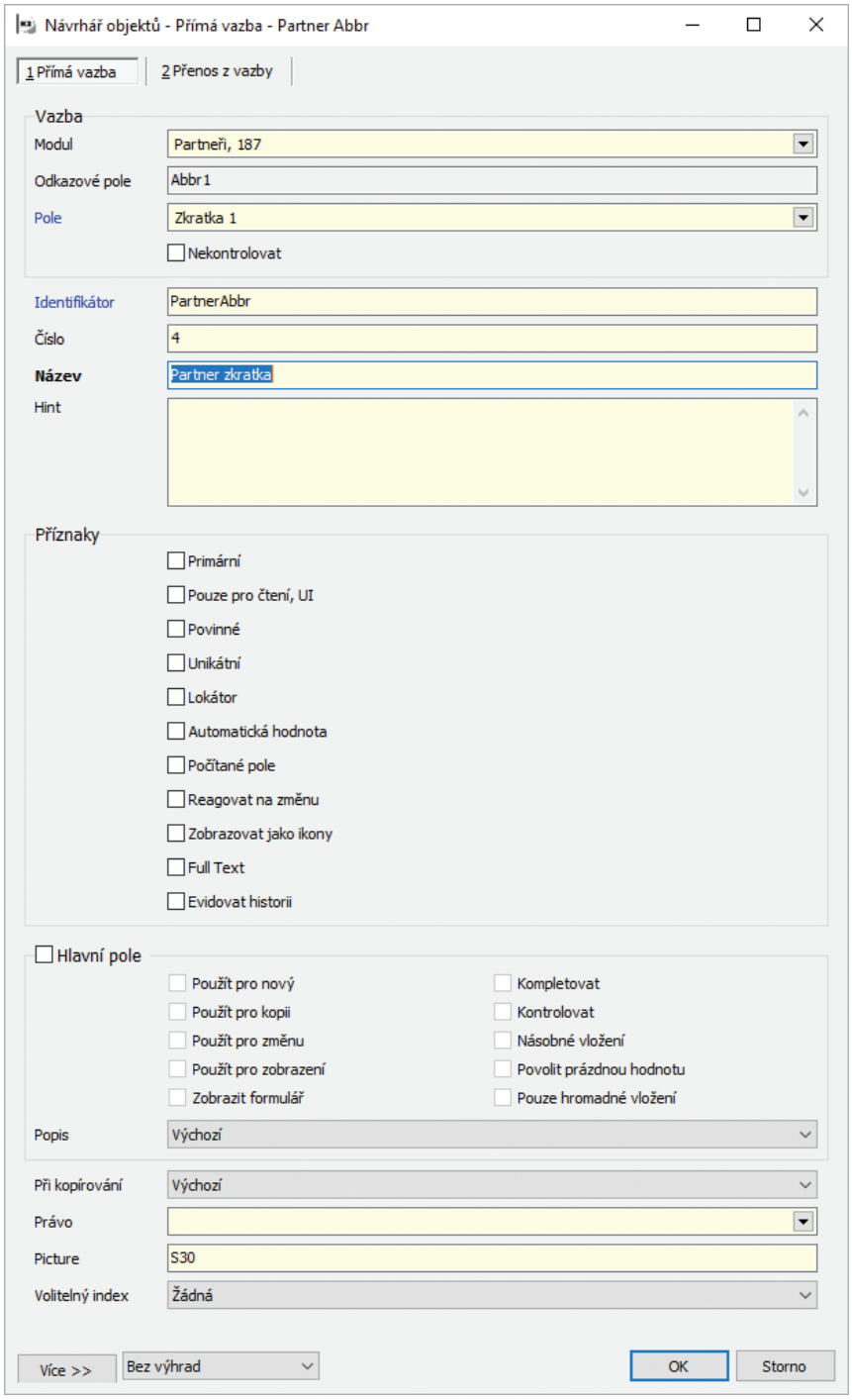
POZNÁMKA: Tento typ vazeb doporučujeme používat pouze v nezbytných případech. Díky tomu, že vazba není realizována skrz cizí klíč, může dojít k nekonzistencím. V implicitním nastavení, při vkládání a editaci záznamů, provádí K2 kontrolu, zda existuje hodnota, která byla zadána do vazby. Tedy v případě zakládání a editace záznamů, se nestane, že by došlo k nekonzistencím. Dále v textu si ukážeme, že i tuto kontrolu lze potlačit, což je důrazně nedoporučeno.

POZNÁMKA: Jako možné použití přímé vazby lze za předpokladu, pokud chceme mít vazbu na zkratku místo čísla (např. na zkratku zboží, a ne jeho číslo). Doporučuje se použít na tabulky s ***„malým id“*** jako je Zboží, Zákazník. Pomocí přímé vazby si můžeme vybrat, zda chceme vazbu na zkratku, nebo číslo (podle toho co je nabídnuto v poli ***„pole“***), tato vazba se nám potom promítne i v lookupu v datovém modulu (pokud zvolíme vazbu na číslo, uvidíme číslo, pokud na zkratku, uvidíme zkratku).

POZNÁMKA: Doporučená varianta definice vazeb, je využití vazby přes cizí klíč.
Pro představu si ještě představme následující příklad vzniku nekonzistence, který bude dostatečným příkladem toho, že by se měl tento typ vazby používat jen výjimečně.

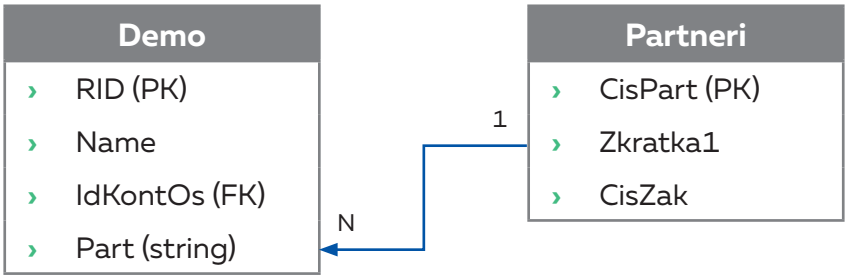
PŘÍKLAD: Vytvoříme přímou vazbu do datového modulu ***„Partneři“*** a jako vazební pole vybereme pole, které představuje zkratku 1 partnera, tedy ***„Abbr1“***. Vložíme nový záznam do datového modulu, na kterém je definována tato přímá vazba. Do vazby vložíme partnera se zkratkou ***„AB GROUP“***. V případě, že někdo změní na této kartě partnera zkratku 1, přestane naše vazba fungovat, protože nebude existovat karta partnera, na kterého byl skrz jeho zkratku 1 navázán.

Po výběru typu pole ***„Přímá vazba“*** se zobrazí formulář Obrázek 38 - Definice pole jako přímá vazba v datovém modulu. Jednotlivé části si nyní popíšeme.



Obrázek 38 - Definice pole jako přímá vazba v datovém modulu

Na Obrázek 39 - Schéma přímé vazby je schéma principu přímé vazby.



Obrázek 39 - Schéma přímé vazby

4.2.3.1. Vazba

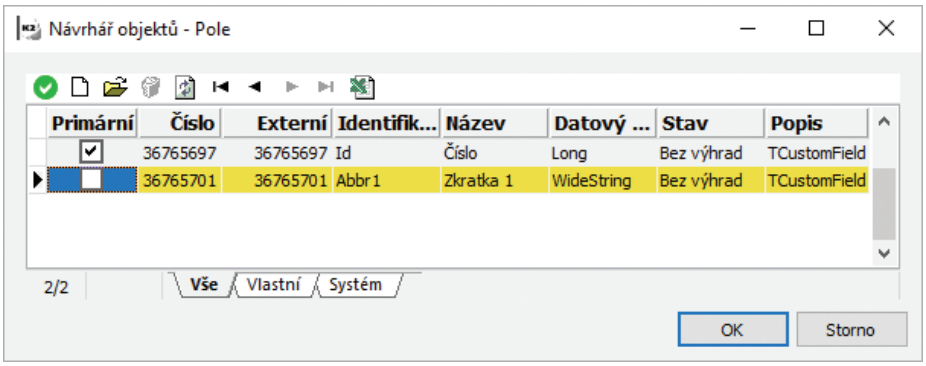
Protože pole typu ***„přímá vazba“*** je odkaz do jiného datového modulu, musíme nejprve definovat, do jakého datového modulu tato vazba bude směřovat a také nadefinovat další parametry této vazby.

Modul
Otevřeme rozbalovací nabídku s názvem ***„Modul“*** a vybereme ze seznamu potřebný datový modul, například ***„Partneři“***.

Odkazové pole
Má stejnou funkci jako v kapitole 4.2.2.1. Vazba – modul.

Pole
Na rozdíl od klasické vazby, která je provázána s datovým modulem pomocí primárního klíče vázaného modulu, u tohoto typu vazby se definuje, přes které pole se provázání uskuteční. Tedy v nabídce ***„Pole“*** otevřeme rozbalovací nabídku a vybereme pole ze seznamu, který je automaticky vyplněn dle polí z vybraného modulu. Seznam obsahuje pouze ta pole, přes které má smysl vytvářet vazbu. Jsou to pole, která jsou definovány jako unikátní a jejich klíč nesmí být složen z více polí. Tedy musí mít pouze jeden segment. Tato pole nabízí návrhář objektů automaticky. Nemůže se stát, že použijete pole, které by nesplňovalo tyto podmínky.

V předchozím kroku jsme vybrali modul ***„Partneři“***, zde tedy vybere pole z partnerů, přes které vazba bude realizována. Například vybereme pole ***„Zkratka1“*** viz Obrázek 40 - Výběr pole po přímou vazbu v datovém modulu.



Obrázek 40 - Výběr pole po přímou vazbu v datovém modulu

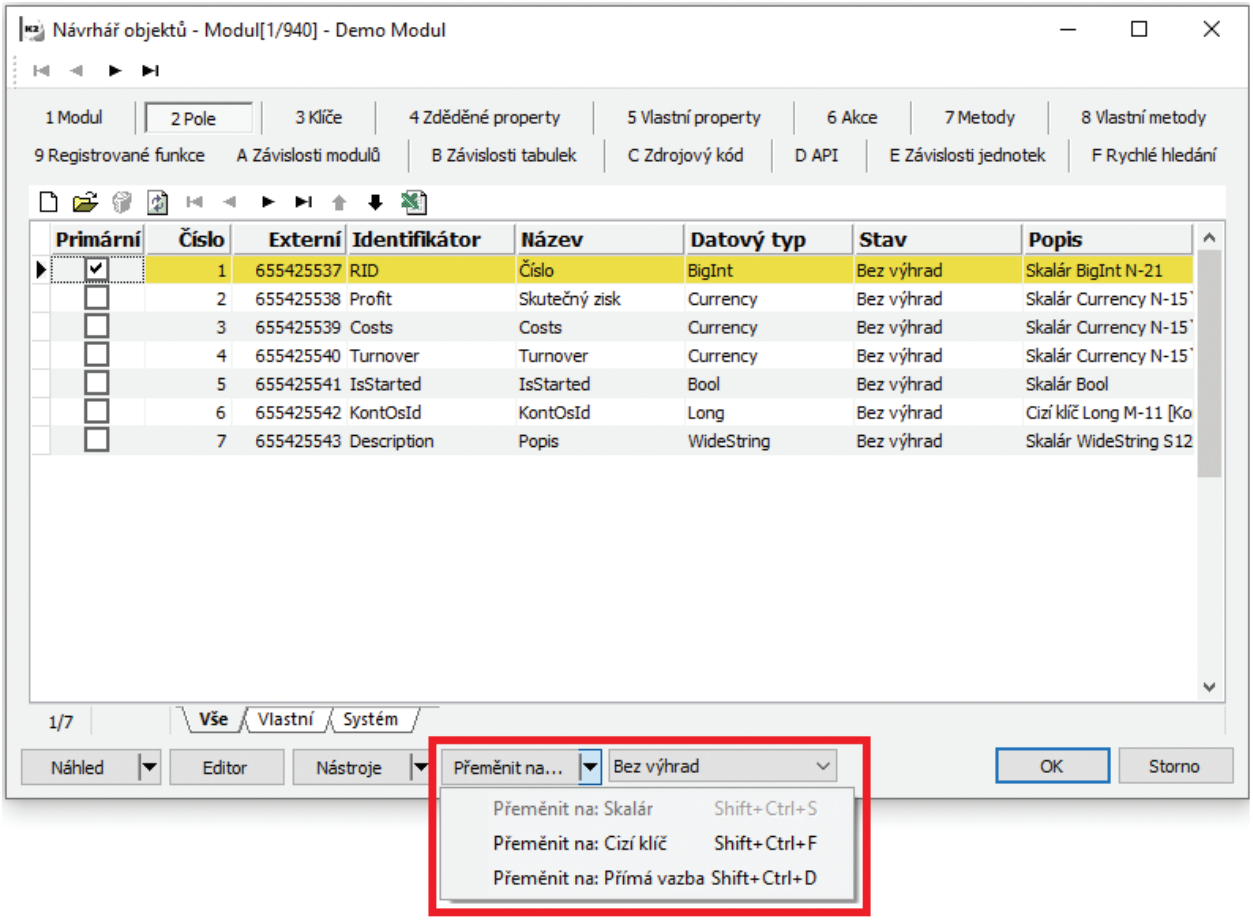
Nekontrolovat
Jak již bylo napsáno výše, implicitně je zadávaná hodnota do vazby kontrolovaná na existenci ve vázaném datovém modulu. Pokud hodnota neexistuje, nedovolí formulář uložení záznamu. Tuto kontrolu lze potlačit pomocí tohoto nastavení – ***„Nekontrolovat“***.

POZNÁMKA: Kontrola existence hodnot ve vázaném datovém modulu je důrazně nedoporučena, stejně jako použití přímé vazby, místo vazby přes cizí klíč.

Ostatní parametry vytvářeného pole jsou shodné s parametry popsanými v předchozí podkapitole 4.2.1. Skalár.

4.2.4. RYCHLÁ ZMĚNA TYPU POLE – FUNKCE „PŘEMĚNIT NA“

V dolní části formuláře v sekci ***„Pole“*** v datovém modulu existuje funkce ***„Přeměnit na“***, která slouží k rychlému převodu typů polí. Pomocí této funkce můžete převést pole typu ***„Skalár“*** na ***„Cizí klíč“*** nebo ***„Přímá vazba“***, stejně tak z typu ***„Cizí klíč“*** na ***„Skalár“*** apod. Po stisknutí tlačítka (případně pomocí klávesových zkratek) se nám nabídnou volby změny dle toho, na jakém typu pole aktuálně máme nastavenou pozici v seznamu, viz Obrázek 41 - Funkce "Přeměnit na" na poli datového modulu. Po stisknutí změny se zobrazí formulář s definicí pole, které je převedeno na požadovaný typ se zkopírovanými hodnotami z původního typu pole, dle toho, zda mají pro aktuální typ smysl.



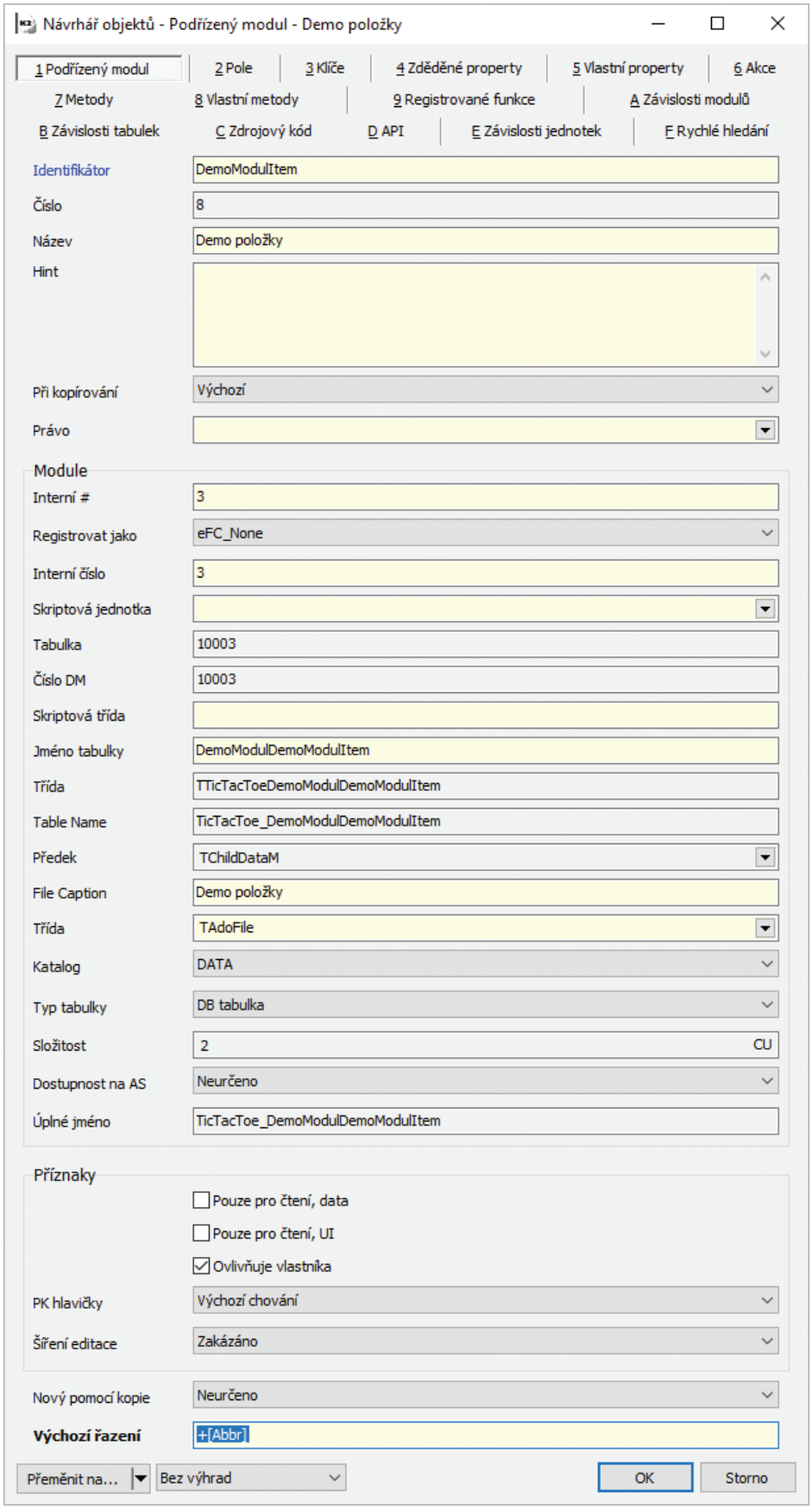
Obrázek 41 - Funkce "Přeměnit na" na poli datového modulu

POZNÁMKA: Jedno z využití této funkce je hlavně při importu databázových tabulek do návrháře objektů, viz kapitola 13.6. Import, kdy po importu, vznikají v tabulce všechna pole jako skalární. Pomocí přeměny, pak můžeme jednoduše změnit skalární pole na cizí klíče, případně přímé vazby dle požadavků.

4.2.5. POLOŽKY

Vložení nového pole typu ***„Položky“*** slouží k vytvoření nového modulu, který představuje položky, tedy podřízený modul.

Po výběru typu pole ***„Položky“*** se zobrazí formulář Obrázek 42 - Definice pole jako položka v datovém modulu. Téměř všechny parametry nového podřízeného modulu jsou stejné jako u klasického datového modulu, který popisujeme v aktuální kapitole, popíšeme si tedy jen ty částí, které se liší.



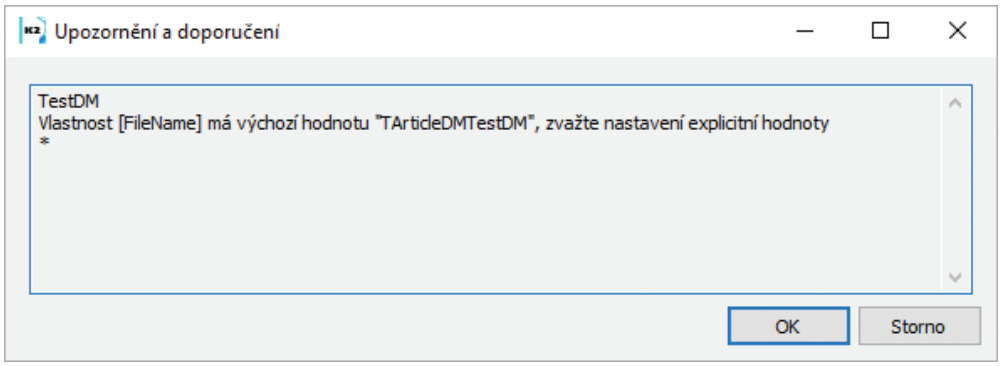
Obrázek 42 - Definice pole jako položka v datovém modulu

Všechny parametry nového podřízeného modulu vyplníme dle popisu výše. V následující části se zaměříme pouze na ty, které jsou k dispozici navíc oproti klasickému datovému modulu.

4.2.5.1. Základní sekce

V základní sekci se nachází pouze dvě vlastnosti, která je navíc oproti definici klasického datového modulu. Jedná se o ***„Nový pomocí kopie“*** a ***„Výchozí řazení“***. Vlastnost ***„Jméno tabulky“*** má v položkách navíc ještě funkcionalitu, kterou si popíšeme níže.

Jméno tabulky
Stejně jako v případě modulu daná vlastnost určuje název tabulky pro položkový modul. Navíc ale při akcích ***„Deploy“*** a ***„Balíček“*** probíhá kontrola, kdy se porovnává název tabulky s názvem, který automaticky vytvoří NO (týká se rozšíření standardních DM). V případě že se tyto názvy shodují, vyskočí po akcích ***„Deploy“*** nebo ***„Balíček“*** hlášení s upozorněním/doporučením, na změnu názvu Obrázek 43 - Protokol po "Deploy".



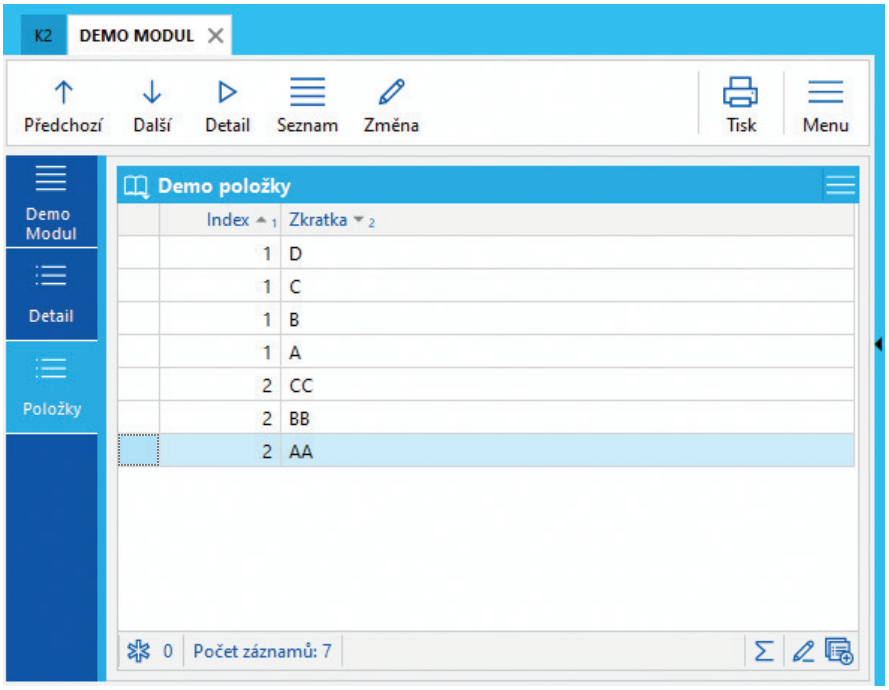
Obrázek 43 - Protokol po "Deploy"

Nový pomocí kopie
V případě, že máme nastaveno na hodnotu ***„Ano“***, pak při vkládání nového záznamu, tedy při vytváření nové položky, je nový záznam automaticky vyplněn hodnotami z poslední položky, v rámci nadřízeného modulu, samozřejmě v případě, že položka existuje. Tato vlastnost se týká pouze standardních formulářů. V univerzálních formulářích je nový záznam vytvářen ***„čistý“***, tedy opravdu odpovídající definici ***„nový“***.

Výchozí řazení
Pomocí tohoto nastavení je možné nadefinovat výchozí setřídění záznamů datového modulu. Nastavení se provádí pomocí definičního řetězce, který je ve tvaru například ***„+[Date]-[Zbo;Naz]“***. Každé pole v syntaxi musí být ohraničeno hranatými závorkami ***„+[název pole]“*** a musí začínat symbolem ***„+“*** pro vzestupné setřídění a ***„-“*** pro sestupné setřídění. V závorkách můžeme použít i vazbu, kde každé pole ve vazbě je odděleno symbolem ***„;“*** (středník). Jednotlivá pole nemusíme od sebe oddělovat.

PŘÍKLAD: V datovém modulu ***„Demo Modul“*** si založíme položky, které budou automaticky obsahovat pole ***„Master“***, které slouží k provázání s hlavičkou položek. Samozřejmě založíme pole ***„RID“***, který bude primárním klíčem. Dále pole ***„Index“*** typu číslo a ***„Abbr“*** typu string. Výchozí setřídění na položkách nastavíme na hodnotu ***„+[Index]-[Abbr]“***, viz Obrázek 44 - Příklad definice použití výchozího řazení položek v datovém modulu, tedy třídíme vzestupně dle indexu a dále sestupně dle zkratky. Po spuštění formuláře a zobrazení seznamu položek, můžeme vidět setřídění dle naší definice (samozřejmě si musím DM nakrmit nějakými daty), viz Obrázek 45 - Výsledek příkladu použití výchozího setřídění na položkách v datovém modulu.

Obrázek 44 - Příklad definice použití výchozího řazení položek v datovém modulu



Obrázek 45 - Výsledek příkladu použití výchozího setřídění na položkách v datovém modulu

4.2.5.2. Module

Zde jsou stejné vlastnosti jako při zakládání hlavičky datového modulu, nebudou zde již dále popsány, protože jejich popis naleznete v kapitole Modul 4.1. Řeč bude pouze o předkovi a třídě.

Předek
Vlastnost definující předka nového datového modulu není možné v tomto případě měnit. Je přednastavena na *„TChildDataM“*.

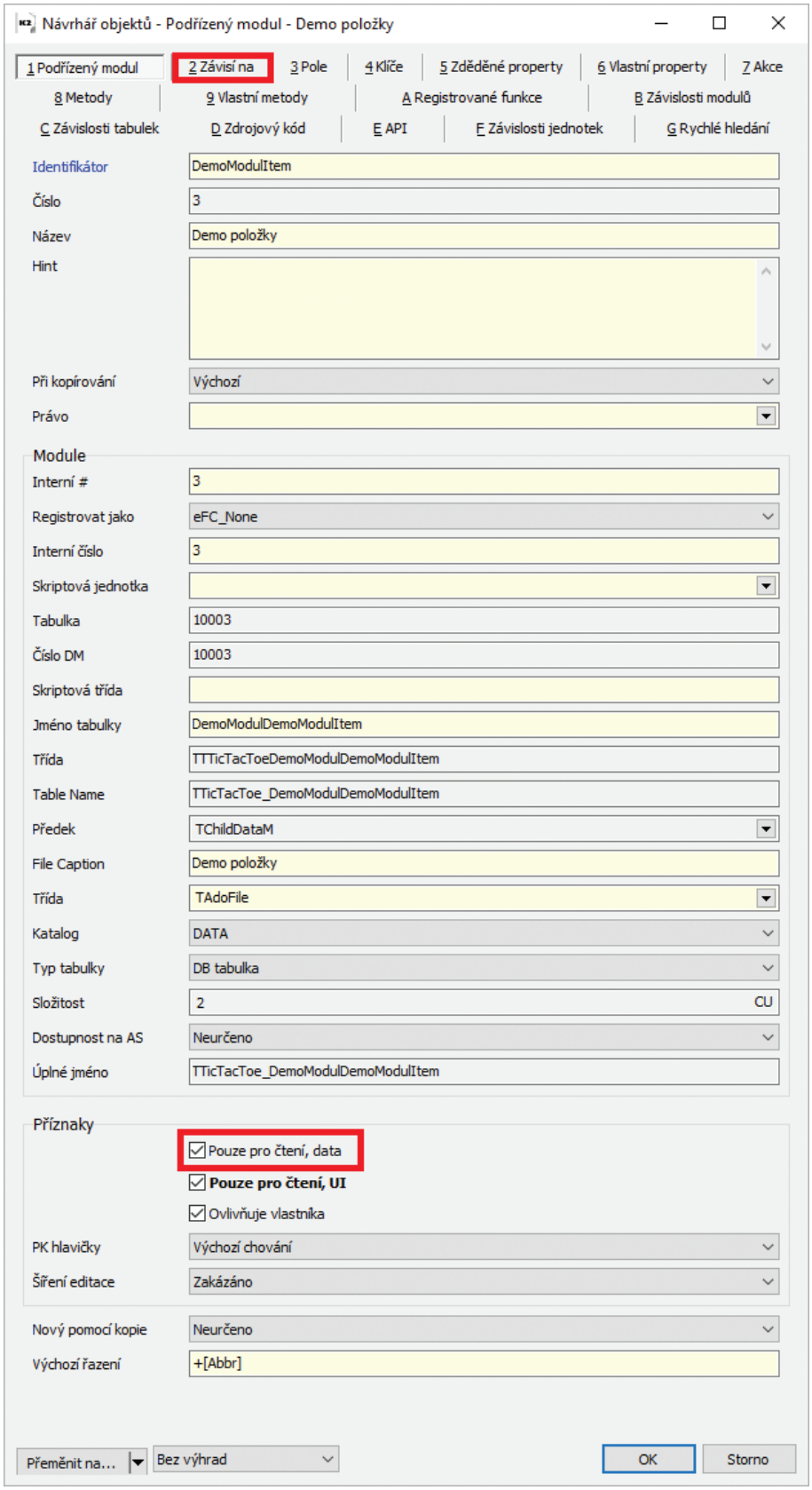
Třída
Vlastnost, která definuje, zda se jedná databázovou tabulku *„TAdoFile“*, nebo o paměťovou tabulku *„TMemFile“*. Pokud se zvolí paměťová tabulka, přibyde navíc vlastnost *„Editovatelné položky“* v sekci *„Příznaky“*, více si popíše v následující podkapitole.

4.2.5.3. Příznaky

Pouze pro čtení, data
Podřízený modul je tímto nastaven do režimu pouze pro čtení. Zápis není povolen ani na úrovni skriptu. Jedná se o striktní zákaz zápisu do podřízeného modulu.

POZNÁMKA: Pokud je tento příznak zatržen, objeví se nová záložka *„Závisí na“*, kde lze dát závislost položek (viz Obrázek 46 - Zobrazení záložky "Závisí na") na konkrétní pole, podobně jako je tomu u počítaného pole.

Záložka *"Závisí na"* nezmizí, pokud něco obsahuje, i když by být zobrazena neměla. Je to kvůli tomu, aby nedošlo k nechtěné ztrátě informací. Prvky z kolekce *"Závisí na"* je potřeba odstranit explicitně. Pokud je tam něco navíc/zbytečně kontrola v rámci operací *"Deploy"* a *"Balíček"* na to upozorní.



Obrázek 46 - Zobrazení záložky "Závisí na"

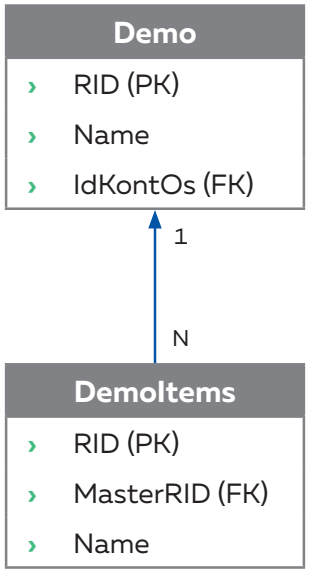
Pouze pro čtení, UI
Podřízený modul je tímto nastaven do režimu pouze pro čtení. Zápis není povolen pouze z prostředí formulářů, kde toto omezení hlídá validace vstupů na formuláři.

POZNÁMKA: Pokud nastavíme ***„Pouze pro čtení, data“***, automaticky se nastavuje i příznak ***„Pouze pro čtení, UI“***.

Ostatní záložky pro definici polí, klíčů apod., jsou popsány v této kapitole, která popisuje obecně tvorbu datového modulu. V případě, že se bude logika v některých místech lišit, čtenář bude na tuto skutečnost upozorněn.

POZNÁMKA: V případě založení nového podřízeného datového modulu se automaticky pro modul generuje pole s názvem ***„Master“***, které je odkazem na nadřízený modul a musí v modulu být k zajištění funkčnosti. K tomuto poli se automaticky generuje i klíč.

Na Obrázek 47 - Schéma struktury hlavička - položky je zobrazeno schéma popisující napojení položkového modulu.



Obrázek 47 - Schéma struktury hlavička - položky

Editovatelné položky
Tento příznak se objeví pouze tehdy, pokud se jedná o paměťovou tabulku ***„TMemFile“*** ve vlastnosti ***„Třída“***. Property nám určuje, zda jsou položky v paměťové tabulce editovatelné, či nikoliv. Pokud zatrhneme, že ***„Ano“***, lze položky uživatelsky přidávat, mazat a editovat.

POZNÁMKA: Nejčastěji se setkáme ale s volnou nezatržení této vlastnosti, protože paměťové položky se používají především pro předky ***„TOneRecordDM“*** a ***„TMTBaseDM“***, které jsou taktéž paměťové.

Ovlivňuje vlastníka
Vlastnost určuje, zda je možné vkládat a editovat položky, v režimu prohlížení hlavičky. Tedy pokud nemáme nastaveno, můžeme vkládat a editovat záznamy, aniž bychom museli uvést hlavičku do režimu ***„změna“*** nebo ***„nový záznam“***. Klasickou variantou použití je práce s položkami typu ***„Poznámky“***.

V případě nastavení tohoto příznaku, není možné pracovat s položkami, bez uvedení hlavičky do stavu ***„změna“*** nebo ***„nový záznam“***. Využívá se v případě, že je na základě změn položkových záznamů, prováděna změna i v hlavičce.

PK hlavičky
Jedná se o vynucení speciálního režimu, který zajišťuje, že dojde k vyvolání mechanismu načtení položek jen v případě, že existuje hodnota primárního klíče. Každý modul si hodnotu této vlastnosti řídí automaticky sám dle typu použití. Například nevlastněné položky připojené k datovému modulu, který je předkem speciálního případu ***„TOneRecordDM“*** musí mít nastavenu hodnotu na ***„Není třeba pro načtení“*** jinak by nedošlo nikdy k načtení těchto položek. Je to z toho důvodu, že záznamy v tomto modulu jsou vždy ve stavu ***„nový“***, tedy nemáme ještě hodnotu primárního klíče. V tomto případě je tento stav zároveň výchozím.

Dostupné hodnoty jsou následující:

Výchozí chování
Hodnotu si určuje modul dle svého nejlepšího uvážení – v závislosti na předkovi, typu položek apod.

Není potřeba pro načtení
Načtení položek není závislé na existenci hodnoty primárního klíče. Modul se je tedy pokouší načítat i v případě například nového záznamu.

Hodnota je potřeba pro načtení
Položky se načítají pouze v případě, že je vyplněna hodnota primárního klíče.

POZNÁMKA: Je doporučeno hodnotu nechat nastavenu na výchozí. Explicitní určení je vhodné jen při velmi specifických situacích.

Šíření editace
Vlastnost, která říká, jakým způsobem má být nastaveno šíření editace. Může nabývat hodnot ***„Zakázáno“***, ***„Vždy“*** nebo ***„Podle hlavičky“***. Daná vlastnost je závislá na vlastnosti ***„Ovlivňuje vlastníka“***, pokud je tato vlastnost zatržená.

Zakázáno
Šíření je zakázáno.

Vždy
Znamená, že lze v uživatelském rozhraní přidávat položky kdykoliv, v nezávislosti na to, zda je hlavička v editaci. Je to obdoba vlastnosti „Ovlivňuje vlastníka“, ale v tomto případě jsou položky neustále ve změně.

Podle hlavičky
Znamená, že položky se řídí podle hlavičky, je-li hlavička ve změně, tak položky také.

4.2.6. NEVLASTNĚNÉ POLOŽKY

Vložení nového pole typu ***„Nevlastněné položky“*** slouží k vytvoření propojení existujícího modulu s aktuálním, tak jako by připojovaný modul byl podřízeným modulem. Dalo by se říci, že se jedná o vazbu do jiného modulu, který je v tomto případě použit jako položky, které nejsou vlastněny, tedy nejsou podřízeným modulem, daného modulu. Proto, aby mohla vazba dobře fungovat, musíme vazbu pečlivě nadefinovat. V datovém modulu, který připojujeme jako ***„nevlastněné položky“***, musí existovat pole, které odkazuje do modulu, který bude jeho nadřízeným modulem. Před popisem formuláře je tedy dobré uvést příklad použití takového typu pole.

PŘÍKLAD: Typickým příkladem může být stav, kdy máme nějaký doklad, třeba ***„Zakázky“***, víme, že v zakázkách je odkaz do datového modulu ***„Odběratelé“***. Můžeme pak na datovém modulu odběratelů nadefinovat nevlastněné položky typu zakázka, které propojíme pomocí pole na odběratele. Můžeme si pak jednoduše zobrazit všechny zakázky u každého odběratele.

Popíšeme si definici ***„nevlastněných položek“*** ve formuláři a následně se ještě vrátíme k uvedenému příkladu, který si ještě popíšeme na obrázcích formulářů, jak takovou konkrétní vazbu nadefinovat.

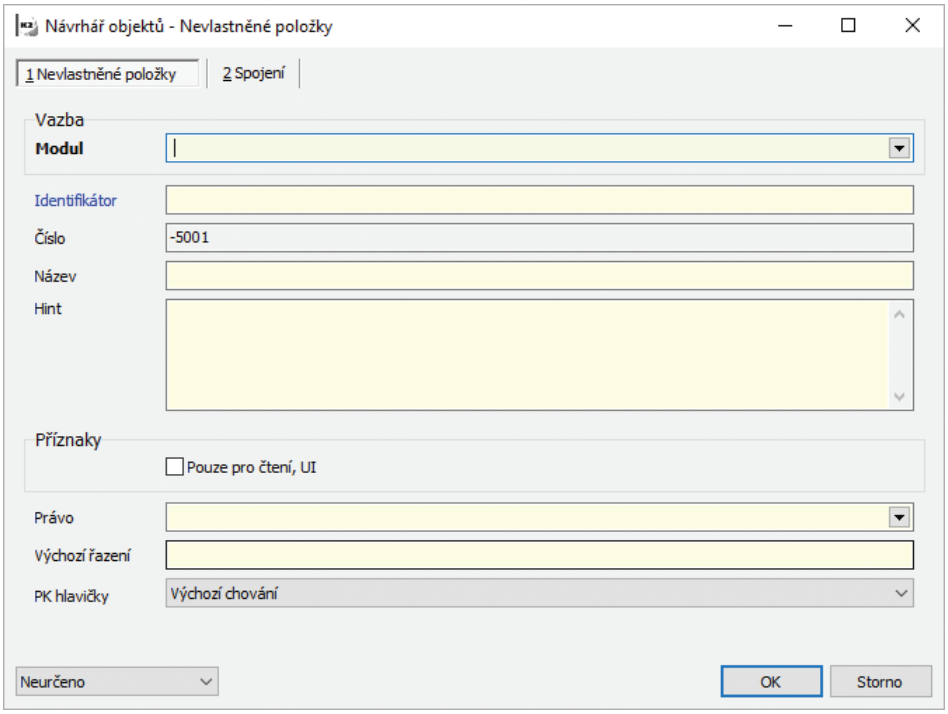
Po výběru typu pole ***„Nevlastněné položky“*** se zobrazí formulář Obrázek 48 - Definice pole typu nevlastněné položky v datovém modulu – záložka 1. Jak můžeme vidět, obsahuje dvě záložky ***„Nevlastněné položky“*** a ***„Spojení“***. V první záložce se definují základní vlastnosti pole, v druhé pak definice propojení mezi spojovanými moduly. Některé vlastnosti definovaného pole jsou opět stejné jako v popsaných částech, budeme se tedy věnovat pouze těm, které jsou specifické pro nevlastněné položky.

4.2.6.1. Záložka „Nevlastněné položky“

Modul

Protože pole typu ***„Nevlastněné položky“*** je definován odkaz do jiného datového modulu, musíme nejprve definovat, do kterého bude vazba směřovat. Otevřeme rozbalovací nabídku s názvem ***„Modul“*** a vybereme ze seznamu potřebný datový modul.

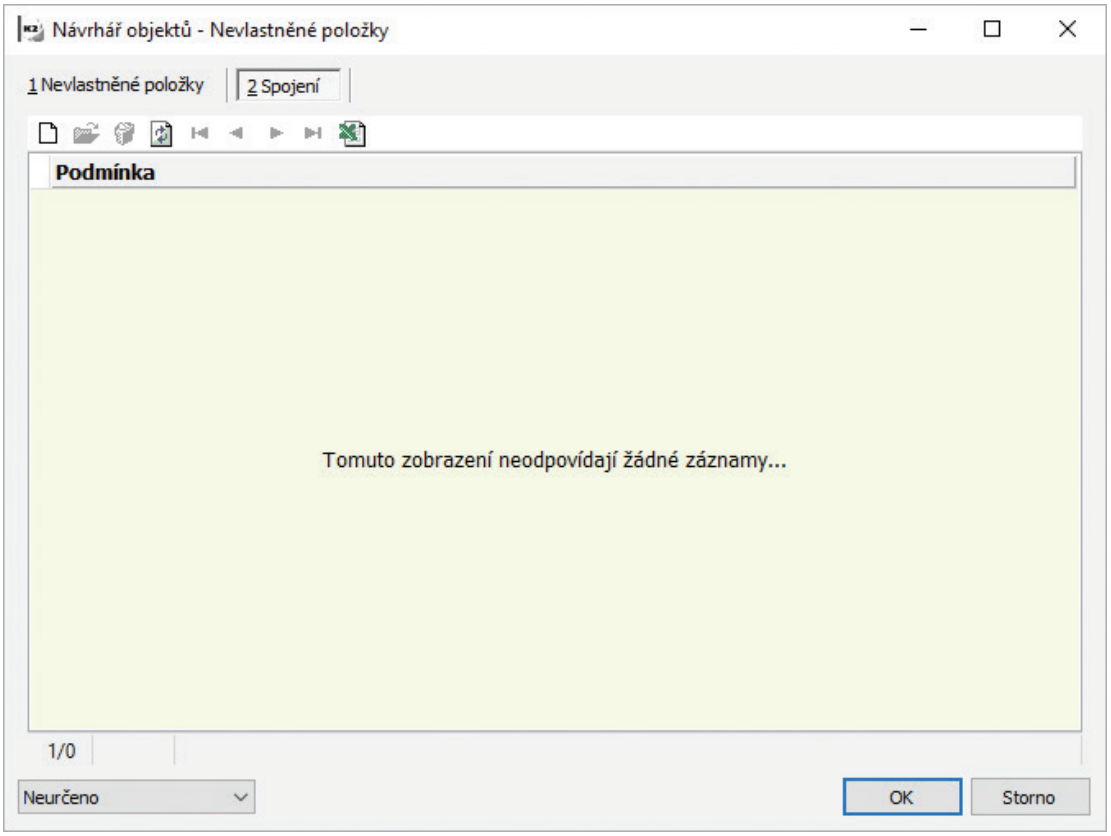
Všechna ostatní nastavení jsou shodná jako u standardní definice podřízeného modulu. Není tedy nutné je dále popisovat.



Obrázek 48 - Definice pole typu nevlastněné položky v datovém modulu – záložka 1

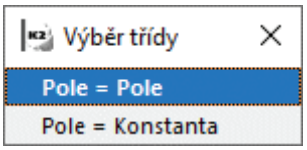
4.2.6.2. Záložka „Spojení“

Jak je vidět na obrázku Obrázek 49 - Definice pole typu nevlastněné položky v datovém modulu – záložka 2, na záložce ***„Spojení“*** je k dispozici seznam nadefinovaných vazeb mezi nadřízeným a podřízeným modulem. Protože vazba může být definována přes více polí, je zde uveden seznam. V jednoduchých případech si vystačíme s jednoduchou podmínkou na jedno pole.



Obrázek 49 - Definice pole typu nevlastněné položky v datovém modulu – záložka 2

Spojení nadefinujeme pomocí tlačítka prázdného listu v nabídce nad seznamem nebo pomocí stisku klávesy Insert. Zobrazí se nám formulář pro výběr typu spojení, viz Obrázek 50 - Výběr typu spojení v definici nevlastněných položek datového modulu.



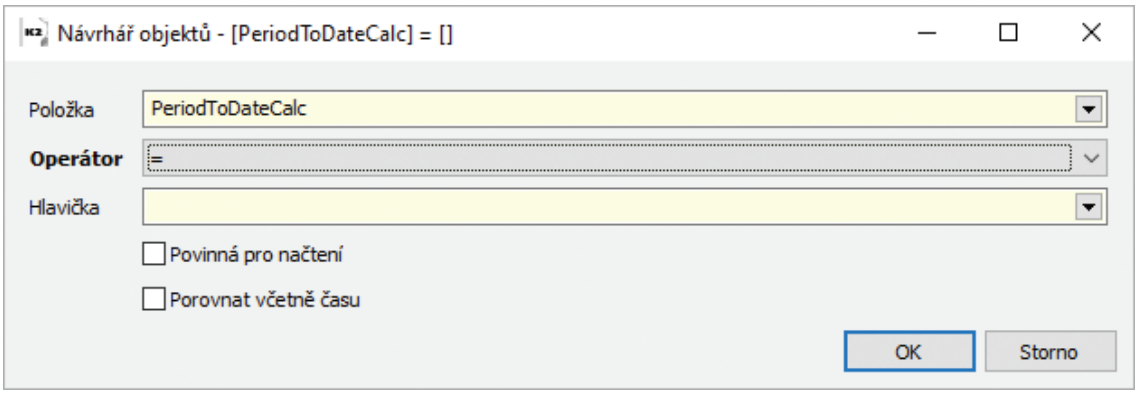
Obrázek 50 - Výběr typu spojení v definici nevlastněných položek datového modulu

V případě, že vazba je definována na obou stranách pomocí pole, pak vybíráme první variantu. Zobrazí se nám formulář Obrázek 51 - Definice spojení typu pole - pole pro nevlastněné položky v datovém modulu. První údaj, který musíme vyplnit je pole, které se nachází v podřízeném datovém modulu, tedy v budoucích nevlastněných položkách. Dále vybereme operátor, který definuje vztah mezi jednotlivými poli. Posledním údajem je pole, které je v nadřízeném datovém modulu. Spojení je pak definováno na základě definice těchto dvou polí a operátorem porovnání hodnoty mezi těmito poli. Definice umožňuje nadefinovat i operátory typu ***„větší“***, ***„menší“*** apod.

Povinná pro načtení
V případě, že potřebujeme omezit načítání položek pouze pro případ, kdy jsou pole v podmínce vyplněna, nastavíme na spojení povinnost.

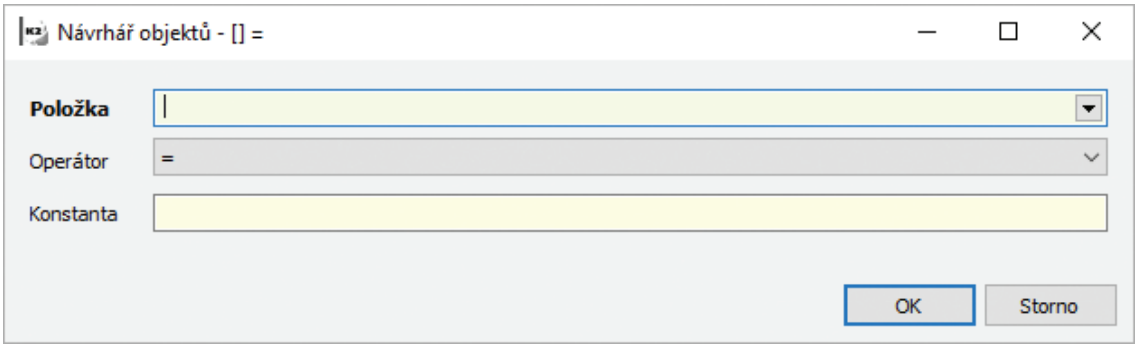
Obrázek 51 - Definice spojení typu pole - pole pro nevlastněné položky v datovém modulu

Porovnat včetně času
Tato možnost se objeví tehdy (Obrázek 52 - Definice spojení typu pole - pole (s typem pole TDateTime)), pokud je aspoň jedno pole typu TDateTime. Znamená to že se pole mají porovnávat včetně času.



Obrázek 52 - Definice spojení typu pole - pole (s typem pole TDateTime)

Druhou variantou, je pak definice spojení kde se připojí s podřízeného modulu ty záznamy, které ve vybraném poli obsahují konstantu. Kdy porovnání konstanty definuje operátor. Tedy například hodnota v poli s názvem ***„A“*** je ***„větší“*** než ***„1000“***. Formulář je vidět na obrázku Obrázek 53 - Definice spojení typu pole – konstanta pro nevlastněné položky v datovém modulu. V prvním údaji ***„Položka“*** vybíráme pole s podřízeného modulu, v ***„Operátor“*** vybíráme podmínku pro vyhodnocení a nakonec napíšeme do ***„Konstanta“*** údaj, proti, kterému se vyhodnocuje operátor.

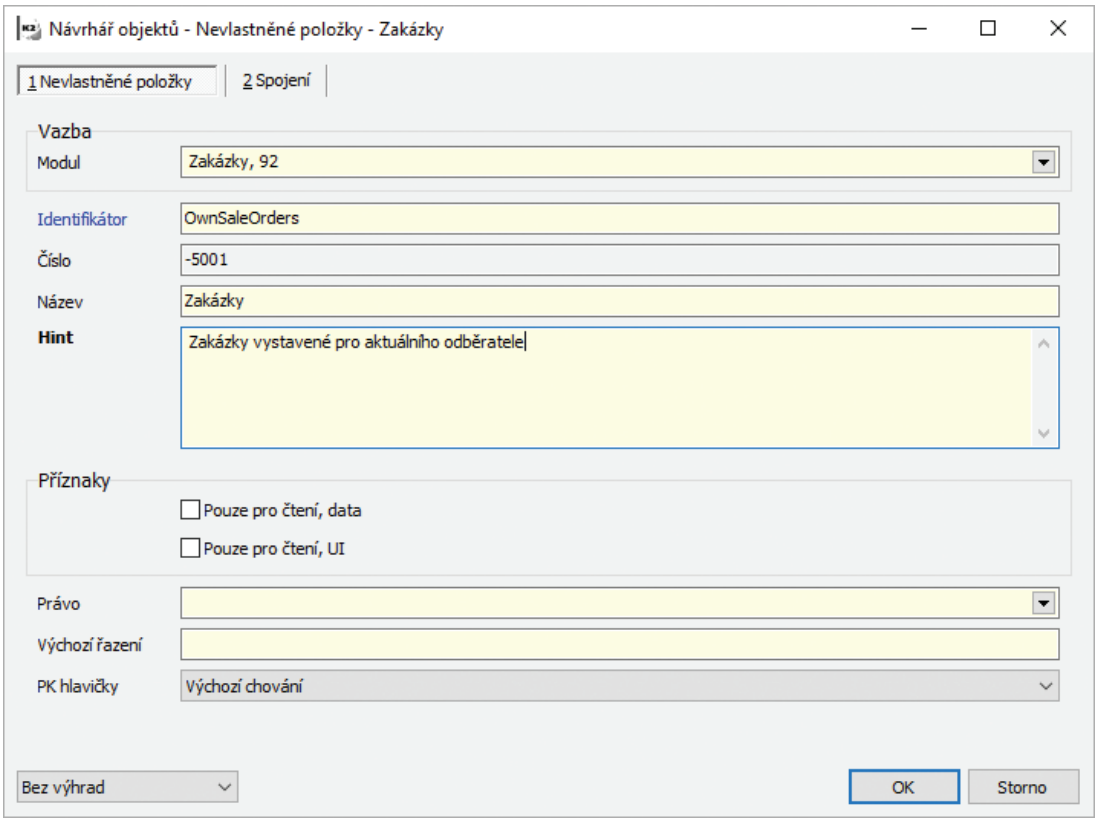


Obrázek 53 - Definice spojení typu pole – konstanta pro nevlastněné položky v datovém modulu

Pokud je vazba definována více poli nebo více podmínkami, pak je potřeba přidat do seznamu další podmínku a nastavit jednotlivá pole a operátor, případně pole a konstantu.

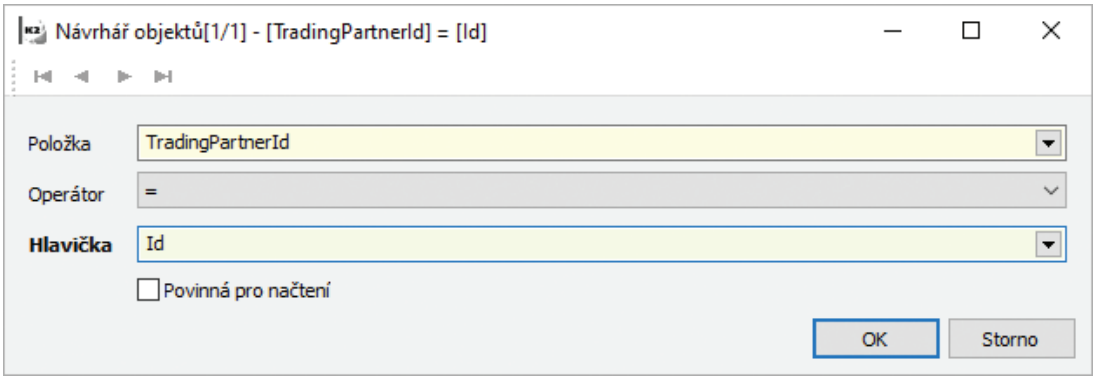
PŘÍKLAD: Výše máme popsaný příklad, kdy jsme chtěli připojit ke každému odběrateli jeho zakázky. Potřebujeme tedy v datovém modulu ***„Odběratelé“*** nadefinovat nevlastněné položky typu ***„Zakázky“***.

Otevřeme si, v seznamu všech datových modulů v návrháři objektů, datový modul ***„Odběratelé“***. Přepneme se na záložku ***„Pole“*** a stiskneme ikonu nového záznamu nebo stiskneme tlačítko Insert. Vybereme variantu ***„Nevlastněné položky“***. Zobrazí se nám formulář Obrázek 54 - Příklad definice nevlastněných položek "Odběratel" - "Zakázky", kde vyplníme vazbu do modulu ***„Zakázky“***, popíšeme identifikátor, název a nápovědu v případě najetí myší.



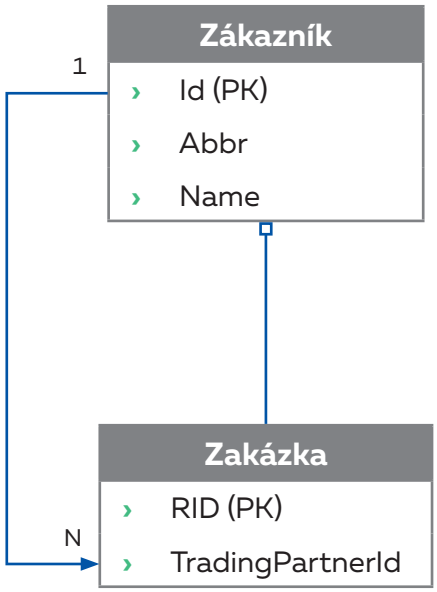
Obrázek 54 - Příklad definice nevlastněných položek "Odběratel" - "Zakázky"

Následně se přepneme na záložku ***„Spojení“*** viz obrázek Obrázek 55 - Příklad definice spojení pro nevlastněné položky "Odběratel" - "Zakázky". V poli ***„Položka“*** vybereme z podřízeného datového modulu ***„Zakázky“*** pole, které obsahuje vazbu do datového modulu ***„Odběratelé“***, tedy ***„TradingPartnerId“***. Jako operátor ponecháme ***„=“***, protože potřebujeme jednoduchou podmínku. Posledním údajem v definici spojení je ***„Hlavička“***, kdy vybíráme pole z nadřízeného modulu, pomocí kterého se moduly propojí. V našem případě potřebujeme primární klíč, tedy číslo odběratele ***„Id“***.



Obrázek 55 - Příklad definice spojení pro nevlastněné položky "Odběratel" - "Zakázky"

Na Obrázek 56 - Schéma propojení nevlastněných položek můžeme vidět schéma propojení nevlastních položek.



Obrázek 56 - Schéma propojení nevlastněných položek

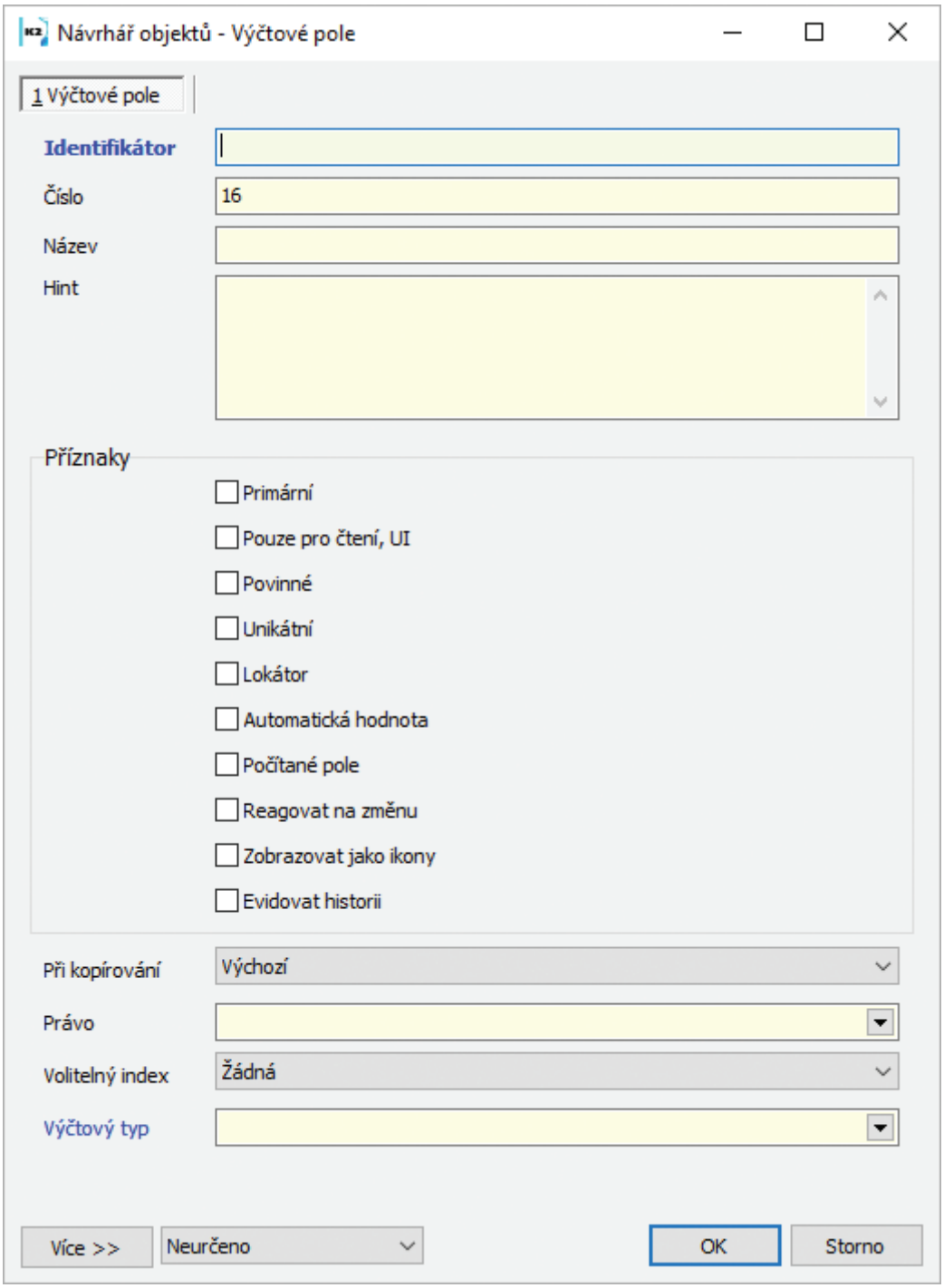
Po tom, co takovou úpravu promítneme do systému K2 pomocí akce ***„Deploy“***, která bude popsána v kapitole 13.1. „Deploy" – aplikování úprav do K2, můžeme ve formuláři nadefinovat zobrazení seznamu zakázek pro každého odběratele, tak že se seznam napojí na nové pole v datovém modulu ***„Odběratelé“***, které představuje nevlastněné položky.

POZNÁMKA: V příkladu jsme si ukázali rozšíření existujícího datového modulu ***„Odběratelé“***, který je součástí K2 již od instalace, tedy jedná se o tovární datový modul. Pomocí návrháře objektů nemůžeme upravovat a měnit tovární datové moduly, pouze můžeme definovat těmto modulům nové ***„Položky“*** a ***„Nevlastněné položky“***, tak jak tomu bylo v tomto příkladu, nebo také přidat ***„Počítané pole“***.

4.2.7. VÝČET HODNOT

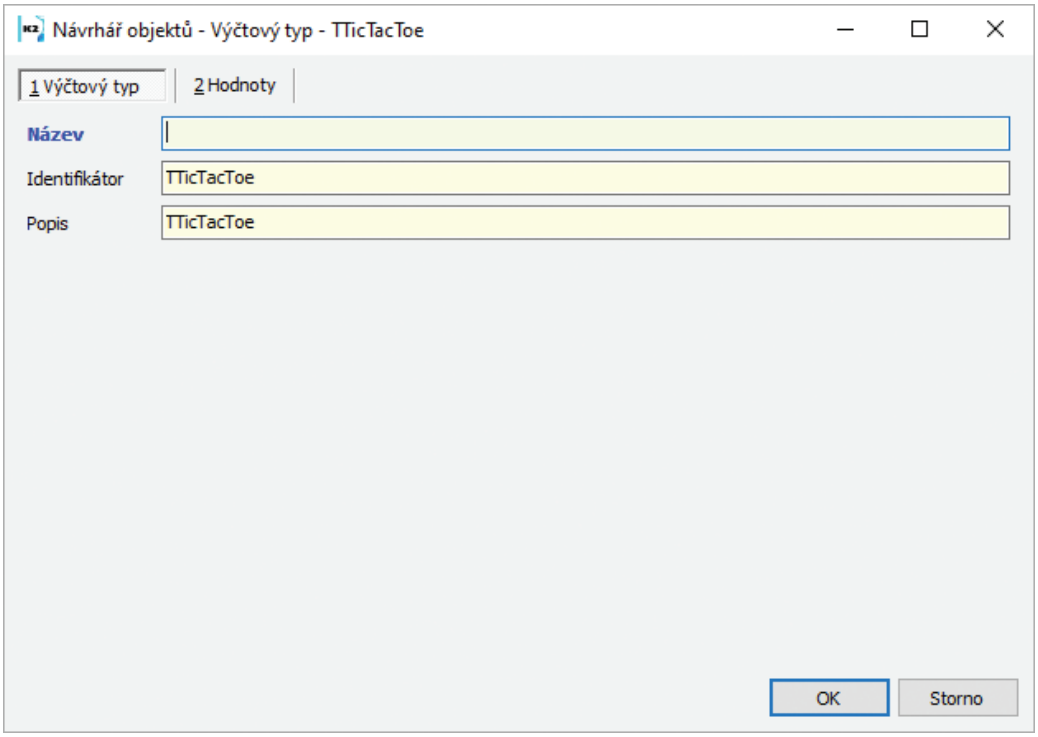
Jako poslední možností vytvoření pole je výčet hodnot.

Toto pole slouží k tomu, pokud chceme mít v určitém poli na výběr z předem definujících hodnot. Po výběru tohoto typu se nám zobrazí formulář, který je podobný typu pole skalár. Chybí zde výběr datového typu, protože výčet hodnot má již svůj typ ***„enum"***. Navíc je ale zde vlastnost Výčtový typ (viz Obrázek 57 - Vlastnosti pole "Výčtový typ").



Obrázek 57 - Vlastnosti pole "Výčtový typ"

Po rozkliknutí se nám zobrazí seznam pro námi definované výčtové typy. Pomocí tlačítka Nový nebo klávesy Insert založíme nový typ. Následně se nám zobrazí formulář, který si níže popíšeme.



Obrázek 58 - Definice pro výčtový typ

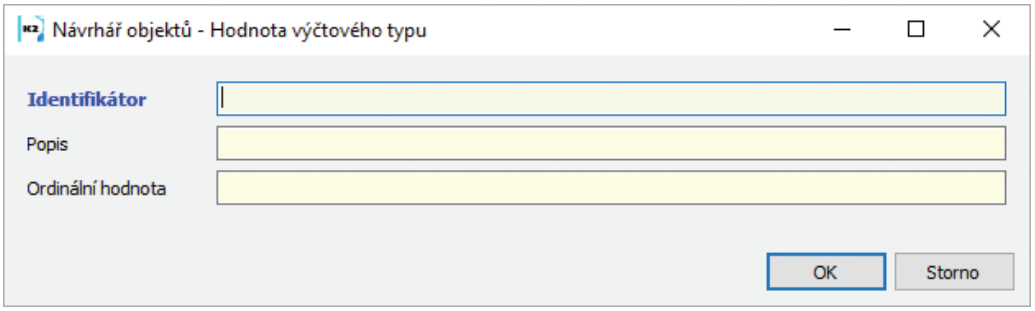
Záložka „Výčtový typ“

Název - Jedná se o název výčtového typu a musí být jedinečný v rámci identifikátoru.

Identifikátor – Automaticky se tvoří z jmenného prostoru. I když se jedná o editovatelnou informaci, doporučuje ponechat tak, jak nám jej NO připraví.

Popis – Jedná se o popis výčtového typu.

Záložka „Hodnoty“
Na této záložce se definují jednotlivé hodnoty výčtového typu. Novou hodnotu založíme buď to pomocí tlačítka Nový nebo klávesy Insert. Objeví se formulář pro zadání identifikátoru a popisu, jak můžeme vidět na obrázku Obrázek 59 - Vytvoření hodnoty pro výčtový typ



Obrázek 59 - Vytvoření hodnoty pro výčtový typ

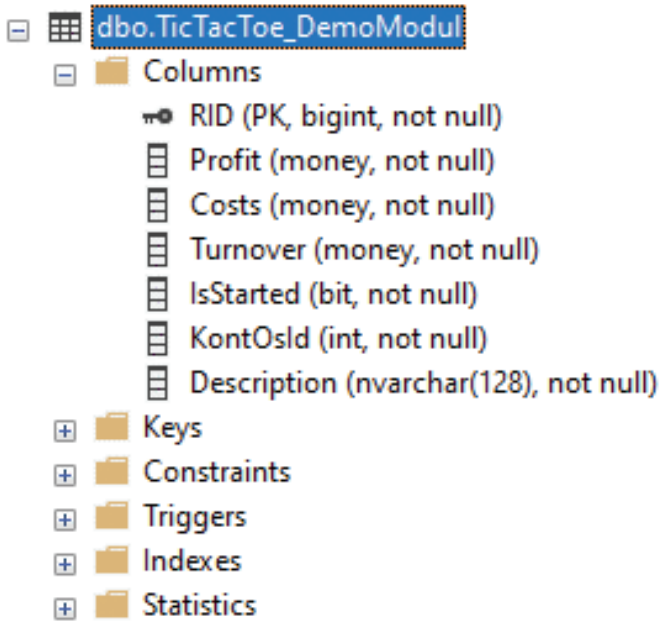
Identifikátor – jedinečný identifikátor v rámci daného výčtového typu

Popis – popis hodnoty

Ordinální hodnota – tato hodnota se vyplní automaticky, nemusí se vyplňovat.

4.2.8. VÝSTUP V DATABÁZI

V případě, že se jedná o datový modul, pro který existuje databázová tabulka, po tom co se změny publikují do K2 (více v 13.1. „Deploy" – aplikování úprav do K2), vznikne v databázi tabulka, která obsahuje sloupce definované pomocí předchozího návodu. Například v případě databáze typu MS SQL je vidět vytvořená tabulka na Obrázek 60 - Vytvořená databázová tabulka v MS SQL serveru dle definice datového modulu pro datový modul ***„Demo modul"*** vytvořený v návrháři objektů se jmenným prostorem definovaným jako ***„TicTacToe"***.



Obrázek 60 - Vytvořená databázová tabulka v MS SQL serveru dle definice datového modulu

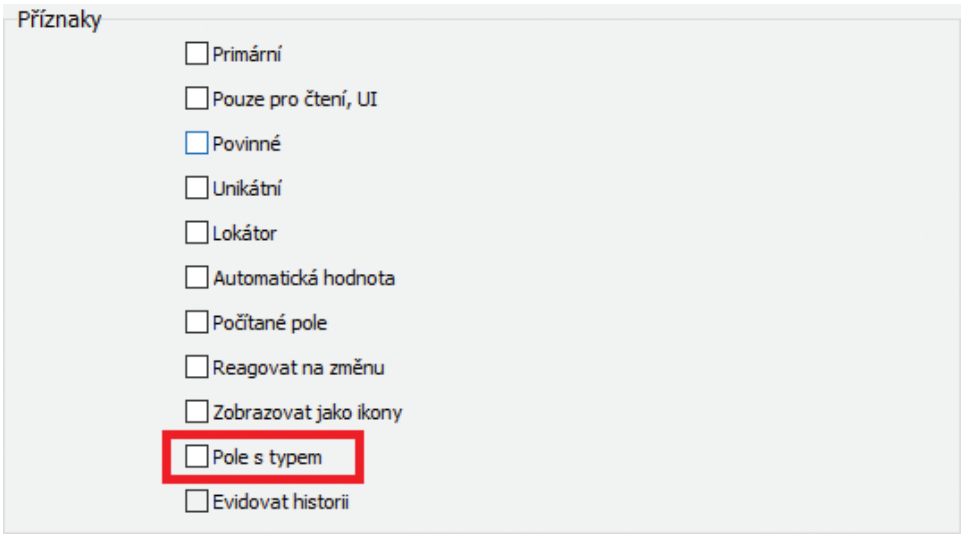
4.3. ROZDÍLY V NASTAVENÍ POLÍ DLE PŘEDKA DATOVÉHO MODULU

V případě, že použijeme jiného předka datového modulu, než základní variantu ***„TBaseDataM"***, je potřeba dbát na specifická nastavení, která mohou potomci některých datových modulů vyžadovat. V následující části textu se budeme věnovat potomkům, kteří právě takovou speciální péči vyžadují.

4.3.1. NASTAVENÍ POLÍ U PŘEDKA TYPU TTYPEDATAM

V případě, že zvolíme variantu předka ***„TTypeDataM"*** je potřeba dbát na nastavení, které nám zajistí správné chování typového číselníku. Konkrétně se jedná o dvě nutné podmínky, které musí být splněny na úrovní definice polí v tomto datovém modulu.

První podmínkou je, že musí existovat na datovém modulu pole, které má zatržen příznak ***„Pole s typem"*** viz Obrázek 61 - Definice pole jako typové pro datový modul jako typový číselník. Tato volba se objeví v sekci ***„Příznaky"*** ve formuláři pro definici pole, pouze v případě, že máme jako předka datového modulu zvolený typ ***„TTypeDataM"***. Návrhář objektů nedovolí uživateli uložit modul s touto zatrhnout volbu na více polích v rámci datového modulu. Typicky se jedná o pole, které si například nazveme ***„TypeId"***, a u tohoto pole zatrhneme tuto volbu.



Obrázek 61 - Definice pole jako typové pro datový modul jako typový číselník

Druhou podmínkou je, že musí existovat pole, které bude sloužit k uložení zkratky. Toto pole musí být nastaveno jako ***„Unikátní“***. Typickým názvem pro pole se zkratkou je ***„Abbr“***.

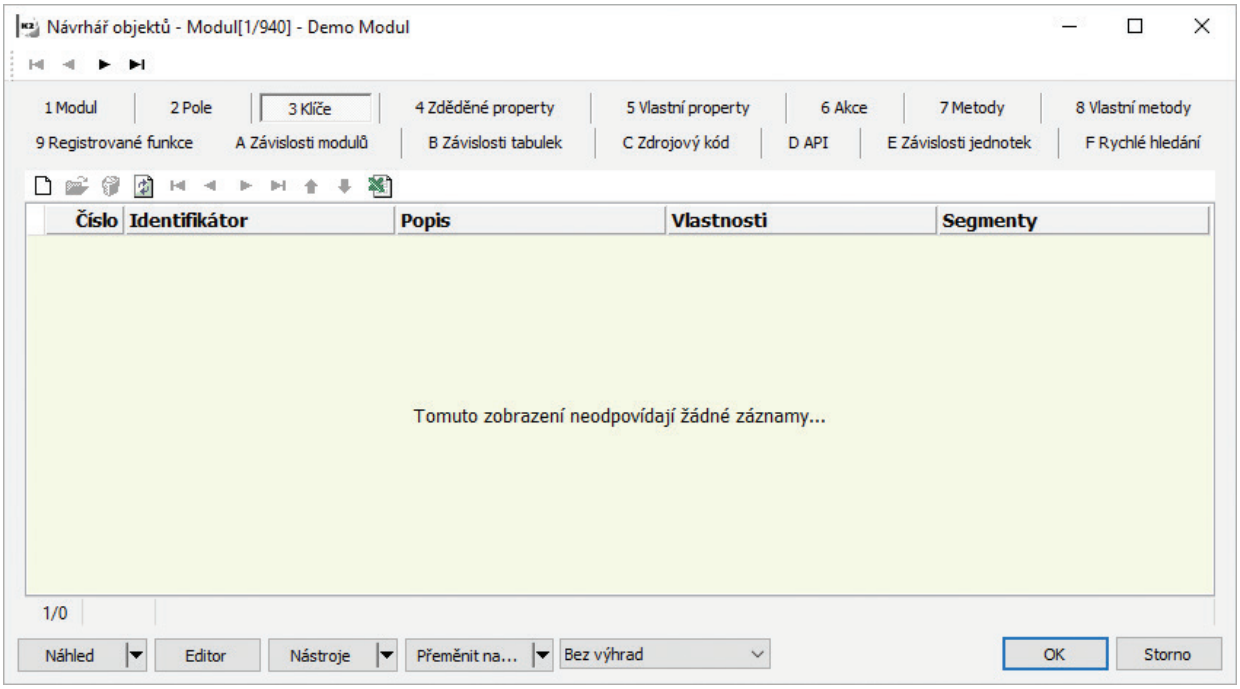
POZNÁMKA: Po akci ***„Deploy“***, návrhář objektů vygeneruje na databázové úrovni unikátní index, který v sobě zahrnuje dvě pole. Jsou jimi výše definovaná zkratka a typ, které jednoznačně identifikují každý záznam typového číselníku. V návrháři objektů je na záložce ***„Klíče“*** vidět vygenerovaný index, který ale neobsahuje tyto dva segmenty. Je vytvořen pouze nad polem, které slouží ke zkratce a je označeno jako unikátní. Rozšíření indexu o typ je prováděno až na úrovni databáze a není tedy konzistentní s definicí v návrháři objektů pro daný index.

POZNÁMKA: Existuje ještě jedno nastavení, které musí být splněno, aby typový číselník fungoval. Nejedná se o nastavení polí, ale o nastavení ***„zděděných vlastností“***, kterými se zabývá jiná kapitola 4.5.2. Vlastnosti předka TTypeDataM.

4.4. KLÍČE

Další záložkou jsou ***„Klíče“***. Tato sekce slouží k definici všech klíčů datového modulu. Klíči myslíme databázový primární klíč a indexy. Při publikování úprav do K2, viz kapitola 13.1. „Deploy“ – aplikování úprav do K2, dochází k vytváření klíčů a indexů přímo v databázových tabulkách daných datových modulů.

Při přepnutí na záložku ***„Klíče“*** máme k dispozici seznam polí a lištu tlačítek s funkcemi pro práci se seznamem. Nový klíč vložíme do seznamu pomocí ikony prázdného listu v liště tlačítek (v toolbaru) nebo pomocí stisku tlačítka Insert viz Obrázek 62 - Záložka "Klíče" v datovém modulu.

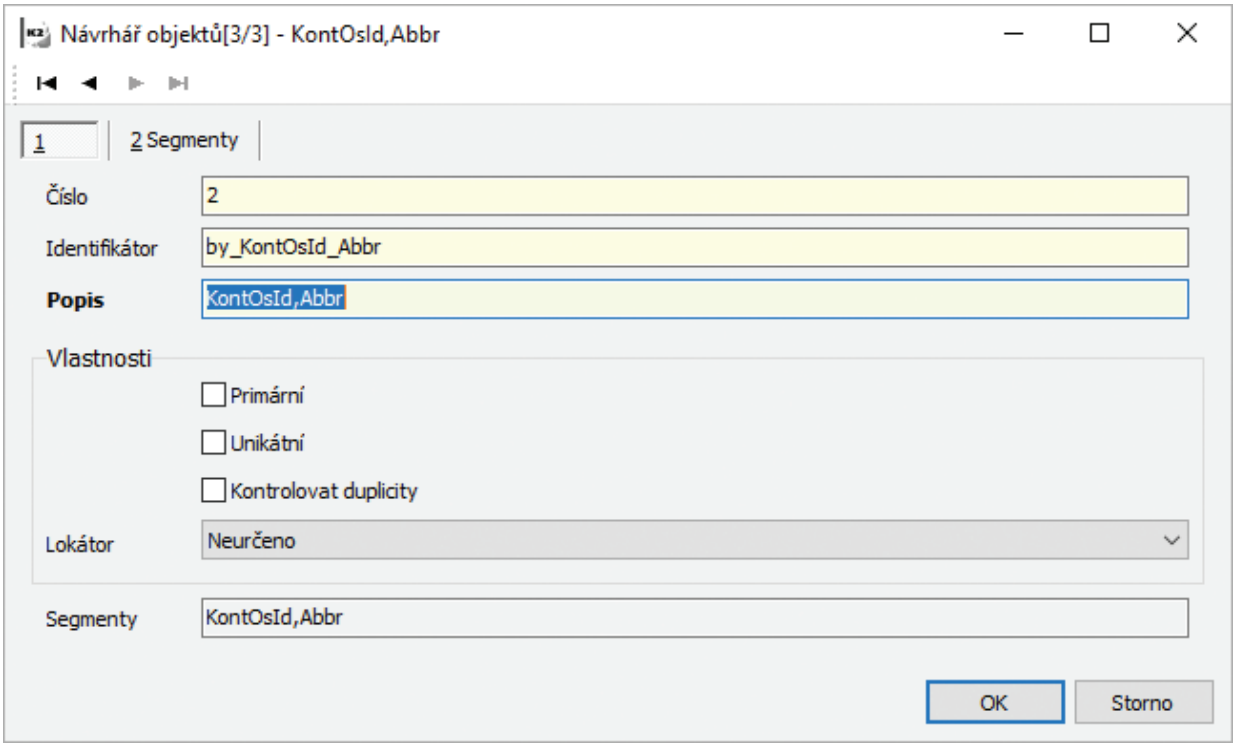


Obrázek 62 - Záložka "Klíče" v datovém modulu

POZNÁMKA: Návrhář objektů usnadňuje tvůrci datových modulů práci tím, že klíče přidává automaticky vždy, když shledá, že je to vyžadováno nebo vhodné. Tedy například, když je vloženo pole, které má příznak ***„Primární“***, automaticky je vytvořen klíč, který je také ***„Primární“*** a ***„Unikátní“***. V případě, že je vytvořeno pole, které má nastaven příznak ***„Unikátní“***, automaticky je vygenerován klíč, který je typu ***„Unikátní“***, viz podkapitola 4.2. Pole. Tyto generované definice nelze měnit ani mazat. Editaci a mazání klíčů je možné provádět jen nad uživatelskými klíči, tedy takovými, které nejsou automaticky generovány návrhářem objektů, ale jsou definovány uživatelem.

POZNÁMKA: V ideálním stavu by uživatel vůbec neměl vytvářet vlastní klíče v návrháři objektů. Všechny, které potřebuje, vytváří návrhář objektů automaticky sám. V případě potřeby je samozřejmě možné klíč nadefinovat.

Po vyvolání akce na vložení nového záznamu se zobrazí formulář Obrázek 63 - Definice nového klíče datového modulu. Je potřeba doplnit několik základních informací o klíči a dále nadefinovat vlastnosti vytvářeného klíče.



Obrázek 63 - Definice nového klíče datového modulu

4.4.1. ZÁKLADNÍ ÚDAJE

Tak jako u všech objektů i zde je potřeba doplnit několik údajů, které jasně definují vytvářený klíč.

Číslo
Unikátní číselný identifikátor klíče v rámci datového modulu. Návrhář objektů automaticky doplní údaj, který je korektní. V případě potřeby můžete změnit.

Identifikátor
Textové označení, které je automaticky generované návrhářem objektů. Struktura názvu je tvořena výčtem názvů všech polí, která jsou jednotlivými složkami klíče, oddělené podtržítkem. Na Obrázek 63 - Definice nového klíče datového modulu je vidět definice nového klíče, který je tvořen dvěma složkami (segmenty) – pole ***„KontOsId“*** a ***„Abbr“***. Návrhář objektů tedy vytvoří identifikátor ***„KontOsId_Abbr“***. Hodnotu tohoto pole je možné změnit vloženým textem do pole ***„Identifikátor“***.

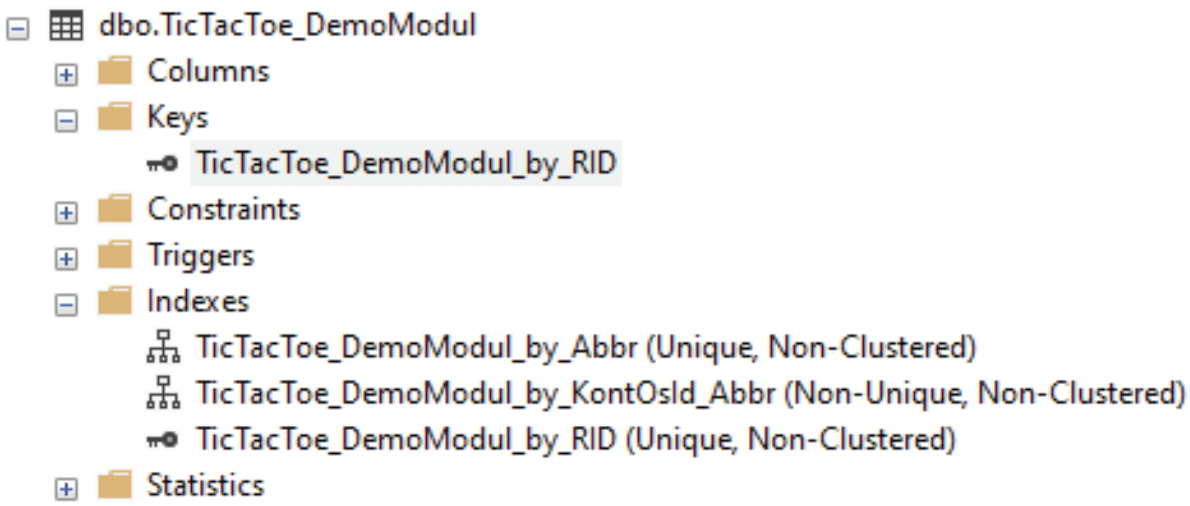
Popis
Slouží ke krátkému popisu daného klíče.

4.4.2. VLASTNOSTI

U každého klíče je možné nastavit vlastnosti, které specifikují jeho chování. Například zda je klíč unikátní. Následující výčet popisuje dostupné vlastnosti.

Primární
Tímto příznakem označujeme klíč, který je primární, tedy slouží jednoznačné identifikaci záznamů v daném datovém modulu. V tabulce by měl vždy existovat jeden takový klíč, který je tvořen segmenty všech polí, která jsou označena jako „Primární“. Primární klíč vytvořený přes NO by měl být vždy jednosložkový. Vícesložkový primární klíč se **důrazně nedoporučuje**.

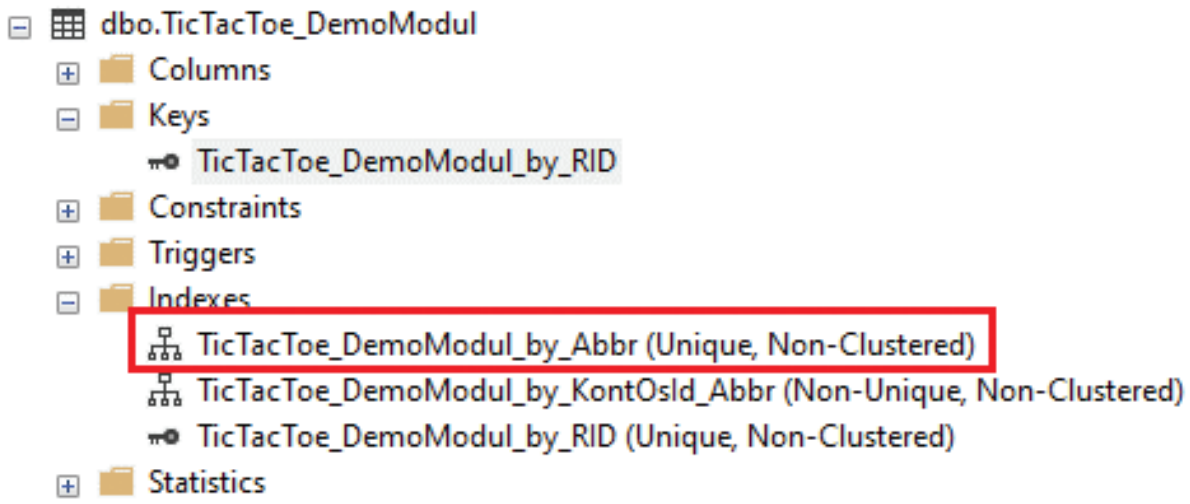
POZNÁMKA: Návrhář objektů tento klíč generuje automaticky po vložení polí s příznakem ***„Primární“***. Po tom, co se změny publikují do K2 (více v 13.1. „Deploy“ – aplikování úprav do K2), vznikne v databázi klíč a unikátní index pro tento klíč. V případě databáze typu MS SQL jsou vidět vytvořené objekty na Obrázek 64 - Vytvořený "primární" klíč v databázi MS SQL pro datový modul ***„DemoModul“*** vytvořený v návrháři objektů se jmenným prostorem definovaným jako ***„TicTacToe“***.



Obrázek 64 - Vytvořený "primární" klíč v databázi MS SQL

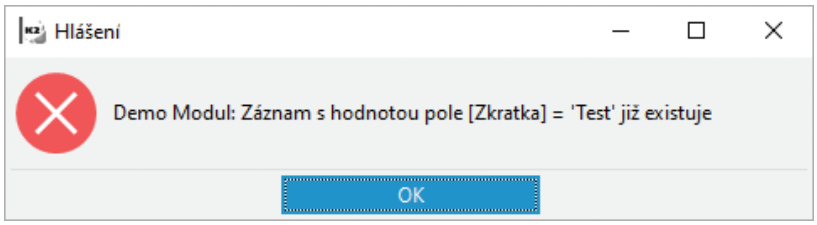
Unikátní
Tímto příznakem označujeme klíč, který je unikátní, tedy používá se u polí, která obsahují v rámci celé tabulky v každém řádku unikátní hodnotu. Tento klíč zajistí na úrovní databázové tabulky, že nelze vložit duplicitní záznam.

POZNÁMKA: Návrhář objektů tento klíč generuje automaticky po vložení polí s příznakem ***„Unikátní“***. Po tom co se změny publikují do K2 (více v 13.1. „Deploy“ – aplikování úprav do K2), vznikne v databázi unikátní index. V případě databáze typu MS SQL je vidět vytvořený objekt na Obrázek 65 - Vytvořený unikátní index v databázi MS SQL pro pole ***„Abbr“*** (zkratka) v datovém modulu ***„DemoModul“*** vytvořený v návrháři objektů se jmenným prostorem definovaným jako ***„TicTacToe“***.



Obrázek 65 - Vytvořený unikátní index v databázi MS SQL

Kontrola duplicity
V případě nastavení unikátního indexu, je na databázové úrovni kontrolováno, zda nedochází při ukládání nebo editaci k duplicitě v daném poli. V případě, že dojde k duplicitě, je zobrazeno hlášení ***„Obecná chyba“***. V návrháři objektů se unikátní klíče automaticky nastavují s příznakem ***„Kontrola duplicity“*** za předpokladu, že zvolíme příslušnému poli vlastnost ***„Unikátní“***, což má za následek, že je hlášení, v případě duplicity, zobrazeno v čitelnější podobě. Například máme nastaveno pole ***„Zkratka“*** jako unikátní a pokusíme se vložit duplicitní hodnotu ***„Test“***. Zobrazí se hlášení ***„Demo Modul: Záznam s hodnotou pole [Zkratka] = ‘Test‘ již existuje“***, viz Obrázek 66 - Hlášení o duplicitě v datovém modulu.



Obrázek 66 - Hlášení o duplicitě v datovém modulu

V případě vytváření vlastního unikátního klíče si můžeme tuto vlastnost zapnout/vypnout.

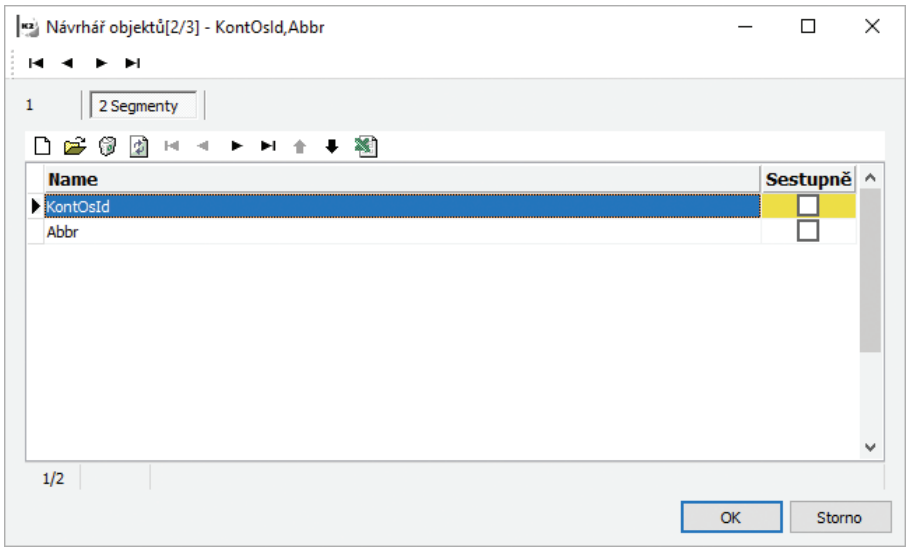
Lokátor
V případě, že požadujeme, aby nad daným klíčem fungoval lokátor, nastavíme hodnotu na ***„Ano“***.

Segmenty
Tento údaj je neměnný a jen zobrazuje z jakých polí, tedy z jakých složek (segmentů), je klíč složen. Nemusíme tedy přepínat na záložku ***„Segmenty“*** v případě, že potřebujeme vidět, pro která pole je klíč vytvořen.

4.4.3. SLOŽKY KLÍČE (SEGMENTY)

Každý klíč je vytvořen minimálně nad jedním polem, ale může být složen z libovolného množství polí. To nad jakými poli je klíč postaven definuje druhá záložka s názvem ***„Segmenty“***.

Když se přepneme na tuto záložku, např. nad klíčem, který je vytvořený na Obrázek 63 - Definice nového klíče datového modulu – pole ***„KontOsId“*** a ***„Abbr“***, uvidíme seznam polí, která jsou segmenty vytvářeného klíče. V seznamu je vidět vždy název pole, číselný identifikátor a příznak, zda se má pole v klíči setřídit vzestupně či sestupně.

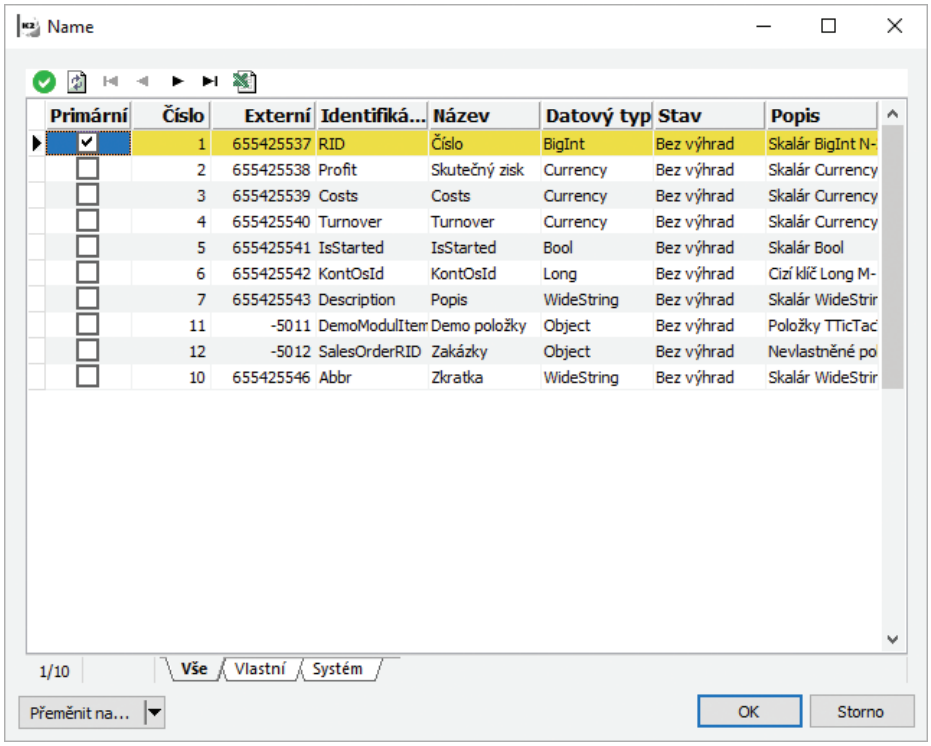


Obrázek 67 - Definice segmentů klíče datového modulu

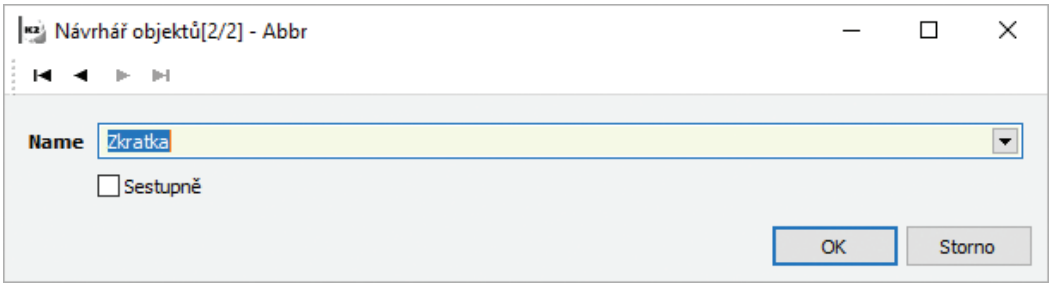
Nový segment klíče vložíme stisknutím tlačítka Insert nebo stisknutím tlačítka Nový. Zobrazí se nám formulář s výběrem z polí pro segment, po výběru se nám zobrazí formulář s definicí Obrázek 69 - Definice segmentu klíče v datovém modulu kde můžeme vidět, že je potřeba pouze vybrat pole z datového modulu, které bude vloženo do klíče, identifikátor je automaticky doplněn dle pole, které bylo vybráno.

Posledním parametrem segmentu klíče je jeho setřídění. Je možné nastavit, tak aby klíč pracoval nad segmentem, který je setříděn vzestupně, což je výchozí hodnota, nebo sestupně, což musí nastavit uživatel.

POZNÁMKA: Změnu setřídění segmentu klíče je možné provádět v datového modulu, který je potomkem ***„TMemFile“*** (paměťový datový modul) i ***„TAdoFile“*** (tabulkový datový modul).

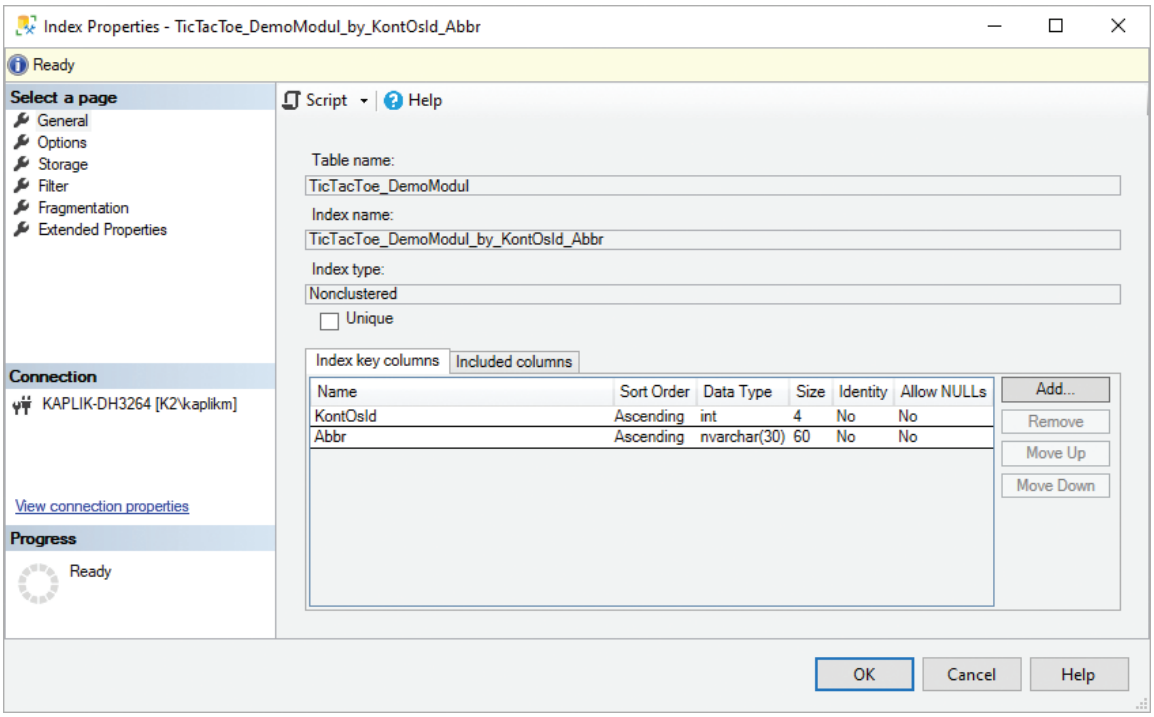


Obrázek 68 - Výběr pole pro segment klíče



Obrázek 69 - Definice segmentu klíče v datovém modulu

Po tom co se změny publikují do K2 (více v 13.1. „Deploy" – aplikování úprav do K2), vznikne v databázi index, který je složen z definovaných segmentů. V případě databáze typu MS SQL je například vidět vytvořený index ***„KontOsID_Abbr"*** na Obrázek 70 - Vytvořený více segmentový index v databázové tabulce na MS SQL serveru pro pole ***„KontOs"*** a ***„Abbr"*** v datovém modulu ***„DemoModul"*** vytvořeném v návrháři objektů se jmenným prostorem definovaným jako ***„TicTacToe"***.

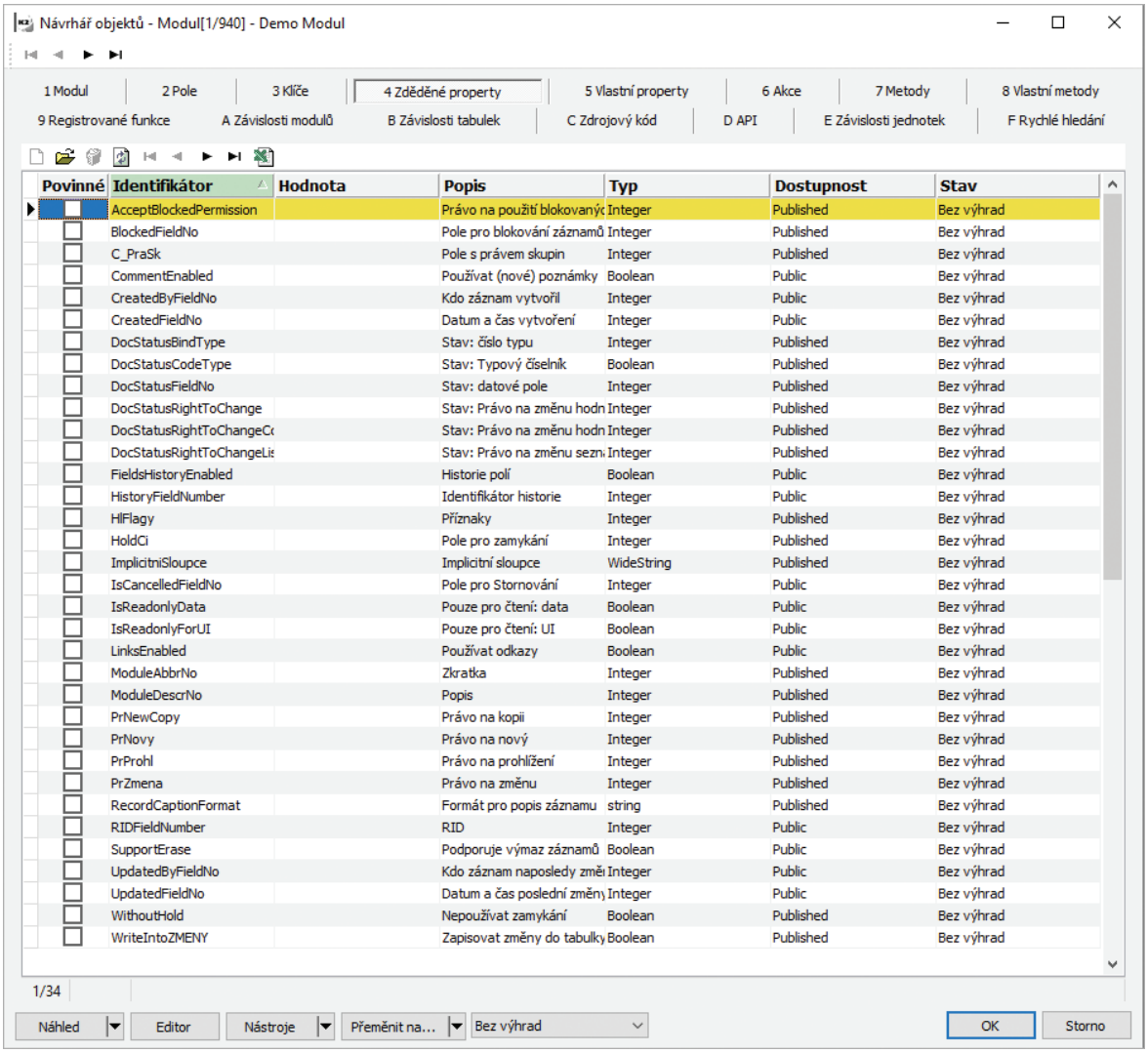


Obrázek 70 - Vytvořený více segmentový index v databázové tabulce na MS SQL serveru

POZNÁMKA: U segmentů klíče záleží na pořadí. V případě, že potřebujeme přesunout pořadí jednotlivých složek klíčů, je k dispozici, v nástrojové liště tlačítek nad seznamem segmentů klíčů, tlačítko pro posun jednotlivých segmentů klíče dolů nebo nahoru, dle potřeby.

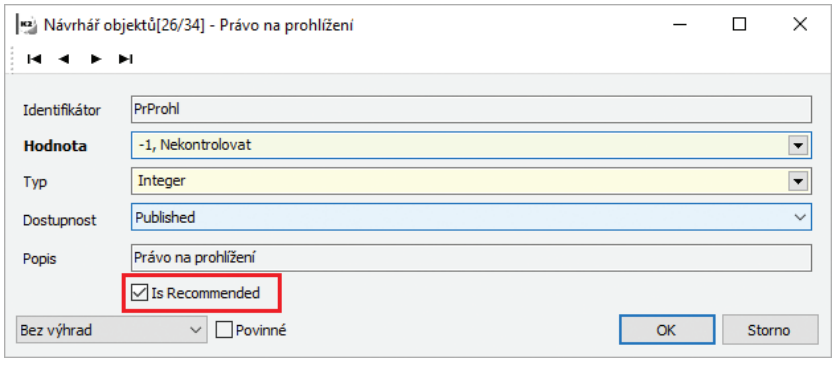
4.5. ZDĚDĚNÉ PROPERTY

Další záložkou při definici datového modulu je ***„Zděděné property"*** neboli vlastnosti datového modulu, které jsou k dispozici (zděděny) z předka datového modulu, který je definován na záložce ***„Modul"***, viz podkapitola 4.1. Modul. U každé vlastnosti je uveden identifikátor a popis. Dále je uvedena povinnost, tedy zda musí být vlastnost nastavena. V případě, že je povinná a není nastavena, nedovolí nám návrhář objektů uložit datový modul. Dále je uveden datový typ, tedy jestli se jedná o textovou nebo třeba číselnou vlastnost. Zajímavým údajem je i ***„Dostupnost"***, která udává použitelnost vlastnosti napříč různými moduly. Samozřejmě nesmí chybět případná hodnota vlastnosti, viz Obrázek 71 - Seznam vlastností předka "TBaseDataM" v datovém modulu.



Obrázek 71 - Seznam vlastností předka "TBaseDataM" v datovém modulu

Nastavení jednotlivých vlastností provedeme dvojitým kliknutím na vlastnost, kterou chceme nastavit, nebo stiskem tlačítka Změna v nástrojové liště nad seznamem vlastností nebo stiskem klávesy F5. Zobrazí se formulář, který je vidět na Obrázek 72 - Definice hodnoty zděděné vlastnosti datového modulu. Protože se jedná o zděděné vlastnosti, u kterých nelze měnit jejich chování, nastavujeme pomocí tohoto formuláře pouze hodnoty vlastností. Například v případě, že se jedná o datový typ vlastnosti ***„Boolean“***, tedy nabývá hodnot ***„pravda“*** nebo ***„nepravda“***, či ***„ano“*** a ***„ne“***, v poli ***„Hodnota“*** vybereme z jedné z uvedených hodnot. Dle datového typu vlastnosti se může hodnota lišit. Po stisku tlačítka OK se hodnota nastaví a formulář se uzavře.

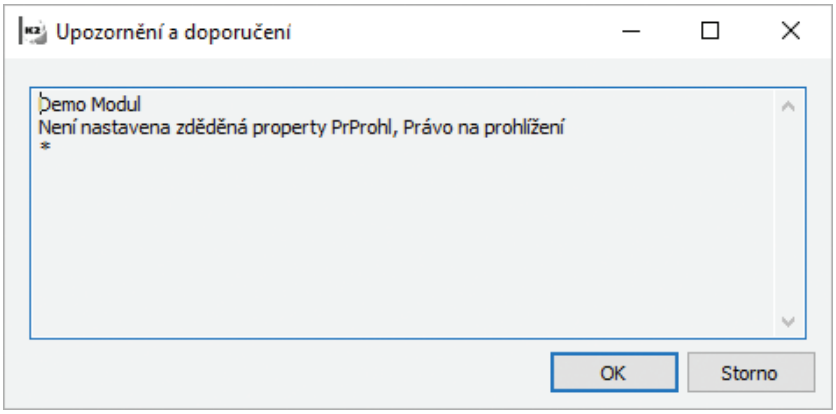


Obrázek 72 - Definice hodnoty zděděné vlastnosti datového modulu

Recommended (doporučeno)
U každé vlastnosti je uvedeno, zda je doporučené ji nastavit pro datový modul či ne. Tuto informaci podává příznak ***„Recommended“***, který je vidět na Obrázek 72 - Definice hodnoty zděděné vlastnosti datového modulu v červeném obdélníku.

Bez těchto vlastností bude datový modul sice fungovat, ale pouze v omezeném režimu. Například nenastavení práva na prohlížení má za následek, že datový modul nepůjde otevřít. Nenastavení implicitních sloupců má zase za následek, že se nám do seznamu záznamů ve formuláři, žádné nenačtou.

V případě, že tyto vlastnosti nenastavíme, je při aplikování definice rozšíření – akce ***„Deploy“***, zobrazen formulář, ve kterém nalezneme souhrn všech doporučených vlastností, které nejsou nastaveny, a to pro každý datový modul z definice rozšíření. Na Obrázek 73 - Informace o doporučeném nastavení můžeme vidět varování o nenastavené vlastnosti ***„Právo na prohlížení“*** v datovém modulu ***„Demo Modul“***.



Obrázek 73 - Informace o doporučeném nastavení

V následujícím textu si popíšeme všechny dostupné vlastnosti z jednotlivých bázových datových modulů, ze kterých můžete dědit při vytváření datového modulu. Vlastnosti, které jsou společné, budou samozřejmě popsány jen jednou v místě, kde na ně poprvé narazíme.

4.5.1. VLASTNOSTI PŘEDKA TBASEDATAM

Prvním předkem, který je ve výčtu dostupný, je ***„TBaseDataM“***. Tento modul je zároveň použitý jako výchozí. Předek ***„TBaseDataM“*** je základním předkem všech datových modulů v K2. Všichni další potomci, jsou odvozeni od ***„TBaseDataM“***.

Při použití tohoto předka získáváte následující výčet vlastností. Seznam je možné vidět i na obrázku Obrázek 71 - Seznam vlastností předka "TBaseDataM" v datovém modulu.

4.5.1.1. LinksEnabled

Určuje, zda bude datový modul používat odkazy. V případě zapnutí bude pro modul zavedena podpora odkazů, včetně logiky s nimi spojenou. V generovaném formuláři pak můžeme vidět novou záložku ***„Odkazy“***.

Jako hodnotu této vlastnosti vybíráme z variant ***„True“*** - ***„ano“*** / ***„False“*** - ***„ne“***.

4.5.1.2. CommentEnabled

Určuje, zda bude datový modul používat nové poznámky. V případě zapnutí bude pro modul zavedena podpora nových poznámek, včetně logiky s nimi spojenou. V generovaném formuláři pak můžeme vidět novou záložku ***„Komentáře“***.
Jako hodnotu této vlastnosti vybíráme z variant ***„True“*** - ***„ano“*** / ***„False“*** - ***„ne“***.

4.5.1.3. WriteIntoZMENY

Pomocí této vlastnosti určíme, zda požadujeme, aby změny provedené v tomto datovém modulu, tedy vkládání a editace záznamů, byly logovány, zapisovány do tabulky ***„ZMENY“***, která je k tomuto určena.
Jako hodnotu této vlastnosti vybíráme z variant ***„True“*** - ***„ano“*** / „False“ - ***„ne“***.

POZNÁMKA: Pokud chceme zapisovat změny jako je změna, nový nebo potvrzení nad naším DM, je nutné nastavit parametr mandanta ***„Do změn zapisovat i akci změna, nový a potvrzení“***.

4.5.1.4. CreatedFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o datu a času pro každý záznam, který byl vytvořený. Tedy, když budeme vkládat nový záznam, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží datum a čas, kdy byl záznam vytvořen.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ ***„DateTime“***.

4.5.1.5. CreatedByFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o tom, kdo daný záznam vložil do tohoto datového modulu. Tedy, když budeme vkládat nový záznam, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží odkaz na uživatele přihlášeného do IS K2, tedy na toho, který záznam vytvořil.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu.

4.5.1.6. UpdatedFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o datu a času pro každý záznam, který se změnil. Tedy, když budeme editovat záznam, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží datum a čas, kdy byl záznam změněn.
Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ ***„DateTime“***.

4.5.1.7. UpdatedByFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o tom, kdo daný záznam změnil. Tedy, když budeme měnit záznam, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží odkaz na uživatele přihlášeného do IS K2, tedy na toho, který záznam změnil.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu.

4.5.1.8. FieldsHistoryEnabled

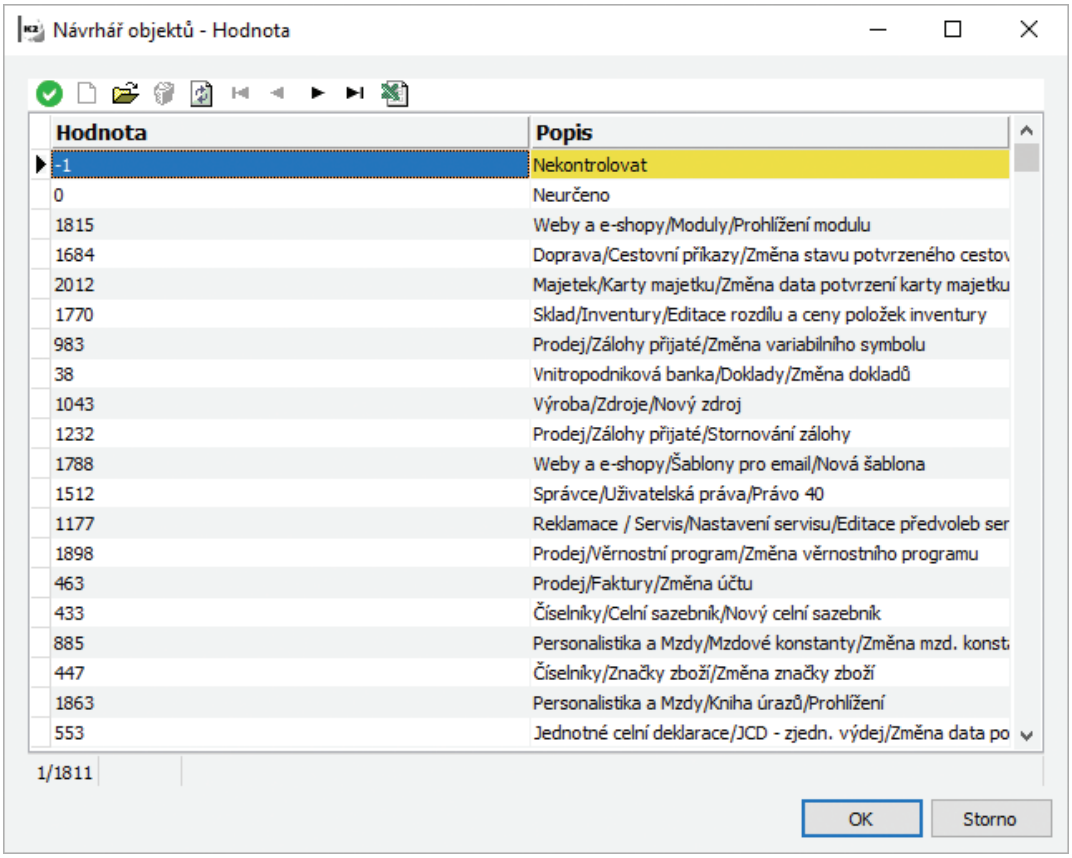
Nastavením této vlastnosti definujeme, zda má být aktivní logika, která pro pole datového modulu, která mají nastaven příznak ***„Evidovat historii“*** eviduje historii hodnot vkládaných do pole v rámci jednoho záznamu. Lze tedy hromadně vypnout tento mechanismus pro všechny pole datového modulu s evidencí historie.

Jako hodnotu této vlastnosti vybíráme z variant ***„True“*** - ***„ano“*** / ***„False“*** - ***„ne“***.

4.5.1.9. PrProhl

Každý datový modul, který vytvoříme, musí mít nastaveno oprávnění, kdo může modul používat. Existuje několik úrovní práv. Vlastností ***„PrProhl“*** nastavujeme základní právo pro vytvářený datový modul, a tím je právo na prohlížení. Pokud tuto vlastnost nenastavíme, nikdo nebude moci tento modul otevřít. Hodnotou této vlastnosti je vazba do číselníku práv. Tedy v případě, že přidělujeme právo, otevírá se nám číselník práv, kde zvolíme patřičné právo, které bude svázáno se čtením dat z tohoto modulu. Viz Obrázek 74 - Výběr práva pro vlastnost "PrProhl" datového modulu.

POZNÁMKA: V případě, že chceme povolit přístup všem, nastavíme právo na hodnotu ***„-1 - Nekontrolovat“***. Důležité je, že hodnota musí být nastavena, jinak modul neotevřeme.



Obrázek 74 - Výběr práva pro vlastnost "PrProhl" datového modulu

4.5.1.10. PrNovy

Dalším druhem práva je ***„právo na nový záznam“***. Stejně jako v předchozím případě u ***„PrProhl“*** i zde musíme hodnotu nastavit, pokud chceme, aby někdo mohl vkládat záznamy. V případě, že nechceme tuto vlastnost omezovat, vložíme hodnotu ***„-1 - Nekontrolovat“***, jinak vybíráme konkrétní právo stejně jako u ***„PrProhl“***.

4.5.1.11. PrZmena

Pomocí této vlastnosti nastavujeme ***„právo na změnu“*** záznamů v aktuálním modulu. Nastavuje se stejně, jako v předchozích dvou variantách - ***„PrProhl“*** a ***„PrNovy“***.

4.5.1.12. PrNewCopy

Pomocí této vlastnosti nastavujeme ***„právo na kopii“*** záznamů v aktuálním modulu. Nastavuje se stejně, jako v předchozích variantách – např. u ***„PrProhl“***.

4.5.1.13. IsCancelledFieldNo

Do této property vložíme pole pro stornování (doporučuje se pole typu ***„Boolean“***). Pokud zmáčkneme F8, záznam se stornuje, jak jsme zvyklí např. z Dokladů. **Funguje pouze nad TCustomDataM**.

POZNÁMKA: Tato property je závislá na property ***„ConfirmedOnFiledNo“*** – pole pro datum potvrzení. Pokud tato property není vyplněná, stronování/odstronování nefunguje.

4.5.1.14. DocStatusFieldNo

Pomocí této property určíme, které datové pole bude sloužit pro ukládání informace o statusu (známe např. status ze Zboží nebo Zakázky). Vybrané pole musí být celočíselného typu.

POZNÁMKA: Pokud použijeme status pro předka TCustomData, TCustomListDataM nebo TCustomDocumentDataM, můžete pro změnu statusu použít i klávesovou zkratku Ctrl + Alt + F5.

4.5.1.15. DosStatusBindType

Tato property určuje typ. Vkládáme pouze celé kladné číslo.

4.5.1.16. DocStatusCodeType

Pomocí této property určíme, zda se jedná o typový číselník, či nikoliv. Pokud ano (***„True“***), pole pro typ je vazbou do tabulky CODETYPE, pokud zvolí ne (***„False“***) typ je vazbou do tabulky SPECIALSTRINGLIST.

4.5.1.17. DocStatusRightToChangeConfirmed

Pro jednotlivé úkony statusu můžeme definovat různá práva. Tato property slouží pro nastavení práva na změnu hodnoty v potvrzeném záznamu. Pokud uživatel dané právo nemá, nemůže měnit hodnotu v poli pro status, pokud je záznam potvrzený.

4.5.1.18. DocStatusRightToChange

Pomocí této property můžeme nastavit právo na změnu hodnoty v poli pro status. Pokud uživatel dané právo nemá, nemůže měnit hodnotu v poli pro status.

4.5.1.19. DocStatusRightToChangeList

Pomocí této property můžeme nastavit právo na změnu seznamu pro status. Pokud uživatel nemá dané právo, nemůže vytvářet další statusy.

4.5.1.20. AcceptBlockedPermission

Pomocí této vlastnosti nastavujeme ***„právo na použití blokovaných záznamů“*** v aktuálním modulu. Jedná se o povolení, zda uživatel může pracovat s blokovanými záznamy. Nastavení je stejné, jako v předchozích variantách – např. u ***„PrProhl“***.

4.5.1.21. BlockedFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o tom, zda je daný záznam blokovaný či ne. Tedy, když označíme záznam o tom, že je záznam blokovaný, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží informace o blokaci záznamu.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ ***„Boolean“*** nebo se musí jednat o takový typ, který je možný pomocí funkce ***„AsBool“*** převést na ***„Boolean“***. Například typ ***„Byte“***, který v případě hodnoty ***„0“*** bude nabývat hodnoty ***„False“***, cokoliv jiného bude znamenat ***„True“***.

4.5.1.22. C_Prask

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o tom, které skupiny práv mají oprávnění k tomuto záznamu.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ minimálně ***„Short“***.

4.5.1.23. WithoutHold

Nastavením vlastnosti na hodnotu ***„True“*** je datový modul nastaven do režimu, kdy nebude používat holdování. V tomto případě nemusí být nastavena ani vlastnost ***„HoldCi“***, která je uvedena v další části.

4.5.1.24. HoldCi

V případě, že datový modul podporuje operaci ***„holdování“*** (uzamykání záznamů), je potřeba tuto vlastnost upřesnit pomocí vlastnosti ***„HoldCi“***, která určuje, které pole v datovém modulu bude sloužit k ukládání informace o tom, že je záznam aktuálně měněn jiným uživatelem K2. Tedy vlastnost slouží k definici ***„pole pro zamykání“***.

V případě, že vlastnost nenastavíme a datový modul podporuje „holdování“, knihovna vybere za běhu takové pole, které je pro tuto logiku vhodné. Pravděpodobně se bude jednat o primární klíč.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí být typu **primární klíč**.

4.5.1.25. HlFlagy

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o příznacích záznamu. Přesněji se jedná o jeden příznak, a tím je ***„Storno“***. Tento příznak se nachází na pozici číslo ***„4“***, tedy je nutné, aby pole bitů mělo nejméně délku čtyři.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ pole bitů.

4.5.1.26. RIDFieldNumber

V případě, že datový modul má definován primární klíč datového typu ***„Int64“***, který se nejmenuje ***„RID“***, je nutné tuto skutečnost nastavit do vlastnosti ***„RIDFieldNumber“***. Jako hodnotu vybíráme pole, které slouží jako primární klíč datového typu ***„Int64“***.

V případě, že se primární pole jmenuje ***„RID“*** není potřeba tuto vlastnost nastavovat.

4.5.1.27. HistoryFieldNumber

Pokud nastane situace, kdy potřebujeme mít primární klíč jiný, než RID (myšleno jeho pojmenování) a potřebujeme evidovat historii, využijeme property ***„HistoryFieldNumber“***. Do této property vložíme pole, které je primárním klíčem (jiný než RID). Pro primární klíč musí být splněno, že se jedná o celočíselný typ a je unikátní. Poté můžeme evidovat historii u námi vybraných polí pomocí fajfky na poli ***„Evidovat historii“***.

4.5.1.28. SupportErase

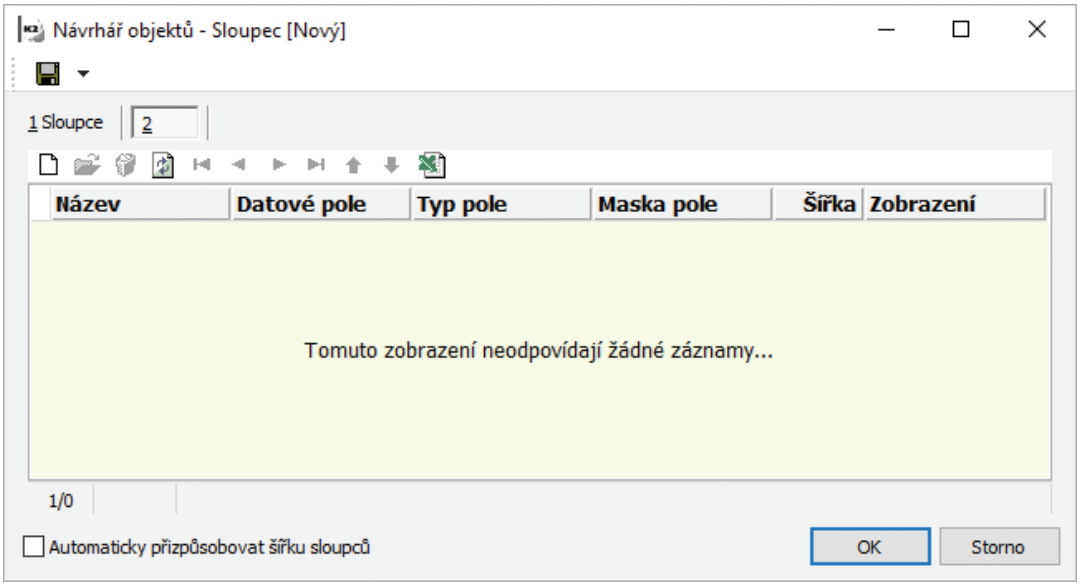
Aktivuje podporu pro mazání záznamů z tohoto datového modulu. Mazat záznamy je povoleno pouze ze skriptu. Používá se konstrukce TDataM.Erase.

POZNÁMKA: Při mazání není implementována kontrola na existenci mazaného záznamu ve vazbách jiných modulů či na existenci záznamů v odkazech a dokumentech na deváté straně datových modulů. Může tedy dojít k nekonzistenci.

4.5.1.29. ImplicitniSloupce

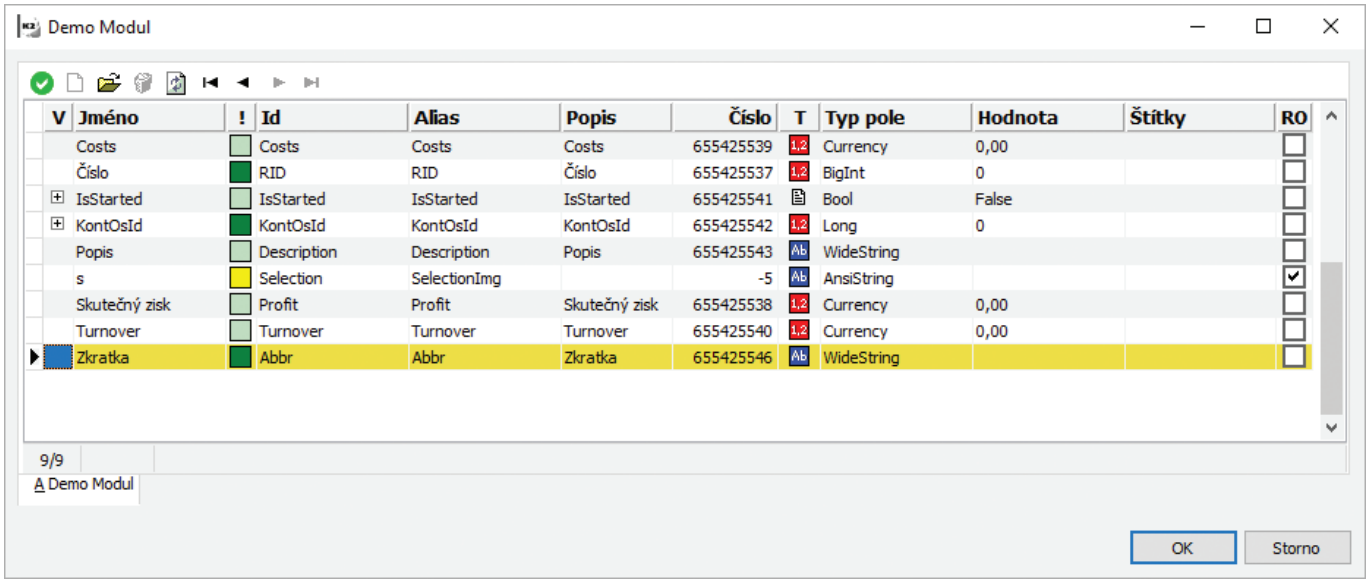
Pomocí této vlastnosti určujeme, která pole budou v datovém modulu implicitní. Tedy která budou načtena a zobrazena v seznamu v aktuálním datovém modulu ve výchozím stavu.

Hodnota této vlastnosti je typu ***„řetězec“***, který má specifický formát zápisu. Každé pole je ohraničeno hranatými závorkami. V případě, že potřebujeme zobrazit hodnotu z vazby nebo z více vazeb, pak je mezi polem a vazbou použitý znak středník ***„ ; “***. Pokud potřebujeme změnit ***„Picture“*** (formát zobrazení) pro pole, můžeme uvést jeho hodnotu, tak že za název pole vložíme dvakrát dvojtečku - ***„ :: “*** a následně hodnotu ***„picture“***. Tento definující řetězec není nutné vytvářet ručně, ale je možné využít formuláře, který nabízí pohodlnější variantu. Kliknutím na rozevírací nabídku v poli ***„Hodnota“*** viz Obrázek 79 - Výsledná definice implicitních sloupců, se zobrazí formulář Obrázek 75 - Formulář pro definici implicitních sloupců, do kterého můžeme klávesou Insert nebo pomocí stisku tlačítka Nový v panelu akcí nad seznamem, vložit sloupec.



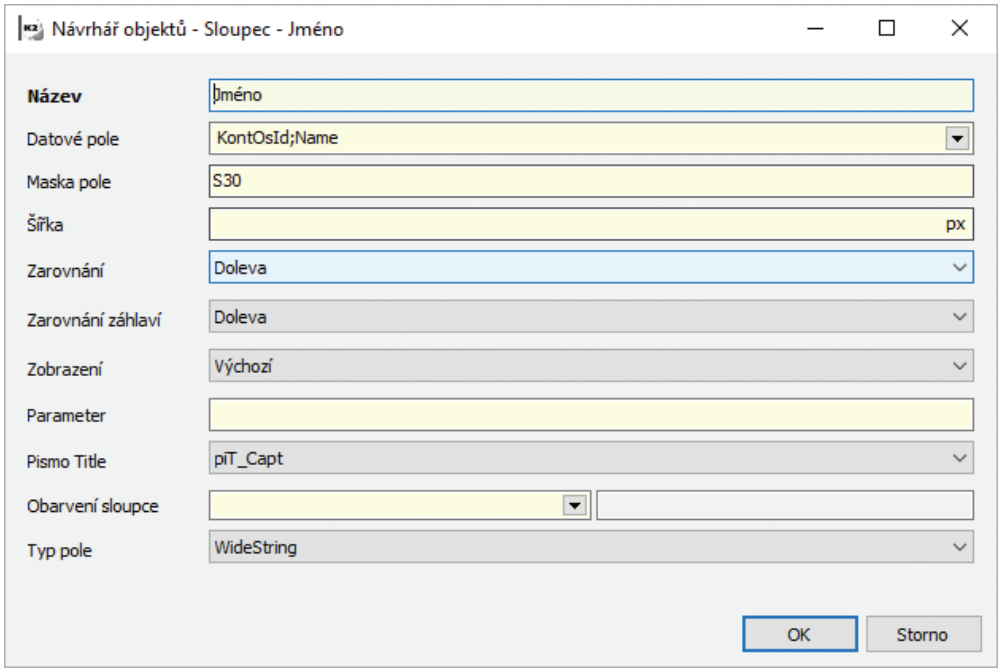
Obrázek 75 - Formulář pro definici implicitních sloupců

Při vkládání, se zobrazí formulář se seznamem sloupců dostupných v aktuálním datovém modulu, viz Obrázek 76 - Výběr sloupců pro implicitní sloupce.



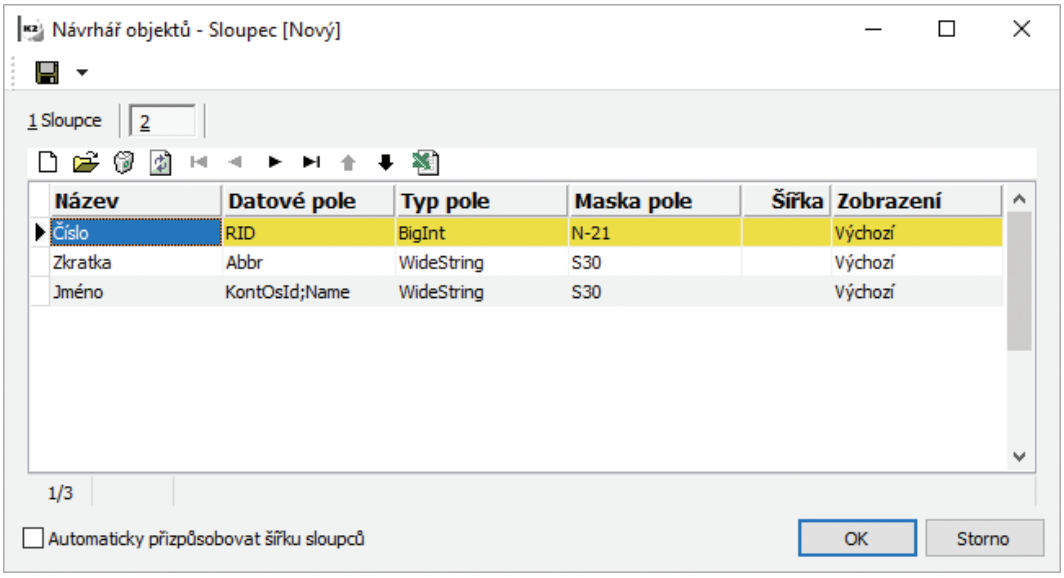
Obrázek 76 - Výběr sloupců pro implicitní sloupce

Výběrem záznamu se nám zobrazí formulář Obrázek 77 - Nastavení sloupce v implicitních sloupcích kde můžeme upravit nastavení zobrazení. Potvrzením sloupec vložíme do seznamu. V případě, že potřebujeme vložit sloupec z vazby, můžeme rozbalit seznam polí z vazebního modulu pomocí kliknutí na ikonu se symbolem „+“, která je v prvním sloupci seznamu všech polí viz Obrázek 76 - Výběr sloupců pro implicitní sloupce. Poté již můžeme vybírat ze seznamu polí vazebního modulu.



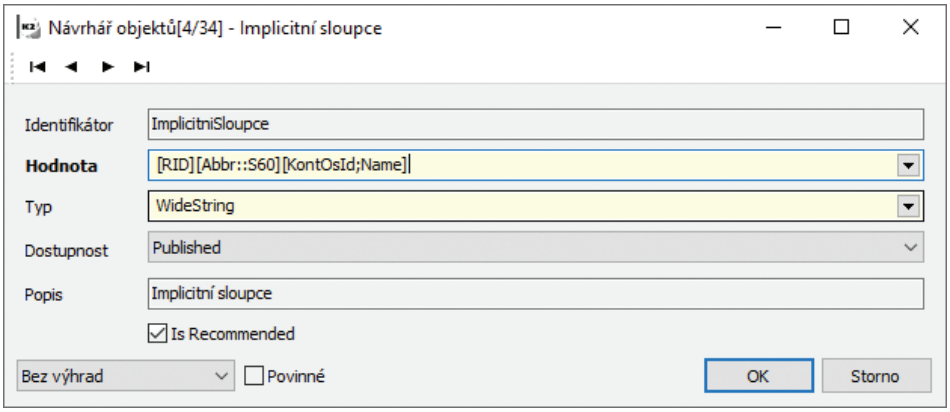
Obrázek 77 - Nastavení sloupce v implicitních sloupcích

Výsledný seznam pak může vypadat jako na obrázku Obrázek 78 - Seznam sloupců v implicitních sloupcích.



Obrázek 78 - Seznam sloupců v implicitních sloupcích

Po uzavření formuláře pro definici seznamu výchozích sloupců můžeme vidět vytvořenou definici včetně nastavení zobrazení polí. V tomto konkrétním příkladu máme vybrány tři sloupce, z nichž jeden je vazební (jedná se o vazbu do datového modulu ***„Kontaktní osoby“***) a u pole ***„Zkratka“*** je změněn implicitní formát zobrazení na délku ***„60“*** - ***„[RID][Abbr::S60][KontOsId:Name]“***.



Obrázek 79 - Výsledná definice implicitních sloupců

4.5.1.30. ModuleAbbrNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k zobrazení zkratky ve formuláři v případě vazby. Tedy pokud bude jiný modul odkazovat do aktuálního modulu (existuje vazba do aktuálního modulu), pak ve formuláři uvidíme v textovém poli, které představuje vazbu, hodnotu tohoto pole – ***„ModuleAbbrNo“*** pro vybraný záznam. Dalo by se říci, že se jedná o tzv. ***„výstupní pole“***.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu.

4.5.1.31. ModuleDescrNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k zobrazení popisu ve formuláři v případě vazby. Funguje na stejném principu jako předchozí vlastnost ***„ModuleAbbrNo“***. Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu.

4.5.1.32. RecordCaptionFormat

Pomocí této vlastnosti můžeme definovat ***„formát pro popis záznamu“***. V případě, že záznam zobrazujeme, můžeme nadefinovat formát tohoto zobrazení. Typickým příkladem může být pojmenování při notifikaci.

Formát má speciální syntaxi. Je možné použít následující tři typy hodnot:

- statický text
- hodnoty z polí z datového modulu. Zápis je ve tvaru ***„{DataM.název fieldu}“***
- hodnoty z vlastností datového modulu, které mají modifikátor viditelnosti nastaven na ***„published“***. K vlastnostem se přistupuje pomocí zápisu ***„{název vlastnosti}“***

4.5.1.33. IsReadonlyData

Modul je tímto nastaven do režimu pouze pro čtení. Zápis není povolen ani na úrovni skriptu. Jedná se o striktní zákaz zápisu do datového modulu.

4.5.1.34. IsReadonlyForUI

Modul je tímto nastaven do režimu pouze pro čtení. Zápis není povolen pouze z prostředí formulářů, kde toto omezení hlídá validace vstupů na formuláři.

POZNÁMKA: Pokud nastavíme „IsReadonlyData“, automaticky se nastavuje je DM i „IsReadonlyForUI“.

4.5.2. VLASTNOSTI PŘEDKA TTYPEDATAM

Dalším předkem, který je ve výčtu dostupný, je ***„TTypeDataM“***. V případě, že budete vytvářet nový datový modul, který bude sloužit jako typový číselník, je vhodné použít právě tohoto předka.

Při použití získáváte následující výčet vlastností. Vlastnosti, které jsou společné jako v předchozím bázovém modulu ***„TBaseDataM“***, zde již popisovat nebudeme.

4.5.2.1. TypCis

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k určení typu typového číselníku. Tedy pomocí tohoto pole definujeme skupiny záznamů v typovém číselníku.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole celočíselného typu.

POZNÁMKA: Tato vlastnost je povinná. Pokud tedy definujeme datový modul, který je potomkem ***„TTypeDataM“*** musíme vždy nastavit tuto vlastnost, která je poděděná z tohoto předka.

4.5.2.2. TypFlg

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k definici pole bitů – příznaků. V tomto poli bude uložena informace o blokování záznamů typového číselníku.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole typu pole bitů.

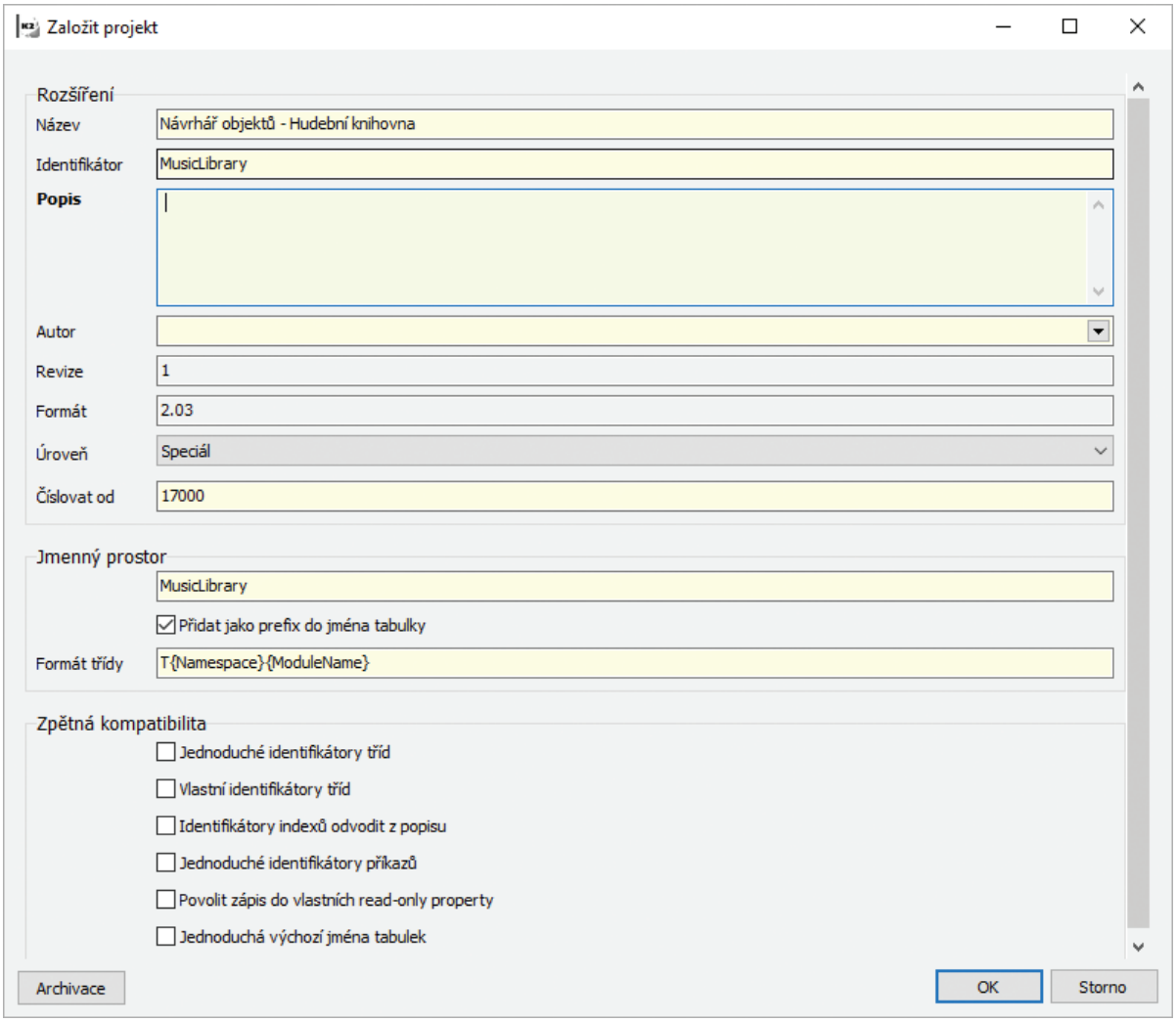
4.5.2.3. TypFlgBlock

Zde určujeme, který bit z pole bitů, uvedeném v předchozí vlastnosti ***„TypFlg“***, bude sloužit k blokování záznamů.

PŘÍKLAD: Na následujícím příkladu si ukážeme vytvoření datového modulu, který nám bude sloužit jako typový číselník. Součástí příkladu bude také modul, který vytvořený typový číselník použije.

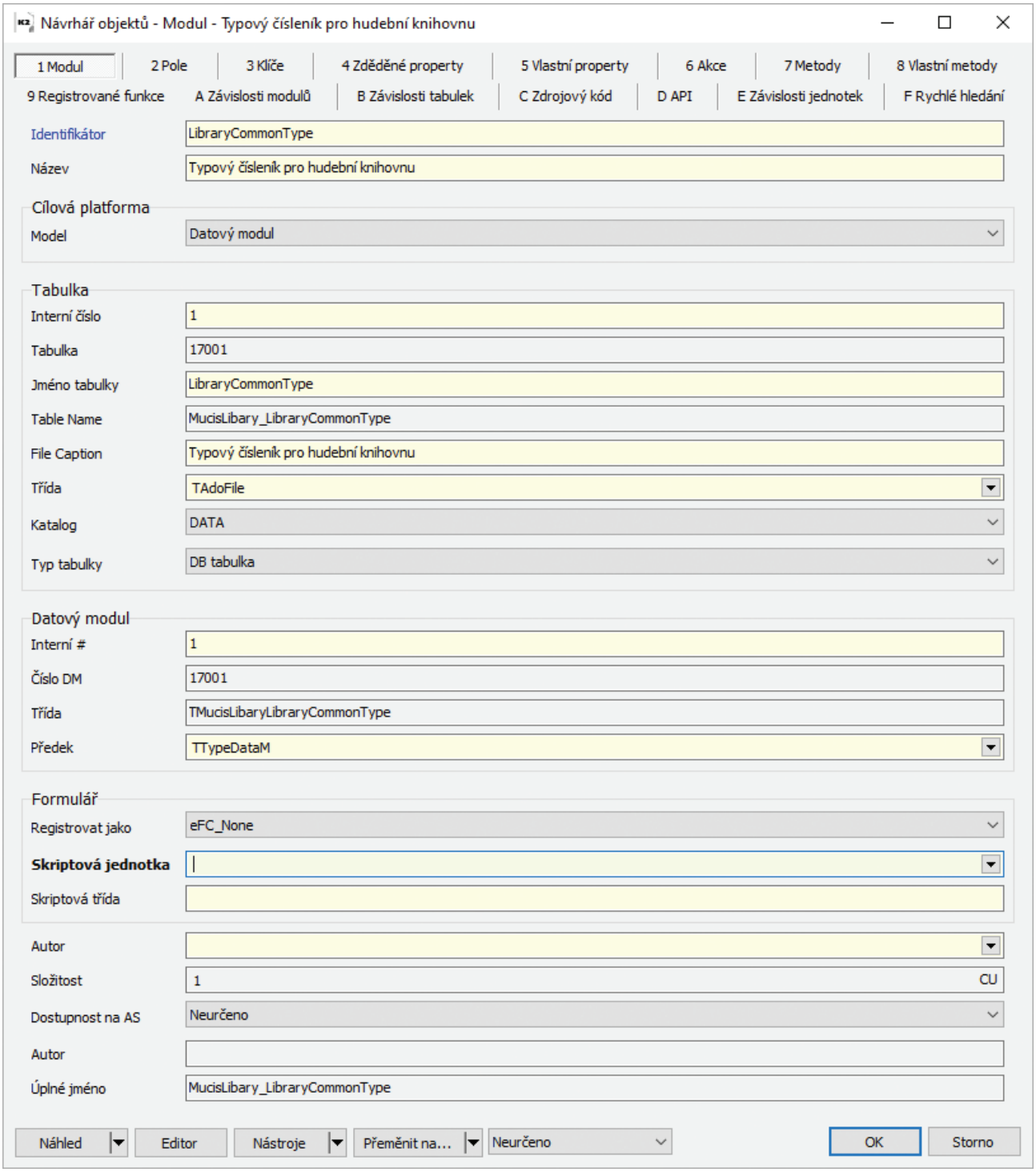
Představme si evidenci pro hudební nosiče. Tedy hudební knihovnu. U každého záznamu potřebujeme evidovat několik vlastností. Například druh hudebního nosiče, typ hudby apod. Pro tyto účely se nám nabízí vytvoření typového číselníků, který nám bude sloužit k evidenci hodnot jednotlivých typů vlastností pro hudební nosiče.

Založíme si nové rozšíření v návrháři objektů, kde nadefinujeme jmenný prostor jako ***„MusicLibrary“*** a rozsah číslování ***„Číslovat od“*** nastavíme na ***„17000“***, viz Obrázek 80 - Příklad vytvoření datového modulu jako typový číselník.



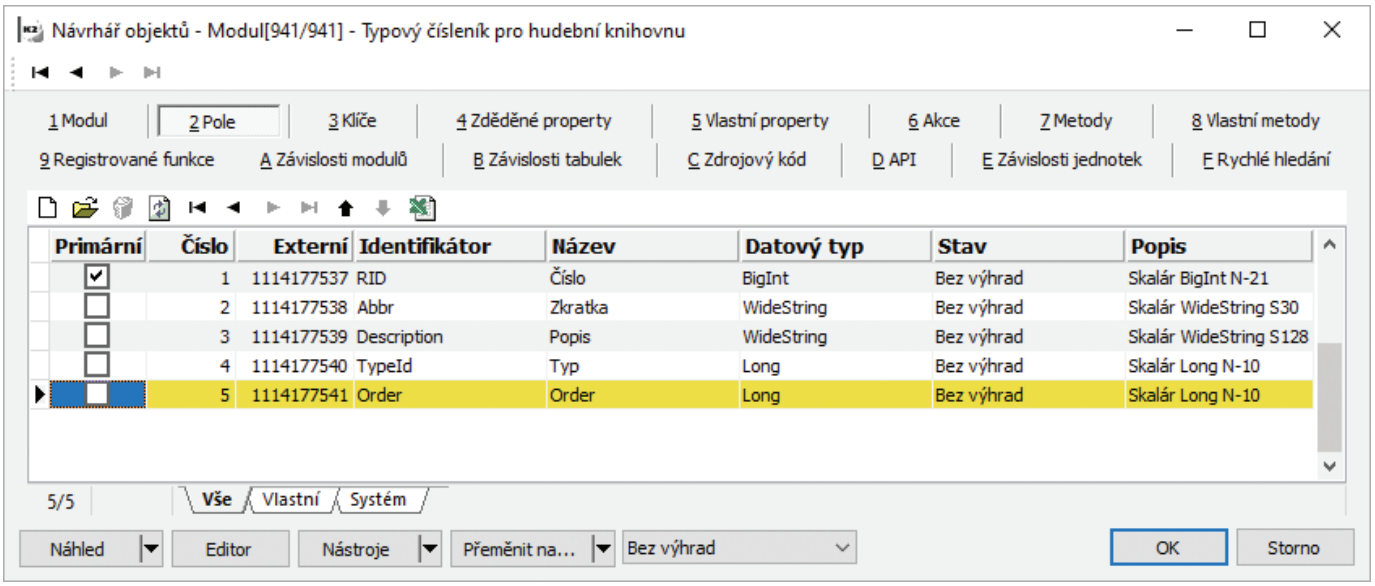
Obrázek 80 - Příklad vytvoření datového modulu jako typový číselník

Dále založíme nový datový modul, který bude sloužit jako typový číselník. Nazveme ho ***„LibraryCommonType“*** – ***„Typový číselník pro hudební knihovnu“***. Jako předka nastavíme ***„TTypeDataM“***, viz Obrázek 81 - Datový modul jako typový číselník.



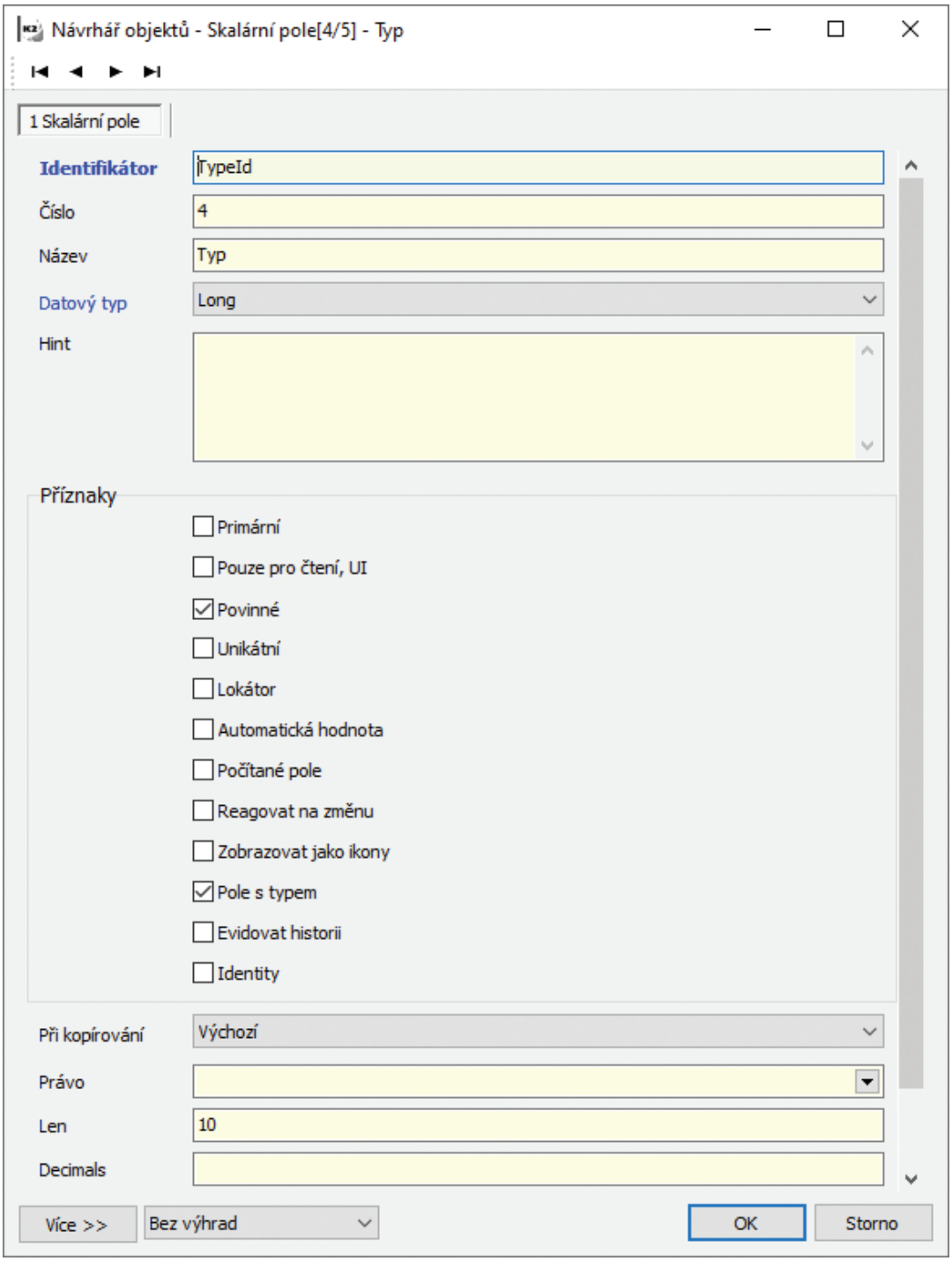
Obrázek 81 - Datový modul jako typový číselník

V datovém modulu nadefinujeme pole ***„RID“*** jako primární klíč, ***„Abbr“*** jako zkratku, ***„Description“*** jako popis, ***„TypeId“*** sloupec pro rozlišení typů v číselníku, a nakonec pole ***„Order“***, které můžeme využít k definici setřídění záznamů, viz Obrázek 82 - Definice polí datového modulu jako typový číselník.



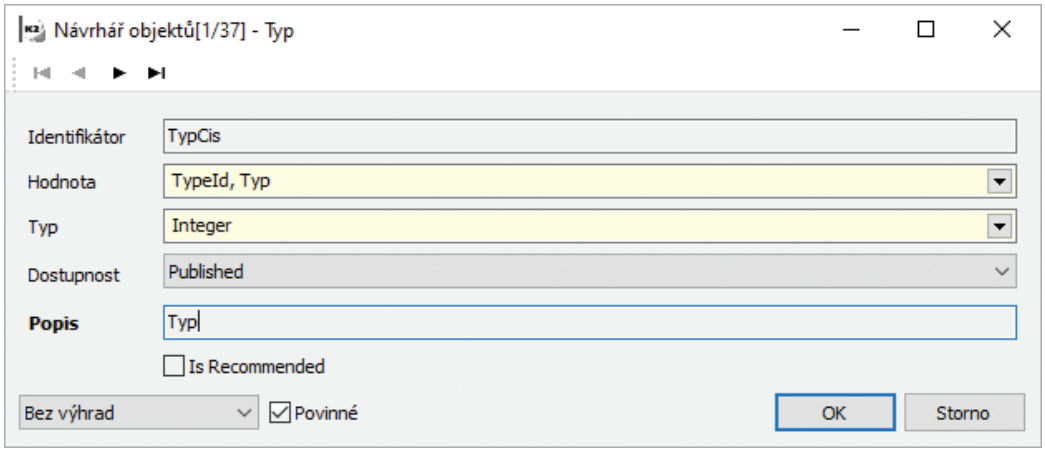
Obrázek 82 - Definice polí datového modulu jako typový číselník

Aby datový modul fungoval správně jako typový číselník, je důležité nastavit pole, pomocí kterého rozlišujeme jednotlivé typy záznamů v číselníku. V našem případě se jedná o pole ***„TypeId“***. V jeho definici musíme zatrhnout volbu ***„Pole s typem“***, což je vidět na Obrázek 83 - Nastavení pole s typem v typovém číselníku.



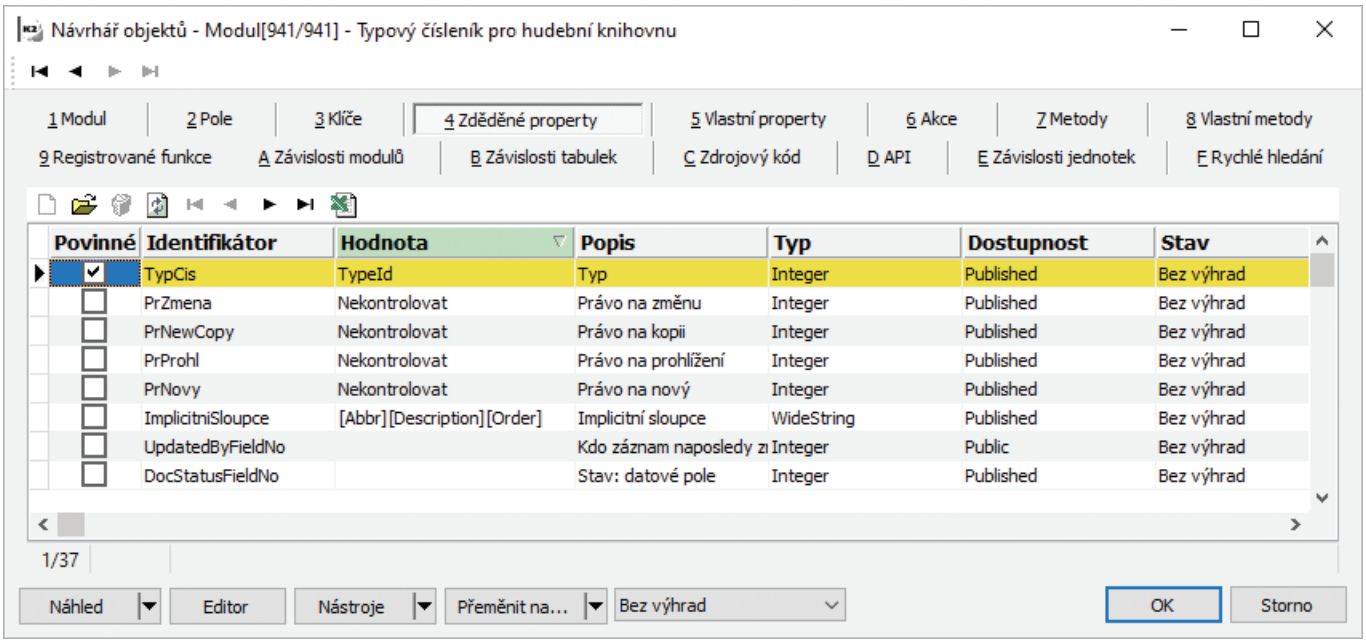
Obrázek 83 - Nastavení pole s typem v typovém číselníku

Dále je nutné nastavit zděděnou vlastnost ***„TypCis“***, do které musíme vložit pole, na kterém máme zatrženu volbu ***„Pole s typem“***, viz Obrázek 84 - Nastavení zděděné vlastnosti TypCis pro typový číselník.



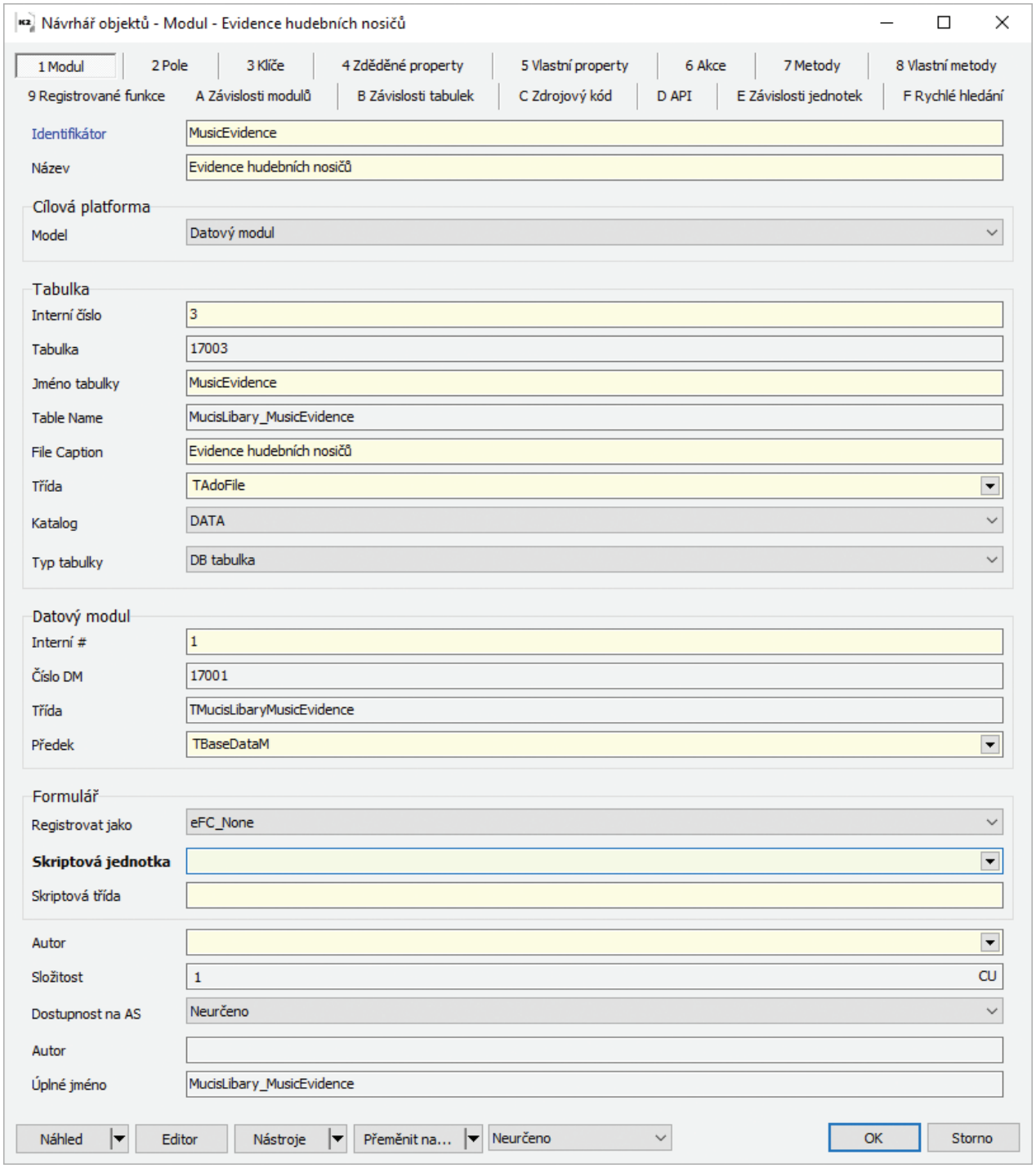
Obrázek 84 - Nastavení zděděné vlastnosti TypCis pro typový číselník

Samozřejmě je nutné nastavit i další vlastnosti, jako implicitní sloupce, právo na prohlížení, zápis apod. Souhrn všech nastavených vlastností je vidět na Obrázek 85 - Nastavení zděděných vlastností typového číselníku.



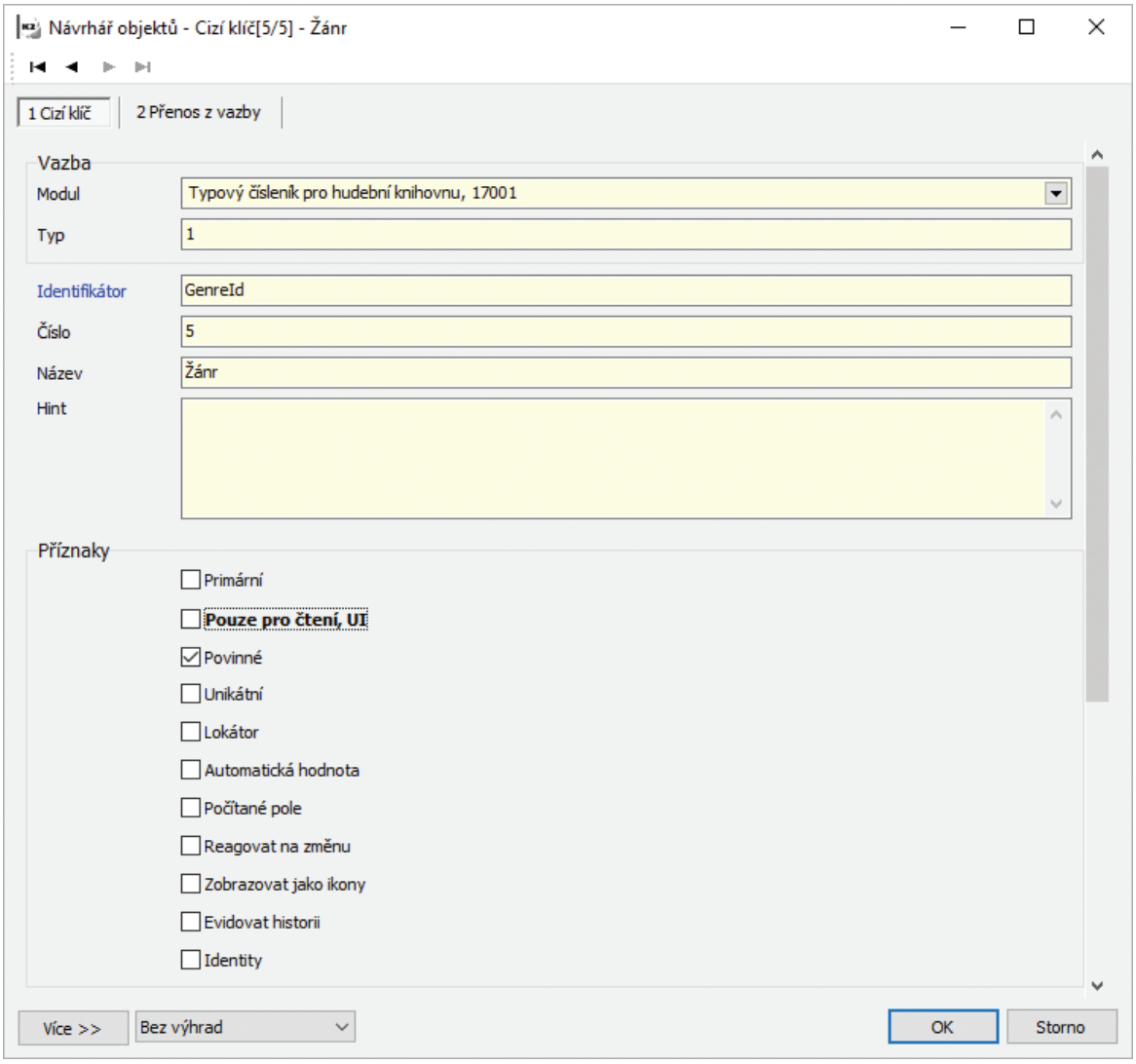
Obrázek 85 - Nastavení zděděných vlastností typového číselníku

Máme nadefinovaný datový modul jako typový číselník. V dalším kroku vytvoříme datový modul, který bude používat několik typů z typového číselníku. Modul nazveme ***„MusicEvidence“*** – ***„Evidence hudebních nosičů“***, který bude klasickým modulem ve výchozím nastavení, viz Obrázek 86 - Datový modul pro evidenci hudebních nosičů.



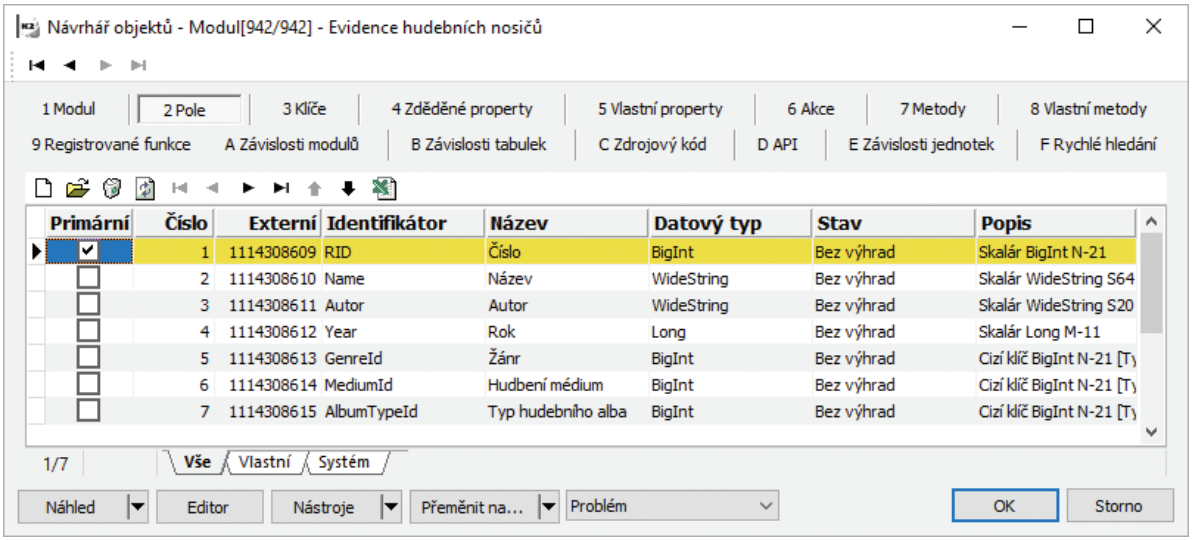
Obrázek 86 - Datový modul pro evidenci hudebních nosičů

Postupně nadefinujeme jednotlivá pole datového modulu. ***„RID“*** jako primární klíč, ***„Name“*** pro název alba, ***„Autor“*** pro autora a ***„Year“*** jako rok vydání hudebního alba. Nakonec nadefinujeme tři cizí klíče, které budou směřovat do vytvořeného typového číselníku z předchozího kroku. Prvním cizím klíčem bude sloužit k evidenci žánru hudebního alba, druhý cizí klíč pak k definici hudebního média a poslední bude sloužit k určení typu alba. Definici provedeme vložením nového pole typu ***„cizí klíč“***, jako modul vybereme vytvořený typový číselník ***„Typový číselník pro hudební knihovnu“***. Návrhář objektů zjistí, že se jedná o typový číselník a nabídne vstupní pole pro definici typu, pod kterým bude tato vazba přistupovat do typového číselníku. V první vazbě, pro žánr, nastavíme ***„1“***, pro hudební médium nastavíme ***„2“*** a u poslední vazby, typ alba, nastavíme ***„3“***. Všechny tři pole (GenreId, MediumId, AlbumTypeId) si dáme jako povinné, viz Obrázek 87 - Definice vazby do typového číselníku.



Obrázek 87 - Definice vazby do typového číselníku

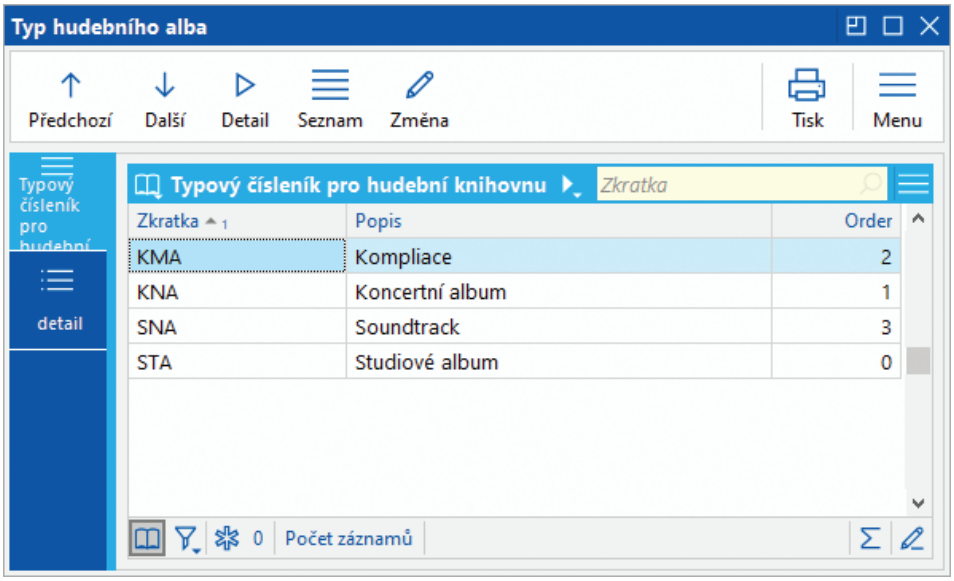
Souhrn všech polí datového modulu je pak vidět na Obrázek 88 - Seznam polí datového modulu pro evidenci hudebních nosičů.



Obrázek 88 - Seznam polí datového modulu pro evidenci hudebních nosičů

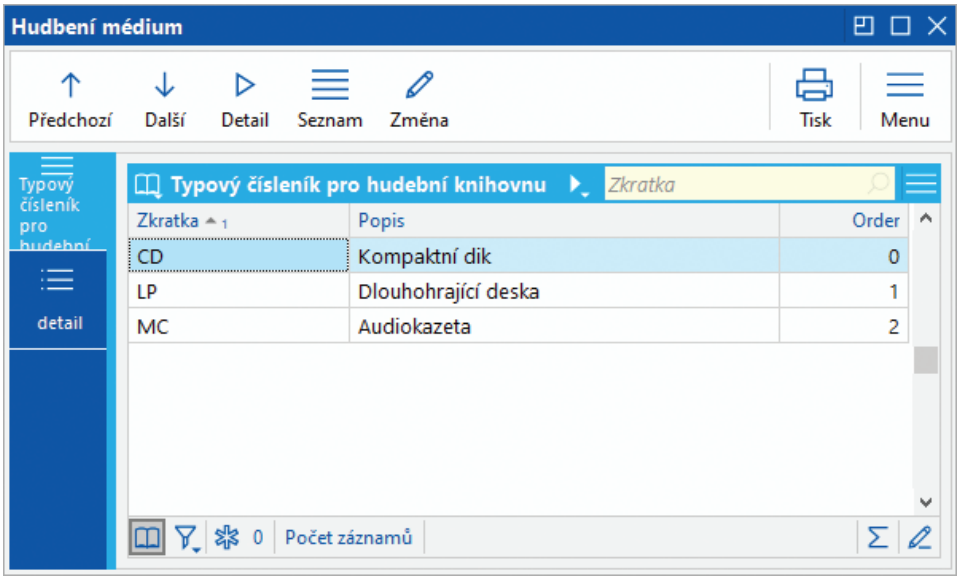
Nesmíme zapomenout nadefinovat i zděděné property pro tento datový modul jako ***„PrNovy“***, ***„PrProhl“***, ***„PrNewCopy“***, ***„PrZmena“*** a ***„Implicitní sloupce“***. Datové moduly máme nadefinovány. Provedeme akci ***„Deploy“*** dle kapitoly 13.1. „Deploy“ – aplikování úprav do K2. Po případném restartování K2 můžeme vyzkoušet nadefinované datové moduly a naplnit je daty. Nejprve vložíme data do jednotlivých typů typových číselníků.

V prvním číselníku ***„Typ alba“*** nadefinujeme hodnoty typu ***„Soundtrack“***, ***„Studiové album“***, ***„Kompilace“*** apod., viz Obrázek 89 - Data v typovém číselníku pro typ alba.



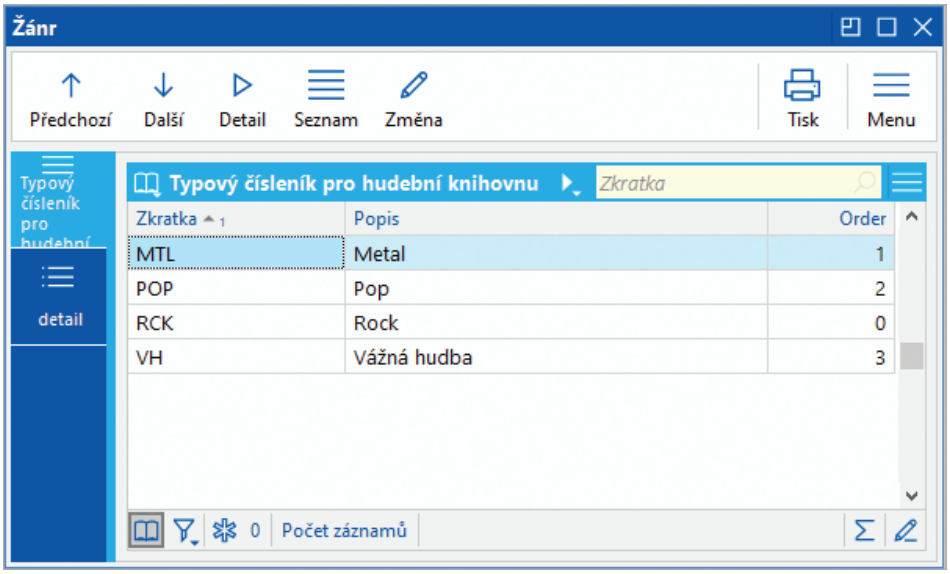
Obrázek 89 - Data v typovém číselníku pro typ alba

V druhém číselníků, ***„Druh média“***, vytvoříme záznamy typu ***„Kompaktní disk“***, ***„Audiokazeta“*** a ***„Dlouhohrající deska“***, viz Obrázek 90 - Data v typovém číselníku pro druh média.



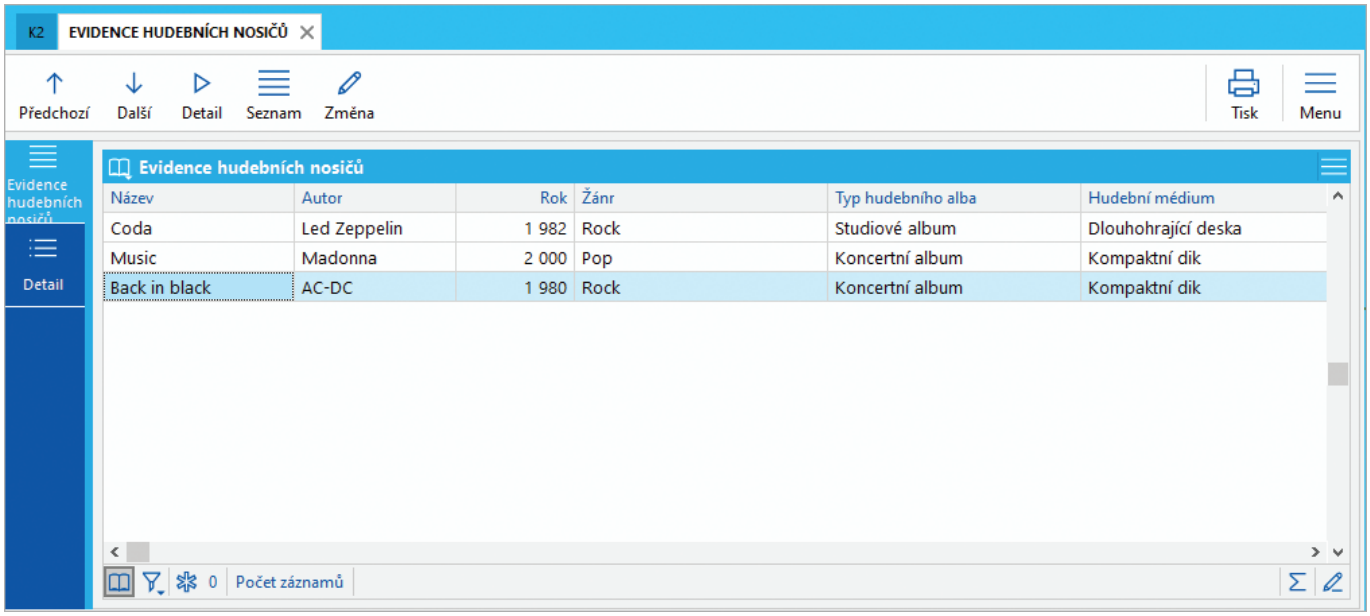
Obrázek 90 - Data v typovém číselníku pro druh média

V posledním číselníku, ***„hudební žánr“***, nadefinujeme hodnoty ***„Metal“***, ***„Pop“***, ***„Rock“*** apod., viz Obrázek 91 - Data v typovém číselníku pro hudební žánr.



Obrázek 91 - Data v typovém číselníku pro hudební žánr

Do datového modulu, ***„Evidence hudebních nosičů“***, pak vložíme záznamy, kde využijeme vazeb do typového číselníků. Ukázka je vidět na Obrázek 92 - Evidence hudebních nosičů - formulář.



Obrázek 92 - Evidence hudebních nosičů - formulář

4.5.3. VLASTNOSTI PŘEDKA TCHILDDATAM

Dalším předkem je ***„TChildDataM“***. Tento předek se používá v případě, že vytváříme datový modul, který je podřízený jinému modulu. Tedy, když vytváříme položky neboli položkový datový modul.

POZNÁMKA: Položkový datový modul vytváříme vkládáním nového pole do nadřízeného modulu, jak již bylo popsáno v předchozích kapitolách. V tomto postupu je předek pro předka ***„TChildDataM“*** nastaven automaticky pro nový podřízený modul.

Při použití získáváte následující výčet vlastností. Vlastnosti, které jsou společné jako v předchozích bázových modulech, zde již popisovat nebudeme.

4.5.3.1. ItemNoField

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit pro číslování položek. Tedy do tohoto pole se bude vkládat údaj o aktuální pozici vkládané položky.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole celočíselného typu.

POZNÁMKA: Jako pole pro pořadí doporučujeme použít s názvem ***„ItemNo"***.

POZNÁMKA: Nastavení výchozího řazení na ***„+[ItemNo]"*** je i není nutný. Operace přesouvání záznamů nahoru/dolů a vkládání za aktuální záznam jsou funkční/povoleny pouze v případě, že jsou záznamy seřazeny dle pole ItemNo.

4.5.3.2. ItemNoOptions

Vlastnosti číslování položek. Zásadním krokem je nastavení na hodnotu inoSplit, pokud chceme docílit možnosti vkládat mezi položky.

4.5.4. VLASTNOSTI PŘEDKA TCUSTOMDATAM

Dalším předkem je ***„TCustomDataM"***. Tento předek se používá v případě, že potřebujeme mít podporu poznámek a dokumentů.

Při použití získáváme následující výčet vlastností. Vlastnosti, které jsou společné jako v předchozích bázových modulech, zde již popisovat nebudeme.

4.5.4.1. NemaPoznamky

Pomocí této vlastnosti určujeme, zda bude datový modul obsahovat podporu ***„poznámek"*** včetně logiky s nimi spojenou. Implicitně je nastaveno, že datový modul obsahuje záložku ***„DTF_Poznámky"***. V případě, že zapneme tento parametr, tedy vypneme poznámky, záložka z formuláře datového modulu zmizí včetně podpory.

Jako hodnotu této vlastnosti vybíráme z variant ***„True"*** - ***„ano"*** / ***„False"*** - ***„ne"***.

4.5.4.2. MaDokumenty

Pomocí této vlastnosti určujeme, zda bude datový modul obsahovat podporu ***„dokumentů"*** včetně logiky s nimi spojenou. V případě, že bude tato volba zapnuta, přibude ve formuláři datového modulu záložka ***„Dokumenty"***.

Jako hodnotu této vlastnosti vybíráme z variant ***„True"*** - ***„ano"*** / ***„False"*** - ***„ne".***

4.5.4.3. MaHistorii

Pomocí této vlastnosti určujeme, zda bude datový modul obsahovat podporu „historie změny dokladů" včetně logiky s ní spojenou. V případě, že bude tato volba zapnuta, přibude ve formuláři datového modulu záložka ***„DTF_HistUcto"*** a ***„DTF_Doklady"***.

Jako hodnotu této vlastnosti vybíráme z variant ***„True"*** - ***„ano"*** / ***„False"*** - ***„ne"***.

4.5.4.4. ConfirmedOnFieldNO

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o datu a času pro každý záznam, který se potvrdil. Tedy, když budeme potvrzovat záznam, logika předka datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží datum a čas, kdy byl záznam potvrzen. Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole musí mít datový typ ***„DateTime“***.

Daná vlastnost je i důležitá pro vlastnost na stornování ***„IsCancelledFieldNo“***, aby stornování fungovalo, protože při stornování dochází i k zápisu, do pole, které je v této vlastnosti (***„ConfirmedOnFiledNO“***).

POZNÁMKA: Potvrdit/odpotvrdit záznam na datovém modulu lze tak, jak jsme zvyklí z dokladů pomocí Alt + F2 / Ctrl + F2. Automaticky se přidá i počítané pole, které vyhodnotí příslušný obrázek např. zeleného zámečku. Pole se jmenuje ***„ConfCancelledCalc“***. K2 si sama v tomto případě ošetřuje i to, že nelze měnit uzamčený záznam.

4.5.4.5. ConfirmedByFieldNO

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k ukládání informace o tom, kdo daný záznam potvrdil. Tedy když, budeme potvrzovat záznam, logika datového modulu zajistí, že do pole definovaného v této vlastnosti se vloží odkaz na uživatele přihlášeného do IS K2, tedy na toho, který záznam potvrdil. Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Pole by mělo být cizím klíčem z datového modulu ***„TK2UserLookupDM“***.

POZNÁMKA: Potvrdit/odpotvrdit záznam na datovém modulu lze tak, jak jsme zvyklí z dokladů pomocí Alt +F2 / Ctrl +F2. Automaticky se přidá i počítané pole, které vyhodnotí příslušný obrázek např. zeleného zámečku. Pole se jmenuje ***„ConfCancelledCalc“***. K2 si sama v tomto případě ošetřuje i to, že nelze měnit uzamčený záznam.

4.5.5. VLASTNOSTI PŘEDKA TCUSTOMLISTDATAM

Dalším předkem je ***„TCustomListDataM“***. Je potomkem předchozího ***„TCustomDataM“***. Přidává podporu pro zneplatňování záznamů, která je dostupná v datovém modulu pod klávesou F8. Po stisku klávesy se záznam stává zneplatněným, případně opět zplatněným.

Při použití získáváte následující výčet vlastností. Vlastnosti, které jsou společné jako v předchozích bázových modulech, zde již popisovat nebudeme.

4.5.5.1. DisableFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit pro zneplatnění záznamu. Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole typu ***„Boolean“*** nebo pole bitů. V případě, že použijeme pole typu ***„Boolean“***, není nutné definovat číslo bitu, které se nastavuje v další vlastnosti ***„DisableBitNo“***.

POZNÁMKA: Automaticky se přidá počítané pole ***„InvalidRecordImgCalc“***, v kterém lze graficky vidět, že je záznam zneplatněný/platný.

4.5.5.2. DisableBitNo

Definuje číslo bitu, který bude sloužit k zneplatnění záznamů. Bit se nachází v poli bitů, které je definováno v předchozí vlastnosti ***„DisableFieldNo“***.

4.5.6. VLASTNOSTI PŘEDKA TCUSTOMDOCUMENTDATAM

Dalším předkem je ***„TCustomDocumentDataM“***. Tento předek se používá v případě, že vytváříte modul, který bude sloužit ve formě dokladu. Tedy díky tomuto předkovi bude existovat podpora knih a období, na které jsme zvyklí z dokladů v K2.

Při použití získáváte následující výčet vlastností. Ty, které jsou společné jako v předchozích bázových modulech, zde již popisovat nebudeme.

4.5.6.1. BusinessYearFieldNumber

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit pro ukládání období pro vytvářený doklad. Nastavení této vlastnosti je povinné.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole, které je typu ***„skalár“*** a je celočíselného typu.

4.5.6.2. HlRada

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit pro ukládání řady pro vytvářený doklad. Nastavení této vlastnosti je povinné.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole, které je typu ***„skalár“*** a je celočíselného typu.

4.5.6.3. UserBook

Pomocí této vlastnosti určíme identifikátor ze seznamu ***„TUserBook“***, ve kterém se bude pamatovat řada, kterou měl uživatel vybranou v rámci dokladu. Nastavení této vlastnosti je povinné.

Hodnotu této vlastnosti vybíráme ze seznamu všech typů ***„TUserBook“***. Je doporučeno, aby hodnota ***„TUserBook“*** odpovídala hodnotě ze stejné oblasti, která je definována ve vlastnosti ***„TBookModule“***. Tedy například máme nastaveno ***„BookModule“*** na prodejní řady, tedy ***„bmSale“***, pak by hodnota ***„UserBook“*** měla být nastavena na ***„ub_PRO“***.

4.5.6.4. BookModule

Pomocí této vlastnosti určíme, z jakého modulu knih je použitá kniha ve vytvářeném dokladu. Nastavení této vlastnosti je povinné.

Jako hodnotu této vlastnosti vybíráme hodnotu ze seznamu dostupných modulů pro řady, které jsou typu ***„TBookModule“***.

Například když vybereme řady s označením ***„bmSale“***, znamená to, že doklady v datovém modulu bude možné ukládat s řadami, které jsou definovány pro prodej. V případě, že chceme založit vlastní množinu řad, která se bude v datovém modulu používat, je potřeba zvolit některý z uživatelských záznamů – ***„bmCustom1“*** až ***„bmCustom5“***. V případě výběru vlastních řad, je potřeba ve správě řad v K2, nadefinovat jednotlivé řady v rámci vybrané hodnoty ***„BookModule“***, viz kapitola 13.2.4. Správa knih.

4.5.6.5. BlockingModule

Tato vlastnost nastavuje modul pro blokace, z které se bere blokovaná perioda. Konstanty jednotlivých modulů nalezneme v unitě ***„BlockingPeriodConst“***.

POZNÁMKA: Pokud jsou nastaveny obě vlastnosti *„BlockingModule“* a *„DocumentDateFieldNo“*, pak se blokuje potvrzování/odpotvrzování/stornování dokladů k datumu dokladu dle nastavení dané blokace.

4.5.6.6. DocumentDateFieldNo

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit k porovnání data s blokovaným datem. Logika předka datového modulu zjistí, že pole definované v této vlastnosti je porovnané s blokační periodou určitého modulu nadefinovaného ve vlastnosti *„BlockingModule“*. Pole může mít datový typ buď *„Date“* nebo *„DateTime“*.

POZNÁMKA: Pokud jsou nastaveny obě vlastnosti *„BlockingModule“* a *„DocumentDateFieldNo“*, pak se blokuje potvrzování/odpotvrzování/stornování dokladů k datumu dokladu dle nastavení dané blokace.

4.5.6.7. BookDocumentTypeID

Slouží k rozlišení typů dokladů v rámci použití jednoho typu řad.

4.5.6.8. IndexByBook

Určuje, který index bude sloužit pro čtení dat přes řady. Index je nutné vytvořit ručně a musí obsahovat segmenty – pole, které představuje řadu dokladu (odpovídá vlastnosti *„HlRada“*), období (odpovídá vlastnosti *„BusinessYearFieldNumber“*), typ (pokud je definován) a číslo (odpovídá vlastnosti *„HlCi“*).

POZNÁMKA: V případě špatně definovaného klíče, nebude datový modul správně fungovat. Návrhář objektů nekontroluje správnost vytvoření klíče.

4.5.6.9. HlCi

Pomocí této vlastnosti určíme, které pole v datovém modulu bude sloužit pro ukládání čísla dokladu. Nastavení této vlastnosti je povinné.

Jako hodnotu této vlastnosti vybíráme pole ze seznamu polí aktuálního datového modulu. Mělo by se jednat o pole, které je celočíselného typu.

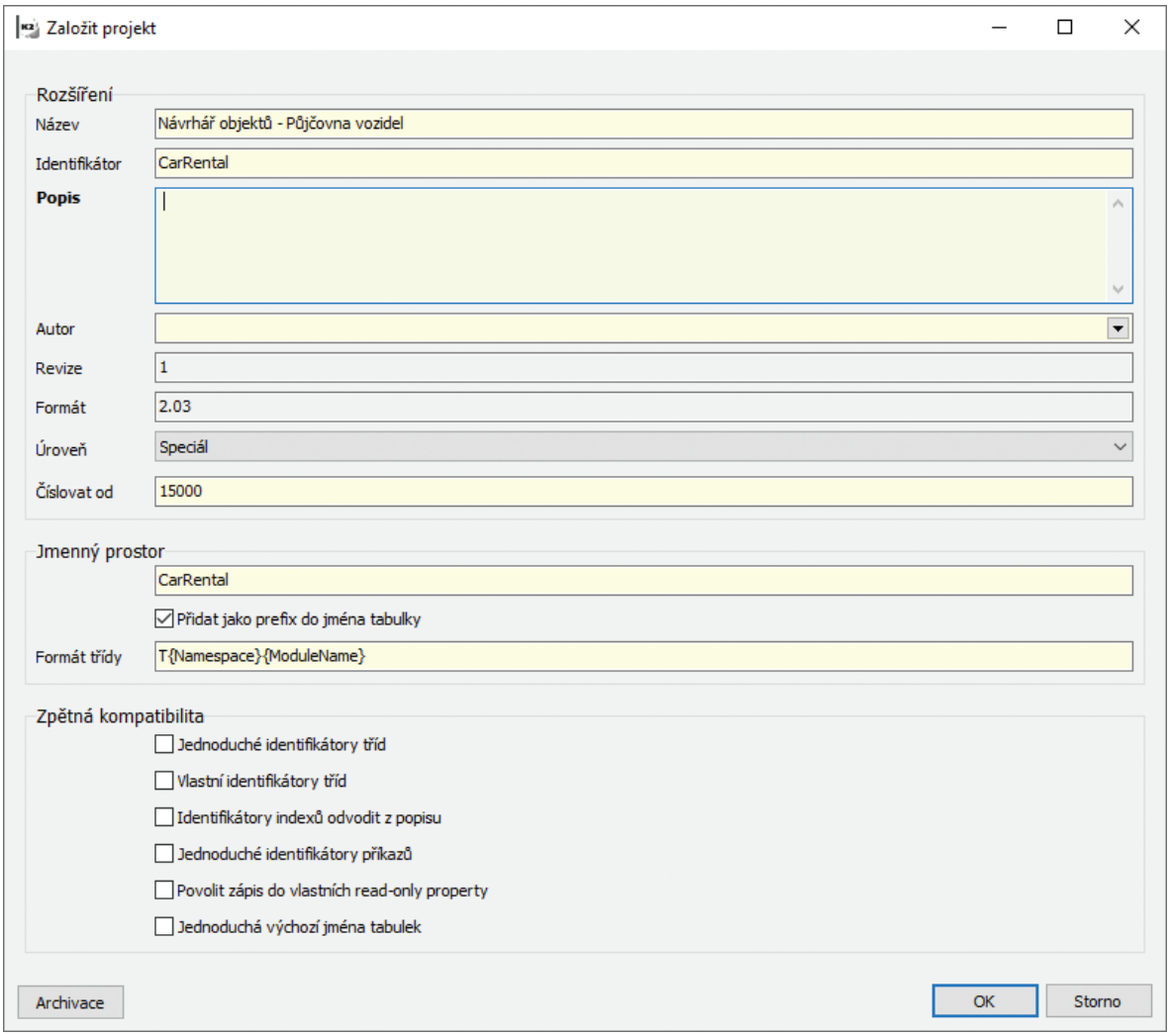
POZNÁMKA: V datovém modulu pro doklad musí existovat pole, do kterého se ukládá číslo dokladu. Pozor, nemůžeme využít pole s primárním klíčem, musíme mít zvlášť definované pole, například *„Num“*.

POZNÁMKA: Pole nesmí mít nastavenu vlastnost *„Povinné“*. Toto pole je automaticky nastaveno na *„Pouze pro čtení“* a jeho hodnota je nastavována jádrem K2 při ukládání. Pokud by bylo nastaveno *„Povinné“* formulář by hlásil chybu při ukládání z důvodu nevyplnění tohoto pole.

PŘÍKLAD: Abychom lépe demonstrovali použití předka *„TCustomDocumentDataM“* popíšeme si v následujícím textu příklad, na kterém ukážeme celé nastavení tohoto typu datového. Příkladem bude *„Půjčovna vozidel“*. Zadání je následující.

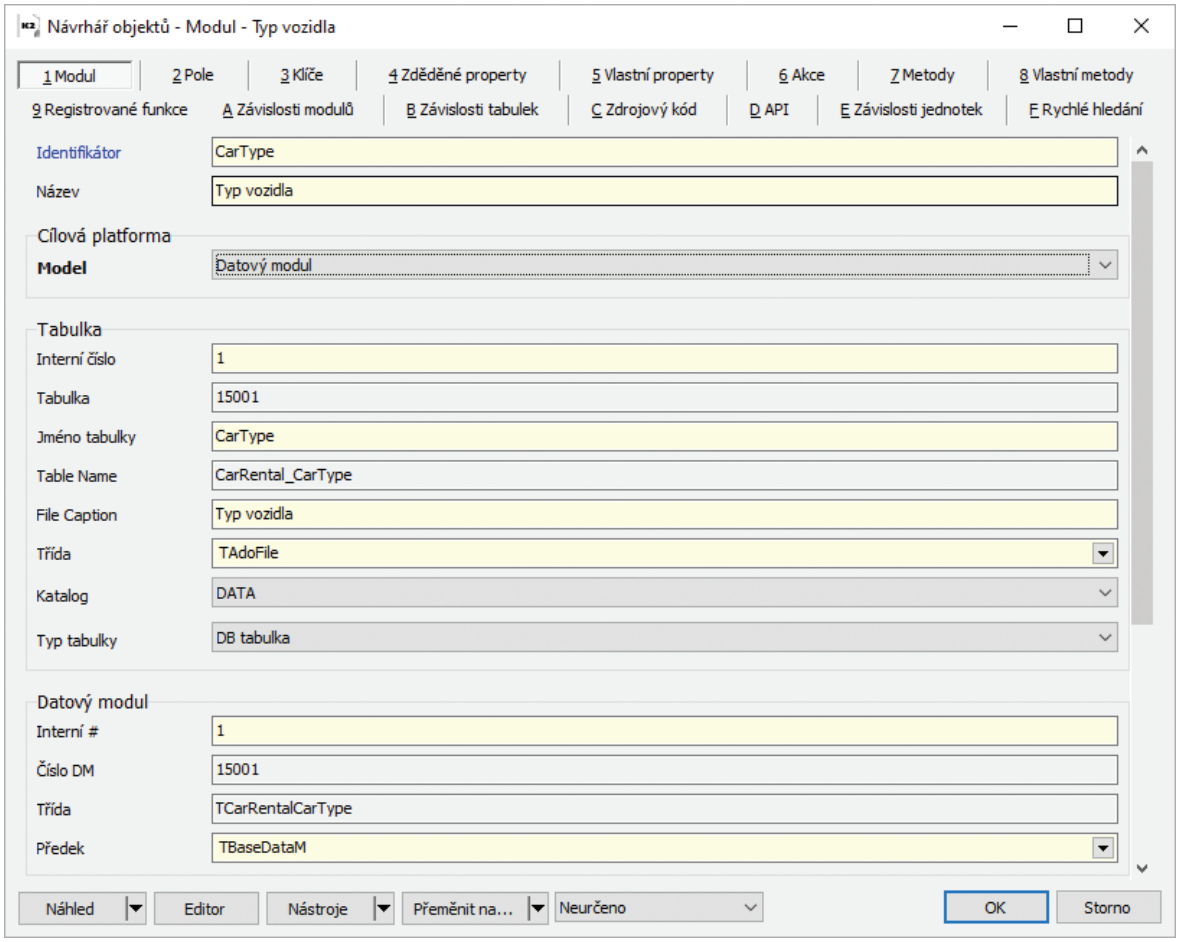
Existuje půjčovna vozidel, která eviduje jednotlivé výpůjčky. Každá výpůjčka obsahuje datum vypůjčení a vrácení, osobu půjčující si vůz, vůz. Výpůjčka bude mít položky, ve kterých budeme evidovat platby půjčených vozů. Evidence bude vedena v několika knihách, které budou sloužit k rozlišení výpůjček z online systému od přímých výpůjček. Celkově budeme potřebovat množina o třech datových modulech – typ vozidla, vozidlo a samotná půjčovna.

Založíme si novou definici rozšíření, kterou nazveme *„Půjčovna vozidel“* s identifikátorem a jmenným prostorem *„CarRental“*. Rozsah číslování nastavíme na 15000, viz Obrázek 93 - Nastavení rozšíření v příkladu Půjčovna vozidel.



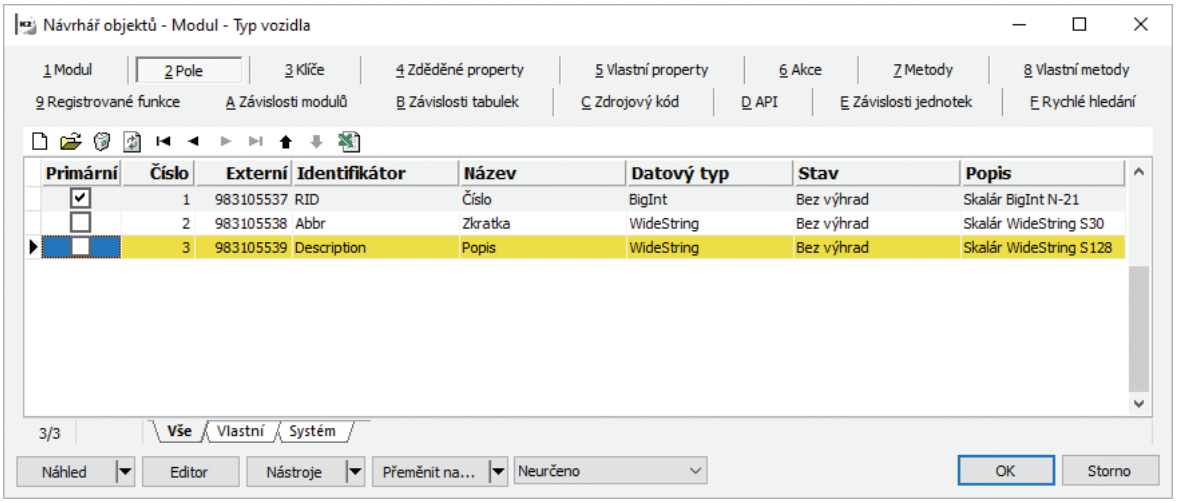
Obrázek 93 - Nastavení rozšíření v příkladu Půjčovna vozidel

Jako první datový modul si vytvoříme *„Typ vozidla“*, který bude sloužit jako číselník. Identifikátor modulu nazveme *„CarType“* a necháme modul ve výchozím nastavení, tedy potomek bude *„TBaseDataM“*, viz Obrázek 94 - Definice datového modulu Typ vozidla v příkladu Půjčovna vozidel.



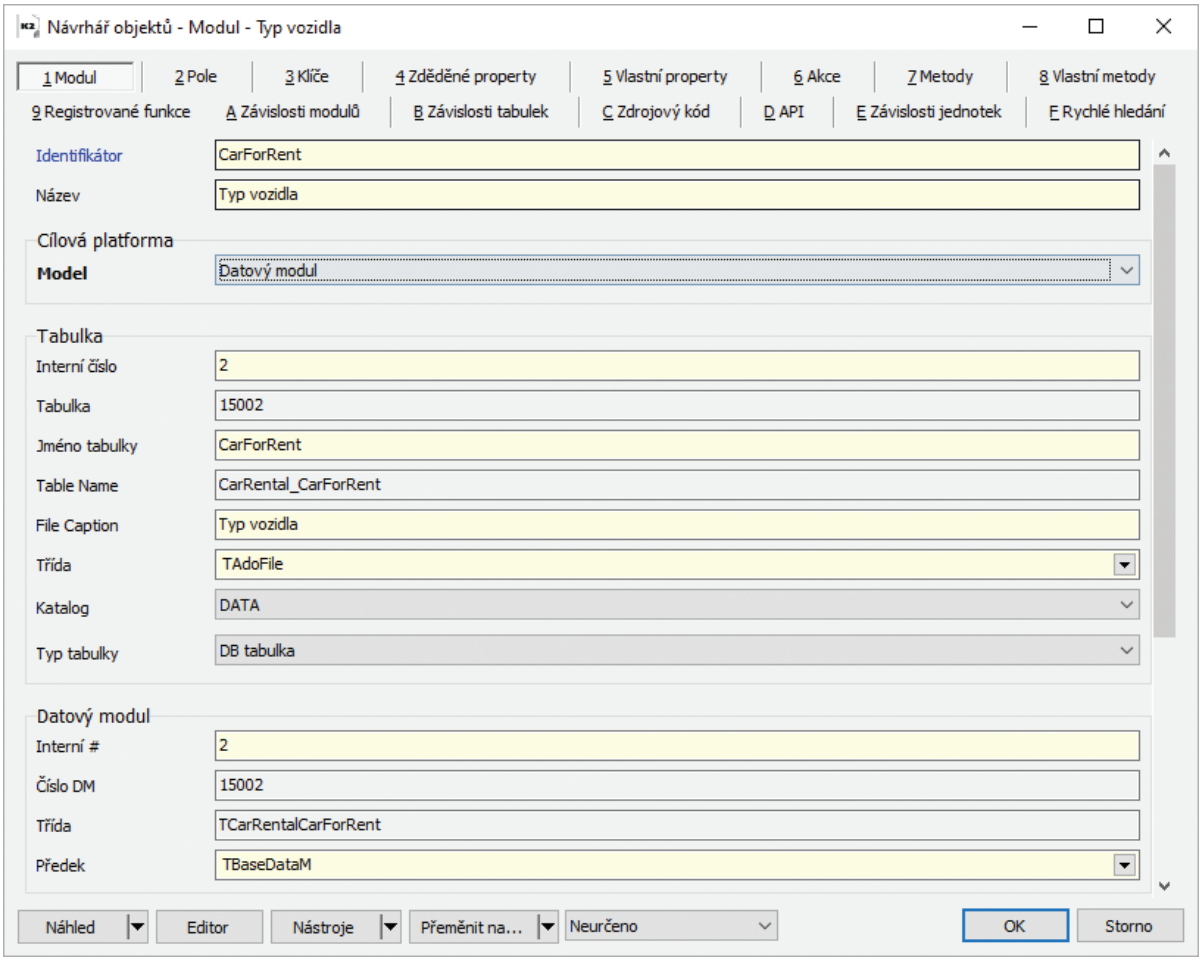
Obrázek 94 - Definice datového modulu Typ vozidla v příkladu Půjčovna vozidel

Datový modul bude mít pouze tři sloupce. ***„RID“*** jako klíčový sloupec, ***„Abbr“*** jako zkratka a ***„Description“*** jako popis typu vozidla, viz Obrázek 95 - Pole v datovém modulu typ vozidla.



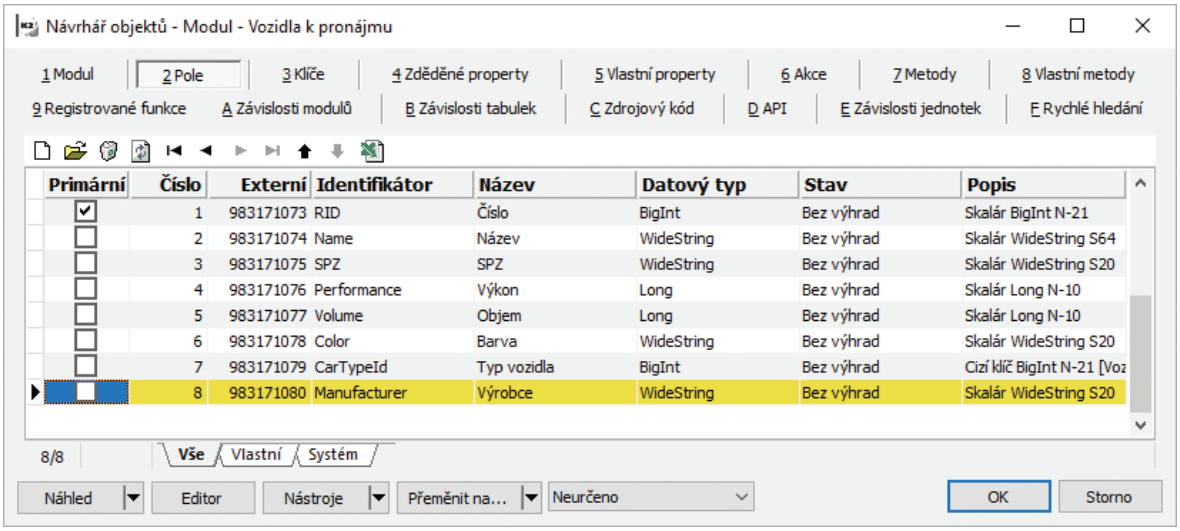
Obrázek 95 - Pole v datovém modulu typ vozidla

Dalším datovým modulem je evidence vozidel, které můžeme pronajímat. Pojmenujeme ho ***„Vozidla k pronájmu“*** a identifikátor nastavíme na ***„CarForRent“***. Ostatní nastavení necháme ve výchozím stavu, tedy bude potomkem ***„TBaseDataM“***, viz Obrázek 96 - Definice datového modulu Vozidla k pronájmu v příkladu Půjčovna vozidel.



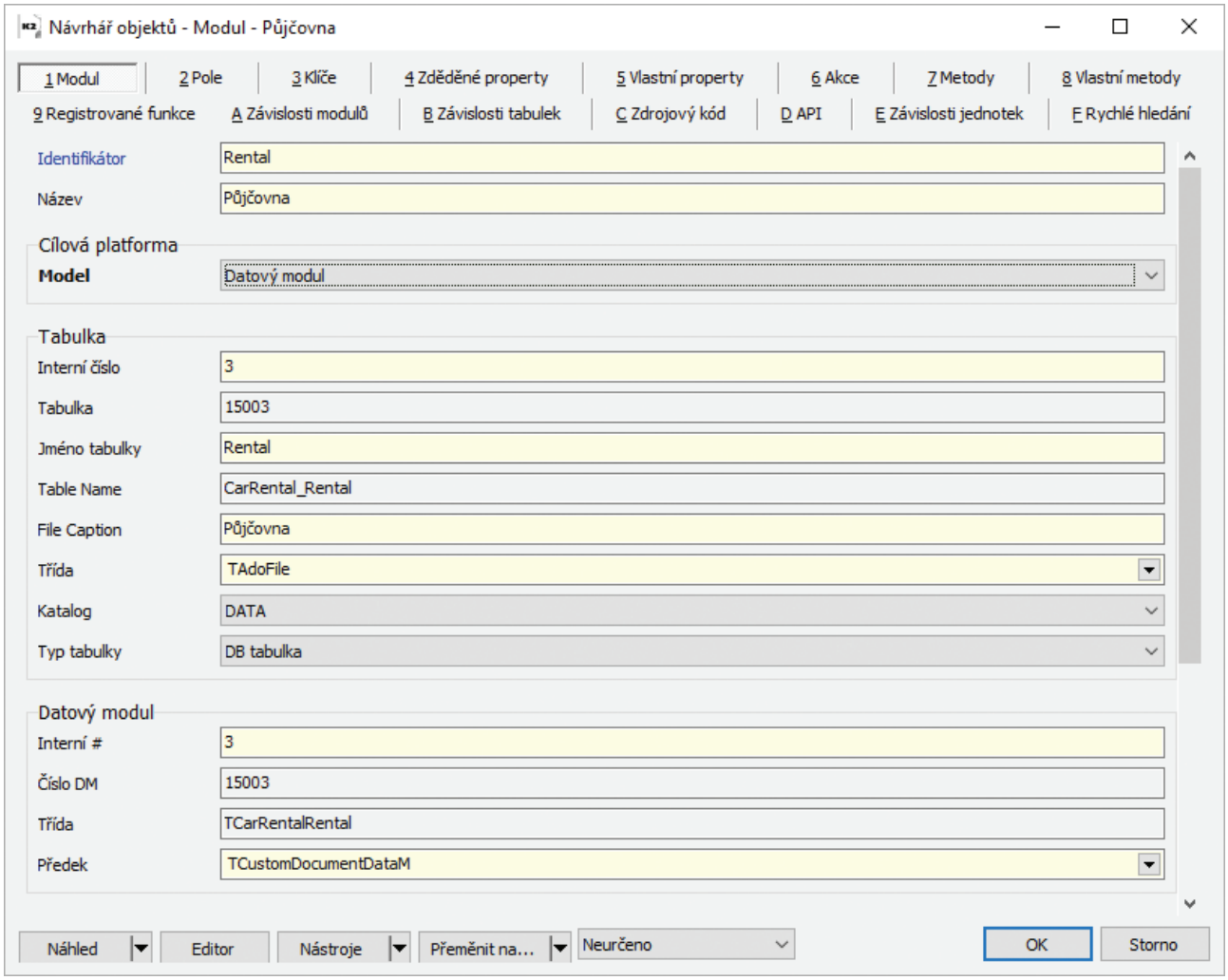
Obrázek 96 - Definice datového modulu Vozidla k pronájmu v příkladu Půjčovna vozidel

V datovém modulu vytvoříme množinu polí, které budou popisovat jednotlivá vozidla. Pole ***„RID“***, které bude jako primární, ***„Name“*** pro název vozu, ***„SPZ“*** jako poznávací značka, ***„Performance“*** pro výkon vozidla, ***„Volume“*** pro objem motoru vozidla, ***„Color“*** pro barvu, ***„Manufacturer“*** pro výrobce a poslední ***„CarTypeId“*** bude cizím klíčem, který bude směřovat do datového modulu ***„Typ vozidel“***, který jsme si vytvořili v předchozím kroku. Seznam všech polí včetně jejich typů je vidět na Obrázek 97 - Pole v datovém modulu Vozidla k pronájmu.



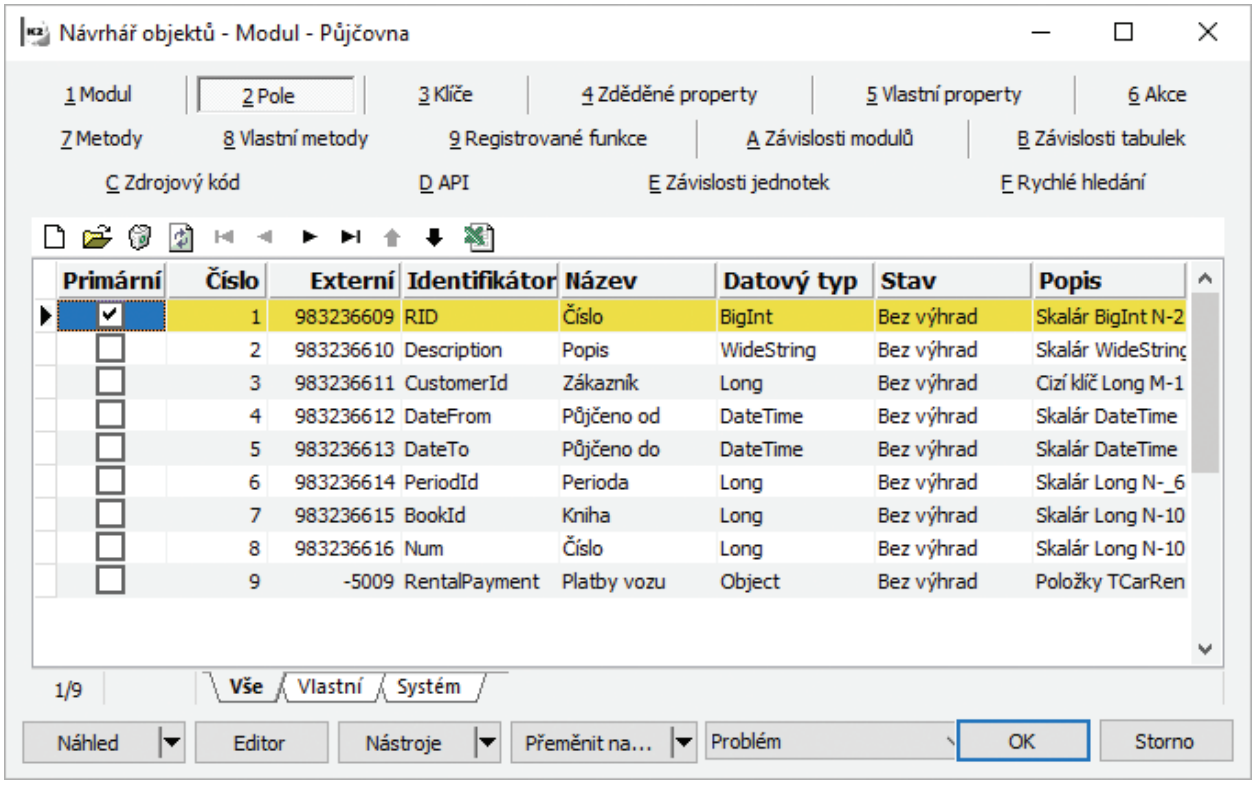
Obrázek 97 - Pole v datovém modulu Vozidla k pronájmu

Posledním datovým modulem je samotná evidence výpůjček jednotlivých vozů. Nazveme ji ***„Půjčovna“*** s identifikátorem ***„Rental“***. Zde využijeme předka ***„TCustomDocumentDataM“***, kvůli kterému tvoříme tento příklad. Základní definici modulu můžeme vidět na Obrázek 98 - Datový modul Půjčovna v příkladu Půjčovna vozidel.



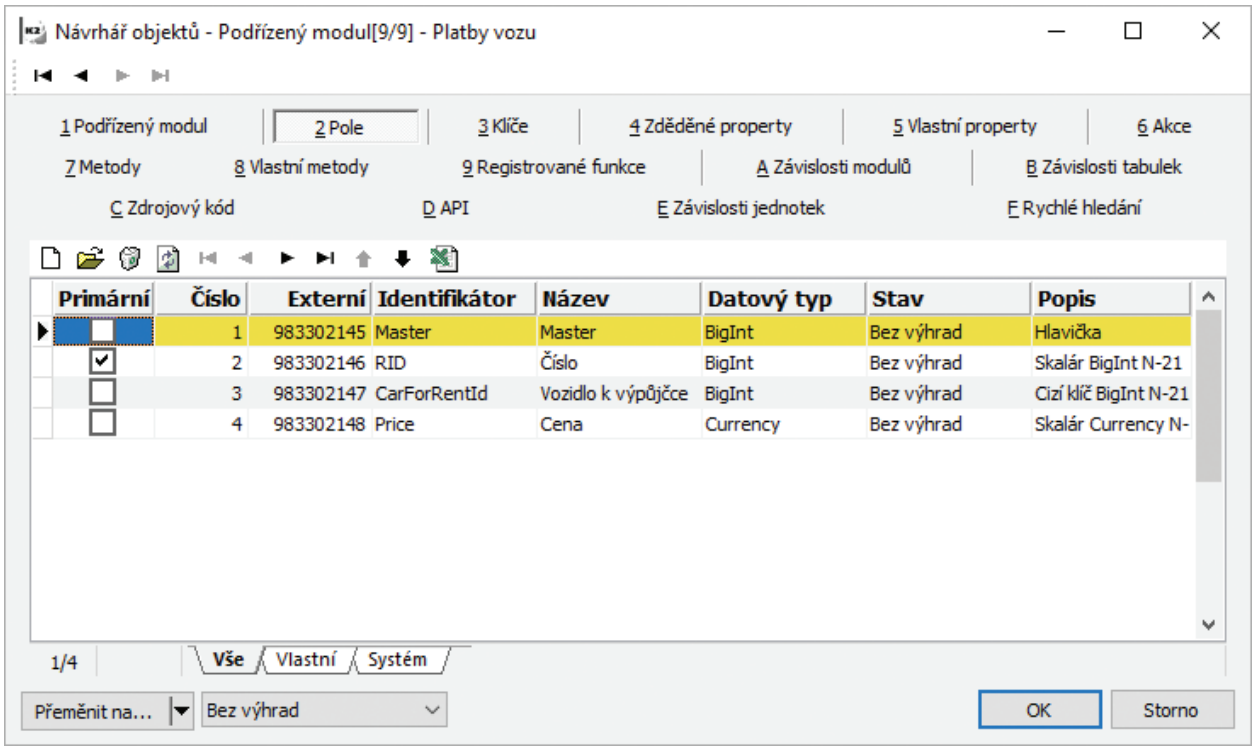
Obrázek 98 - Datový modul Půjčovna v příkladu Půjčovna vozidel

V datovém modulu nadefinujeme sloupce, které budou tvořit evidenci půjčování vozidel. Jedná se o pole ***„RID“***, které je klíčové. Dále ***„Popis“*** pro popis záznamu, ***„CustomerId“***, který je cizím klíčem z datového modulu ***„Kontaktní osoby“*** a slouží k záznamu zákazníka, který si půjčil vozidlo. Pole ***„DateFrom“*** a ***„DateTo“*** k záznamu od kdy do kdy je vozidlo zapůjčeno. ***„PeriodId“***, které bude sloužit k ukládání období dokladu, ***„BookId“*** k ukládání informace do jaké řady patří záznam a nakonec pole ***„Num“***, které slouží k číslování dokladů, tedy jedná se o ***„Číslo dokladu“***. Na závěr vložíme pole, které bude typu položka a bude definovat podřízený datový modul k evidenci plateb výpůjčky vozidla. Pole je nazváno ***„RentalPayment“***. Všechna pole a jejich datové typy můžeme vidět na Obrázek 99 - Seznam polí datového modulu Půjčovna.



Obrázek 99 - Seznam polí datového modulu Půjčovna

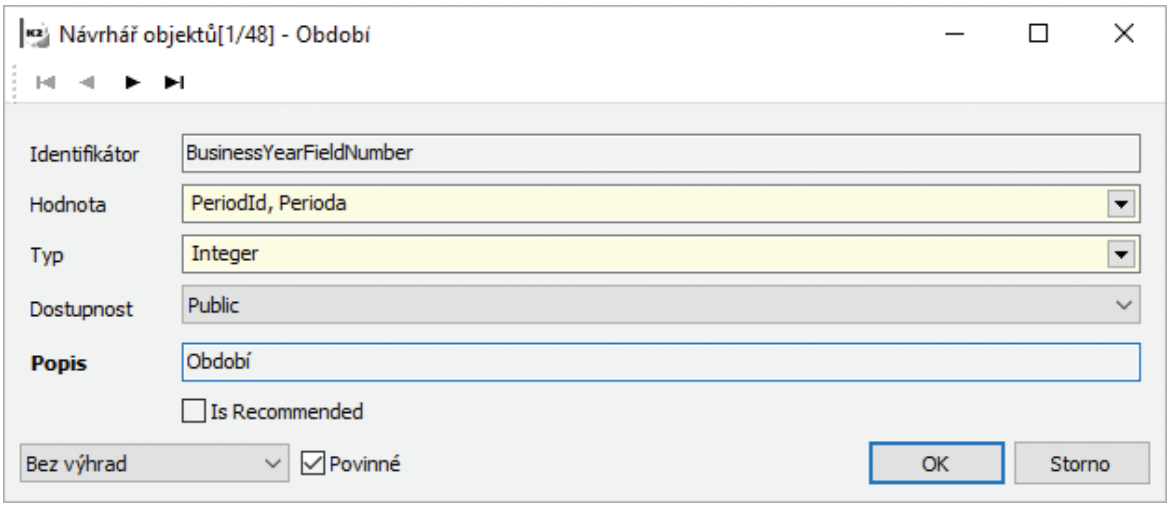
V podřízeném modulu ***„Platby vozu“*** nadefinujeme pole pro evidenci plateb. Jedná se o ***„RID“*** jako primární klíč, automaticky máme doplněno pole ***„Master“*** jako odkaz na hlavičku záznamu, pole ***„CarForRentId“***, které je cizím klíčem z modulu ***„Vozidlo k výpůjčce“***, které jsme definovali výše. Na závěr pole ***„Price“***, které je určeno pro částku k placení za vypůjčené vozidlo. Souhrn všech polí a jejich datových typů je vidět na Obrázek 100 - Seznam polí podřízeného modulu Platba vozu.



Obrázek 100 - Seznam polí podřízeného modulu Platba vozu

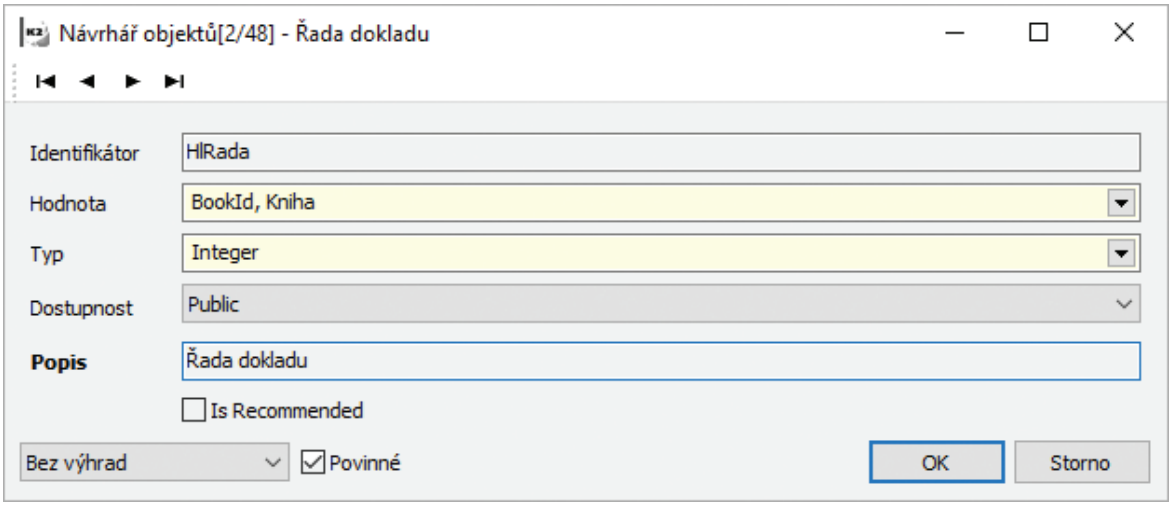
Pole v datovém modulu máme nastavené. Teď musíme přistoupit k dalšímu kroku, kterým je nastavení povinných zděděných vlastností, které jsou nutné ke správnému fungování datového modulu jako doklad.

Prvním z nich je vlastnost ***„BusinessYearFieldNumber“***, která slouží k definici pole, které obsahuje informaci o období dokladu. Tuto vlastnost nastavíme na pole ***„PeriodId“***, které jsme vytvořili v předchozím kroku. Definici vlastnosti můžeme vidět na Obrázek 101 - Vlastnost "BusinessYearFieldNumber" v datovém modulu TCustomDocumentDataM.



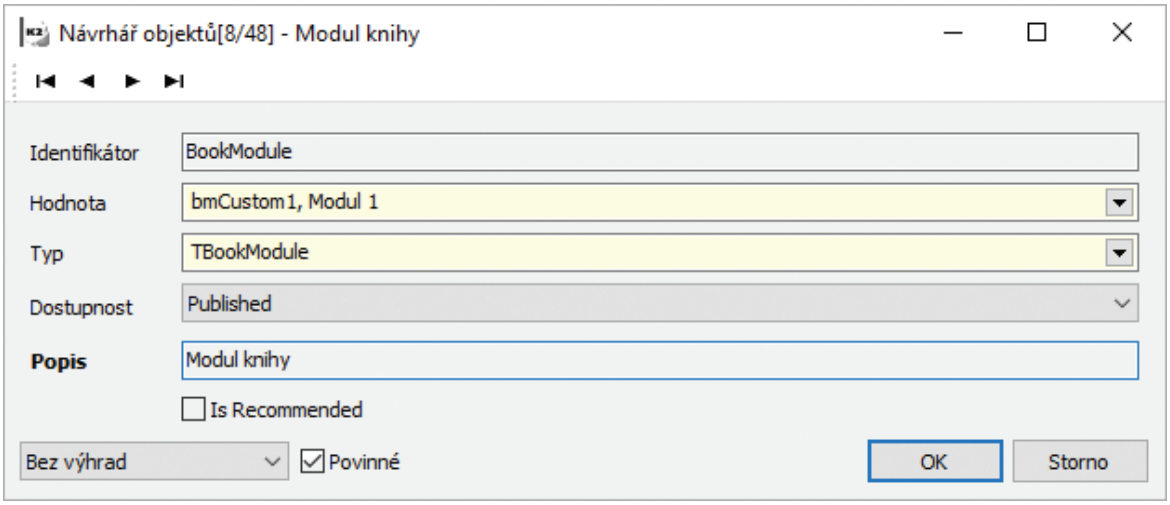
Obrázek 101 - Vlastnost "BusinessYearFieldNumber" v datovém modulu TCustomDocumentDataM

Další vlastností je ***„HlRada“,*** která slouží k definici pole, které obsahuje informaci o řadě dokladu. Vlastnost nastavíme na pole ***„BookId“***, které jsme si vytvořili v předchozím kroku. Definici vlastnosti můžeme vidět na Obrázek 102 - Vlastnost "HlRada" v datovém modulu TCustomDocumentDataM.



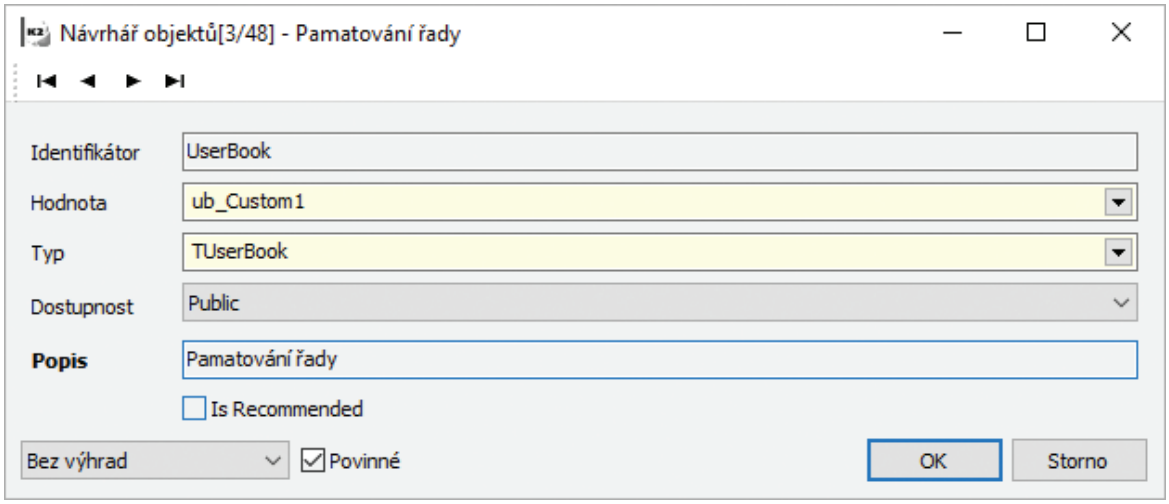
Obrázek 102 - Vlastnost "HlRada" v datovém modulu TCustomDocumentDataM

Definici pole pro knihu máme. Teď musíme nadefinovat do jaké množiny řad doklady budou patřit. K tomuto slouží vlastnost ***„BookModule“***. K výběru máme několik předdefinovaných modulů, které se v K2 běžně používají, například pro nákup nebo prodej. V případě, že uživatel chce vytvořil vlastní množinu knih, musí využít některé z uživatelských. V našem příkladu si nadefinujeme vlastní řady, které budou patřit do vlastní množiny řad. Tedy v této vlastnosti vybereme variantu ***„bmCustom1“***. Definici této vlastnosti můžeme vidět na Obrázek 103 - Vlastnost "BookModule" datového modulu TCustomDocumentDataM.



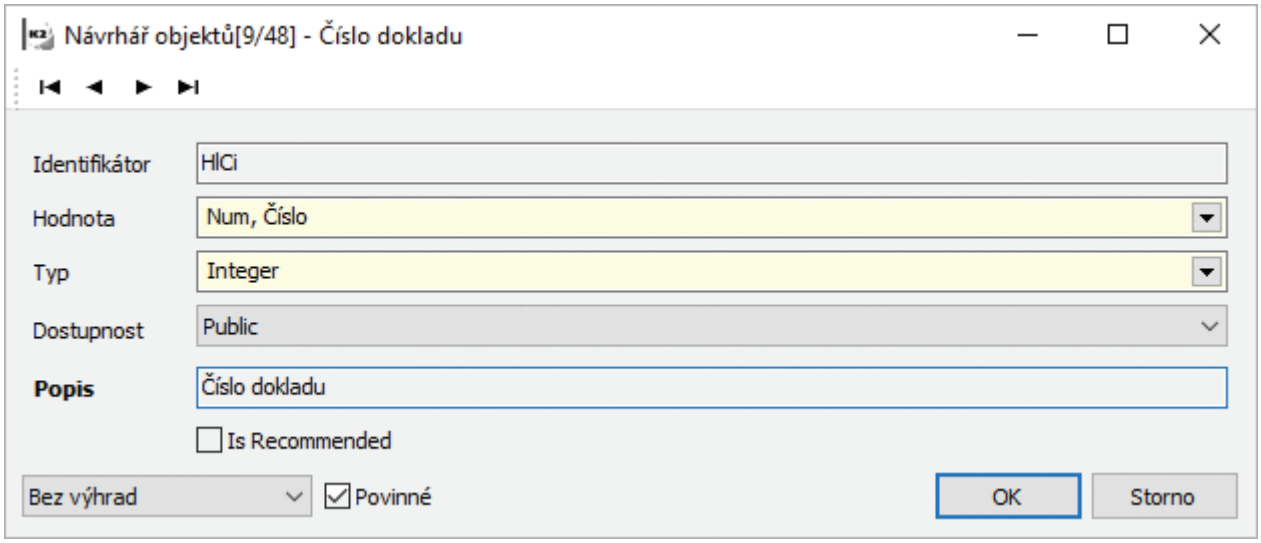
Obrázek 103 - Vlastnost "BookModule" datového modulu TCustomDocumentDataM

Další povinná vlastnost pro doklad je ***„UserBook“***, která slouží k definici proměnné, která se využije k uložení informace, kterou řadu v datovém modulu naposled použil uživatel. Z tohoto místa se pak načítá poslední použitá kniha při opětovném otevření datového modulu stejným uživatelem. Protože jsme se rozhodli, že nepoužijeme předdefinované knihy, ale vytvoříme si vlastní množinu, kterou si později nadefinujeme, je vhodné zde vybrat pro ukládání některý z uživatelských slotů, které jsou připraveny právě pro uživatelské množiny knih. Protože jsme v našem případě využili ***„BookModule“*** s hodnotu ***„bmCustom1“*** (definice předchozí vlastnosti), je vhodné i zde využít hodnotu určenou pro uživatelské knihy 1, tedy „ub_Custom1“. Definice je vidět na Obrázek 104 - Vlastnost "UserBook" datového modulu TCustomDocumentDataM.



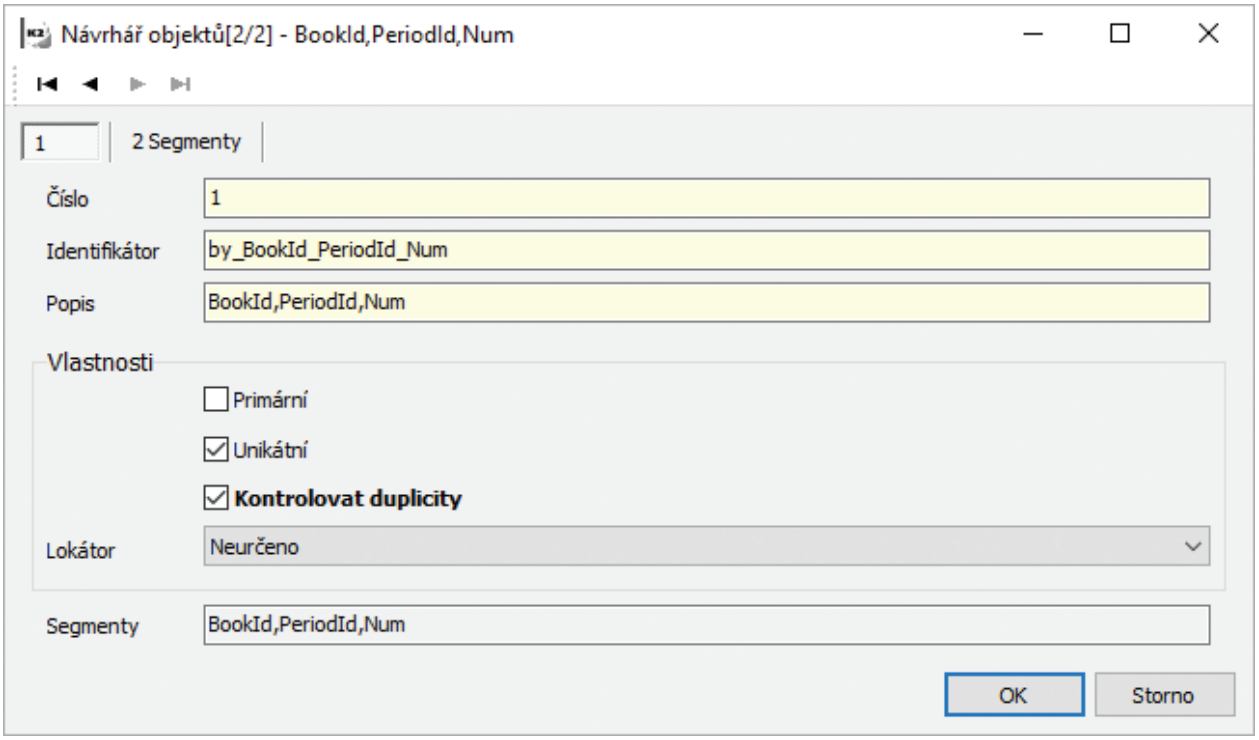
Obrázek 104 - Vlastnost "UserBook" datového modulu TCustomDocumentDataM

Pro nastavení pole, které bude sloužit k automatickému číslování dokladů v rámci období a řady, slouží vlastnost ***„HlCi“***. Ta se musí nastavit na hodnotu pole, do kterého se bude tato informace ukládat. V našem příkladu se jedná o pole ***„Num“***. Nastavení vlastnosti je vidět na Obrázek 105 - Vlastnost "HlCi" na datovém modulu TCustomDocumentDataM.



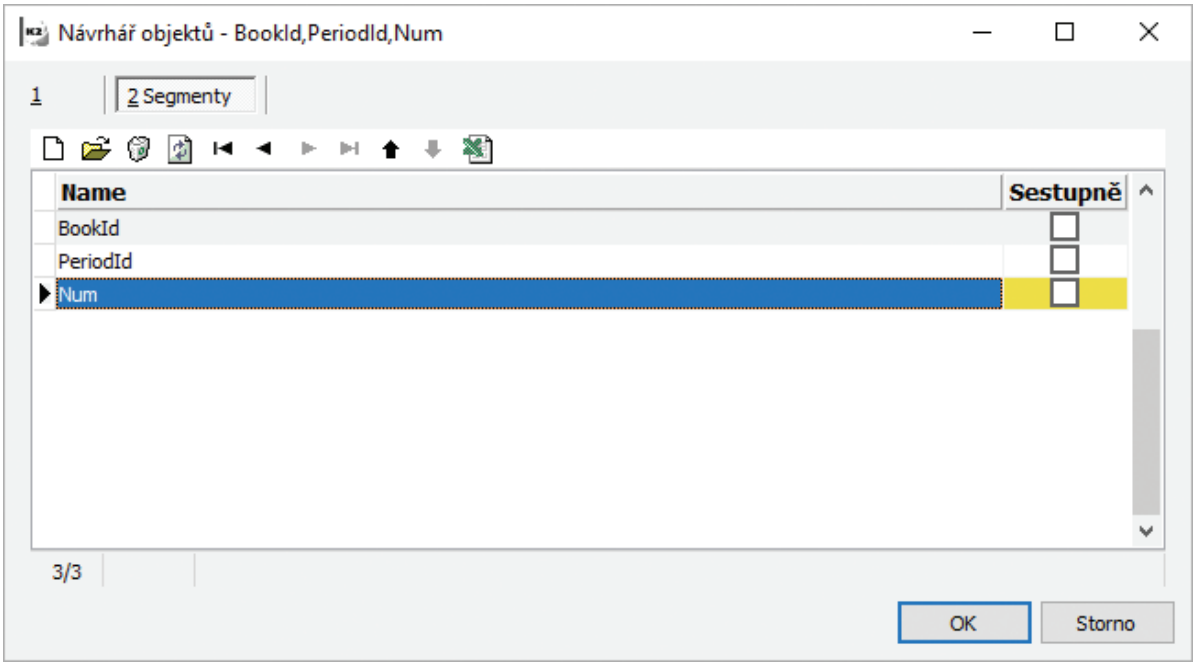
Obrázek 105 - Vlastnost "HlCi" na datovém modulu TCustomDocumentDataM

Poslední povinnou vlastností je ***„IndexByBook“***. Zde musíme označit klíč, který bude využíván k číslování dokladů. Abychom mohli vlastnost nastavit je nejprve nutné takový klíč vytvořit. Přepneme se na záložku ***„Klíče“*** datového modulu. Zde založíme nový klíč, který můžeme vidět na Obrázek 106 - Definice klíče pro číslování dokladů v datovém modulu TCustomDocumentDataM.



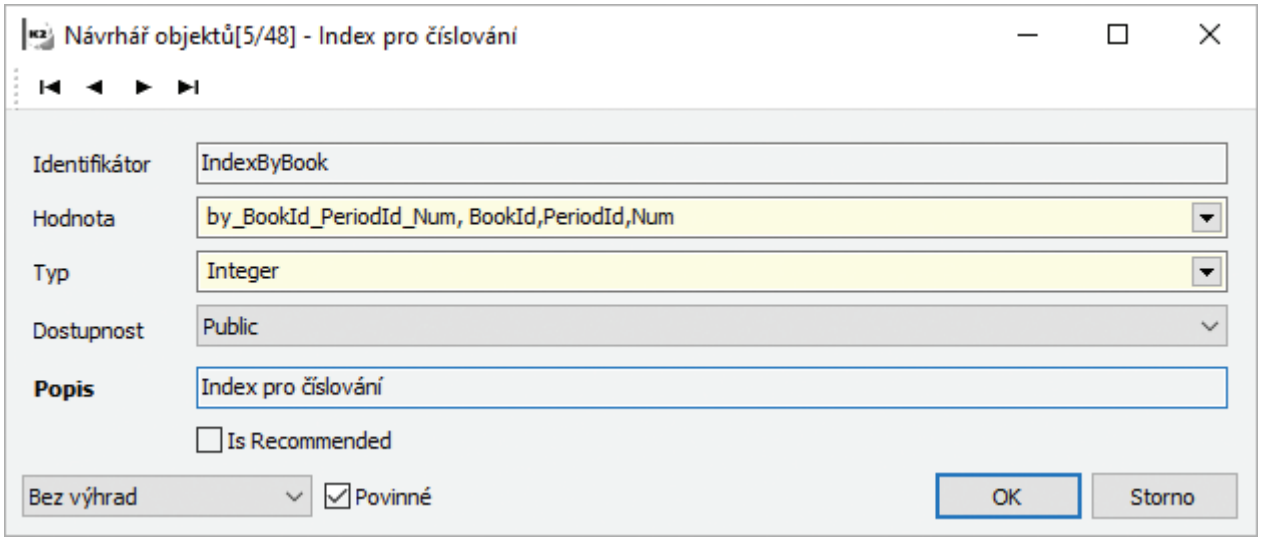
Obrázek 106 - Definice klíče pro číslování dokladů v datovém modulu TCustomDocumentDataM

Důležité v tomto konkrétním klíči je definice jeho segmentů. Musíme ve správném pořadí nastavit jako první segment pole ***„BookId“***, jako druhé pole ***„Periodeld“*** a jako poslední pole ***„Num“***. Tedy pole představující řadu, období, a nakonec číslo dokladu. Definici segmentů klíče můžeme vidět na Obrázek 107 - Definice segmentů klíče pro číslování dokladu v datovém modulu TCustomDocumentDataM.



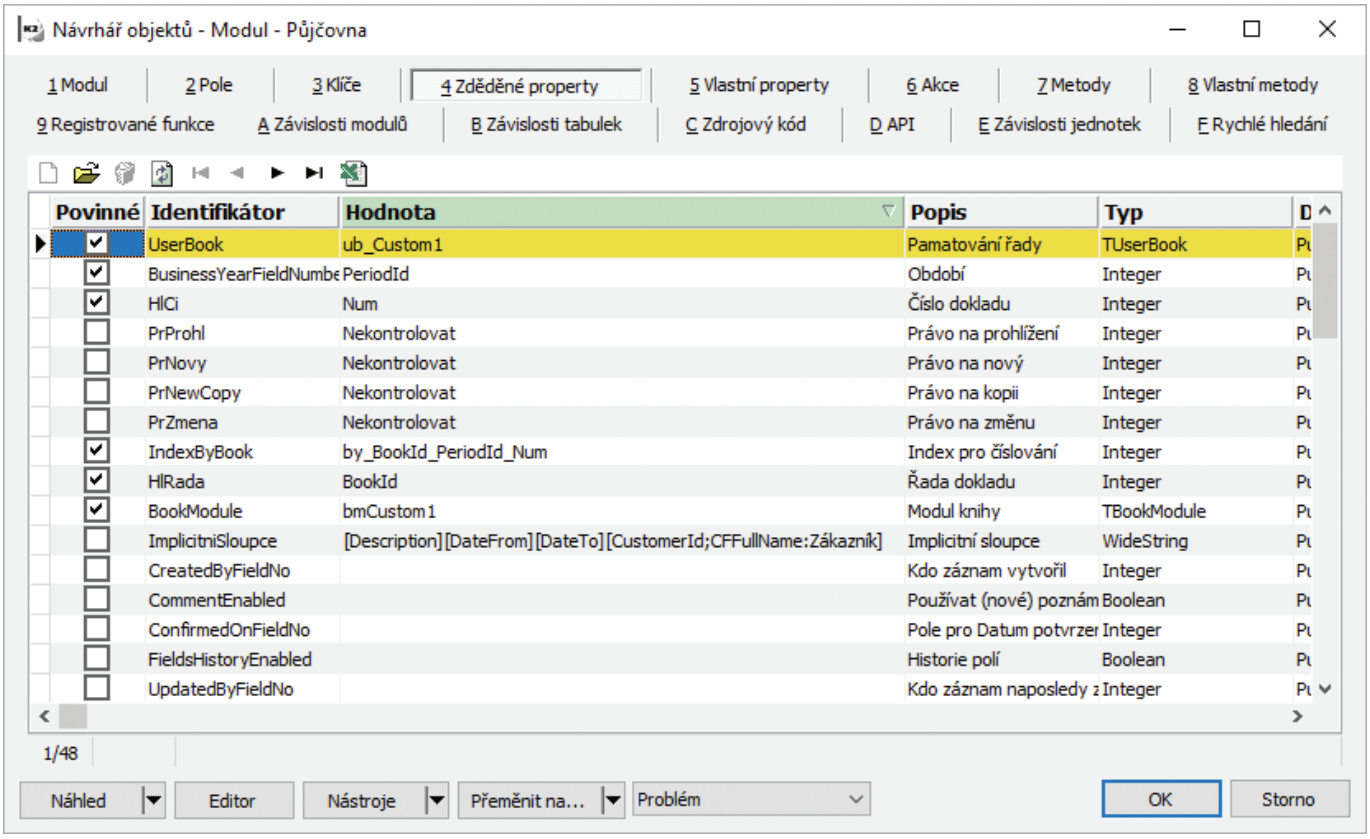
Obrázek 107 - Definice segmentů klíče pro číslování dokladu v datovém modulu TCustomDocumentDataM

Takto vytvořený klíč pak nastavíme do již zmíněné vlastnosti ***„IndexByBook“***. Definici můžeme vidět na Obrázek 108 - Nastavení vlastnosti "IndexByBook" v datovém modulu TCustomDocumentDataM.



Obrázek 108 - Nastavení vlastnosti "IndexByBook" v datovém modulu TCustomDocumentDataM

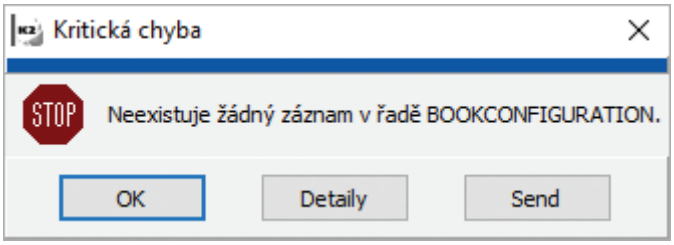
Kompletní nastavení vlastností, které jsou povinné a doporučené pak můžeme pro rekapitulaci vidět na Obrázek 109 - Seznam nastavených vlastností datového modulu TCustomDocumentDataM.



Obrázek 109 - Seznam nastavených vlastností datového modulu TCustomDocumentDataM

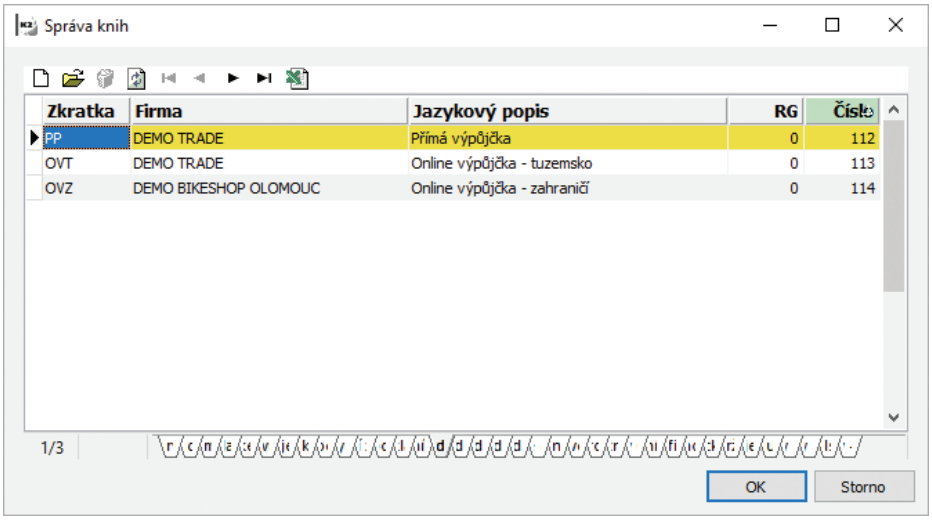
POZNÁMKA: Všechny výše popsané vlastnosti potomka datového modulu ***„TCustomDocument DataM“*** musíme vždy všechny nastavit, abychom zajistili správnou funkčnost datového modulu v režimu doklad. Návrhář objektů hlídá nastavení těchto vlastností. Jsou označeny jako povinné a nedovolí nám uložit datový modul, který by neměl nastavené všechny povinné vlastnosti.

Vlastnosti pro datový modul v režimu doklad máme nastavené. Pokud bychom teď chtěli modul použít, K2 by zahlásila při otevírání tohoto modulu chybu, že neexistuje záznam v ***„Books Configuration“***, viz Obrázek 110 - Chybové hlášení o neexistujícím záznamu v BookConfiguration. Tedy nemáme nastavenou žádnou knihu pro danou množinu knih datového modulu. Musíme otevřít správu knih v K2 a knihy nastavit. Zde můžeme jít standardní cestou nebo využít zkratky přímo z návrháře objektů, kde je pod tlačítkem Nástroje další tlačítko Správa knih, viz kapitola 13.2.4. Správa knih. Po otevření se nám zobrazí formulář, kde si najdeme ve spodních záložkách množinu knih, které jsme se rozhodli použít v našem datovém modulu. Tato množina je definována pomocí vlastnosti ***„BookModule“***. Tuto vlastnost jsme nastavili na ***„bmCustom1“*** což je množina pojmenovaná jako ***„Modul 1“***. V této záložce vytvoříme požadované knihy, pod kterými budeme moct zakládat záznamy v našem datovém modulu.



Obrázek 110 - Chybové hlášení o neexistujícím záznamu v BookConfiguration

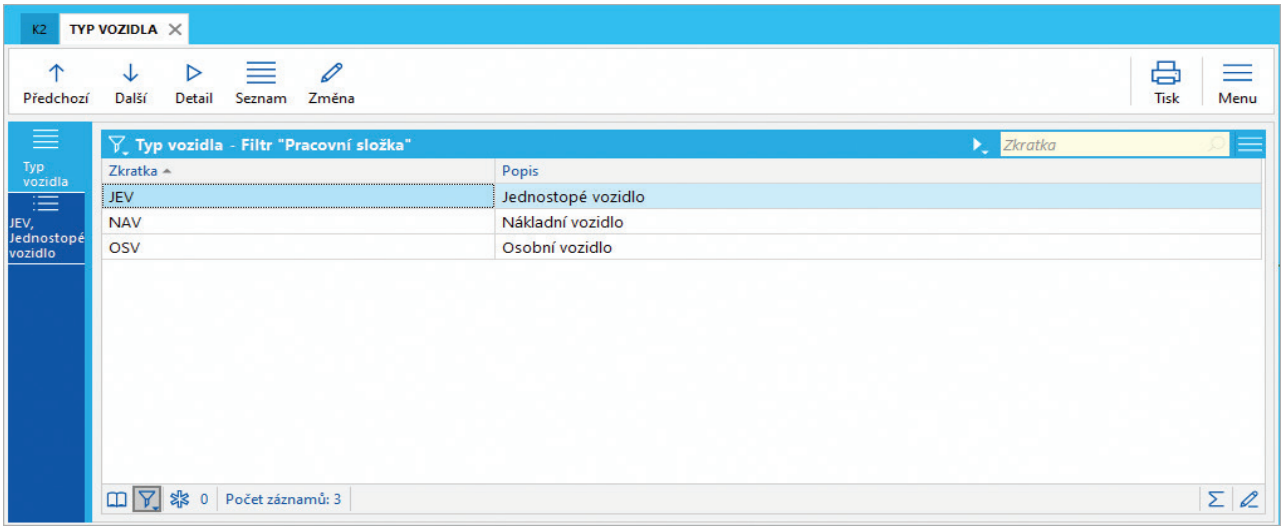
V našem příkladu jsme nadefinovali tři knihy. První jsme nazvali ***„Přímá výpůjčka“*** a je určena vlastní firmě ***„DEMO TRADE“***, druhá je nazvaná ***„Online výpůjčka – tuzemsko“*** určená také pro vlastní firmu ***„DEMO TRADE“***. Poslední kniha je nazvaná ***„Online výpůjčka – zahraničí“***, která je určena pro vlastní firmu ***„DEMO BIKESHOP OLOMOUC“***. Celou definic knih můžeme vidět na Obrázek 111 - Definice vlastních knih v množině Modul 1.



Obrázek 111 - Definice vlastních knih v množině Modul 1

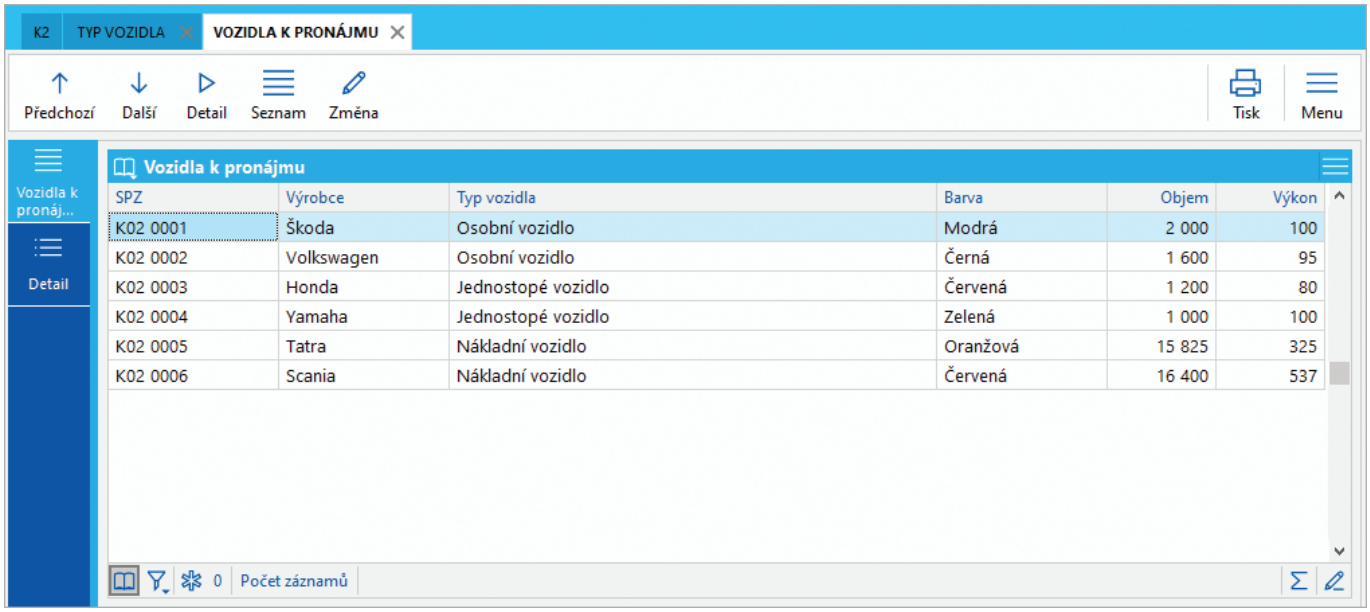
Všechny datové moduly, včetně vlastností a definice knih máme definovány. Teď můžeme provést operaci ***„Deploy“,*** tedy promítnutí změn do K2, viz kapitola 13.1. „Deploy“ – aplikování úprav do K2. Po jejím úspěšném provedení a případném restartu K2 můžeme otevřít jednotlivé moduly a začít je používat.

Nejprve otevřeme číselník typů vozidel a nadefinujeme data. Vložíme si tři záznamy. Jsou to ***„Jednostopé vozidlo“***, ***„Nákladní vozidlo“*** a ***„Osobní vozidlo“***. Definic dat a datový modul můžeme vidět na Obrázek 112 - Datový modul Typy vozidel v příkladu Půjčovna vozidel.



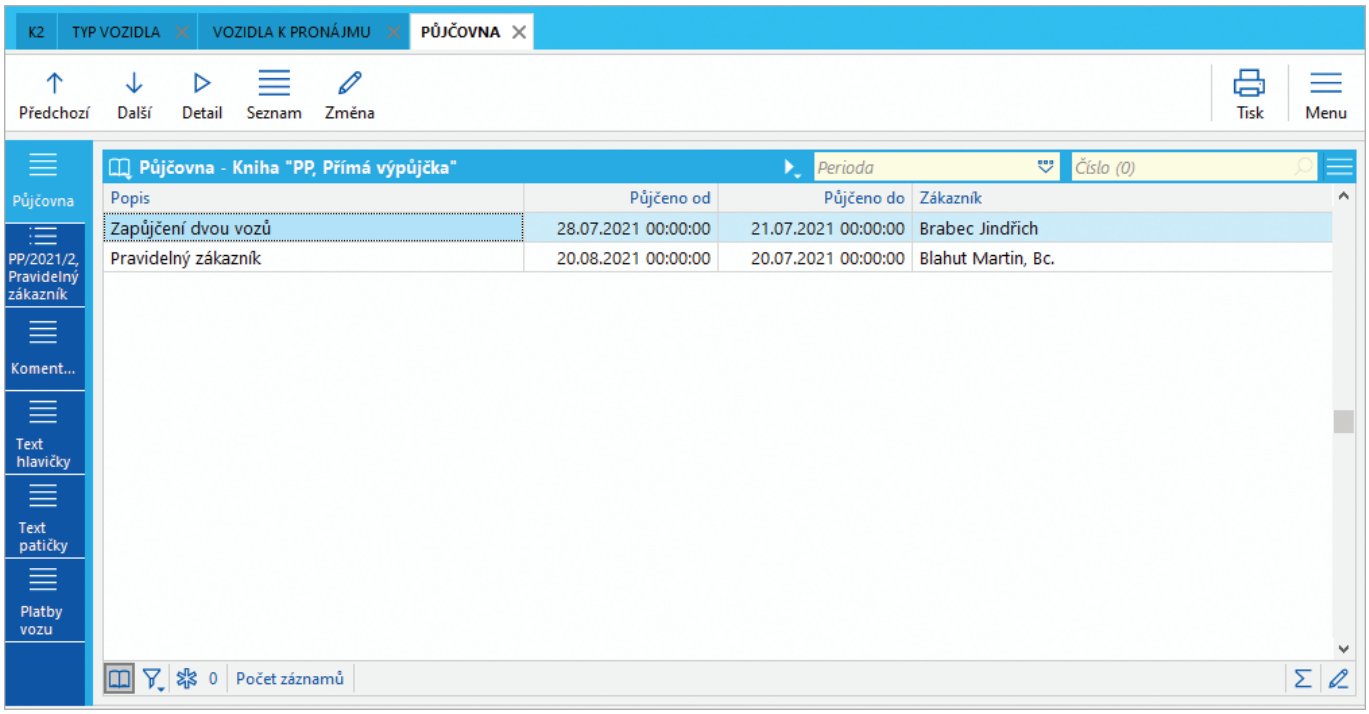
Obrázek 112 - Datový modul Typy vozidel v příkladu Půjčovna vozidel

Dalším datový modulem je pak samotná evidence číselníku vozidel, které, můžeme půjčovat. Otevřeme datový modul ***„Vozidla k pronájmu“*** a nadefinujeme několik záznamů pro vozidla kde u každého využijeme typu, který jsme nadefinovali v předchozím kroku. Seznam vozidel, které můžeme půjčovat můžete vidět na Obrázek 113 - Datový modul Vozidla k pronájmu v příkladu Půjčovna vozidel.



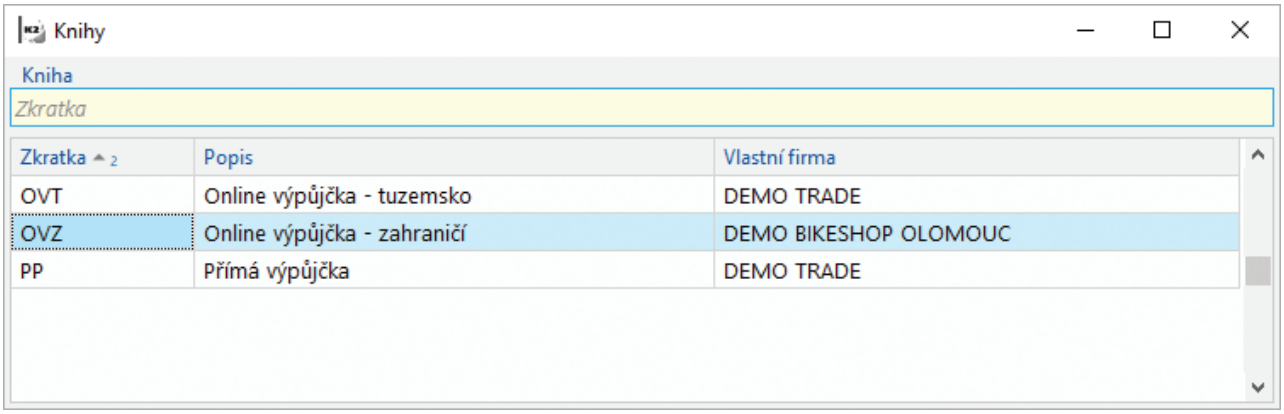
Obrázek 113 - Datový modul Vozidla k pronájmu v příkladu Půjčovna vozidel

Posledním modulem je samotná půjčovna. Zde již můžeme evidovat vozidla, která půjčíme konkrétním osobám. Otevřeme si nový datový modul ***„Půjčovna“***. Modul se nám otevře v režimu kniha a bude mít nastavenou první knihu z našeho seznamu definovaných knih. V této knize můžeme začít zakládat záznamy. Na Obrázek 114 - Datový modul Půjčovna v příkladu Půjčovna vozidel můžeme vidět dvě evidované výpůjčky, které jsou evidovány v knize ***„PP“*** což je kniha pro ***„Přímý prodej“***.



Obrázek 114 - Datový modul Půjčovna v příkladu Půjčovna vozidel

V případě, že bychom chtěli knihu změnit, postupujeme stejně jako kdekoliv jinde v K2. To znamená klávesou **Alt + F10** nebo tlačítkem pod seznamem záznamů. Po vyvolání změny se nám zobrazí seznam knih z naší množiny knih. Zde můžeme knihu změnit a pokračovat dále v práci. Seznam dostupných knih můžeme vidět na Obrázek 115 - Výběr knihy pro datový modul Půjčovna v příkladu Půjčovna vozidel.



Obrázek 115 - Výběr knihy pro datový modul Půjčovna v příkladu Půjčovna vozidel

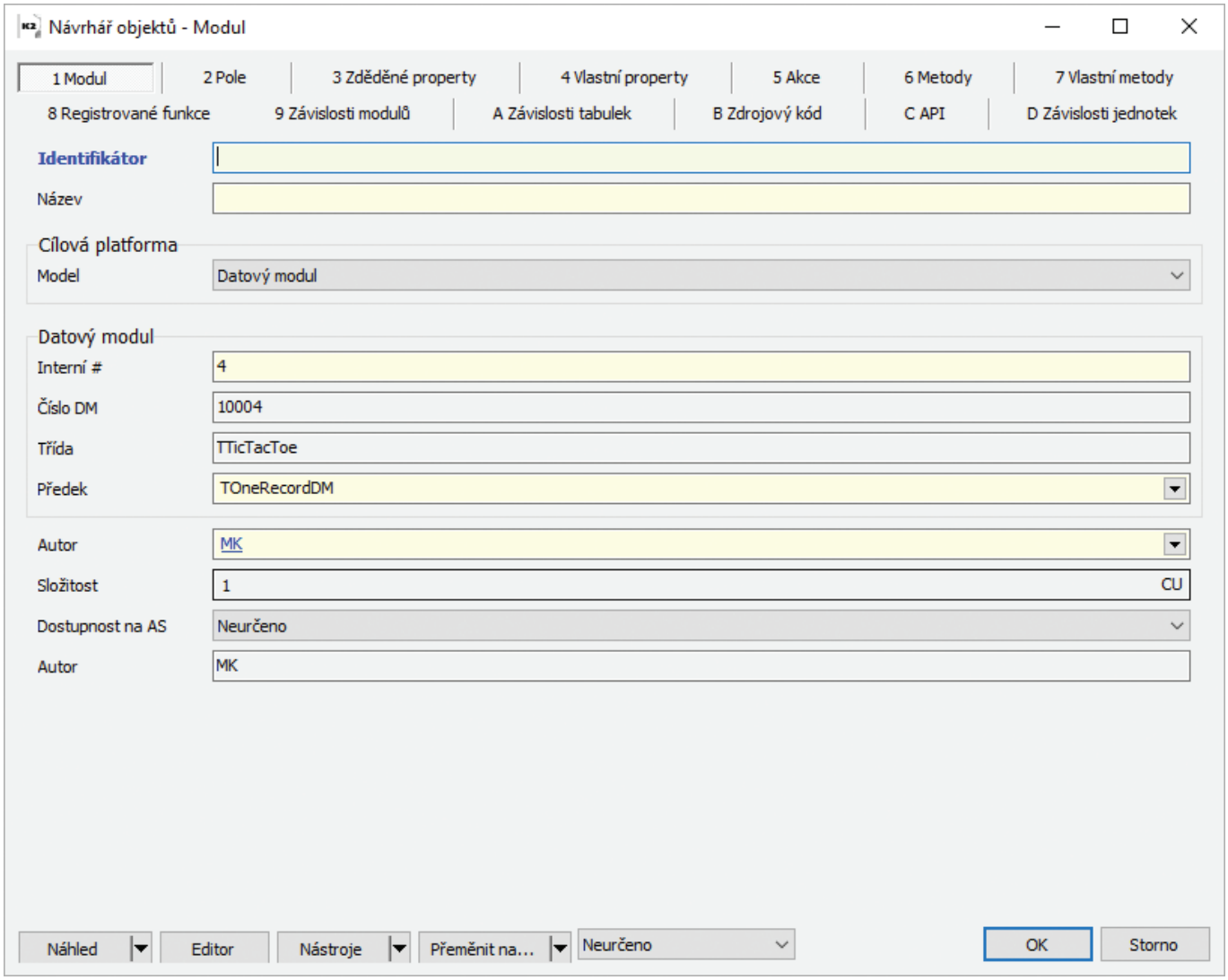
4.5.7. VLASTNOSTI PŘEDKA TONERECORDDM

V případě, že potřebujeme vytvořit datový modul, který bude sloužit jako editační nad jedním záznamem, tak jak tomu bylo ve formulářích K2_DDFM, pak je možné využít tento tohoto předka pro nový datový modul.

Takový datový modul je pak automaticky typu ***„paměťový“***, tedy neexistuje pro jeho data databázová tabulka. Veškeré naplnění daty provádíme v paměti pomocí vlastní implementace.

Pro nastavení výchozích hodnot jednoho záznamu v datovém modulu pak slouží metoda z předka ***„DefaultValues“***, která je popsána v kapitole zabývající se metodami datového modulu. Její použití bude vysvětleno i zde na příkladu.

V případě, že nastavíme předka datového modulu na ***„TOneRecordDM“***, pak se automaticky skryjí nepotřebná pole a formulář se zjednoduší, viz stejně tak je omezena i možnost nastavení zděděných property datové modulu.



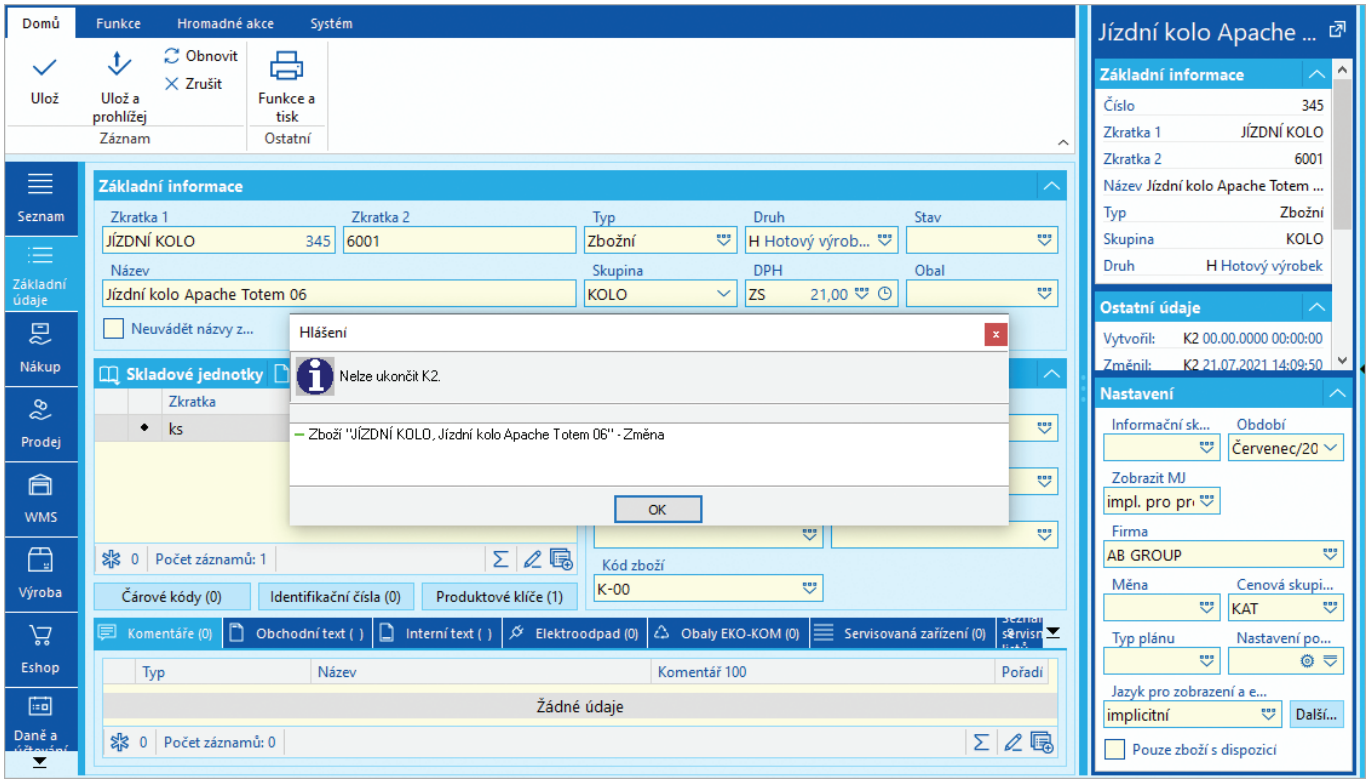
Obrázek 116 - Nastavení datového modulu - předek TOneRecordDM

POZNÁMKA: Podpora pro „TOneRecordDM“ datové moduly je implementována pouze v univerzálních formulářích.

Předek TOneRecordDM má oproti předchozím jednu zděděnou vlastnost navíc.

4.5.7.1. CloseWithoutQuery

Při pokusu o uzavření aplikace K2 se provádí kontrola na rozpracované záznamy ve všech otevřených modulech. Například, pokud máme rozpracovaný nový záznam v modulu ***„Zboží“***, nedovolí K2 aplikaci ukončit. Uživateli se zobrazí hlášení, viz Obrázek 117 - Hlášení o rozpracovaném záznamu při ukončení K2.



Obrázek 117 - Hlášení o rozpracovaném záznamu při ukončení K2

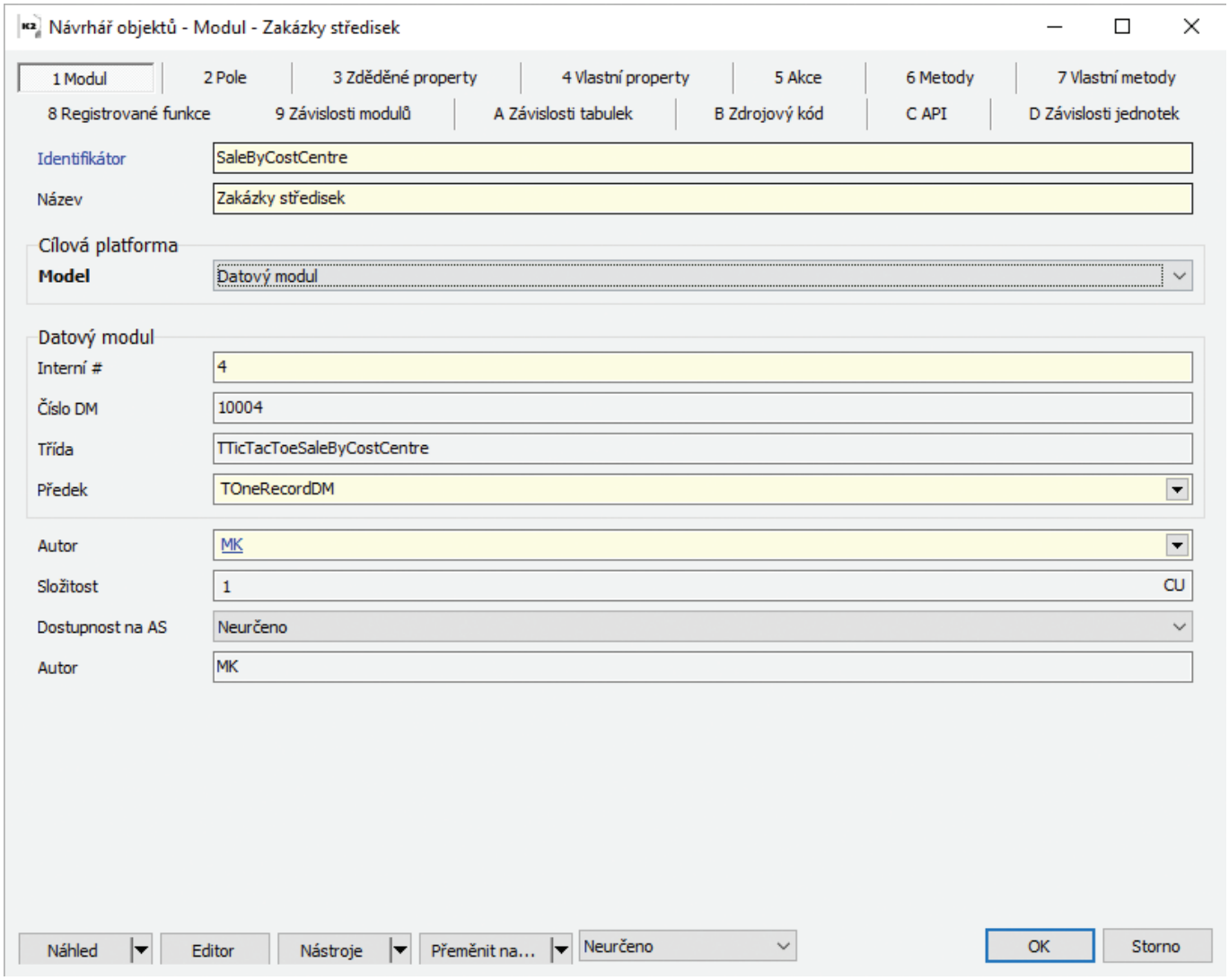
V případě modulu, který je potomkem ***„TOneRecordDM“*** je každý záznam vždy ve stavu ***„Nový / Změna“***. Tedy dokud s ním pracujeme, K2 nelze ukončit. Protože se ve spojení s tímto typem modulu často jedná pouze o operace, které pracují v paměti a neprovádí žádné zápisy do databáze a neblokují jiné uživatele, je žádoucí potlačit tuto kontrolu nastavením vlastnosti na ***„True“***.

Jako hodnotu této vlastnosti vybíráme z variant ***„True“*** - ***„ano“*** / ***„False“*** - ***„ne“***.

Použití ***„TOneRecordDM“*** si nejlépe demonstrujeme na příkladu.

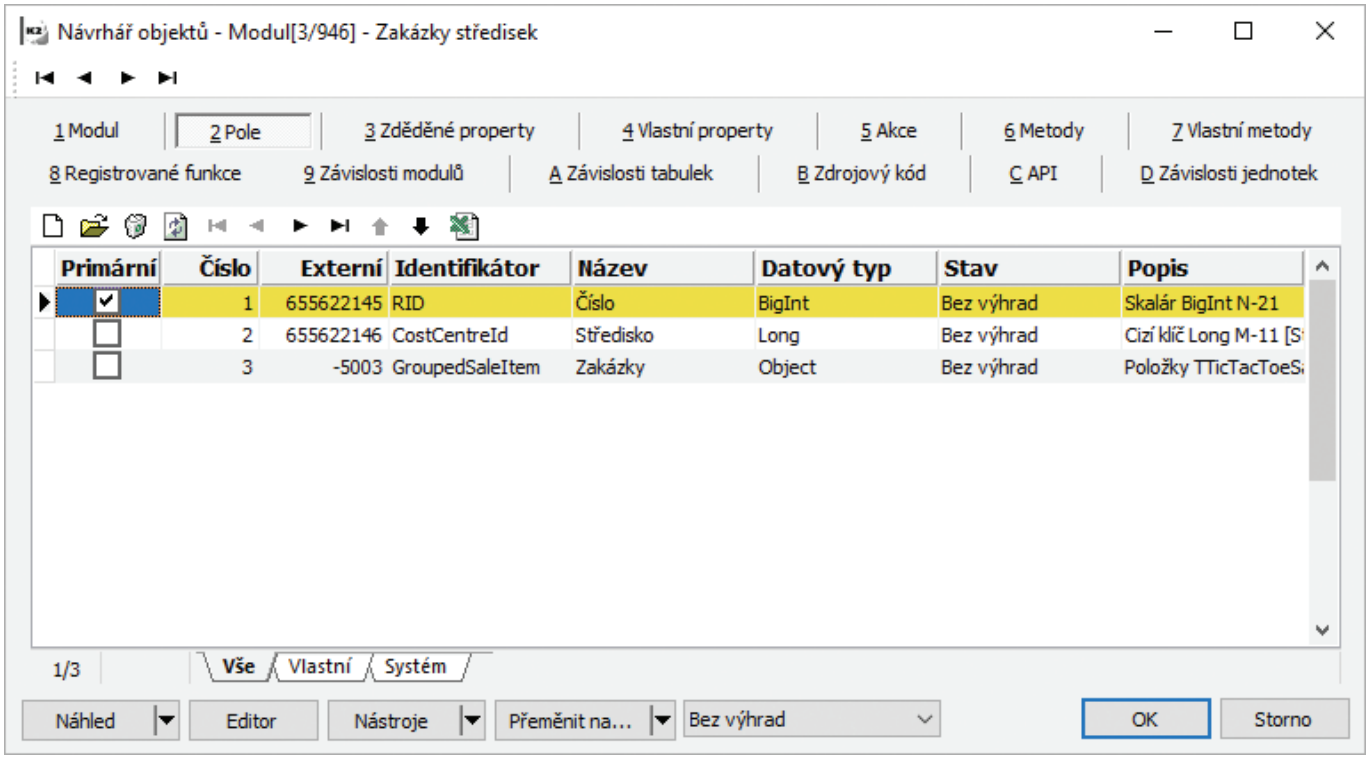
PŘÍKLAD: Jako příklad si vytvoříme datový modul, který bude zobrazovat seskupené zakázky dle způsobu dopravy a filtrované dle vybraného střediska. Pro každý způsob dopravy se budou součtovat ceny zakázky. Budeme tak mít přehled o celkovém součtu zakázek dle středisek rozepsaných na způsoby dopravy. Bude potřeba založit datový modul s předkem ***„TOneRecordDM“*** a přiřadit mu položky, které budou představovat rozpis seskupených zakázek. Tyto položky budou paměťové. Implementace paměťových položek je popsána v kapitole, která se věnuje metodám datových modulů. I zde se této části dotkneme v příkladu.

Vytvoříme datový modul, který nazveme ***“SaleByCostCentre“*** – zakázky středisek. Jako předka datového modulu nastavíme hodnotu ***„TOneRecordDM“***. Formulář se zredukuje, vše je vidět na Obrázek 118 - Příklad TOneRecordDM - Seskupení zakázek - datový modul.



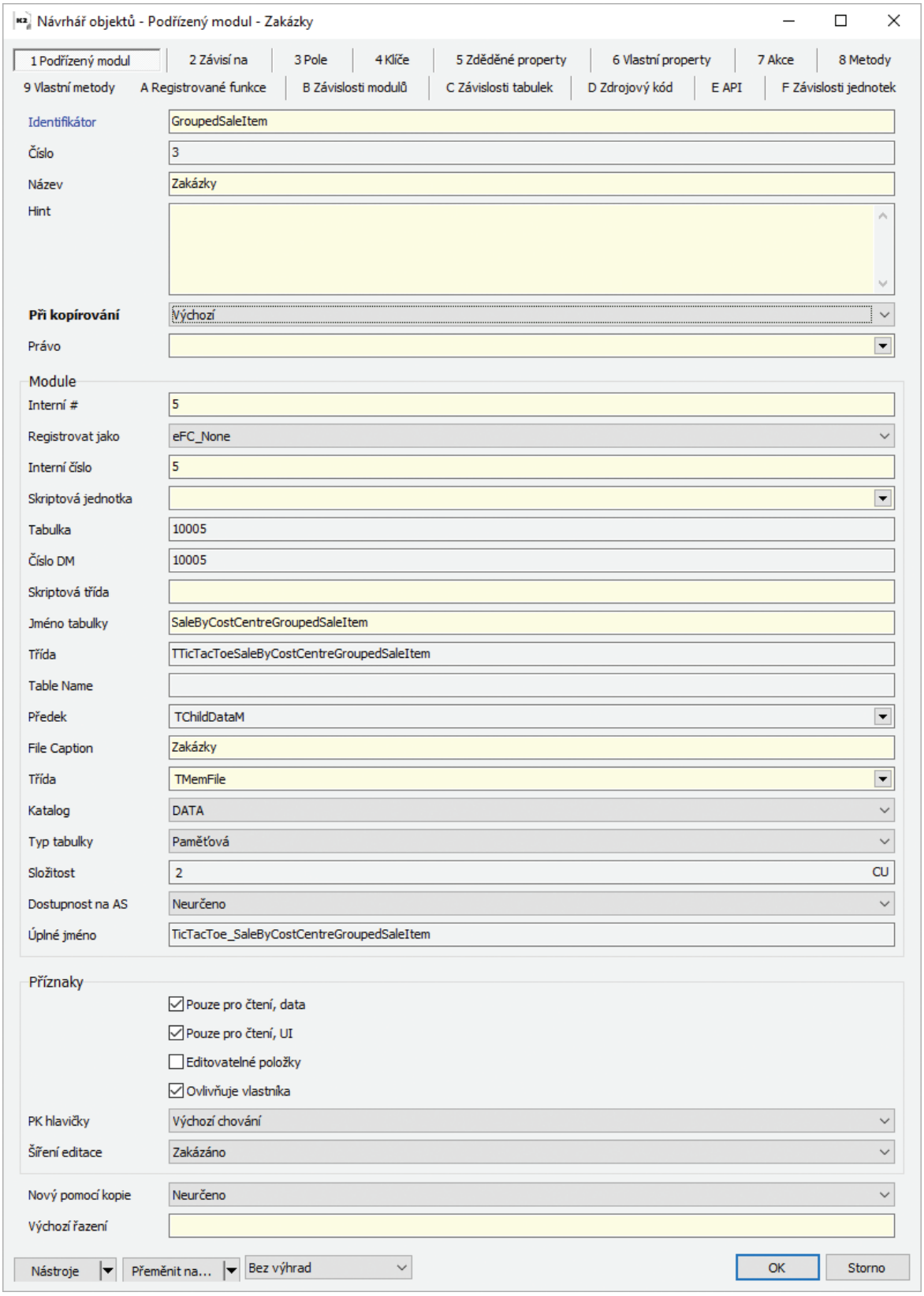
Obrázek 118 - Příklad TOneRecordDM - Seskupení zakázek - datový modul

V hlavičkovém záznamu nám stačí definovat pole, které bude určovat, pro které středisko budeme načítat zakázky. Vytvoříme si tedy pole ***„CostCentreId"*** typu cizí klíč do datového modulu středisek. I když v tomto případě nebudeme potřebovat primární klíč, datový modul je obecně postaven tak, že ho vyžaduje, založíme tedy pole ***„RID"*** typu primární klíč. Jako poslední vytvoříme pole, které bude představovat položky, viz Obrázek 119 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – položky.



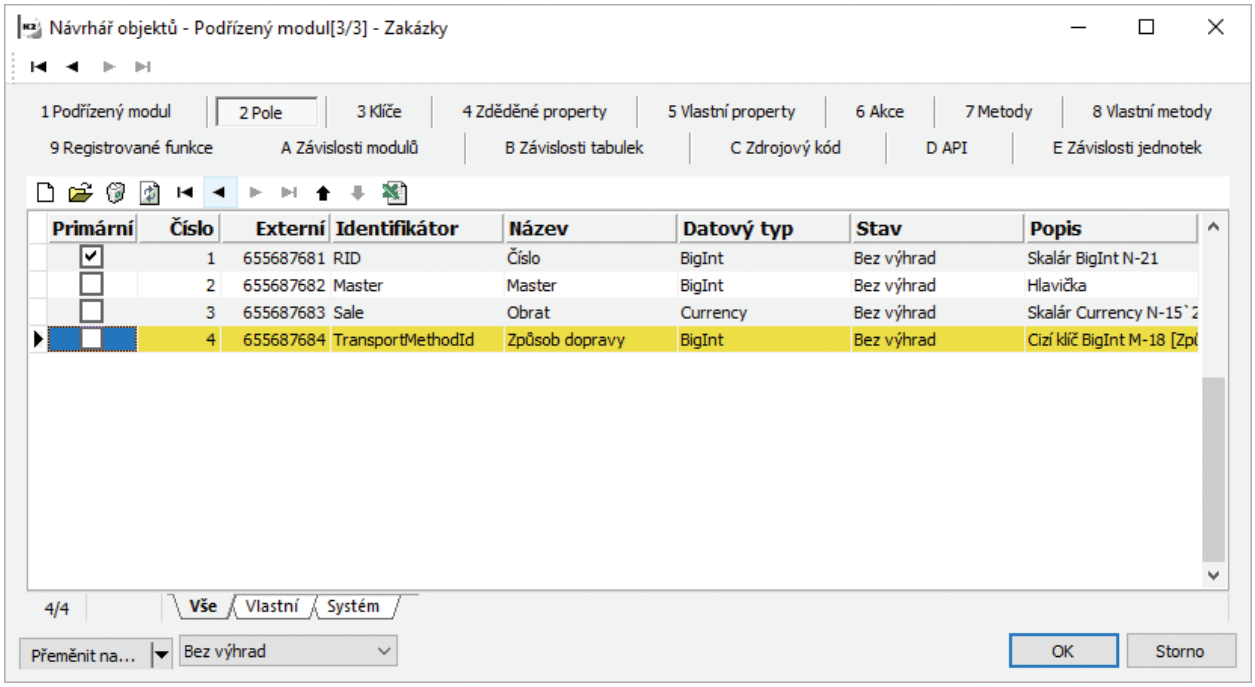
Obrázek 119 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – položky

Pole pro položky nazveme ***"GroupedSaleItem"***. Protože se bude jednat o položky, které budou napočítávány v paměti, nebude tedy existovat databázová tabulka, je potřeba nastavit třídu datového modulu na hodnotu ***„TMemFile"***, viz Obrázek 120 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky.



Obrázek 120 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky

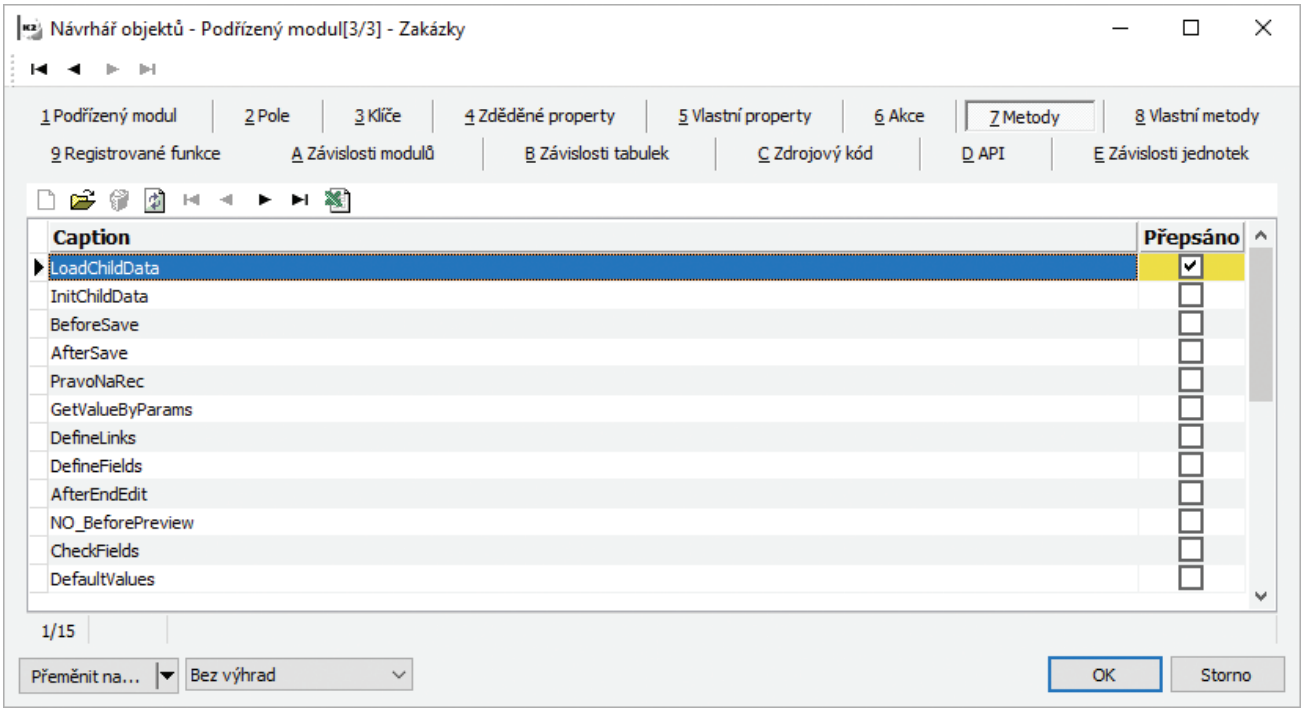
Jako první vytvoříme primární klíč – RID. V položce budou zobrazena seskupená data ze zakázek dle způsobu dopravy pro vybraná střediska. U každého záznamu bude cena. Vytvoříme tedy pole pro seskupený obrat - ***„Sale“*** – ***„Obrat“*** typu ***„Currency“***, dále pole pro způsob dopravy – ***„TransportMethodId“*** jako cizí klíč do datového modulu ***„TTransportMethodDM“***, viz Obrázek 121 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky - pole.



Obrázek 121 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky - pole

Strukturu položkového modulu máme připravenou. Přejdeme k implementaci plnění dat. Pro tyto případy obsahuje položkový modul metodu k implementaci ***„LoadChildData“***, která slouží k naplnění daty, viz Obrázek 122 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky – LoadChildData. Popisem této metody se detailněji budeme zabývat v kapitole o metodách. Zde se také setkáme s jejím použitím.

POZNÁMKA: Aby se metoda ***„LoadChildData“*** zavolala, je nutné, aby byly položky nastaveny pouze pro čtení pomocí příznaku ***„Pouze pro čtení, data“***, viz Obrázek 119 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – položky.



Obrázek 122 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – detail položky – LoadChildData

Implementace bude následující. Procedura ***„LoadChildData“*** má na vstupu jeden parametr ***„AChildFile“*** typu ***„TxFile“***. Tento objekt představuje samotné položky. Daný objekt je vždy při zavolání procedure ***„LoadChildData“*** automaticky vymazán, takže je prázdný (nemusíme jej už mazat my). Zavoláme SQL dotaz, který nám vrátí seskupené zakázky dle způsobů dopravy a filtrované dle střediska, které bude vybráno v hlavičkovém záznamu. Pro každou skupinu se provede součet ceny. Následně budeme výsledek SQL dotazu kopírovat postupně záznam po záznamu do ***„AChildFile“*** pomocí procedury ***„Add“***. Vše je detailně vidět na skriptu níže.

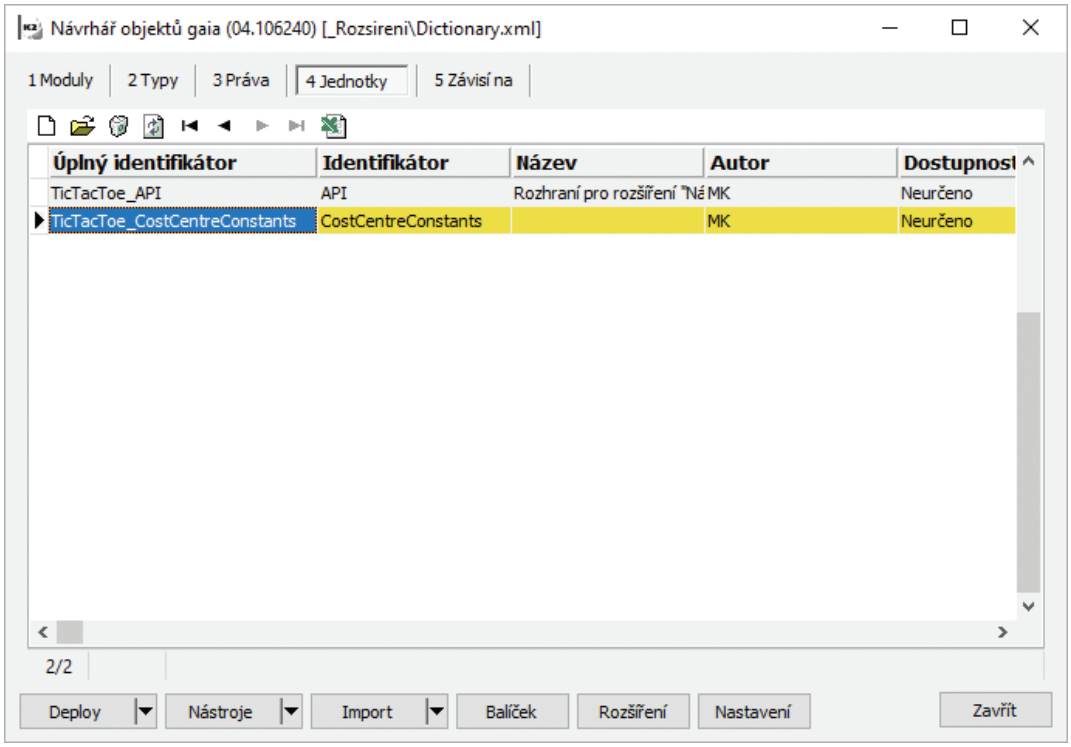
```
var
  lMFile : TMFile;
  lSql : string;
  lIndex : integer;

begin
  try
    lIndex := 1;
    lSql :=
+ ' SELECT @TransportMethodRID@, SUM(@AmountNet@) as Sale'
+ ' FROM %DBO.%@SALESORDER@ '
+ ' WHERE @CostCentreId@ = ? '
+ ' GROUP BY @TransportMethodRID@ ';

 RunSQLSelectDM(Self.MasterModule, lMFile, lSql, [Self.MasterModule.AsLong[SaleByCostCentre_
CostCentreId]], [ftLong], [0], ceDATA, True);
    lMFile.SetFirst;
    while lMFile.DoNext do
    begin
     AChildFile.SetAsInt64Pn('TransportMethodId', lMFile.AsInt64Pn('TransportMethodRID'));
      AChildFile.SetAsFloatPn('Sale', lMFile.AsFloatPn('Sale'));
      AChildFile.SetAsInt64Pn('RID', lIndex);
      Inc(lIndex);
      AChildFile.Add;
    end;
  finally
    lMFile.Free;
  end;
end;
```

Tímto máme připravené položky. Zbývá doplnit drobnosti do hlavičkového modulu, aby vše fungovalo dle požadavků.

V první řade musíme zajistit nastavení polí jednoho hlavičkového paměťového záznamu. Toto nastavení se provádí pomocí metody ***„DefaultValues“***, která slouží k definici výchozích hodnot nových záznamů datových modulů. Více o metodě v kapitole, která se metodami zabývá. Implementujeme tedy metodu následujícím způsobem. Nastaveníme pole ***„RID“*** na hodnotu ***„1“***. Vždy bude obsahovat pouze jeden záznam, není tedy potřeba řešit jiné hodnoty. Dále musíme nastavit pole pro středisko, dle kterého se budou filtrovat paměťové položky. Nastavíme tedy pole ***„CostCentreId“*** na hodnotu některého ze středisek dostupných v číselníku středisek.

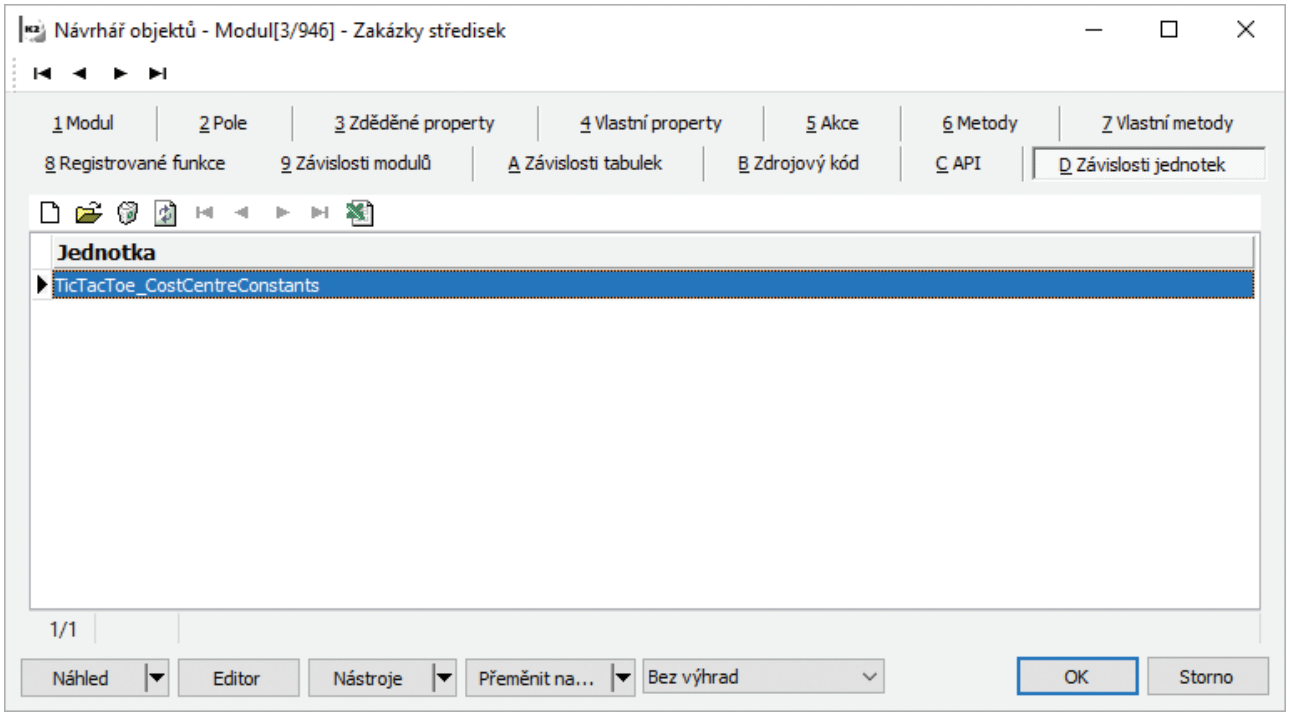
V příkladu se jako výchozí středisko nastavíme hodnotu ***„Ostrava“***. V rámci implementace je lepší identifikaci tohoto střediska definovat jako konstantu, kterou následně použijeme v našem kódu. Konstanty si můžeme definovat ve vlastní jednotce, která bude vytvořena v rámci aktuální definice rozšíření a bude tak nedílnou součástí celého projektu.

Založíme tedy novou jednotku s názvem ***„CostCentreConstants“*** na záložce ***„Jednotky“***, viz Obrázek 123 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – konstanty.



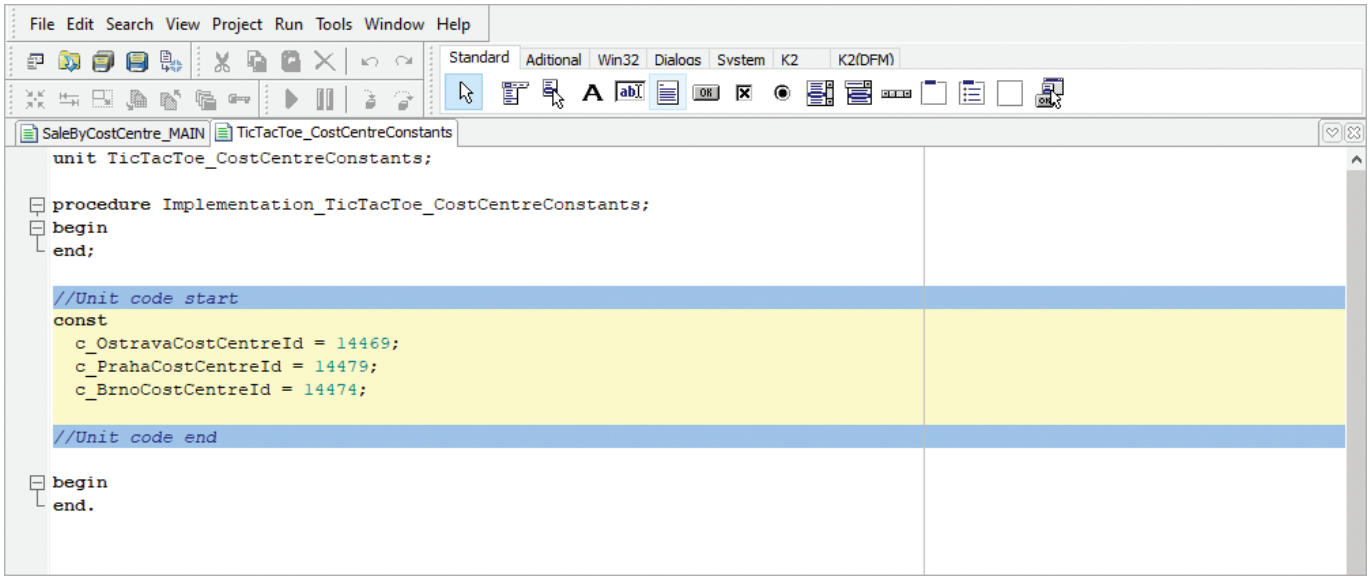
Obrázek 123 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – konstanty

Na stradu ***„D“*** (Závislosti jednotek) v modulu Zakázek středisek připojíme námi vytvořenou jednotku ***„TicTacToe_CostCentreConstants“***, kam budeme vkládat konstanty viz Obrázek 124 - Příklad TOneRecordDM - Seskupení zakázek – datový modul - vložení jednotky.



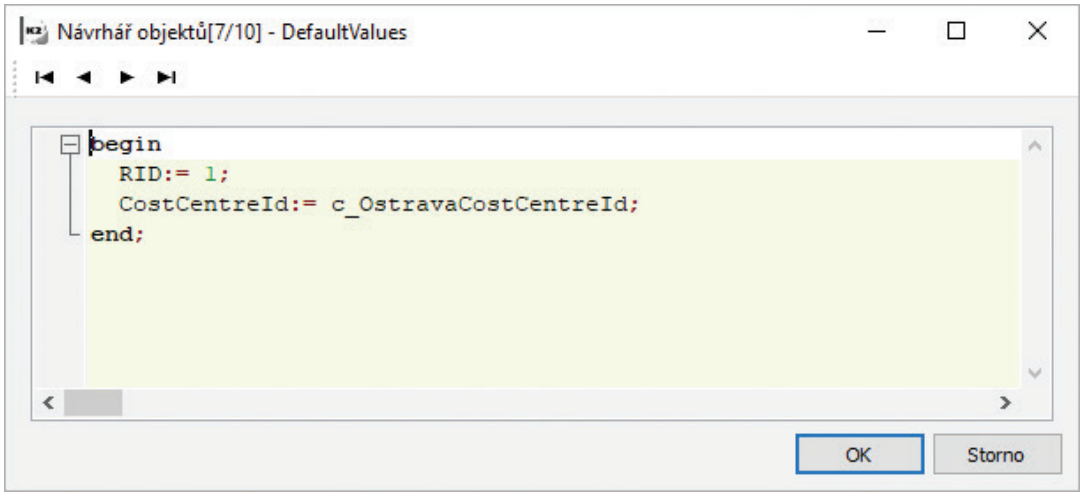
Obrázek 124 - Příklad TOneRecordDM - Seskupení zakázek – datový modul - vložení jednotky

Spustíme editor skriptu a definujeme konstanty pro střediska, která budeme chtít používat fixně v implementaci skriptu. Například se bude jednat o střediska pro Ostravu, Prahu a Brno, viz Obrázek 125 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – konstanty – implementace



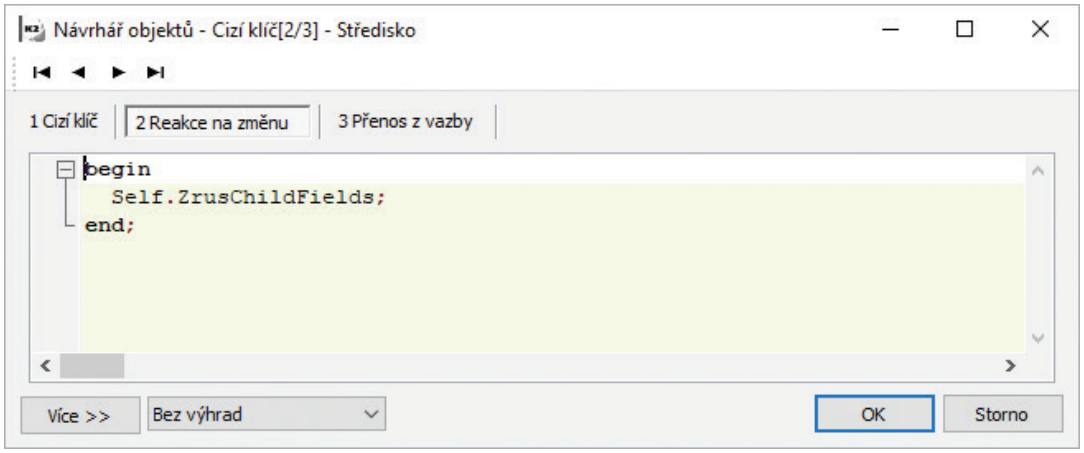
Obrázek 125 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – konstanty – implementace

Vrátíme se zpět do metody pro definici výchozích hodnot. Středisko nastavíme na naši konstantu – ***„OstravaCostCentreId“***.



Obrázek 126 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – DefaultValues

Po spuštění se vždy vytvoří jeden záznam, který bude mít nastaveno středisko na hodnotu ***„Ostrava“***. Položky se nám tedy vždy načtou odfiltrované pro toto středisko. Ještě nám zbývá implementovat chování, které bude zajišťovat, že při přepnutí střediska se přepočítají i samotné položky – tedy zavolá se metoda ***„LoadChildData“*** na položkovém modulu. Této funkčnosti docílíme pomocí volání procedury ***„ZrusChildFields“*** hlavičkového modulu, která zajistí opětovné načtení položek. Tuto proceduru musíme volat při změně střediska. Zatrhneme tedy na poli ***„CostCentreId“*** příznak ***„Reagovat na změnu“*** a na záložce ***„Reagovat na změnu“*** zajistíme volání zmíněné procedury, viz Obrázek 127 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – ZrusChildFields.



Obrázek 127 – Příklad TOneRecordDM – Seskupení zakázek – datový modul – ZrusChildFields

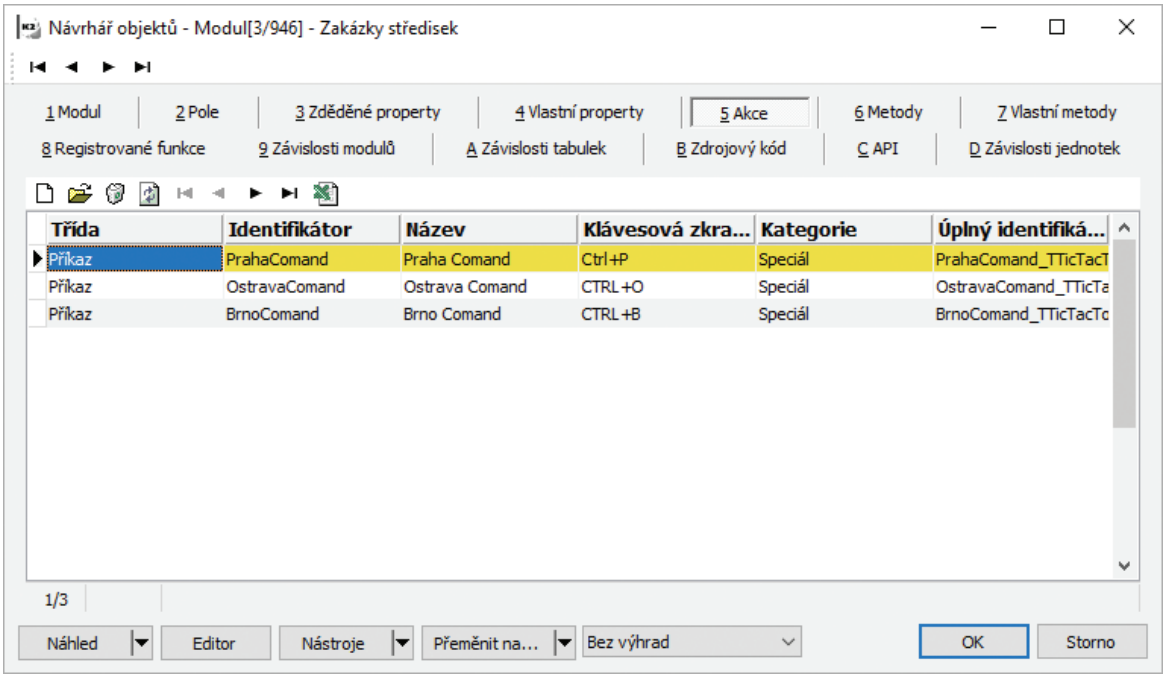
Na závěr si vybereme ještě tři nejdůležitější střediska, která bychom chtěli mít k přepnutí pomocí zkratkové klávesy a tlačítkem na formuláři. Zbylá střediska, ta méně důležitá, pak půjdou přepnout výběrem z rozevírací nabídky.

Na záložce ***„Akce"*** založíme nový ***„Příkaz"*** (více se o záložce akce dozvíte v kapitole 4.7. Akce), který nazveme ***„PrahaCommand"*** pro akci, která nám nastaví středisko na hodnotu ***„Praha"***. Ve výkonné části příkazu pak pouze nastavíme pole ***„CostCentreId"*** na hodnotu střediska Praha, pro které jsme si v předchozích krocích založili konstantu.

POZNÁMKA: Nesmíme zapomenout nastavit pole zápisem ***„NázevPole^"***, které nám zajistí nastavení formou ***"ControlValueChange"***, tedy zavolá se při tom další logika obsluhy pole v jádře programu, viz Obrázek 128 – Příklad TOneRecordDM – Seskupení zakázek – datový modul – Akce.

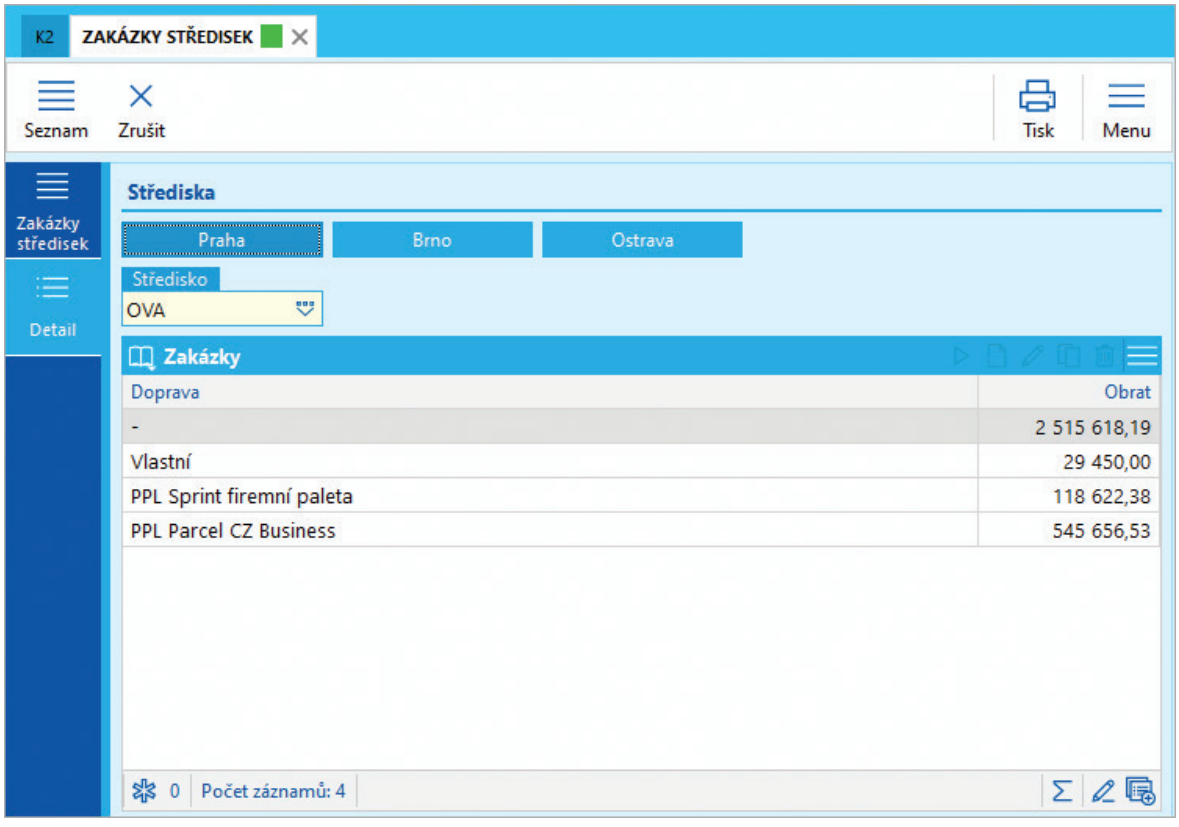
Obrázek 128 – Příklad TOneRecordDM – Seskupení zakázek – datový modul – Akce

Obdobně vytvoříme další akce – pro střediska ***„Ostrava"*** a ***„Brno"***, včetně klávesových zkratek, viz Obrázek 129 – Příklad TOneRecordDM – Seskupení zakázek – datový modul – Všechny akce.



Obrázek 129 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – Všechny akce

Na závěr provedeme operaci ***„Deploy“*** a můžeme vyzkoušet nový datový modul. Formulář může vypadat podobně jako tomu je na Obrázek 130 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – formulář. Máme definována tři tlačítka napojená na definované akce, dále rozbalovací nabídku do seznamu středisek. Pod těmito vstupy pak máme zobrazené napočtené položky z procedury ***„LoadChildData“***. V případě, že změníme středisko nebo stiskneme některé z tlačítek, přepočítají se položky a zobrazí v seznamu na formuláři.



Obrázek 130 - Příklad TOneRecordDM - Seskupení zakázek - datový modul – formulář

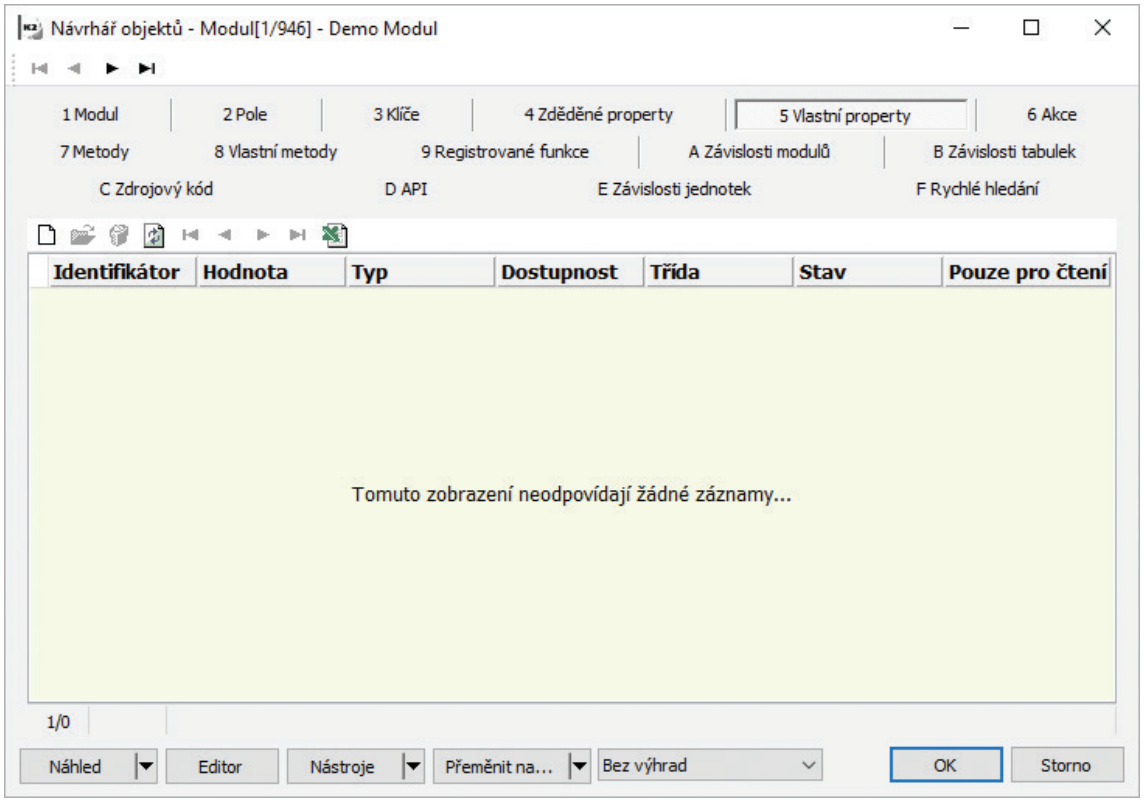
4.5.8. VLASTNOSTI PŘEDKA TMTBASEDM

Předka můžeme využít v případě, kdy potřebujeme modifikovat standardní řešení čteček. Princip implementace modifikací čteček je nad rámec kurzu, nebudeme se jím tedy zabývat.

4.6. VLASTNÍ PROPERTY

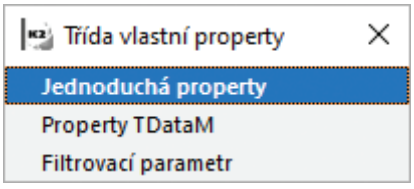
V předchozí kapitole jsme si popsali dostupné vlastnosti, které datový modul získá ze svého předka. Tyto vlastnosti je možné využít k úpravě chování datového modulu, v případě povinných, dokonce nutné použít. V aktuální kapitole si povíme o uživatelských vlastnostech. Tedy o vlastnostech, které je možné nadefinovat a následně používat v datovém modulu.

Nacházíme se na záložce ***„Vlastní property“*** viz Obrázek 131 - Záložka "Vlastní property" v datovém modulu. Novou vlastnost přidáme pomocí stisknutí tlačítka **Nový** v nástrojové liště nad seznamem vlastností, nebo pomocí klávesy **Insert**.



Obrázek 131 - Záložka "Vlastní property" v datovém modulu

Zobrazí se nám nabídka, kde určíme, jaký typ vlastnosti budeme vkládat. Máme tři možnosti. Buď ***„Vlastnost TDataM“***, ***„Jednoduchá property“*** nebo ***„Filtrovací parametr“*** viz Obrázek 132 - Výběr typu vlastní property v datovém modulu.

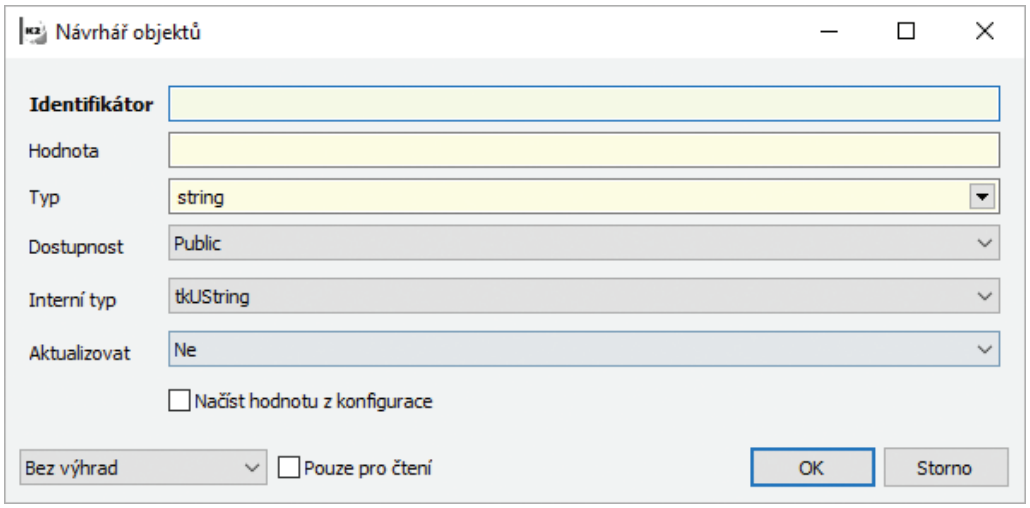


Obrázek 132 - Výběr typu vlastní property v datovém modulu

4.6.1. JEDNODUCHÁ PROPERTY (VLASTNOST)

Slouží k rychlé definici vlastnosti pro datový modul. Může nabývat různých datových typů, které K2 nabízí.

Při výběru ***„Jednoduché property“***, se otevře formulář pro zadání vlastnosti, viz Obrázek 133 - Vytvoření nové jednoduché vlastní property v datovém modulu. Popis vstupů na formuláři a jejich použití si vysvětlíme v následující části.



Obrázek 133 - Vytvoření nové jednoduché vlastní property v datovém modulu

Identifikátor
Slouží jako identifikátor a název vlastnosti. V datovém modulu musí být unikátní. Pokud budeme dále používat tuto vlastnost, např. v jiných metodách datového modulu, používáme právě tento identifikátor.

POZNÁMKA: Identifikátor musí být platný, tedy může obsahovat velká a malá písmena ***„a..z“***, číslice ***„0..9“*** a podtržítko). Identifikátor musí být unikátní v rámci ostatních identifikátorů použitých v datovém modulu, nesmí být v konfliktu s identifikátorem žádného pole, zděděné property, metody, akce apod. NO se snaží konflikty kontrolovat a hlásit, ale zcela bezpečně nelze konflikt odhalit. Vždy se může stát, že bude do předka ***„TDataM“*** přidána ***„published property“*** s konfliktním identifikátorem. Nejbezpečnější je používat nějaký prefix.

Hodnota
Slouží k zadání hodnoty, kterou představuje tato vlastnost. Jedná se o nepovinný údaj, který lze pro skalární hodnoty využít k nastavení výchozí hodnoty. Pro objekty použít nelze, ty jsou inicializovány vždy na ***„nil“***.

Jako vstup očekává NO konstantu správného typu. Výrazy jako například skriptový výraz ***„(1+2)“*** používat nelze. Výchozí hodnota odpovídá ***"nule"*** pro daný typ.

Typ
Určuje datový typ vlastnosti. Jedná se o stejný identifikátor typu, který byste použili při deklaraci proměnné / property ve skriptu. Pokud použijete identifikátor, který NO nezná, objeví se v poli ***"Interní typ"*** hodnota ***„NEURČENO“***, což nutně neznamená, že takový typ nelze použít (viz poznámka). Naopak pro známý typ se objeví jiná hodnota. Například pro ***„Integer“*** a ***„Byte“*** se objeví ***„Integer“***, pro ***„TDate“*** je to ***„Float“***, pro ***„string“*** je to ***„UString“***. Pro platný objektový typ, například ***„TDataM“***, se objeví ***„Class“***. Na Obrázek 134 - Definice vlastní property typu Integer v datovém modulu s „Pouze pro čtení“, je vidět definice vlastnosti, která je datového typu ***„Integer“***. Pokud bude v poli ***„Hodnota“*** jiný typ, než jsme zvolili, NO jej nedovolí uložit a vyskočí příslušné hlášení.

POZNÁMKA: U datových typů, kde interní typ je Float (např. Currency, TDate, TTime ...) lze zapsat hodnotu s oddělovačem desetinných míst jak pomocí čárky, tak pomocí tečky. U speciálního typu TDate lze zapsat hodnotu i ve formátu dd:mm:rrrr (01.01.2021) nebo zkráceně dd:mm:rr (01.01.21). U typu TTime lze zapsat hodnotu ve formátu hh:mm:ss (12:15:32) nebo zkráceně hh:mm (12:15). U typu TDateTime lze zapsat hodnotu ve formátu dd:mm:rrrr hh:mm:ss (01.01.2021 12:15:32) nebo zkráceně dd:mm:rr hh:mm(01.01.21 12:15).

POZNÁMKA: Pro ***„TDataM“*** se nedoporučuje používat jednoduchá property, ale speciální typ property ***„TDataM“***. Více k danému typu je popsáno níže.

POZNÁMKA: Návrhář objektů nezná všechny typy, které lze ve skriptu použít. Pokud se v poli *"Interní typ"* objeví hodnota ***„NERUČENO“***, nemusí to být na překážku. Znamená to pouze to, že použitý identifikátor nezná návrhář objektů. Důležité je, aby byl tento typ známý ve skriptu. V případě, že typ skutečně použít nelze, dojde při překladu datového modulu k chybě (např. ***„Exception EAssertionFailed“***).

Dostupnost
Definuje viditelnost vlastnosti v rámci instancí datových modulů. Typy, které jsou zde k dispozici, vycházejí z definice zapouzdření v objektově orientovaného programování. Tedy vycházejí ze standardu OOP. K dispozici jsou následující možnosti.

Public – vlastnost je viditelná všude

Private – vlastnost je viditelná pouze v rámci třídy

Protected – vlastnost je viditelná pouze v rámci třídy a v jejích předcích

Published – vlastnost je viditelná všude, navíc je viditelná ve formulářích K2

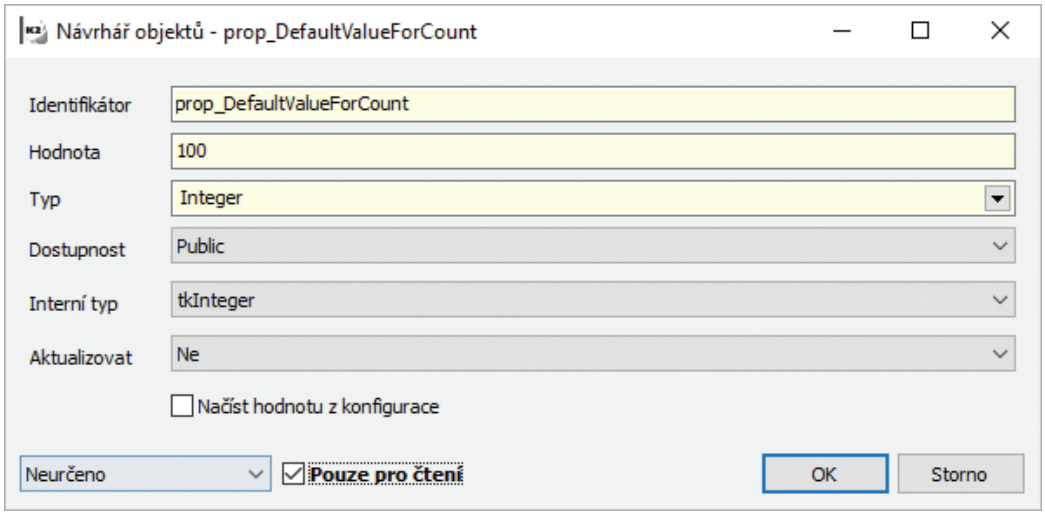
Interní typ
Slouží jen jako doplňková informace, kterou nelze změnit. ***„Interní typ“*** se nastaví automaticky dle zvoleného datového typu vlastnosti. V případě, že je datový typ špatně vyplněn, je nastaveno na ***„tkUnknown“***.

Aktualizovat
Slouží k automatickému načtení dat po aktualizaci hodnoty property. Vlastnost můžeme využít například, když na základě hodnot property filtrujeme seznam záznamů. Po změně hodnoty se data znovu načtou a filtr se aplikuje. Výchozí hodnota je ***„Ne“***.

Načíst hodnotu z konfigurace
Slouží pro načtení hodnoty ze souboru. Pokud je daná property zatržena, i přesto lze vyplnit pole ***„hodnota“***, která se plní jako výchozí hodnota. Tato hodnota může být následně přepsána hodnotou z INI souboru.

Pokud chceme načíst hodnotu z konfigurace, tak se knihovna datového modulu při vytváření instance pokusí najít soubor se jménem {ContextName}.ini nejprve v adresáři {FirmPath} a pokud jej nenalezne, poté zkusí adresář {ConfPath}. Pokud dojde k tomu, že existuje soubor v obou adresářích, použije se pouze {FirmPath}. Hodnota {ContectName} je pro většinu datových modulů shodná s {ClassName}. V .ini souboru bude jako první řádek obsahovat ***„[Property]“*** a do dalších řádků lze již zadat {název property} = {hodnota}, jednotlivé property zapisujeme pod sebe.

Pouze pro čtení
Vlastnost nelze měnit. Využívá se tedy hodnoty, která je zadána v poli ***„Hodnota“***. Obrázek 134 - Definice vlastní property typu Integer v datovém modulu s „Pouze pro čtení“.



Obrázek 134 - Definice vlastní property typu Integer v datovém modulu s „Pouze pro čtení“

Automaticky uvolnit

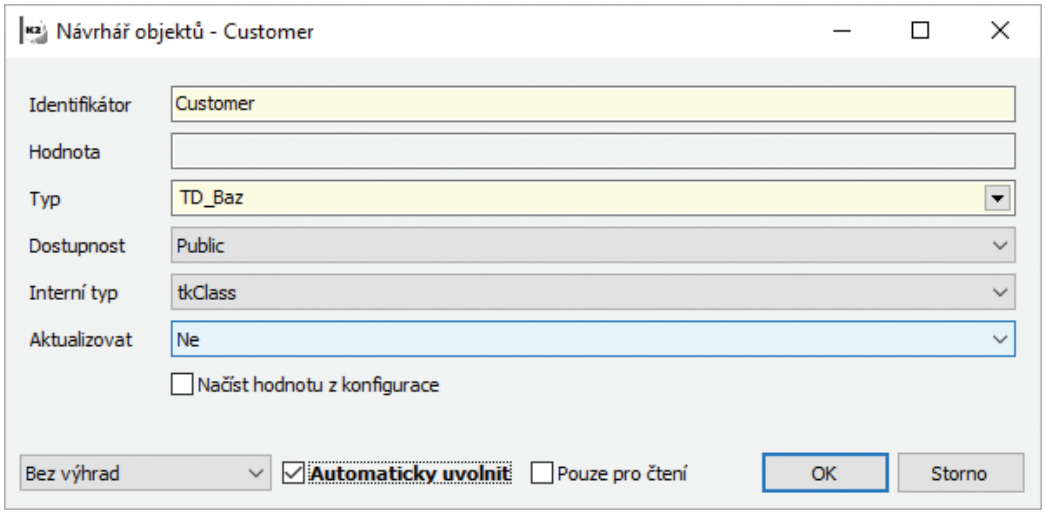
V případě, že nastavíme typ vlastnosti na objekt, například ***„TDataM“*** (s interním typem ***„Class“***) je možné navíc nastavit příznak ***"Automaticky uvolnit"***, viz Obrázek 135 - Definice vlastní property typu objekt v datovém modulu.

Pomocí tohoto nastavení můžeme ovlivnit uvolňování instance přiřazené do nadefinované vlastnosti. Pokud nastavíme na ***„Automaticky uvolnit“*** je instance uvolněna při uvolnění instance datového modulu.

Jak bylo napsáno již výše, v jednoduché property se nedoporučuje používat ***„TDataM“***.

POZNÁMKA: Automatické uvolňování je možno nastavit pouze v případě, že víme, že nikdo jiný tuto instanci nepoužívá a nikdo jiný ji neuvolňuje. **Je zakázáno uvolňovat originální datové moduly**, které vrací metody ***„CreateODM“***.

V případě, že pracujeme s duplikátem datového modulu, nastavujeme volbu ***„Automaticky uvolnit“*** vždy, jinak by docházelo k neuvolňování paměti, což může mít za následek pád aplikace.

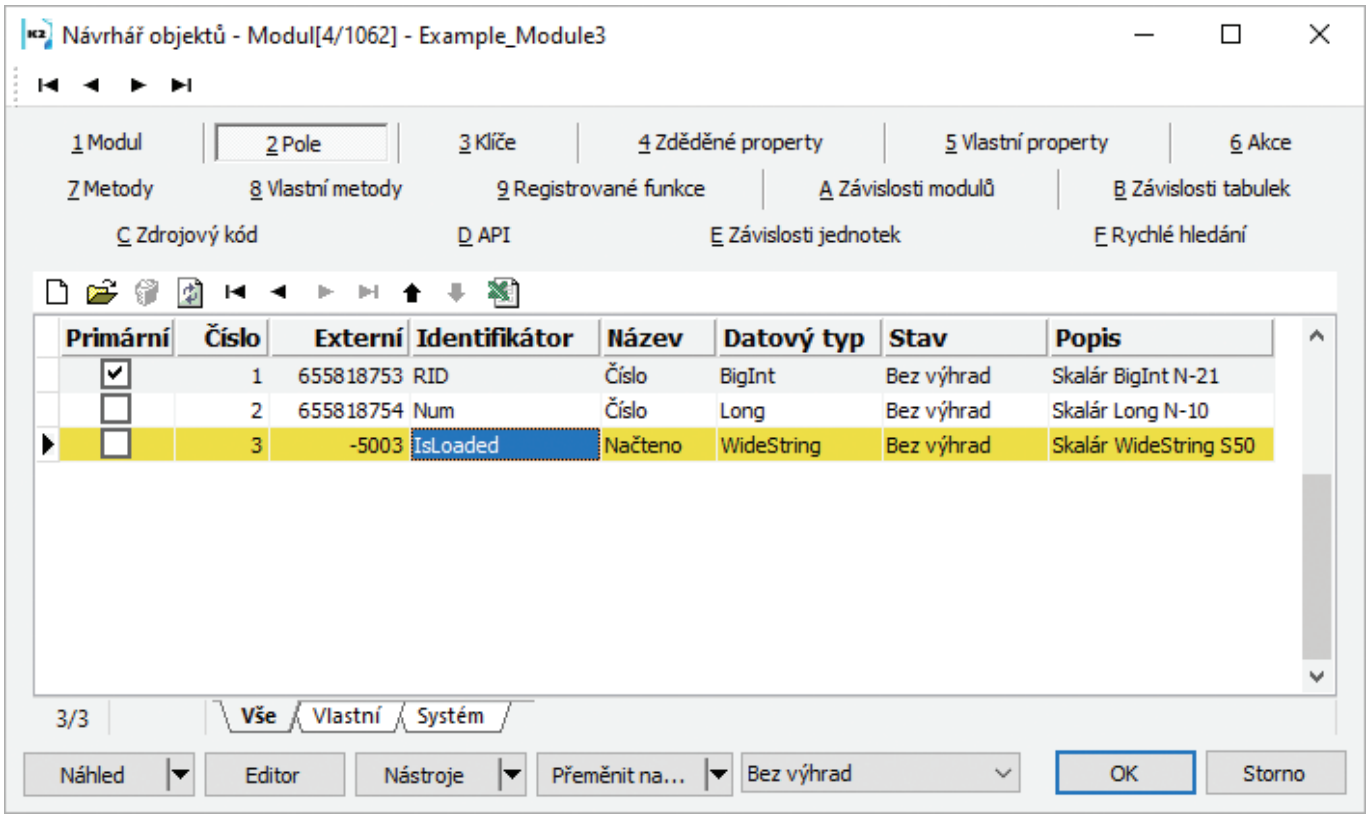


Obrázek 135 - Definice vlastní property typu objekt v datovém modulu

POZNÁMKA: Typickým použitím jednoduché property může být třeba typ ***„boolean“***, pomocí které si můžeme řídit ***„drahé“*** čtecí operace. Například ve funkci provádím složitý výpočet dat, který je časově a výkonově náročný a v našem případě postačí, když se spustí jen jednou. Pomocí jednoduché vlastnosti typu ***„boolean“*** si můžu řídit, zda mám data již načtena či nikoliv. Jednoduše řečeno, načítání podmíníme nastavením této vlastnosti. Při prvním výpočtu nastavím vlastnost na ***„true“*** a při dalším pokusu již znovu neproběhne.

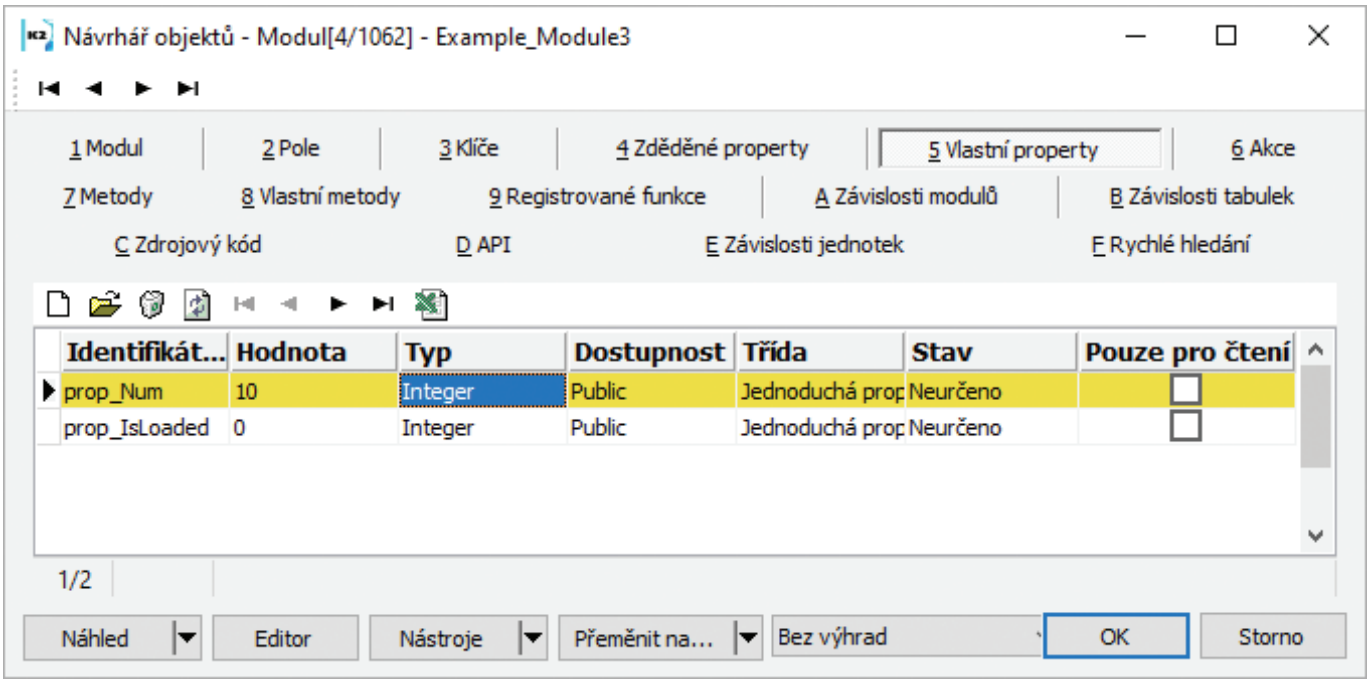
PŘÍKLAD: Vytvoříme si datový modul, kde budeme ověřovat, kolikrát se hodnota v poli Nu, změnila. Využijeme k tomu dvě jednoduché property. Jedna nám bude sloužit jako hledaná hodnota, kterou načteme z parametru a druhá property bude počítadlo.

Datový modul si pojmenujeme ExampleModul3 – Ukázkový modul 3, kterému nastavíme předka ***„TBaseDataM“***. Vytvoříme si samozřejmě pole ***„RID“***, dále pole ***„Num“*** (integer) což bude pole, kde budeme měnit číslo. Dále si vytvoříme počítané pole IsLoaded (widestring) jako popis. Toto pole bude vracet počet, kolikrát se změnila hodnota v poli ***„Num“*** na námi hledanou hodnotu, včetně nějakého textu. Přehled nadefinovaných polí můžeme vidět na obrázku Obrázek 136 - Definice polí pro příklad.



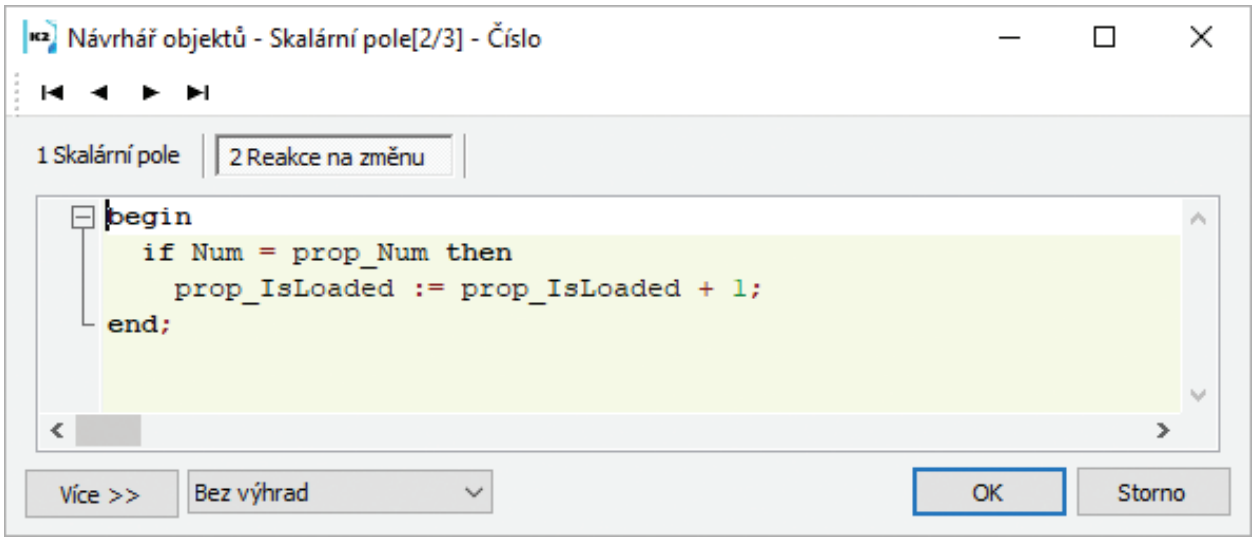
Obrázek 136 - Definice polí pro příklad

Vytvoříme si vlastní property, nejprve pro identifikaci hledaného čísla ***„prop_Num“***, které bude integer a výchozí hodnotu může mít 10. Nezapomeneme si zatrhnout volbu ***„Načíst hodnotu z konfigurace“***. Další property bude pro počet načtení čísla ***„prop_IsLoaded“*** a výchozí hodnota bude také 0. Definici těchto property můžeme vidět na obrázku Obrázek 137 - Definice property pro příklad.



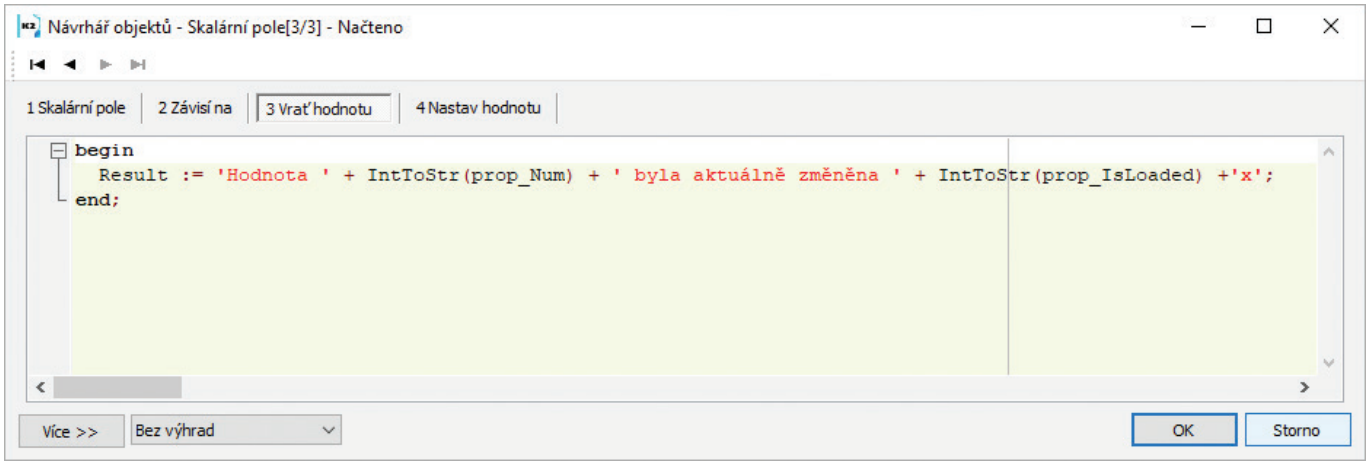
Obrázek 137 - Definice property pro příklad

Vrátíme se k naším polím, kde si v poli ***„Num“*** zatrhneme příznak ***„Reagovat na změnu“*** a do záložky ***„Reakce na změnu“*** zapíšeme kód, který bude počítat změnu pole na naší hodnotu načtenou z konfigurace (Obrázek 138 - Kód pro reakci na změnu pro pole "Num").



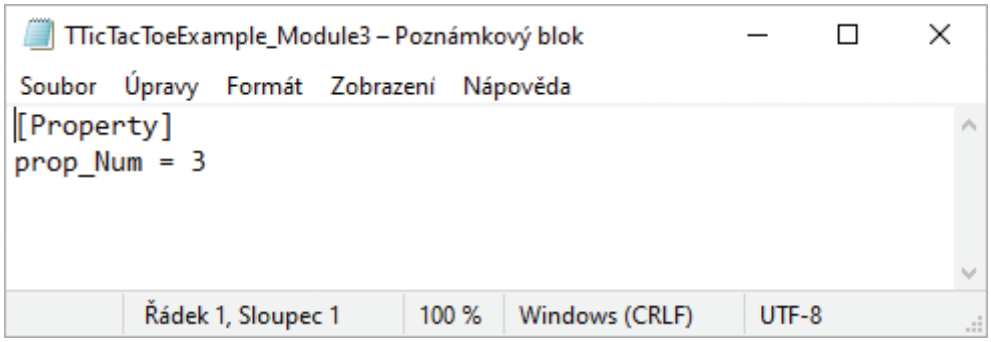
Obrázek 138 - Kód pro reakci na změnu pro pole "Num"

Do počítaného pole vložíme kód, který nám bude vracet text o tom jaká je počítaná hodnota a kolikrát byla načtená (Obrázek 139 - Výpočet počítaného pole "IsLoaded").



Obrázek 139 - Výpočet počítaného pole "IsLoaded"

Nezapomeneme nastavit i zděděné property. Provedeme akci Deploy. Nakonfigurujeme si konfigurační soubor pro property. Vlezeme tedy do firemního adresáře, kde založíme soubor ***„TTicTacToeExample_Module3.ini“*** Do tohoto souboru jako první napíšeme ***„[property]“*** a na druhý řádek nastavení dané property. Celou definici konfiguračního souboru lze vidět na obrázku Obrázek 140 - Nastavení konfiguračního souboru pro vlastní property. Následně si můžeme v K2 daný modul otevřít a vyzkoušet.

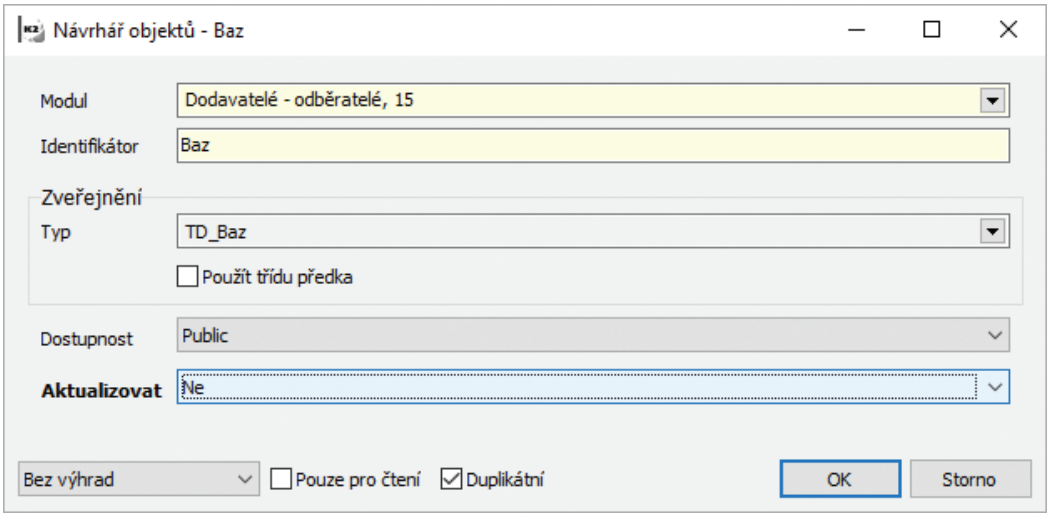


Obrázek 140 - Nastavení konfiguračního souboru pro vlastní property

4.6.2. PROPERTY TDATAM

Dalším typem vlastnosti, kterou můžeme na datovém modulu nadefinovat je ***„TDataM“***. Jedná se o speciální případ vlastní property, která je typu registrovaný datový modul. Na rozdíl od jednoduché property, tato obsahuje podporu pro automatické vytvoření instance a také následné uvolnění po uvolnění datového modulu. Vlastnosti tohoto typu se většinou používají v implementaci akcí a počítaných polí.

Při vkládání této vlastnosti se nám zobrazí formulář, který je vidět na Obrázek 141 - Definice vlastní property typu TDataM. K nastavení máme pouze tři parametry.



Obrázek 141 - Definice vlastní property typu TDataM

Modul
V tomto parametru vybíráme typ TDataM, který bude obsažen ve vlastnosti. Po kliknutí na nabídku se nám otevírá seznam všech zaregistrovaných datových modulů, ze kterých si můžeme vybrat požadovaný.

Identifikátor
Zde platí stejná logika jako u jednoduché property. Po nastavení modulu se nám hodnota identifikátoru automaticky nastaví. Je doporučeno změnit dle vlastního požadavku.

Typ
Automaticky dosazená hodnota, o jaký typ se jedná.

Použít třídu předka
Vlastnost, pomocí které můžeme nastavit, že místo třídy modulu, chceme použít tu z předka daného modulu.

Dostupnost
Definuje viditelnost vlastnosti v rámci instancí datových modulů. Typy, které jsou zde k dispozici, vycházejí z definice zapouzdření v objektově orientovaného programování. Tedy vycházejí ze standardu OOP. Dané možnosti byly popsány již výše u ***„Jednoduché property“***.

Aktualizovat
Slouží k automatickému načtení dat po aktualizaci hodnoty property. Vlastnost můžeme využít například, když na základě hodnot property filtrujeme seznam záznamů. Po změně hodnoty se data znovu načtou a filtr se aplikuje. Výchozí hodnota je ***„Ne“***.

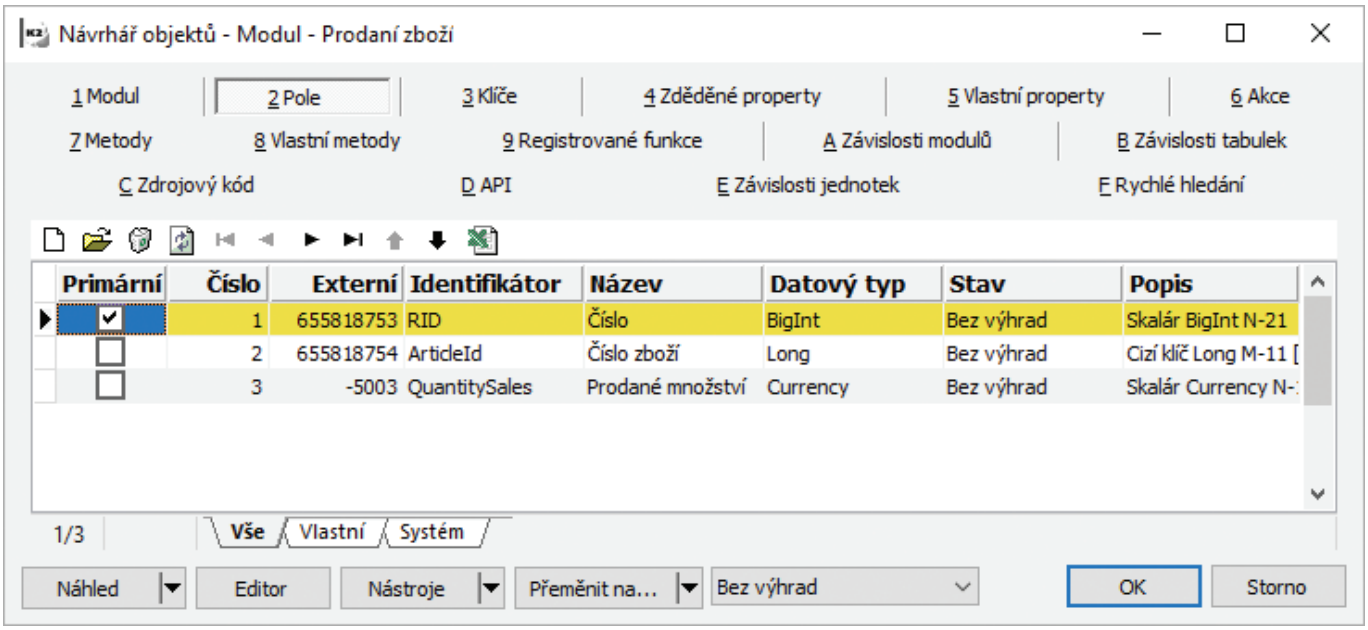
Pouze pro čtení
Pokud je daná vlastnost zatržena, property je pouze pro čtení. V opačném případě můžeme místo původní instance dát jinou, ale stejného typu.

Duplikátní
Pokud potřebujeme pracovat s originálem datového modulu, tuto vlastnost odtrhneme. Jinak ponecháme nastaveno, tak jak je ve výchozím stavu, tedy ***„Duplikátní“***. Doporučuje se pracovat s duplikátem datového modulu.

POZNÁMKA: Tento typ vlastnosti zajišťuje automaticky vytvoření instance a také její uvolnění. Samozřejmě zohledňuje nastavení ***„Duplikátní“***.

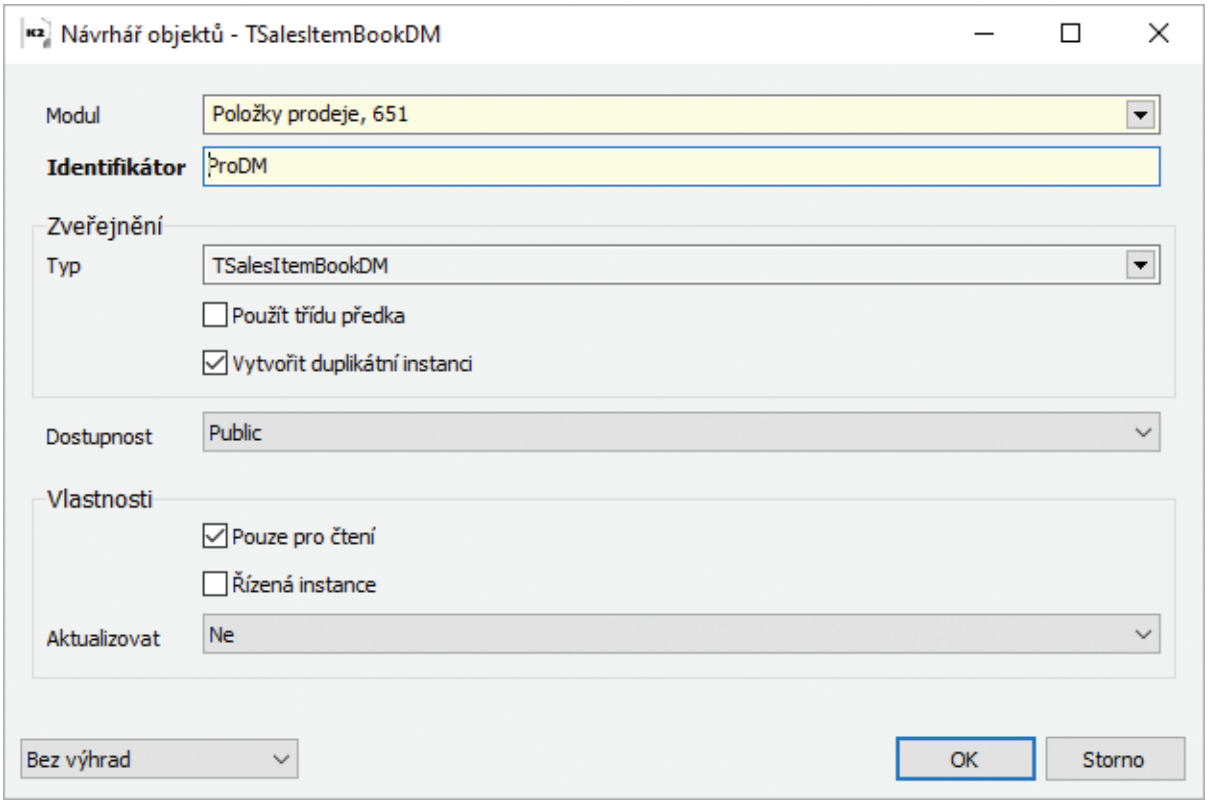
POZNÁMKA: U tohoto typu vlastnosti není potřeba definovat závislosti v datovém modulu, tedy není nutné přidávat do sekce ***„modules“*** typ datového modulu z vlastnosti. Návrhář objektů automaticky generuje tuto sekci dle typů definovaných ve vlastnostech ***„TDataM“***.

PŘÍKLAD: Použití property typu ***„TDataM“*** si demonstrujeme na následujícím příkladu. Máme datový modul, ve kterém budeme chtít zobrazit pro vybrané zboží, definované jako cizí klíč, součet prodaného množství. K sumě množství zboží využijeme property typu ***„TSalesItemBookDM“***. Vytvoříme si tedy datový modul např. prodané zboží ***„SalesArticle“***. Pro jednoduchost budeme mít jen klíčové pole ***„RID“***, cizí klíč do datového modulu ***„Zboží“*** a počítané pole ***„QuantitySales“***, které bude sloužit k zobrazení prodaného množství. Seznam polí můžeme vidět na Obrázek 142 - Seznam polí datového modulu v příkladu pro vlastní property TDataM.



Obrázek 142 - Seznam polí datového modulu v příkladu pro vlastní property TDataM

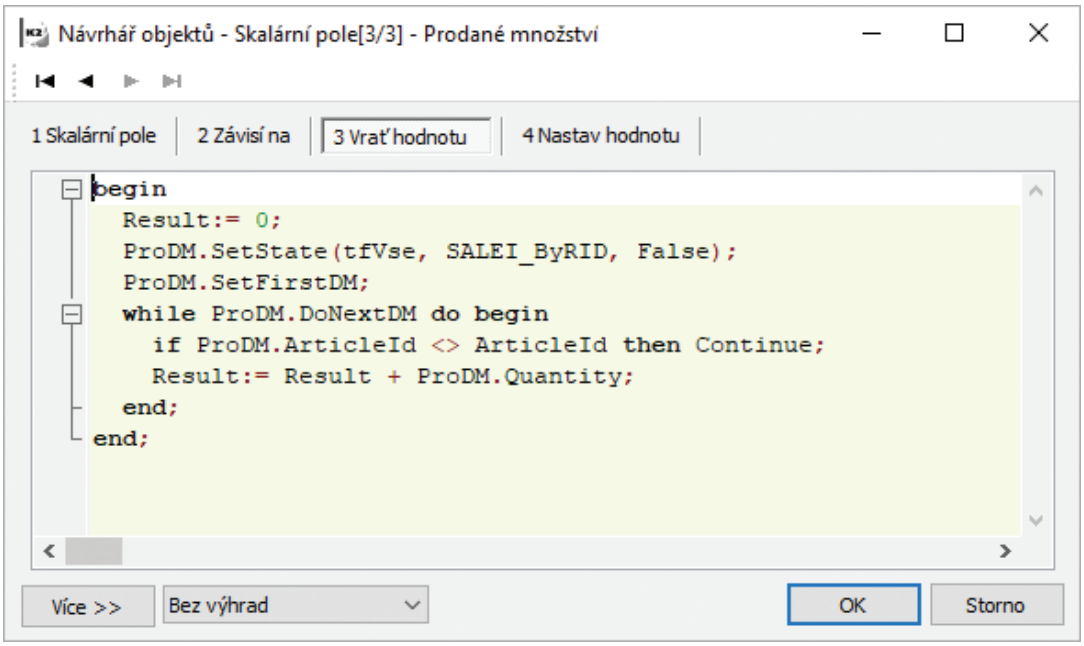
Dalším krokem bude definice samotné property, která nám bude sloužit k vytvoření součtu prodaného množství průchodem přes položky prodeje daného zboží. Na straně ***„Vlastní property“*** vložíme novou property typu ***„TDataM“***, kterou si nazveme ***„ProDM“***, do hodnoty ***„Modul“*** vybereme datový modul ***„Položky prodeje“*** a necháme nastavenu hodnotu ***„Duplikátní“*** a ***„Pouze pro čtení“***, viz Obrázek 143 - Příklad vytvoření vlastní property typu "TDataM".



Obrázek 143 - Příklad vytvoření vlastní property typu "TDataM"

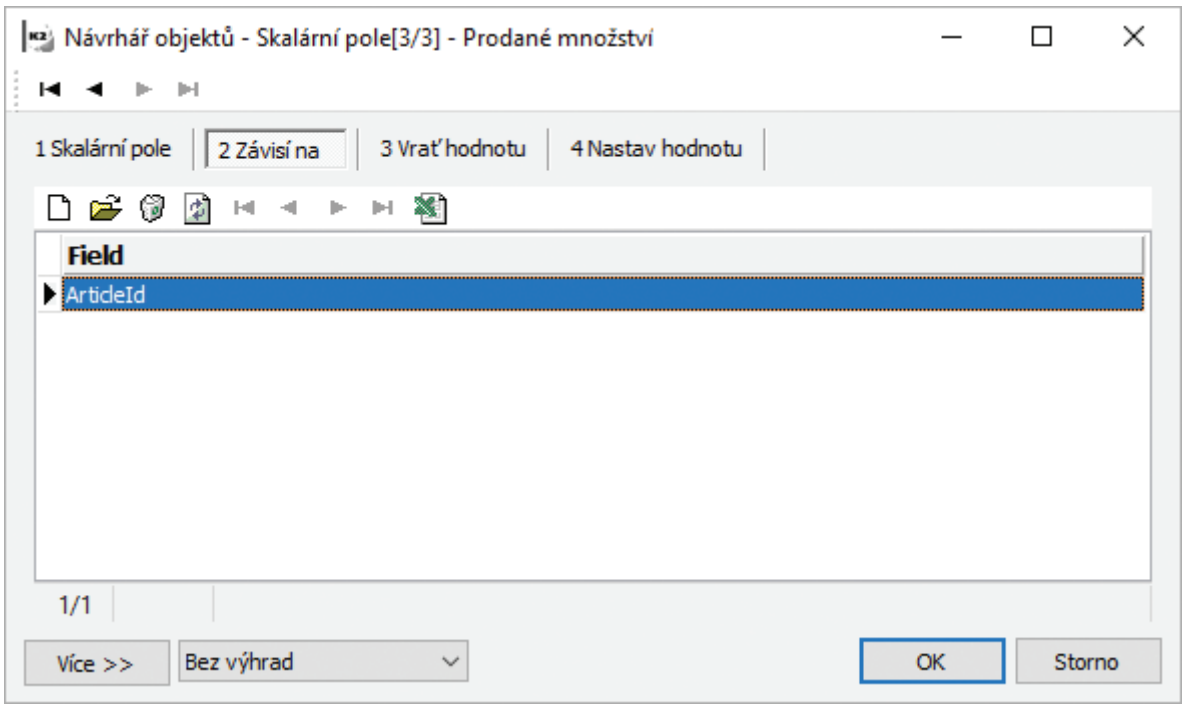
Vrátíme se zpět do definice polí a otevřeme detail počítaného pole ***„QuantitySales“***. Na záložce ***„Vrať hodnotu“*** vložíme skriptový kód (nebo pomocí editoru NO), který provede následující:

Nastavíme datový modul položek prodeje na první záznam a postupně projdeme všechny záznamy s tím, že jakmile položka prodeje obsahuje v poli ***„ArticleId“***, což je hodnota zboží, hodnotu shodnou s cizím klíčem ***„ArticleId“*** v našem datovém modulu, tak přičteme celkové prodané množství do výstupu dané funkce. Implementace je vidět na Obrázek 144 - Implementace počítaného pole s použitím vlastní property TDataM.



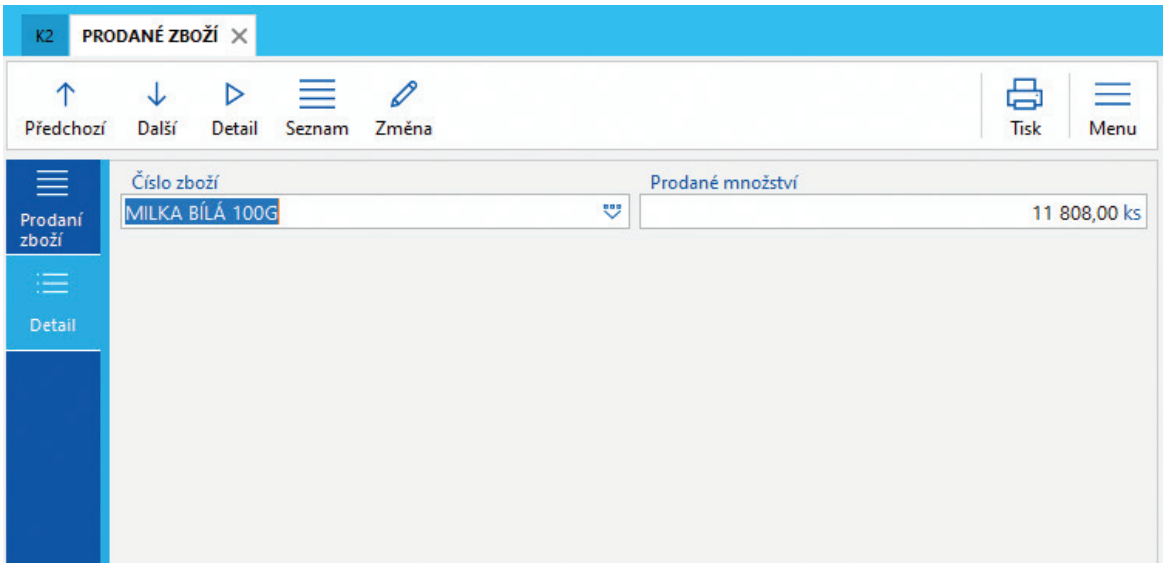
Obrázek 144 - Implementace počítaného pole s použitím vlastní property TDataM

V případě, že bychom chtěli sumu prodaného množství zboží aktualizovat vždy po tom, co se změní hodnota pole ***„ArticleId“***, můžeme na počítaném poli v sekci ***„Závisí na“***, toto pole nastavit. Jinak se přepočet provede při otevření, případně uložení záznamu v datovém modulu. Definici závislosti můžeme vidět na Obrázek 145 - Nastavení závislostí počítaného pole v příkladu na vlastní property TDataM.



Obrázek 145 - Nastavení závislostí počítaného pole v příkladu na vlastní property TDataM

Příklad máme hotový. Po provedení akce ***„Deploy“*** můžeme datový modul vyzkoušet. Samozřejmě si zkontrolujeme i doporučené vlastnosti na záložce ***„Zděděné property“***. Zobrazíme si ho v univerzálních formulářích a vložíme záznam. Například založíme záznam se zákazníkem ***„MILKA BÍLÁ 100G“***, v poli ***„Prodané množství“*** se nám spočítá prodané množství ze všech prodaných položek vybraného zboží v základní jednotce zboží, viz Obrázek 146 - Záznam v datovém modulu v příkladu na property typu TDataM.



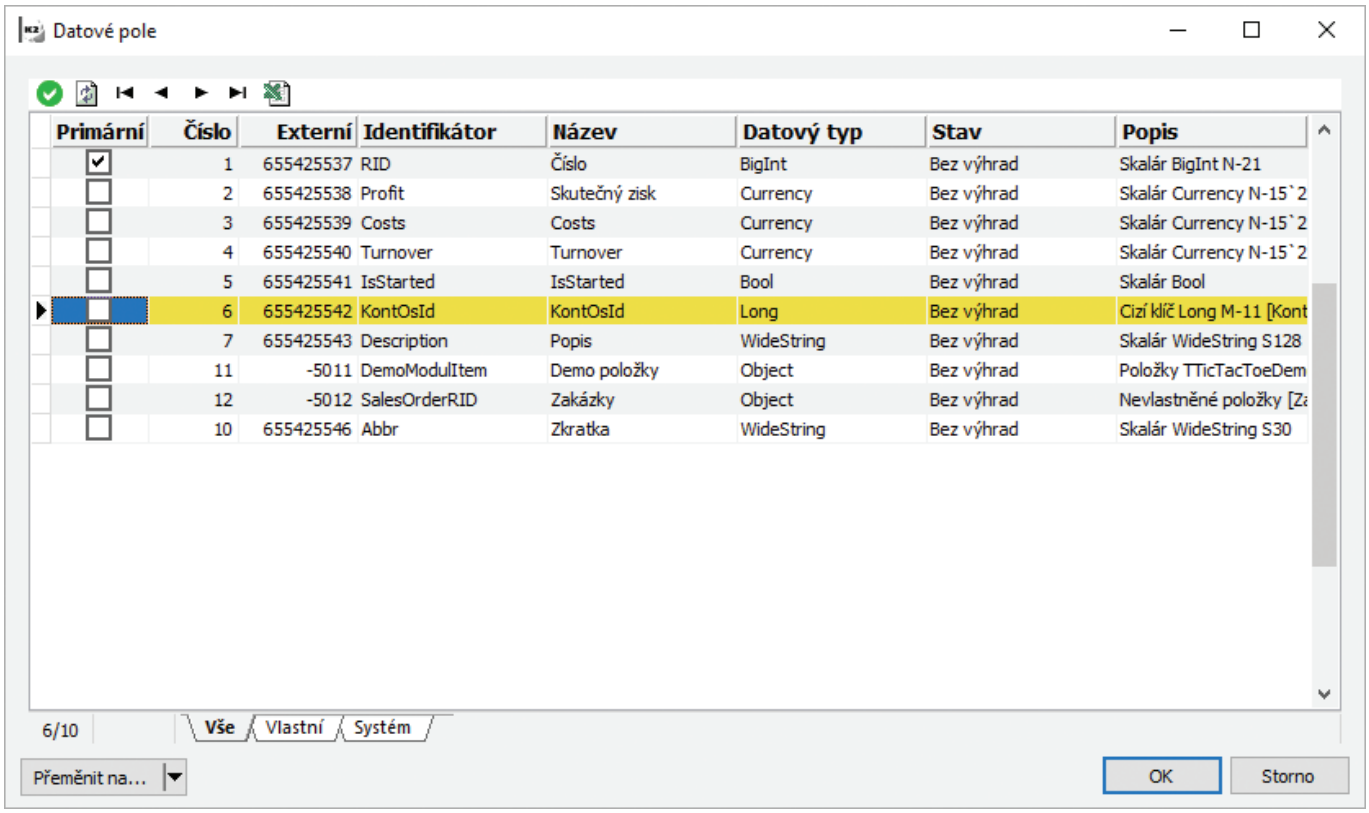
Obrázek 146 - Záznam v datovém modulu v příkladu na property typu TDataM

POZNÁMKA: V případě definice property *„TDataM“* není nutné přidávat do závislostí typ datového modulu, který máme definovaný v property. Návrhář objektů nám tuto operaci zajistí sám tím, že automaticky generuje sekci *„modules“* ve zdrojovém kódu, což bychom mohli vidět na záložce *„Zdrojový kód“* datového modulu. Pokud ale potřebujeme konstanty, je potřeba na stranu *„Závislosti tabulek“* přidat příslušnou tabulku.

4.6.3. FILTROVACÍ PARAMETR

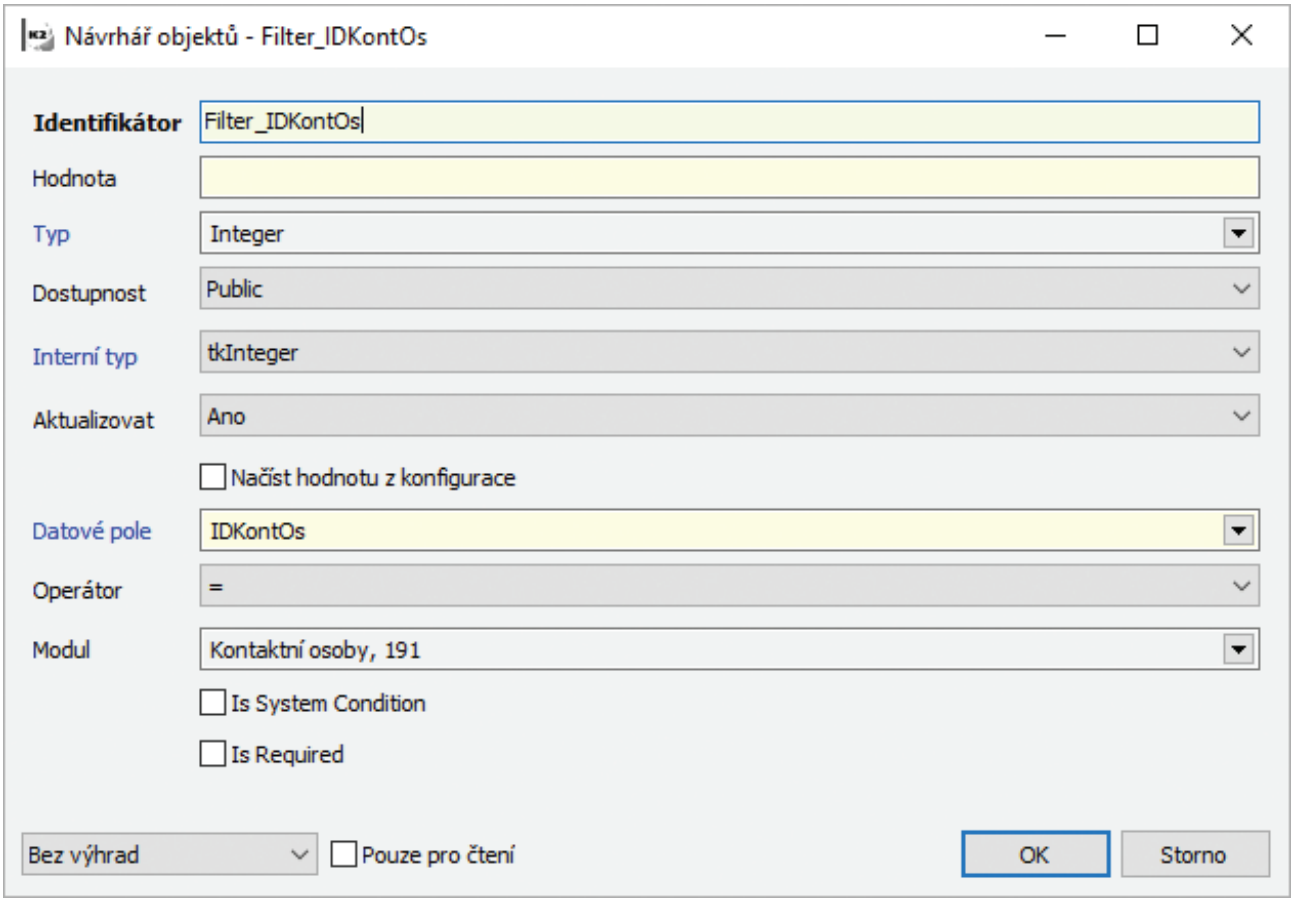
Tento typ vlastnosti slouží k definici parametru, který bude sloužit jako implicitní podmínka k omezování záznamů v datovém modulu ve stavu *„Kniha“*. Tzn. datový modul, bude ve stavu kniha filtrován dle podmínek, které jsou zde definovány.

Po výběru vlastnosti typu *„Filtrovací parametr“*, se zobrazí formulář Obrázek 147 - Výběr pole pro filtrovací parametr vlastnosti datového modulu, ve kterém se nám nabídnou pole z našeho datového modulu (demonstrováno na *„Demo modulu“*). Následně vybereme pole, které budeme parametrem nastavovat. Například pole *„KontOsId“*, což je vazba do datového modulu *„Kontaktní osoby“*. Tímto nadefinujeme podmínku, na základě, které můžeme omezovat záznamy v našem datovém modulu ve stavu *„Kniha“* dle hodnoty kontaktní osoby, která je v každém záznamu nastavena.



Obrázek 147 - Výběr pole pro filtrovací parametr vlastnosti datového modulu

Po výběru pole pro parametr, se otevře formulář pro samotné nastavení filtrovacího parametru. Většina údajů je vyplněna automaticky dle vybraného pole z předchozího formuláře. Na Obrázek 148 - Definice filtrovacího parametru jako vlastní property v datovém modulu jsou vidět další vlastnosti filtrovacího parametru. Některé nelze měnit, návrhář objektů je nastavuje automaticky. Ty zde nebudeme popisovat, slouží pouze pro informaci a jejich význam je jednoduše odvoditelný.



Obrázek 148 - Definice filtrovacího parametru jako vlastní property v datovém modulu

V následující části si popíšeme jednotlivé vlastnosti filtrovacího parametru.

Identifikátor
Jednoznačný název v rámci datového modulu. Tedy v datovém modulu vznikne vlastnost, která bude mít tento název. Pokud budeme z vlastnosti v instanci datového modulu číst nebo do ní zapisovat, pak k ní přistupujeme právě tímto identifikátorem.

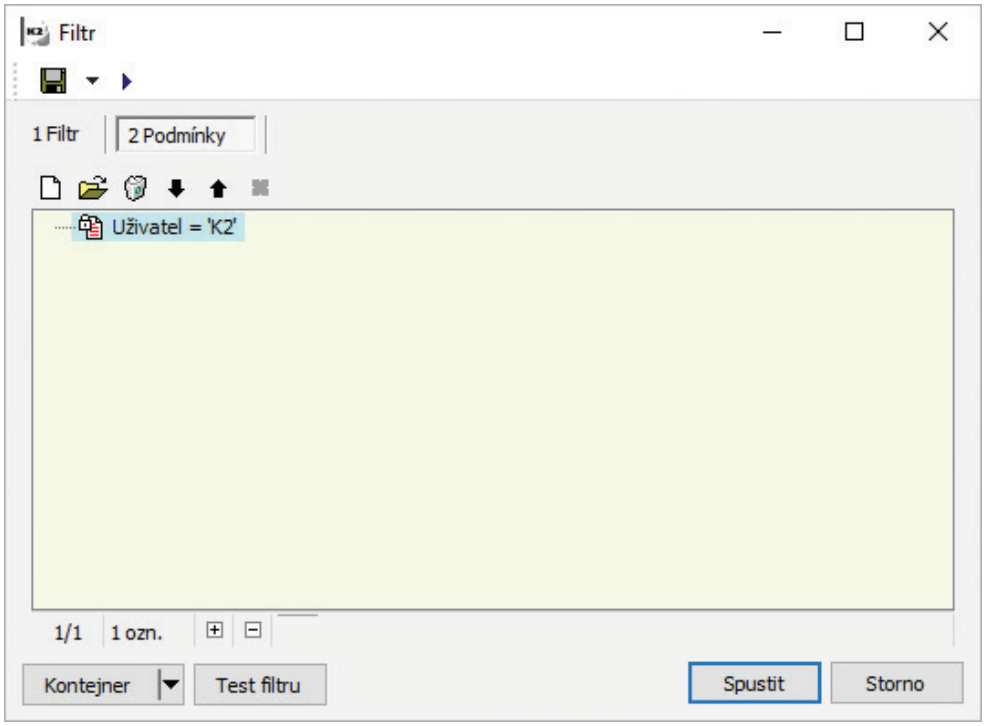
Hodnota
Slouží k definici výchozí hodnoty, která bude nastavena do parametru.

Načíst hodnotu z konfigurace
Slouží pro načtení hodnoty ze souboru. Jak daná vlastnost funguje bylo popsáno výše, kde jsme se věnovali ***„Jednoduché property“***.

Datové pole
Obsahuje hodnotu pole, které jsme vybrali při zakládání vlastnosti. Viz Obrázek 147 - Výběr pole pro filtrovací parametr vlastnosti datového modulu.

„Is System Condition“ – Systémová podmínka
Definuje, že se jedná o podmínku, kterou nelze změnit. V případě, že vytvoříme v datovém modulu filtr, překlopí se do jeho definice i tato systémová podmínka.

Například omezíme záznamy pouze na ty, které vytvořil uživatel K2, viz Obrázek 149 - Systémová podmínka Filtrovací parametr.



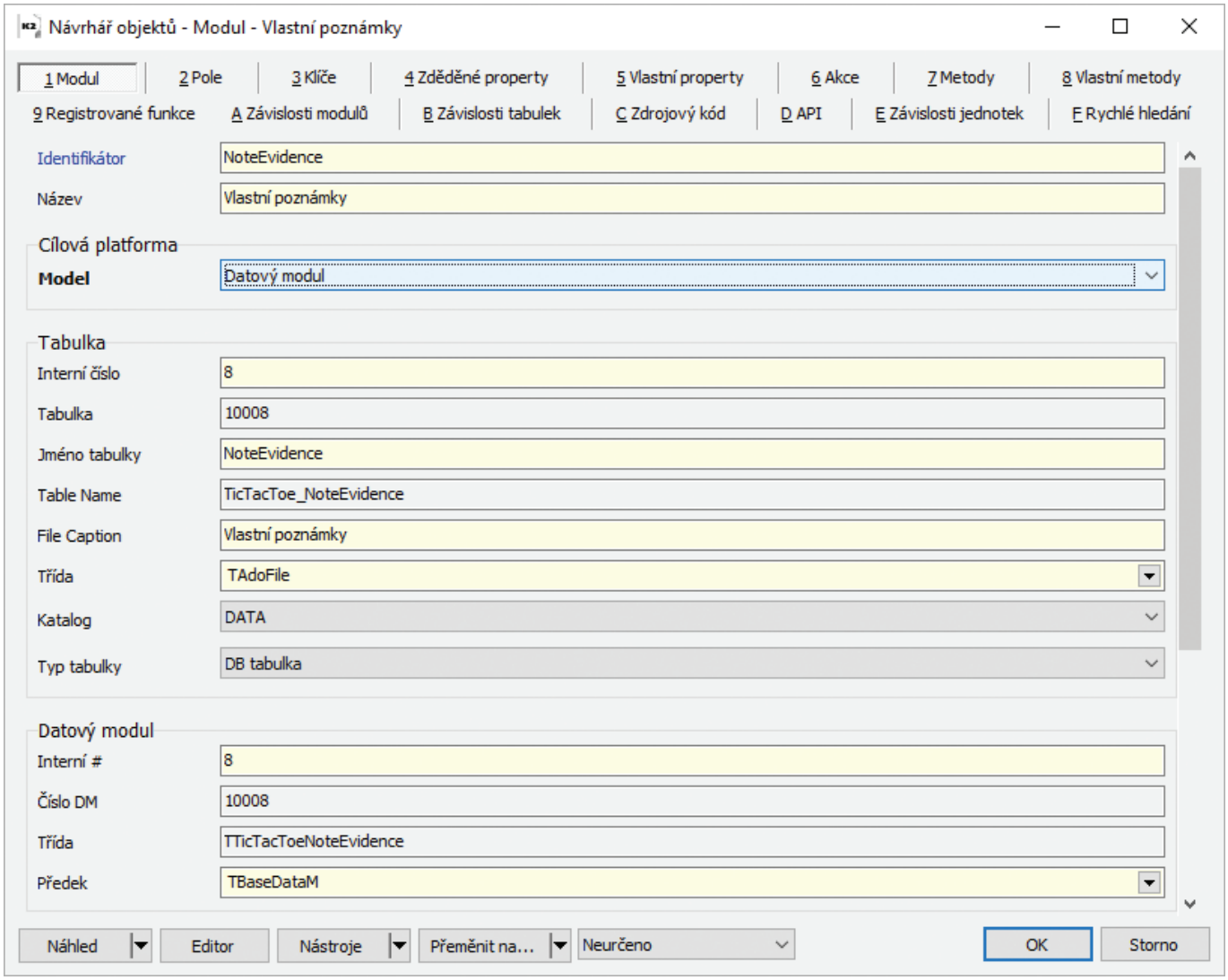
Obrázek 149 - Systémová podmínka Filtrovací parametr

„Is Required" – Povinný
Určuje, zda je parametr povinný či ne. Dle nastavení tohoto příznaku pak můžeme aktivovat či deaktivovat filtrovací parametr pomocí formuláře na nastavení filtrovacích podmínek.

POZNÁMKA: Definice podmínky je omezena na výchozí hodnotu, která je určena v poli ***„Hodnota"***, pole z datového modulu a operátor ***„="***. Dynamické hodnoty pak musíme měnit na vhodném místě, například ve skriptu, který spouští datový modul nebo například v akci (commandu). Vše si popíšeme na následujícím příkladu.

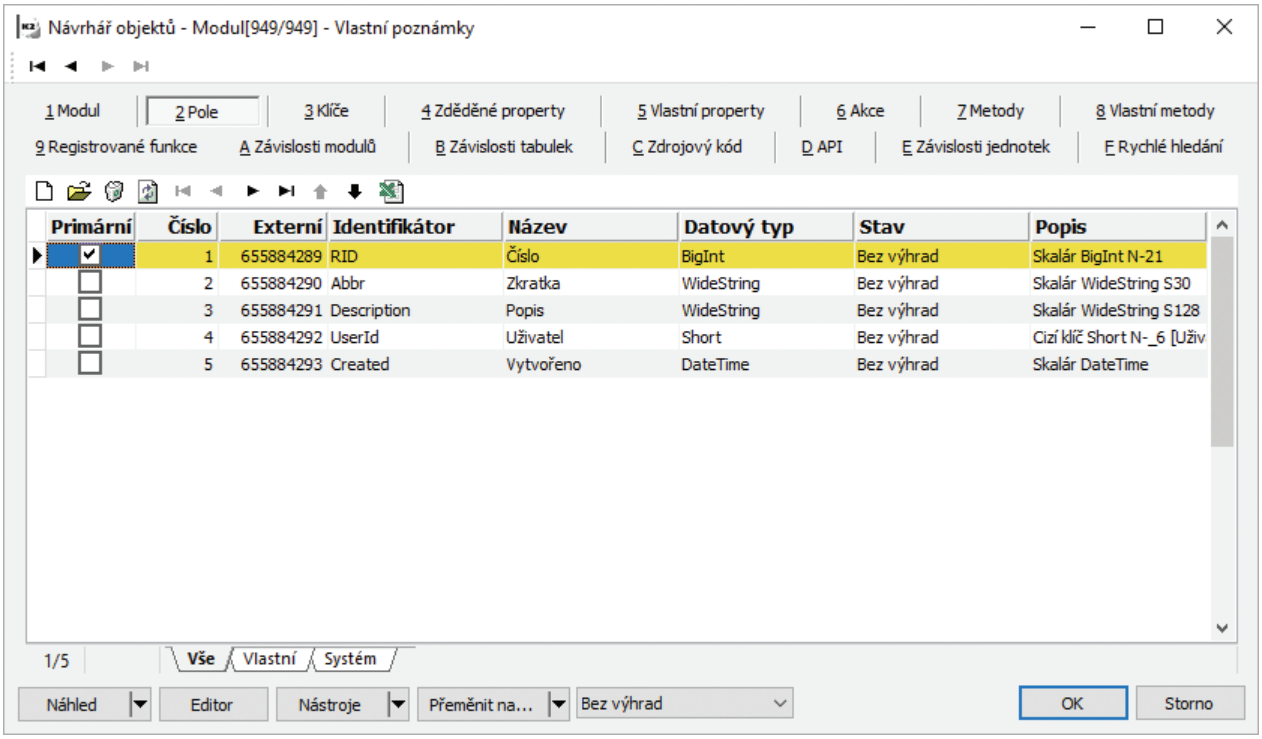
PŘÍKLAD: Vytvoříme jednoduchý datový modul, který bude sloužit k evidenci poznámek. Předpokladem je, že každý uživatel bude mít přístup pouze ke svým poznámkám. Uvidíme, že v závislosti na nastavení si budeme moci zpřístupnit i poznámky jiných osob.

Založíme nový datový modul ***„NoteEvidence"*** – ***„Vlastní poznámky"***, viz Obrázek 150 - Příklad Vlastní poznámky - datový modul.



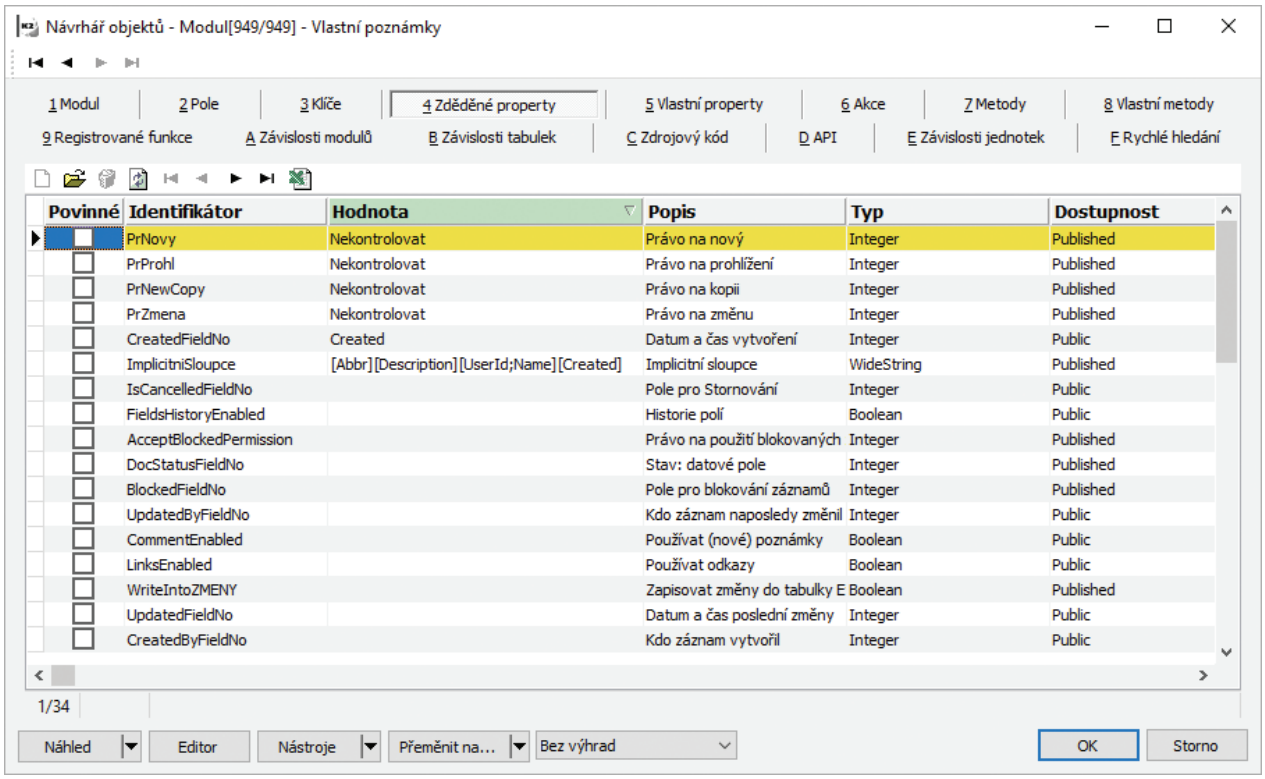
Obrázek 150 - Příklad Vlastní poznámky - datový modul

Přepneme se na záložku ***„Pole“*** a nadefinujeme primární klíč ***„RID“***, dále ***„Abbr“*** – zkratka, typu řetězec. Zde nám návrhář objektů automaticky nastaví toto pole na unikátní, což nepožadujeme, tak příznak zrušíme. Dalším polem bude ***„Description“*** – ***„Popis“***, typu řetězec, pole ***„UserId“***, které je cizím klíčem do datového modulu uživatelů K2. Posledním pak bude ***„Created“*** typu datum, kde se bude ukládat datum a čas vytvoření záznamu, viz Obrázek 151 - Příklad Vlastní poznámky – pole.



Obrázek 151 - Příklad Vlastní poznámky – pole

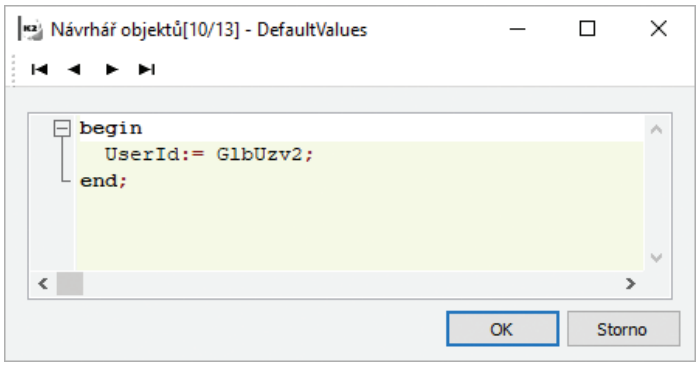
Na záložce ***„Zděděné property“*** nastavíme práva na prohlížení, zápis, kopii apod. na hodnotu ***„-1“***, abychom mohli s modulem pracovat bez omezení. Dále nastavíme výchozí sloupce, aby se zobrazil správně seznam a nakonec nastavíme vlastnost ***„CreatedFieldNo“*** na pole ***„Created“***, abychom měli zajištěno, že se automaticky doplňuje datum vytvoření záznamu, viz Obrázek 152 - Příklad Vlastní poznámky - zděděné property.



Obrázek 152 - Příklad Vlastní poznámky - zděděné property

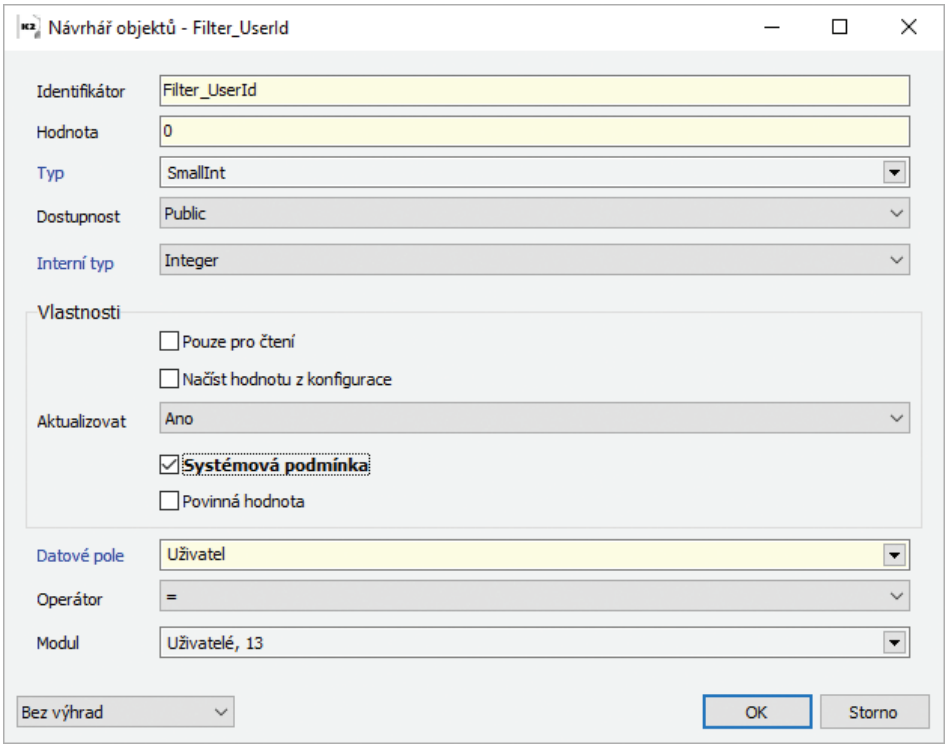
Každý záznam, který budeme vkládat do datového modulu, musí mít nastaveno pole ***„UserId“*** na hodnotu uživatele, který je aktuálně přihlášen. Samozřejmě by tato hodnota neměla jít změnit. Díky tomuto pak dokážeme odfiltrovat záznamy jen pro přihlášeného uživatele. Pole ***„UserId“*** nastavíme na ***„Pouze pro čtení, UI“***. Nastavení hodnoty na aktuálně přihlášeného uživatele zajistíme pomocí metody ***„DefaultValues“***, která slouží k nastavení polí do výchozího nastavení. Kód této metody bude obsahovat pouze řádek ***„UserId := GlbUzv2“***, viz Obrázek 153 - Příklad Vlastní poznámky - výchozí nastaven.

POZNÁMKA: Přiřazení do pole UserId aktuálně přihlášeného uživatele, lze samozřejmě pomocí zděděné property ***„CreatedByFieldNo“*** a tento způsob se také doporučuje. V příkladu je demonstrováno použití metody ***„DefaulValues“***.



Obrázek 153 - Příklad Vlastní poznámky - výchozí nastaven

K zajištění odfiltrování záznamů pak použijeme filtrovací parametr. Přepneme se na stranu ***„Vlastní property“*** a vložíme nový záznam typu ***„Filtrovací parametr“***. Zobrazí se formulář na výběr pole, přes které potřebujeme filtrovat. Zde vybereme pole ***„UserId“***. Ve formuláři definující parametr doplníme do pole hodnota ***„0“*** a zatrhneme volbu ***„Is system condition“***, aby bylo zajištěno, že podmínka nepůjde uživatelem zrušit, viz Obrázek 154 - Příklad Vlastní poznámky - filtrovací parametr.



Obrázek 154 - Příklad Vlastní poznámky - filtrovací parametr

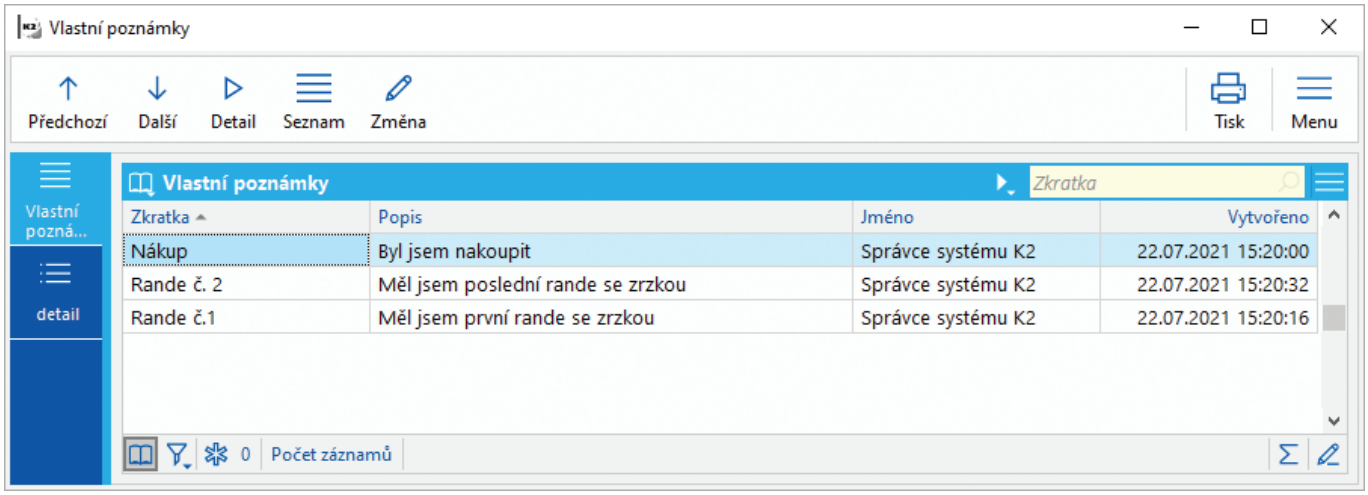
POZNÁMKA: Návrhář objektů zatím neumožňuje v hodnotě definovat výraz. Pro nastavení vytvořené filtrovací vlastnosti, tak musíme využít jiných mechanismů. V našem příkladu spustíme datový modul skrz jiný skript a tam si vlastnost nastavíme. Pokud bychom tak neučinili, hodnota by byla stále ve výchozím stavu, což se v našem případě rovná hodnotě ***„0"***, tím pádem bychom neviděli žádná data.

Po ukončení vytváření datového modulu a spuštění akce ***„Deploy"*** na promítnutí změn do K2, můžeme přistoupit k vytvoření skriptu pro spuštění datového modulu s nastavením filtrovacího parametru.

Skript bude obsahovat vytvoření instance datového modulu, který musíme samozřejmě mít definovaný v sekci ***„modules"***. Na vytvořené instanci nastavíme filtrovací parametr ***„Filter_UserId"*** na hodnotu ***„GlbUzv2"*** (číslo aktuálně přihlášeného uživatele). Poté stačí vytvořit objekt, který slouží ke spuštění datového modulu v univerzálních formulářích – ***„TViewExecutor"*** a přiřadit mu instanci vytvořeného datového modulu.

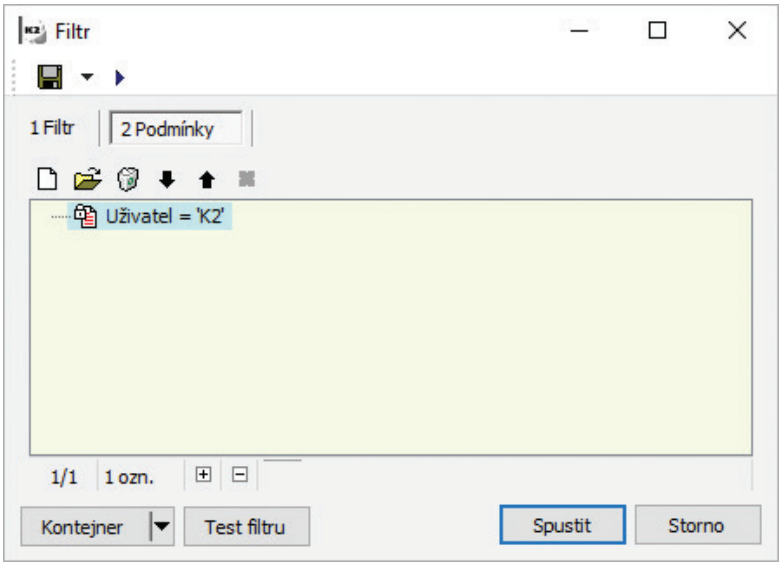
```
uses
  TicTacToe_API;

modules
  TTicTacToeNoteEvidence;

var
  NotesDM : TTicTacToeNoteEvidence;
  VE : TViewExecutor;

begin
  try
    NotesDM := TTicTacToeNoteEvidence.CreateDM;
    NotesDM.SetState(tfVse, NoteEvidence_by_RID, False);
    NotesDM.Filter_UserId := GlbUzv2;

    VE := TUniFormManager.CreateViewExecutor;
    VE.Instance := NotesDM;
    VE.ShowInModalForm(True);
  finally
    NotesDM.Free;
    VE.Free;
  end;
end.
```

Po spuštění skriptu se zobrazí generovaný univerzální formulář, kde budeme mít odfiltrovány pouze ty záznamy, které vytvořil aktuálně přihlášený uživatel, viz Obrázek 155 - Příklad Vlastní poznámky – formulář.



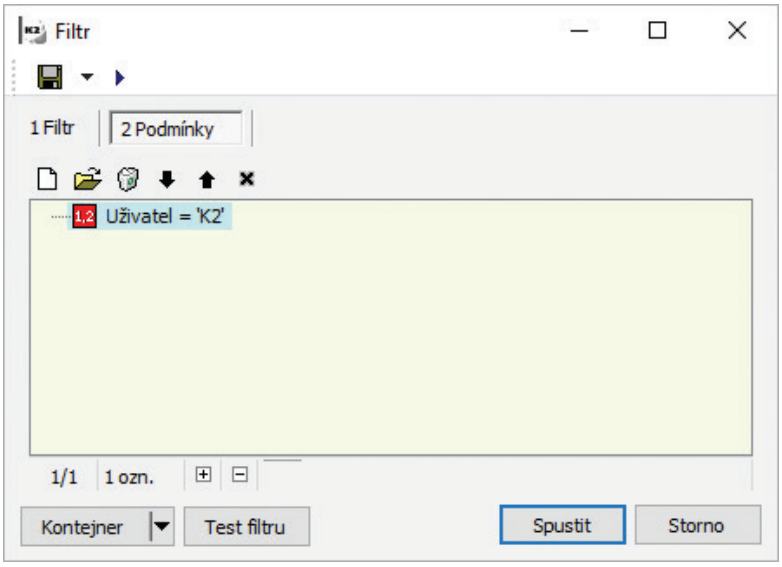
Obrázek 155 - Příklad Vlastní poznámky – formulář

V případě, že vyvoláme pomocí Shift + Ctrl + F9 úpravu podmínek filtru, zobrazí se formulář, kde můžeme vidět, že existuje systémová podmínka na omezení záznamů dle pole ***„UserId“***. Tato podmínka nejde nijak potlačit a v případě, že budeme vytvářet nad modulem filtry, pak bude vždy součástí i tato podmínka. Systémovou podmínku poznáme dle ikony se zámkem, viz Obrázek 156 - Příklad Vlastní poznámky - systémová podmínka.



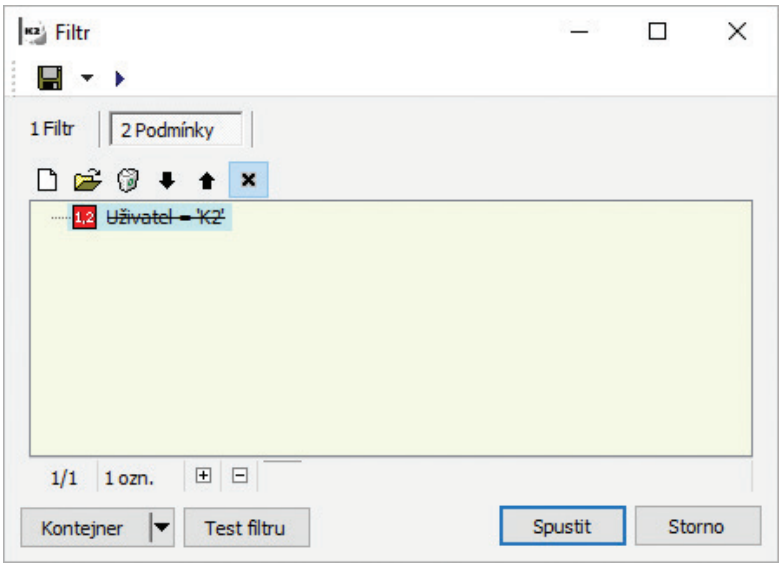
Obrázek 156 - Příklad Vlastní poznámky - systémová podmínka

Na začátku příkladu bylo zmíněno, že si ukážeme, jak je možné podmínku potlačit. Je nutné zrušit nastavení ***„Systémová podmínka“***. Pak se tato podmínka ve správě filtru zobrazuje bez ikony zámku, viz Obrázek 157 - Příklad Vlastní poznámky - podmínka ve filtru.



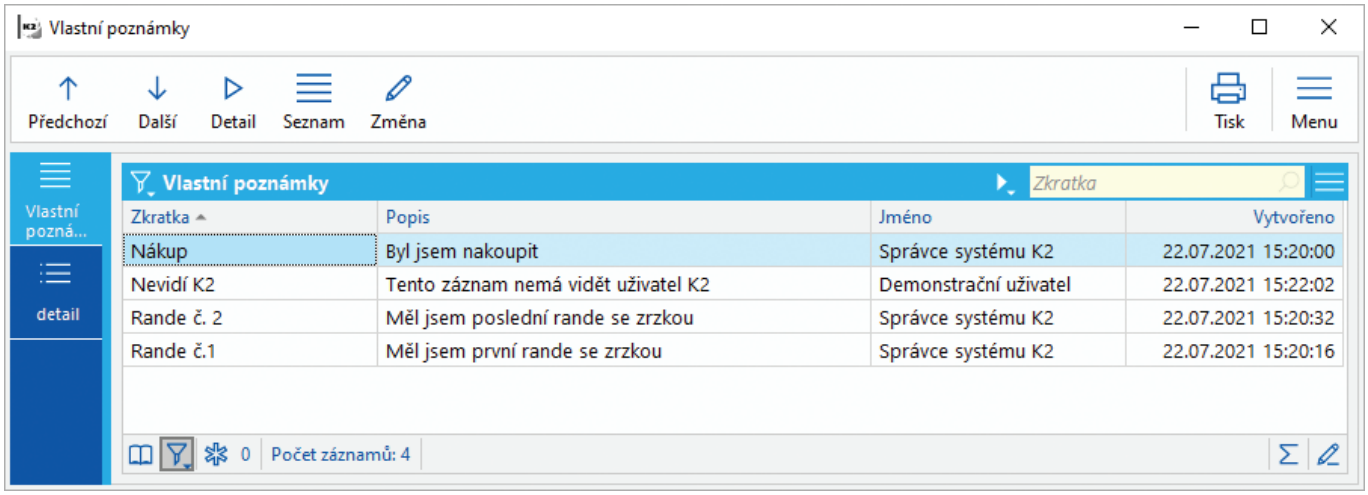
Obrázek 157 - Příklad Vlastní poznámky - podmínka ve filtru

Podmínku potlačíme stisknutím ikony s křížkem nebo otevřením detailu podmínky a nastavením ***„Potlačit"***. Poznáme to tak, že je přeškrtnutá, viz Obrázek 158 - Příklad Vlastní poznámky - potlačená podmínka filtru.



Obrázek 158 - Příklad Vlastní poznámky - potlačená podmínka filtru

Po zobrazení formuláře s deaktivovanou podmínkou, můžeme vidět, že se v modulu nacházejí i jiné záznamy, od dalších uživatelů, viz Obrázek 159 - Příklad Vlastní poznámky - potlačená podmínka filtru - data.

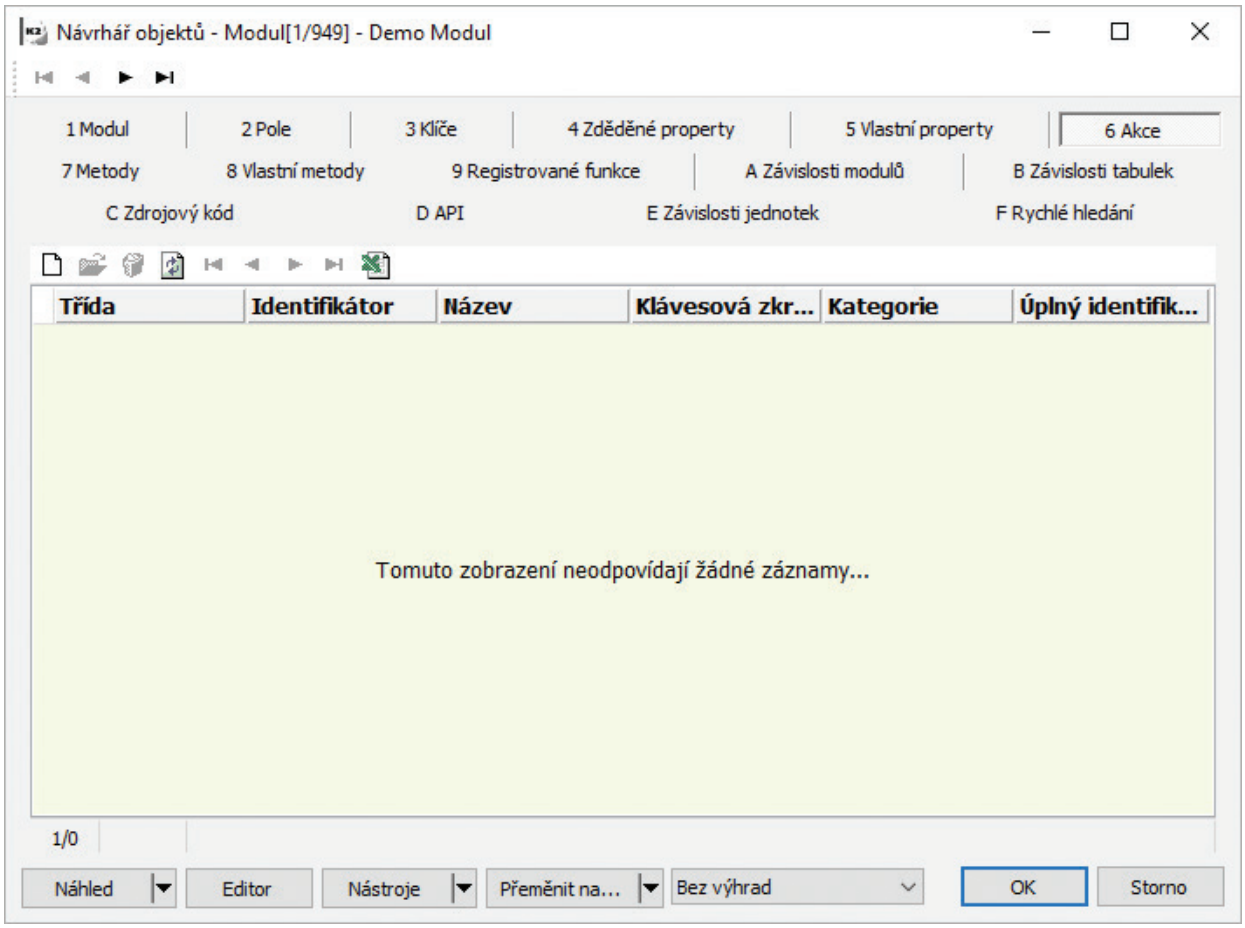


Obrázek 159 - Příklad Vlastní poznámky - potlačená podmínka filtru - data

4.7. AKCE

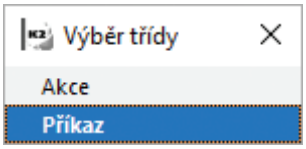
Další důležitou částí vytváření datového modulu, je sekce nazvaná ***„Akce"*** neboli ***„Commandy"***. Zde můžeme definovat metody, které je pak možné v rámci datového modulu spouštět, například stiskem definovaného tlačítka na formuláři.

Nový záznam vložíme pomocí stisknutí tlačítka Nový v nástrojové liště nebo stisknutím klávesy Insert, viz na Obrázek 160 - Seznam akcí datového modulu.



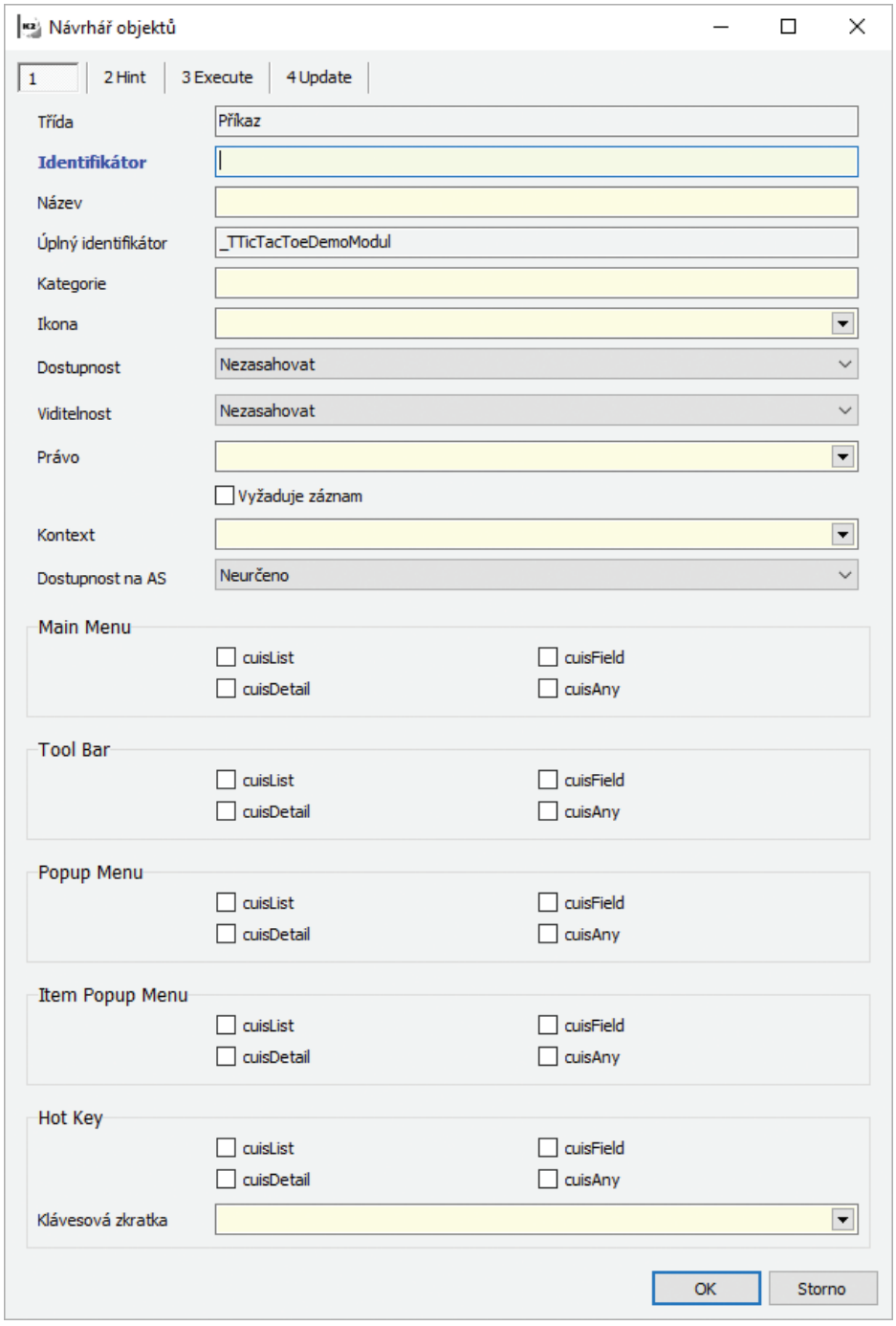
Obrázek 160 - Seznam akcí datového modulu

POZNÁMKA: Vždy při výběru třídy vybíráme možnost „Příkaz“ pro vytvoření nového „commandu“. Možnost „Akce“ je zde zatím z historických důvodů. Viz Obrázek 161 - Výběr možnosti "Příkaz".



Obrázek 161 - Výběr možnosti "Příkaz"

Po vyvolání akce vložení záznamu se zobrazí formulář Obrázek 162 - Vytvoření nové akce (příkazu) v datovém modulu.



Obrázek 162 - Vytvoření nové akce (příkazu) v datovém modulu

4.7.1. ZÁKLADNÍ NASTAVENÍ

V této sekci nalezneme nastavení, které je základem pro definice akce (příkazu). V následující části si je popíšeme.

Identifikátor
Slouží k identifikaci akce (příkazu) - commandu. Musí být unikátní v rámci datového modulu.

Název
Uživatelský název. Pojmenování, které je u akce (příkazu) zobrazeno a pomůže uživateli lépe pochopit, co akce (příkaz) může provádět.

Úplný identifikátor
Identifikátor doplněný o informaci, do kterého datového modulu patří. Zajišťuje unikátnost v celé K2. Používá se ve formulářích a ostatních místech kde je potřeba akci (příkaz) spustit.

Kategorie
Slouží k vytvoření skupiny akcí (příkazů). Například kde vložíme u akcí (příkazů) hodnotu ***„Speciál“***, vznikne v kontextových nabídkách, kde je akce (příkaz) vidět, nejprve pod menu ***„Speciál“*** a až následně akce.

Ikona
Zde definujeme ikonu, která bude zobrazena v případě zařazení akce (příkaz) na formulář. Jakým způsobem je tvořen identifikátor ikony je popsáno v kapitole 4.2.1. Skalár, přesněji v sekci ***„Zobrazovat jako ikony“***.

Dostupnost
Tato vlastnost udává, za jakých okolností je akce (příkaz) dostupná v datovém modulu. Možné nastavení shrnuje následující tabulka. V případě, že nemá být akce (příkaz) dostupná, tlačítko bude ve formuláři existovat, ale nepůjde stisknout.

HODNOTA DOSTUPNOSTI	POPIS
Nezasahovat	Řídí se vnitřní logikou jádra K2
Vždy	Dostupná vždy
Pouze ve změně	Dostupná pouze, když je datový modul ve stavu ***„změna“***
Pouze v prohlížení	Dostupná pouze, když je datový modul ve stavu ***„prohlížení“***
Pouze je-li vlastník ve změně	Používá se v případě položkového modulu. Akce (příkaz) je dostupná pouze, když je nadřízený datový modul ve stavu ***„změna“***
Pouze je-li vlastník v prohlížení	Používá se v případě položkového modulu. Akce (příkaz) je dostupná pouze, když je nadřízený datový modul ve stavu ***„prohlížení“***.

Viditelnost
Tato vlastnost udává, za jakých okolností je akce (příkaz) viditelná v datovém modulu. Možné varianty jsou stejné jako u vlastnosti ***„Dostupnost“*** v předchozí tabulce. V případě, že nemá být akce (příkaz) viditelná, tlačítko nebude ve formuláři vůbec existovat.

Právo
Pomocí této vlastnosti nastavujeme právo na použití akce (příkaz). Tedy pouze uživatelé, kteří budou mít přiděleno právo, které bude nastaveno této akci (příkazu), budou moci tuto akci (příkaz) používat. Hodnotou je tedy vazba do číselníku práv. V případě, že přidělujeme právo, otevírá se nám číselník práv, kde zvolíme patřičné právo, které bude svázáno s použitím této akce (příkazem) uživatelem.

Vyžaduje záznam
Akce (příkaz) vyžaduje ke správné funkčnosti záznam, který je aktuálně vybraný v datovém modulu, který akci vlastní.

Kontext
Dané pole se zatím nevyužívá, jedná se o budoucí přípravu.

Dostupnost na AS
Tato vlastnost udává, za jakých okolností je akce (příkaz) dostupná na aplikačním serveru. Může nabývat hodnot ***„Neurčeno“***, ***„Nezveřejněno“*** a ***„Zvěřejněno“***. Pokud zvolíme ***„Neurčeno“***, bude se řídit jádrem K2. ***„Nezveřejněno“*** nám daný příkaz (command) nezveřejní na AS, opakem je potom možnost ***„Zvěřejněno“***.

Umístění
V této sekci se nachází několik variant umístění s příznaky, které je možné nastavit. Účelem je automatické zpřístupnění akce (příkaz) ve formuláři bez zásahu uživatele. Následující tabulka popisuje jednotlivé varianty umístění s následným popisem nastavení každí z nich.

VARIANTA UMÍSTĚNÍ	POPIS
Main Menu	Akce (příkaz) se zobrazí v hlavním panelu
Tool Bar	Akce (příkaz) se zobrazí v nabídce nad seznamem
Popup Menu	Akce (příkaz) se zobrazí v kontextové nabídce
Item Popup Menu	Akce (příkaz) se zobrazí v kontextové nabídce nad položkou seznamu
Hot Key	Akce (příkaz) funguje na uvedenou zkratkovou klávesu

V každé variantě umístění existují čtyři parametry, které určují, za jakých podmínek se má akce (příkaz) v daném umístění zobrazit.

PARAMETRY UMÍSTĚNÍ	POPIS
cuisList	Akce (příkaz) je dostupná pouze za předpokladu, že se nachází datový modul v režimu seznam
cuisDetail	Akce (příkaz) je dostupná pouze za předpokladu, že se nachází datový modul v režimu detail
cuisField	Akce (příkaz) je dostupná pouze za předpokladu, že je vyvolána nad jedním polem v detailu nebo seznamu
cuisAny	Akce (příkaz) je dostupná ve všech výše zmíněných variantách

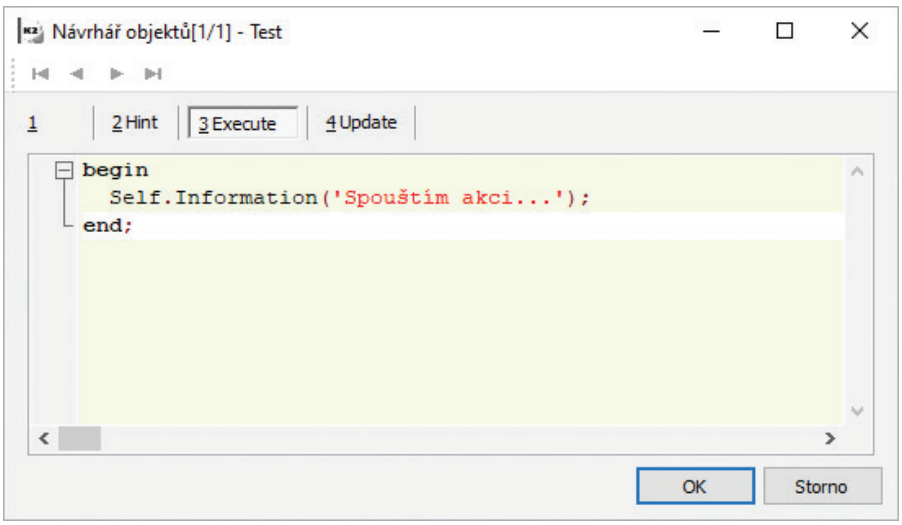
Klávesová zkratka
Slouží k nastavení klávesové zkratky, která po stisknutí ve formuláři vyvolá výkonný kód dané akce (příkazu), tedy akci (příkaz) spustí. Klávesová zkratka se definuje jednoduše textem. Tedy například ***„Ctrl + K“***, ***„Alt + S“***, ***„Ctrl + Alt + K“*** nebo ***„Shift + Ctrl + N“***.

4.7.2. ZÁLOŽKA „HINT“

Tato záložka v definici akce (příkazu), slouží k definici nápovědy, která se zobrazí v případě, že je akce (příkaz) vložena do formuláře a uživatel na akci (příkaz) najedeme myší.

4.7.3. ZÁLOŽKA „EXECUTE“

V této záložce implementujeme již samotný skript, který je vykonán při spouštění akce (příkazu). Přesněji se jedná o implementaci procedury, která má jako parametr proměnnou ***„Sender“*** typu ***„TObject“***, která obsahuje instanci ***„TDataAction“***, která obsahuje informaci o právě spuštěné akci (příkazu). Například kód, který je vidět na Obrázek 163 - Implementace výkonného kódu akce (příkazu) v datovém modulu, zobrazí hlášení ***„Spouštím akci...“*** při spuštění akce z formuláře.



Obrázek 163 - Implementace výkonného kódu akce (příkazu) v datovém modulu

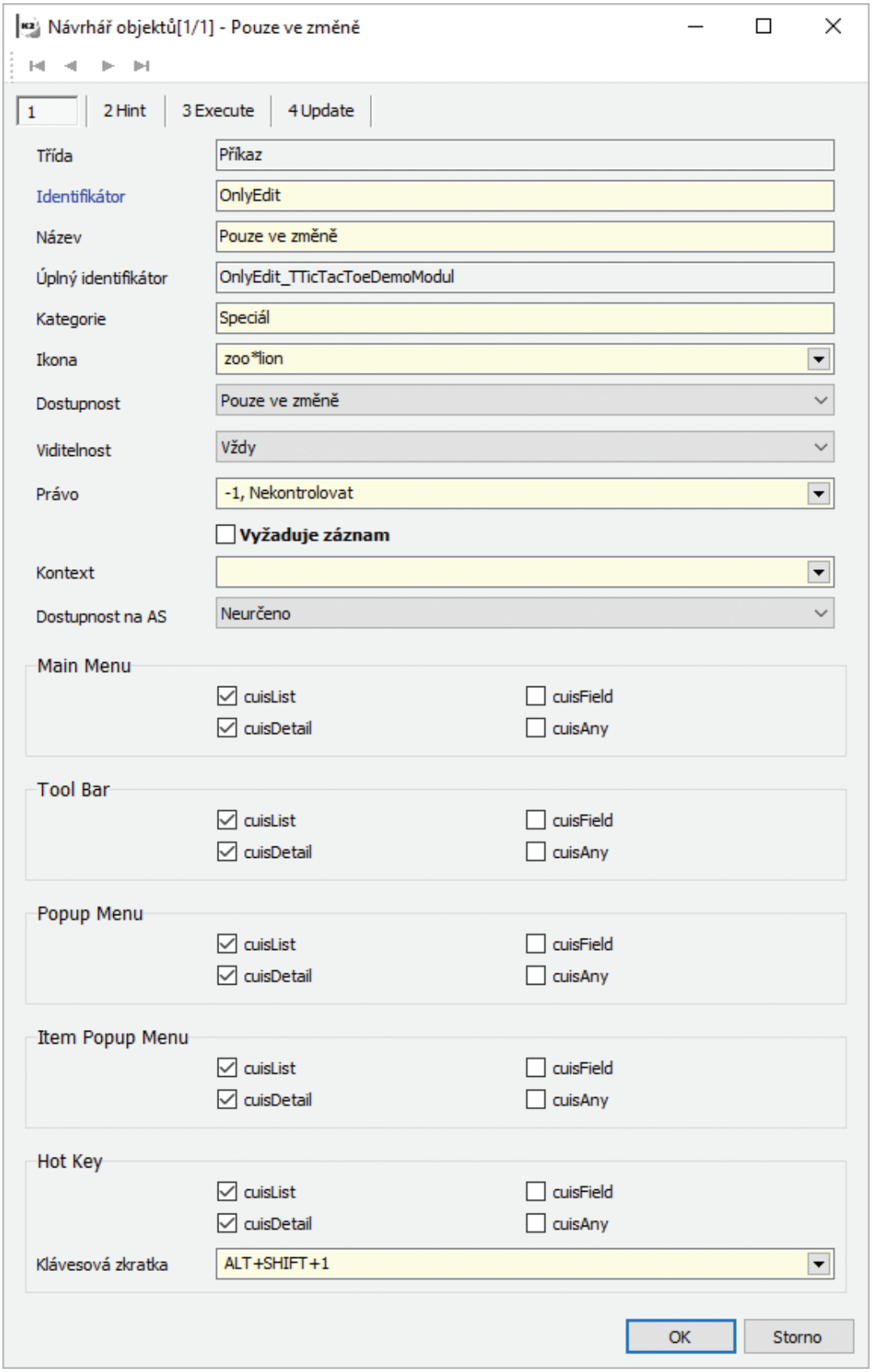
4.7.4. ZÁLOŽKA „UPDATE“

V této záložce implementujeme skript, který je vykonán v případě spouštění akce (příkazu), při překreslení formuláře apod. Zde si můžeme ošetřit spouštění akce (příkazu) tak, aby odpovídalo požadavkům, nebo můžeme na základě definované logiky měnit ostatní parametry akce (příkazu), jako například zobrazenou ikonu. Jedná se o implementaci procedury, která má jako parametr proměnnou ***„Sender“*** typu ***„TObject“***, která obsahuje instanci ***„TDataAction“***, která obsahuje informaci o právě spuštěné akci (příkazu).

POZNÁMKA: Třída ***„TDataAction“*** obsahuje vlastnosti akce (příkaz) jako například název, viditelnost, povolení ke spuštění apod. V metodě ***„Execute“*** a ***„Update“*** pak můžeme k těmto vlastnostem přistupovat a měnit je na základě jiných okolností.

PŘÍKLAD: Abychom lépe pochopili nastavení umístění, dostupnosti a viditelnosti, uvedeme si zde příklad nastavení příkazu. Vytvoříme si příkaz, kterou nazveme ***„Pouze ve změně“*** a jako identifikátor ji nastavíme ***„OnlyEdit“***. Jako ikonu zvolíme hodnotu ***„zoo*lion“***. U dostupnosti zvolíme hodnotu ***„Pouze ve změně“***, díky čemuž se zobrazí, ale pouze ve stavu změna datového modulu, bude tlačítko funkční. Viditelnost necháme nastavenu na ***„Vždy“***. Příkaz bude ve formuláři za všech okolností viditelná. V sekci umístění nastavíme u ***„Main Menu“***, ***„Tool Bar“***, ***„Popup menu“***, ***„Item Popup Menu“*** a ***„Hot Key“*** hodnoty ***„cuisList“*** a ***„cuisDetail“***. Znamená to, že se nám příkaz ve výchozím zobrazení vloží do hlavního menu, do menu nad seznamem, do kontextové nabídky nad datovým modulem, do kontextové nabídky nad jedním záznamem v seznamu a také bude dostupná pomocí klávesové zkratky Alt + Shift + 1. Tím, že jsme všem variantám umístění nastavili

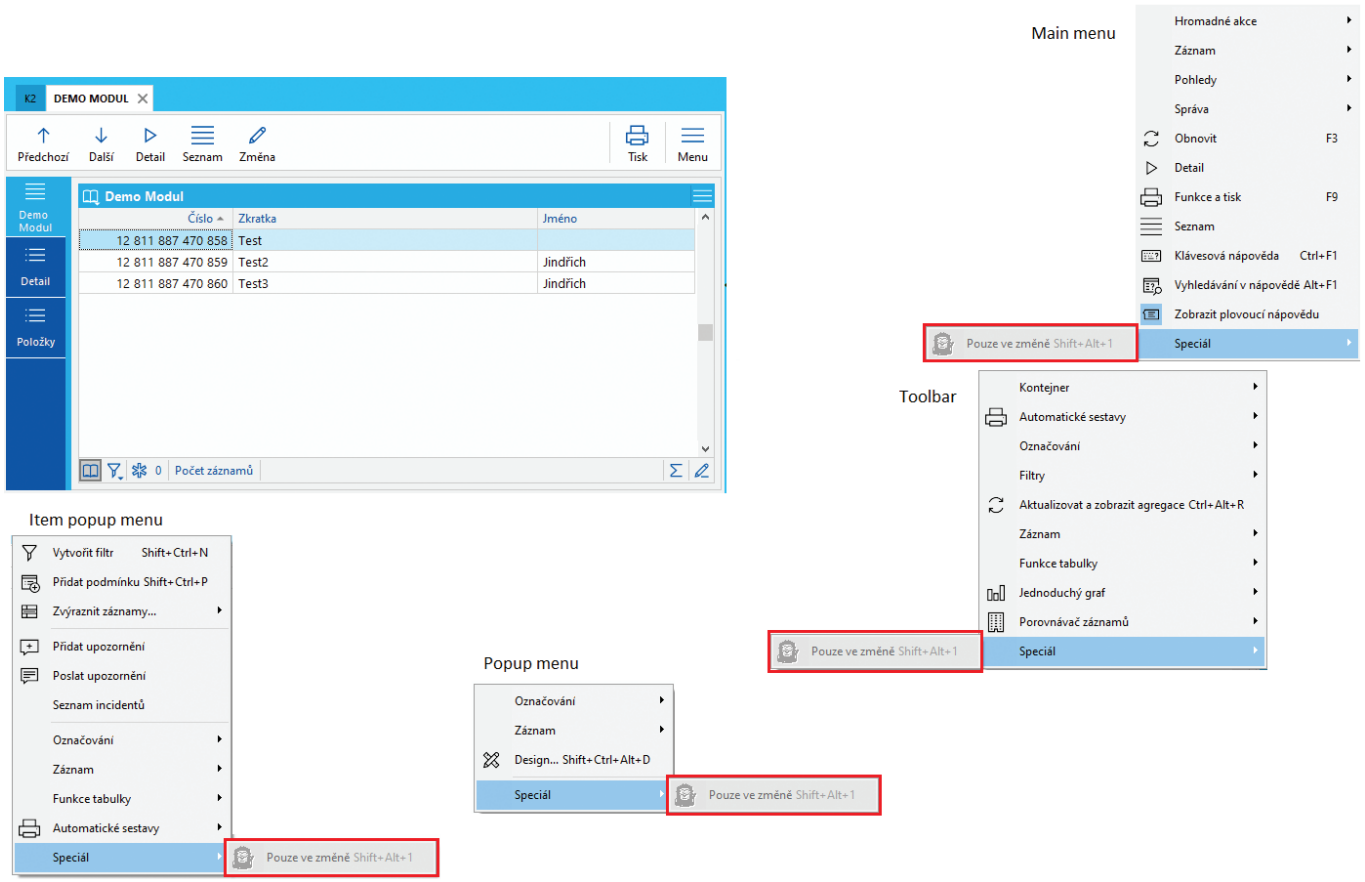
„cuisList“ a *„cuisDetail“* budou ve výše zmíněných oblastech umístěny jak v seznamu, tak v detailu datového modulu. Celé nastavení je vidět na Obrázek 164 - Příklad vytvoření příkazu v datovém modulu.



Obrázek 164 - Příklad vytvoření příkazu v datovém modulu

Po promítnutí změn do K2 a spuštění formuláře máme příkaz ***„Pouze ve změně“*** viditelnou ve všech dostupných umístěních dle naší definice. Na Obrázek 165 - Schéma umístění příkaz z datového modulu ve formuláři je akce vidět ve formuláři dle příkladu. Díky tomuto obrázku je lépe pochopitelné a představitelné nastavení umístění akce. Protože je datový modul ve stavu ***„prohlížení“*** a akci jsme nastavili vlastnost ***„Dostupnost“*** na hodnotu ***„Pouze ve změně“***, je akce ***„šedá“*** – nedostupná. V případě, že bychom akci vůbec nechtěli vidět ve stavu ***„prohlížení“***, nastavili bychom i vlastnost ***„Viditelnost“*** na hodnotu ***„Pouze ve změně“***. Pak by akce úplně z formuláře zmizela do momentu přepnutí do stavu ***„změna“***.

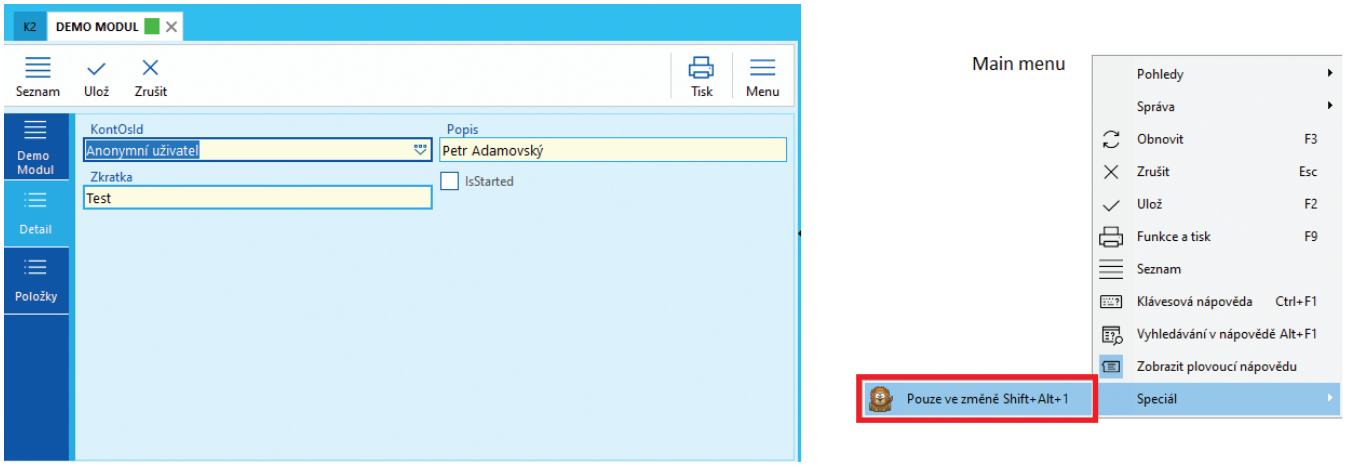
POZNÁMKA: Všimněte si ikony u akce v jednotlivých umístěních a také nápovědy pro klávesovou zkratku, kterou jsme definovali pro danou akci.



Obrázek 165 - Schéma umístění příkaz z datového modulu ve formuláři

Po přepnutí datového modulu do stavu ***„změna“***, se akce ve všech definovaných umístěních zpřístupní dle definice dostupnosti a viditelnosti. Na Obrázek 166 - Schéma umístění příkazu z datového modulu ve formuláři II je vidět zpřístupnění příkazu v ***„Main Menu“***, již není ***„šedá“***, a zároveň je vidět, že příkaz je dostupná i ve stavu ***„Detail“***, který jsme si u příkazu nastavili.

POZNÁMKA: Akce musí mít implementovánu proceduru ***„Execute“***. V případě, že ji mít nebude, zobrazí se ve formuláři, ale nebude spustitelná, tlačítko bude nedostupné – ***„šedé“***.

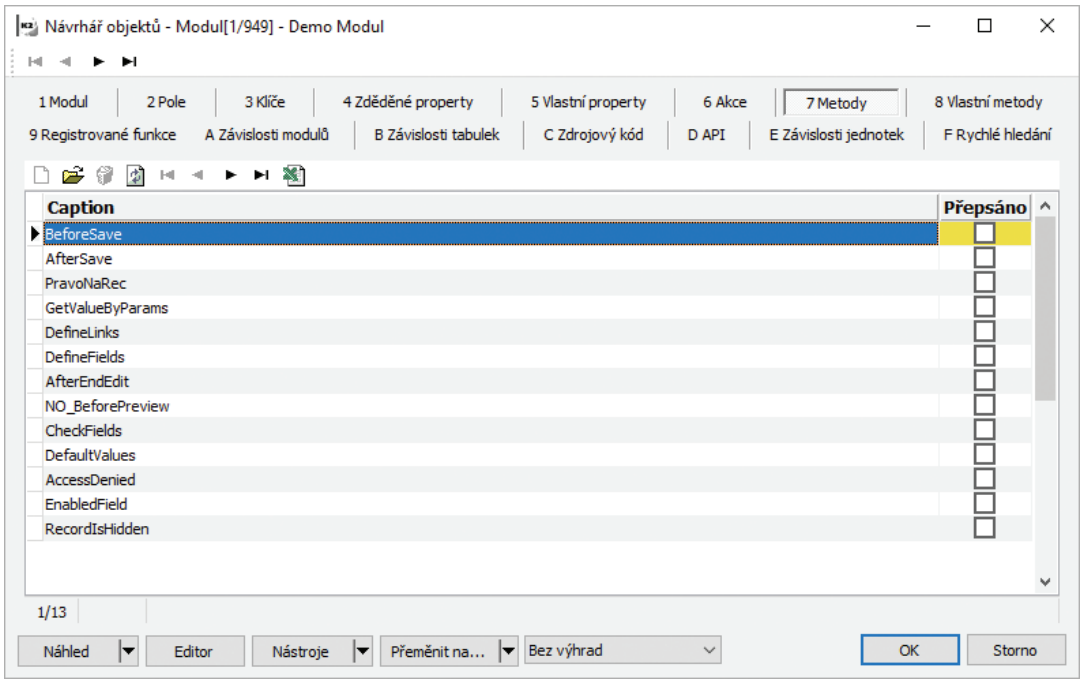


Obrázek 166 - Schéma umístění příkazu z datového modulu ve formuláři II

4.8. METODY

V předchozí kapitole ***„Zděděné property“*** jsme si popisovali, jak používat vlastnosti, které datový modul podědí z datového modulu, ze kterého vychází, tedy který je jeho předkem. Podobným způsobem vytvářený modul získává i metody, které si programátor či administrátor může ve svém datovém modulu implementovat. Na Obrázek 167 - Seznam zděděných metod datového modulu je vidět seznam metod, které jsou v každém datovém modulu k dispozici. Metody jsou používány předkem datového modulu ve specifických situacích. Například ***„před uložením záznamu“***, ***„po uložení záznamu“*** apod. V případě, že potřebujeme v takovýchto okamžicích provést nějakou akci, můžeme metodu, která je vhodná pro naši situaci, implementovat, tedy ***„Přepsat“***.

POZNÁMKA: Metody, které jsme přepsali, mají zatrhnutou volbu ***„Přepsáno“***. Což je pak na první pohled vidět v seznamu metod.



Obrázek 167 - Seznam zděděných metod datového modulu

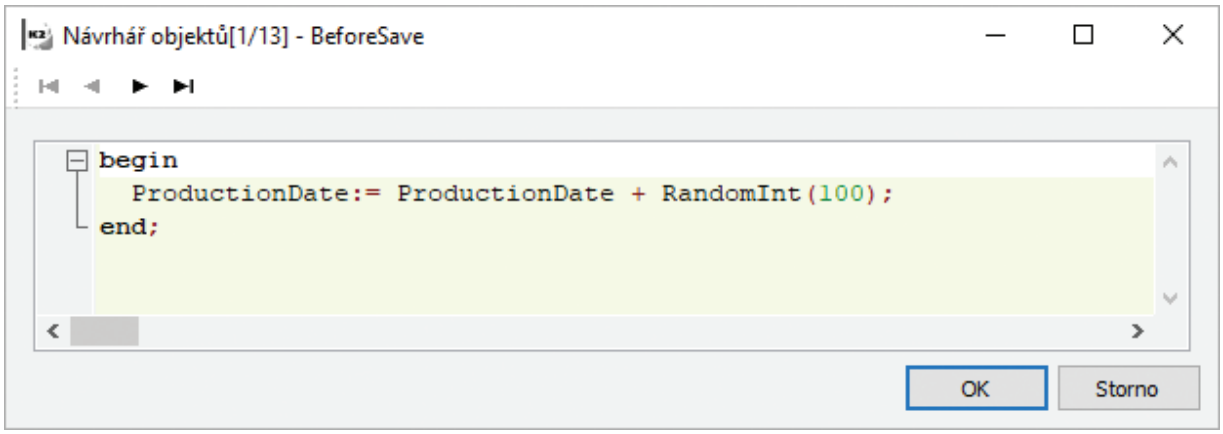
V aktuální části textu si popíšeme jak a k čemu použít jednotlivé metody, které jsou k dispozici k přepsání.

4.8.1. METODA „BEFORESAVE“

Jak již název této metody napovídá, je metoda spuštěna před samotným uložením záznamu v datovém modulu. Je jedno, jestli se jedná o nový záznam či o editaci existujícího. Pokud tedy potřebujeme implementovat akci, která bude vykonána před uložením záznamu, pak je tato metoda správným místem.

Postup implementace metod je stejná jako u předchozích implementací, například v případě počítaného pole, akce nebo zobrazení pole jako ikona, viz Obrázek 168 - Implementace metody v datovém modulu.

POZNÁMKA: Jak již bylo uvedeno v předchozích kapitolách, jednotlivé skriptové implementace je možné provádět pomocí editoru, který je popsaný v kapitole 4.18. Editor. **Použití editoru je podstatně jednodušší a výrazně snižuje chybovost implementace**. Doporučujeme tedy veškeré implementace provádět pomocí editoru a formulář, který je vidět na obrázku využívat spíše k náhledu na implementaci jedné metody.



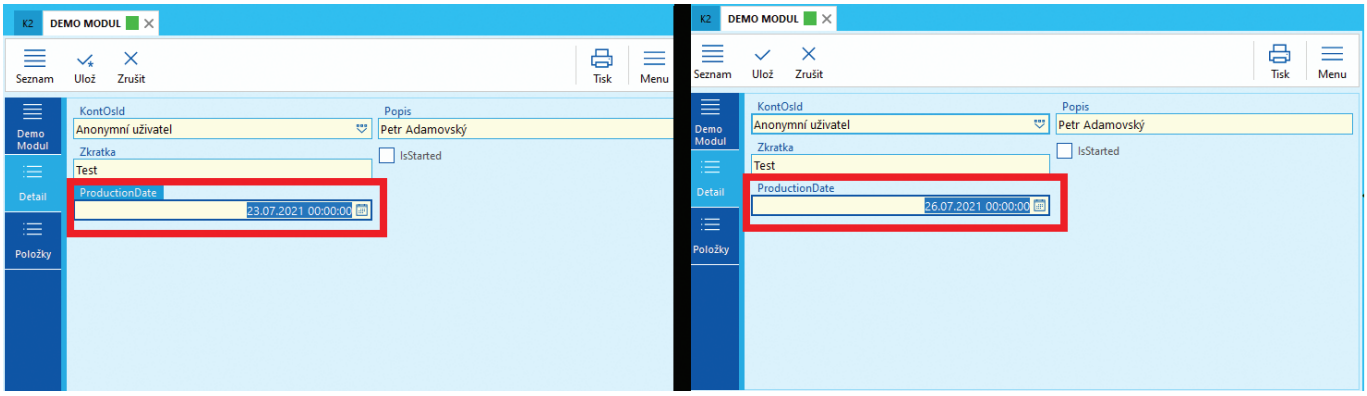
Obrázek 168 - Implementace metody v datovém modulu

Hlavička procedury
procedure BeforeSave;

Pro tuto metodu se implementuje procedura, která nemá žádné vstupní parametry.

PŘÍKLAD: V následujícím příkladu je před uložením záznamu k vybrané hodnotě v poli ***„ProductionDate“*** typu ***„TDateTime“*** připočteno náhodné číslo z rozsahu 0 až 100, což způsobí posun ve dnech dle náhodného čísla. Do pole ***„ProductionDate“*** zapíše dnešní datum a jak vidíme, při uložení se dané datum změní, o náhodné číslo, viz Obrázek 169 - Ukázka použití metody "BeforeSave" v datovém modulu (vlevo před uložením, vpravo po uložení).

```
begin
  ProductionDate := ProductionDate + RandomInt(100)
end;
```



Obrázek 169 - Ukázka použití metody "BeforeSave" v datovém modulu (vlevo před uložením, vpravo po uložení)

POZNÁMKA: Při ukládání záznamu by se nemělo používat zobrazování hlášení.

4.8.2. METODA „AFTERSAVE“

Metoda je spuštěna po samotném uložení záznamu v datovém modulu. Je jedno, jestli se jedná o nový záznam či o editaci existujícího. Pokud tedy potřebujeme implementovat akci, která bude vykonána po ukládání záznamů, pak je tato metoda správným místem.

Hlavička procedury
procedure AfterSave;

Pro tuto metodu se implementuje procedura, která nemá žádné vstupní parametry.

4.8.3. METODA „PRAVONAREC“

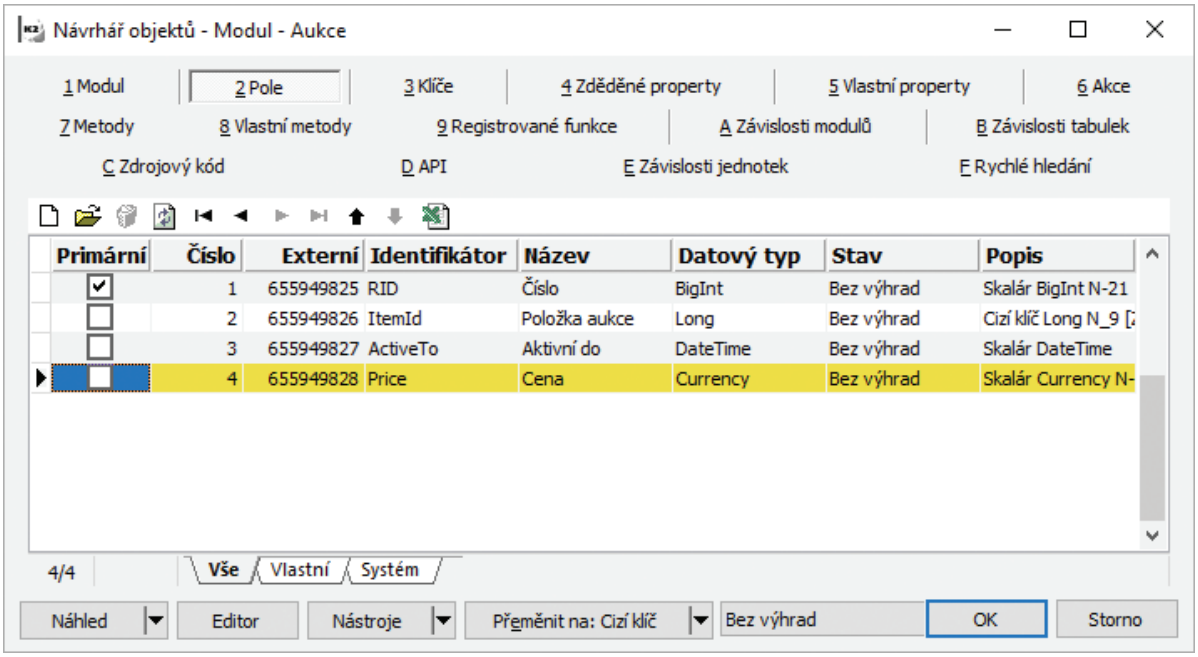
Metoda slouží ke zjištění, zda má aktuální uživatel právo na záznam. V případě, že potřebujeme tuto funkčnost implementovat vlastní logikou, pak použijeme tuto metodu.

Hlavička funkce
function PravoNaRec(AZmena: Boolean): Boolean;

Pro tuto metodu se implementuje funkce, která má na vstupu jeden parametr, který je typu ***„boolean“*** a je nazvaný ***„AZmena“***. Určuje, zda se záznam nachází ve stavu ***„změna“***. Návratovým typem funkce je také ***„boolean“***. V případě, že uživatel má právo na záznam, vrací se ***„true“***, jinak ***„false“***.

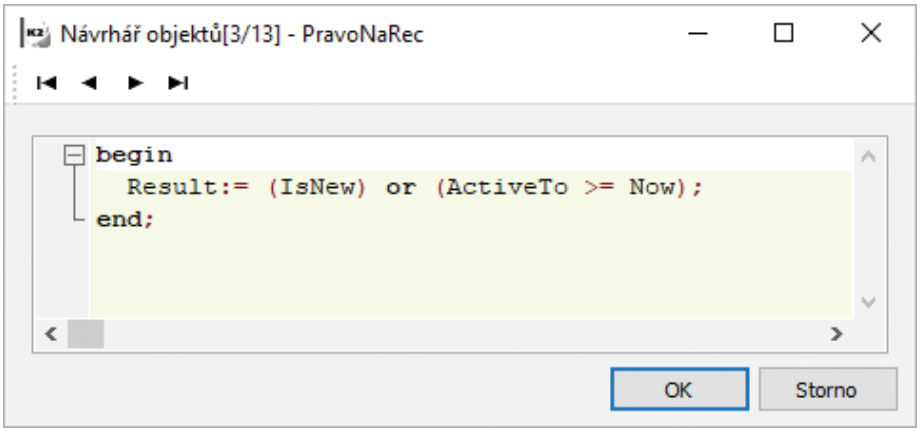
PŘÍKLAD: Ukázku funkčnosti této metody si ukážeme na uměle vytvořeném příkladu online aukčního portálu. Ten nabízí k prodeji dražené zboží, které se zobrazuje v případě, že je nabídka aktivní. Potom co uplyne, ztrácí prohlížející právo na ukončené akce.

Máme datový modul, kde je k dispozici klíčové pole ***„RID“***, položka k dražbě ***„ItemId“***, která je cizím klíčem do zboží, datum do kdy je aukce aktivní ***„ActiveTo“*** a cena za zboží ***„Price“***, viz Obrázek 170 - Definice polí v DM pro příklad použití metody PravoNaRec.



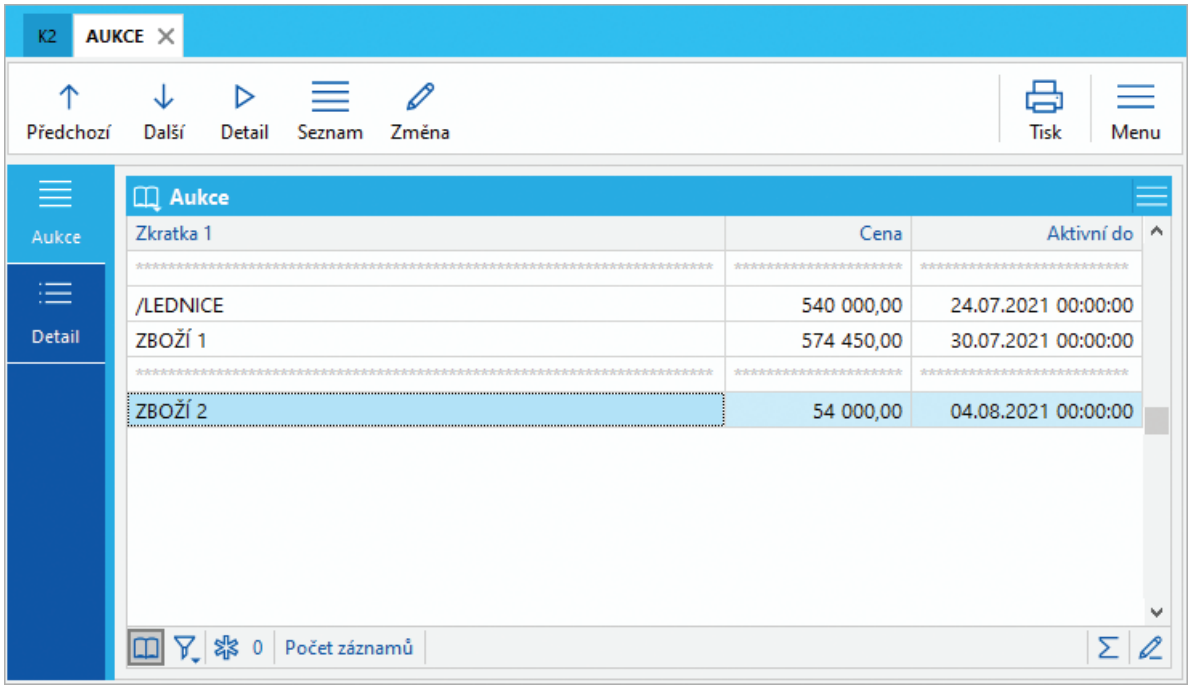
Obrázek 170 - Definice polí v DM pro příklad použití metody PravoNaRec

V metodě ***„PravoNaRec“*** pak imlementujeme jednoduchý skript, viz Obrázek 171 - Příklad implementace metody datového modulu PravoNaRec, který vrací, že má uživatel právo na záznam pouze za předpokladu, že je nový nebo je datum ***„ActiveTo“*** menší rovno aktuálnímu datu a času.



Obrázek 171 - Příklad implementace metody datového modulu PravoNaRec

Po provedení akce ***„Deploy“***, otevřeme datový modul, vložíme pár záznamů a u těch, kterým postupně uplyne doba můžeme pozorovat, jak se stávají nedostupnými přihlášenému uživatelu. Ve všech sloupcích jsou zobrazeny hvězdičky, viz Obrázek 172 - Výsledek příkladu na použití metody datového modulu PravoNaRec.



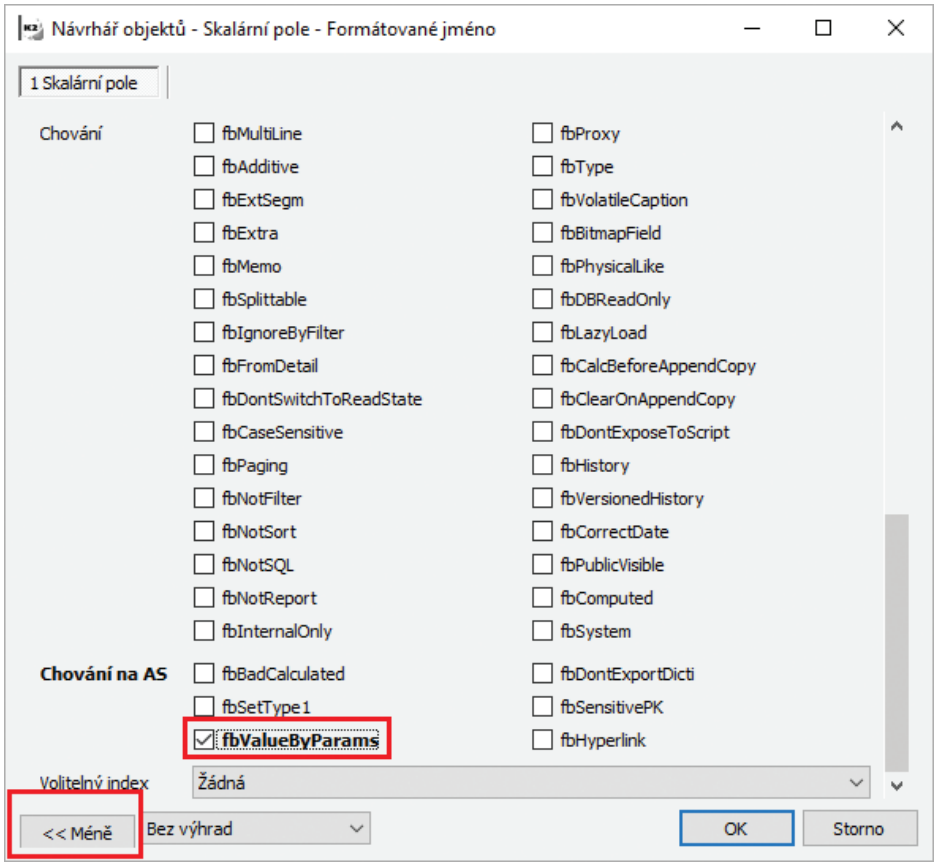
Obrázek 172 - Výsledek příkladu na použití metody datového modulu PravoNaRec

4.8.4. METODA „GETVALUEBYPARAMS“

Slouží k implementaci logiky čtení polí, která jsou parametrická. **Tuto metodu lze použít pouze pro server webových služeb**. Pomocí webových služeb můžeme speciální konvencí zažádat o pole, kterému předáme parametr a v této metodě implementujeme logiku, pomocí které vyhodnotíme vstupní parametry pro pole a vrátíme patřičnou hodnotu.

POZNÁMKA: Speciální konvence pro dotaz na parametrické pole je ve tvaru: ***„název pole^hodnota parametru“***. Používá se při skládání URL dotazů na webové služby v sekcích, ve kterých se pracuje s poli datových modulů, například ***„fields“*** nebo ***„conditions“***. Příklad použití speciální konvence bude popsán v příkladu níže.

Pokud chceme některé pole parametrizovat, je nutné nejprve takové pole označit speciálním příznakem. Ten najdeme přímo v definici pole v pokročilejší části nastavení. Na levé straně formuláře je tlačítko Více. Po jeho stisknutí se zobrazí další možná nastavení pro pole datového modulu. Najdeme zde sekci, která se vztahuje pouze k nastavení chování pro aplikační server. Konkrétně nás zajímá příznak ***„fbValueByParams“***, který na poli musíme nastavit, viz Obrázek 173 - Definice pole pro podporu metody GetValueByParams.



Obrázek 173 - Definice pole pro podporu metody GetValueByParams

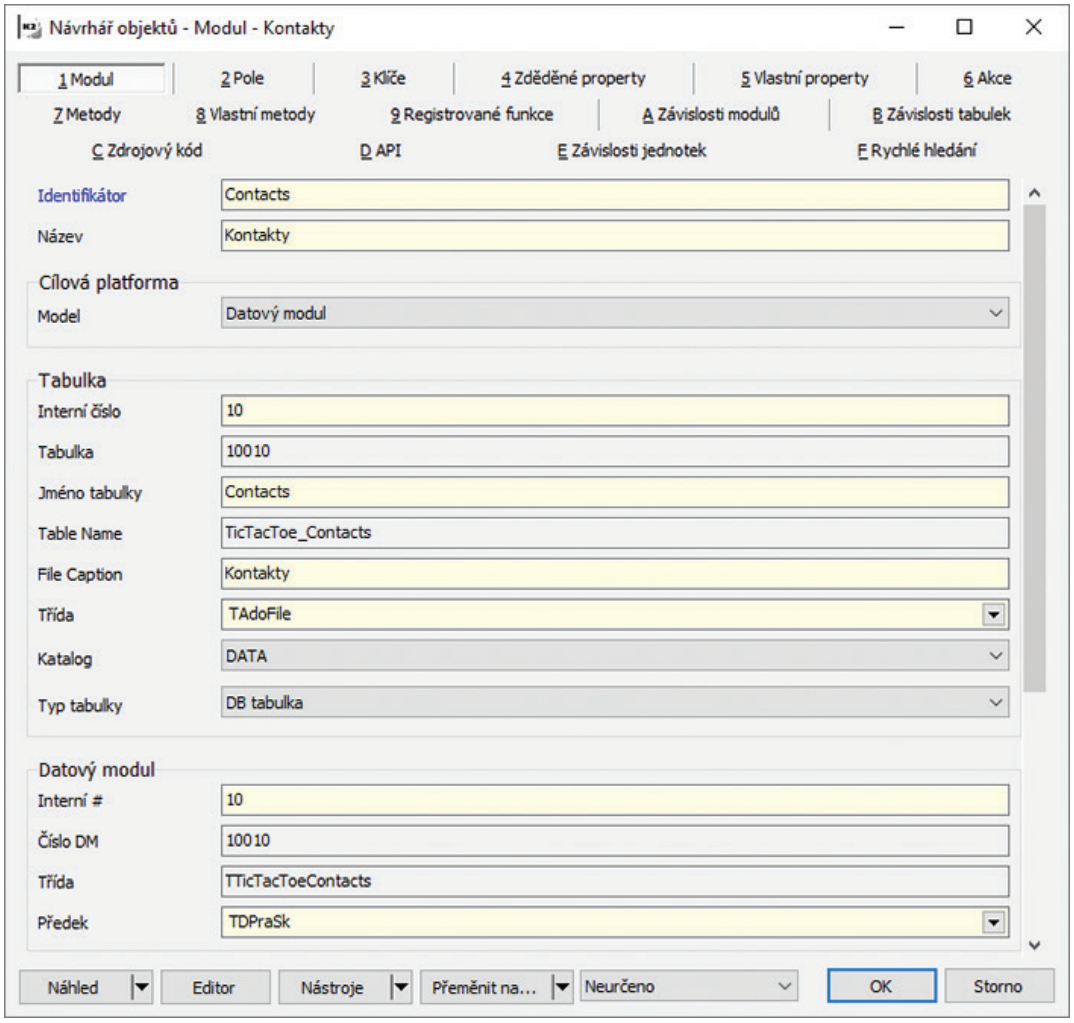
Následně můžeme implementovat metodu „GetValueByParams". Více uvidíme v příkladu níže.

Hlavička funkce
function ValueByParams(ADataField: TDataField; AParams: string): Variant;

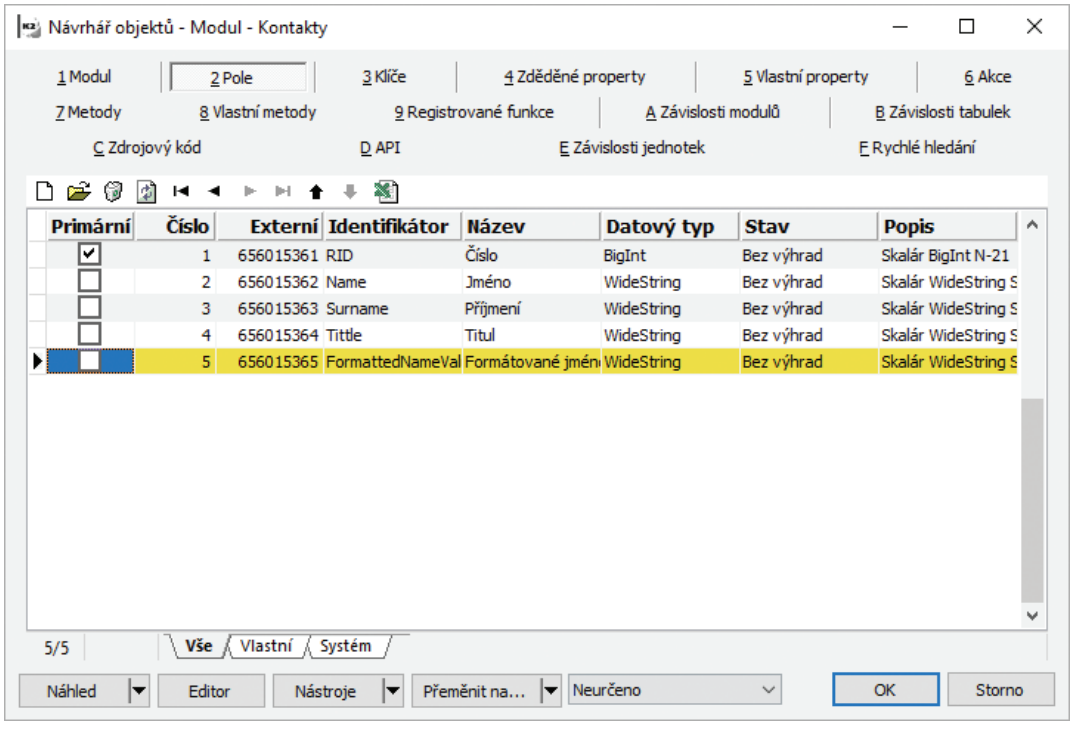
Pro tuto metodu se implementuje funkce, která má na vstupu dva parametry. Prvním je proměnná s názvem ***„ADataField"*** a s datovým typem ***„TDataField"***. Jedná se o instanci pole, pro které se bude v tomto volání vyhodnocovat logika parametrů. Druhým vstupním parametrem, je proměnná s názvem ***„AParams"*** a s datovým typem ***„string"***. Zde nalezneme hodnotu parametru, která byla předána speciální konvencí na server webových služeb. Návratovým typem funkce je typ ***„Variant"***. Funkce vrací hodnotu pole v závislosti na jeho vstupním parametru.

Pro lepší pochopení si vše popíšeme na příkladu.

PŘÍKLAD: Vytvoříme datový modul pro evidenci kontaktů, viz Obrázek 174 - Datový modul Kontakty pro příklad použití funkce GetValueByParams. Budeme evidovat jméno, příjmení a titul. Vytvoříme počítané pole formátované jméno ***„FormattedNameValue"***, pomocí kterého budeme zobrazovat formátovanou hodnotu jména, viz Obrázek 175 - Seznam polí v příkladu na funkci GetValueByParams. Jméno se může zobrazit ve tvaru ***„titul příjmení jméno"***, ***„příjmení jméno"*** nebo ***„jméno příjmení"***. Jak se zobrazí počítané pole, definujeme právě pomocí parametru, pro který implementujeme logiku ve funkci ***„GetValueByParams"***.

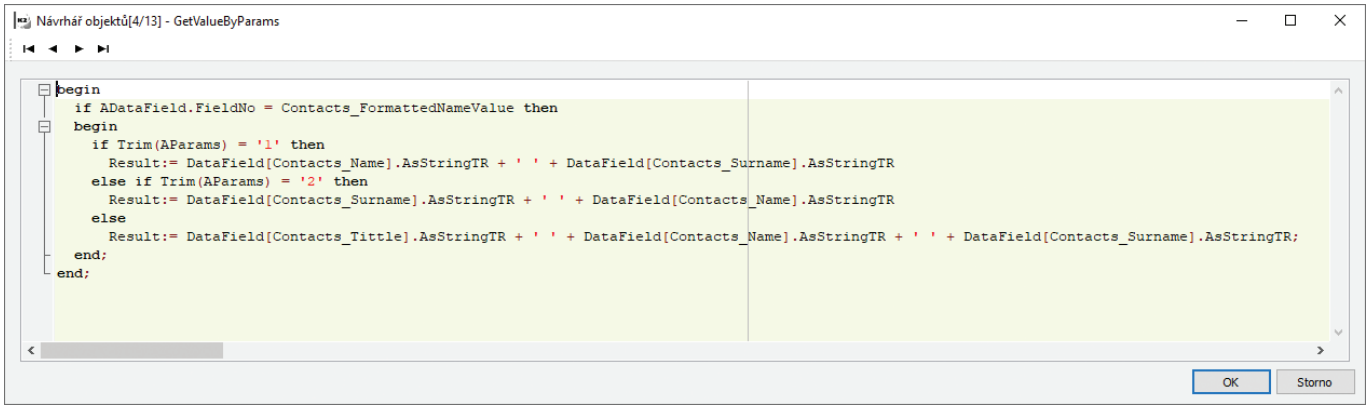


Obrázek 174 – Datový modul Kontakty pro příklad použití funkce GetValueByParams



Obrázek 175 – Seznam polí v příkladu na funkci GetValueByParams

Na poli ***„FormattedNameValue“*** nastavíme dle Obrázek 173 - Definice pole pro podporu metody GetValueByParams příznak ***„fbValueByParams“*** a implementujeme logiku v metodě ***„GetValueByParams“***, viz Obrázek 176 - Zdrojový kód pro funkci GetValueByParams. Implementace říká, v případě, že vyhodnocujeme parametr pro pole ***„FormattedNameValue“***, pak jestliže hodnota parametru ***„AParams“*** je rovna ***„1“*** zobraz hodnotu jako ***„jméno příjmení“***. Pokud bude hodnota parametru ***„2“***, zobrazí se hodnota pole jako ***„přijmení jméno“***. V jiných případech je zobrazeno jako ***„titul jméno příjmení“***.



Obrázek 176 - Zdrojový kód pro funkci GetValueByParams

Použití parametru v URL dotazu pak můžeme vidět na následujících variantách dotazů, které popisují všechny dostupné varianty.

http://localhost/RestService/Data/TNODemoContacts?fields=FormattedNameValue^1


```
Items:
  0:
    DOClassName:   "TNODemoContacts"
    FieldValues:
      0:
        Name:      "RID"
        Value:     9702331121665
      1:
        Name:      "FormattedNameValue^1"
        Value:     "Petr Novák"
```


Obrázek 177 - Výstup webové služby v příkladu GetValueByParams - varianta parametru 1

http://localhost/RestService/Data/TNODemoContacts?fields=FormattedNameValue^2


```
Items:
  0:
    DOClassName:   "TNODemoContacts"
    FieldValues:
      0:
        Name:      "RID"
        Value:     9702331121665
      1:
        Name:      "FormattedNameValue^2"
        Value:     "Novák Petr"
```


Obrázek 178 - Výstup webové služby v příkladu GetValueByParams - varianta parametru 2

http://localhost/RestService/Data/TNODemoContacts?fields=FormattedNameValue^3

Obrázek 179 - Výstup webové služby v příkladu GetValueByParams - varianta parametru 3

4.8.5. METODA „DEFINELINKS“

Procedura slouží k definici vazeb (cizích klíčů) na jednotlivá pole v datovém modulu. Je volána již při sestavování instance datového modulu.

Hlavička procedury
procedure DefineLinks;

Pro tuto metodu se implementuje procedura, která nemá žádné vstupní parametry.

4.8.6. METODA DEFINEFIELDS

Procedura slouží k definici polí datového modulu. Je volána již při sestavování instance datového modulu.

Hlavička procedury
procedure DefineFields;

Pro tuto metodu se implementuje procedura, která nemá žádné vstupní parametry.

4.8.7. METODA „AFTERENDEDIT“

Metoda je spuštěna po ukončení editace záznamu v datovém modulu.

Hlavička procedury
procedure AfterEndEdit;

Pro tuto metodu se implementuje procedura, která nemá žádné vstupní parametry.

4.8.8. METODA NO_BEFOREPREVIEW

Metoda se spouští pouze při otevírání modulů jako náhled z návrháře objektů. Jedná se o metodu, která slouží pouze k ladění modulu, ve kterém potřebujeme pro jeho správnou funkčnost, nejprve nastavit nějaké definované parametry.

Hlavička procedury
procedure DataMBeforePreview

Pro tuto metodu se implementuje procedura, která nevyžaduje vstupní parametry.

4.8.9. METODA „CHECKFIELDS“

Metoda slouží ke kontrole polí při ukládání záznamů v datovém modulu. Programátor ji může využít ke kontrole správnosti naplnění polí a v případě, že je potřeba ukládání přerušit, vyvolá pomocí procedury ***„RDSHlas“*** ukončení této procedury s patřičným hlášením o důvodu neuložení záznamu.

Hlavička procedury
procedure CheckFields(ANovy: Boolean);

Pro tuto metodu se implementuje procedura, která má jeden vstupní parametr, který je typu ***„boolean“*** a určuje, zda se jedná o ukládání nového záznamu či o editaci existujícího.

POZNÁMKA: Procedura ***„CheckFields“*** se ukončuje voláním procedury ***„RDSHlas“*** v případě, že nelze záznam uložit, protože neprošel naší kontrolou. V žádném případě není možné proceduru ukončovat pomocí vyvolání výjimky pomocí ***„raise K2Except.Create“***.

PŘÍKLAD: Jako příklad použití procedury si představme následující implementaci. V případě, že je vyvolán pokus o uložení záznamu v datovém modulu, je provedena kontrola, zda hodnota v poli ***„Abbr“*** začíná písmenem ***„a“***. V případě, že ne, je vyvoláno hlášení a datový modul zůstává v editačním režimu. V případě, že kontrola projde, záznam je uložen.

```
begin
  if not AnsiStartsStr('a', Abbr) then
     RDSHlas(Self,'Vyplněná zkratka nezačíná na písmeno "a"!');
end;
```

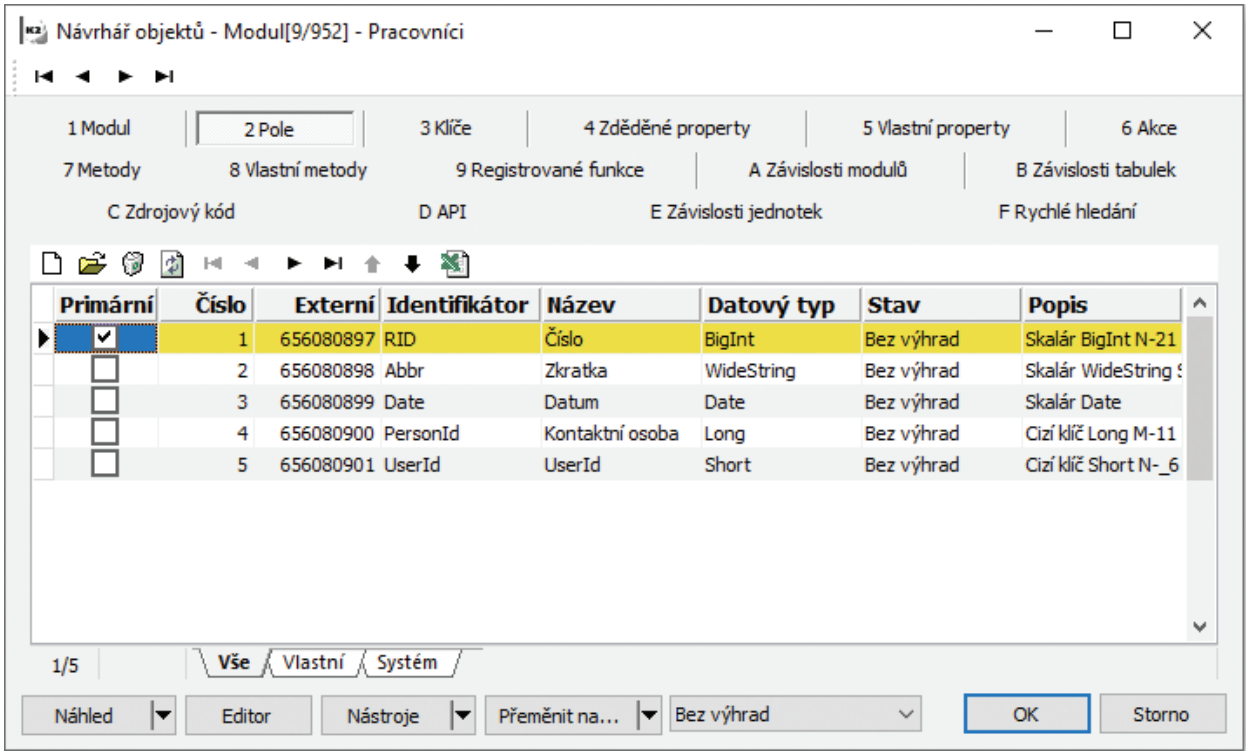
4.8.10. METODA „DEFAULTVALUES“

Procedura ***„DefaultValues“*** slouží k nastavení polí na výchozí hodnoty dle logiky, kterou si programátor implementuje právě v této proceduře. Procedura se spouští v momentě vytváření nového záznamu ještě předtím, než se zobrazí uživateli ve formuláři.

Hlavička procedury
procedure DefaultValues(AEditMode: TRecordEditMode);

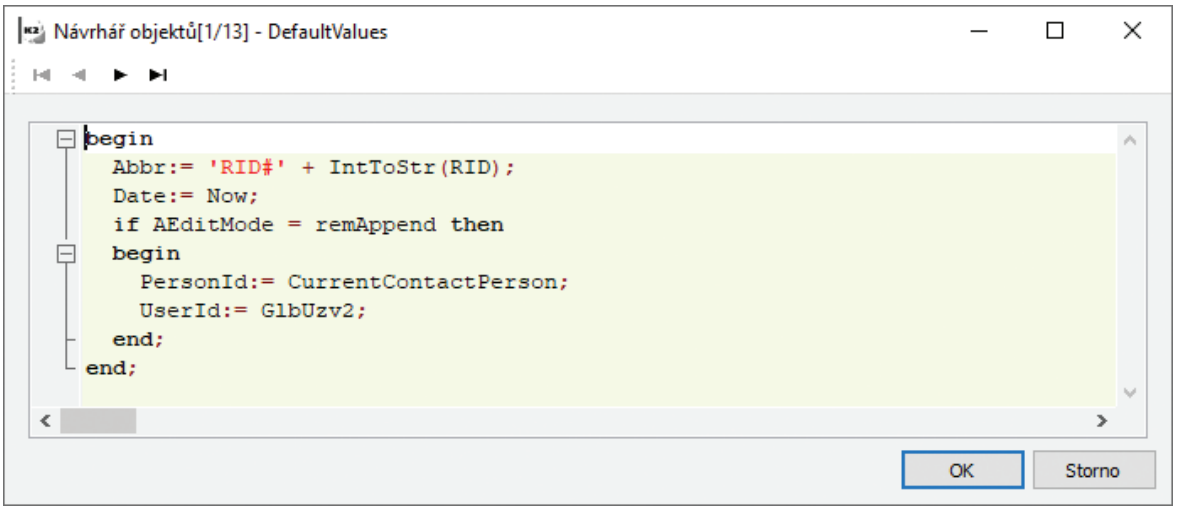
Pro tuto metodu se implementuje procedura, která má jeden vstupní parametr, který je typu výčtový typ ***„TRecordEditMode“*** a určuje, zda se jedná o nový záznam či nový záznam pomocí kopie. Výčtový typ může nabývat hodnoty ***„remAppend“***, což znamená, že se jedná o nový záznam nebo ***„remAppendCopy“*** vyjadřující nový záznam pomocí kopie.

PŘÍKLAD: Jako příklad použití procedury si představme následující situaci. Vytvoříme si datový modul pracovníků ***„Workers“***, kde vytvoříme pole ***„RID“***, pole pro zkratku – ***„Abbr“***, pole pro datum – ***„Date“***, pole pro kontaktní osobu – ***„PersonId“*** s odkazem do kontaktních osob a pole pro uživatele – ***„UserId“*** s odkazem do uživatelů, viz Obrázek 180 - Příklad na metodu "DefaultValues" - založení polí pro modul.



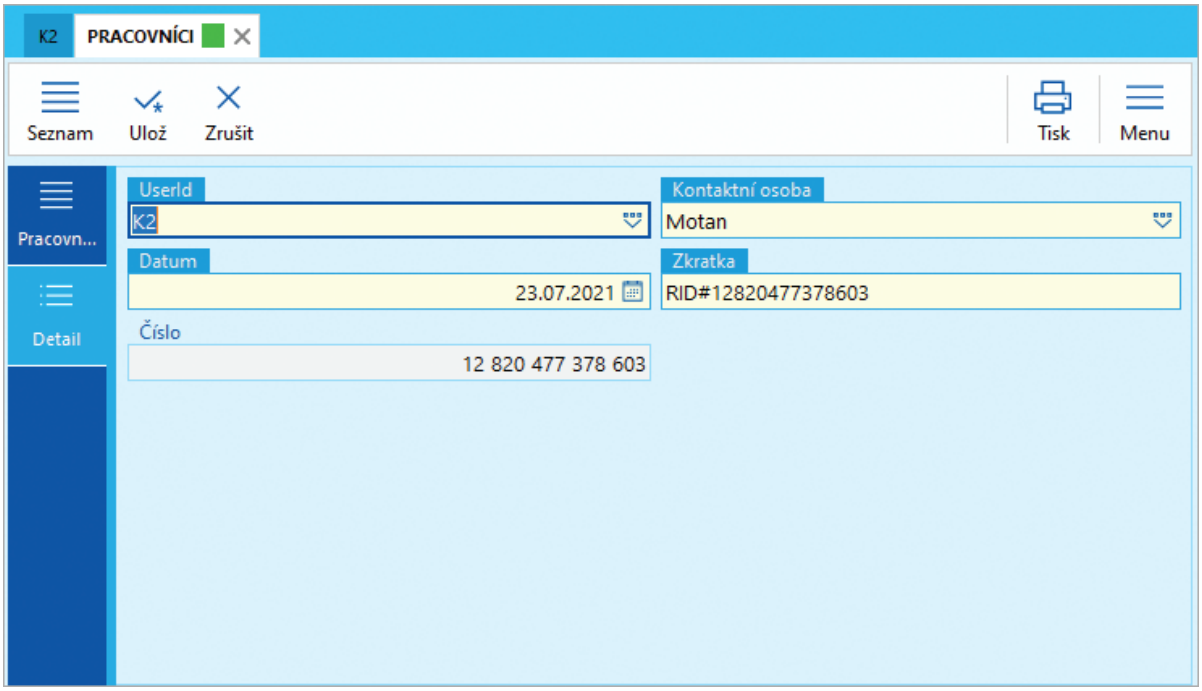
Obrázek 180 - Příklad na metodu "DefaultValues" - založení polí pro modul

Chtěli bychom, aby při vytvoření nového záznamu byla tato pole přednastavená dle následující logiky. Do pole ***„Abbr“*** vložíme text ***„RID#“*** a připojíme hodnotu primárního klíče, do pole ***„Date“*** vložíme aktuální datum, do pole ***„ContactPersonId“*** vložíme, za předpokladu, že se jedná o čistý nový záznam, hodnotu aktuálně přihlášené kontaktní osoby, a do pole ***„UserId“***, za předpokladu, že se jedná o čistý nový záznam, vložíme hodnotu aktuálně přihlášeného uživatele K2. Implementace vypadá poté jako na Obrázek 184 - Příklad použití metody "AccessDenied" v datovém modulu.



Obrázek 181 - Příklad na metodu "DefaultValues" - kód metody

Po promítnutí změn do K2 a otevření náhledového formuláře, můžeme, při zakládání nového záznamu vidět automaticky vyplněný formulář dle naší logiky viz Obrázek 182 - Příklad na metodu "DefaultValues“ - použití.



Obrázek 182 - Příklad na metodu "DefaultValues" - použití

4.8.11. METODA „ACCESSDENIED“

Procedura ***„AccessDenied“*** slouží k řízení práce se záznamy v datovém modulu. Umožňuje na základě podmínek na konkrétní hodnoty aktuálního záznamu, či na stav záznamu – nový, editace, kopie apod. zakázat přístup k záznamům či operacím. Tedy mohu si například zakázat editaci záznamů na základě hodnot polí nebo můžu úplně zakázat vkládat nové záznamy pomocí kopie apod.

Hlavička procedury
procedure AccessDenied(AEditMode: TRecordEditMode; var ADeny: Boolean; var AReason: string);

Pro tuto metodu se implementuje procedura, která má vstupní parametr, který je typu výčtový typ ***„TRecordEditMode“*** a určuje, zda se jedná o nový záznam, editaci, kopii apod. Výčtový typ může nabývat hodnoty ***„remAppend“***, což znamená, že se jedná o nový záznam, ***„remAppendCopy“*** vyjadřující nový záznam pomocí kopie, ***„remEdit“***, což znamená, editace záznamu a ***„remDelete“*** vyjadřující smazání záznamu. Hodnot výčtového typu je více. Výše uvedené jsou nejčastěji používané.

Dalším parametrem je výstupní parametr ***„ADeny“*** typu ***„Boolean“***, který slouží k nastavení, zda je povolen přístup k práci se záznamem.

Třetí parametr ***„AReason“*** je typu ***„string“*** a slouží k nastavení textu, který se má zobrazit jako důvod nepovolení přístupu k práci se záznamem.

POZNÁMKA: Metoda je volána před přepnutím editovacího režimu, tzn. záznam je ještě v prohlížení. Nelze se tedy řídit hodnotami ***„NovyDM“***, ***„NastavZmenu“*** apod., relevantní je pouze hodnota parametru ***„AEditMode“***.

POZNÁMKA: Odepření přístupu se provádí výhradně nastavením parametru ***„ADeny“*** na ***„True“*** + (nepovinně) nastavením textu do parametru ***„AReason“***, který by měl obsahovat důvod odmítnutí. Metoda nesmí vyvolat výjimku a nesmí mít žádné závažné vedlejší účinky. Pro rekapitulaci uvádíme shrnutí, co je v metodě ***„AccessDenied“*** povoleno a co ne.

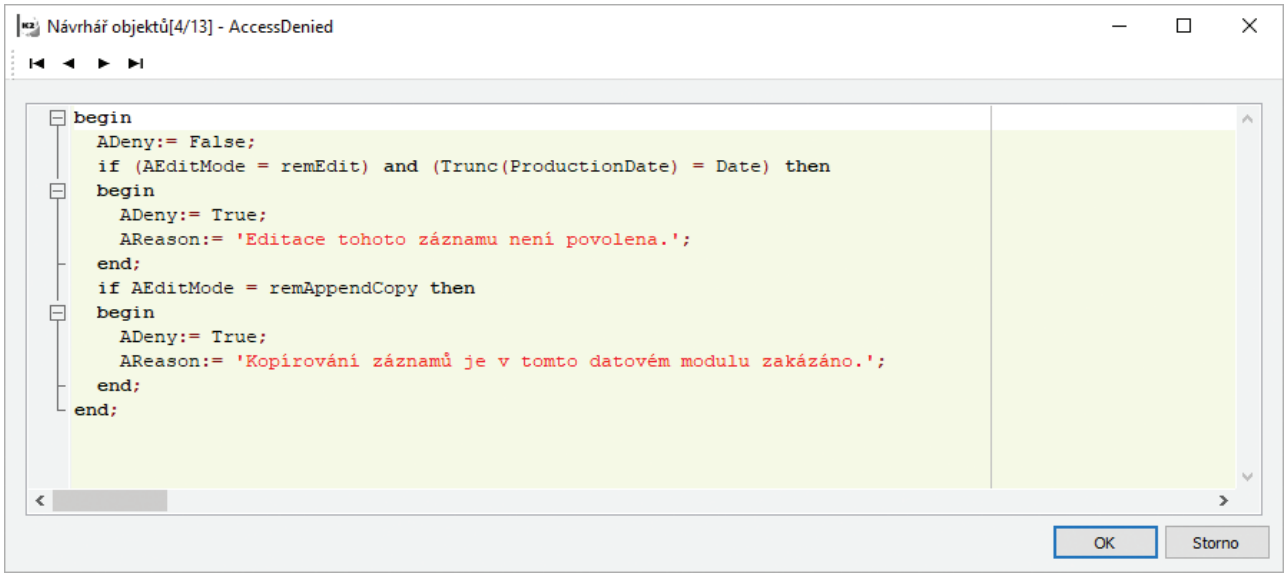
Povoleno je:

- číst hodnoty datových polí
- testování/kontrola práv

Zakázáno je:

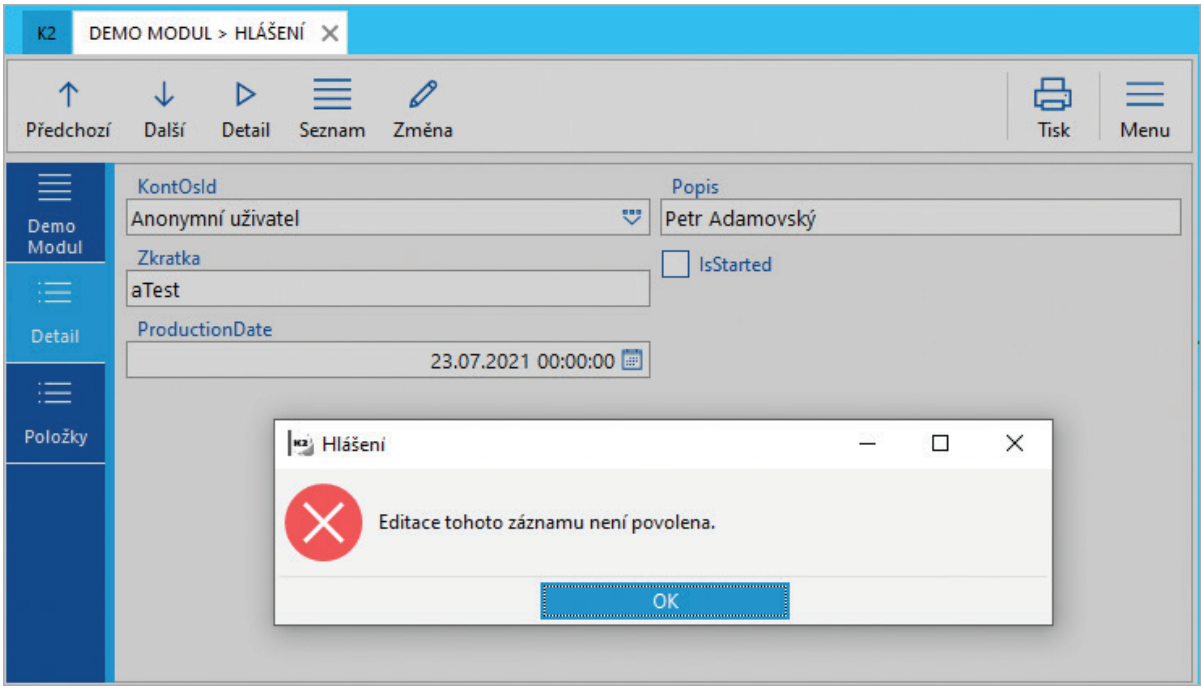
- vyvolávat (záměrně) výjimky
- načítat do *„Self“* jiný záznam
- zapisovat cokoliv do databáze

PŘÍKLAD: Následující příklad demonstruje použití metody *„AccessDenied“*. Implementace říká, v případě, že vstupuji do změny záznamu a zároveň pole *„ProductionDate“* typu *„TDateTime“* se rovná dnešnímu dni, tak zakaž editaci tohoto záznamu. Editace se zakáže pomocí nastavení výstupního parametru procedury *„ADeny“* na hodnotu *„True“*. Do druhého výstupního parametru *„AReason“* je vložen text, který uživatele upozorní na danou situaci. Další kontrola v příkladu je na vkládání nového záznamu pomocí kopie existujícího. V tomto případě je patřičnou podmínkou tento stav zakázán s patřičným hlášením pro uživatele, viz Obrázek 183 - Příklad použití metody "AccessDenied" - zápis kódu.



Obrázek 183 - Příklad použití metody "AccessDenied" - zápis kódu

Ukázka zablokování editace pro záznam s dnešním datem je vidět na Obrázek 184 - Příklad použití metody "AccessDenied" v datovém modulu.



Obrázek 184 - Příklad použití metody "AccessDenied" v datovém modulu

4.8.12. METODA „ENABLEDFIELD“

Procedura ***„EnabledField“*** slouží k řízení stavu jednotlivých polí datového modulu. Umožňuje na základě podmínek nastavit na každém poli, zda je přístupné pro změnu či nikoliv.

Hlavička procedury
procedure EnabledField(ADataField: TDataField; var AState: TTripleState);

Pro tuto metodu se implementuje procedura, která má vstupní parametr ***„ADataField“*** typu ***„TDataField“*** což je instance pole, pro které se provádí kontrola.

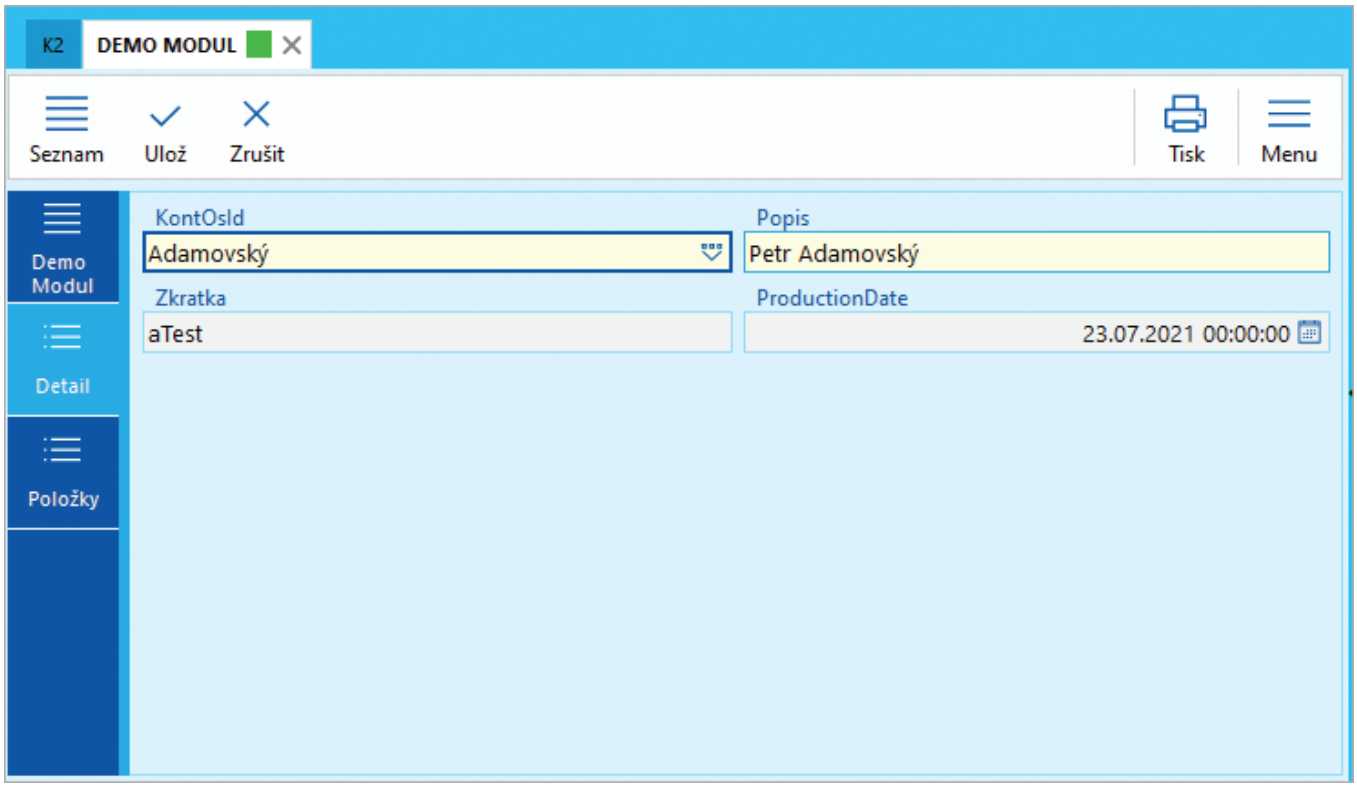
Dalším je výstupní parametr ***„AState“*** výčtového typu ***„TTripleState“***, který slouží k nastavení, zda je povoleno pole měnit či nikoliv. Výčtový typ nabývá hodnot ***„tsNo“*** – ***„je nedostupné“***, „tsYes“ – ***„je dostupné“*** a ***„tsNone“*** – ***„je mi to jedno (hodnota se nastaví na základě jiných informací, např. v jádře K2)“***.

POZNÁMKA: Popis a postup použití metody by měl být následující. Na začátku ověřování v jádře K2 je hodnota ***„AState“*** nastavena na ***„tsNone“***. Při volání metody ***„EnabledField“*** už může být hodnota ***„AState“*** nastavena jiným kódem (například z jádra K2). **Změnu hodnoty *„AState“* provádějte pouze v případě, že skutečně chcete nějakou hodnotu explicitně nastavit**, tzn. povolit nebo zakázat editaci pole. Ve většině případů půjde o zákaz, tzn. nastavení hodnoty ***„AState“*** = ***„tsNo“***.

PŘÍKLAD: Následující příklad demonstruje použití metody ***„EnabledField“***. Implementace říká, v případě, že vstupuji do změny záznamu a zároveň pole ***„ProductionDate“*** typu ***„TDateTime“*** se nerovná dnešnímu dni, tak zakaž editaci polí ***„ProductionDate“*** a ***„Abbr“***. Editace se zakáže pomocí nastavení výstupního parametru procedury ***„AState“*** na hodnotu ***„tsNo“*** v případě, že parametr procedury ***„ADataField“*** odpovídá některému ze zakazovaných polí.

Ukázka zablokování editace pro pole je vidět na Obrázek 185 - Příklad použití metody "EnabledField" v datovém modulu.

```
begin
  if (not IsNew) and (ProductionDate <> Date) then begin
  begin
    case ADataField.FieldNo of
      DemoModul_Abbr, DemoModul_ProductionDate : AState := tsNo;
    end;
  end;
end;
```



Obrázek 185 - Příklad použití metody "EnabledField" v datovém modulu

4.8.13. METODA „RECORDISHIDDEN“

Metoda je vlastně skipovací funkce. Pokud chceme některé záznamy načtené v gridu mít skryté, docílíme tím touto metodou.

Hlavička procedury
Procedure DataMRecordIsHidden(var AIsHidden: TTripleState);

Pro tuto metodu se implementuje procedura, která má výstupní parametr ***„AIsHidden“*** typu TTripleState. Parametr ***„AIsHiddem“*** může nabývat těchto stavů:

tsNone = ponechát hodnotu definovanou v knihovně.

tsYes = záznam je skrytý.

tsNo = záznam není skrytý.

4.8.14. METODA „LOADCHILDDATA“ PŘEDKA CHILDDATAM

Metoda je určena primárně pro načtení položek, které jsou paměťového charakteru. Tedy jejich předkem je ***„TMemFile“*** a neexistuje pro ně v zobrazované podobě databázová tabulka.

Hlavička procedury
procedure LoadChildData(AChildFile: TxFile);

Pro tuto metodu se implementuje procedura, která má vstupní parametr ***„AChildFile“*** typu ***„TxFile“*** což je instance položkového modulu. Skrz tento objekt můžeme položky vkládat, mazat apod.

POZNÁMKA: Aby se metoda v datovém modulu zavolala, je nutné nastavit položky pouze pro čtení pomocí příznaku ***„Pouze pro čtení, data“***.

POZNÁMKA: Nedoporučujeme používat metodu pro spojování dat paměťových v kombinaci s klasickými položkami. Za určitých okolností nemusí některé funkce fungovat. Například filtrování, třídění apod.

Příklad, ve kterém je využita tato metoda je k dispozici v kapitole o zděděných property, u popisu datového modulu, který je paměťový a jeho předkem je ***„TOneRecordDM“***.

4.8.15. METODA „INITCHILDDATA“ PŘEDKA CHILDDATAM

Metoda je určena pro přidání/změnu/výmaz záznamů ve vlastněných položkách bezprostředně po uvedení hlavičkového záznamu do změny. Je určena primárně pro položky, které jsou paměťového charakteru. Tedy jejich předkem je ***„TMemFile“*** a neexistuje pro ně v zobrazované podobě databázová tabulka.

Hlavička procedury
procedure InitChildData(AChildFile: TxFile);

Pro tuto metodu se implementuje procedura, která má vstupní parametr ***„AChildFile“*** typu ***„TxFile“*** což je instance položkového modulu. Skrz tento objekt můžeme položky vkládat, mazat apod.

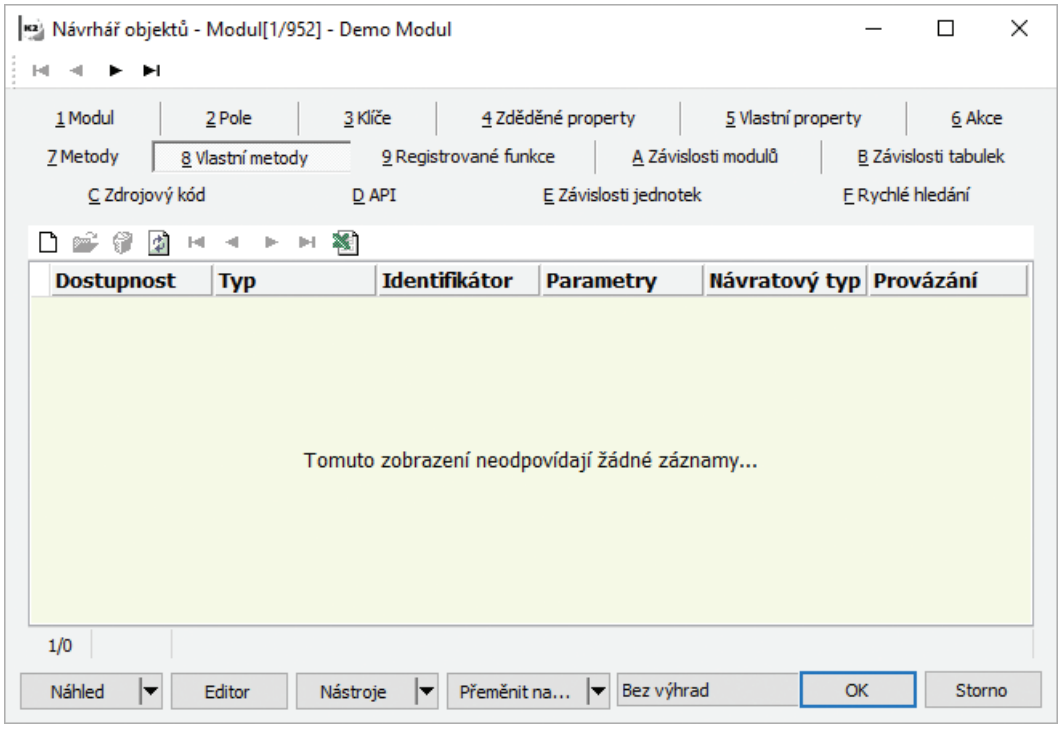
POZNÁMKA: Nedoporučujeme používat metodu pro spojování dat paměťových v kombinaci s klasickými položkami. Za určitých okolností nemusí některé funkce fungovat. Například filtrování, třídění apod.

4.9. VLASTNÍ METODY

V rámci datového modulu je možné implementovat vlastní metody, které je možné využit například ke zdrojovému kódu, který se často opakuje. Místo duplikování kódu jednoduše vytvoříme vlastní metodu, kterou voláme z potřebných míst hlavního programu.

4.9.1. DEFINICE METODY

Seznam vlastních metod datového modulu nalezneme na záložce ***„Vlastní metody“***. Novou metodu do seznamu založíme pomocí stisknutím tlačítka Nový v nástrojové liště nad seznamem nebo stisknutím klávesy Insert, viz Obrázek 186 - Seznam vlastních metod.



Obrázek 186 - Seznam vlastních metod

Zobrazí se formulář pro založení nové metody. Vstupní parametry jsou následující.

Dostupnost
Definuje viditelnost metody v rámci instancí datových modulů. Typy, které jsou zde k dispozici, vycházejí z definice zapouzdření v objektově orientovaného programování. Tedy vycházejí ze standardu OOP. K dispozici jsou následující možnosti.

public – vlastnost je viditelná všude

private – vlastnost je viditelná pouze v rámci datového modulu

protected – vlastnost je viditelná pouze v rámci datového modulu a v jeho potomcích

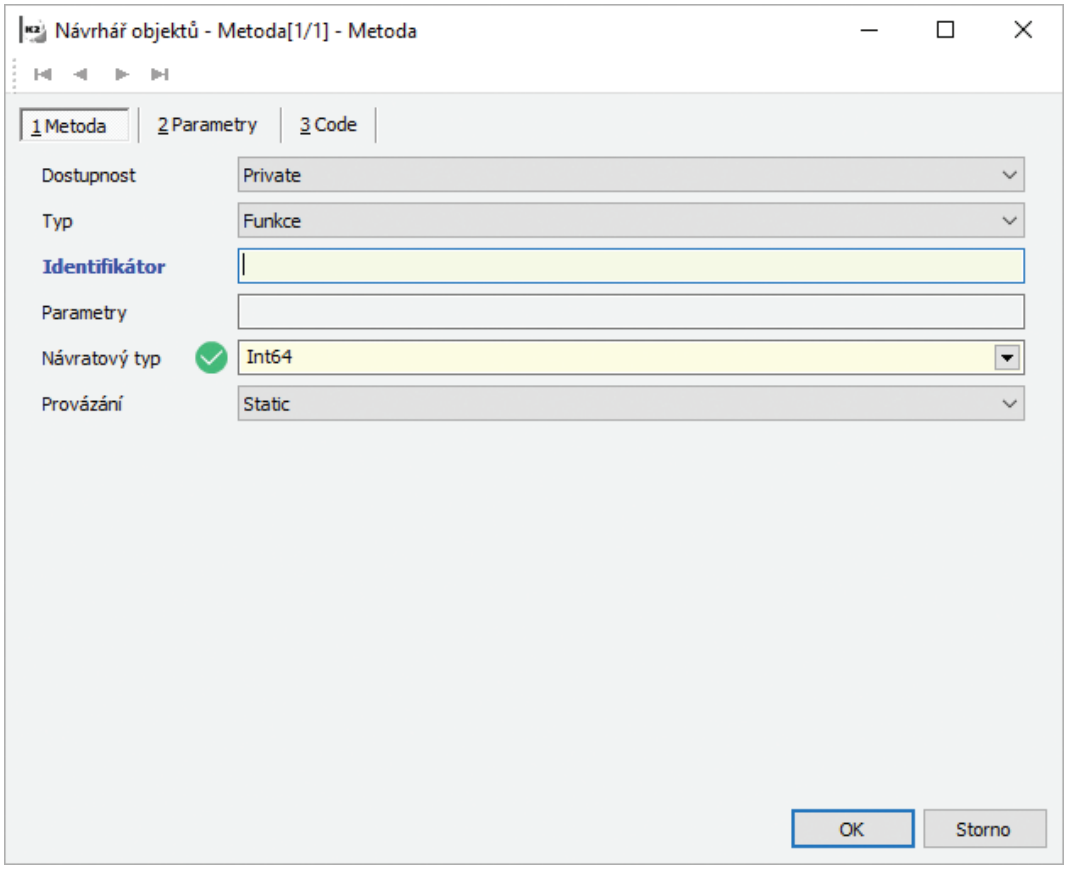
published – vlastnost je viditelná všude, navíc je viditelná ve formulářích K2

Typ
Slouží k určení, zda se jedná o funkci nebo proceduru. Funkce je definována jako metoda, která má vstupy a návratovou hodnotu, která je předem definovaného typu. Procedura je metoda, která má pouze vstupy bez návratové hodnoty.

Identifikátor
Jednoznačná identifikace metody v rámci datového modulu. Využívá se při volání v rámci zdrojového kódu.

Parametry
Seznam definovaných vstupních parametrů metody. Parametry se definují na záložce „Parametry".

Návratový typ
Tato vlastnost se objeví tehdy, pokud zvolíme typ metody ***„Funkce"***. Je zde možnost si vybrat ze seznamu známých typů, v takovém případě se v editoru objeví zelené kolečko s ***„fajfkou"*** viz Obrázek Obrázek 187 - Definice návratového typu u funkce. V opačném případě tam není nic, což znamená, že se může jednat o neznámý typ, ale nedá se říct, že to nebude přeložitelné. Ty nejpoužívanější typy jsou ale v nabídce, takže se spíše potkáme s ***„fajfkou"***.



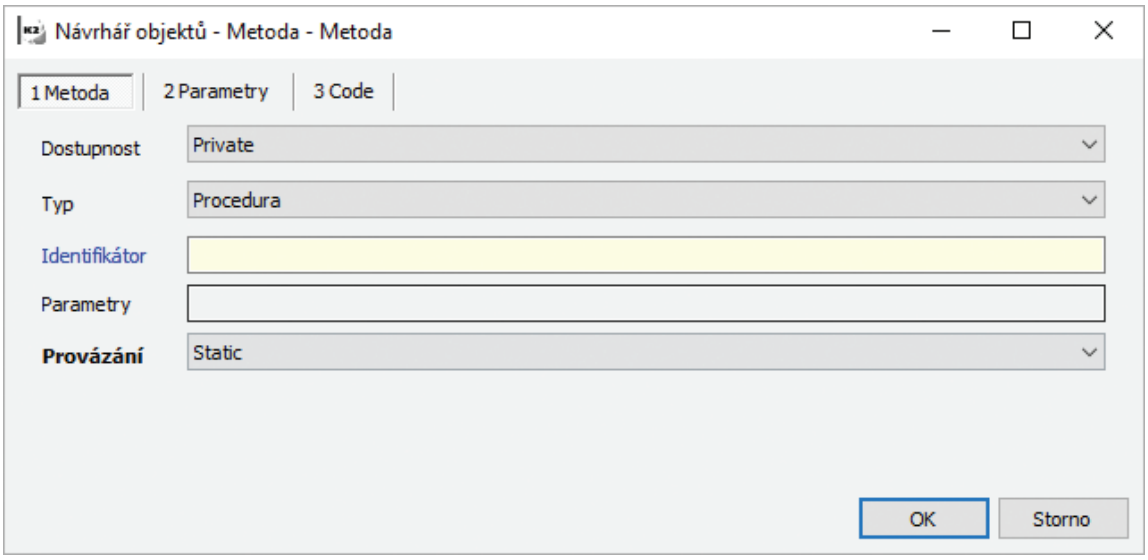
Obrázek 187 - Definice návratového typu u funkce

Provázání
Slouží k určení provázání metody s datovým modulem.

Static – třídní metoda, kterou lze volat bez instance datového modulu

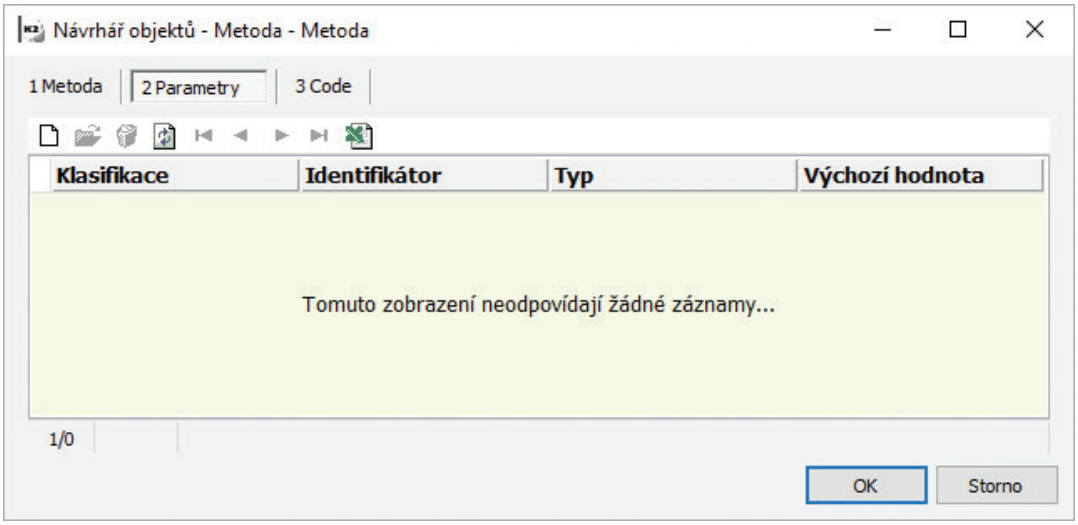
Virtual – instanční metoda, kterou lze v potomcích ***„překrýt"*** (implementovat novým zdrojovým kódem)

PTrepeat – třídní metoda, která získává instanci datového modulu v podobě vstupního parametru



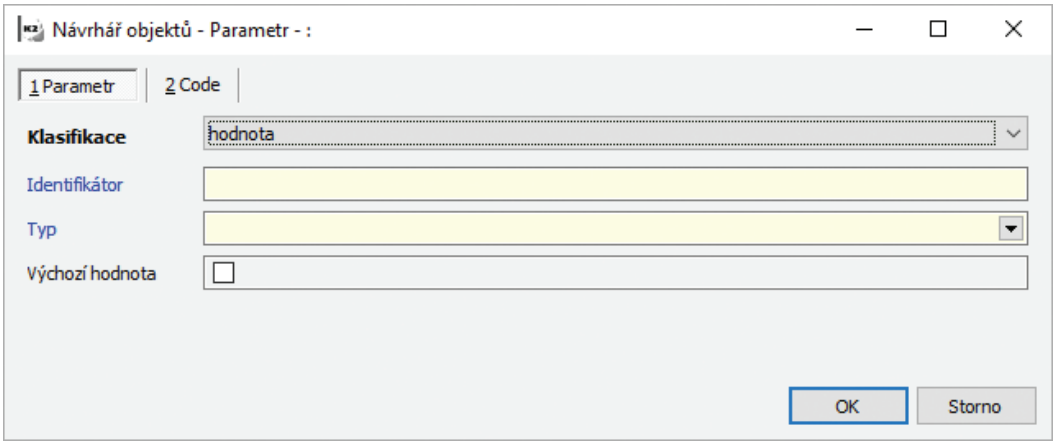
Obrázek 188 - Definice vlastní metody

Na druhé záložce ***„Parametry"*** je k dispozici seznam všech parametrů metody. Nový parametr vložíme stisknutím tlačítka Nový v nástrojové liště nad seznamem nebo stisknutím klávesy Insert, viz Obrázek Obrázek 187 - Definice návratového typu u funkce.



Obrázek 189 - Seznam parametrů vlastní metody

U každého parametru je nutné definovat jeho vlastnosti, viz Obrázek 190 - Definice parametru vlastní metody.



Obrázek 190 - Definice parametru vlastní metody

Klasifikace
Pomocí klasifikace definujeme chování parametru. Dostupní jsou následující varianty.

hodnota – jednoduchý parametr, nejčastější varianta

proměnná [var] – jedná se o tzv. vstupní parametr. V případě, že tento parametr modifikujeme uvnitř metody, je změna promítnuta i v proměnné, ze které byla hodnot předána metodě při jejím volání.

konstantní [const] – jedná se o parametr typu konstanta

výstupní [out] – jedná se o tzv. výstupní parametr. V případě volání metody, předáváme parametr, u kterého očekáváme, že jeho hodnota bude nastavena uvnitř metody a je tedy jejím výstupním parametrem. Výsledek máme uloženy po volání metody v parametru, který jsme předali do metody při jejím volání. Klasické využití této varianty je v případě funkcí, kdy potřebujeme, aby bylo možné z funkce získat více návratových hodnot.

Identifikátor
Název parametru, který je používán ve zdrojovém kódu při jeho používání.

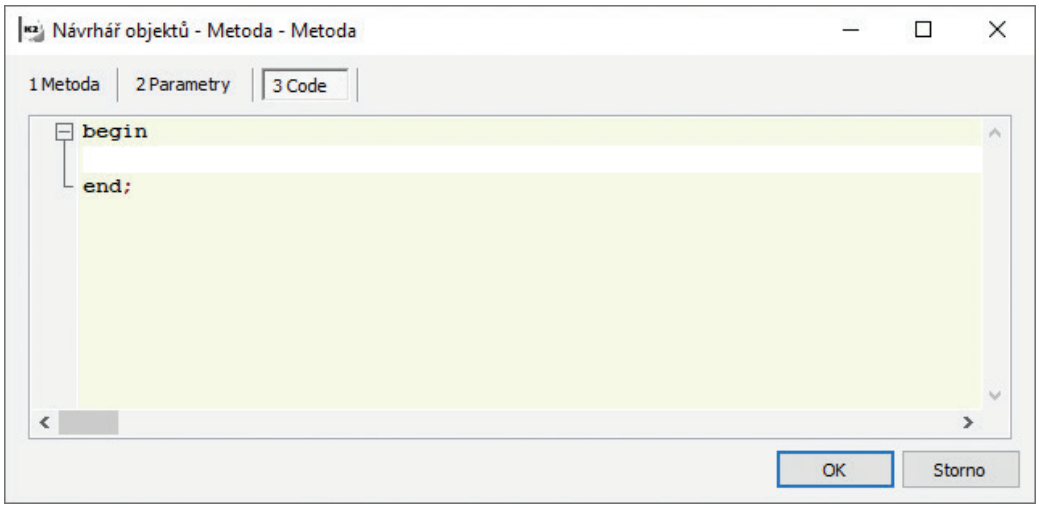
Typ
Definuje datový typ parametru. Opět zde máme možnost si vybrat ze seznamu typů jako tomu bylo u návratového typu metody typu ***„Funkce“***.

Výchozí hodnota
Parametr může být definován jako tzv. výchozí. Znamená to, že se stává nepovinným při volání metody. Při této volbě je nutné nastavit výchozí hodnotu, která je použitá v případě, že nebude při volání metody parametr zadán.

4.9.2. IMPLEMENTACE METODY

Poslední částí při tvorbě metody, je její samotná implementace. Na záložce ***„Code“*** je k dispozici vstup pro implementaci, viz Obrázek Obrázek 191 - Zdrojový kód vlastní metody.

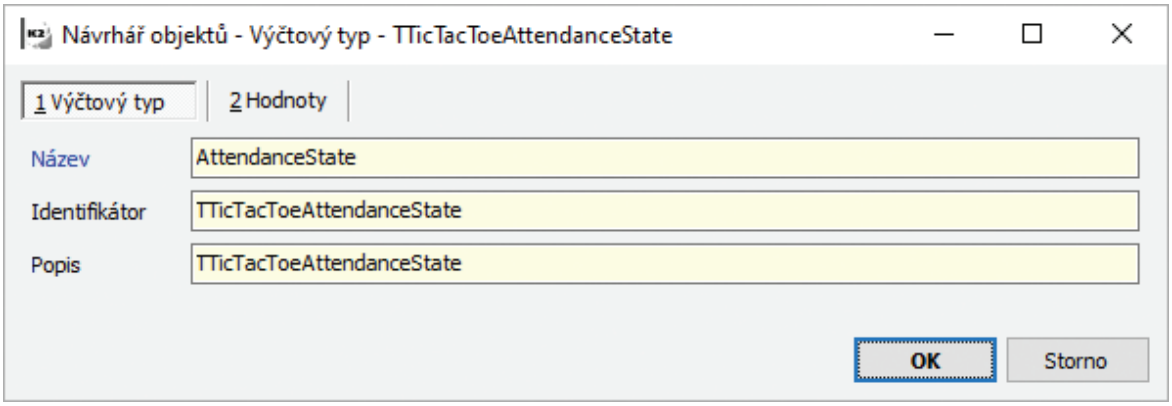
POZNÁMKA: K implementaci těla metody můžeme využít editor skriptu, který spustíme nad daným datovým modulem. Tato varianta je doporučená, protože umožnuje kontrolu správné syntaxe skriptu formou jeho překladu, tak jak je zvykem při implementaci skriptu.



Obrázek 191 - Zdrojový kód vlastní metody

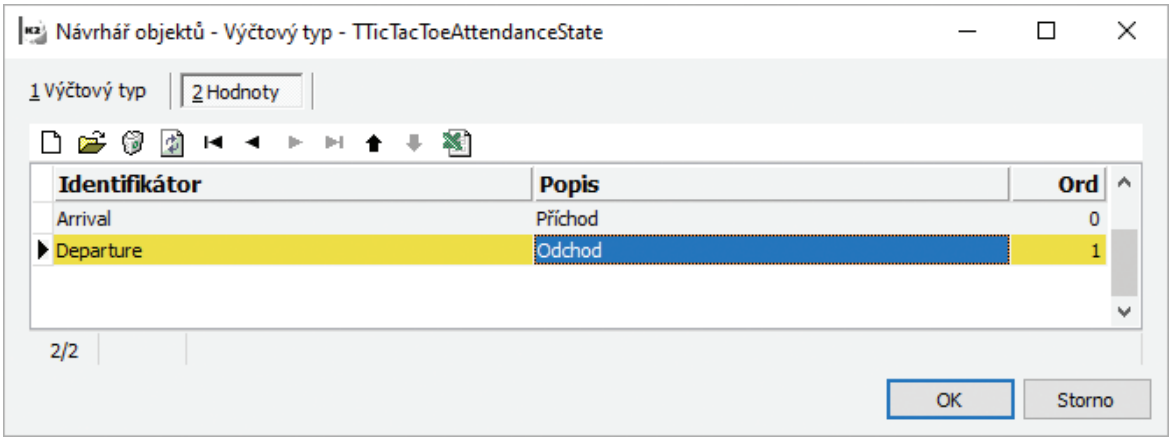
PŘÍKLAD: Použití vlastních metod si ukážeme na příkladu, který nazveme „jednoduchá docházka“. Vytvoříme datový modul, ve kterém budeme evidovat příchody a odchody zaměstnanců. Vždy zapíšeme číslo zaměstnance (kontaktní osoba), datum a čas a také informaci, zda se jedná o příchod nebo odchod. K evidenci číselníku příchod/odchod využijeme výčtového typu. K zápisu příchodu a odchodu využijeme příkazů, které budou po spuštění volat vlastní metodu, která bude implementovat založení záznamu s aktuálně přihlášeným uživatelem v K2 a stavem, zda se jedná o příchod či odchod.

Úplně na začátek si založíme výčtový typ, který bude evidovat stav. Nazveme ho ***„AttendanceState“***, viz Obrázek 192 - Příklad - vlastní metody - výčtový typ.



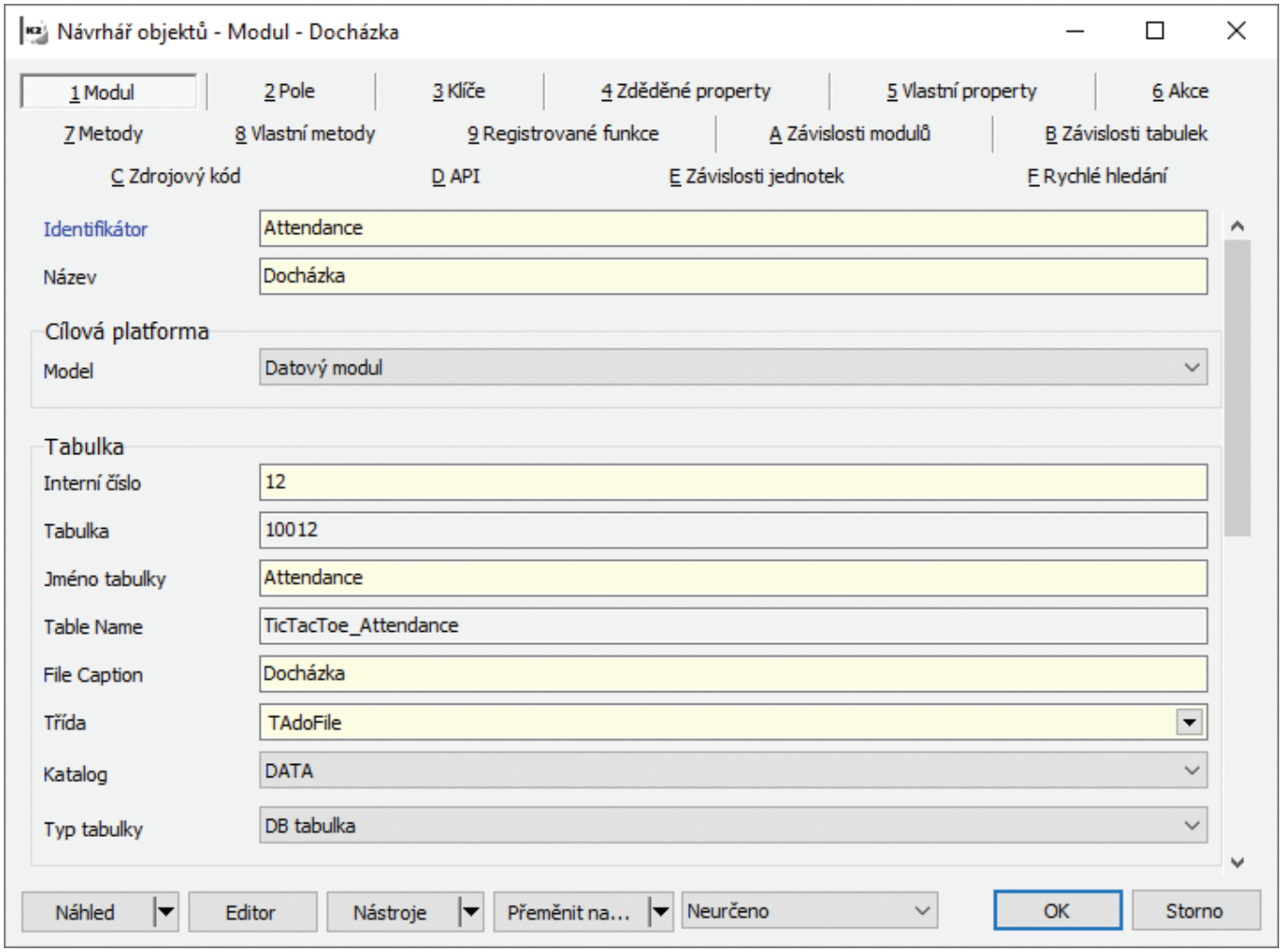
Obrázek 192 - Příklad - vlastní metody - výčtový typ

Hodnoty založíme dvě. ***„Arrival“*** – ***„Příchod“*** a „Departure“ – ***„Odchod“***, viz Obrázek 193 - Příklad - vlastní metody - výčtový typ - hodnoty.



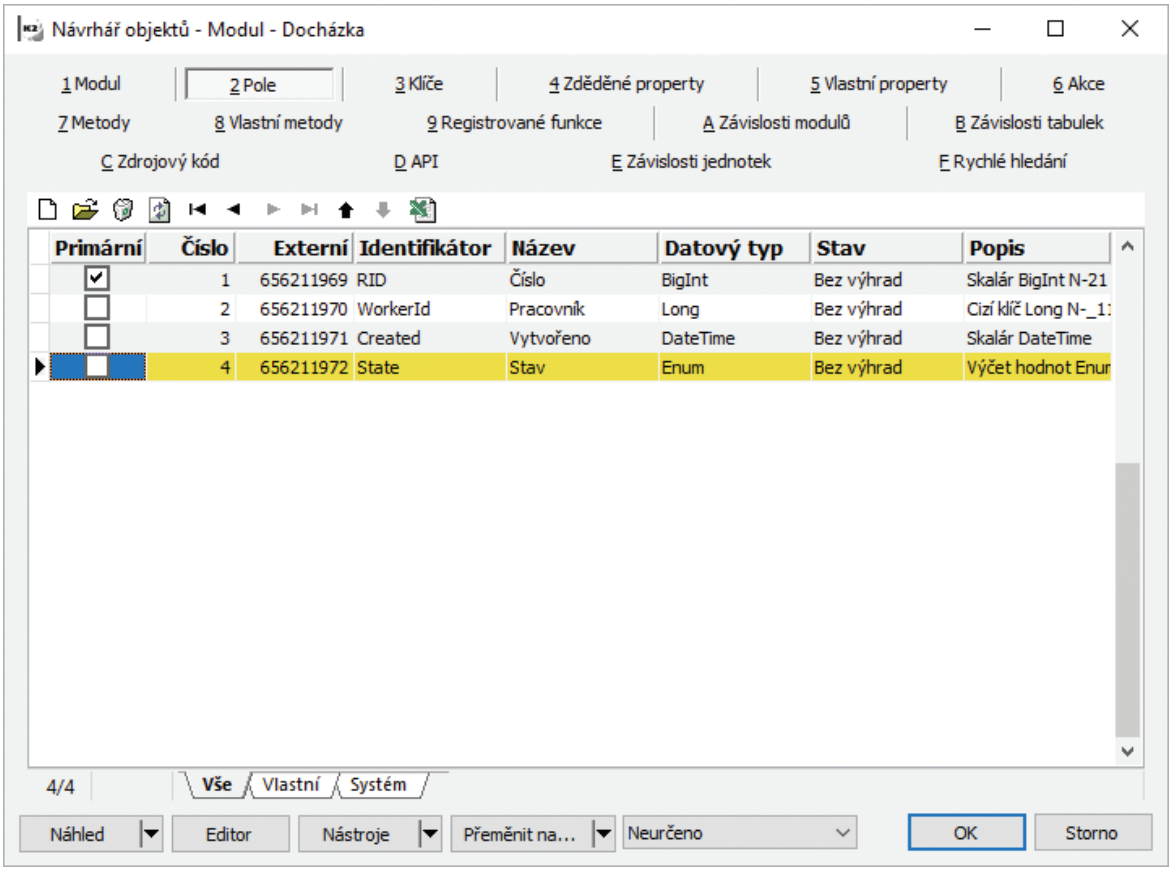
Obrázek 193 - Příklad - vlastní metody - výčtový typ - hodnoty

Založíme nový datový modul ***„Attendance“*** – ***„Docházka“***, který bude potomkem ***„TBaseDataM“***, viz Obrázek 194 - Příklad - Vlastní metody – datový modul.



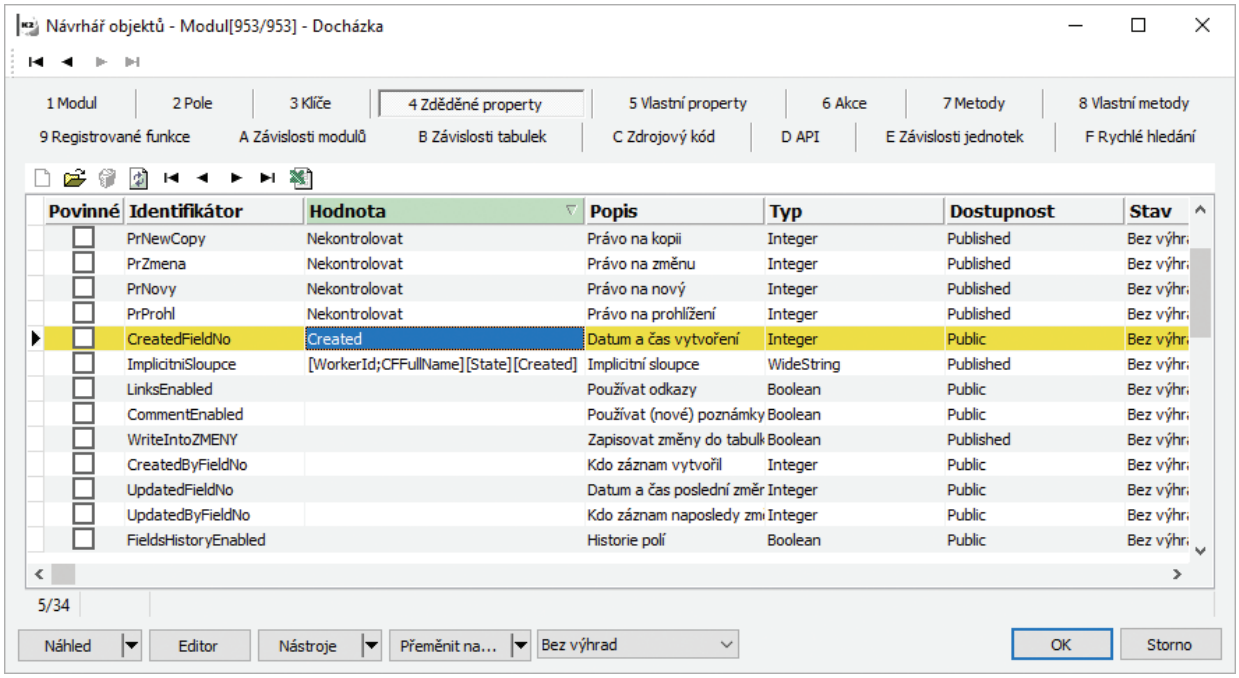
Obrázek 194 - Příklad - Vlastní metody – datový modul

Vytvoříme pole pro primární klíč – RID, dále cizí klíč ***„WorkerId“*** – ***„Pracovník“***, který bude odkazovat do modulu ***„Kontaktní osoby“***, dále pole ***„State“***, které bude typu ***„výčet hodnot“***, který jsme definovali v předchozích krocích – ***„AttendanceState“***. Na závěr pole ***„Created“***, které bude datumová informace o příchodu a odchodu zaměstnance, viz Obrázek 195 - Příklad - vlastní metody - pole v DM.



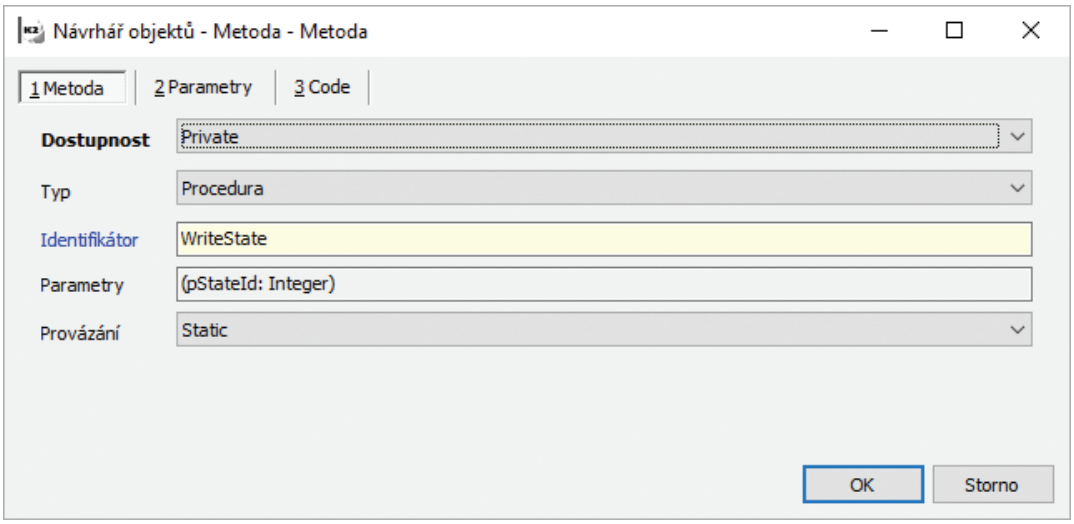
Obrázek 195 - Příklad - vlastní metody - pole v DM

U pole ***„Create“*** využijeme možnosti nastavení data při založení záznamu, tedy nastavíme pole do vlastních property do ***„CreatedFieldNo“***, viz Obrázek 196 - Příklad - vlastní metody - zděděné property. Nebudeme tedy muset tuto informaci nastavovat, ale K2 ji provede za nás.



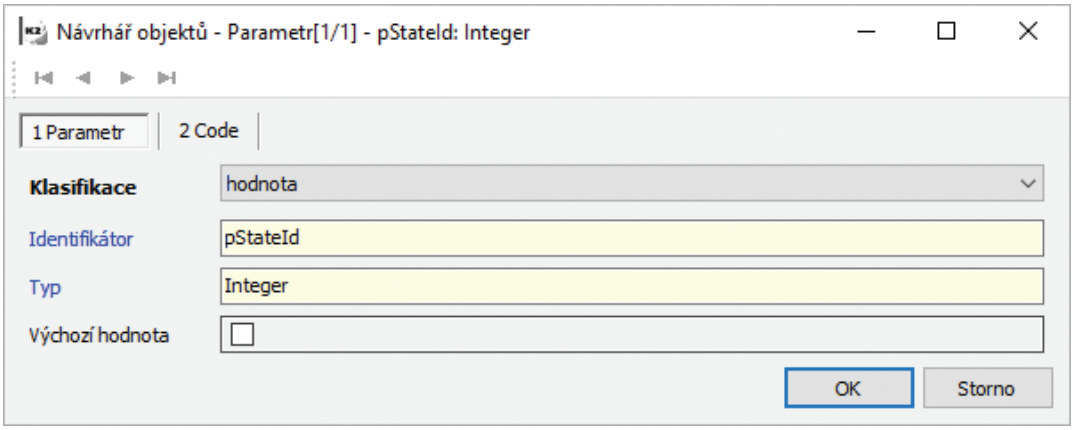
Obrázek 196 - Příklad - vlastní metody - zděděné property

Na záložce ***„Metody“*** založíme novou metodu, která bude sloužit k zakládání nového záznamu do modulu docházka. Nazveme ji ***„WriteState“***, dostupnost ponecháme ***„private“***, typ jako ***„Procedura“***, provázání ***„Static“***, viz Obrázek 197 - Příklad - vlastní metody – metoda.



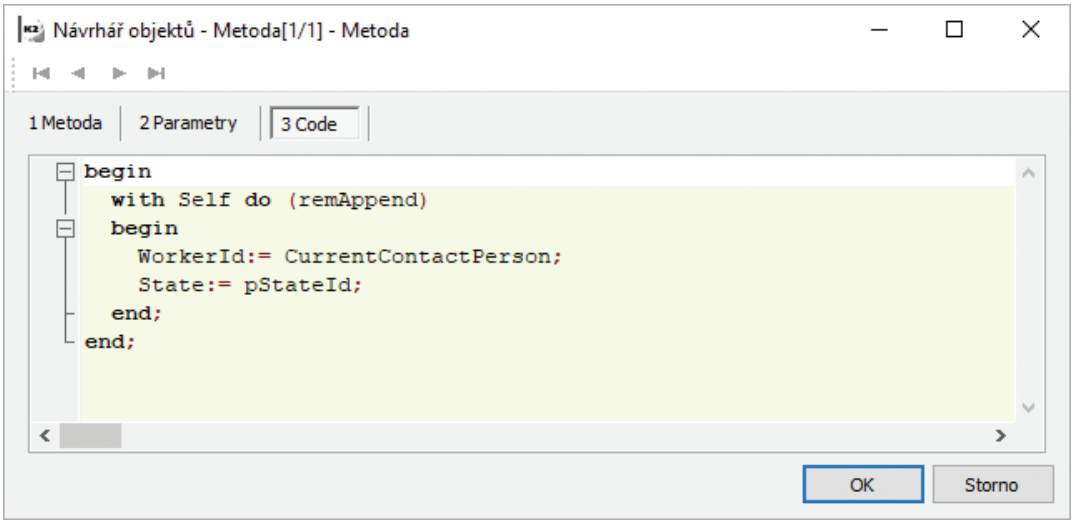
Obrázek 197 - Příklad - vlastní metody – metoda

Na záložce ***„Parametry“*** vytvoříme parametr ***„pStateId“***, který bude typu ***„integer“*** a klasifikován jako ***„hodnota“***. Tímto parametrem předáme do procedury informaci o stavu – ***„příchod/odchod“***, viz Obrázek 198 - Příklad - vlastní metody - metoda- parametry.



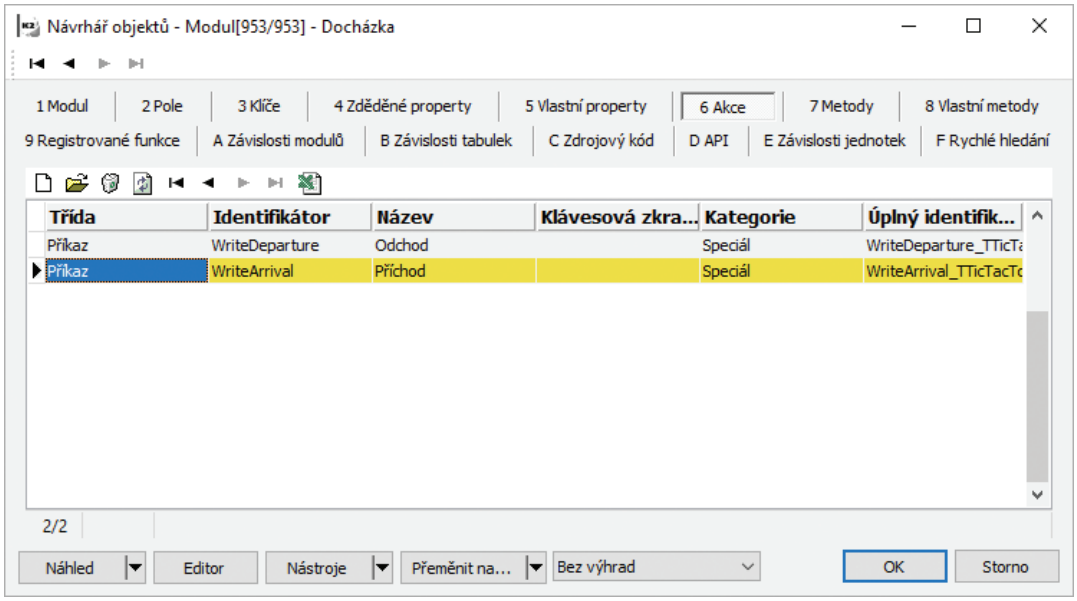
Obrázek 198 - Příklad - vlastní metody - metoda- parametry

Nakonec je potřeba provést samotnou implementaci. Doporučujeme využít editoru skriptu, kde je možné zkontrolovat správnost syntaxe implementace. Pro založení záznamu do modulu postačí jednoduchá konstrukce ***„with“***, kde nastavíme pole ***„WorkerId“*** na hodnotu ***„CurrentContactPerson“*** – aktuálně přihlášená kontaktní osoba, a pole ***„State“*** na hodnotu parametru procedury ***„pStateId“***, kterou předáme při volání procedury, viz Obrázek 199 - Příklad - vlastní metody – implementace.



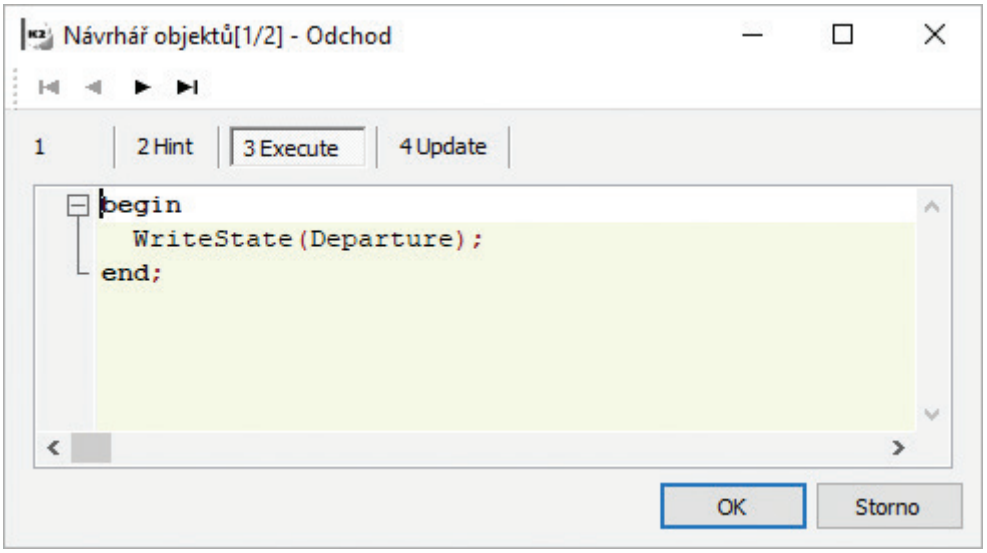
Obrázek 199 - Příklad - vlastní metody – implementace

Abychom mohli zakládat záznamy pomocí nové vlastní metody, je potřeba ji zpřístupnit pomocí příkazů. Založíme tedy na záložce ***„Akce“*** datového modulu dva příkazy. První z nich se bude jmenovat ***„WriteDeparture“*** – ***„Odchod“*** a druhý pak ***„WriteArrival“*** – ***„Příchod“***, viz Obrázek 200 - Příklad - vlastní metody – příkazy.



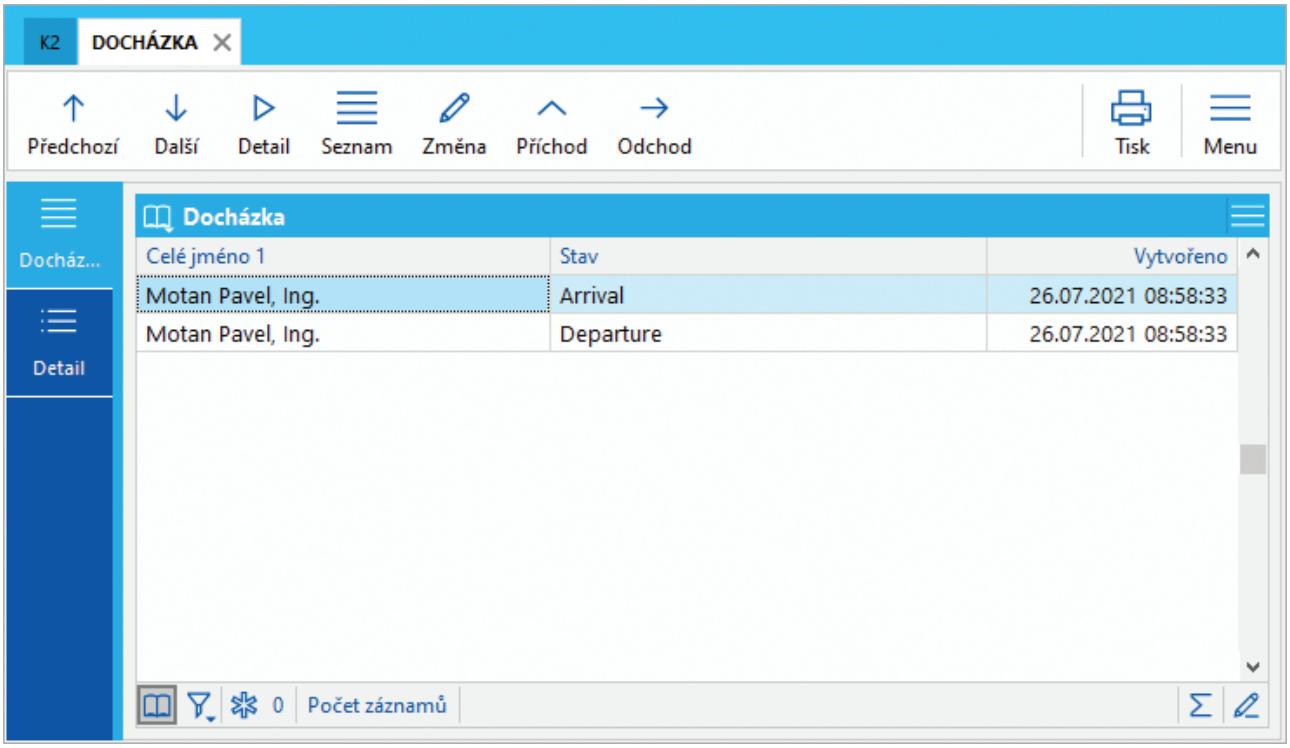
Obrázek 200 - Příklad - vlastní metody – příkazy

V implementace jednotlivých příkazů postačí zavolat vlastní metodu – ***„WriteState“*** vytvořenou v předchozím kroku s předáním patřičného parametru. Zde využijeme hodnot výčtového typu ***„Departure“*** nebo ***„Arrival“*** dle typu příkazu, viz Obrázek 201 - Příklad - vlastní metody - Implementace příkazu.



Obrázek 201 - Příklad - vlastní metody - Implementace příkazu

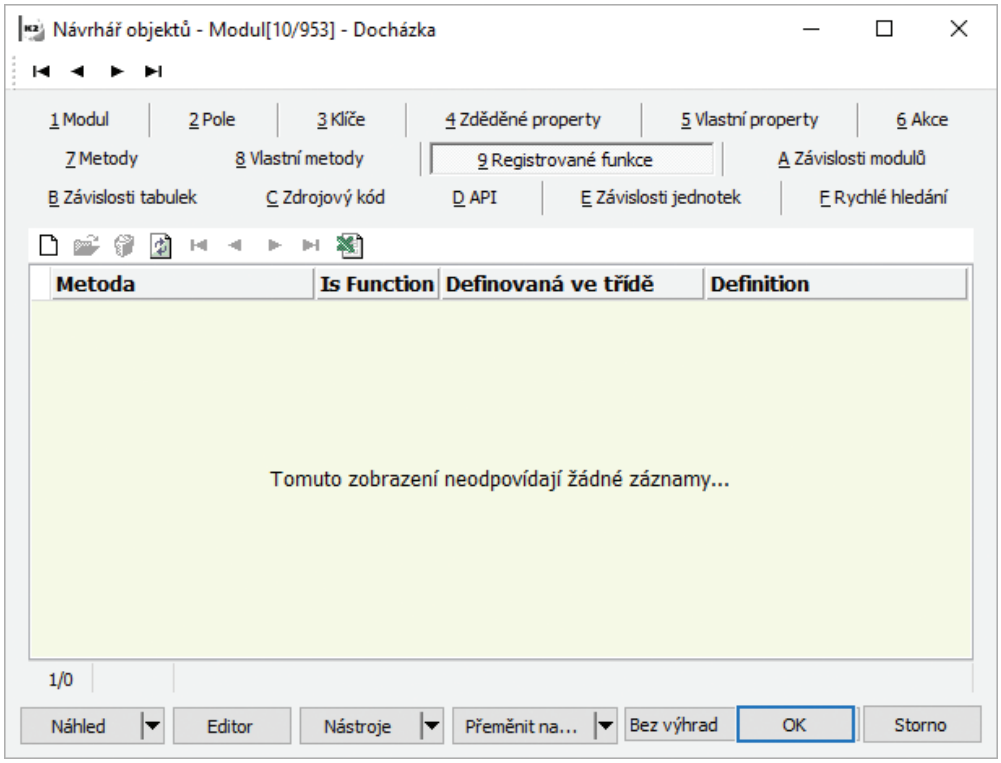
Po operaci ***„deploy"*** a úpravě formuláře, do kterého si přidáme vytvořené příkazy do menu, může vypadat formulář následovně, viz Obrázek 202 - Příklad - vlastní metody - výsledný formulář.



Obrázek 202 - Příklad - vlastní metody - výsledný formulář

4.10. REGISTROVANÉ FUNKCE

V každém datovém modulu v návrháři objektů je možné implementovat registrované funkce, které jsou dostupné pro daný datový modul. Seznam implementovaných registrovaných funkcí je vidět na záložce ***„Registrované funkce"***, který je vidět na Obrázek 203 - Seznam registrovaných funkcí datového modulu.



Obrázek 203 - Seznam registrovaných funkcí datového modulu

Novou registrovanou funkci přidáme pomocí stisknutí tlačítka Nový nebo pomocí klávesy Insert. Zobrazí se nám formulář, který je vidět na Obrázek 204 - Seznam registrovaných funkcí k implementaci v datovém modulu. Je zde vidět všechny registrované funkce, které lze v návrháři objektů implementovat pro daný datový modul.

POZNÁMKA: Registrované funkce lze mít i ve dvou a více různých souborech rozšíření (zavolají se všechny, v pořadí, které jim přidělí uživatel), nicméně se doporučuje mít registrované funkce pouze v jednom souboru rozšíření.

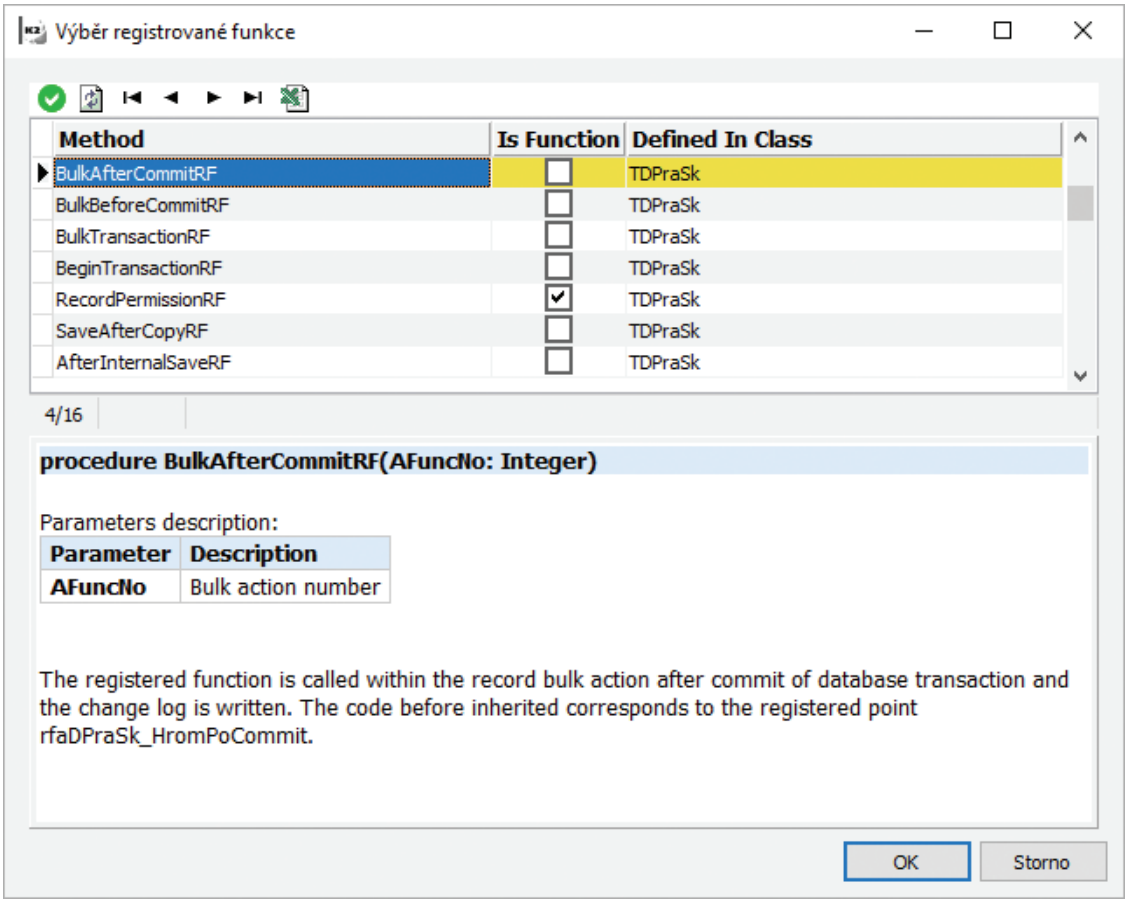
4.10.1. DOSTUPNÉ REGISTROVANÉ FUNKCE

V seznamu jsou u každé registrované funkce dostupné informace, které se dělí na následující výčet.

Method
Název metody pro registrovanou funkci.

Is Function
Příznak, zda se jedná o proceduru či funkci.

Defined In Class
Udává v jaké třídě je funkce definována.



Obrázek 204 - Seznam registrovaných funkcí k implementaci v datovém modulu

4.10.1.1. Mapování registrovaných bodů na registrované funkce

V následující tabulce jsou k dispozici registrované body definované v K2 a jejich registrované funkce, které jsou k dispozici v návrháři objektů.

REGISTROVANÝ BOD	REGISTROVANÁ FUNKCE
rfaDM_ControlValueChanged	TDataM.ControlValueChangedRF
rfaDM_AfterControlValueChanged	po inherited TDataM.ControlValueChangedRF
rfaDM_AfterEndEdit	TDataM.AfterEndEditRF
rfaDM_CanChangeControlValue	TDataM.CanChangeControlValueRF
rfaDM_BeforeExportExcel	TDataM.BeforeExportRF(etExcel)
rfaDM_BeforeExportCSV	TDataM.BeforeExportRF(etCSV)
rfaDPraSk_UlozPredTran	TDPrask.BeginTransactionRF
rfaDPraSk_UlozPoTran	po inherited TDPrask.BeginTransactionRF
rfaDPraSk_UlozPredCommit	TDPrask.BeforeCommitRF
rfaDPraSk_UlozPoCommit	TDPrask.AfterCommitRF
rfaDPraSk_HromPredTran	TDPrask.BulkTransactionRF
rfaDPraSk_HromPoTran	Po inherited TDPrask.BulkTransactionRF

rfaDPraSk_HromPredCommit	TDPrask.BulkBeforeCommitRF
rfaDPraSk_HromPoCommit	TDPrask.BulkAfterCommitRF
rfaDPraSk_InternalSave	TDPrask.AfterInternalSaveRF
rfaDPraSk_SaveAfterCopy	SaveAfterCopyRF
rfaDPraSk_RecordPermission	TDPrask.RecordPermissionRF
rfaDPraSk_InitEditedRecord	po inherited TDataM.InitEditedRecordRF
rfaDPraSk_BeforeInitEditedRecord	TDataM.InitEditedRecordRF

4.10.1.2. Registrovaná funkce „TDataM.ControlValueChangedRF“

Registrovaná funkce volaná po změně hodnoty pole. V *„inherited“* se volá registrovaný bod *„rfaDM_ControlValueChanged“* a také kód datového modulu reagující na změnu pole. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDM_ControlValueChanged“*. Kód po *„inherited“* odpovídá registrovanému bodu *„rfaDM_AfterControlValueChanged“*.

4.10.1.3. Registrovaná funkce „TDataM.CanChangeControlValueRF“

Registrovaná funkce volaná před změnou hodnoty pole. Změna se dá odmítnout vyvoláním výjimky. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDM_CanChangeControlValue“*.

4.10.1.4. Registrovaná funkce „TDataM.AfterEndEditRF“

Registrovaná funkce se volá na konci metody *„EndEdit“*. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDM_AfterEndEdit“*.

4.10.1.5. Registrovaná funkce „TDataM.InitEditedRecordRF“

Registrovaná funkce slouží k inicializaci záznamu při přechodu do změny. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDPraSk_InitEditedRecord“*.

4.10.1.6. Registrovaná funkce „TDataM.BeforeExportRF“

Registrovaná funkce se volá před provedením exportu a umožňuje tento export zakázat vyvoláním výjimky. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDM_BeforeExportExcel“*, nebo *„rfaDM_BeforeExportCSV“*.

4.10.1.7. Registrovaná funkce „TDPrask.AfterCommitRF“

Registrovaná funkce se volá po uložení záznamu a zápisu do změn. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDPraSk_UlozPoCommit“*.

4.10.1.8. Registrovaná funkce „TDPrask.BeforeCommitRF“

Registrovaná funkce se volá před potvrzením databázové transakce v rámci ukládání záznamu. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDPraSk_UlozPredCommit“*.

4.10.1.9. Registrovaná funkce „TDPrask.BulkAfterCommitRF“

Registrovaná funkce se volá v rámci hromadné akce nad záznamem po potvrzení databázové transakce a zápisu do změn. Kód před *„inherited“* odpovídá registrovanému bodu *„rfaDPraSk_HromPoCommit“*.

4.10.1.10. Registrovaná funkce „TDPrask.BulkBeforeCommitRF“

Registrovaná funkce se volá v rámci hromadné akce na záznamu před voláním potvrzení databázové transakce. Kód před ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_HromPredCommit“***.

4.10.1.11. Registrovaná funkce „TDPrask.BulkTransactionRF“

Registrovaná funkce se volá v rámci hromadné akce na záznamu před zahájením transkce. Kód před ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_HromPredTran“***. Kód po ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_HromPoTran“***.

4.10.1.12. Registrovaná funkce „TDPrask.BeginTransactionRF“

Registrovaná funkce se volá v rámci ukládání záznamu a zahajuje databázovou transakci. Kód před ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_UlozPredTran“***. Kód po ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_UlozPoTran“***.

4.10.1.13. Registrovaná funkce „TDPrask.RecordPermissionRF“

Registrovaná funkce se volá v rámci zjišťování práva na záznam.

4.10.1.14. Registrovaná funkce „TDPrask.SaveAfterCopyRF“

Registrovaná funkce se volá v rámci ukládání záznamu, před samotnou databázovou operací ***„Add/Put/Delete“***. V kódu ***„inherited“*** se volá metoda ***„SaveAfterCopy“***. Kód po ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_SaveAfterCopy“***.

4.10.1.15. Registrovaná funkce „TDPrask.AfterInternalSaveRF“

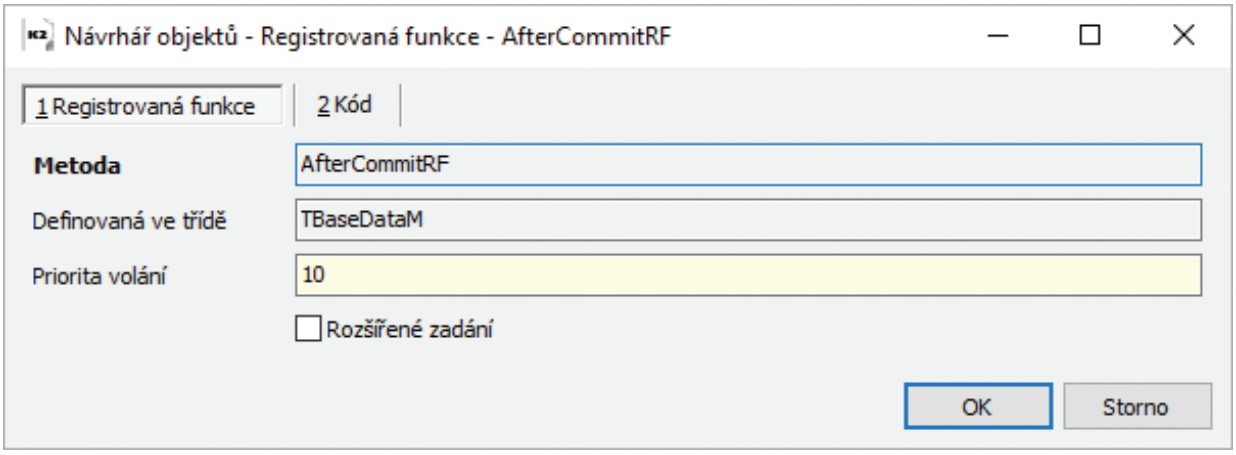
Registrovaná funkce se volá v rámci ukládání záznamu, po úspěšné databázové operaci. Kód před ***„inherited“*** odpovídá registrovanému bodu ***„rfaDPraSk_InternalSave“***.

POZNÁMKA: V návrháři objektů je při výběru registrované funkce popsán i anglický popis k dané funkci, jak je vidět na Obrázek 204 - Seznam registrovaných funkcí k implementaci v datovém modulu.

4.10.2. POUŽITÍ REGISTROVANÝCH FUNKCÍ

Při výběru některé z dostupných registrovaných se zobrazí formulář k její samotné implementaci, viz Obrázek 205 - Detail registrované funkce v datovém modulu.

Na formuláři je vidět název metody, ve které třídě je definována. Tyto údaje jsou pouze informativní. Další volby jsou již ovlivnitelné uživatelem. Jedná se o následující parametry.



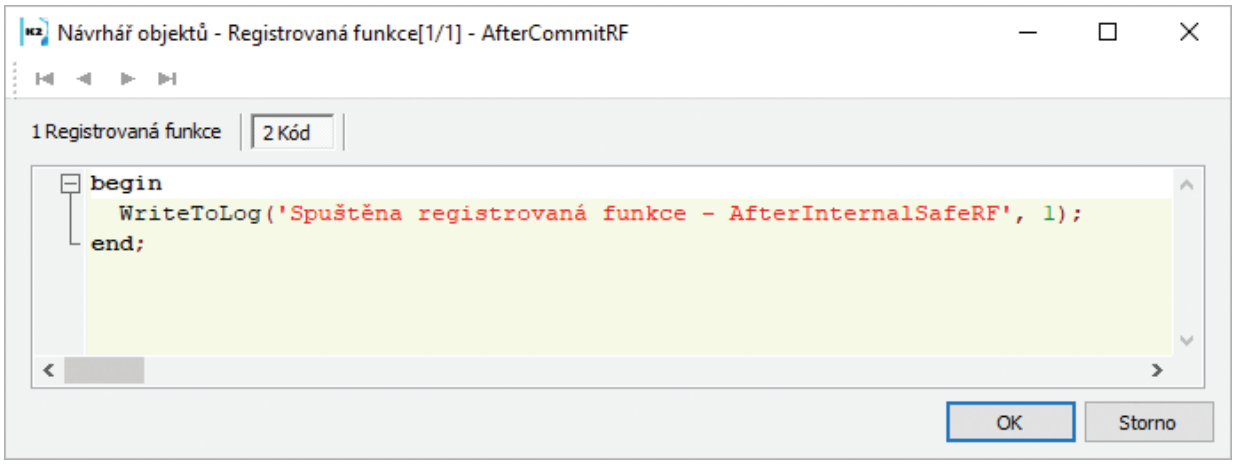
Obrázek 205 - Detail registrované funkce v datovém modulu

Priorita volání
Udává hodnotu priority volání v případě, že by existovalo více implementací tohoto registrovaného bodu, například v různých rozšířeních návrháře objektů, pak je tímto parametrem řečeno, v jakém pořadí se má registrovaná funkce spustit. Spouští se od nejmenšího čísla.

Rozšířené zadání
Tímto parametrem přepneme registrovanou funkci do pokročilého režimu. V jednoduchém režimu máme k dispozici jednu možnou implementaci metody. Nachází se na záložce ***„Kód"***. V případě pokročilejšího režimu máme k dispozici více možností, o kterých se dozvíme dále v textu.

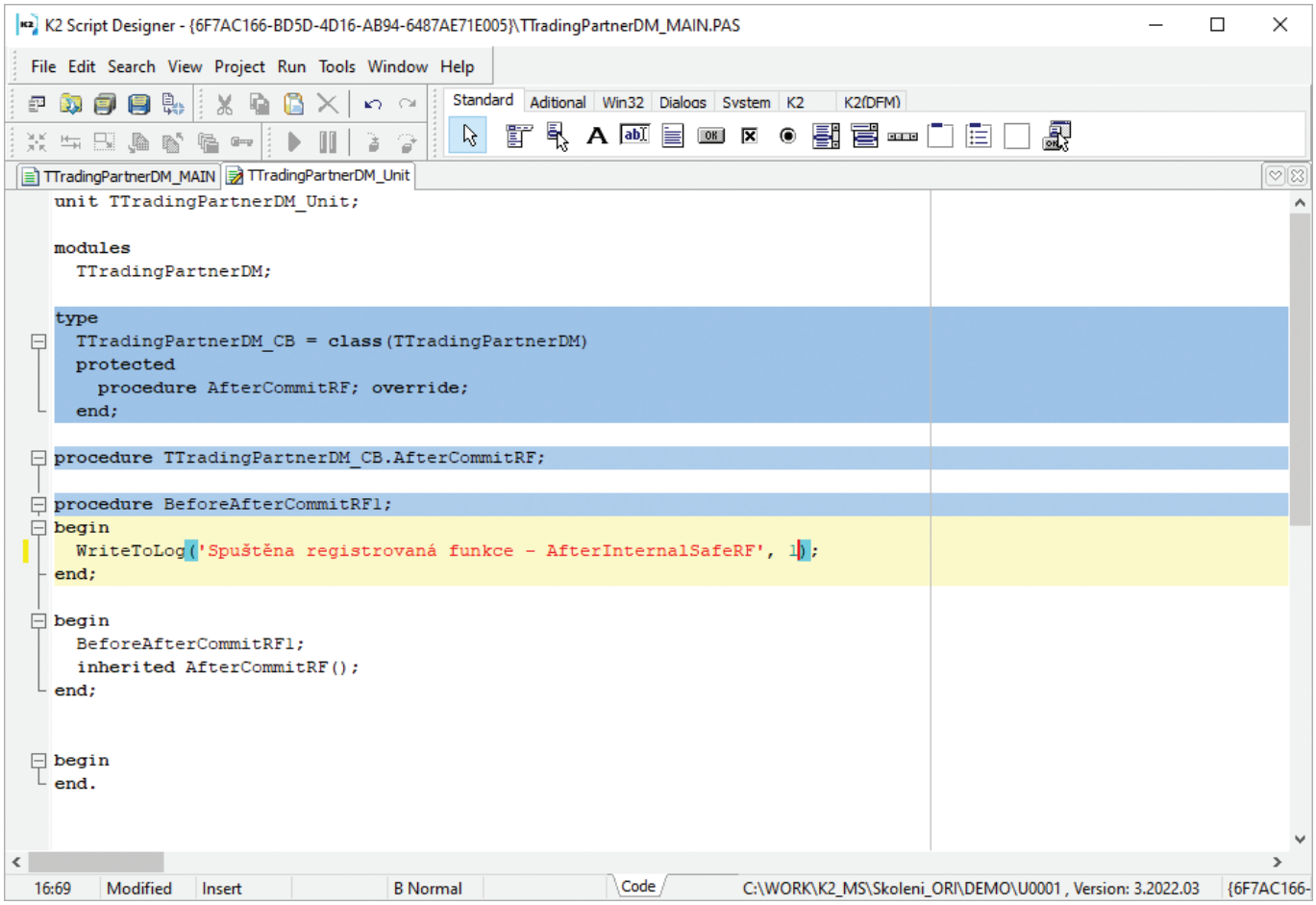
POZNÁMKA: V případě, že k registrované funkci existují části, které se spouští před a po daném umístění, je automaticky zobrazen formulář v režimu ***„Rozšířené zadání"***.

Pro implementaci metody v ***„jednoduchém režimu"*** se přepneme na záložku ***„Kód"*** a provedeme jeho implementaci. Na Obrázek 206 - Implementace registrované funkce AfterInternalSaveRF v datovém modulu je vidět implementace, která je vyvolána po úspěšné databázové operaci uložení záznamu v datovém modulu. V našem konkrétním příkladu se zapíše do logu informace o spuštění registrované funkce.



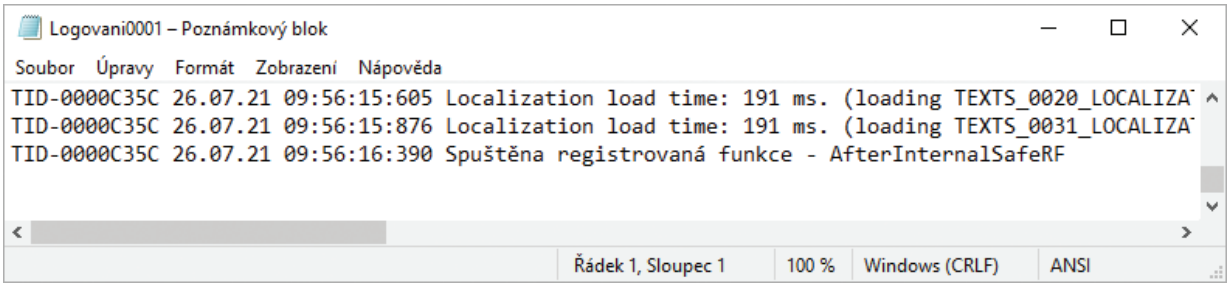
Obrázek 206 - Implementace registrované funkce AfterInternalSaveRF v datovém modulu

Implementaci funkce můžeme provést i pomocí editoru skriptu. Po jeho spuštění je vidět implementace dané metody pro registrovaný bod. Vzniká procedura ***„BeforeAfterInternal SaveRF1“***, která obsahuje naší implementaci. Vše je vidět na Obrázek 207 - Implementace registrované funkce AfterInternalSaveRF v datovém modulu - editor skriptu.



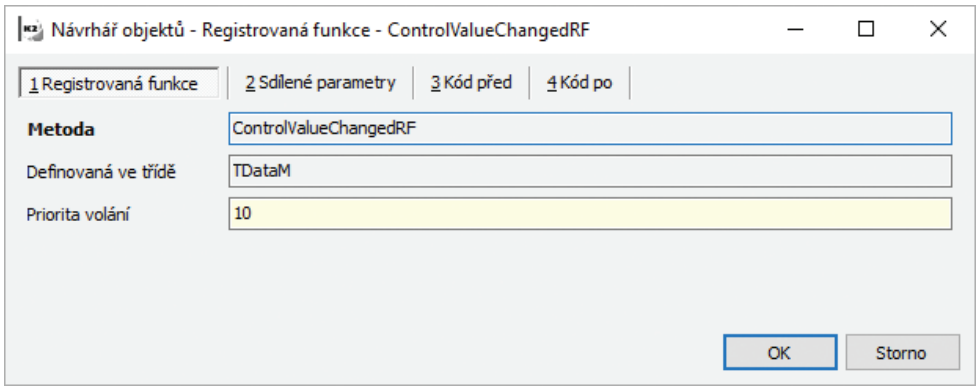
Obrázek 207 - Implementace registrované funkce AfterInternalSaveRF v datovém modulu - editor skriptu

Po provedení akce ***„Deploy“*** můžeme výše definovanou implementaci vyzkoušet. Implementovali jsme registrovanou proceduru, která je vyvolána při ukládání záznamu, a to v standardním datovém modulu ***„Dodavatelé / odběratelé“***. Po otevření datového modulu a přepnutí existujícího záznamu do stavu editace, provedeme uložení záznamu. Při nastaveném logování a otevření logu můžeme vidět, že se zapíše do logu spuštění naší implementace registrovaného bodu při ukládání záznamu. Vše je vidět na Obrázek 208 - Projev registrované funkce AfterInternalSaveRF v datovém modulu.



Obrázek 208 - Projev registrované funkce AfterInternalSaveRF v datovém modulu

POZNÁMKA: Aby se nám log vytvořil, je nutné mít v K2 zapnuté logování s LOGLEVEL=1.
V případě, že budeme chtít provést detailnější nastavení registrované funkce, zapneme ve formuláři definice registrované funkce, volbu ***„Rozšířené zadání"***. Zobrazí se navíc záložky ***„Sdílené parametry"***, a záložka ***„Kód"*** se rozdělí na dvě ***„Kód před"*** a ***„Kód po"***, viz Obrázek 209 - Rozšířené zadání pro registrované funkce datového modulu.

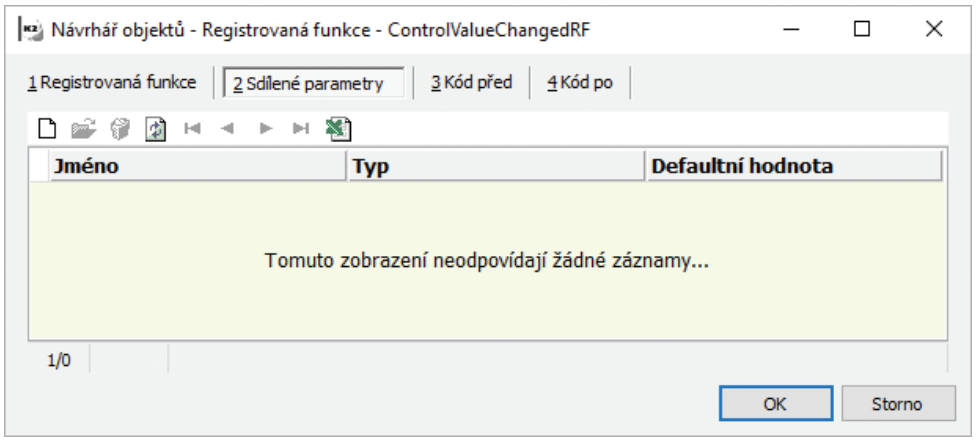


Obrázek 209 - Rozšířené zadání pro registrované funkce datového modulu

4.10.3. ZÁLOŽKA „SDÍLENÉ PARAMETRY"

Na této záložce můžeme definovat parametry, pomocí kterých si můžeme předávat hodnoty mezi metodami ***„Kód před"*** a ***„Kód po"***, které budou popsány dále v textu.

Na záložce vidíme seznam všech sdílených parametrů, viz Obrázek 210 - Seznam sdílených parametrů v registrované funkci datového modulu.



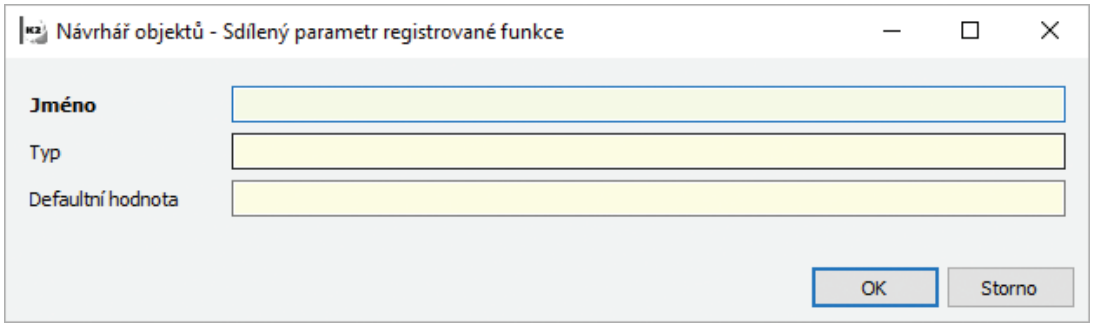
Obrázek 210 - Seznam sdílených parametrů v registrované funkci datového modulu

Stisknutím tlačítka Nový nebo stisknutím klávesy Insert se zobrazí formulář pro zadání nového sdíleného parametru, viz Obrázek 211 - Nový sdílený parametr registrované funkce datového modulu. Vstupem jsou následující hodnoty.

Jméno
Název parametru. Všechny definované parametry jsou pak definovány jako parametry funkcí (procedur) v ***„Kód před"*** a ***„Kód po"***.

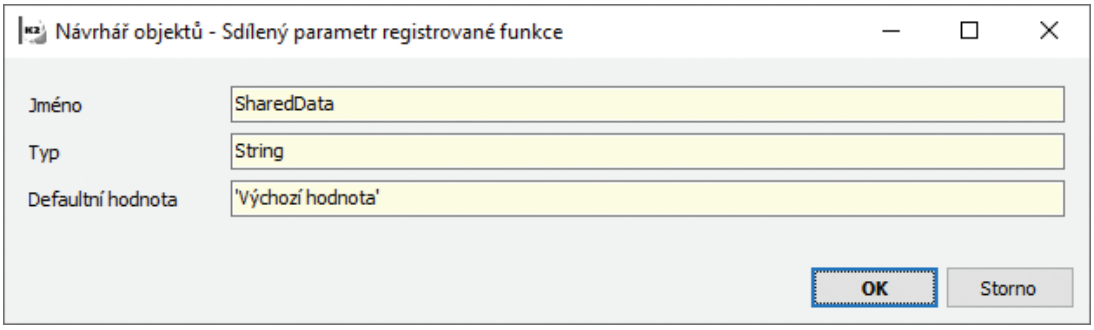
Typ
Udává typ proměnné. V této části není definován výčtový typ pro výběr. Je tedy nutné dbát na přesnou syntaxi typu, která odpovídá skriptovým datovým typům. Například pro text použijeme ***„string"***.

Defaultní hodnota
Zde nastavujeme výchozí hodnotu sdíleného parametru.



Obrázek 211 - Nový sdílený parametr registrované funkce datového modulu

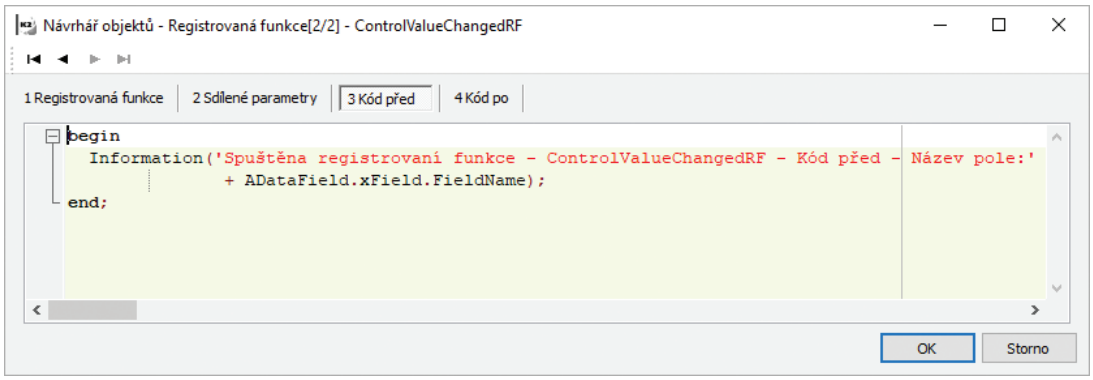
Jako příklad si vytvoříme parametr, který nazveme ***„SharedData“***, jako typ nastavíme ***„string“*** a jeho výchozí hodnotu nastavíme na text ***„Výchozí hodnota“***, viz Obrázek 212 - Vytvoření sdíleného parametru registrované funkce. Tento parametr bude fungovat jako ***„vstupně/výstupní“*** a to jak v proceduře definované v kódu ***„Kód před“***, tak v proceduře definované v ***„Kód po“***. Vše je vidět na obrázku, který shrnuje definici registrované funkce, viz Obrázek 215 - Editor skriptu a implementace procedur pro pokročile nastavení registrovaných funkcí.



Obrázek 212 - Vytvoření sdíleného parametru registrované funkce

4.10.4. ZÁLOŽKA „KÓD PŘED“

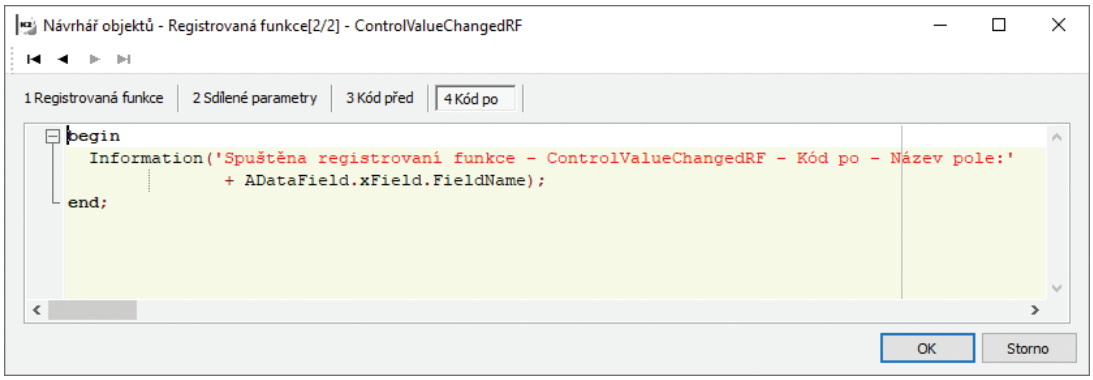
Na záložce ***„Kód před“*** můžeme implementovat proceduru, která je vyvolána při změně pole datového modulu. V našem příkladu je implementace zobrazení hlášení, že dochází ke spuštění registrované funkce pro změnu v poli záznamu ve fázi ***„před“***, viz Obrázek 213 - Záložka "Kód před" v registrované funkci datového modulu.



Obrázek 213 - Záložka "Kód před" v registrované funkci datového modulu

4.10.5. ZÁLOŽKA „KÓD PO“

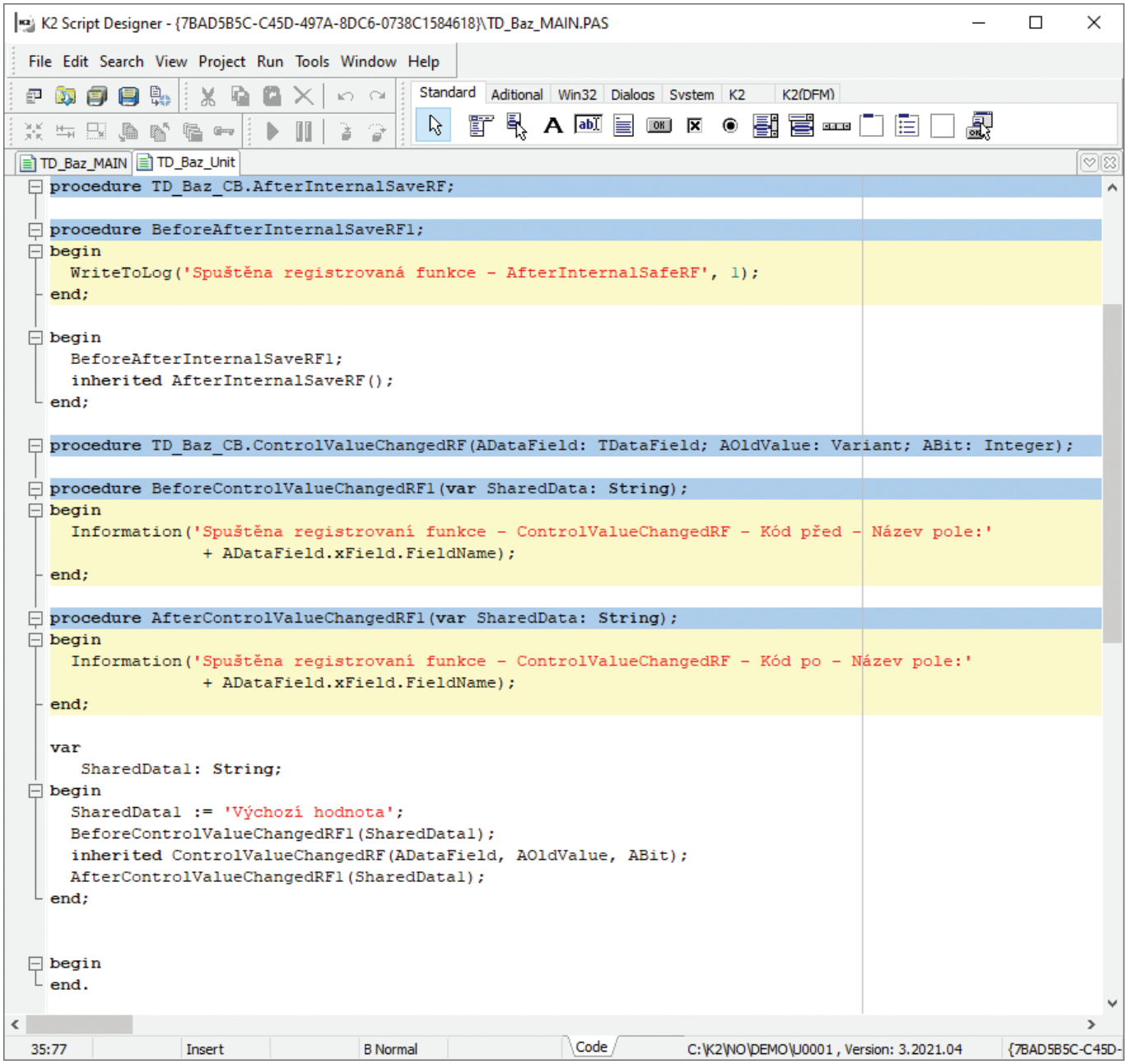
Na záložce ***„Kód po“*** můžeme implementovat proceduru, která je vyvolána při změně pole datového modulu. V našem příkladu je implementace zobrazení hlášení, že došlo ke spuštění registrované funkce pro změnu v poli záznamu ve fázi ***„po“***, viz Obrázek 214 - Záložka "Kód po" v registrované funkci datového modulu.



Obrázek 214 - Záložka "Kód po" v registrované funkci datového modulu

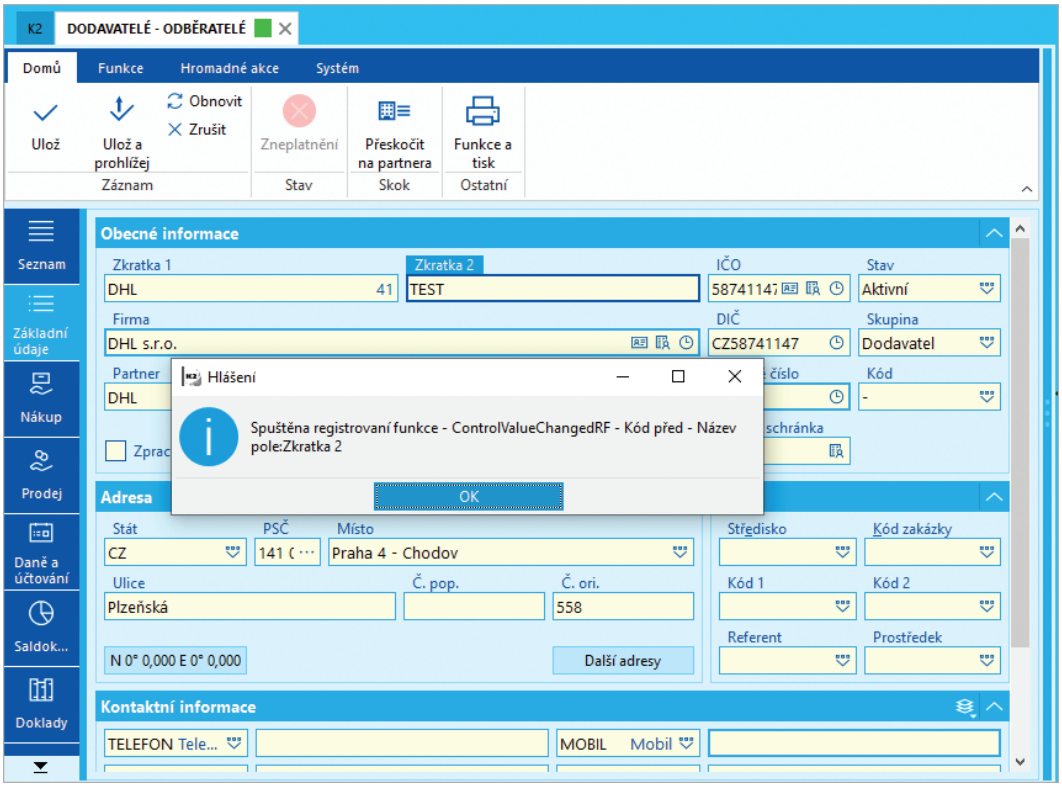
POZNÁMKA: Samozřejmě můžeme implementaci těchto procedur provést přímo v editoru skriptu. Můžeme vidět, že pro naše metody byly vytvořeny procedury ***„BeforeControlValueChangedRF1“*** a ***„AfterControlValueChangedRF1“***. Vše je vidět na Obrázek 215 - Editor skriptu a implementace procedur pro pokročile nastavení registrovaných funkcí.

POZNÁMKA: Na obrázku je vidět definice procedur ***„BeforeControlValueChangedRF1“*** a ***„AfterControlValueChangedRF1“***, kde obě obsahují ***„var“*** (vstupně/výstupní) sdílený parametr ***„SharedData“*** typu ***„string“***, který jsme si definovali při popisu dané funkčnosti. Pomocí tohoto parametru si můžeme předávat data v jednotlivé fázi spuštění registrované metody.



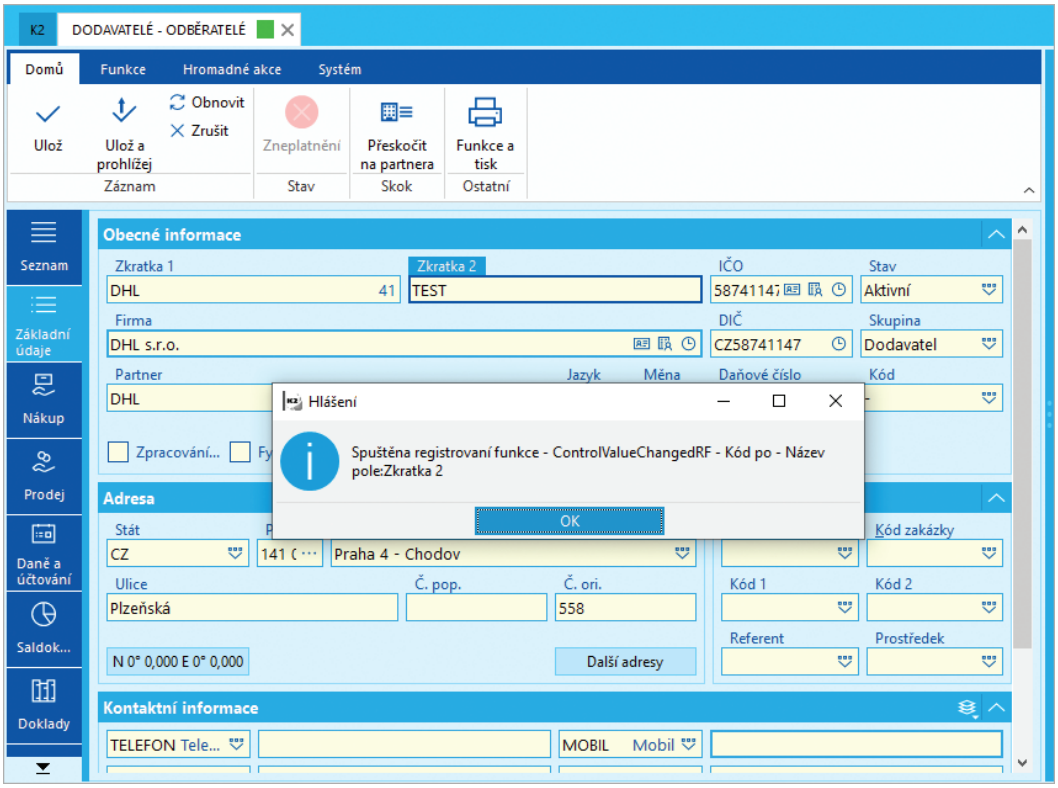
Obrázek 215 - Editor skriptu a implementace procedur pro pokročile nastavení registrovaných funkcí

Po provedení akce ***„Deploy“*** můžeme výše definovanou implementaci vyzkoušet. Implementovali jsme registrovanou proceduru, která je vyvolána při změně hodnoty pole, a to ve standardním datovém modulu ***„Dodavatelé / odběratelé“***. Po otevření datového modulu a přepnutí existujícího záznamu do stavu editace, provedeme změnu některého pole a opustíme ho. Můžeme vidět, že se provede nejprve spuštění naší implementace registrovaného bodu při změně pole před spuštěním kódu datového modulu, který reaguje na změnu, viz Obrázek 216 - Spuštění registrované funkce po změně pole v datovém modulu – Kód za.



Obrázek 216 - Spuštění registrované funkce po změně pole v datovém modulu – Kód za

Dále je vyvoláno spuštění naší implementace registrovaného bodu při změně pole po spuštění kódu datového modulu, který reaguje na změnu, viz Obrázek 217 - Spuštění registrované funkce při změně pole v datovém modulu – Kód po.



Obrázek 217 - Spuštění registrované funkce při změně pole v datovém modulu – Kód po

4.11. ZÁVISLOSTI MODULŮ

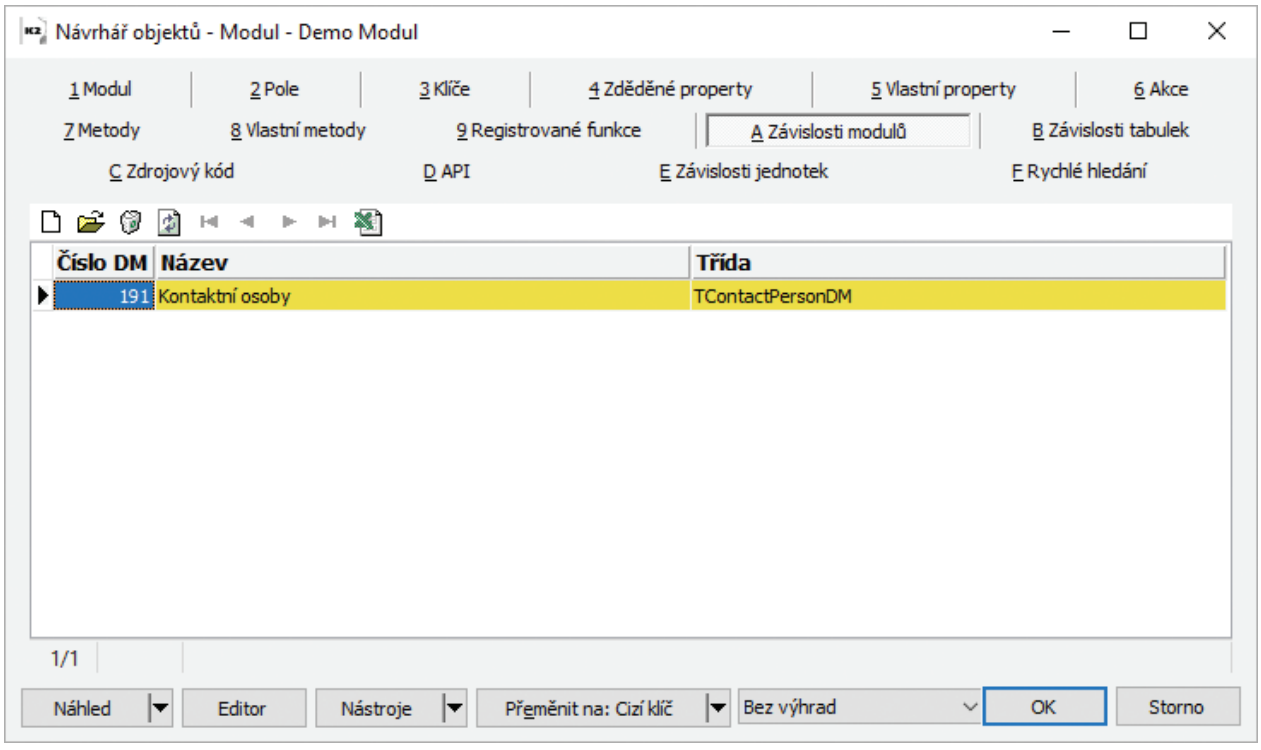
Pod vytvořenými datovými moduly v návrháři objektů se skrývá jejich implementace ve skriptu K2. Což si ukážeme později v další kapitole. Někdy je potřeba ve skriptu přistupovat k jiným datovým modulům, využívat jejich funkce, číst, vkládat data apod. Abychom ve skriptu datové moduly mohli používat, je potřeba každý takový modul připojit, aby ho náš skript ***„znal"***. Připojení se provádí pomocí definice ***„modules"*** přímo ve skriptu.

Datový modul rozšíříme, o znalost jiného modulu, přidáním závislosti. Vložení provedeme stisknutím tlačítka Nový záznam v nástrojové liště nad seznamem závislostí nebo stisknutím klávesy Insert. Zobrazí se seznam všech datových modulů, které jsou zaregistrované v K2. Po výběru požadovaného modulu, se zobrazí formulář, se souhrnem výběru datového modulu viz Obrázek 219 - Definice závislosti v datovém modulu. Tlačítkem Ok potvrdíme vkládanou závislost, která se následně objeví v seznamu, jak je vidět na Obrázek 218 - Seznam závislostí v datovém modulu.

PŘÍKLAD: Připojení datového modulu kontaktních osob, jak je vidět na Obrázek 218 - Seznam závislostí v datovém modulu, pak ve skriptu vypadá následovně.

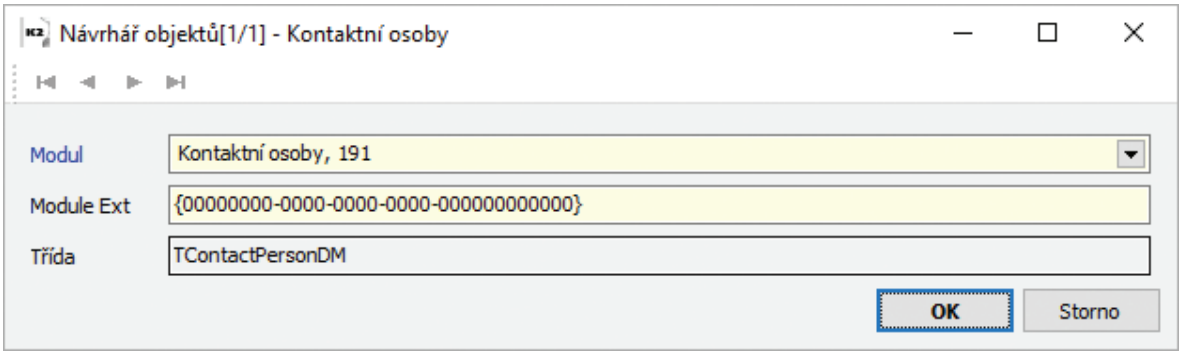
```
modules
   TContactPersonDM;
```

Tedy skript, který je vygenerovaný návrhářem objektů pro náš datový modul, má připojen modul ***„Kontaktní osoby"*** a my ho můžeme použít ve svých akcích, metodách apod.



Obrázek 218 - Seznam závislostí v datovém modulu

Změnu již definované závislosti provedeme dvojím kliknutím myší na záznam, stisknutím tlačítka Upravit záznam v nástrojové liště nad seznamem závislostí nebo stisknutím klávesy F5. Zobrazí se formulář pro editaci, viz Obrázek 219 - Definice závislosti v datovém modulu.



Obrázek 219 - Definice závislosti v datovém modulu

4.12. ZÁVISLOSTI TABULEK

V IS K2 existují skriptové jednotky, které začínají prefixem ***„FM_“***. Tyto jednotky existují pro každou databázovou tabulku K2 a jejich obsahem je seznam všech polí a indexů, které daná tabulka obsahuje. Seznam je ve formě konstant, které je možné využít v případě přístupu ke standardním modulům K2.

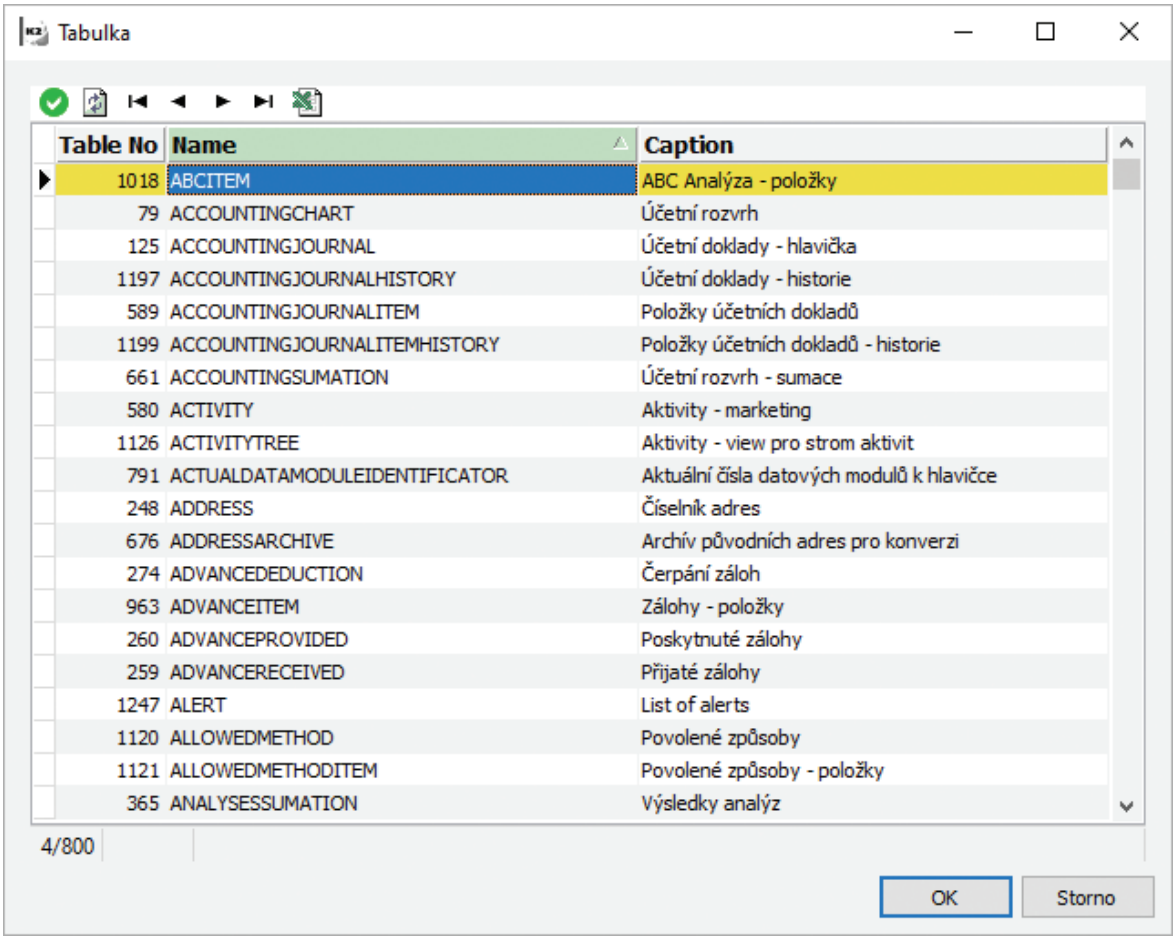
POZNÁMKA: Do návrháře objektů nelze do závislostí připojit žádné standardní skripty K2. Lze použít pouze skripty, které vznikly v NO. Tato vlastnost neumožňovala používat ***„FM“*** soubory, které v K2 existovaly v podobě fyzických skriptových souborů. Soubory proto byly zrušeny a převedeny na generované v paměti K2. Díky tomuto je možné je použít i v návrháři objektů.

PŘÍKLAD: Chceme využít modul dodavatelů / odběratelů v našem vlastním modulu. S modulem budeme pracovat na úrovni skriptu, kde budeme potřebovat znát čísla polí a klíčů připojeného modulu. Abychom nemuseli tyto informace duplikovat nebo psát do kódu přímo hodnoty místo konstant, připojíme si tabulku, která odpovídá modulu, se kterým pracujeme. Jedná se o tzv. ***„FM“*** soubory, které obsahují tyto informace.

Závislost na tabulce založíme stisknutím klávesy Insert nebo stisknutím tlačítka Nový nad seznamem závislostí v záložce ***„A Závislosti tabulek“***.

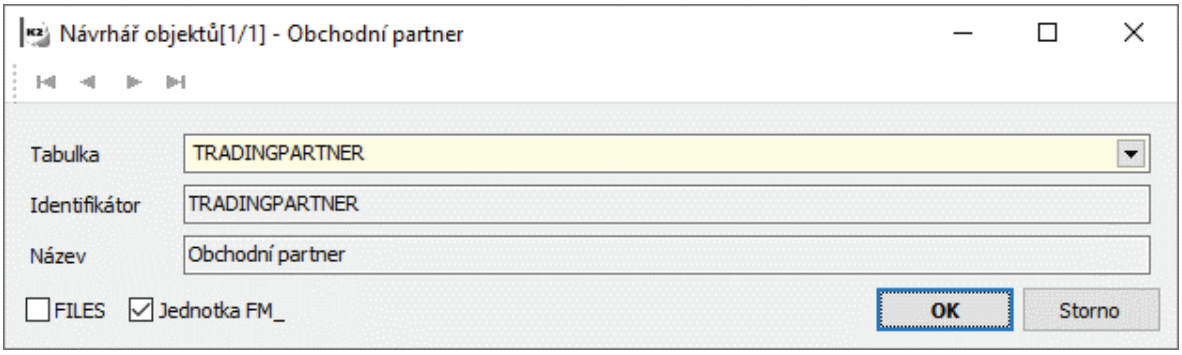
Po stisknutí se zobrazí seznam všech existujících ***„FM“*** souborů, ze kterých vybereme požadovanou tabulku, viz Obrázek 219 - Definice závislosti v datovém modulu.

POZNÁMKA: V tomto seznamu můžeme využít redukční hledání přímým psaním hledaných slov.



Obrázek 220 - Seznam tabulek ze závislostí

Po výběru tabulky se zobrazí formulář pro potvrzení vybraného záznamu, viz Obrázek 221 - Potvrzení výběru závislé tabulky.

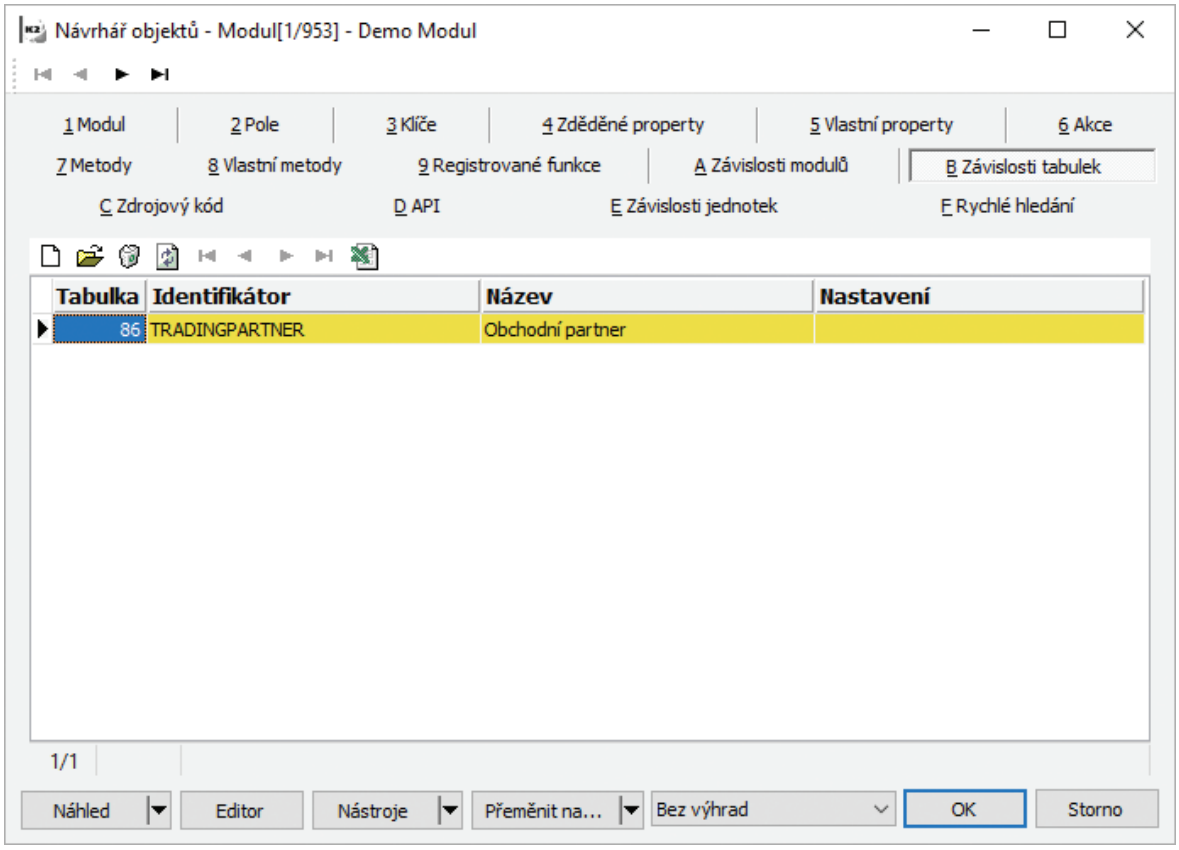


Obrázek 221 - Potvrzení výběru závislé tabulky

Příznak – *„FILES“*
Pokud potřebujeme do závislostí připojit i odkaz na ***„xfile“*** daného modulu, je možné zatrhnout volbu ***„FILES“***. Tabulka se následně do skriptu vloží do seznamu ***„files“***.

Následně se tabulka vloží do seznamu, viz Obrázek 222 - Seznam závislých tabulek datového modulu.

Příznak – *„Jednotka FM_“*
Tento příznak se automaticky zatrhne při výběru tabulky, kdybychom nepotřebovali z nějakého důvodu mít připojenou jednotku FM_, můžeme daný příznak shodit.



Obrázek 222 - Seznam závislých tabulek datového modulu

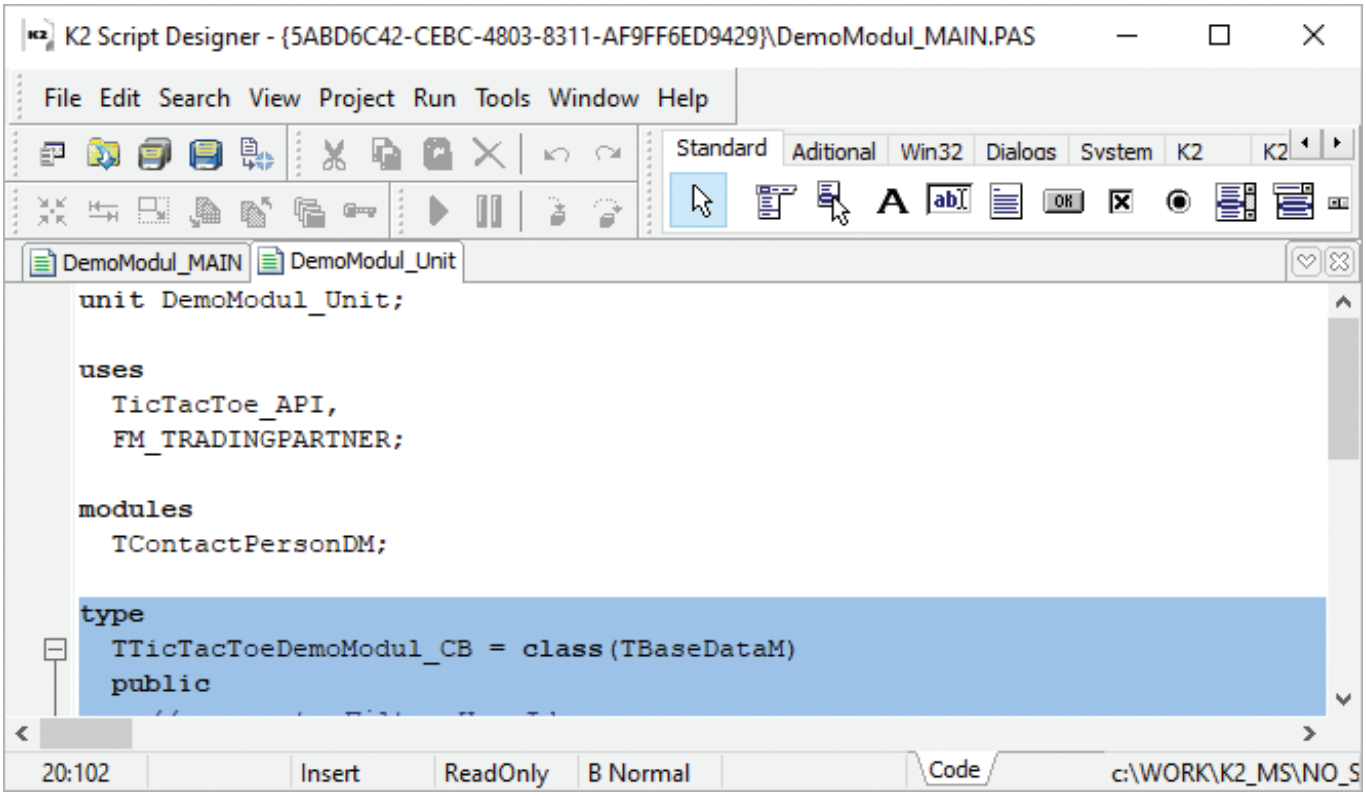
V kódu návrháře objektů pak vypadá následovně.

```
uses
   FM_TRADINGPARTNER;
```

Tedy skript, který je vygenerovaný návrhářem objektů pro náš datový modul, má připojenu jednotku ***„TRADINGPARTNER“***, čímž ji můžeme použít ve svých akcích, metodách apod., viz Obrázek 223 - Připojená tabulka v kódu datového modulu.

V případě zatržení volby ***„FILES“***, se do zdrojového kódu navíc přidá odkaz na souborový modul – sekce ***„files“***.

```
files
   TRADINGPARTNER;
```

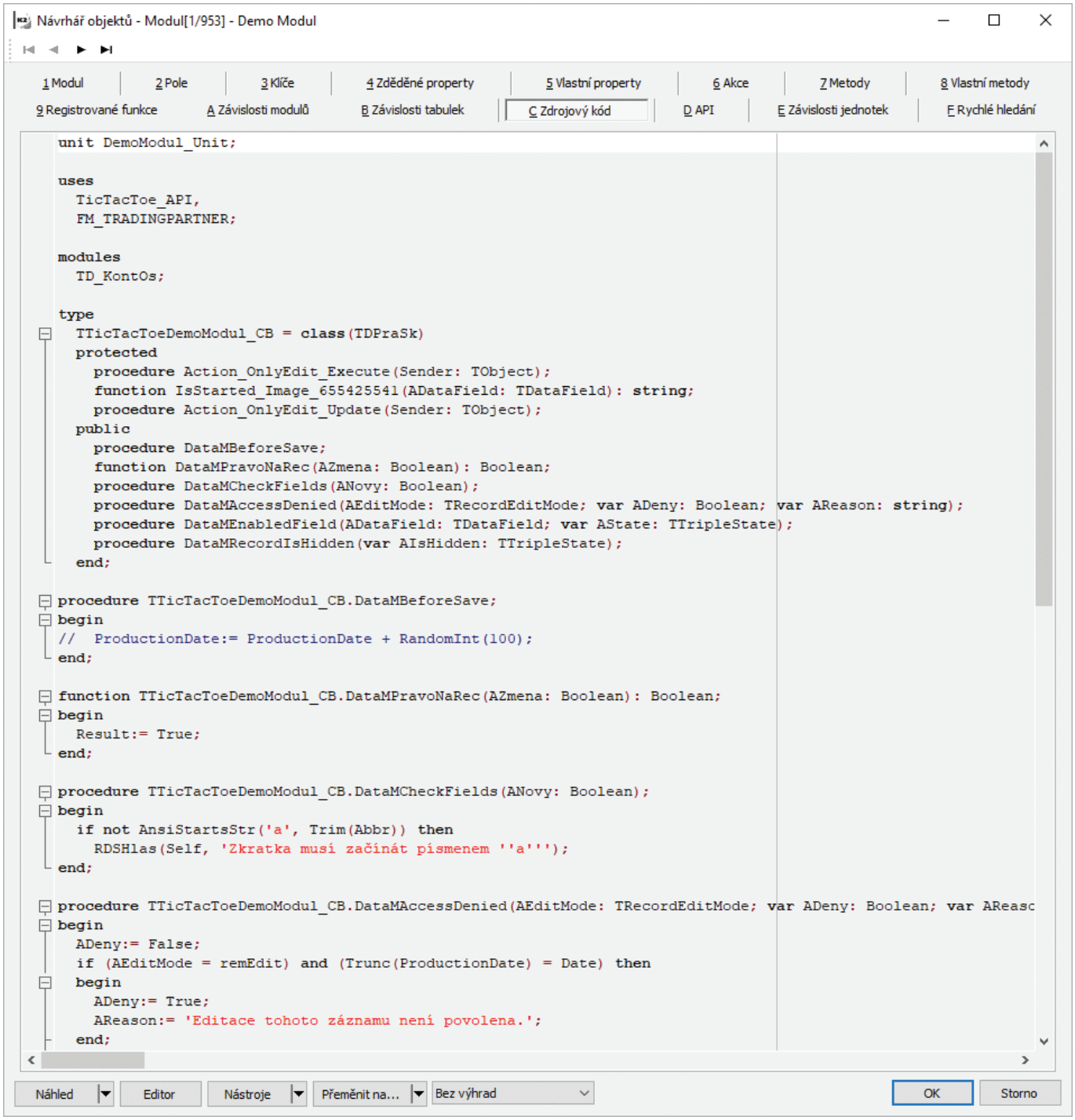


Obrázek 223 - Připojená tabulka v kódu datového modulu

4.13. ZDROJOVÝ KÓD

V předchozí podkapitole ***„Závislosti"*** jsme nastínili, že pro datové moduly generuje návrhář objektů skript, který modul v podstatě definuje a obsahuje veškeré akce, vlastnosti a metody, které v návrháři objektů nadefinujeme. Na této záložce můžeme celý skript prohlížet. Editace v této záložce není možná. Slouží pouze pro náhled zdrojového kódu datového modulu. Editaci si popíšeme v další kapitole ***„Editor"***.

Na Obrázek 224 - Zdrojový kód datového modulu je vidět vygenerovaný zdrojový kód pro vytvořený datový modul. Jsou zde vidět některé závislosti, které jsme zatím v průběhu školení zkoušeli na modulu ***„DEMO"***.



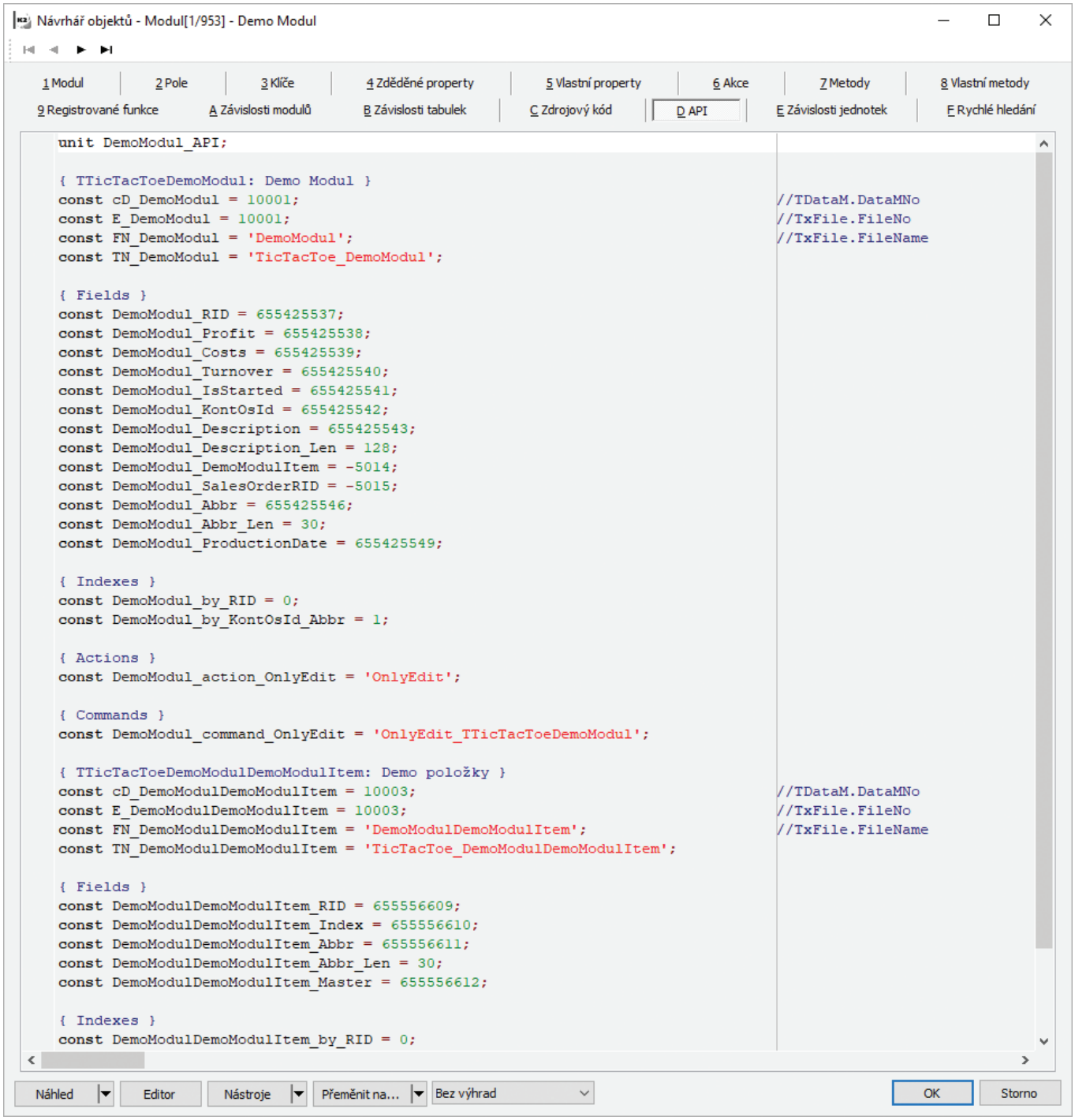
Obrázek 224 - Zdrojový kód datového modulu

4.14. API

Další záložkou v definici datového modulu je ***„API“*** neboli ***„aplikační rozhraní“***. Slouží k zobrazení seznamu všech konstant, které jsou vytvořeny pro aktuální datový modul. Tedy identifikátory datového modulu, všech polí, klíčů a akcí, které jsou v datových modulech nadefinovány. Tento skriptový soubor je připojen, pomocí klíčového slova ***„uses“***, ke zdrojovému kódu, což je vidět na Obrázek 224 - Zdrojový kód datového modulu v předchozí kapitole. Skript s vygenerovaným API je vidět na Obrázek 225 - API datového modulu.

POZNÁMKA: Pro každý datový modul vznikne

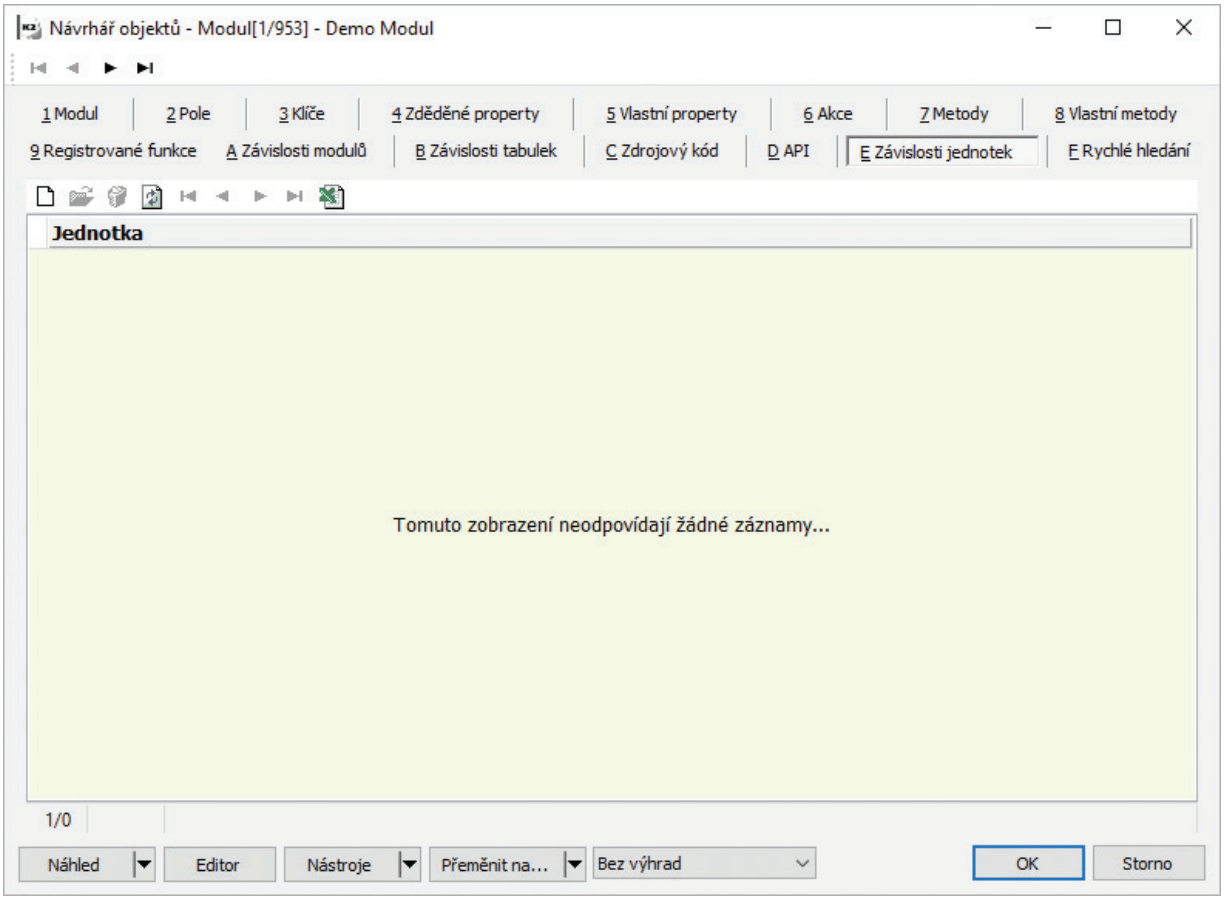
- konstanta s prefixem *„cD_“* + identifikátor modulu, která obsahuje číslo datového modulu
- konstanta s prefixem *„E_“* + identifikátor databázové tabulky, která obsahuje číslo databázové tabulky datového modulu
- konstanta s prefixem *„FN_“* + identifikátor databázové tabulky, která obsahuje název databázové tabulky datového modulu bez případného prefixu
- konstanta s prefixem *„TN_“* + identifikátor databázové tabulky, která obsahuje název databázové tabulky datového modulu s případným prefixem



Obrázek 225 - API datového modulu

4.15. ZÁVISLOSTI JEDNOTEK

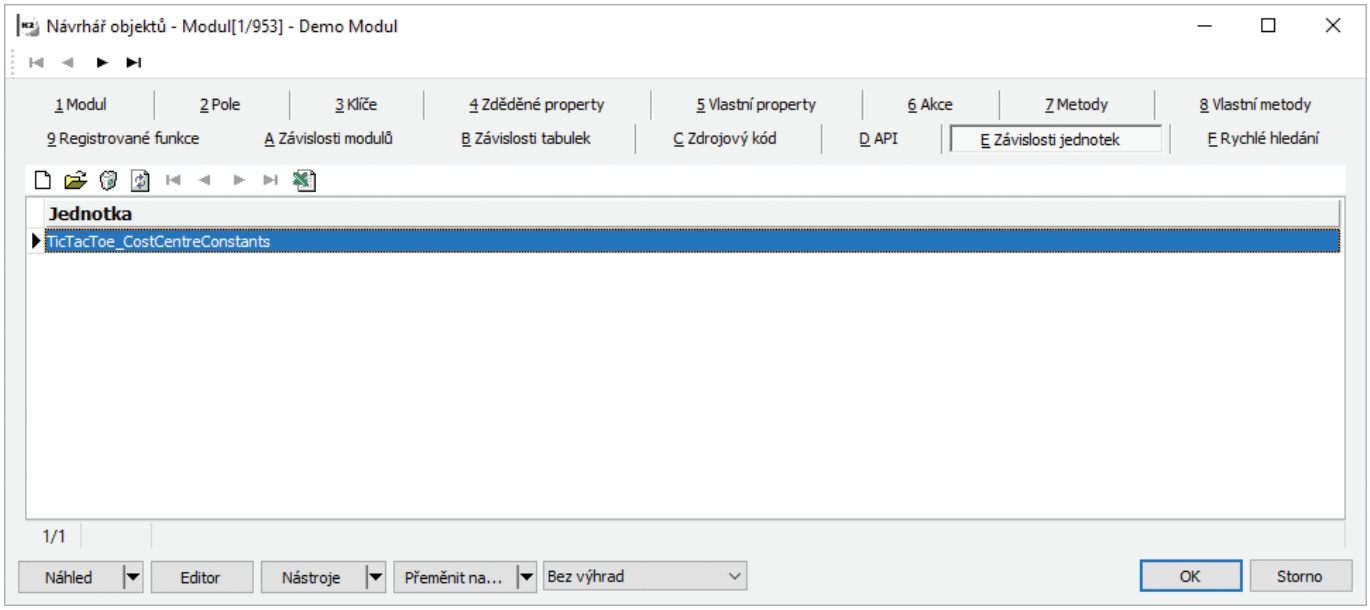
Další dostupnou záložkou pro datový modul je definice závislých jednotek. Funguje na stejném principu jako evidence na záložce ***„Závislosti“***, kde se definuje datové moduly, které potřebujeme nějakým způsobem použít v implementaci modulu. V záložce pro závislosti jednotek se definují jednotky a skriptové třídy (objekty), které potřebujeme použít v našem datovém modulu.



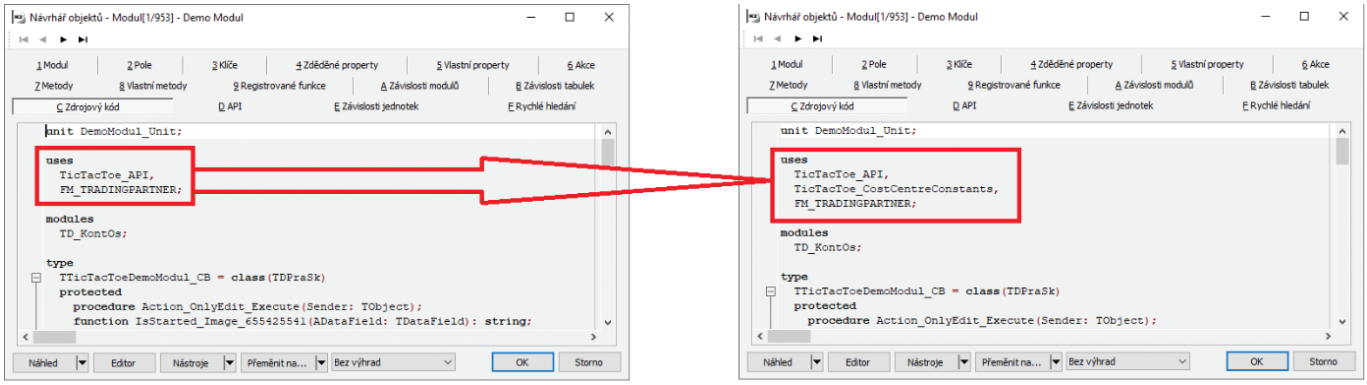
Obrázek 226 - Závislosti jednotek v datovém modulu

Přidat jednotku můžeme pomocí stisknutí tlačítka Nový nebo klávesou Insert. Zobrazí se nám seznam dostupných jednotek a skriptových tříd, které máme dostupné i na záložce ***„Jednotky“***, která je popsána v kapitole 11. Záložka „Jednotky“. Tedy k dispozici budeme mít všechny včetně těch, které používáme z jiných nainstalovaných balíčků a máme je připojené v závislostech na záložce ***„Závisí na“***, viz kapitola 12. Záložka „Závisí na“. Po výběru jednotky se kódu datového modulu vytvoří sekce ***„uses“***, ve které budeme mít připojenou naší jednotku.

V případě, že připojíme skriptovou třídu, kterou máme nazvanou ***„CostCentreConstants“*** z předchozího příkladu, viz Obrázek 227 - Připojení závislého objektu do datového modulu, můžeme vidět ve zdrojovém kódu datového modulu její připojení, viz Obrázek 228 - Závislá jednotka v datovém zdrojovém kódu datového modulu.



Obrázek 227 - Připojení závislého objektu do datového modulu

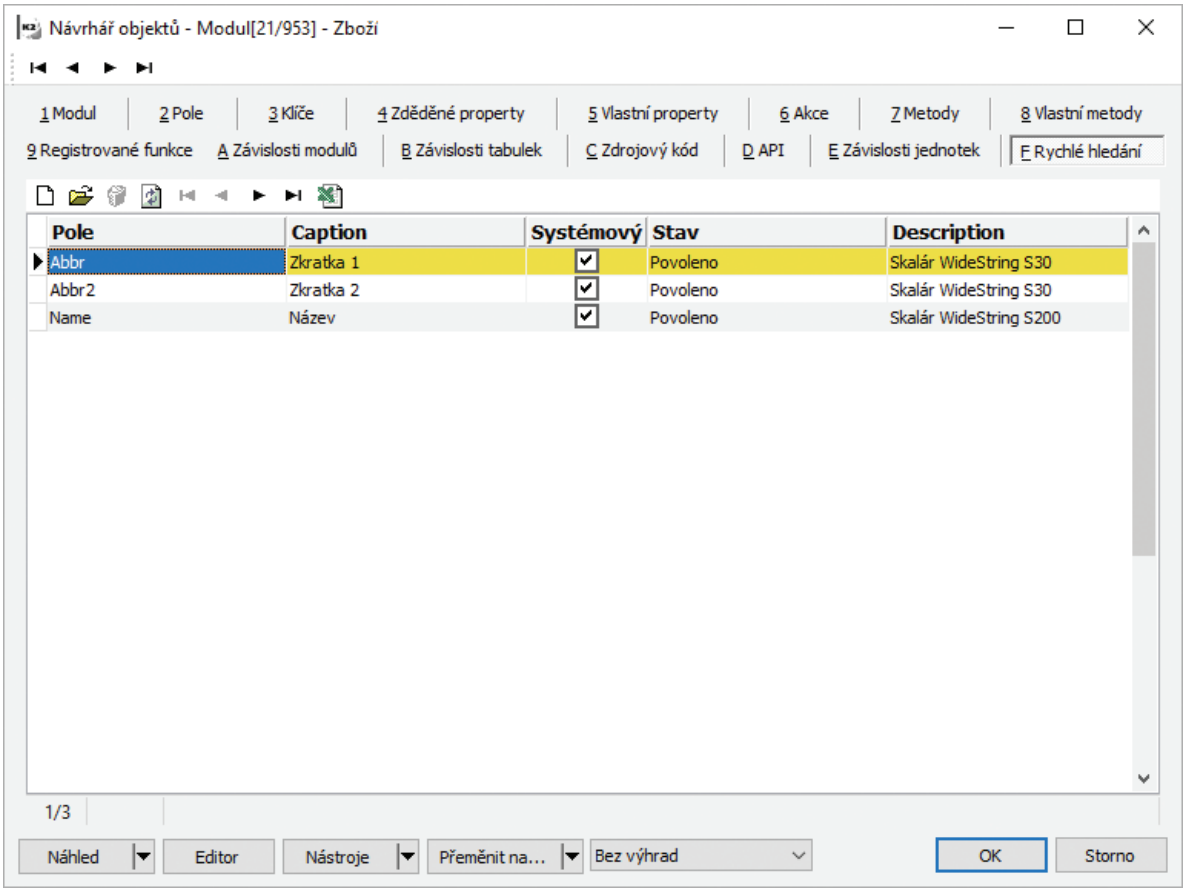


Obrázek 228 - Závislá jednotka v datovém zdrojovém kódu datového modulu

4.16. RYCHLÉ HLEDÁNÍ

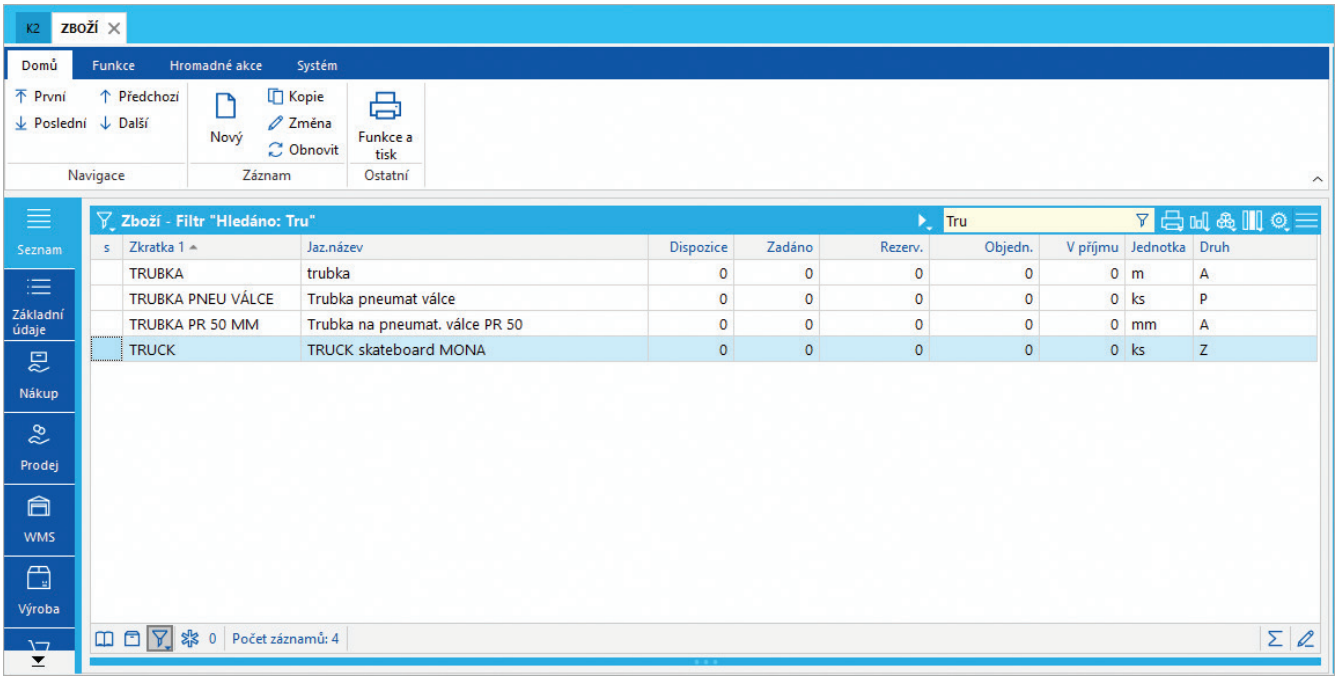
V univerzálních formulářích K2 je možné prohledávat záznamy otevřeného modulu pomocí tzv. ***„Rychlého hledání“***, které je dostupné pod klávesovou zkratkou Ctrl + Shift + F. Po zadání klíčového slova se prohledávají jednotlivé záznamy, a to ve sloupcích, které jsou k tomu určeny. Prohledávací podmínka je postavená pouze na výrazu text ***„začíná na“***. Ve kterých sloupcích bude aplikace prohledávat, určuje definice rychlého hledání, která je dostupná v návrháři objektů, kde je zároveň možné tyto sloupce potlačit nebo přidat vlastní. Vše si popíšeme v následujícím textu.

Například v modulu ***„Zboží“*** je definováno rychlé hledání přes pole ***„Abbr“***, ***„Abbr2“*** a ***„Name“***, viz Obrázek 229 - Definice rychlého hledání na modulu "Zboží".



Obrázek 229 - Definice rychlého hledání na modulu "Zboží"

Po otevření modulu ***„Zboží“*** v univerzálních formulářích a stisku klávesy Ctrl + Shift +F se zobrazí nad seznamem pole pro hledání. Při zadání textu, například ***„Tru“***, a stisku klávesy Enter se odfiltrují ty záznamy, které začínají ve sloupci ***„Abbr“*** nebo ***„Abbr2“*** nebo ***„Name“*** na hodnotu ***„Tru“***, viz Obrázek 230 - Použití rychlého hledání v modulu "Zboží".



Obrázek 230 - Použití rychlého hledání v modulu "Zboží"

Nejprve si popíšeme, jak na vlastním modulu definovat pole pro rychlé hledání, následně popíšeme úpravy standardní definice.

4.16.1. RYCHLÉ HLEDÁNÍ NA VLASTNÍM MODULU

V případě, že chceme využít funkce rychlého hledání na vlastním modulu vytvořeném v návrháři objektů, otevřeme vlastní datový modul a přepneme se na záložku ***„Rychlé hledání“***. Zde uvidíme několik informací ke každému záznamu, viz Obrázek 231 - Definice rychlého hledání na vlastním modulu.

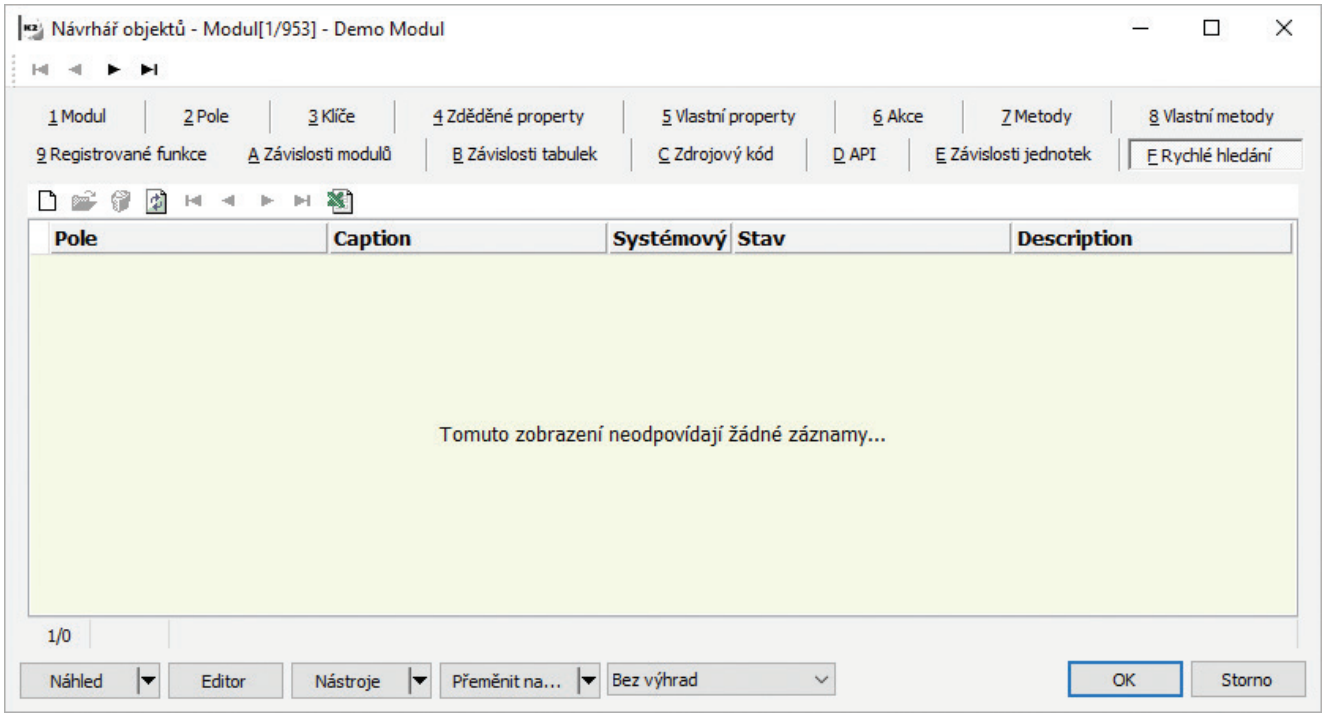
Pole
Identifikátor pole z datového modulu, pro které je definováno rychlé hledání.

Caption
Název pole z datového modulu, pro které je definováno rychlé hledání.

Systémový
Informace, zda se jedná o standardní nebo vlastní definici pro hledání. Jedná se o případy, kdy rozšiřujeme standardní modul v této oblasti.

Potlačit
Informace, zda je pole používáno či potlačeno (deaktivováno). Potlačit lze pouze standardní definice rychlého hledání.

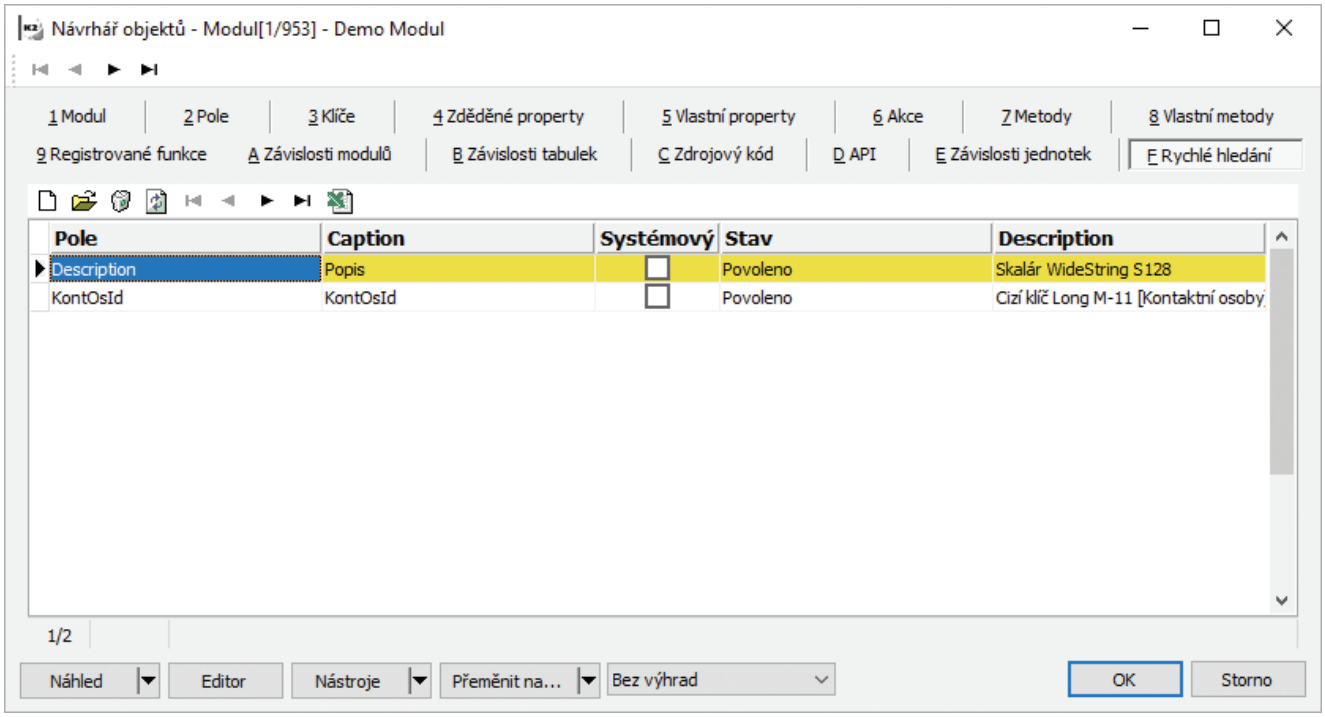
Description
Popis pole, které je zde vybráno.



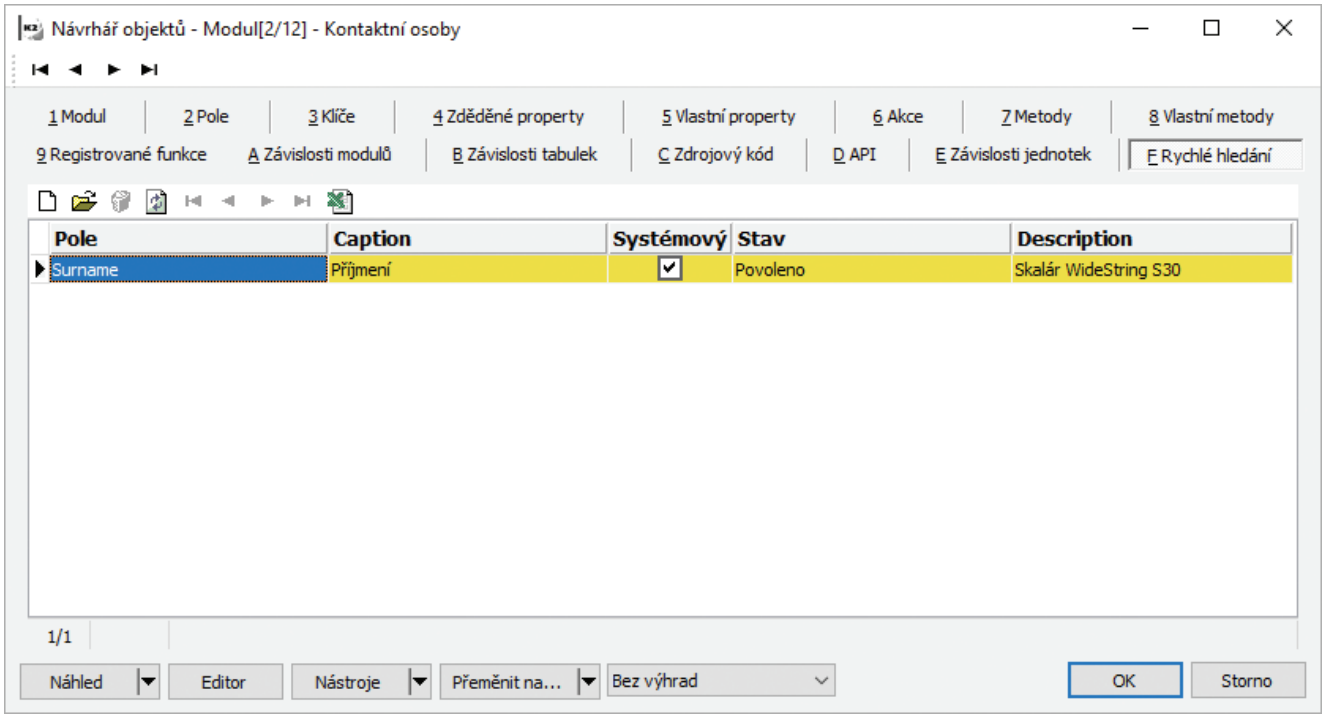
Obrázek 231 - Definice rychlého hledání na vlastním modulu

Novou definici rychlého hledání přidáme stisknutím tlačítka Nový nebo klávesou Insert. Zobrazí se seznam všech polí aktuálního datového modulu. Vybereme pole, které chceme zařadit do hledání a potvrdíme tlačítkem Ok. Například máme pole, které obsahuje popis, tedy ***„Description“***, které můžeme přidat, viz Obrázek 232 - Definice rychlého hledání na vlastním modulu – pole.

Do definice můžeme vložit i pole, která slouží jako cizí klíče do jiných datových modulů, tedy jsou přes ně realizovány vazby. Například v modulu máme vazbu do standardního modulu ***„Kontaktní osoby“***, viz Obrázek 232 - Definice rychlého hledání na vlastním modulu – pole. V tomto případě se navíc prohledávají všechna pole, která jsou definována v rychlém hledání na modulu, do kterého směřuje vazba, viz Obrázek 233 - Definice rychlého hledání na standardním modulu. Tedy v našem příkladu se prohledává pole ***„Description“*** z vlastního modulu a zároveň pole ***„Surname“*** (příjmení) z kontaktních osob, které je v záznamu.



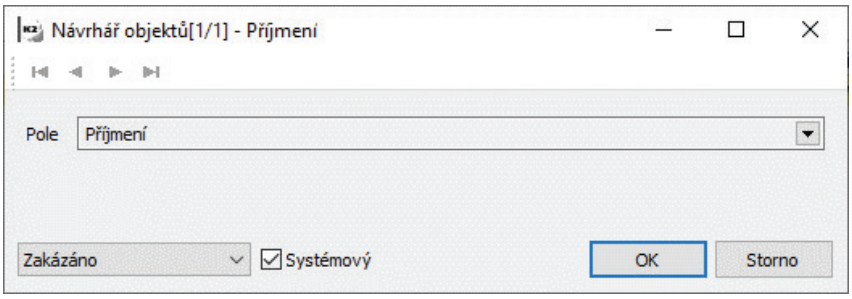
Obrázek 232 - Definice rychlého hledání na vlastním modulu – pole



Obrázek 233 - Definice rychlého hledání na standardním modulu

4.16.2. RYCHLÉ HLEDÁNÍ NA STANDARDNÍM MODULU

Na Obrázek 233 - Definice rychlého hledání na standardním modulu je vidět definice rychlého hledání pro standardní modul ***„Kontaktní osoby“***. Definici můžeme rozšířit o nová pole, přes která se také bude vyhledávat, případně můžeme standardní definici deaktivovat. Záznamy můžeme zakázat tak, že otevřeme záznam například pomocí tlačítka F5 a vlevo dole nastavíme ***„Zakázáno“***, viz Obrázek 234 - Detail definice pole v rychlém hledání datového modulu. Při následném vyhledávání bude toto pole v rychlém hledání vynecháno.

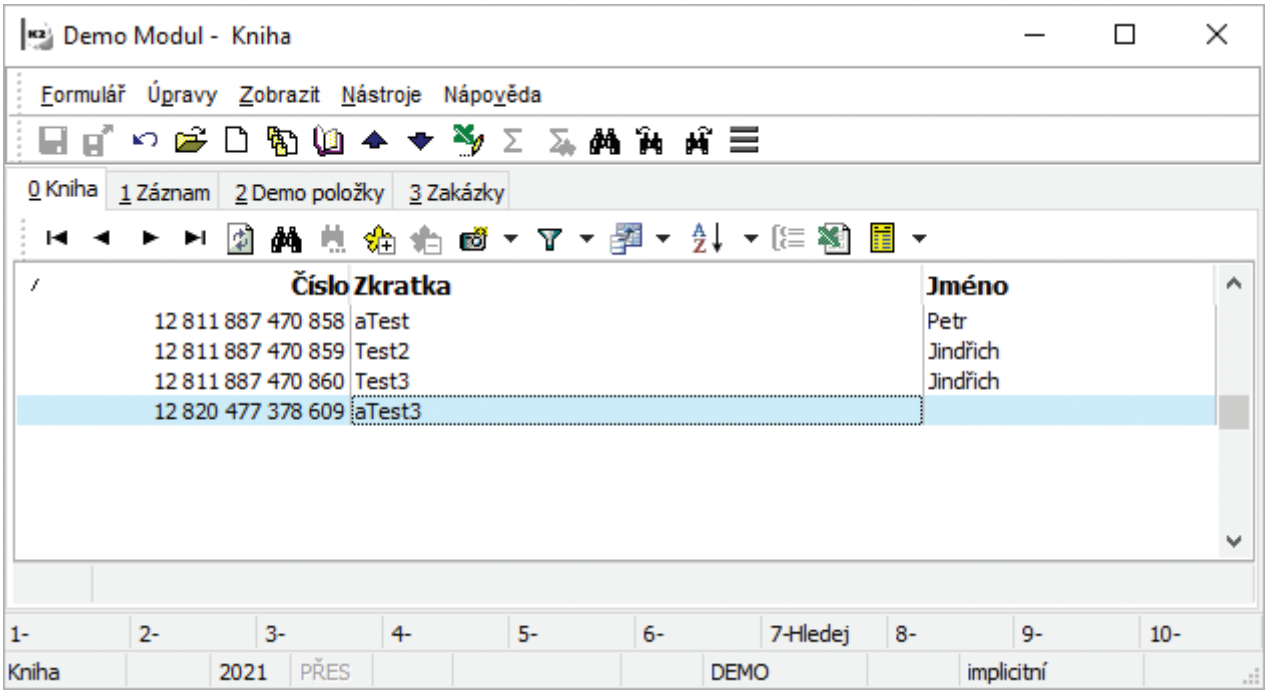


Obrázek 234 - Detail definice pole v rychlém hledání datového modulu

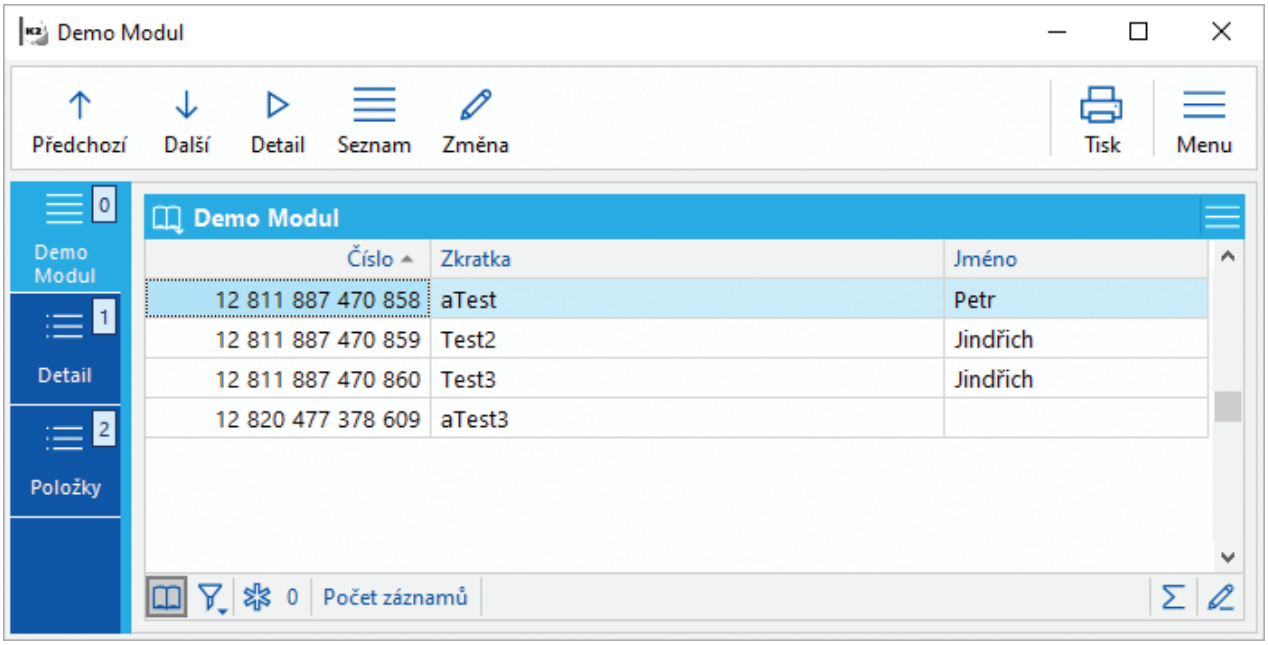
POZNÁMKA: Každé pole v systémové definici může být ***„Povoleno“***, ***„Zakázáno“*** a ***„Potlačeno“***. Poslední volba je určena pouze pro K2, která tento příznak sama řídí. Tento stav je nastaven, pokud nad polem, které je nastaveno v definici rychlého hledání, existuje fulltextový klíč. V tomto případě je rychlé vyhledávání automaticky potlačeno a používá se fulltext.

4.17. NÁHLED

Pro rychlé nahlédnutí na vytvořený datový modul můžeme použít funkci návrháře objektů – ***„Náhled“***. Funkce je dostupná pod tlačítkem ***„Náhled“***, které je vlevo v dolní části formuláře pro definici datového modulu, například na Obrázek 225 - API datového modulu. Návrhář objektů umožňuje otevřít náhled v klasických nebo univerzálních formulářích. Výběr provedeme šipkou vedle tlačítka ***„Náhled“***. Implicitní náhled je nastaven na klasické formuláře. Po stisknutí tlačítka se zobrazí formulář dle výběru typu, který zobrazí na straně ***„Kniha“*** seznam záznamů s definovanými implicitními sloupci. Strana ***„Záznam“*** pak zobrazuje detail vybraného záznamu. Pro každý modul, který je potomkem ***„TChildDataM“***, tedy pro každý položkový modul, který existuje v aktuálním datovém modulu, se zobrazí nová záložka se seznamem záznamů v tomto modulu. Na Obrázek 235 - Náhled datového modulu klasickým formulářem je vidět náhled v podobě generovaného klasického formuláře nebo Obrázek 236 - Náhled datového modulu univerzálním formulářem.



Obrázek 235 - Náhled datového modulu klasickým formulářem



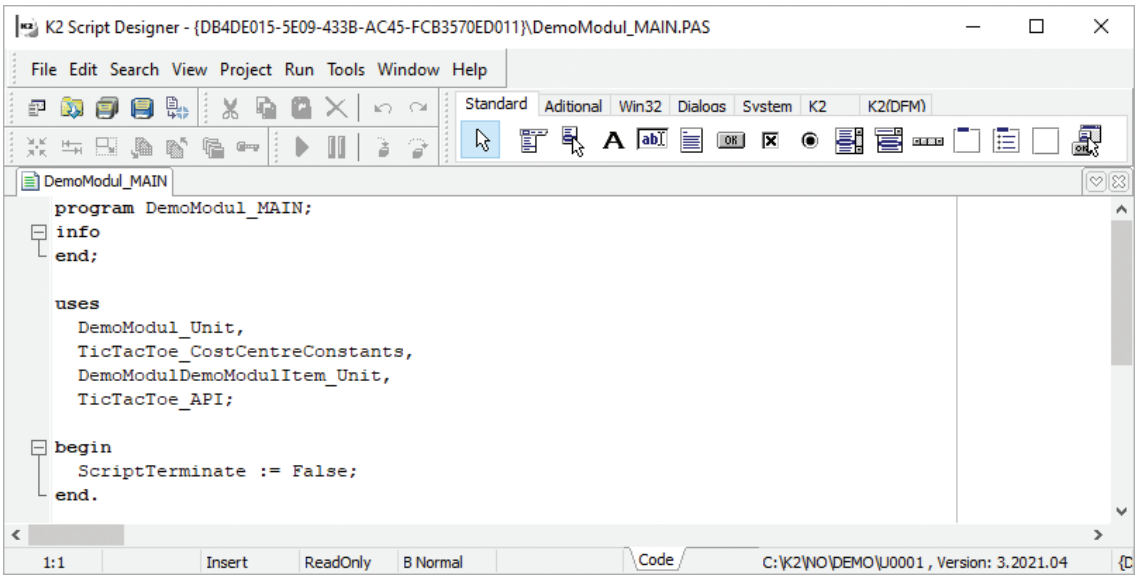
Obrázek 236 - Náhled datového modulu univerzálním formulářem

4.18. EDITOR

V předchozím textu byla několikrát zmínka o tzv. editoru skriptu pro návrhář objektů. Jedná se o velmi užitečný nástroj, pomocí kterého můžete implementovat skriptové části návrhu datových modulů v návrháři objektů. Těmi jsou ***„akce“***, ***„metody“***, ale také chování polí, jako ***„zobrazovat pole jako ikonu“***, ***„počítaná pole“***, ***„reakce na změnu“*** apod. V následujícím textu si popíšeme použití editoru, a to jak pro implementaci, tak pro ladění skriptů.

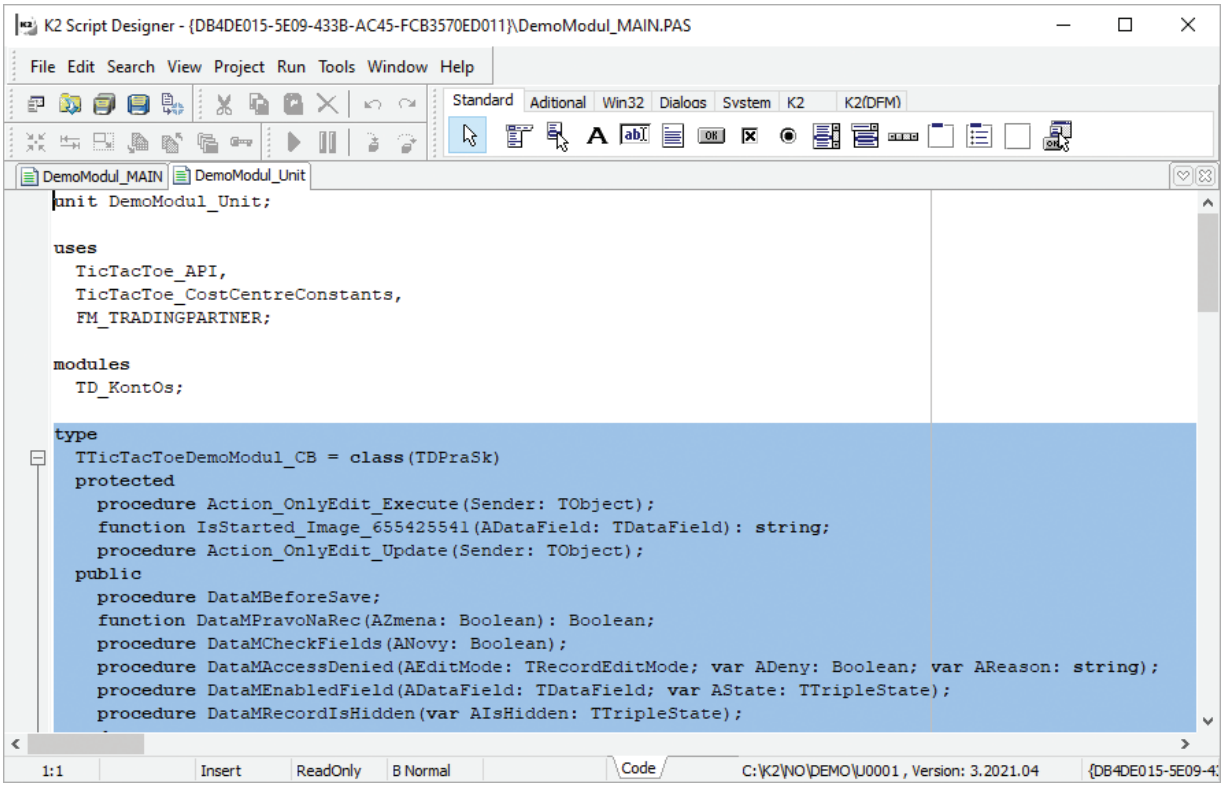
Po stisknutí tlačítka Editor se spustí vývojové prostředí pro K2 skript, viz Obrázek 237 - Editor skriptu – unita pro program. V sekci ***„uses“*** jsou pak vidět všechny připojené (přilinkované) skriptové ***„unity“***, které

jsou také generovány návrhářem objektů. Na obrázku vidíme v této sekci připojenou unitu ***„DemoModul_Unit“***, která odpovídá skriptové unitě pro datový modul ***„Demo“***, dále ***„DemoModulDemoModulItem_Unit“*** což je skriptová unita vygenerovaná pro položky, které máme nadefinovány v datovém modulu ***„Demo“***. Dále je zde připojená naše jednotka ***„TicTacToe_CostCentreConstants“***. Nakonec je připojena unita pro ***„API“***, které je vygenerované pro všechny datové moduly vytvořené v definici ***„TicTacToe“***.



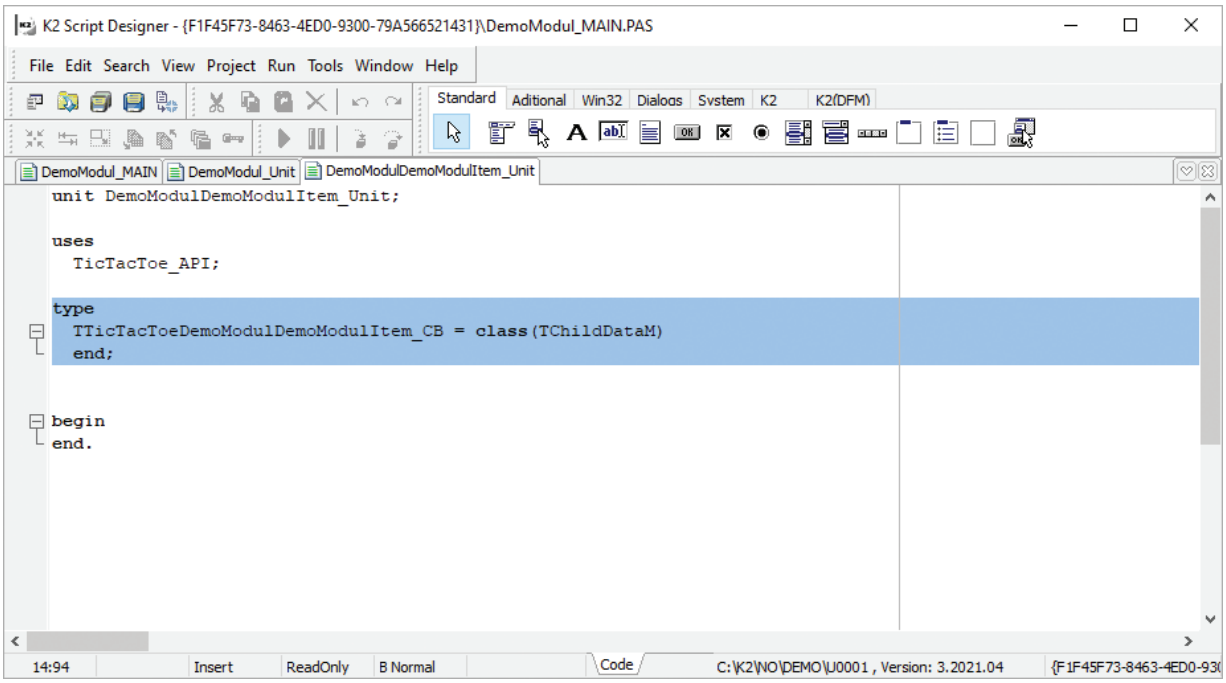
Obrázek 237 - Editor skriptu – unita pro program

Zajímavější částí jsou pak výše uvedené unity, pro datové moduly. Tedy například unita ***„DemoModul_Unit“***, která je vygenerovaná pro datový modul ***„DemoModul“*** a obsahuje tak všechny jeho akce, metody a vlastnosti, které jsme implementovali. Více je vidět na Obrázek 238 - Editor skriptu - unita pro datový modul.



Obrázek 238 - Editor skriptu - unita pro datový modul

Na Obrázek 239 - Editor skriptu – unita pro položkový datový modul pak můžete vidět vygenerovanou unitu pro položkový datový modul ***„DemoModuilItem“***, který patří výše zmíněnému datovému modulu ***„DemoModul“***. Ve skriptu je například vidět, že datový modul je potomkem třídy ***„TChildDataM“***.



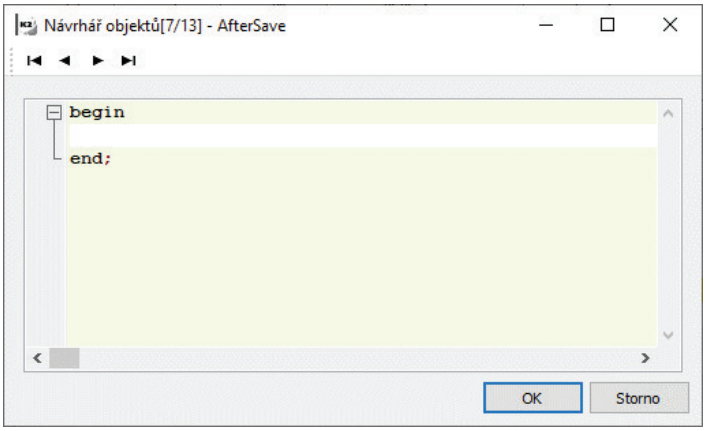
Obrázek 239 - Editor skriptu – unita pro položkový datový modul

Editor neslouží pouze k prohlížení vygenerovaného skriptu, ale hlavně je určen k jeho implementaci a ladění. V následující části si ukážeme, jak se pomocí editoru vytvářejí metody, akce apod., včetně jejich implementace.

V podstatě jsou dva postupy, jak můžete editor použít. Ukážeme si na příkladu implementace metody datového modulu ***„AfterSave“***.

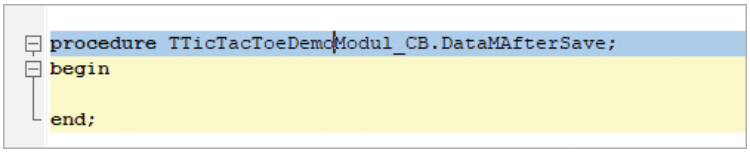
4.18.1. POUŽITÍ Č. 1 - GENEROVÁNÍ HLAVIČEK V NO A NÁSLEDNÁ IMPLEMENTACE V EDITORU

První variantou, je nástin implementace v návrháři objektů, v části, kde se tělo implementuje. Nástinem máme na mysli alespoň začátek a konec procedury či funkce pomocí klíčových slov ***„begin“*** a ***„end;“***, tak jak je vidět na Obrázek 240 - Ukázka implementace těla metody.



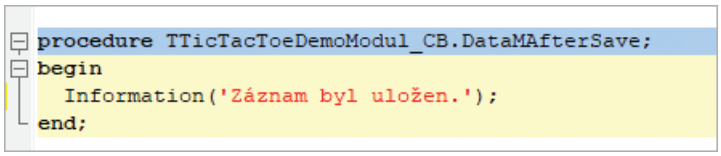
Obrázek 240 - Ukázka implementace těla metody

Následně se přepneme do editoru, kde uvidíme, že máme vygenerovánu proceduru ***„TTictacToeDemo_CB.DataMAfterSave“***, viz Obrázek 241 - Ukázka procedury "AfterSave" ve skriptu. Název procedury je podbarven modrou barvou, což jsou části skriptu, které jsou generované návrhářem objektů a nelze je editovat. Naopak žlutou barvou je podbarvena část, která je plně v rukou uživatele, který s návrhářem pracuje. Procedura je návrhářem vygenerována v momentě, kdy obsahuje alespoň nějaký text definovaný v sekci Obrázek 240 - Ukázka implementace těla metody.

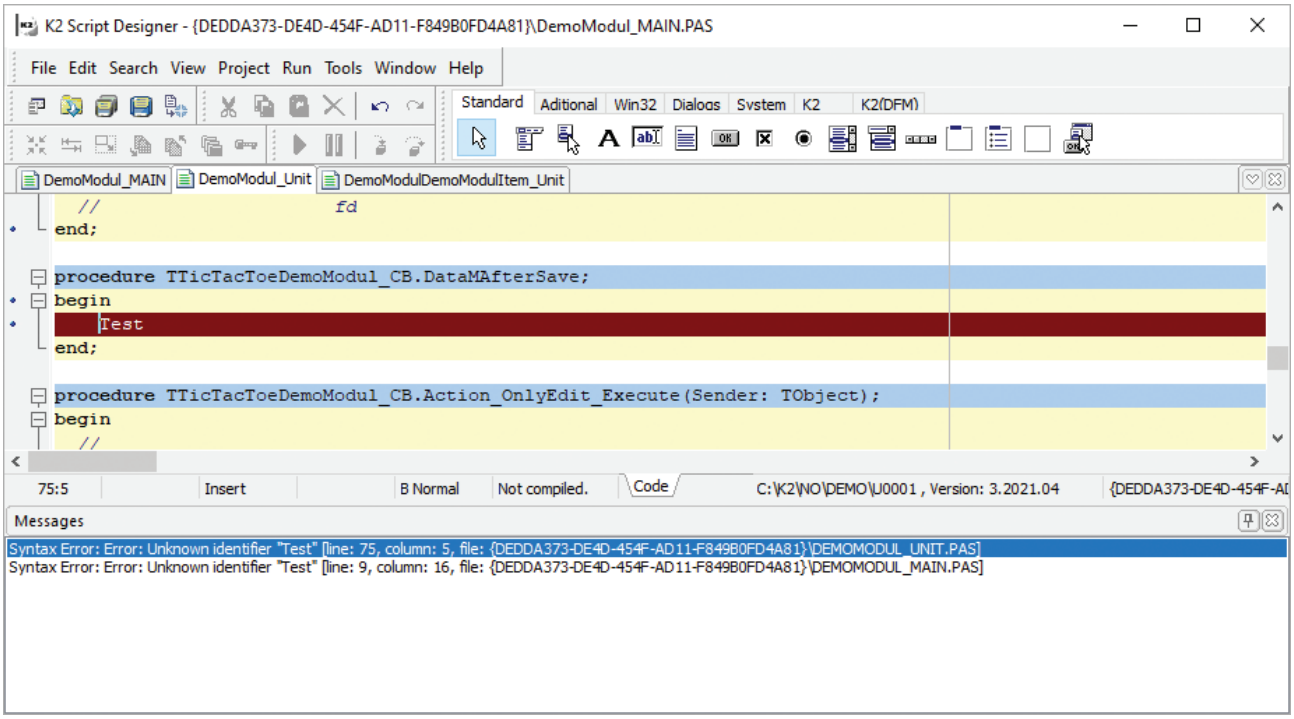


Obrázek 241 - Ukázka procedury "AfterSave" ve skriptu

Důležitou vlastností je, že skript, který implementujeme, si můžeme ihned přeložit a zjistit, zda je syntakticky správně. Překlad se provádí, tak jak jsme zvyklý z K2 skriptu, pomocí klávesové zkratky Ctrl + F9 nebo spuštěním překladu z menu. Když tedy ve skriptu budeme mít chybu, jako například na Obrázek 243 - Chyba při překladu v editoru skriptů, editor nás na tuto skutečnost upozorní. V opačném případě překlad projde a víme, že po syntaktické stránce máme vše správně. Implementace může vypadat třeba jako na Obrázek 242 - Ukázka implementace procedury "AfterSave" ve skriptu, kdy se uživateli zobrazí hlášení, po tom co dojde k uložení záznamu.



Obrázek 242 - Ukázka implementace procedury "AfterSave" ve skriptu



Obrázek 243 - Chyba při překladu v editoru skriptů

Tímto způsobem můžeme postupovat u všech implementací, včetně vlastností polí jako ***„zobrazovat jako ikonu“***, ***„reagovat na změnu“*** či ***„počítané pole“***.

4.18.2. POUŽITÍ Č. 2 - GENEROVÁNÍ HLAVIČEK A JEJICH IMPLEMENTACE PŘÍMO V EDITORU

V druhé variantě se oprostíme od zakládání těl implementací, tak jak jsme uváděli výše, například vkládáním klíčových slov ***„begin“*** a „end;“, ale vygenerujeme si hlavičky funkcí a procedur přímo v editoru, který nám právě tuto podporu nabízí.

Nabídku na vytvoření funkcí a procedur vyvoláme stisknutím pravého tlačítka myši přímo v oblasti skriptu. Zobrazí se nám kontextová nabídka, viz Obrázek 244 - Generování funkcí a procedur přímo v editoru skriptu přímo v editoru skriptu. V nabídce je vidět, že máme k dispozici sekce ***„Add Field“***, což je menu určené pro vytváření nových polí v návrháři objektů. Dále ***„Add Method“***, kde můžeme po rozbalení vidět všechny metody, které jsme si popsali v sekci 4.8. Metody. Dále pak „Add Property“ pro vkládání nových vlastností. Na závěr ***„Add Action“*** pro vkládání akcí (commandů). V následující části si popíšeme jednotlivé sekce blíže.

„Add Field“ – přidat pole
Při otevření této nabídky se zobrazí stejná nabídka jako na Obrázek 11 - Výběr typu nového pole v datovém modulu 2, kde si vybíráme, jaké pole budeme vkládat do datového modulu. Po výběru se pak již otevírá formulář, který jsme si také popsali v kapitole 4.2. Pole. V případě, že u pole bude vyžadována implementace nějaké vlastnosti, např. ***„zobrazit jako ikonu“***, musíme i zde, stejně jako v předchozím případě, napsat alespoň nějaký text do těla implementace, a po vytvoření pole se nám v editoru automaticky vygenerují funkce nebo procedury, které můžeme doimplementovat a ověřit jejich syntaxi.

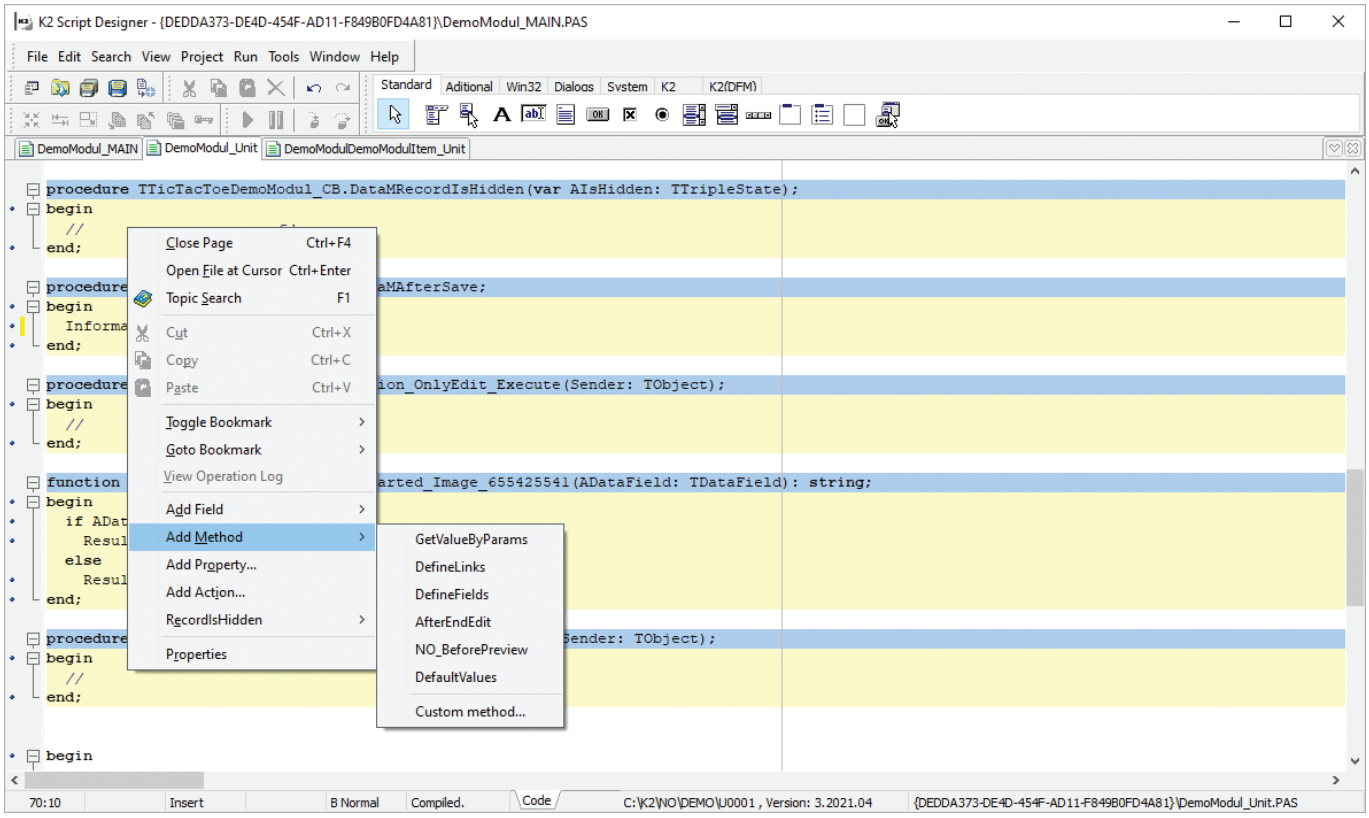
V tomto bodě nám tedy editor usnadní přechod mezi definicí polí a implementací jejich případných vlastností.

„Add Method“ – přidat metodu
Mnohem zajímavější a užitečnější je pak vkládání metod pomocí editoru. V případě, že vybereme nabídku ***„Add Method“*** můžeme vidět, v další nabídce, seznam metod, které můžeme vložit do skriptu. V nabídce se nacházejí pouze ty, které ještě nejsou implementovány. Nemůžeme tedy vložit některou metodu vícekrát. Po výběru se daná metoda vygeneruje ve skriptu i se začátkem ***„begin“*** a koncem metody ***„end;“***. Nám zbývá doplnit její implementaci a vyzkoušet překlad.

Po ukončení editoru můžeme v seznamu metod pro datový modul vidět, že metoda, kterou jsme implementovali je přepsána a v jejím těle můžeme vidět naší implementaci.

„Add Property“ – přidat vlastnost
Při otevření této nabídky se zobrazí stejný formulář jako na Obrázek 132 - Výběr typu vlastní property v datovém modulu, kde definujeme novou uživatelskou vlastnost, která vznikne v datovém modulu. Tato část je popsána v kapitole 4.6 Vlastní property.

„Add Action“ – přidat akci
Při otevření této nabídky se zobrazí stejný formulář jako na Obrázek 244 - Generování funkcí a procedur přímo v editoru skriptu, kde definujeme novou uživatelskou akci (command), která vznikne v datovém modulu. Tato část je popsána v kapitole 4.7. Akce.



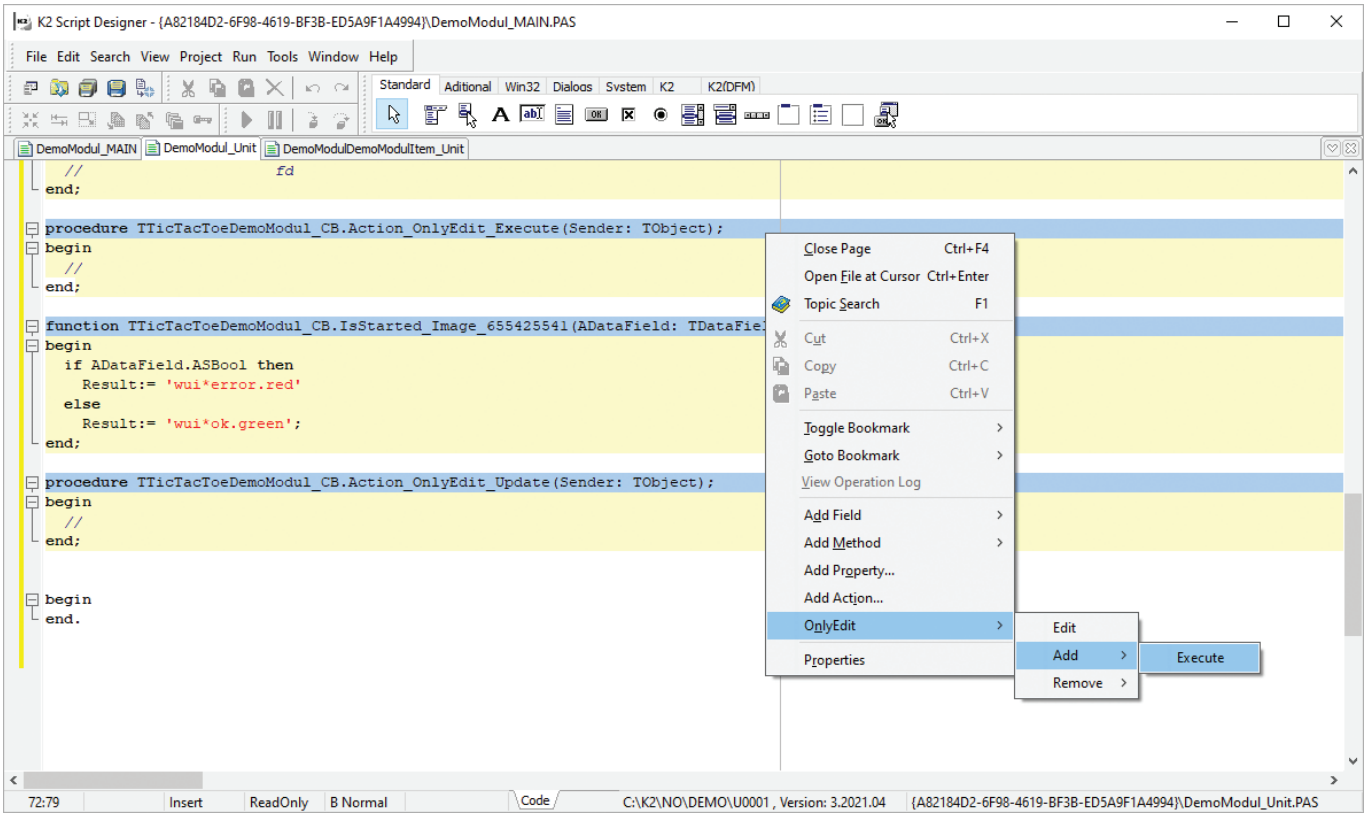
Obrázek 244 - Generování funkcí a procedur přímo v editoru skriptu

Práce s existujícími objekty

V případě, že stisknete pravé tlačítko myši na modré definici konkrétní funkce nebo procedury, zobrazí se v kontextové nabídce, kromě funkcí, které jsou popsány výše, navíc funkce, které jsou nějakým způsobem spjaty s vybranou procedurou nebo funkcí.

Například, když kliknete na proceduru ***„Execute“*** pro akci ***„OnlyEdit“***, zobrazí se v kontextové nabídce menu ***„OnlyEdit“***, kde je možné vyvolat editaci dané akce, pomocí stisku tlačítka Edit. Dále pak je v nabídce menu ***„Add“*** a ***„Remove“***. Pomocí ***„Remove“*** můžete odebírat procedury a funkce, které jsou zde implementovány. Pomocí ***„Add“*** přidáte další, které potřebujete. Například pro akci ***„OnlyEdit“*** máme implementovánu proceduru ***„Execute“***, což je výkonný kód akce. K akcím může navíc existovat procedura ***„Update“***. Protože ji nemáme implementovánu, je možné ji pomocí této nabídky vytvořit. Vše je vidět na Obrázek 245 - Editace vytvořených objektů pomocí editoru skriptu.

Stejná logika je pak i u polí a vlastností. Tedy můžete generovat funkce a procedury pro chování polí jako ***„Reagovat na změnu“***, ***„Počítané pole“***, ***„Zobrazit jako ikony“*** apod.



Obrázek 245 - Editace vytvořených objektů pomocí editoru skriptu

Druhá varianta se jeví více uživatelsky přijatelnější. Každý člověk má různé postupy v práci, tedy na každém pak záleží, kterou variantu bude používat. Troufám si říct, že většina uživatelů bude tyto dvě varianty přirozeně kombinovat v závislosti na typu rozšíření a momentu vytváření nebo rozšiřování objektů v návrháři objektů.

4.18.3. LADĚNÍ SKRIPTU POMOCÍ EDITORU

Abychom mohli začít ladit pomocí editoru skriptu, je potřeba nejprve vybrat datové moduly, které budou připraveny v režimu ladění. Výběr a samotné ladění si popíšeme v kapitole 13.2.5. Ladění.

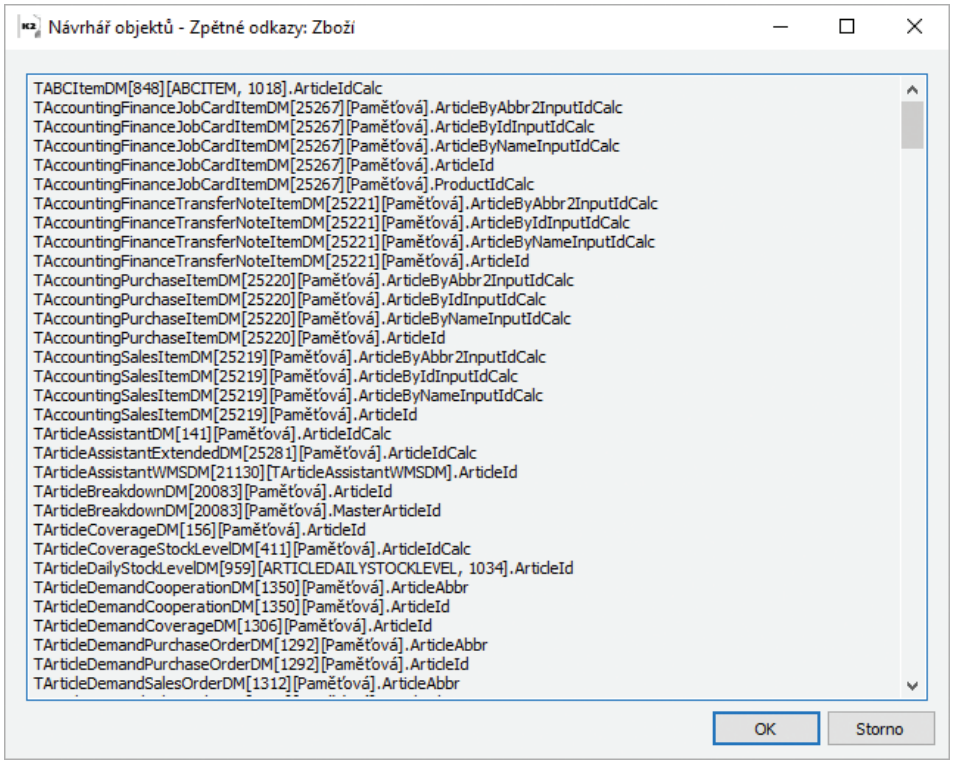
4.19. NÁSTROJE

Pod tímto tlačítkem nalezneme dostupné nástroje pro datový modul, které nám mohou usnadnit práci.

4.19.1. ZPĚTNÉ ODKAZY

Jedním z nástrojů je funkce ***„Zpětné odkazy“***, která slouží k získání seznamu všech datových modulů, ve kterých je aktuálně otevřený datový modul použitý jako cizí klíč, tedy ze kterých modulů existuje vazba do aktuálního modulu.

Po stisknutí tlačítka Zpětné odkazy se zobrazí seznam, viz Obrázek 246 - Výstup funkce "Zpětné odkazy".



Obrázek 246 - Výstup funkce "Zpětné odkazy"

4.19.2. OBNOVENÍ OBSAHU TABULKY

Dalším nástrojem je funkce ***„Obnovení obsahu tabulky“***, která obnovuje tabulku z DES. Po spuštění se nám zobrazí dialogové okno, kde vybereme soubor *.DES, který chceme obnovit.

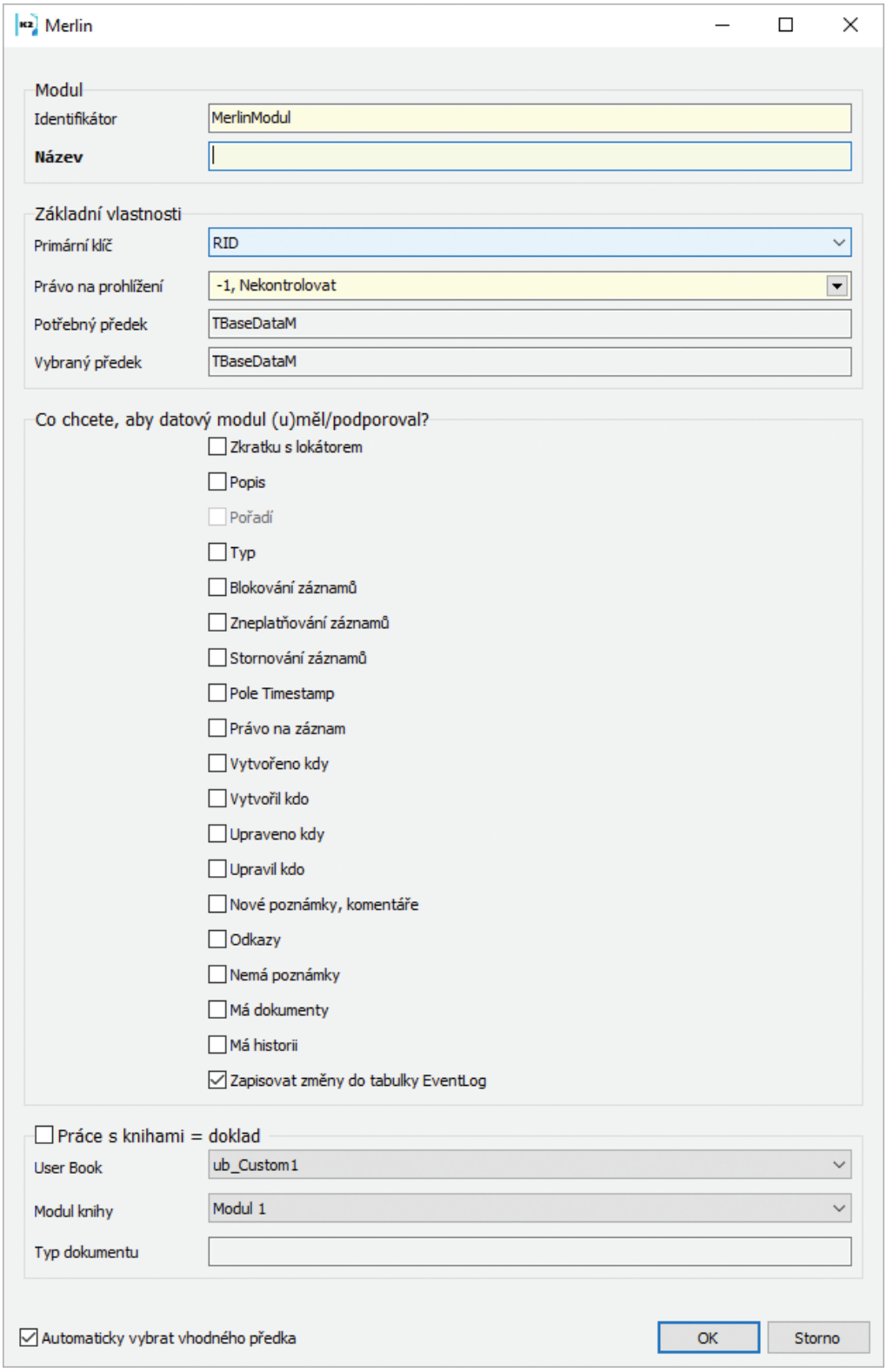
Daná funkce je dostupná až v momentě, kdy jsou načtené všechny DM – opticky to lze poznat podle počtu zpracovaných DM z celkového počtu. Funkci může použít uživatel, který má servisní právo, dále pokud je rozšíření nainstalováno - lze tedy vytvořit DM pracující s tabulkou, kterou je potřeba obnovit.

4.20. MERLIN

Jedná se o nástroj, který byl vytvořen pro vytvoření základní funkční kostry pro datový modul, během pár vteřin. V tomto nástroji lze vyplnit základní údaje pro datový modul, které si níže popíšeme, viz Obrázek 247 - Základní formulář „Merlina".

Merlina spouštíme pomocí Ctrl + M, nad datovým modulem.

POZNÁMKA: Pokud jednou již něco zaškrneme v ***„Merlinovi“***, odstranit některé věci musí již ručně, např. pokud zaškrneme ***„Zkratku s lokátorem“*** a potom odškrneme, musíme ručně úpravu vrátit zpátky, pokud jí nebudeme chtít. Proto se doporučuje vybírat jenom ty vlastnosti, které opravdu chceme.



Obrázek 247 - Základní formulář „Merlina"

Modul
Identifikátor – Jedná se o identifikátor datového modulu
Název – Jedná se o název datového modulu

Základní vlastnosti
Primární klíč – Zde si zvolíme, jaký chceme primární klíč. Doporučujeme použití RIDu.

Právo na prohlížení – Právo na prohlížení, vybrat můžeme ze standardních práv, nebo námi vytvořenými přes NO.

Potřebný předek – Automaticky vyhodnocená informace, jaký je potřebný předek, odvíjí se od vybraných vlastností.

Vybraný předek – Předek, kterého jsme zvolili na záložce ***„Modul"*** před spuštěním ***„Merlina"***. Může se automaticky změnit v závislosti na vybraných vlastnostech.

Co chcete, aby datový modul u(měl)/podporoval?
Některé vlastnosti se vzájemně mohou vylučovat, může se stát, že si zatrhnu ***„Zněplatňování záznamů"*** a tím pádem mi nepůjde zatrhnout vlastnost ***„Typ"***, protože vlastnost ***„Typ"*** je v předkovi TTypeDataM, který nepodporuje zněplatňování.

Zkratka s lokátorem – Vytvoří se pole ***„Abbr"*** (zkratka) s vlastnostmi ***„Povinné"***, ***„Unikátní"***, ***„Lokátor"***. Nastaví se i do zděděných property ***„ModuleAbbrNo".***

Popis – Vytvoří pole ***„Description"*** (popis). Nastaví se do zděděných property ***„ModuleDescrNo"***.

Pořadí – Daná vlastnost je přístupná pouze položkám. Po zaškrnutí se vytvoří pole ***„ItemNo"***, které se automaticky dosadí do zděděné property ***„ItemNoField"***.

Typ – Zda se jedná o typový číselník. Vytvoří pole ***„TypeId (typ)"***. Změní předka na ***„TTypeDataM"***. Do property ***„TypCis"*** doplní nově vzniklé pole ***„TypeId"***.

Blokování záznamů – Vytvoří pole ***„IsBlockedRecords"*** (blokováno) a nastaví ho do property ***„BlockedFieldNo"***.

Zneplatňování záznamů – Vytvoří pole ***„IsInvalidRecord"*** (zněplatněno), změní předka na ***„TCustomListDataM"*** a nastaví property ***„DisableFieldNo"***.

Stornování záznamů – Vytvoří pole ***„IsCanceledRecord"*** (stornováno) a nastaví ho do property ***„IsCancelledFieldNo"***.

Pole Timestamp – Vytvoří pole ***„Timestamp"***.

Právo na záznam – Vytvoří pole ***„RightGroupId"*** a nastaví ho do property ***„C_Prask"***.

Vytvořeno kdy – Vytvoří pole ***„CreatedOn"*** (vytvořeno) a nastaví ho do property ***„CreatedFieldNo"***.

Vytvořil Kdo – Vytvoří pole ***„CreatedBy"*** (vytvořil) a nastaví ho do property ***„CreatedByFieldNo"***.

Upraveno kdy – Vytvoří pole ***„UpdatedOn"*** (změněno) a nastaví ho do property ***„UpdatedFieldNo"***.

Upravil kdo – Vytvoří pole ***„UpdatedBy"*** (změnil) do uživatelů a nastaví ho do property ***„UpdatedByFieldNo"***.

Nové poznámky, komentáře – Nastaví property ***„CommentEnadbled"*** na hodnotu ***„True"***.

Odkazy – Nastaví property ***„LinksEnabled"*** na hodnotu ***„True"***.

Nemá poznámky – Změní předka na ***„TCustomDataM"*** a nastaví property ***„NemaPoznamky"*** na hodnotu ***„True"***.

Má dokumenty – Změní předka na „TCustomDataM" a nastaví property „MaDokumenty" na hodnotu „True".

Má historii – Nastaví předka na ***„TCustomDataM"*** a nastaví property ***„MaHistorii"*** na hodnotu ***„True"***.

Zapisovat změny do tabulky EventLog – Nastaví property ***„WriteIntoZMENY"*** na hodnotu ***„True"***.

Práce s knihami = doklad
Aby se daná část mohla měnit, nesmíme mít zatrženou volbu ***„Zneplatňování záznamů“***. Zatržením této vlastnosti se nám změní předek na ***„TCustomDocumentDataM“*** a zpřístupní níže uvedené vlastnosti, které jsou povinné pro předka ***„TCustomDocumentDataM“***. Dále se vytvoří pole ***„BookId“***, ***„BusinessYearId“***, ***„Number“***, z těchto polí se také vytvoří klíč a nastaví se property ***„HlRada“***, ***„BusinessYearFieldNumber“***, ***„HlCi“*** a ***„IndexByBook“***.
User Book – Nastaví se dle výběru property ***„User Book“***.

Modul knihy – Nastaví se dle výběru property ***„BookModule“***.

Typ dokumentu – nastaví property ***„BookDocumentTypeID“***.

Automaticky vybrat vhodného předka
Zdá se má automaticky vybrat nejvhodnější předek pomocí zatržených vlastností.

5. ROZŠIŘUJEME STANDARDNÍ DATOVÝ MODUL

V předchozí kapitole jsme si popsali vytvoření nového datového modulu. Návrhář objektů umožňuje provádět i rozšiřování standardních datových modulů, které jsou dodávány v IS K2. V následujícím textu si popíšeme možnosti rozšíření standardních datových modulů.

5.1. POLOŽKY VLASTNĚNÉ

Stejně jako se přidávají nové položky (podřízené datové moduly) do vlastních datových modulů, tak se mohou přidávat podřízené moduly do standardních modulů K2. Tedy můžeme si například přidat nový podřízený modul do modulu ***„Zboží“***. Princip rozšiřování je úplně stejný jako při práci s novým datovým modulem. Vše je tedy popsáno v kapitole 4.2.5. Položky.

5.2. POLOŽKY NEVLASTNĚNÉ

Stejně jako se přidávají nevlastněné položky do vlastních datových modulů, tak se mohou přidávat nevlastněné moduly do standardních modulů K2. Tedy můžeme si například přidat nový podřízený modul jako nevlastněné položky do modulu ***„Zboží“***. Princip rozšiřování je úplně stejný jako při práci s novým datovým modulem. Vše je tedy popsáno v kapitole 4.2.6. Nevlastněné položky.

5.3. FYZICKÁ POLE

Tento druh polí zatím není podporován nad standardními datovými moduly.

5.4. POČÍTANÁ POLE

Stejně jako se přidávají nová počítaná pole (pole, které má příznak ***„Počítané pole“***) do vlastních datových modulů, tak se mohou přidávat i do standardních modulů K2. Tedy můžeme si například přidat nové počítané pole do modulu ***„Zboží“***. Princip rozšiřování je úplně stejný jako při práci s novým datovým modulem. Vše je tedy popsáno v kapitole 4.2. Pole.

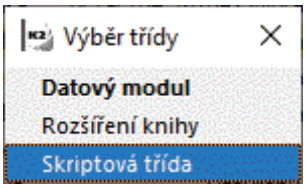
5.5. REGISTROVANÉ FUNKCE

Stejně jako se implementují registrované funkce do vlastních datových modulů, tak se mohou implementovat registrované funkce ve standardních modulech K2. Princip implementace je stejný, jako je popsáno v kapitole 4.10. Registrované funkce.

6. SKRIPTOVÁ TŘÍDA

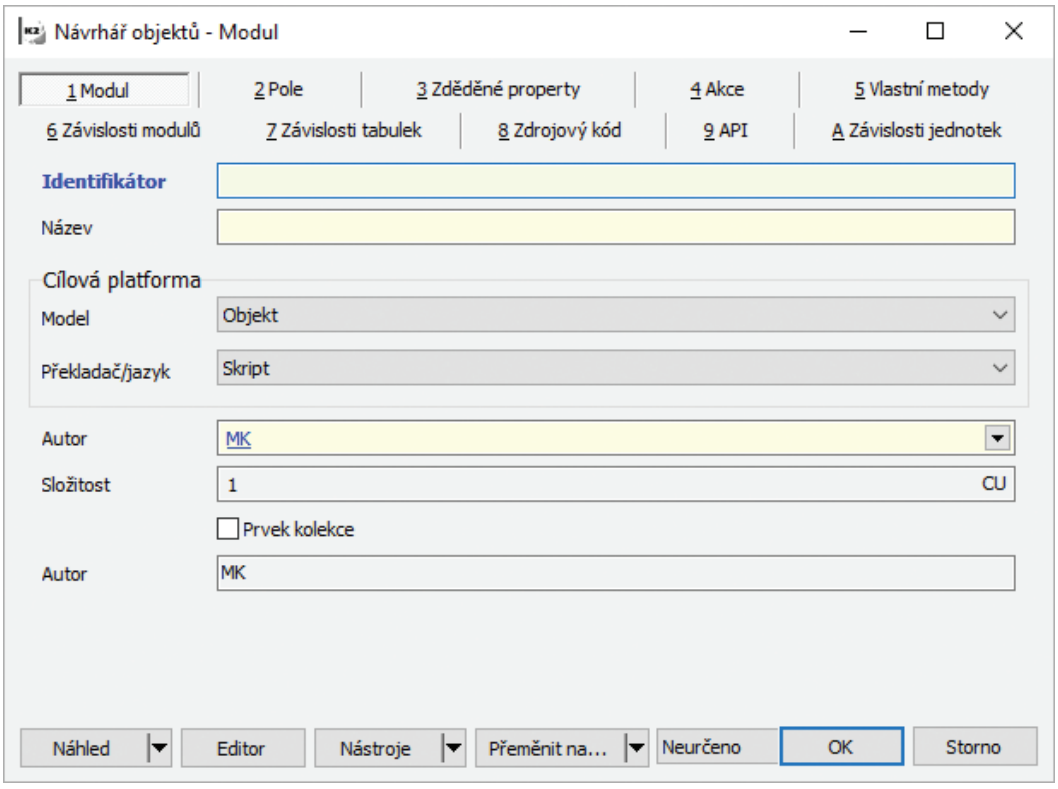
Dalším typem objektu, který můžeme v návrháři objektů vytvořit, je ***„skriptová třída"***. Tento typ struktury slouží k rozšíření K2 o objekty, pomocí nichž můžeme řešit jednodušší struktury, například vstupní hodnoty pro operace s různými výpočty, plnění dat apod.

Novou skriptovou třídu založíme pomocí stisknutá tlačítka Insert nebo tlačítkem nad seznamem modulů na úvodní obrazovce návrháře objektů. Zobrazí se formulář, viz Obrázek 248 - Založení nové skriptové třídy.



Obrázek 248 - Založení nové skriptové třídy

Otevře se nám formulář pro definici nové skriptové třídy, jak můžeme vidět na Obrázek 249 - Definice nové skriptové třídy. V následujícím textu si popíšeme jednotlivé části. Ty, které jsme popsali dříve, vynecháme.



Obrázek 249 - Definice nové skriptové třídy

6.1. NÁHLED

Proto, abychom mohli zkoušet funkčnost a vzhled vytvářené třídy, existuje na formuláři tlačítko Náhled, pomocí kterého vyvoláme formulář pro definovaný objekt. Na tomto formuláři si můžeme vyzkoušet funkčnost nového objektu. Tlačítko je k dispozici ve formuláři v definici skriptové třídy viz Obrázek 249 - Definice nové skriptové třídy.

Po jeho stisknutí se vygeneruje soubor se zdrojovým kódem, tedy s K2 skriptem. Přípona souboru je *„pas“* a soubor nachází se v adresáři *„\CustomModule\Preview“*. Název je ve tvaru *„Module_Identifikátor třídy.pas“* jak je uvedeno v kapitole 6.2. Modul.

Dále se vygenerovaný skript spustí přímo v editoru skriptu K2 v režimu ladění. Tedy zastaví se na začátku vykonávaného skriptu. V případě, že nebudeme ladit, můžeme skript posunout tlačítkem Run nebo zkratkovou klávesou F9. Poté se zobrazí generovaný formulář pro daný objekt, ve kterém si můžeme vyzkoušet funkčnost.

POZNÁMKA: Data, která vložíme pomocí tohoto formuláře, jsou pro potřeby ladění ukládána do souboru ve stejném adresáři, jako v případě skriptu pro třídu. Soubor se jmenuje stejně jako skript, ale je jiného typu - *„xml“*.

6.2. MODUL

Záložka modul, obsahuje základní nastavení nově vytvářené skriptové třídy.

Identifikátor
Zde definujeme název skriptové třídy, který je pak použitý v generovaném zdrojovém kódu a to ve tvaru *„T“* jako prefix, následuje uvedený identifikátor a nakonec je doplněn sufix *„Object“*. Tedy v případě, že definujeme objekt s identifikátorem *„Demo“*, vznikne třída ve zdrojovém kódu s názvem *„TDemoObject“*. Identifikátor je použitý i v názvu unity K2 skriptu. Vždy se k identifikátoru vloží prefix ve tvaru *„Module_“*. V našem příkladu *„Module_Demo“*.

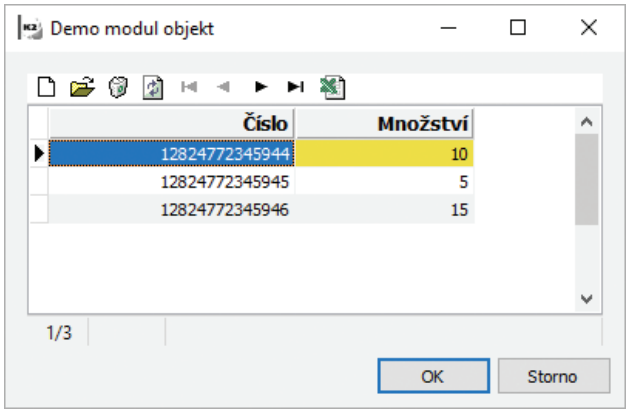
Název
Uvedený text je použitý v záhlaví formuláře, který následně pracuje se skriptovou třídou jako vstupní rozhraní. Je tedy použit jako tzv. *„Caption“* a také k popisu v případě dalšího zpracování.

Model
Tato hodnota je nastavena na *„Objekt“* a nelze ji dále měnit. Definuje typ modelu.

Překladač/jazyk
Tato hodnota je nastavena na *„Skript“* a nelze ji dále měnit. Definuje typ zdrojového kódu, který je návrhářem objektů generován právě v této syntaxi.

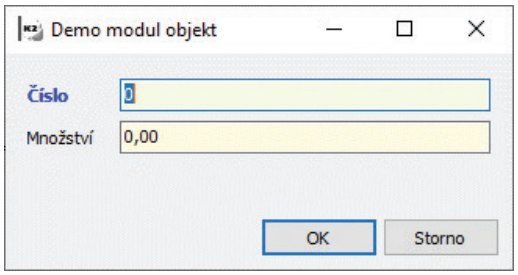
Prvek kolekce
Tato vlastnost definuje, zda se jedná o objekt typu *„záznam“* nebo *„kolekci záznamů“*.

POZNÁMKA: V případě nastavení hodnoty *„Prvek kolekce“* se bude vytvořený objekt prezentovat jako seznam prvků objektů právě definovaných. V opačném případě se bude jednat pouze i jeden záznam. Například objekt se dvěma poli *„Číslo“* a *„Množství“* se v případě kolekce zobrazí jako seznam, viz Obrázek 262 - Použití sufix vlastnosti na poli skriptové třídy.



Obrázek 250 - Skriptová třída jako prvek kolekce

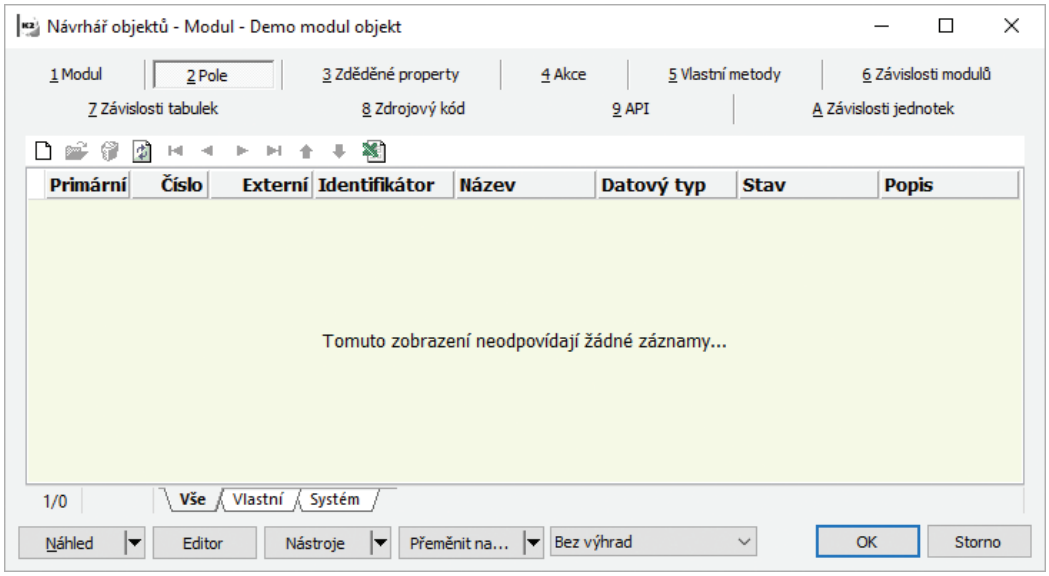
V případě zrušení tohoto příznaků pak jako záznam, viz Obrázek 251 - Skriptová třída jako záznam.



Obrázek 251 - Skriptová třída jako záznam

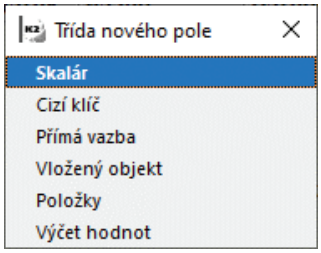
6.3. POLE

Na této záložce nadefinujeme, stejně jako u datového modulu, pole, která budou v objektu k dispozici, viz Obrázek Obrázek 252 - Záložka "Pole" ve skriptové třídě.



Obrázek 252 - Záložka "Pole" ve skriptové třídě

Nové pole založíme stisknutím klávesy Insert nebo tlačítka Nový nad seznamem polí. Na výběr máme k dispozici stejné možnosti jako u datového modulu. Navíc je zde možné vložit vnořený objekt. Celý seznam dostupných typů polí je vidět na Obrázek 253 - Výběr typu pole pro skriptový objekt.



Obrázek 253 - Výběr typu pole pro skriptový objekt

Všechny typy polí obsahují stejná nastavení jako v případě datového modulu. Navíc se zde nachází nové vlastnosti, které budou popsány níže.

Display
Pomocí této vlastnosti můžeme nadefinovat styl zobrazení pole na formuláři.

Na výběr jsou následující hodnoty:

pdsDefault – pole je na formuláři zobrazeno

pdsNone – pole se na formuláři vůbec nezobrazí

pdsInspect – určuje, zda se pole má zobrazit ve stromovém menu ve formuláři. Tato varianta je použitelná pouze pro pole typu ***„Položky“***

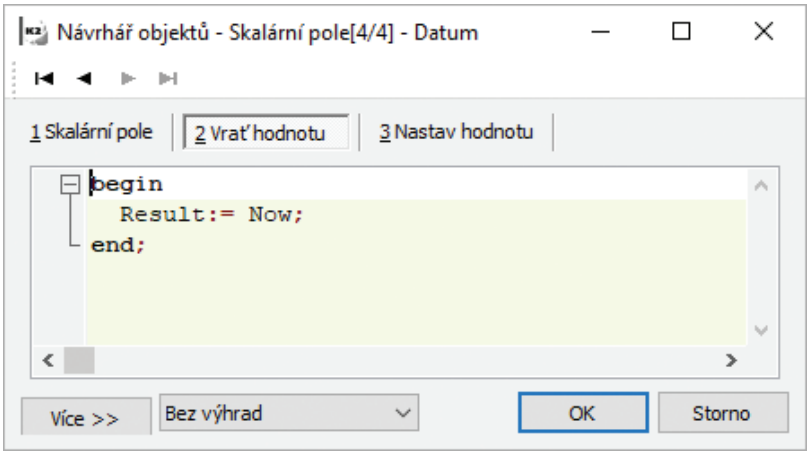
pdsInline – pole je na formuláři zobrazeno – jedná se o výchozí hodnotu, tedy = pdsDefault

pdsBlock – pole se zobrazí jako samostatná záložka

pdsFooter – pole se zobrazí v zápatí formuláře (je viditelné na všech případných záložkách)

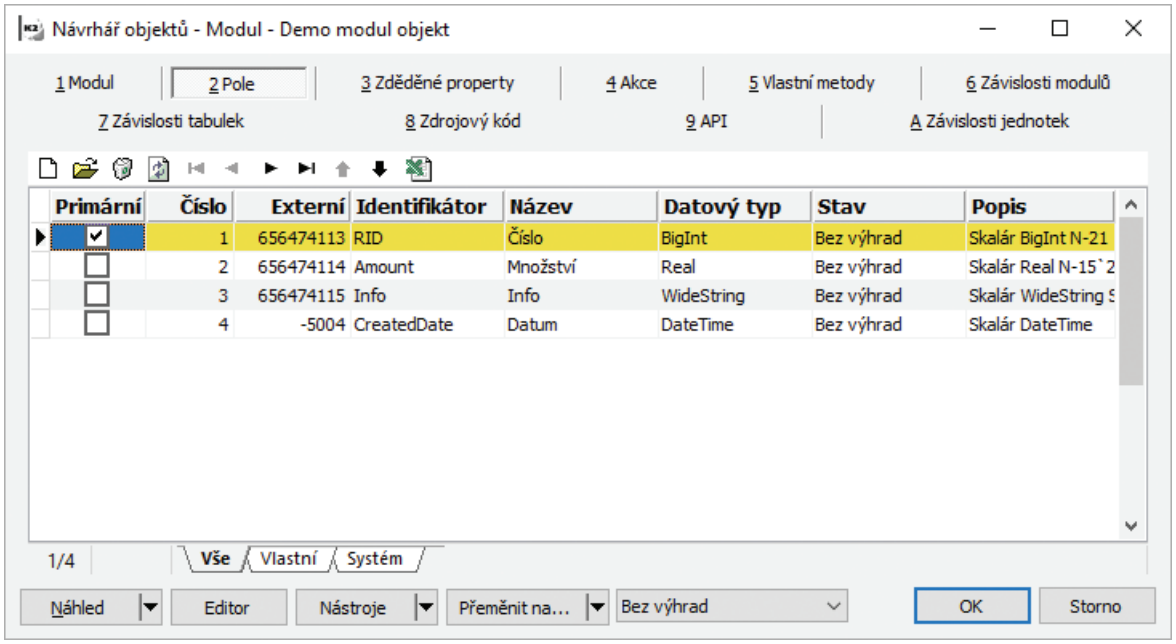
PŘÍKLAD: Na příkladu si ukážeme, jak vypadá styl zobrazení jednotlivých polí dle nastavení vlastnosti ***„Display“***.

Vytvoříme si objekt, který nazveme ***„DemoObject“*** a vložíme pole ***„RID“*** (automaticky se nám vyplní název ***„Číslo“***, dále pole ***„Amount“*** (množství). Tato pole necháme ve výchozím nastavení, co se týká vlastnosti ***„Display“***. Dále přidáme pole ***„Info“*** (informace) kde nastavíme ***„Display“*** jako ***„pdsBlock“***. Nakonec přidáme pole ***„CreatedDate“*** (datum), kde ještě můžeme pole označit jako počítané a do funkce, které představuje jeho návratovou hodnotu, implementovat skript, který zobrazovat aktuální datum v tomto poli, viz Obrázek 254 - Příklad na skriptový objekt - vlastnost display - CF pole. Toto pole ve vlastnosti ***„Display“*** nastavíme jako pdsFooter.



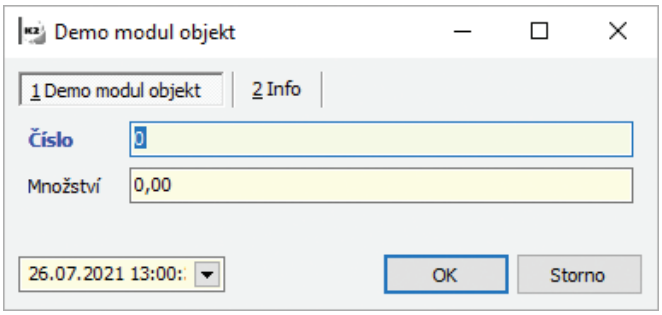
Obrázek 254 - Příklad na skriptový objekt - vlastnost display - CF pole

Souhrn všech polí můžeme vidět na obrázku Obrázek 255 - Příklad na skriptový objekt - vlastnost display – definice polí.



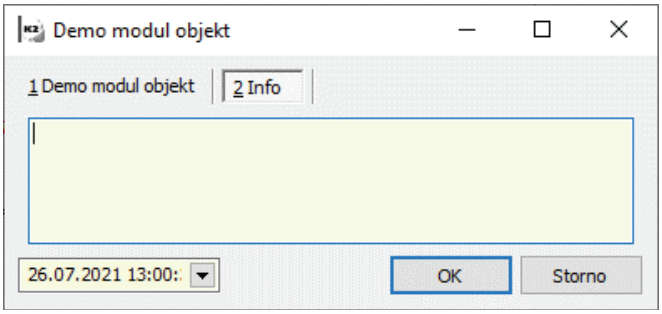
Obrázek 255 - Příklad na skriptový objekt - vlastnost display – definice polí

Následně pomocí stisknutí tlačítka Náhled můžeme vizuálně zobrazit formulář pro definovaný objekt. Na Obrázek 256 - Styly zobrazení pole objektu je vidět standardní zobrazení polí ***„Číslo“*** a ***„Množství“***. Pole ***„Datum“*** je zobrazeno v patičce formuláře a pole ***„Informace“*** je zobrazeno jako samostatná záložka. ***„Datum“*** máme přednastaven na aktuální datum a čas díky definici počítaného pole.



Obrázek 256 - Styly zobrazení pole objektu

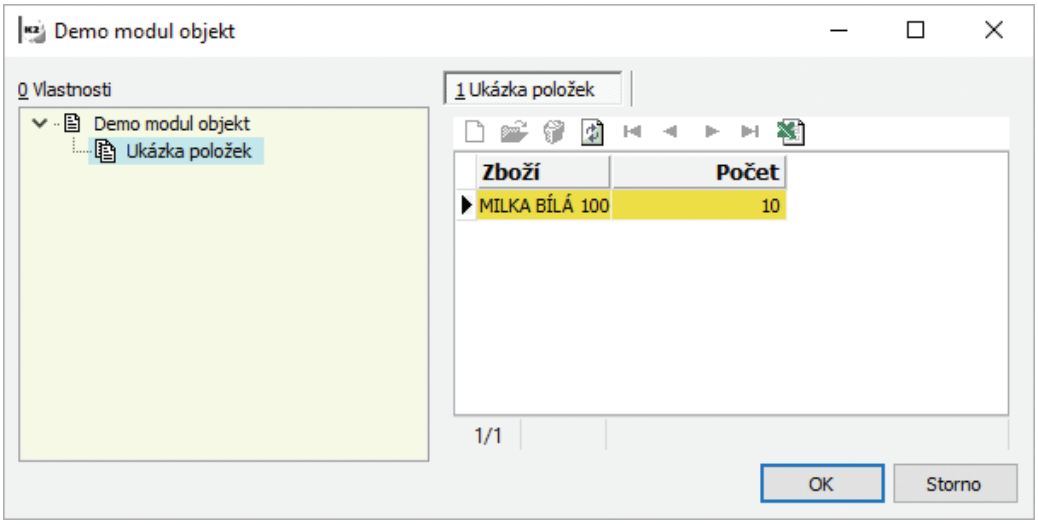
Při přepnutí na záložku pole ***„Informace“*** můžeme vidět, že pole je zobrazeno přes celý formulář. Navíc je zde vidět, že pole ***„Datum“***, které je v zápatí, se zobrazuje na každé záložce.



Obrázek 257 - Styly zobrazení pole objektu – detail

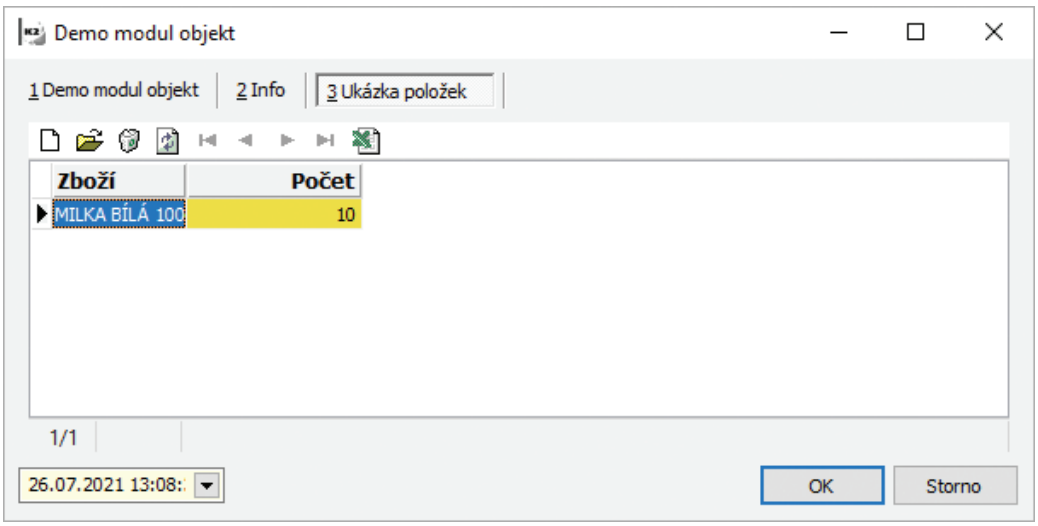
Tedy potřebujeme zadat vstupní data, přičemž popis dat je delšího charakteru, tak máme vytvořenou samostatnou záložku, a protože potřebujeme vědět, kdy byl objekt vytvořen, máme tuto informaci na každé straně formuláře zobrazenu.

PŘÍKLAD: Vytvořený příklad navíc rozšíříme o pole typu položky a ukážeme si tak možnost jejich zobrazení. Založíme tedy další pole typu ***„Položky“*** a pojmenujeme ho ***„DemoObjectItem“*** (ukázka položek). V položkách vytvoříme pole ***„ArticleId“*** (Zboží) a ***„Count“*** (počet kusů). Položkám nastavíme vlastnost ***„Display“*** na hodnotu ***„pdsInspect“*** což způsobí zobrazení položek ve stromovém menu objektu, viz Obrázek 258 - Položky skriptové třídy ve stromu.



Obrázek 258 - Položky skriptové třídy ve stromu

V případě, že bychom na poli nastavili ***„Display“*** na hodnotu ***„pdsBlock“***, vypadalo by zobrazení zase trochu jinak, viz Obrázek 259 - Položky skriptové třídy v bloku.



Obrázek 259 - Položky skriptové třídy v bloku

POZNÁMKA: Rozmístění polí na formuláře – styl zobrazení, je funkční pouze v klasických formulářích. V univerzálních formulářích není zatím podporováno.

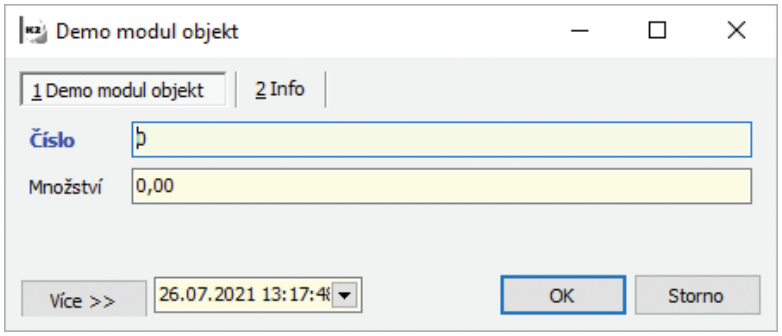
Úroveň
Pomocí této vlastnosti můžeme nastavit, zda se má pole zobrazit standardně na formuláři nebo až po otevření detailní sekce formuláře – tlačítko Více >> a << Méně.

Na výběr jsou následující hodnoty:

pulDefault – pole je zobrazeno standardně na formuláři

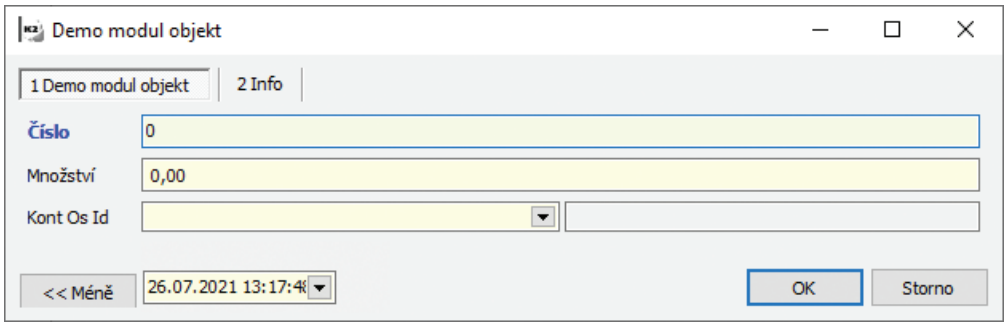
pulExpert – pole je zobrazeno až po stisku tlačítka Více >>

PŘÍKLAD: Nastavení si ukážeme na příkladu. Doplníme příklad z předchozího kroku o nové pole ***„Kontaktní osoba“***, které je cizím klíčem do kontaktních osob. U tohoto pole nastavíme ***„Úroveň“*** na hodnotu ***„pulExpert“***. Po otevření náhledu formuláře můžeme vidět na Obrázek 260 - Nastavení úrovně na poli objektu, přibylo na formuláři tlačítko Více >>.



Obrázek 260 - Nastavení úrovně na poli objektu

Po jeho stisknutí se zobrazí všechna pole definovaná jako ***„pulExpert“*** v ***„Úroveň“*** a tlačítko se přepne do stavu << Méně, viz Obrázek 261 - Nastavení úrovně na poli objektu - detail.

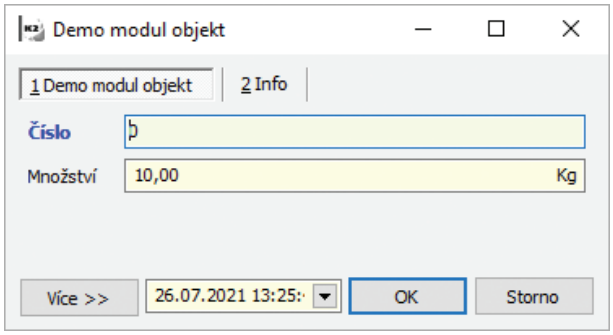


Obrázek 261 - Nastavení úrovně na poli objektu - detail

Prefix
Slouží k definici, co se má zobrazit ve formuláři jako předpona hodnoty pole. Můžeme definovat konstantu, pak u každé hodnoty bude jako předpona uvedena definovaná konstanta. Nebo můžeme uvést název vlastnosti datového modulu, ve tvaru „{Jméno vlastnosti}“. Pak se bude zobrazovat předpona, která odpovídá hodnotě vlastnosti na datovém modulu.

Sufix
Slouží k definici, co se má zobrazit ve formuláři za hodnotou pole. Můžeme definovat konstantu, pak za každou hodnotou bude uvedena definovaná konstanta. Nebo můžeme uvést název vlastnosti datového modulu, ve tvaru „{Jméno vlastnosti}“. Pak se bude zobrazovat hodnota, která odpovídá vlastnosti na datovém modulu.

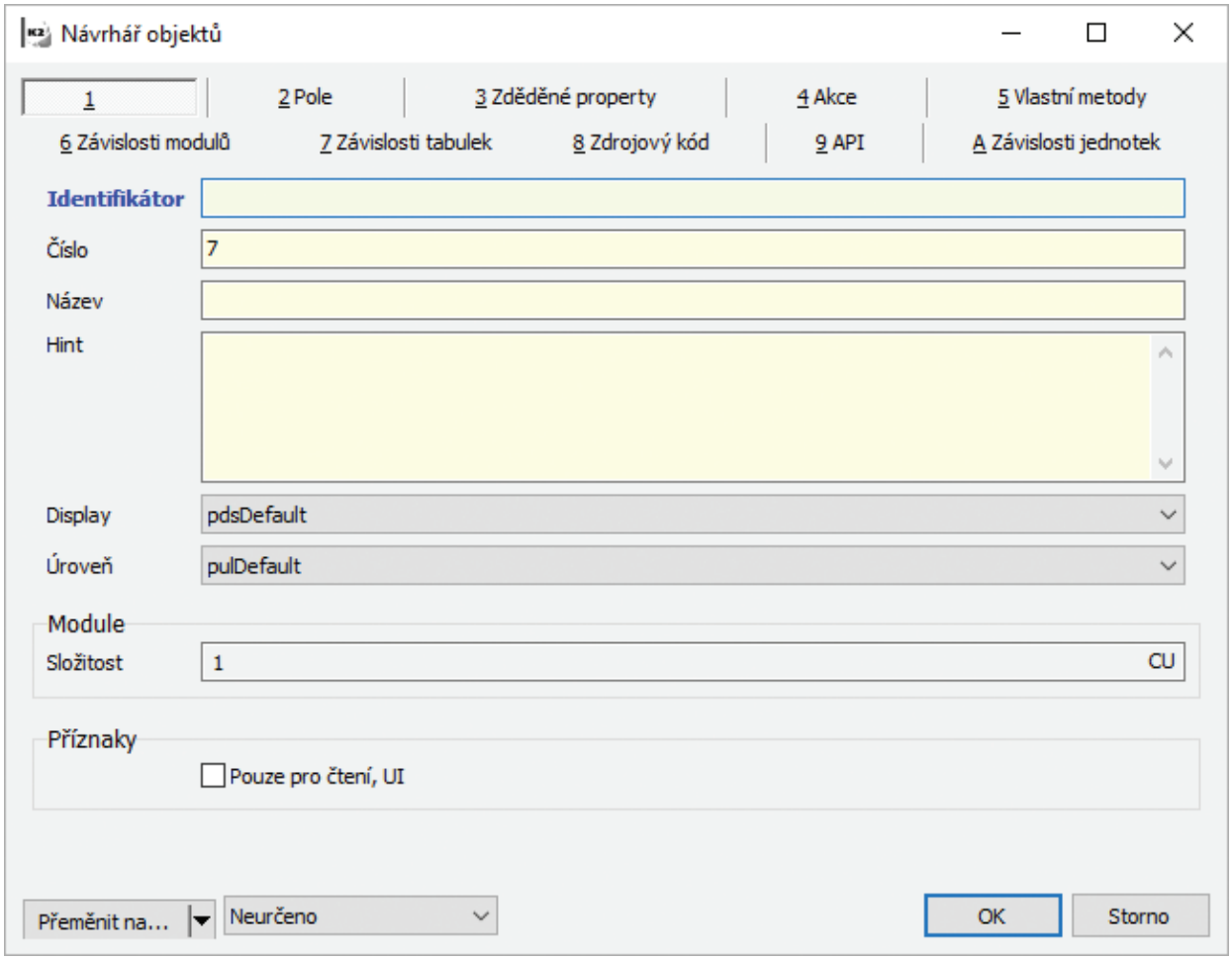
PŘÍKLAD: Použití si můžeme představit na příkladu, kdy zobrazujeme ve formuláři pole, které definuje množství zboží. Jako ***„sufix“*** můžeme uvést jednotku vybraného zboží. Pak se nám zobrazí pole množství, kde bude hodnota např. ***„10“***. Za textem se objeví jednotka, která zpřesňuje význam, např. ***„Kg“***. Nemusíme tedy mít dvě pole vedle sebe ve formuláři, ale nahradíme ho jedním. Viz Obrázek 262 - Použití sufix vlastnosti na poli skriptové třídy.



Obrázek 262 - Použití sufix vlastnosti na poli skriptové třídy

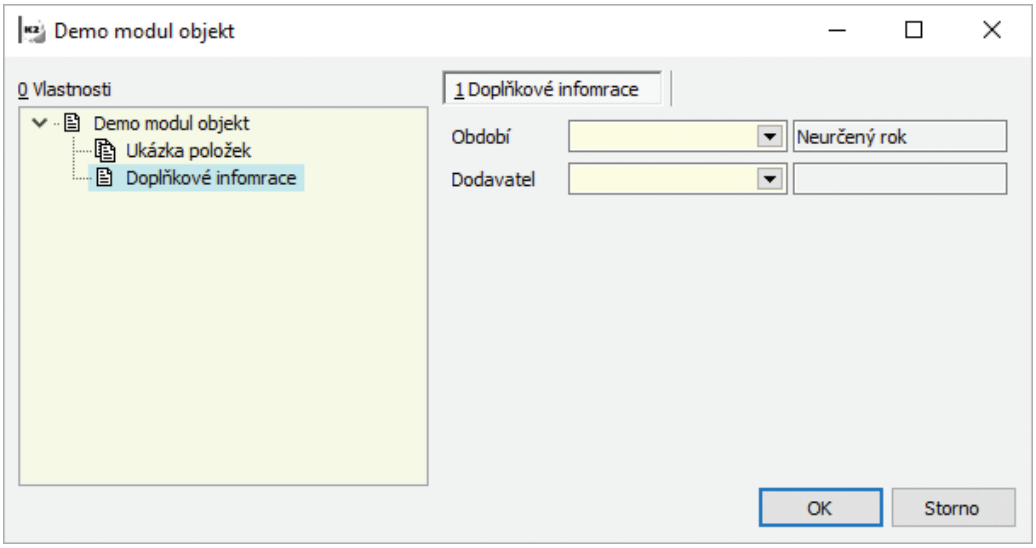
6.3.1. VLOŽENÝ OBJEKT

Ve skriptových třídách je možné nadefinovat typ pole ***„Vložený objekt“***. Jedná se o další objekt, který je zanořený do aktuálního objektu. Máme tedy pole typu třída. V případě výběru tohoto typu pole se zobrazí formulář pro vytvoření objektu. Analogicky funguje zakládání položek v rámci datového modulu. Na formuláři je navíc oproti vytvoření objektu na první úrovni, možnost nastavit objekt ***„Pouze pro čtení“*** a také změnit vlastnosti ***„Display“*** a ***„Úroveň“***, viz Obrázek 263 - Definice vloženého objektu.



Obrázek 263 - Definice vloženého objektu

PŘÍKLAD: Chování a zobrazení vnořeného objektu si ukážeme na příkladu. Do předchozího příkladu doplníme nové pole typu ***„Vnořený objekt“***, který nazveme ***„AdditionalInformation“*** (doplňkové informace). Do polí vložíme pole typu cizí klíč do modulu ***„Prefix“***, tedy (Období), dále pole typu cizí klíč do modulu ***„Dodavatele/odběratelé“*** (Dodavatel). Novému objektu nastavíme vlastnost ***„Display“*** na hodnotu ***„pdsInspect“***. Po otevření objektu v náhledu můžeme vidět dostupný zanořený objekt ve stromu v objektu, viz Obrázek 264 - Vnořený objekt – příklad.



Obrázek 264 - Vnořený objekt – příklad

6.4. ZDĚDĚNÉ PROPERTY

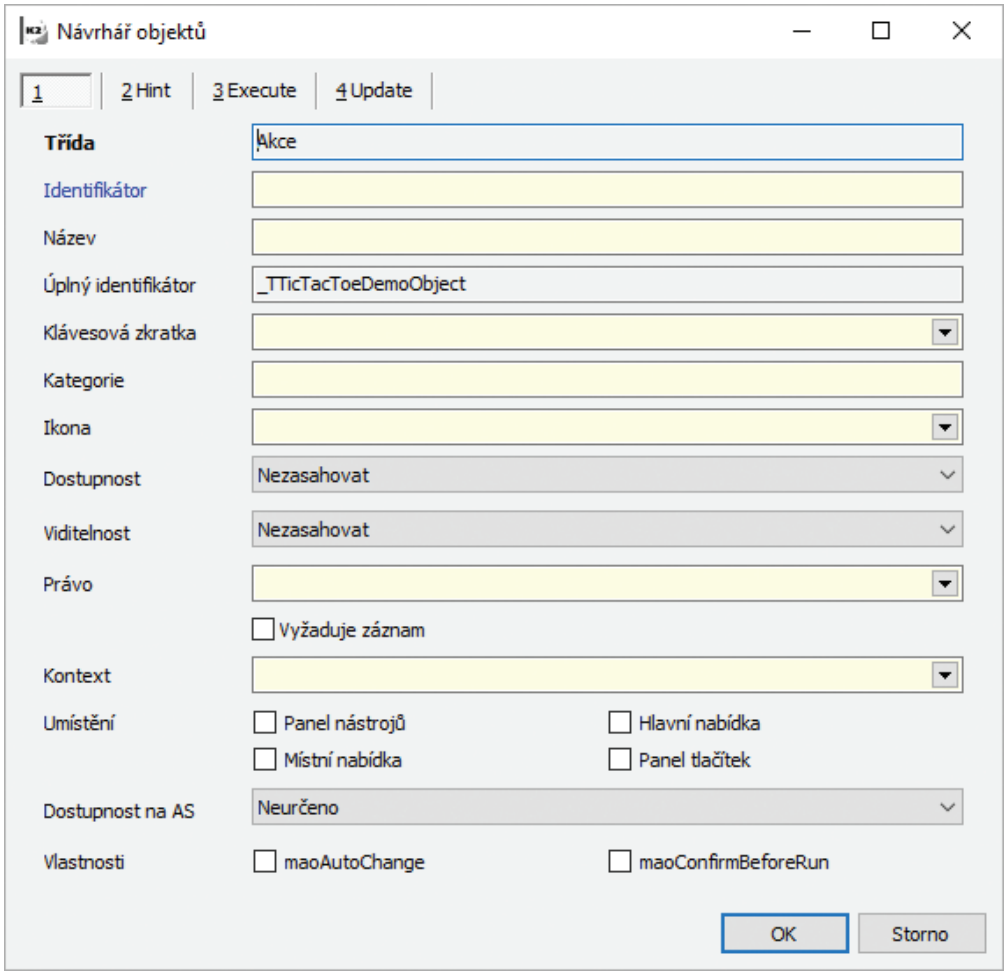
V této záložce se nacházejí zděděné property, stejně tak jak to bylo u modulů. Nacházejí se zde ale rozdílně property, níže si je blíže popíšeme.

6.5. AKCE

Část nazvaná ***„Akce“*** neboli ***„Commandy“*** slouží k implementaci metod, které je možné v rámci skriptové třídy spouštět, například stiskem definovaného tlačítka na formuláři nebo zkratkovou klávesou.

Novou akci vložíme pomocí stisknutí tlačítka Nový v nástrojové liště nebo stisknutím klávesy Insert.

Po vyvolání akce vložení záznamu se zobrazí formulář Obrázek 265 - Vytvoření nové akce ve skriptové třídě.



Obrázek 265 - Vytvoření nové akce ve skriptové třídě

6.5.1. ZÁKLADNÍ NASTAVENÍ

V této sekci nalezneme nastavení, které je základem pro definic akce. V následující části si je popíšeme.

Identifikátor
Slouží k identifikaci akce (commandu). Musí být unikátní v rámci skriptové třídy.

Název
Uživatelský název. Pojmenování, které je u akce zobrazeno a pomůže uživateli lépe pochopit, co akce může provádět.

Klávesová zkratka
Slouží k nastavení klávesové zkratky, která po stisknutí ve formuláři vyvolá výkonný kód dané akce, tedy akci spustí. Klávesová zkratka se definuje jednoduše textem. Tedy například *„Ctrl+K“*, *„Alt+S“*, *„Ctrl+Alt+K“* nebo *„Shift+Ctrl+N“*.

Kategorie
Slouží k vytvoření skupiny akcí. Například kde vložíme u akcí hodnotu ***„Speciál“***, vznikne v kontextových nabídkách, kde je akce vidět, nejprve pod menu ***„Speciál“*** a až následně akce.

Ikona
Zde definujeme ikonu, která bude zobrazena v případě zařazení akce na formulář. Jakým způsobem je tvořen identifikátor ikony je popsáno v kapitole 4.2.1. Skalár, přesněji v sekci ***„Zobrazovat jako ikony“***.

Dostupnost
Tato vlastnost udává, za jakých okolností je akce dostupná v datovém modulu. Možné nastavení jsou shrnuty již v dřívější kapitole 4.7. Akce. V případě, že nemá být akce dostupná, tlačítko bude ve formuláři existovat, ale nepůjde stisknout.

Viditelnost
Tato vlastnost udává, za jakých okolností je akce viditelná v datovém modulu. Možné varianty jsou stejné jako u vlastnosti ***„Dostupnost"***. V případě, že nemá být akce viditelná, tlačítko nebude ve formuláři vůbec existovat.

Právo
Pomocí této vlastnosti nastavujeme právo na použití akce. Tedy pouze uživatelé, kteří budou mít přiděleno právo, které bude nastaveno této akci, budou moci tuto akci používat. Hodnotou je tedy vazba do číselníku práv. V případě, že přidělujeme právo, otevírá se nám číselník práv, kde zvolíme patřičné právo, které bude svázáno s použitím této akce uživatelem.

Vyžaduje záznam
Akce vyžaduje ke správné funkčnosti záznam, který je aktuálně vybraný v datovém modulu, který akci vlastní.

Umístění
V této sekci se nachází několik variant umístění s příznaky, které je možné nastavit. Účelem je automatické zpřístupnění akce ve formuláři bez zásahu uživatele.

Dostupnost na AS
Tato vlastnost udává, za jakých okolností je akce dostupná na aplikačním serveru. Může nabývat hodnot ***„Neurčeno"***, ***„Nezveřejněno"*** a ***„Zvěřejněno"***. Pokud zvolíme ***„Neurčeno"***, bude se řídit jádrem K2. ***„Nezveřejněno"*** nám daný příkaz (command) nezveřejní na AS, opakem je potom možnost ***„Zvěřejněno"***.

Vlastnosti
maoAutoChange
Po skončení akce vynutí přenesení dat z objektu do formuláře = akce tím říká, že provádí nějaké změny, které je potřeba promítnout do formuláře.

maoConfirmBeforeRun
Před spuštěním akce bude uživatel vyzván k potvrzení spuštění. Je vhodně např. u akcí, které trvají dlouho nebo mají ***„destruktivní"*** účinky nebo jsou nevratně (výmaz, hromadná změna atd.).

6.5.2. ZÁLOŽKA „HINT"

Tato záložka v definici akce, slouží k definici nápovědy, která se zobrazí v případě, že je akce vložena do formuláře a uživatel na akci najedeme myší.

6.5.3. ZÁLOŽKA „EXECUTE"

V této záložce implementujeme již samotný skript, který je vykonán při spouštění akce. Implementujeme proceduru, která má jeden parametr ***„Sender"***, který je typu ***„TObjectAction"***. Z instance této třídy můžeme vyčíst informace o spuštěné akci.

6.5.4. ZÁLOŽKA „UPDATE“

V této záložce implementujeme skript, který je vykonán v případě spouštění akce, při překreslení formuláře apod. Zde si můžeme ošetřit spouštění akce tak, aby odpovídalo požadavkům, nebo můžeme na základě definované logiky měnit ostatní parametry akce, jako například zobrazený text. Jedná se o implementaci procedury, která má jako parametr proměnnou ***„Sender“*** typu ***„TObjectAction“***, která představuje instanci spuštěné akce.

POZNÁMKA: V případě obou procedur, je vstupním parametrem proměnná ***„Sender“***, která je v proceduře uvedena jako typ ***„TObject“***. Konkrétním typem této instance je ***„TObjectAction“***, na kterou si můžeme parametr v případě potřeby přetypovat.

6.6. VLASTNÍ METODY

Vlastní metody byly již popsány v kapitole 4.9. Vlastní metody u datového modulu.

6.7. ZÁVISLOSTI MODULŮ

Závislosti se definují stejně jako u datového modulu. Viz kapitola 4.11. Závislosti modulů. I zde platí, že vkládat lze pouze datové moduly. Ve zdrojovém kódu je pak generován blok „modules“.

6.8. ZÁVISLOSTI TABULEK

Závislosti tabulky se definují stejně jako u datového modulu, viz kapitola 4.12. Závislosti tabulek u datových modulů.

6.9. ZDROJOVÝ KÓD

V této záložce si můžeme prohlédnout výslednou implementaci vytvářené skriptové třídy.

6.10. API

Závislosti se definují stejně jako u datového modulu. Viz kapitola 4.14. API.

6.11. ZÁVISLOSTI JEDNOTEK

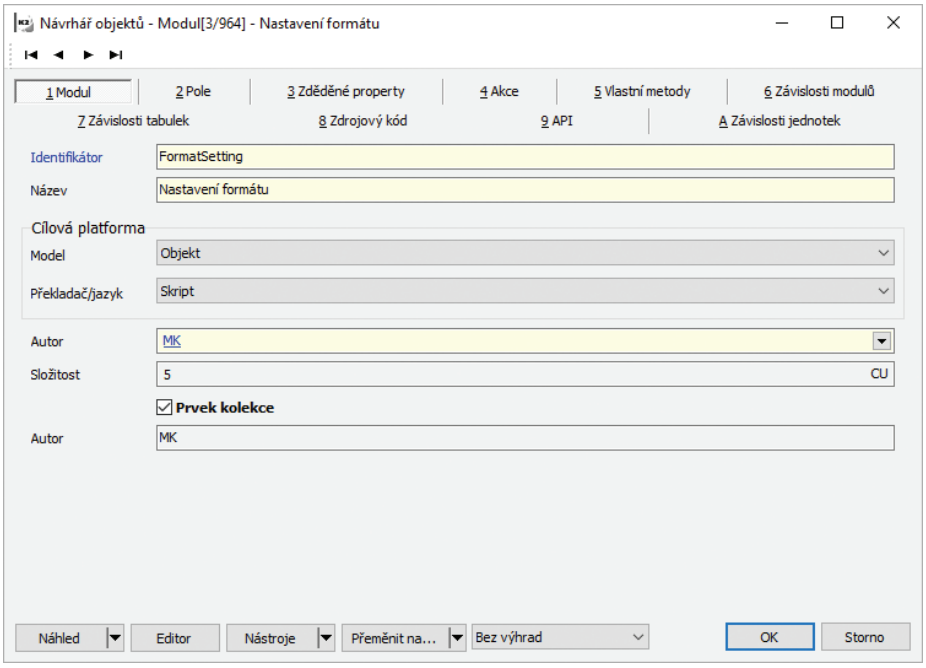
Závislosti jednotek se definují stejně jako u datového modulu. Viz kapitola 4.15. Závislosti jednotek u datových modulů. I zde platí, že vkládat lze jednotky a skriptové třídy. Ve zdrojovém kódu je pak generován blok „uses“.

PŘÍKLAD: Na závěr této kapitoly si vytvoříme komplexní příklad pro skriptové třídy. Bude se jednat o formátování jména.

Skriptové objekty vytvoříme dva. První bude sloužit ke konfiguraci, kdy si zvolíme, jak má formát jména vypadat. Druhá třída bude sloužit k samotnému zobrazení dat a zadání jména.

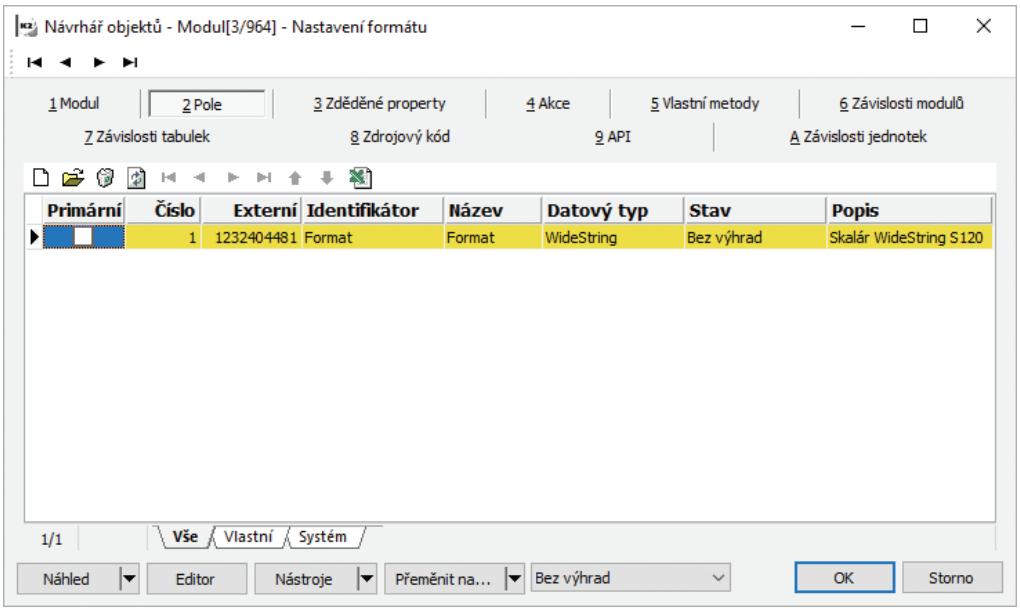
Jako první si tedy vytvoříme třídu, která bude sloužit ke konfiguraci zobrazovaných dat. Bude se jednat o seznam složení jména.

Založíme novou skriptovou třídu a pojmenujeme ji *„FormatSetting“* – *„Nastavení formátu“*. Tento objekt označíme jako prvek kolekce, zatrhneme tedy volbu *„Prvek kolekce“*. Vše je vidět na Obrázek Obrázek 266 - Příklad na skriptovou třídu – Formátování jména – nastavení.



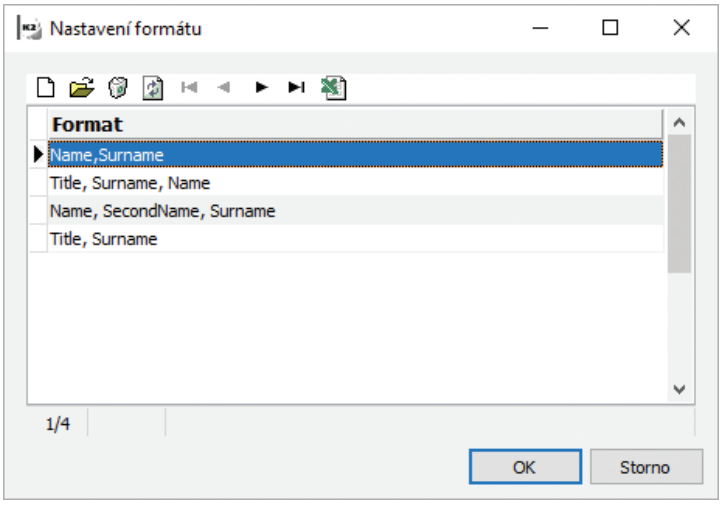
Obrázek 266 - Příklad na skriptovou třídu – Formátování jména – nastavení

Přejdeme na záložku *„Pole“* a nadefinujeme pole, která budeme mít v konfiguračním objektu. Jedná se o formát, ve které budou napsány pole pro formát jména. Vytvoříme pole *„Format“* typu *„widestring“* (řetězec). Jednoduchá definice je vidět na Obrázek 267 - Příklad na skriptovou třídu – Formátování jména – nastavení – pole.



Obrázek 267 - Příklad na skriptovou třídu – Formátování jména – nastavení – pole

V tomto objektu již nepotřebujeme dále nic definovat ani implementovat. Vytvořený objekt si můžeme prohlédnout pomocí funkce *„Náhled“*. Můžeme také vyzkoušet vložit záznamy. Vše je vidět na Obrázek 268 - Příklad na skriptovou třídu – Formátování jména– nastavení – náhled.



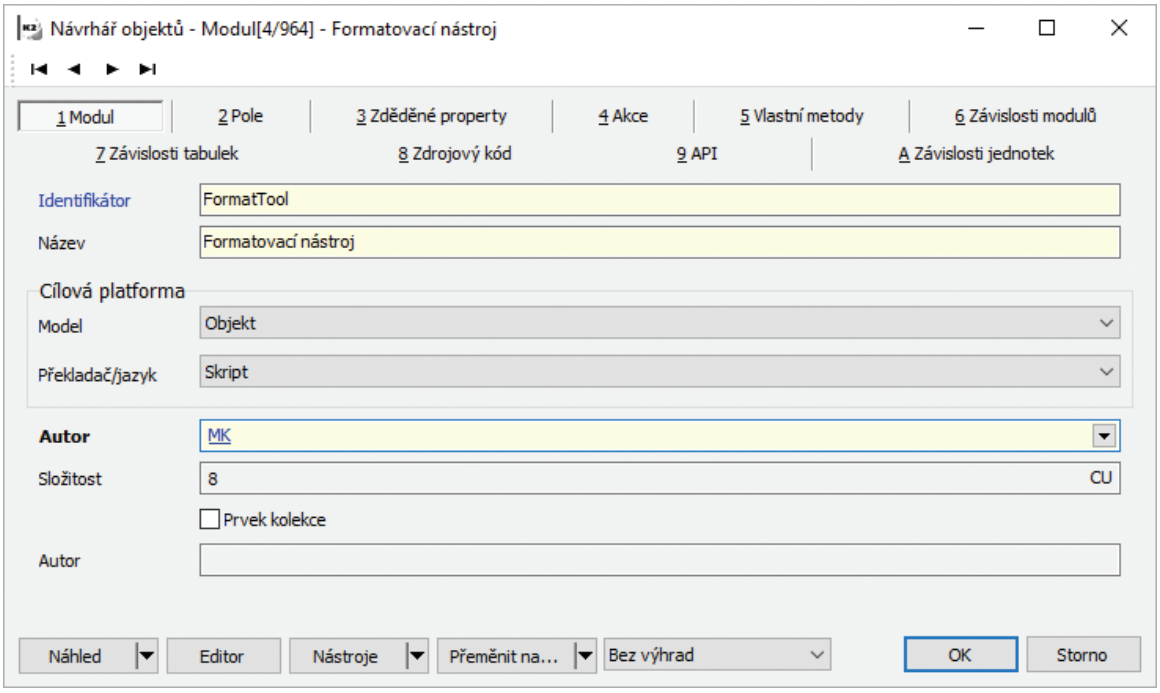
Obrázek 268 - Příklad na skriptovou třídu – Formátování jména– nastavení – náhled

POZNÁMKA: V případě zobrazení formuláře pomocí funkce ***„Náhled"*** jsou data načítána a ukládána v xml souboru, který je uložen v uživatelském adresáři. Tato data slouží pouze k náhledu vytvářeného objektu a nejsou dále používána v případě nasazení objektu do K2.

POZNÁMKA: Funkce ***„Náhled"*** funguje nad skriptovými třídami i v případě, že neproběhnula operace ***„Deploy"***.

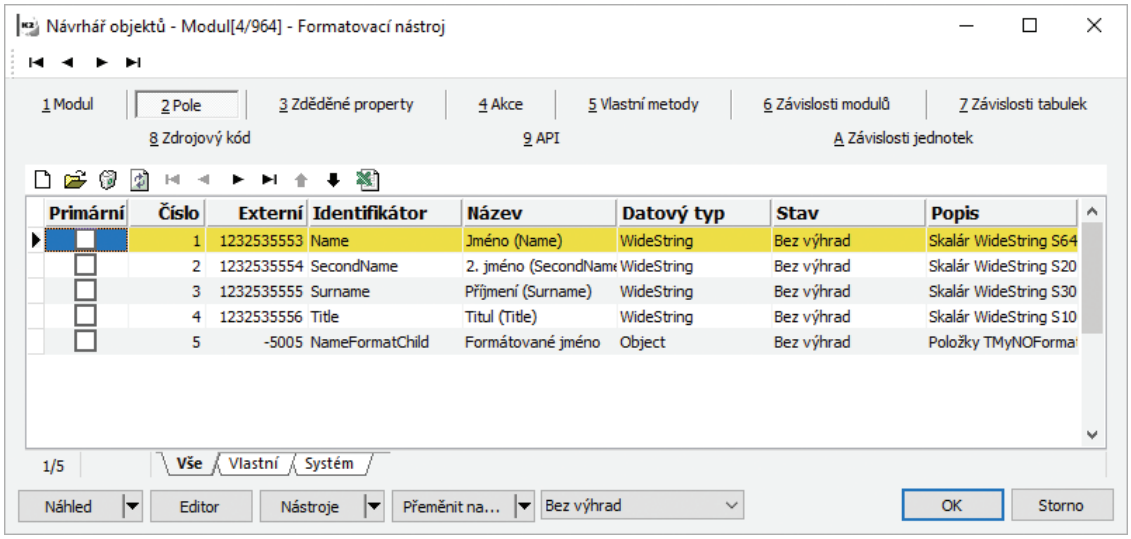
POZNÁMKA: Při spuštění formuláře pomocí operace ***„Náhled"*** je možné celou skriptovou třídu ladit pomocí editoru skriptu.

Další částí našeho příkladu je definice a implementace druhého objektu, který bude sloužit k zobrazení formátu jména dle konfigurace. Výsledkem bude zobrazený formulář, který přečte konfiguraci a jméno, které zadáme ve druhém objektu a zapíše výsledné formáty. Vytvoříme tedy novou skriptovou třídu ***„FormatTool"*** (Formátovací nástroj) - viz Obrázek 269 - Příklad na skriptovou třídu – Formátování jména.



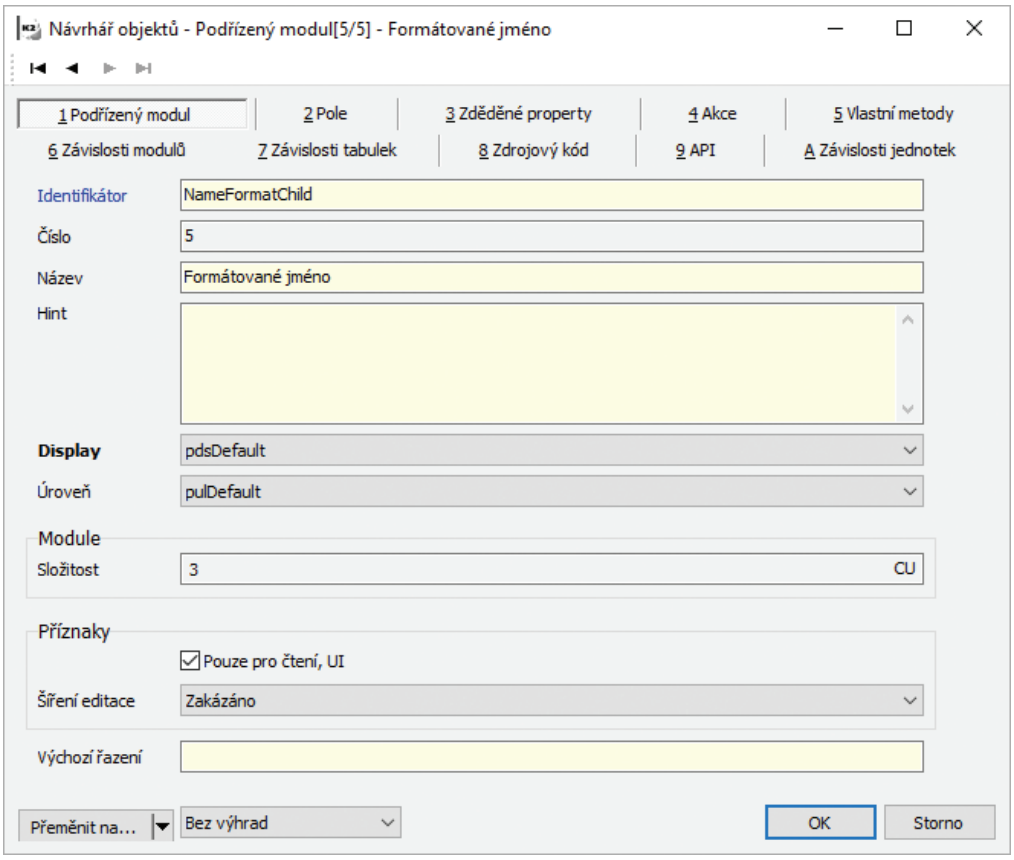
Obrázek 269 - Příklad na skriptovou třídu – Formátování jména

Ve formuláři budou definovány 4 pole typu ***„WideString“***, ***„Name“*** (Jméno), ***„SecondName“*** (2. jméno), ***„Surname“*** (Příjmení) a ***„Title“*** (Titul). Tyto pole budou sloužit pro zadání jména, aby se podle konfigurace jméno naformátovalo. Dále bude pole ***„NameFormatChild“*** (Formátované jméno), typu ***„Položky“***. Vše je vidět na Obrázek 270 - Příklad na skriptovou třídu – Formátování jména – pole.



Obrázek 270 - Příklad na skriptovou třídu – Formátování jména – pole

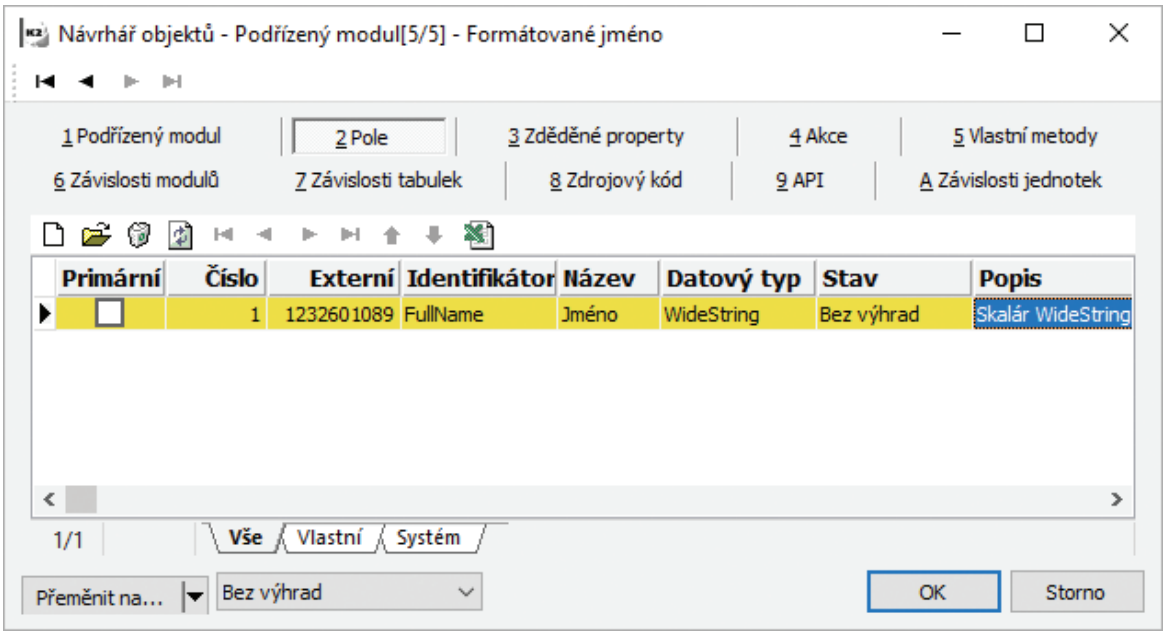
Na následujícím obrázku - Obrázek 271 - Příklad na skriptovou třídu – Formátování jména – položky, pak můžeme vidět definici položkové skriptové třídy ***„NameFormatChild“***. Protože tato data se budou načítat sami, zatrhneme volbu ***„Pouze pro čtení“***.



Obrázek 271 - Příklad na skriptovou třídu – Formátování jména – položky

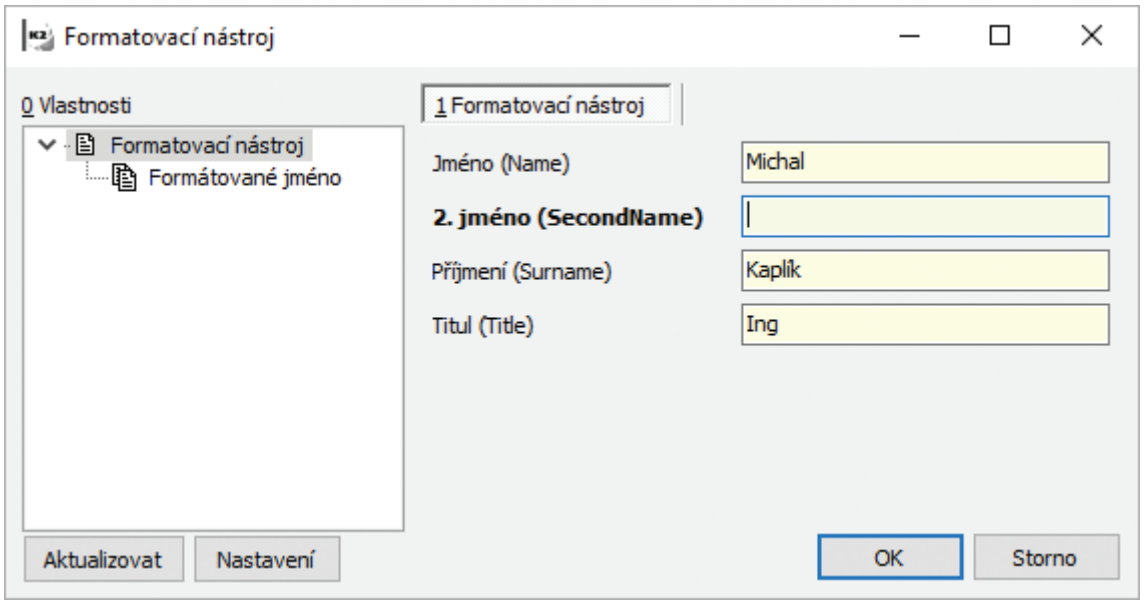
V položkové třídě je potřeba dále nadefinovat pole, které bude obsahovat zformátované jméno. Vytvoříme pole ***„FullName“*** (Jméno) typu řetězce ***„WideString“***. Pole ***„Master“*** je automaticky vloženo z hlavičkového objektu.

Definici polí pro položky je vidět na Obrázek 272 - Příklad na skriptovou třídu – Formátování jména – položky – pole.



Obrázek 272 - Příklad na skriptovou třídu – Formátování jména – položky – pole

Průběžně si můžeme vytvářený objekt zobrazit pomocí funkce ***„Náhled“***. Díky tomu můžeme jednoduše a rychle vyladit rozložení prvků formuláře v případě použití klasického generovaného formuláře, viz Obrázek 273 - Příklad na skriptovou třídu – Formátování jména – náhled std. formuláře.

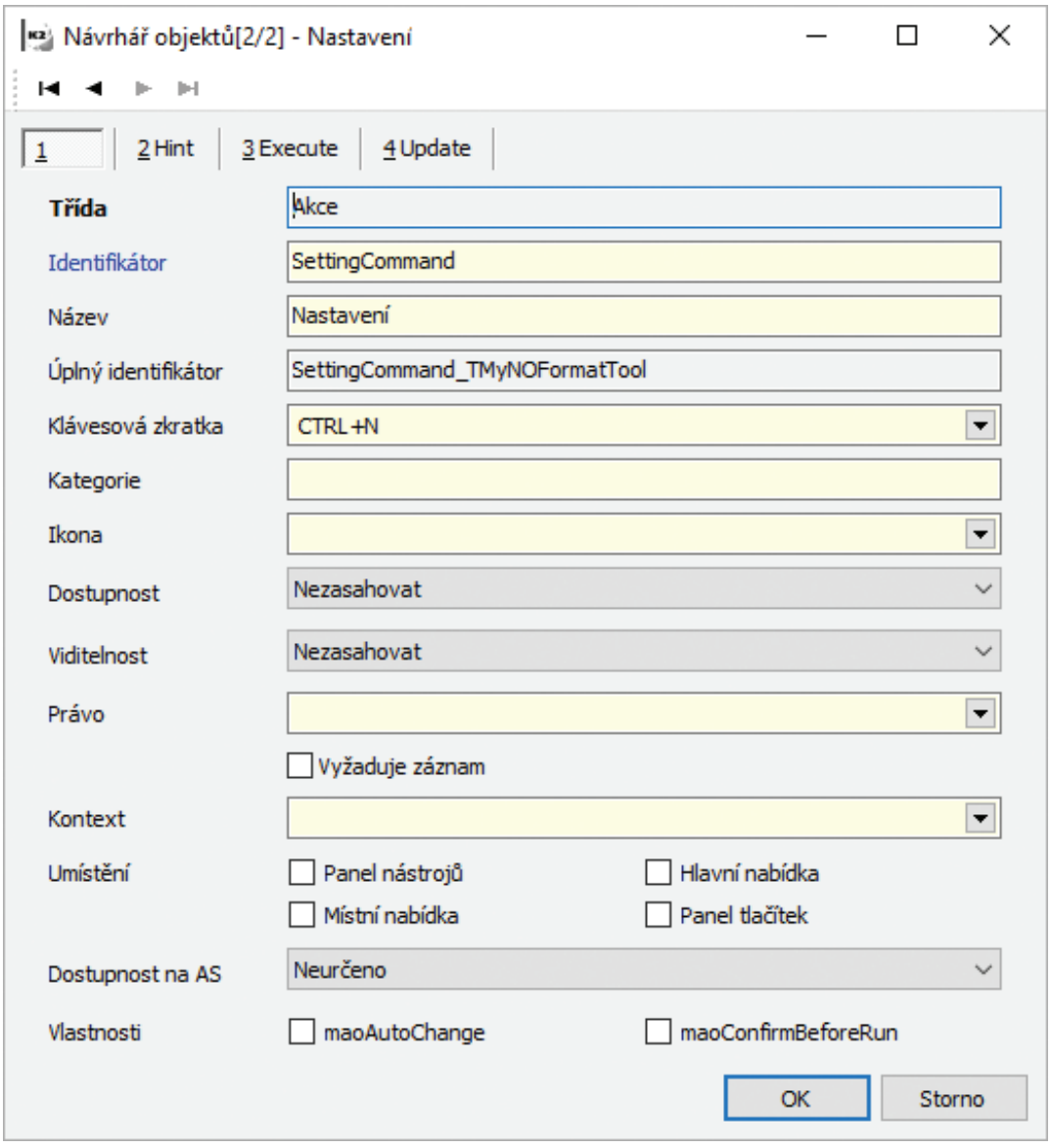


Obrázek 273 - Příklad na skriptovou třídu – Formátování jména – náhled std. formuláře

Máme vytvořené dvě skriptové třídy, co se týká struktury. Zatím neobsahují žádnou logiku, která by prováděla načítání a ukládání dat.

V další fázi si tedy definujeme a implementujeme dvě akce, které budou řešit logiku aplikace. První bude sloužit k zobrazení nastavení konfigurace a následně uložení změn. Druhá akce bude sloužit k přečtení dat dle konfigurace.

Přepneme se na záložku ***„Akce“*** a vytvoříme novou akci ***„SettingsCommand“*** – ***„Nastavení“***. Nastavíme klávesovou zkratku na ***„Ctrl+N“***, viz Obrázek 274 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení".

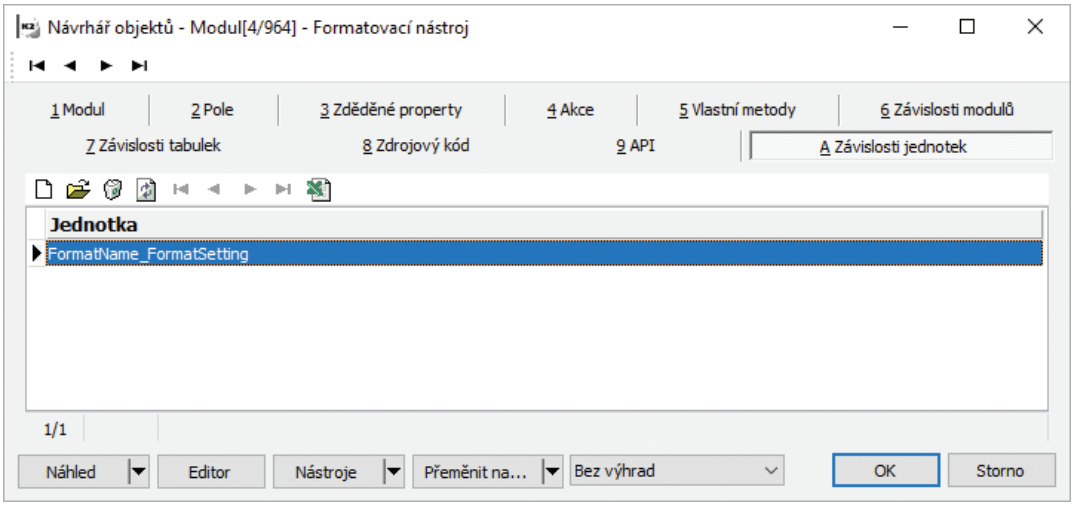


Obrázek 274 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení"

Dále se přepneme na záložku ***„Execute“*** a implementujeme načtení konfigurace, zobrazení ve formuláři a následně provedeme uložení.

POZNÁMKA: Kód můžeme implementovat jak v editoru skriptu, tak v záložce ***„Execute“***. Při spouštění skriptu objektu pomocí funkce ***„Náhled“*** pak můžeme ověřit přeložitelnost nebo laděním i použitelnost vytvořeného kódu.

*V první řadě si potřebujeme vytvořit instanci objektu pro nastavení – **„TFormatSettingCollection “**. Protože tato skriptová třída je jiná než aktuální, ve které implementujeme akci, je nutné skriptovou třídu s nastavením připojit do závislostí. Tedy programátorskou řečí, doplnit do bloku **„uses“** ve skriptu. Vrátíme se tedy do definice skriptové třídy **„TMyNOFormatSetting“** a přepneme se na záložku **„Závislosti jednotek“**. Zde přidáme nový záznam a tím bude naše skriptová třída pro nastavení. Její celý název včetně identifikátoru je **„FormatName_FormatSetting“**. Díky tomu se jednotka přidá do sekce **„uses“** ve výsledném kódu a můžeme používat její definici. Nastavení je vidět na* Obrázek 275 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – závislosti.



Obrázek 275 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – závislosti

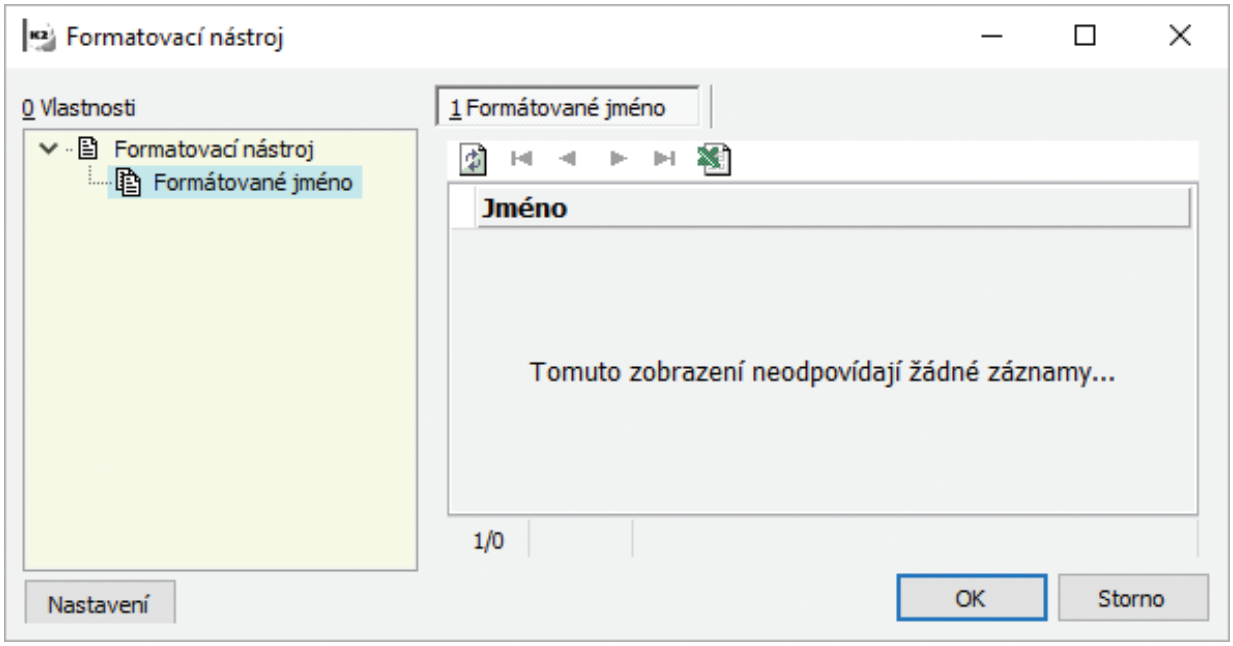
Můžeme tedy vytvořit instanci třídy pro nastavení. Dále provedeme načtení uloženého nastavení ze struktur, které IS K2 nabízí k ukládání stavů objektů. Jedná se o struktury uložené v databázových tabulkách ***„ObjectList“***, ***„ObjectStorage“*** apod. Načtení z těchto struktur provedeme pomocí procedury ***„LoadObjectLevel“***, která má jako vstupní parametry ***„GUID“***, který jednoznačně identifikuje záznam. Guid si musíme vytvořit a dále ho používat. V příkladu je definován jako konstanta. Dalším parametrem je samotná instance naší třídy, tedy ***„settings“***, která se bude ukládat. Posledním parametrem je úroveň, pro kterou je objekt použitelný. Možnosti pro NO jsou ***„olMandant“*** – uložení pro všechny uživatele, kteří jsou přihlášení v aktuálním mandantovi a ***„olUser“*** – uložení jen pro aktuálního uživatele. V našem příkladu si každý uživatel bude moct vytvořit vlastní seznam, tedy použijeme hodnotu ***„olUser“.***

Máme načtený uložený objekt naší skriptové třídy a v dalším kroku je potřeba ho zobrazit ve formuláři. V případě, že budeme chtít zobrazit objekt v šedých formulářích, můžeme použít metodu ***„EditObject“***, které nastavíme jako první parametr ***„nil“*** a druhým parametrem předáme instanci naší skriptové třídy, tedy ***„settings“***. V případě, že bychom chtěli formulář zobrazit v univerzálních formulářích, použijeme funkci objektu TUniFormManager.DialogExecutor.ShowObjectModal, kde prvním parametrem je instance, v našem případě ***„setting“***.

Celá implementace akce je k dispozici zde:

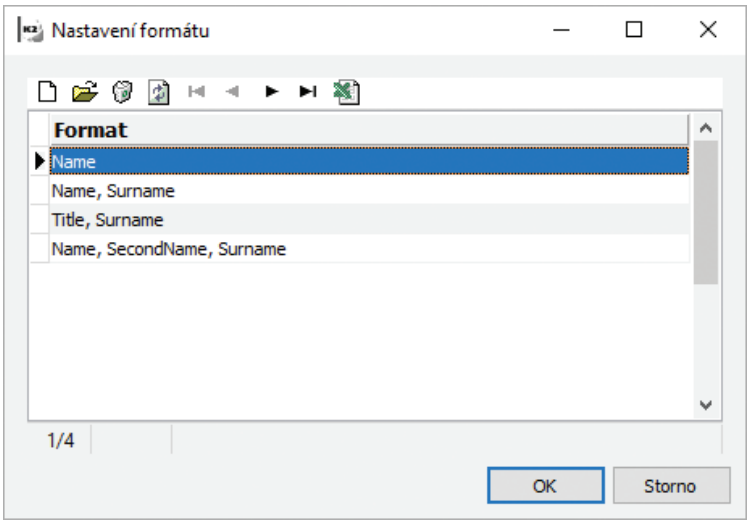
```
const
  GUID = '{50A7E4AE-0DB0-4CEB-8021-2F94C033B0D3}';

var
    settings : TFormatSettingCollection;
begin
    try
      settings := TFormatSettingCollection.Create(nil, TFormatSettingItem);
      LoadObjectLevel(StringToGUID(GUID), settings, olUser);

      if TObjectAction(Sender).UFContext <> nil then
         TUniFormManager.DialogExecutor. ShowObjectModal(settings, '', [], True)
      else
         EditObject(nil, settings);

      SaveObjectLevel(StringToGUID(GUID), settings, olUser, 'FormatSettings', '');
    finally
      settings.Free;
    end;
end;
```

Následně můžeme vyzkoušet funkčnost vytvořené akce. Spustíme formulář pomocí akce ***„Náhled“***. Zobrazí se formulář, kde můžeme vidět nově tlačítko **Nastavení**, které představuje naši novou akci, viz Obrázek 276 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – zobrazení akce.



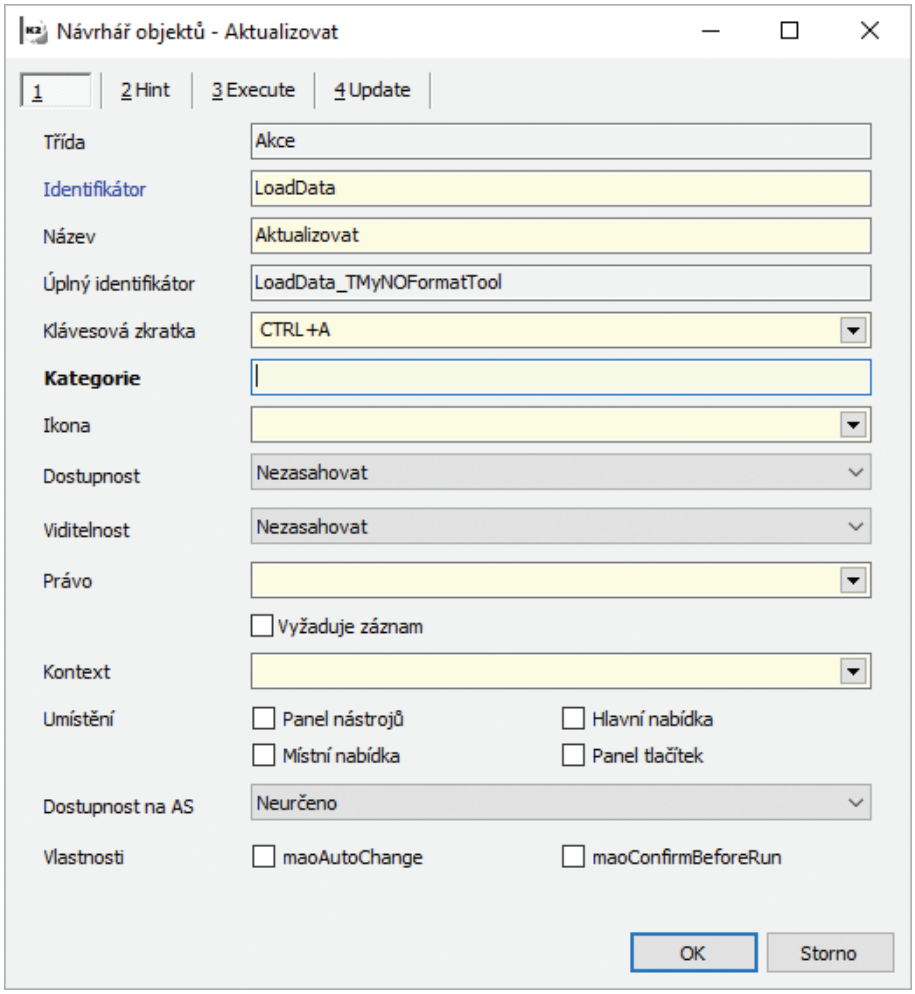
Obrázek 276 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – zobrazení akce

Po stisknutí tlačítka **Nastavení** se zobrazí standardní formulář, kde můžeme vkládat záznamy do konfigurace. Formulář můžeme vidět na Obrázek 277 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – zobrazení.



Obrázek 277 - Příklad na skriptovou třídu – Formátování jména – akce "Nastavení" – zobrazení

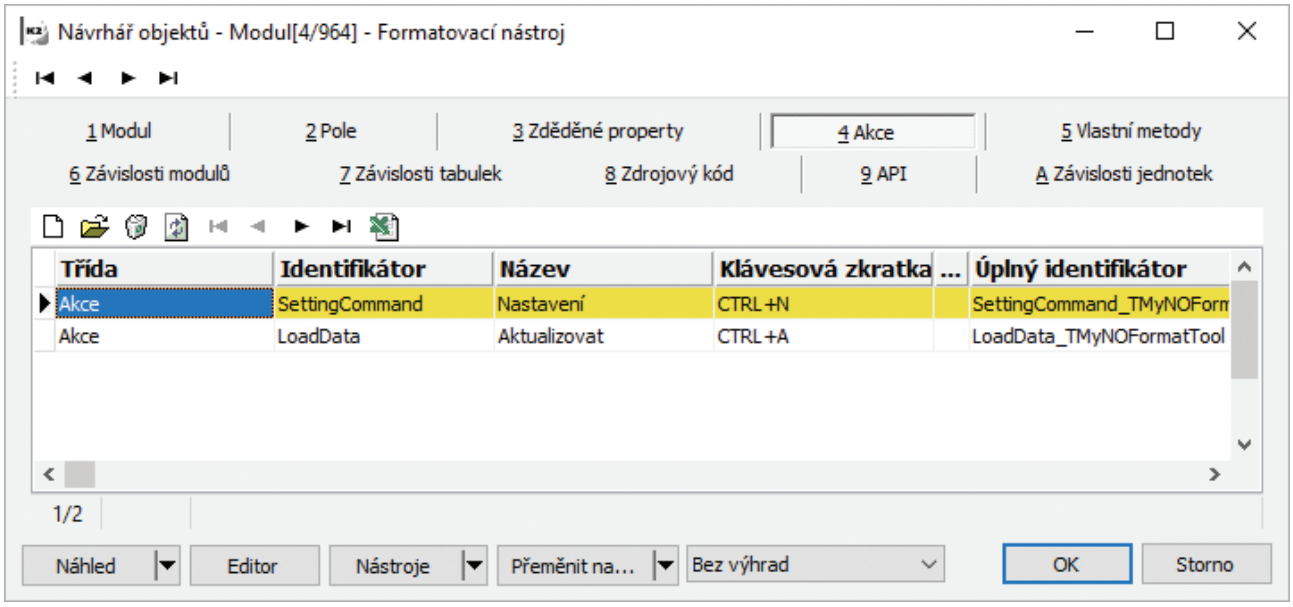
První část příkladu, kterou je konfigurace, máme hotovou. Zbývá implementovat druhou část a tou je samotné načtení dat dle konfigurace. Přepneme se tedy opět na záložku ***„Akce“*** ve skriptové třídě ***„FormatTool“***. Založíme novou akci, kterou nazveme ***„LoadData“*** – ***„Aktualizovat“*** a přiřadíme ji klávesovou zkratku ***„Ctrl+A“***, viz Obrázek 278 - Příklad na skriptovou třídu – Formátování jména – Akce načti data.



Obrázek 278 - Příklad na skriptovou třídu – Formátování jména – Akce načti data

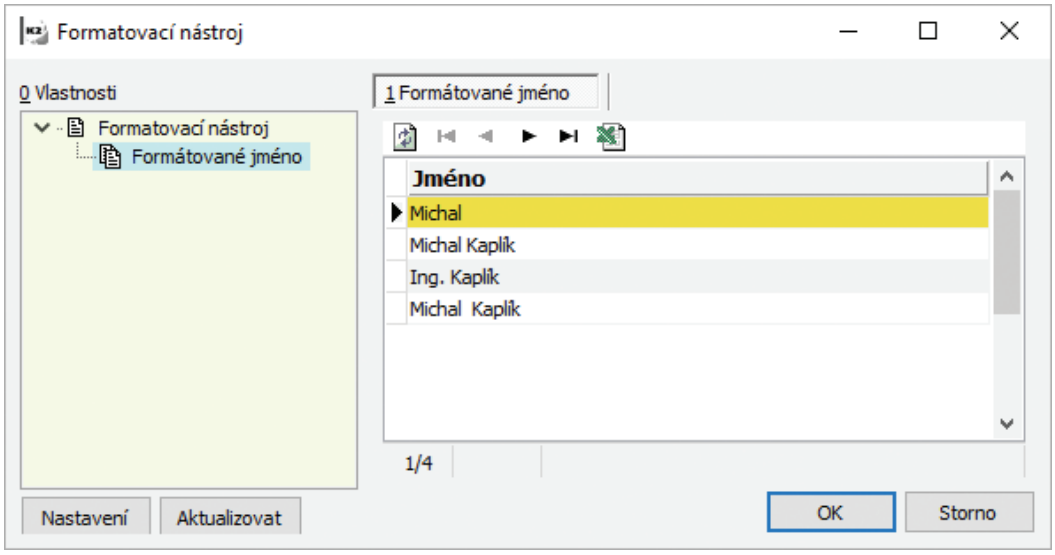
Dále je nutné implementovat samotné načtení dat. Přepneme se na záložku ***„Execute“***. Zde implementujeme načtení dat. Detailně si tuto funkci popisovat nebudeme. Čtenář si jí může nastudovat sám, v rámci školení budeme funkci rozebírat.

Konstrukce je souhrnný pohled na definované akce je vidět na Obrázek 279 - Příklad na skriptovou třídu – Formátování jména – Všechny akce



Obrázek 279 - Příklad na skriptovou třídu – Formátování jména – Všechny akce

Celý příklad máme nyní hotový, co se týká definice. Jediná možnost jak si ho teď prohlédnout je přes funkci ***„Náhled“***, která nám zobrazí formulář. Zobrazení s načtenými daty pomocí tlačítka Aktualizovat, je vidět na Obrázek 280 - Příklad na skriptovou třídu – Formátování jména – Náhled.



Obrázek 280 - Příklad na skriptovou třídu – Formátování jména – Náhled

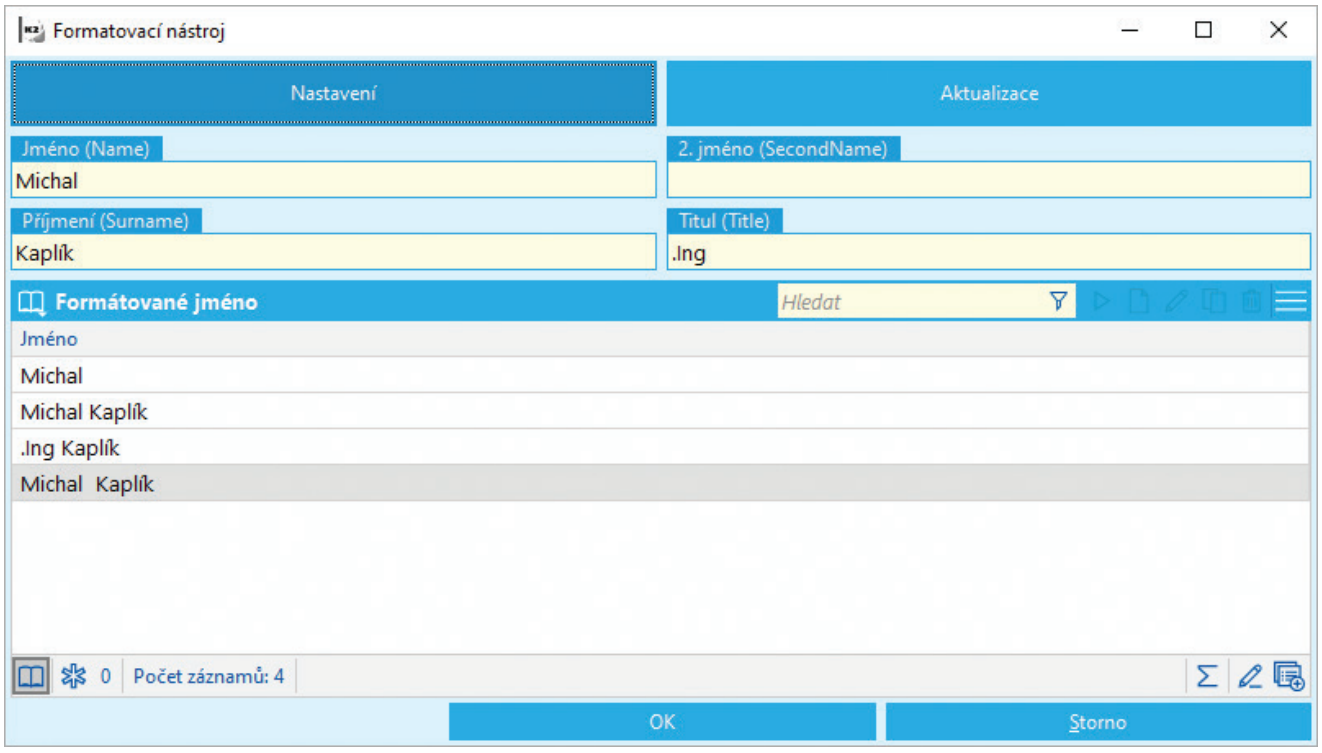
Abychom mohli vytvořené objekty používat v K2, je nutné provést nad daným rozšířením operaci ***„Deploy“***. Poté bude K2 tyto struktury znát. Dalším krokem je pak zakomponování do míst, kde budeme chtít vytvořenou funkčnost používat. V našem příkladu vytvoříme programový skript, ve kterém vytvoříme instanci naší nové skriptové třídy a zobrazíme v univerzálních formulářích.

Vytvoříme tedy nový skript jako program, tedy spustitelný skript. Vložíme do sekce ***„uses“*** odkaz na unitu ***„FormatName_FormatTool“***. Ve výkonné části pak vytvoříme instanci třídy ***„TFormatToolObject“*** a zobrazíme formulář. Pokud bychom chtěli zobrazit formulář pomocí standardních formulářů, pak stačí zavolat proceduru ***„EditObject“*** a předat ji jako parametr instanci třídy. V případě univerzálních formulářů je potřeba použít jinou konstrukci, se kterou jsme se již setkali v případě implementace zobrazení nastavení ***„TUniFormManager.DialogExecutor.EditObjectModal()“***, které předáme instanci objektu k zobrazení.

Celá implementace programu je k dispozici zde:

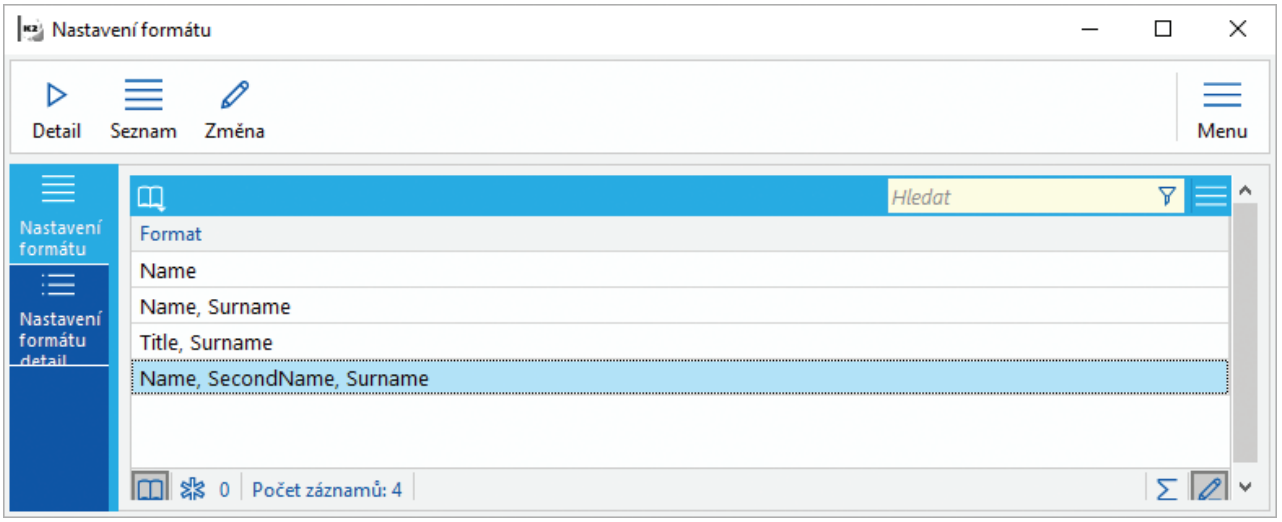
```
uses
  FormatName_FormatTool;

var
  AP : TFormatToolObject;

begin
  try
    AP := TFormatToolObject.Create(nil);
    TUniFormManager.DialogExecutor.EditObjectModal(AP, '', []);
  finally
    AP.Free;
  end;
end.
```

Po spuštění se zobrazí univerzální formulář, který je generovaný. Pomocí designéru formulářů ho můžete jednoduše pře stylovat do podoby, která je vyhovující. Například na Obrázek 281 - Příklad na skriptovou třídu – Formátování jména – Spuštění programu je vidět zobrazený formulář i s načtenými daty.



Obrázek 281 - Příklad na skriptovou třídu – Formátování jména – Spuštění programu

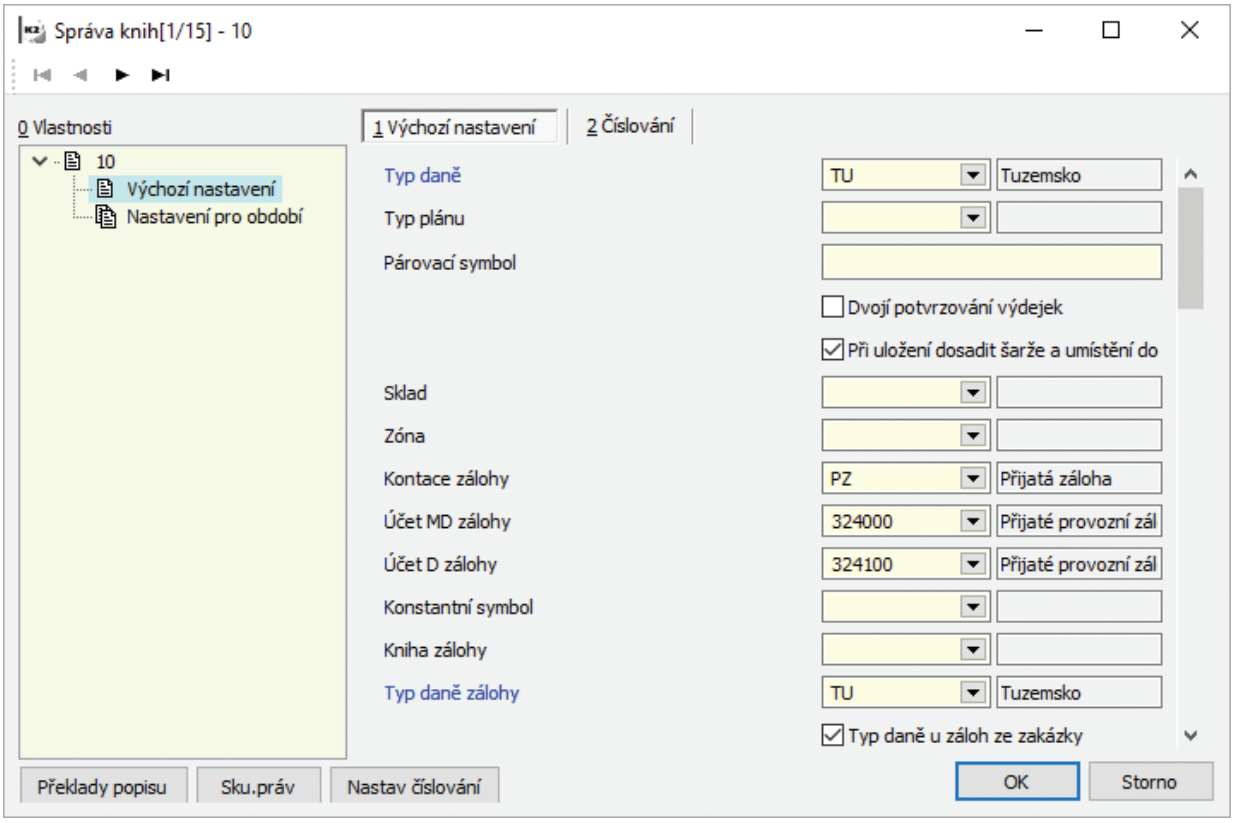
Na Obrázek 282 - Příklad na skriptovou třídu – Formátování jména – nastavení UF je pak vidět spuštěný formulář s konfigurací.



Obrázek 282 - Příklad na skriptovou třídu – Formátování jména – nastavení UF

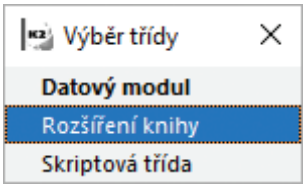
7. ROZŠÍŘENÍ KNIHY

Další oblastí, kterou lze pomocí návrháře objektů rozšiřovat, jsou nastavení pro jednotlivé knihy pro datové moduly. Každý typ knihy má definováno nastavení, které je možné použít. Například kniha s označením ***„10“***, která patří do knih prodeje má rozsáhlé nastavení, které je částečně vidět na Obrázek 283 - Ukázka nastavení knihy 10 – Prodej. Tuto množinu je možné právě pomocí návrháře objektů rozšířit.



Obrázek 283 - Ukázka nastavení knihy 10 – Prodej

Rozšíření knihy vyvoláme stisknutím tlačítka Insert nebo stiskem ikony nad seznamem modulů v hlavní nabídce návrháře objektů. Zobrazí se nabídka, viz Obrázek 284 - Založení rozšíření knihy, kde vybereme volbu ***„Rozšíření knihy“***.



Obrázek 284 - Založení rozšíření knihy

Zobrazí se formulář pro definici nového rozšíření knihy, který si v následujícím textu popíšeme včetně příkladu.

7.1. MODUL

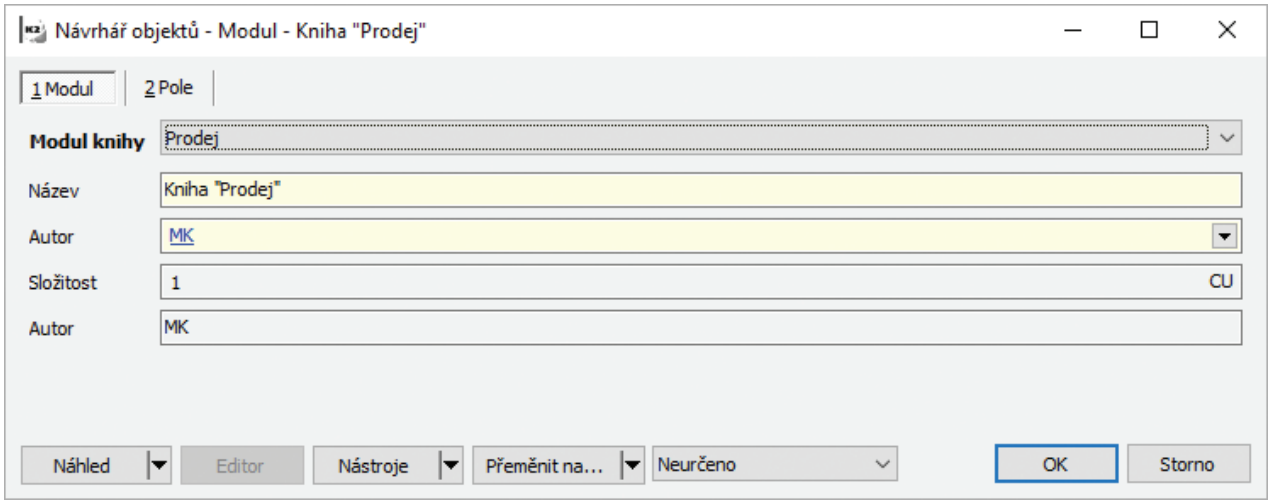
První záložkou formuláře je ***„Modul“***, kde se definuje, kterou knihu budeme rozšiřovat, viz Obrázek 285 - Rozšíření knihy - sekce modul. K dispozici jsou následující vstupní informace.

Modul knihy
Slouží k výběru knihy, kterou budeme rozšiřovat.

Název
Název rozšíření, pod kterým se bude záznam zobrazovat v seznamu všech rozšíření v návrháři objektů.

Autor
Autor, který založil rozšíření knihy.

Složitost
Vypočtená složitost daného rozšíření.

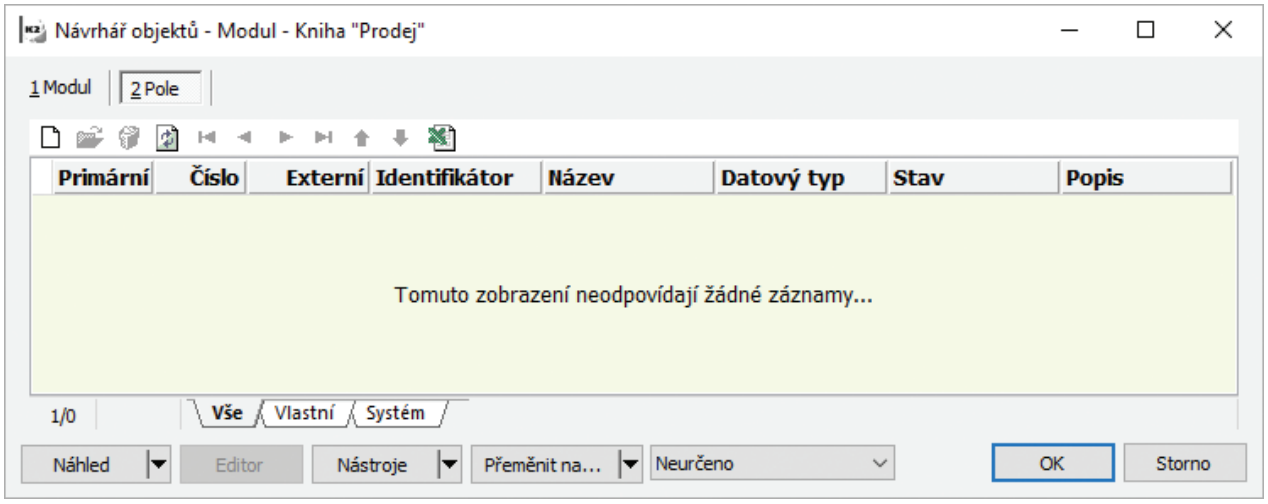


Obrázek 285 - Rozšíření knihy - sekce modul

7.2. POLE

Další záložka ***„Pole"*** slouží k definici polí, kterými budeme rozšiřovat vybraný typ knih. Nové pole přidáme stejně jako u datového modulu, stiskem tlačítka Insert nebo stisknutím ikony nad seznamem polí, viz Obrázek 286 - Rozšíření knihy - sekce pole.

Knihu můžeme rozšířit o skalární pole, cizí klíč a přímou vazbu.

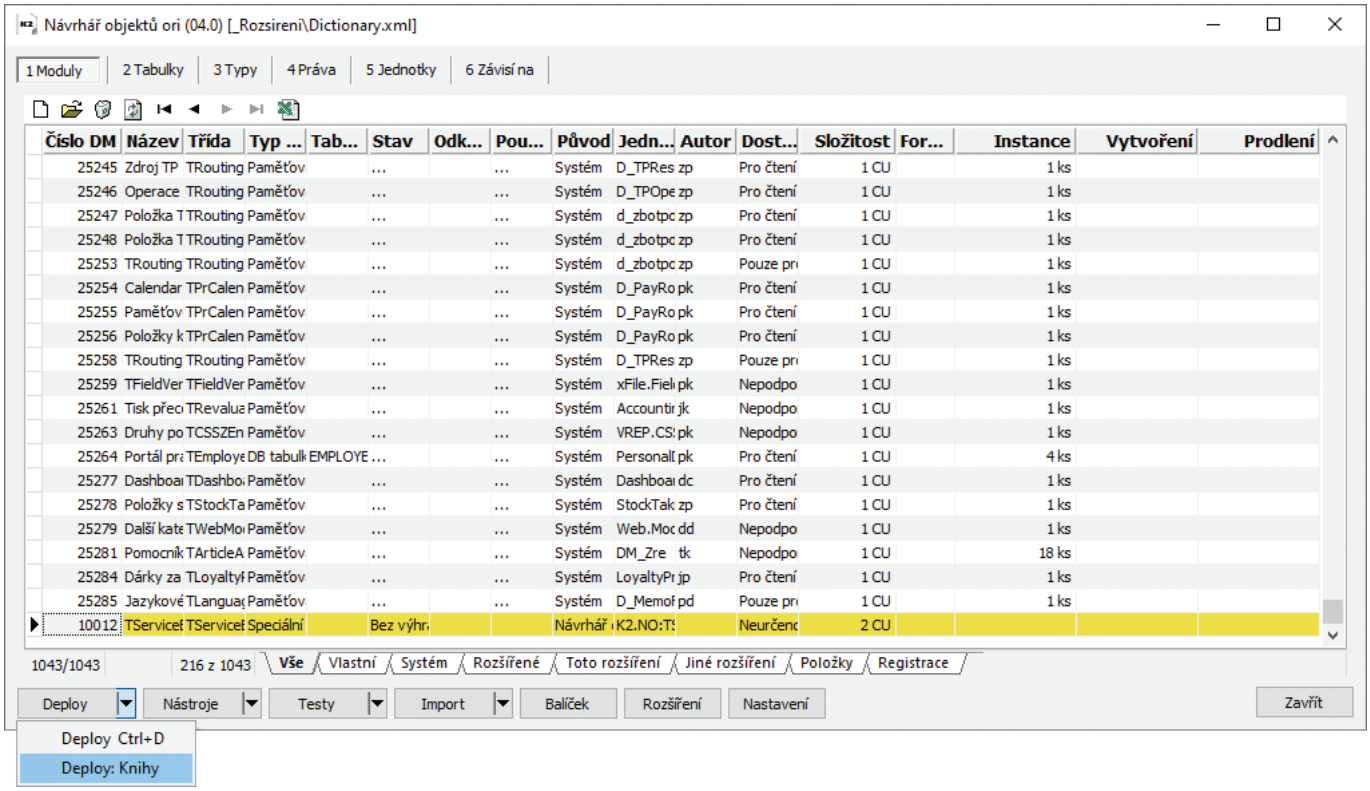


Obrázek 286 - Rozšíření knihy - sekce pole

7.3. APLIKACE ÚPRAV – DEPLOY

Stejně jako u datových modulů, všechny změny, které chceme používat, je nutné tzv. aplikovat. Pro aplikaci změn v rozšíření knih existuje funkce návrháře objektů, která se nazývá ***„Deploy: Knihy“***, kterou máme k dispozici pod tlačítkem Deploy v hlavním okně návrháře objektů, viz Obrázek 287 - Rozšíření knihy – Deploy.

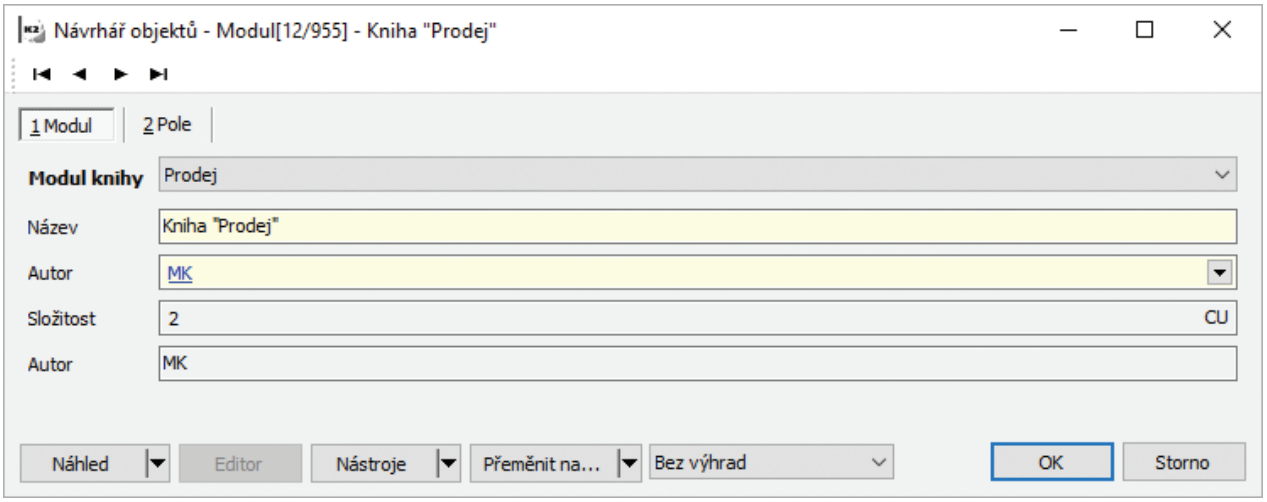
Po provedení akce ***„Deploy“*** může být uživatel vyzván k restartu K2. Dokud takto neučiní, další ***„Deploy“*** nepůjde udělat.



Obrázek 287 - Rozšíření knihy – Deploy

PŘÍKLAD: Pro lepší představu si celý proces rozšíření knihy ukážeme na příkladu. Pomocí návrháře objektů rozšíříme modul knih ***„Prodej“*** o dvě nové informace. Evidenci doplňující poznámky a zodpovědnou osobu za konkrétní knihu.

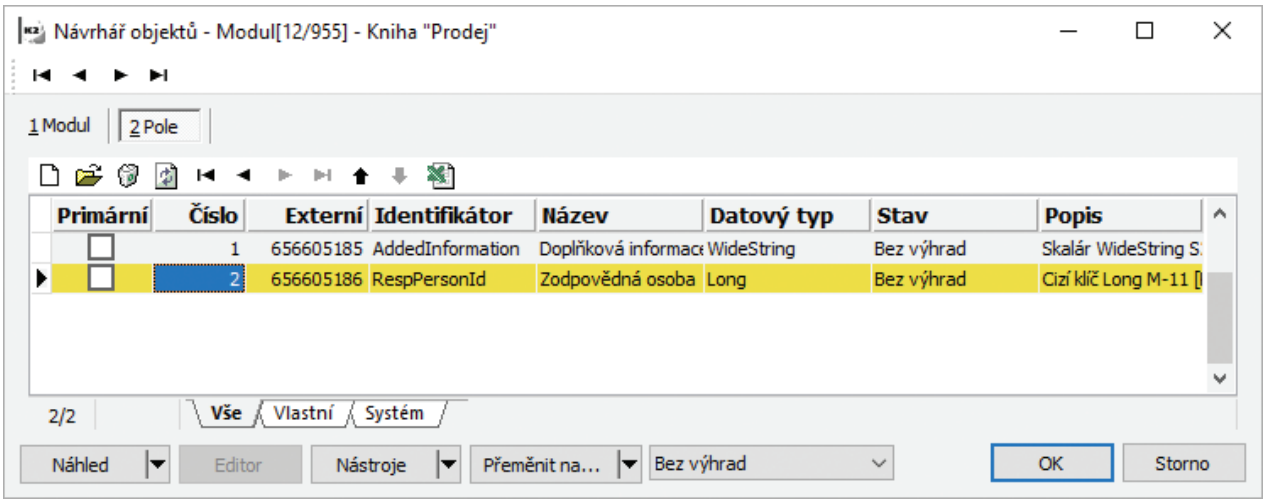
Stiskneme klávesu Insert a vybereme volbu ***„Rozšíření knihy“***, viz Obrázek 7 - Výběr nového typu objektu. Ve formuláři vybereme v poli ***„Modul knihy“*** hodnotu ***„Prodej“***. Jako název můžeme ponechat předvyplněnou hodnotu ***„Kniha Prodej“***, viz Obrázek 288 - Definice rozšíření knihy.



Obrázek 288 - Definice rozšíření knihy

V dalším kroku se přepneme na záložku ***„Pole“*** kde nadefinujeme výše avizovaná dvě pole. Pomocí tlačítka Insert vložíme pole typu ***„Skalár“***. Jako identifikátor uvedeme ***„AddedInformation“*** a zobrazovaný název ***„Doplňková informace“***. Pole bude typu řetězec, tedy typ ***„string“***. Účelem pole bude možnost zapsat ke knize doplňující informace, které jsou vhodné pro uživatele.

Druhým polem, které do rozšíření vložíme, je typu ***„cizí klíč“***. Cílový datový modul vybereme ***„Kontaktní osoby“*** a jako identifikátor uvedeme ***„RespPersonId“***. Zobrazovaný název bude ***„Zodpovědná osoba“***. Pole bude sloužit pro uvedení zodpovědné osoby pro každý záznam jednotlivých knih. Vše můžeme vidět na Obrázek 289 - Definice polí v rozšíření knihy.

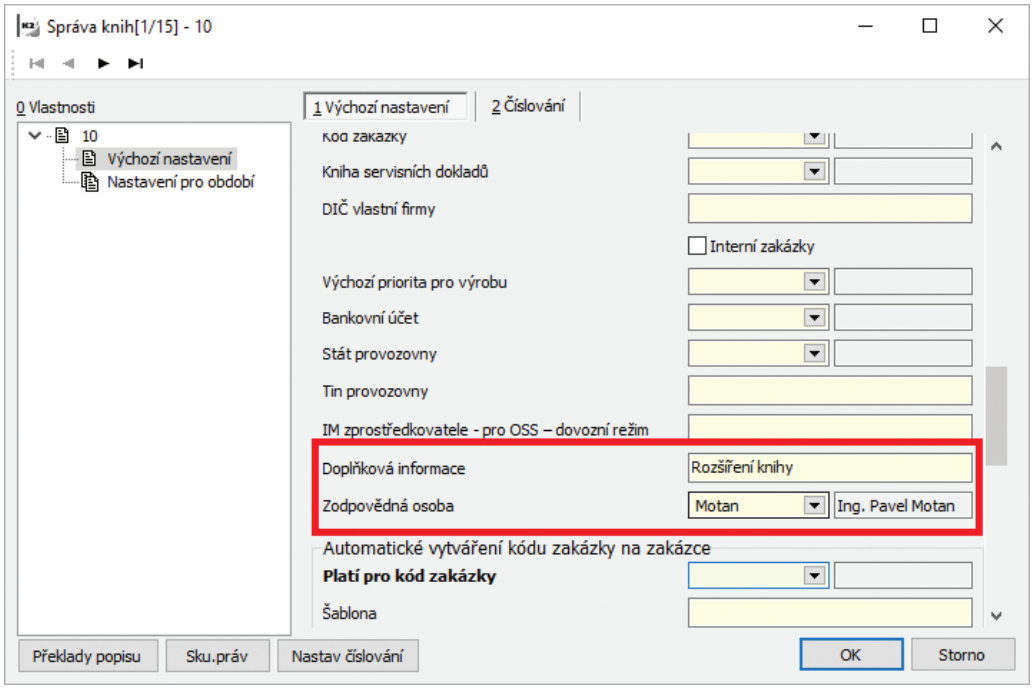


Obrázek 289 - Definice polí v rozšíření knihy

Definici rozšíření ukončíme tlačítkem OK. Pro aplikování nového rozšíření je ještě potřeba spustit akci ***„Deploy Knihy“***, viz Obrázek 287 - Rozšíření knihy – Deploy. Po ukončení akce je nutné restartovat K2.

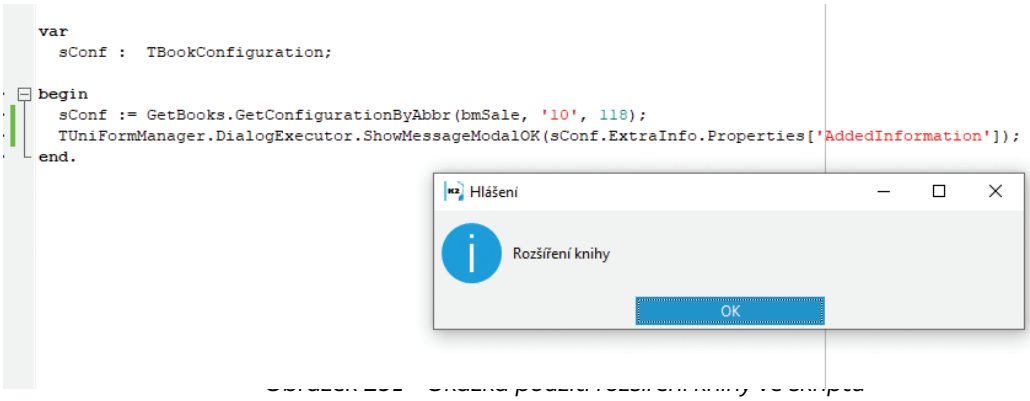
Nyní můžeme vyzkoušet rozšíření přímo ve správě knih a také si ukážeme použití ze skriptu K2. Spustíme tedy ***„Správu knih“***, například ze stromového menu uzel ***„Základní data“*** a protože jsme rozšiřovali knihy prodeje, tak vybereme ***„Knihy prodeje“***. Dvojitým kliknutím nebo klávesou Enter otevřeme knihu s označením ***„10“*** v detailním nastavení. Zobrazí se formulář, viz Obrázek 290 - Ukázka rozšíření knihy Prodeje, kde můžeme vidět přidaná pole ***„Doplňková informace“*** a ***„Zodpovědná osoba“***. Rozšíření je do formuláře přidáno automaticky, nemusíme se tedy starat o

jeho úpravu. Jako doplňkovou informaci uvedeme hodnotu ***„Rozšíření knihy“*** a jako zodpovědnou osobu vybereme z číselníků kontaktních osob ***„Pavel Motan“***. Tlačítkem OK uložíme doplněné hodnoty.



Obrázek 290 - Ukázka rozšíření knihy Prodeje

Na závěr si ještě vyzkoušíme načtení hodnot z rozšíření pomocí K2 skriptu. Založíme si nový skript. Pomocí funkce ***„GetBooks“*** načteme všechny knihy v K2 a nad tímto seznamem zavoláme funkci ***„GetConfigurationByAbbr“***, která slouží k získání konfigurace knihy, kterou musíme vyspecifikovat v parametrech funkce. V našem případě se jedná o knihy prodeje, prvním parametrem tedy bude hodnota ***„bmSale“***, která je výčtovým typem určující knihy prodeje. Druhým parametrem je zkratka knihy, pro kterou chceme načíst konfiguraci. V našem příkladu se jedná o hodnotu ***„10“***. Poslední parametr určuje období. Identifikátor ***„118“*** je v našem příkladu rok 2018. Výsledek funkce si uložíme do proměnné ***„SConf“*** typu ***„TBookConfiguration„***. Rozšiřující pole pak najdeme ve vlastnosti ***„ExtraInfo“***, dále v ***„Properties“***. K hodnotě přistoupíme pomocí zápisu s hranatými závorkami a identifikátorem pole – ***„Properties[AddedInformation]“***. Výslednou hodnotu zobrazíme pomocí hlášení ***„Information“***. Po spuštění se zobrazí hlášení s hodnotou ***„Rozšíření knihy“*** tak jak jsme zadali do knihy s označením ***„10“***. Vše je vidět na Obrázek 291 - Ukázka použití rozšíření knihy ve skriptu.



8. ZÁLOŽKA „TABULKY“

V této části Návrháře objektů můžeme rozšiřovat standardní tabulky o tzv. EX políčka a indexy. Tato záložka vlastně nahrazuje moduly ***„Rozšíření souborových modulů“*** a ***„Rozšíření souborových modulů CONF“***.

Důležité je, že v K2 může být použit jenom jeden způsob rozšíření standardních modulů, a to buď pomocí Rozšíření souborových modulů nebo pomocí Návrháře objektů. Přepnutí do režimu rozšiřování tabulek pomocí NO nelze již vrátit zpět. Tento stav je ale žádoucí. Je cílem mít vše v Návrháři objektů.

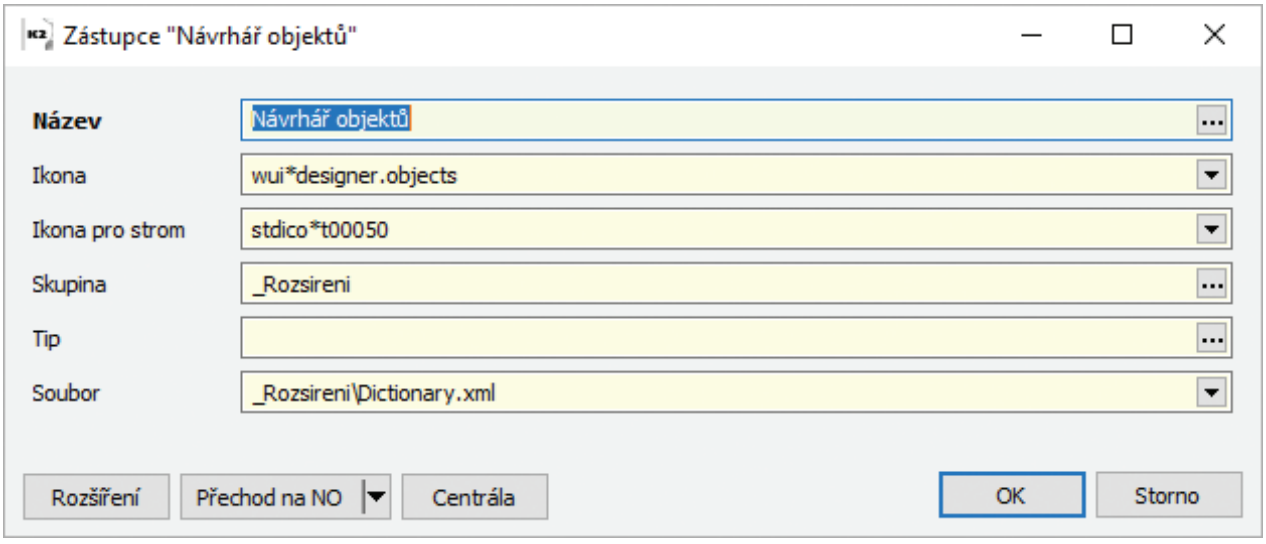
8.1. PŘEPNUTÍ DO REŽIMU ROZŠÍŘENÍ POMOCÍ NO

Při přepnutí do režimu rozšíření pomocí NO je nutné si uvědomit, zda máme K2 rozšířenou o speciální EX pole pomocí modulů ***„Rozšíření souborových modulů“*** nebo ***„Rozšíření souborových modulů CONF“***. Pokud ano, při přepnutí dojde ke konverzi ze souboru TableMod.xml (kde jsou uloženy informace k daným polím nebo indexům). Pro samotné přepnutí doporučujeme se poradit se servisním týmem.

Konverzí proběhne sjednocení EX polí a indexu ze všech mandantu do jedné definice NO. Výsledkem budou stejná EX_ pole a indexy na všech mandantech.

Před samotným přepnutím je důležité si vytvořit zálohy, mít K2 bez chyb, tzn. musí být čistá jak kontrola struktur, tak diagnostika. Pokud se objeví chyba, musí se opravit, jinak nelze přepnout K2 do režimu rozšíření pomocí NO.

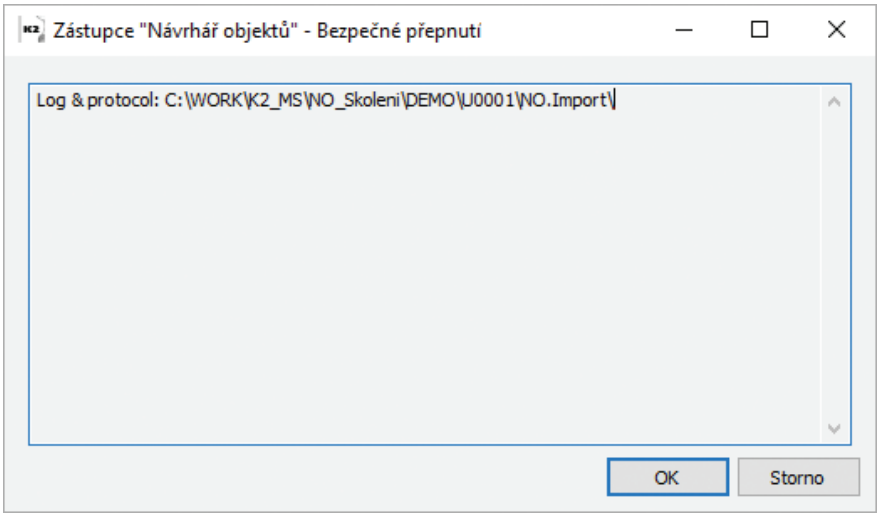
Přepnutí do režimu rozšiřování tabulek pomocí Návrháře objektů lze pomocí pár kliků. V shortcutu pro NO klikněte na možnost ***„upravit“***, kde se zobrazí následující formulář jako na obr. Obrázek 292 - Nastavení shortcutu NO.



Obrázek 292 - Nastavení shortcutu NO

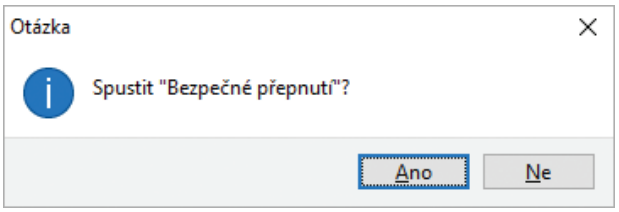
Pomocí volby Přechod na NO lze vybrat možnost ***„Bezpečné přepnutí“***, která zkontroluje, zda je možné bezpečně přejít na rozšíření pomocí NO, tzn. zkontroluje se např. zda se nevyskytuje při více mandantech konflikt ve speciálních EX polích. Po kontrole se objeví formulář s informacemi, jak můžeme vidět např. na obr. Obrázek 293 - Informace po kontrole při spuštění "Bezpečné přepnutí".

Možnost ***„TableMod.xml -> NO“***, která se nachází ve volbě Přechod na NO, a možnost ***„Centrála“*** slouží pro zkušené servisní týmy a není součástí tohoto školení, nebudeme se jí tedy zabývat.



Obrázek 293 - Informace po kontrole při spuštění "Bezpečné přepnutí"

Pokud je vše v pořádku, můžeme pokračovat pomocí tlačítka OK. Následně se nám objeví dotaz, zda chceme spustit ***„Bezpečné přepnutí“*** (Obrázek 294 - Dotaz na spuštění "Bezpečního přepnutí"). Po potvrzení se nám zahájí případná konverze TableModu.xml a IS K2 se přepne do režimu ***„Rozšíření pomocí NO“***. Po operaci ***„Deploy“*** se automaticky IS K2 sám restartuje (vypne). Stačí posléze IS K2 zapnout a již budeme moc rozšiřovat standardní pole pomocí Návrháře objektů a IS K2 bude automaticky přepnut do režimu ***„Rozšíření pomocí NO“*** – tento krok již nelze vrátit.

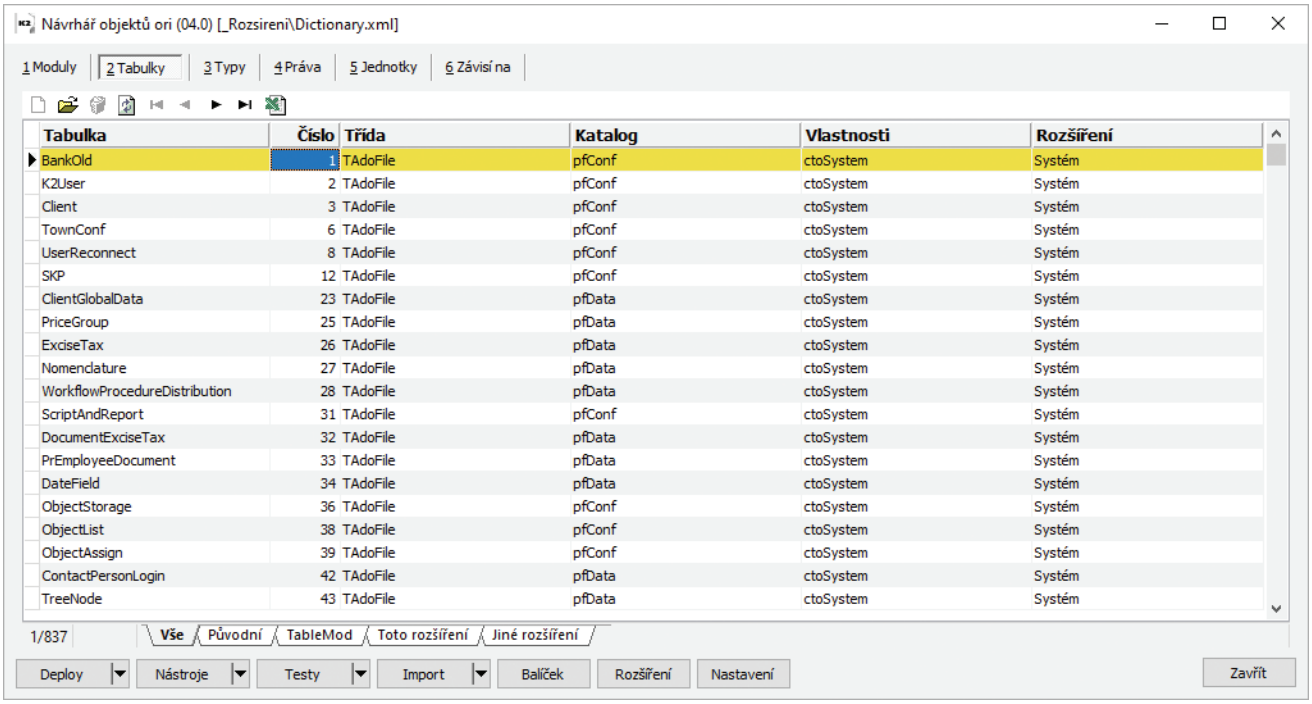


Obrázek 294 - Dotaz na spuštění "Bezpečního přepnutí"

8.2. ROZŠÍŘENÍ TABULEK POMOCÍ NO

Pro rozšíření tabulek pomocí NO, je důležité mít zapnutou tuto možnost, viz předchozí kapitola 9.1. Použití výčtového typu v datovém modulu.

V záložce ***„Tabulky“*** se vyskytují všechny standardní tabulky, které lze rozšířit.



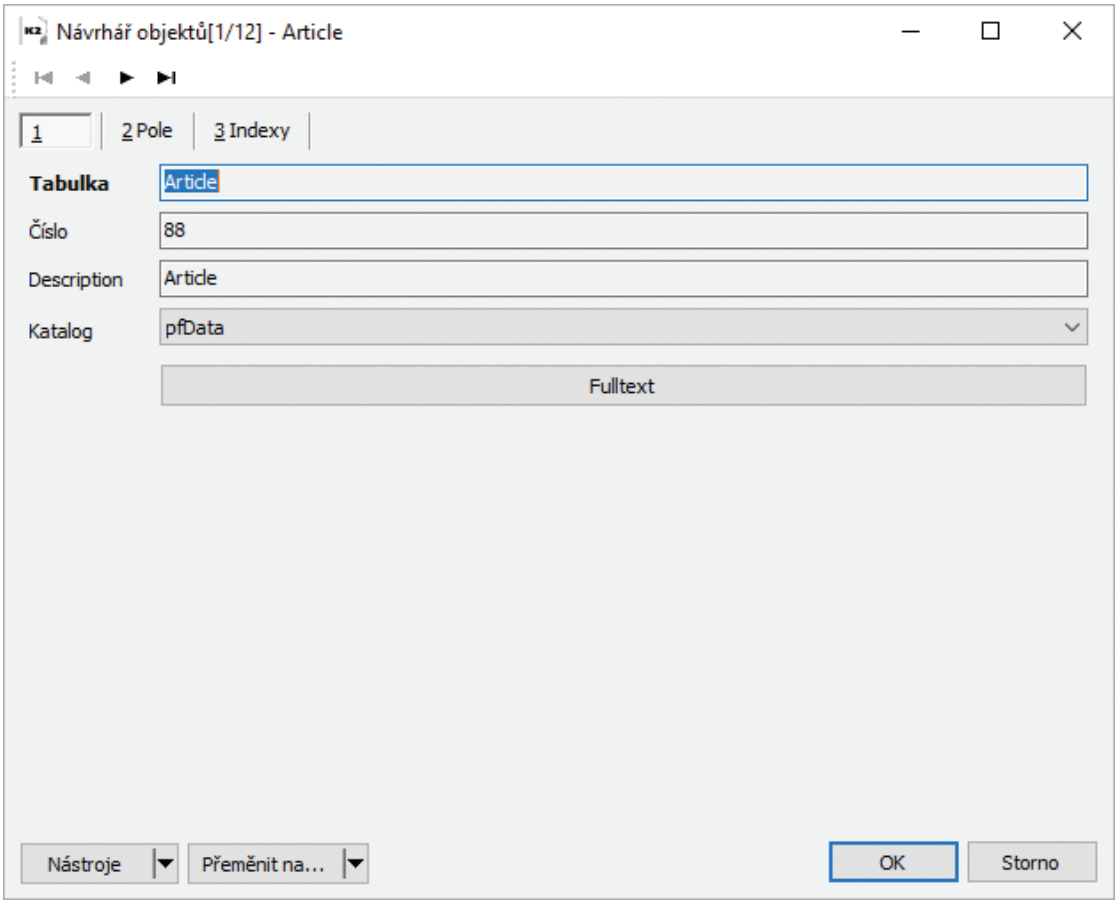
Obrázek 295 - Záložka "Tabulky"

Nyní si popíšeme jednotlivé záložky ve spodní části:

- Vše – jsou zobrazeny všechny tabulky.
- Původní – jsou zobrazeny tabulky, které nejsou rozšířeny o pole, nebo indexy.
- TableMod – jsou zobrazeny tabulky, které jsou rozšířeny pomocí modulů ***„Rozšíření souborových modulů“*** nebo ***„Rozšíření souborových modulů CONF“***.
- Toto rozšíření – jsou zobrazeny tabulky, které jsou rozšířeny v momentálně otevřeném NO.
- Jiné rozšíření – jsou zobrazeny tabulky, které jsou rozšířeny v jiném než momentálně otevřeném NO.

8.2.1. ZÁKLADNÍ INFORMACE TABULKY

Při výběru tabulky, kterou chceme rozšířit se nám zobrazí formulář, kde na 1. straně jsou základní informace k vybrané tabulce.



Obrázek 296 - Základní informace tabulky

Popis polí:

- ***Tabulka*** – název tabulky,
- ***Číslo*** – číslo tabulky,
- ***Description*** – popis tabulky,
- ***Katalog*** – informace, zda se jedná o tabulku mandanta (pfData), nebo conf (pfConf),
- ***Fulltext*** – tato funkčnost je zatím ve vývoji.

8.2.2. POLE

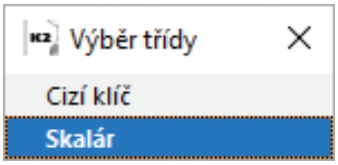
V této záložce se nacházejí standardní i speciální pole. Lze zde zakládat jak pole typu skalár, tak cizí klíč. Ve spodní části formuláře se nachází opět záložky ***„Vše“***, ***„Původní“***, ***„TableMod“***, ***„Toto rozšíření“*** a ***„Jiné rozšíření“***. Význam těchto záložek je stejný jako u záložky ***„Tabulky“***.

Dále zde můžeme najít tlačítko Nástroje a Přeměnit na... Níže si popíšeme co jednotlivé volby znamenají.

Nástroje
Pod touto možností si můžeme zobrazit Náhled dané tabulky.

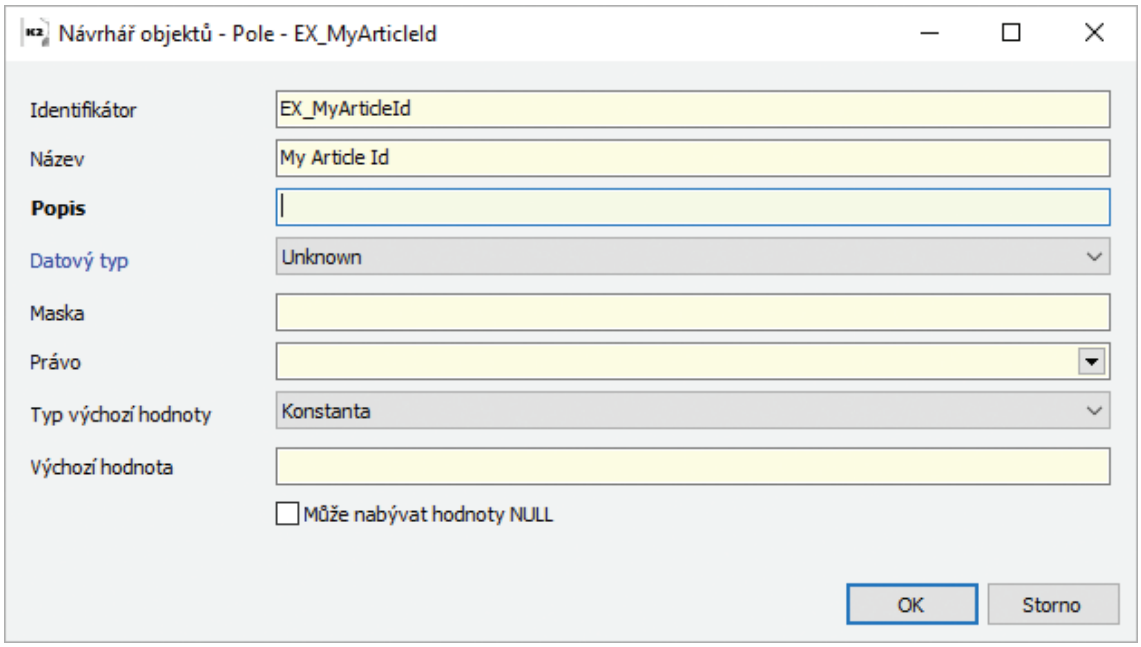
Přeměnit na...
Funguje zde stejná možnost záměny pole buď to skalár na cizí klíč, nebo opačně. Více jsme se o této možnosti dozvěděli v kapitole 4.2.4. Rychlá změna typu pole – funkce „Přeměnit na“.

Nové pole založíme pomocí klávesy Insert nebo tlačítka pro Nový záznam, kde máme na výběr buď ***„Cizí klíč“*** nebo ***„Skalár“***. Obě možnosti již byly detailněji popsány v kapitole 4.2. Pole.



Obrázek 297 - Založení nového pole

Skalár
Po výběru této možnosti se nám zobrazí formulář, jak můžeme vidět na obrázku Obrázek 298 - Vytvoření pole typu skalár.

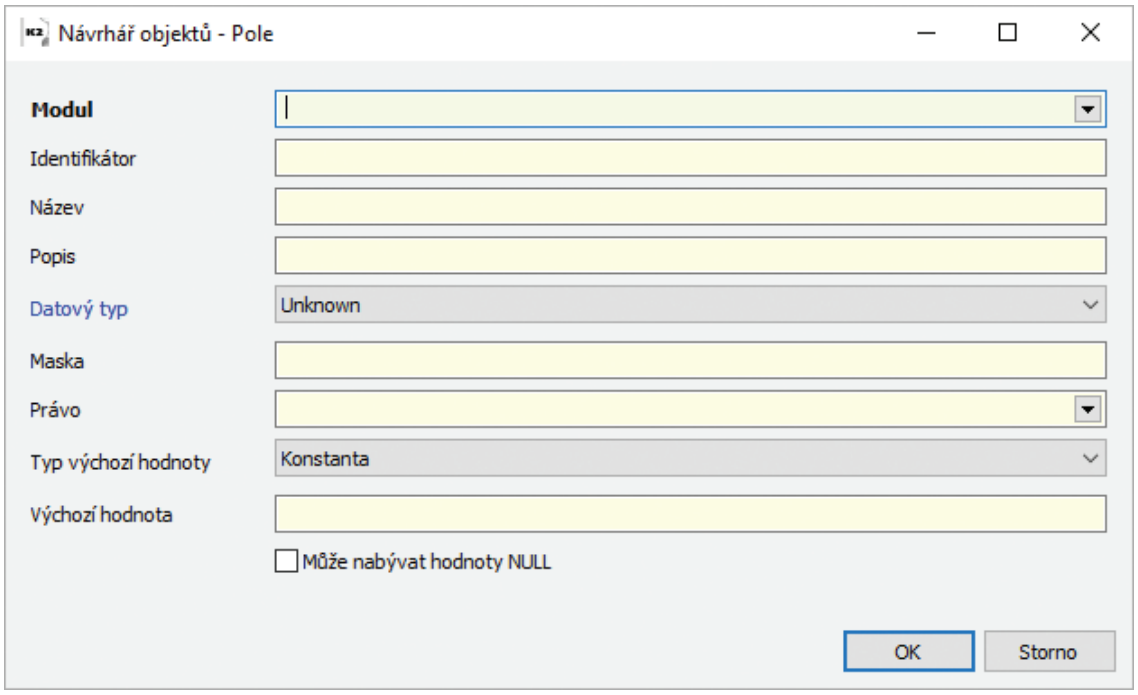


Obrázek 298 - Vytvoření pole typu skalár

Nyní se popíšeme jednotlivé komponenty:

- ***Identifikátor*** – identifikace pole v rámci dané tabulky. Po potvrzení se nám automaticky před identifikátor doplní prefix _EX.
- ***Název*** – název pole.
- ***Popis*** – popis pole.
- ***Datový typ*** – datový typ pole. Dokud je vybrána možnost ***„Unknown"***, nelze změny uložit.
- ***Maska*** – maska pro dané pole (např. S30 pro textové pole o délce 30 znaků).
- ***Právo*** – výběr práva pro dané pole.
- ***Typ výchozí hodnoty*** – implicitně nastavena ***„Konstanta"***. Lze vybrat i hodnoty ***„Výraz"***.
- ***Výchozí hodnota*** – touto property lze nastavit poli výchozí hodnotu.
- ***Může nabývat hodnoty NULL*** – informace, že pole může nabývat hodnoty NULL.

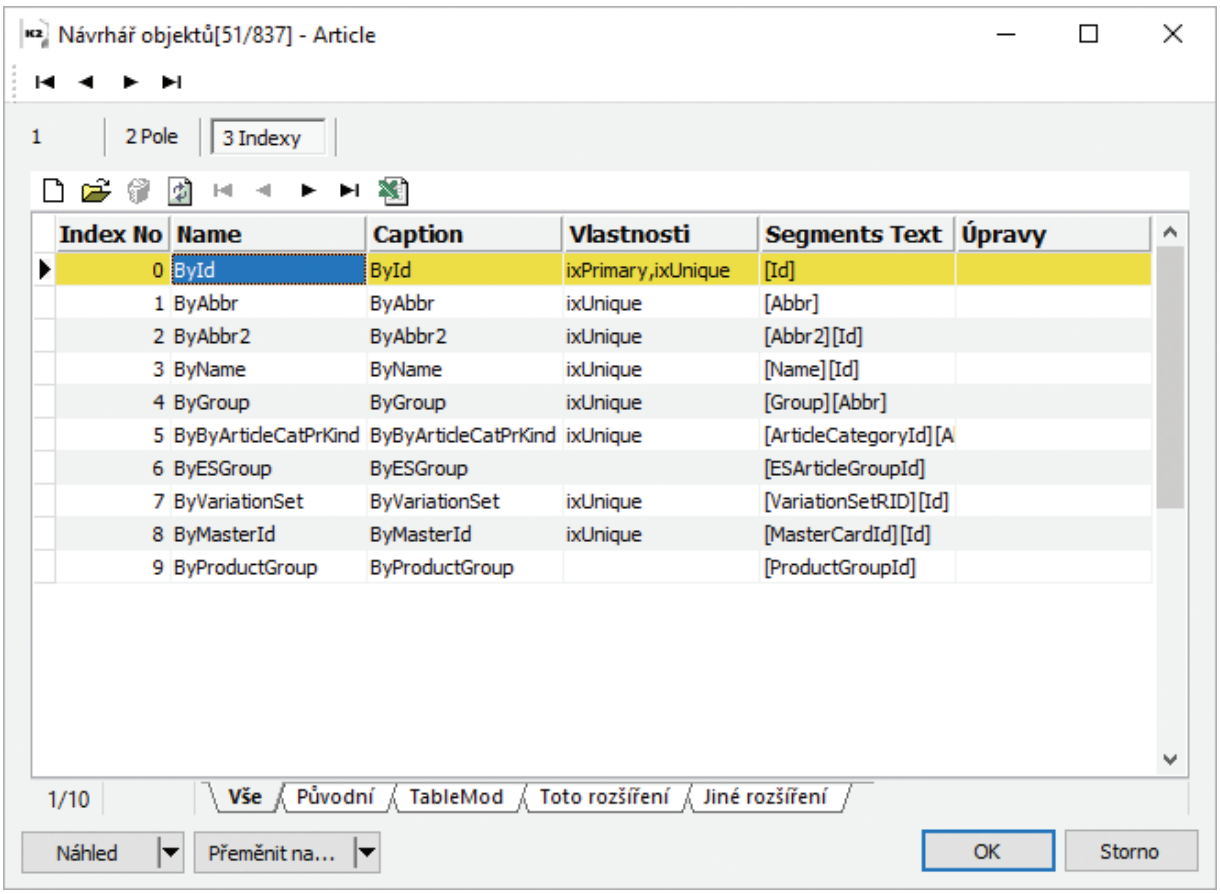
Cizí klíč
Pomocí této volby můžeme vytvořit vazbu do jiného datového modulu. Nemusí se již používat registrované funkce, Návrhář objektů si vše připraví sám. Formulář se moc neliší od volby ***„Skalár"***, je zde pouze navíc property ***„Modul"*** (slouží pro výběr modulu, do kterého chceme vytvořit vazbu). Všechny ostatní volby jsou totožné se Skalárem.



Obrázek 299 - Vytvoření pole typu Cizí klíč

8.2.3. INDEXY

Na této záložce se nacházejí všechny indexy (jak standardní, tak speciální) k dané tabulce. Popíšeme si nyní jednotlivé sloupce, které můžeme vidět na obrázku Obrázek 300 - Záložka indexy.



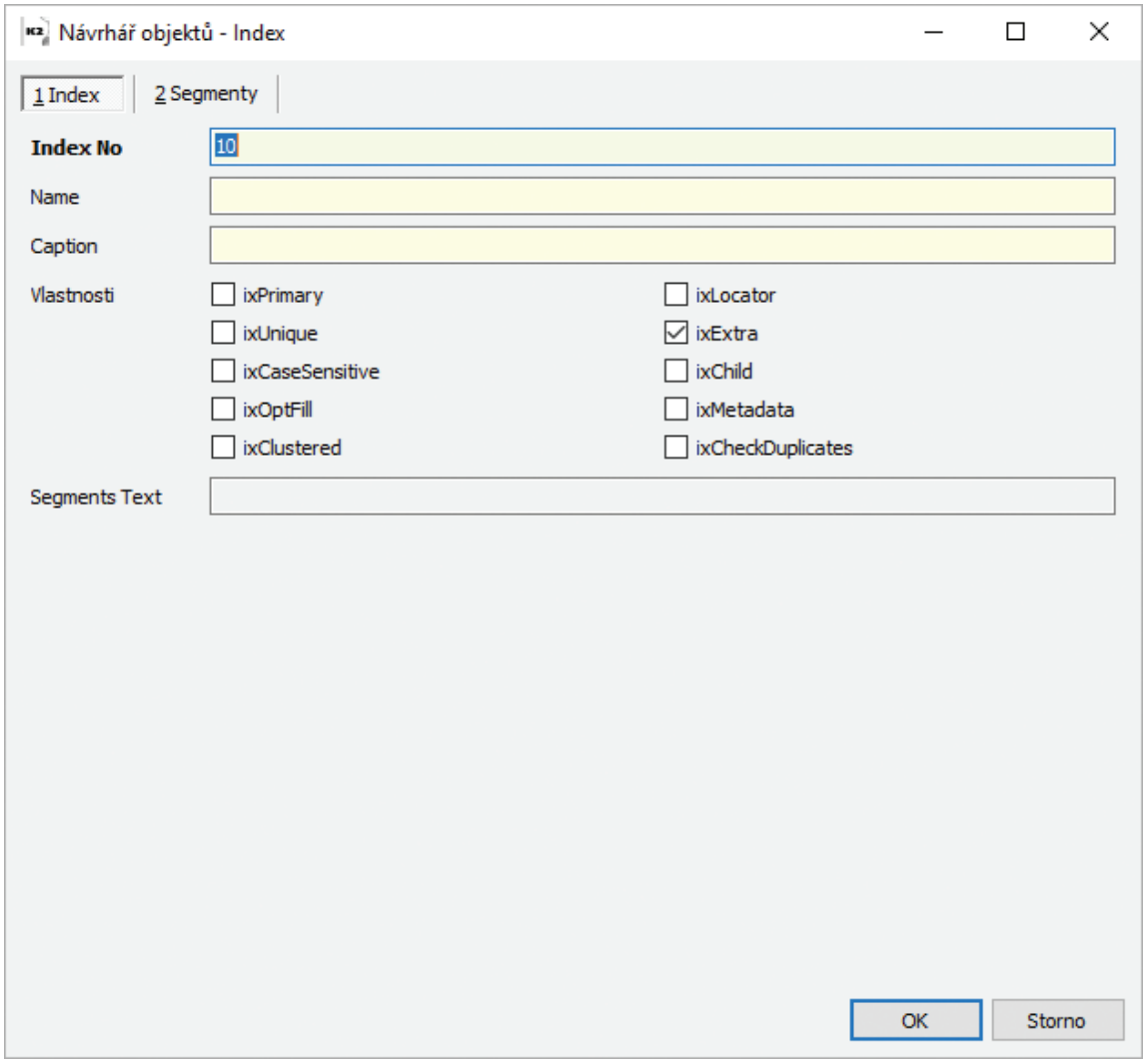
Obrázek 300 - Záložka indexy

Popis sloupců:

- *Index NO* – číslo indexu,
- *Name* – jméno indexu,
- *Caption* – titulek,
- *Vlastnosti* – výčet vlastností indexu,
- *Segments text* – výčet segmentů obsahující daný klíč,
- *Úprava* – kde se nachází úprava vytvořeného indexu.

Nový index založíme buď pomocí klávesy Insert, nebo pomocí tlačítka pro Nový záznam. Zobrazí se nám formulář (), který se skládá ze dvou záložek Index a Segmenty.

Index
Tato záložka slouží k popisu indexu a vyjmenování jeho vlastností.



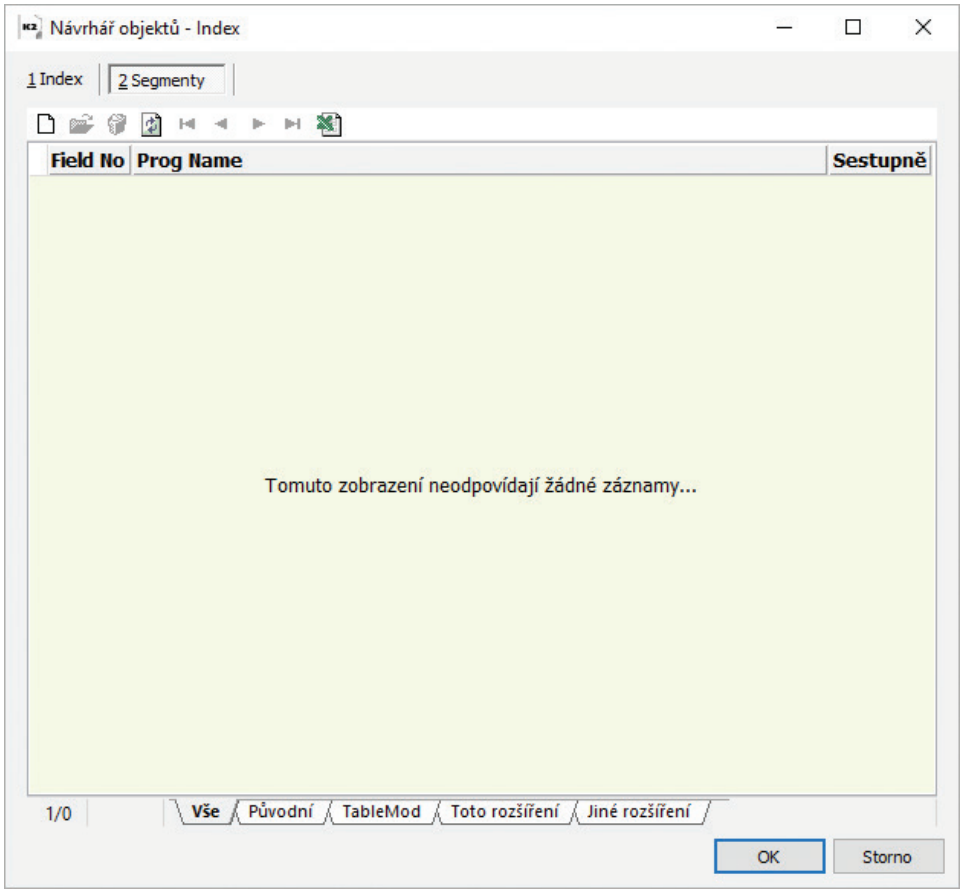
Obrázek 301 - Vytvoření indexu, záložka Index

Popis polí:

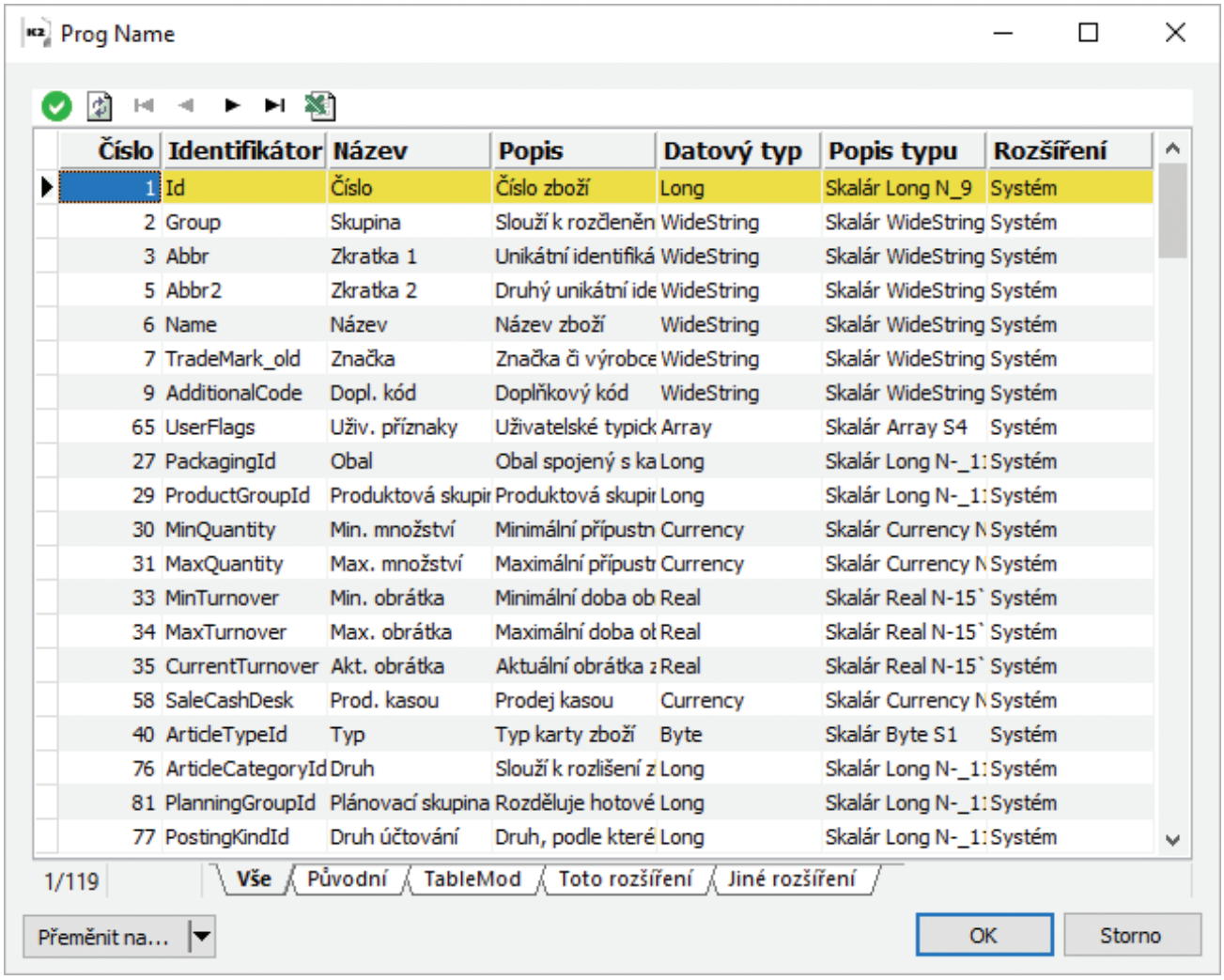
- *Index NO* – číslo indexu, lze jej změnit, ale doporučujeme nechat číslování na NO.
- *Name* – jméno indexu, doporučujeme držet se prefixem By + segmenty.
- *Caption* - titulek indexu.
- *Vlastnosti* – vlastnosti indexu. Všechny důležité vlastnosti jsme si již vysvětlili v kapitole 4.4. Klíče. Níže jsou vypsány vlastnosti s jednoduchým popisem pro zopakování:
  1. ixPrimary – primární index,
  2. ixUnique – unikátní index,
  3. ixCaseSensitive – jedná se o interní záležitost, kterou využívají programátoři,
  4. ixOptFill – jedná se o interní záležitost, kterou využívají programátoři,
  5. ixClustered – jedná se o interní záležitost, kterou využívají programátoři,
  6. ixLocator – lokátor indexu,
  7. ixExtra – jedná se o speciální index, automaticky zatrženo,
  8. ixChild – jedná se o interní záležitost, kterou využívají programátoři,
  9. ixMetadata – jedná se o interní záležitost, kterou využívají programátoři.
  10. ixCheckDuplicates – kontrola duplicit.
- *Segments Text* – textové pole, které vypisuje segmenty klíče.

Segmenty

Tato záložka slouží k vybrání segmentů (polí) pro daný index. Nový segment přidáme buď pomocí tlačítka pro Nový, nebo klávesovou zkratkou Insert. Ve zobrazeném formuláři vybereme jednotlivé segmenty indexu. Nelze vkládat hromadně.



Obrázek 302 - Vytvoření indexu, záložka Segmenty



Obrázek 303 - Vytvoření indexu, vložení segmentu

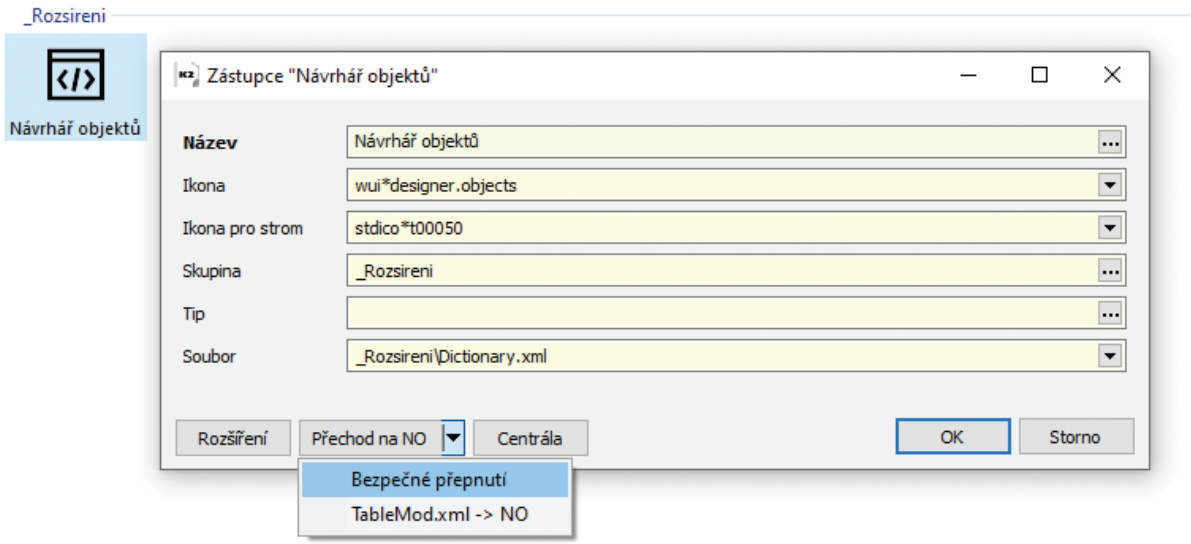
POZNÁMKA: Po vytvoření změn, je nutné tyto změny aplikovat pomocí operace ***„Deploy“***, kterou budeme probírat podrobněji později.

PŘÍKLAD: Pro ukázku si založíme e v tabulce pro zboží (Article) 3 vlastní pole:

- SpecialNumber – WideString[30],
- NewCodeNumber – BigInt,
- ResponsiblePerson – cizí klíč do kontaktních osob.

A poté založímena pole SpecialNumber a NewCodeNumber index.

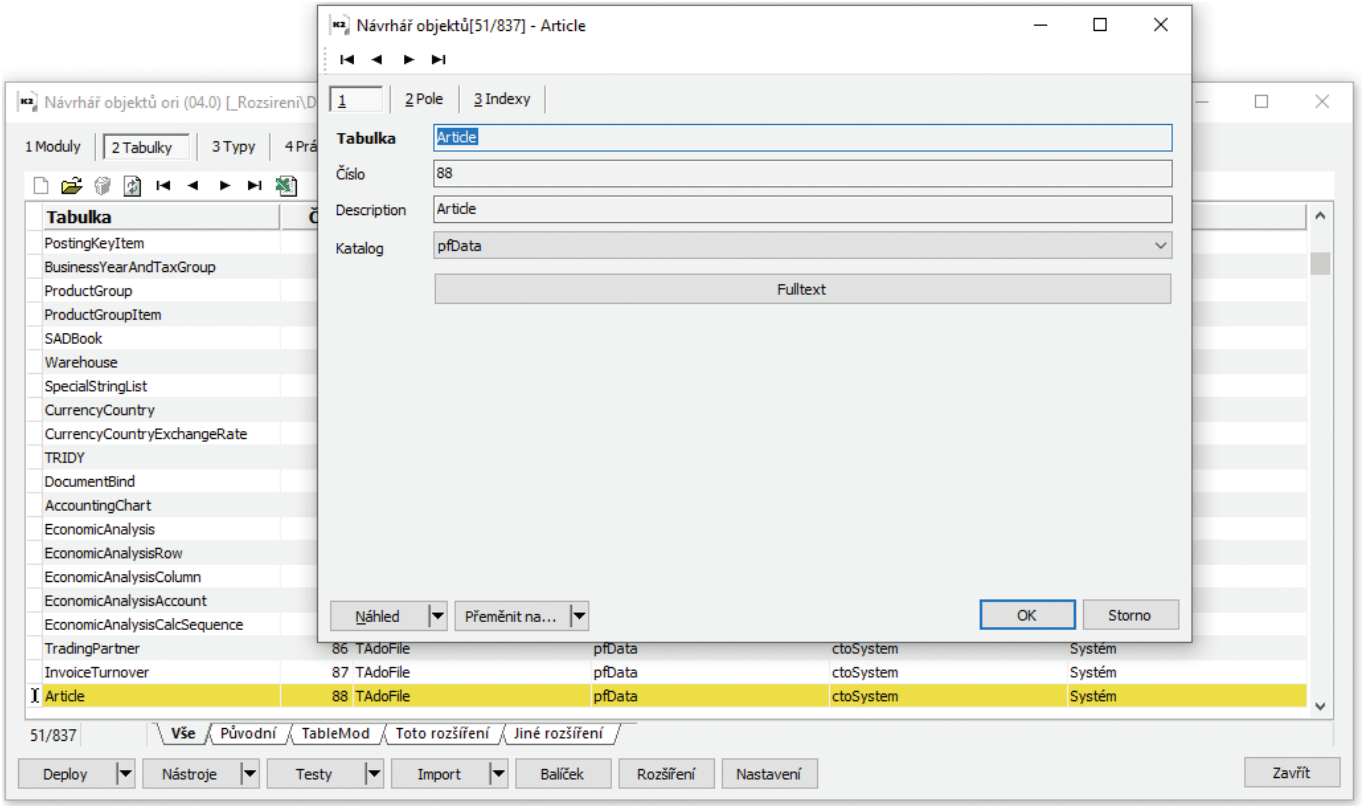
Nejprve si musíme IS K2 přepnout do režimu rozšíření tabulek pomocí NO. Klikneme na možnost Upravit na shortcutu Návrháře objektů a následně v ***„Přechod na NO“*** zvolíme možnost bezpečného přepnutí, jak je vidět na obrázku Obrázek 304 - Přepnutí K2 do režimu rozšíření tabulek pomocí NO.



Obrázek 304 - Přepnutí K2 do režimu rozšíření tabulek pomocí NO

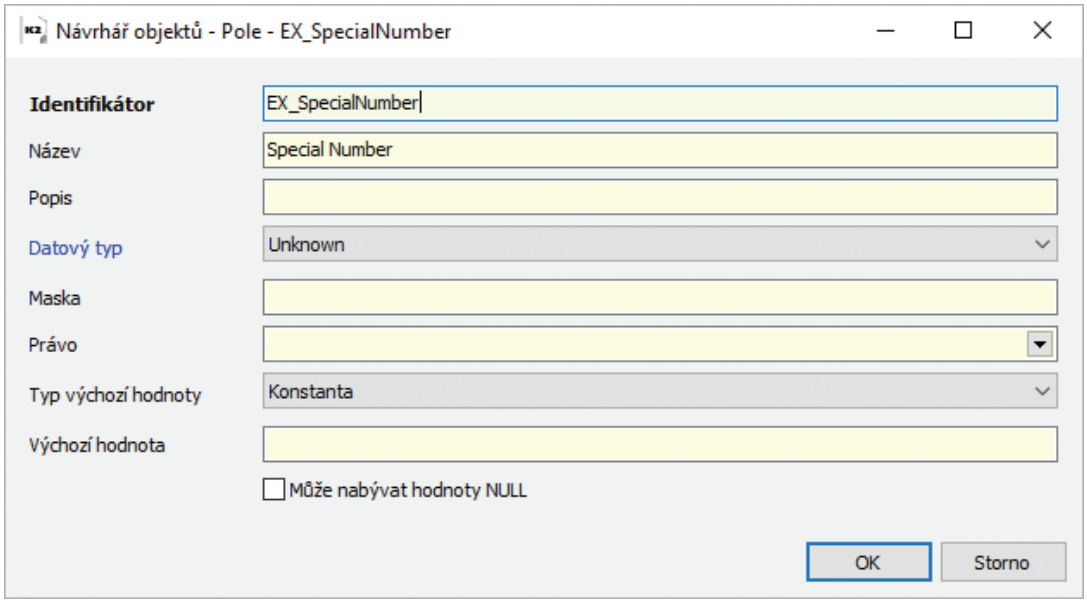
Po akci „Bezpečné přepnutí“ se nám IS K2 restartuje, po zapnutí budeme již v režimu ***„Rozšíření pomocí NO“***.

V NO se přepneme na záložku ***„Tabulky“*** a najdeme tabulku Article (viz obrázek Obrázek 305 - Založení polí v tabulce Article).



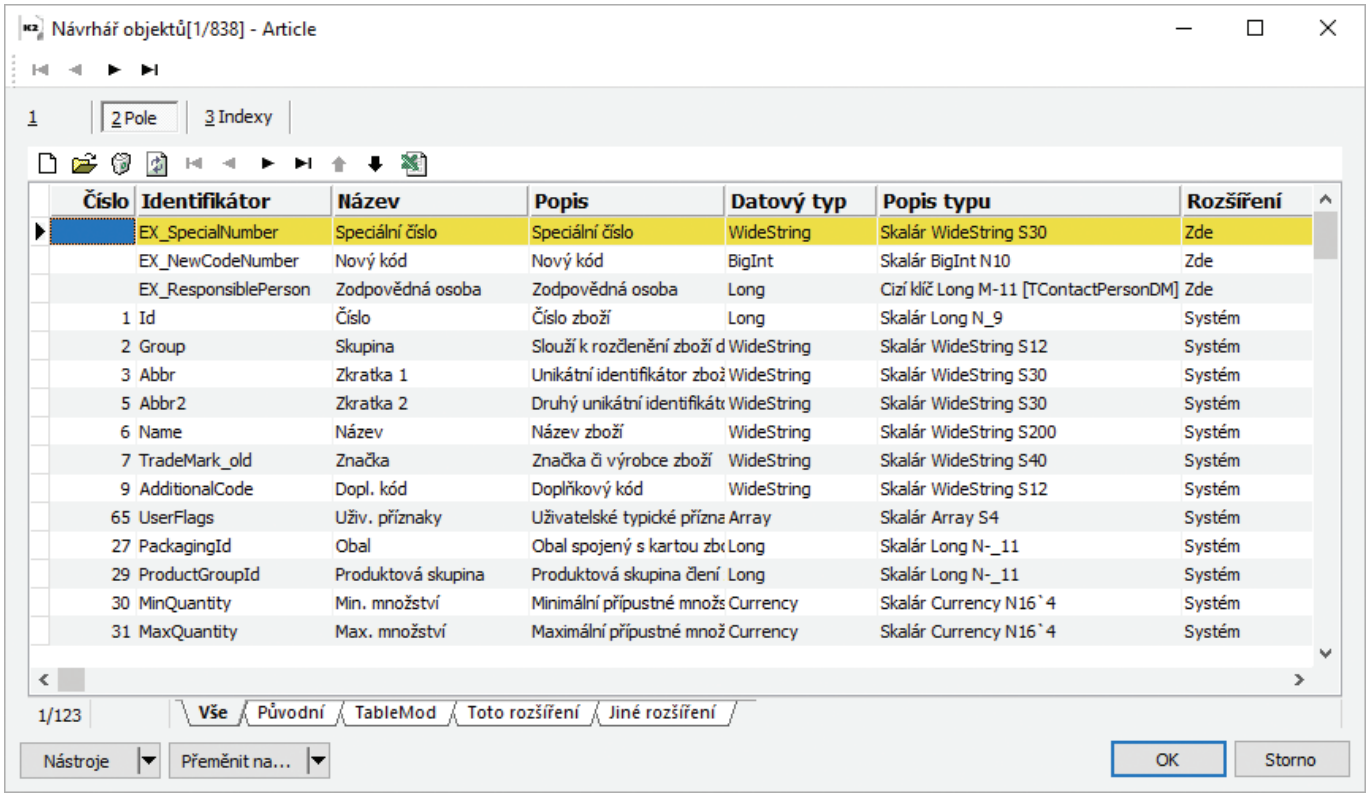
Obrázek 305 - Založení polí v tabulce Article

V záložce pole postupně založíme všechny tři pole. Pole SpecialNumber a NewCodeNumber budou vytvořeny pomocí volby skalár a pole ResponsiblePerson jako cizí klíč do skladů. Můžete si všimnout, že pokud zadáte do identifikátoru jenom název pole (např. SpecialNumber), tak se po potvrzení pomocí klávesy Enter automaticky přidá prefix ***„EX_“***.



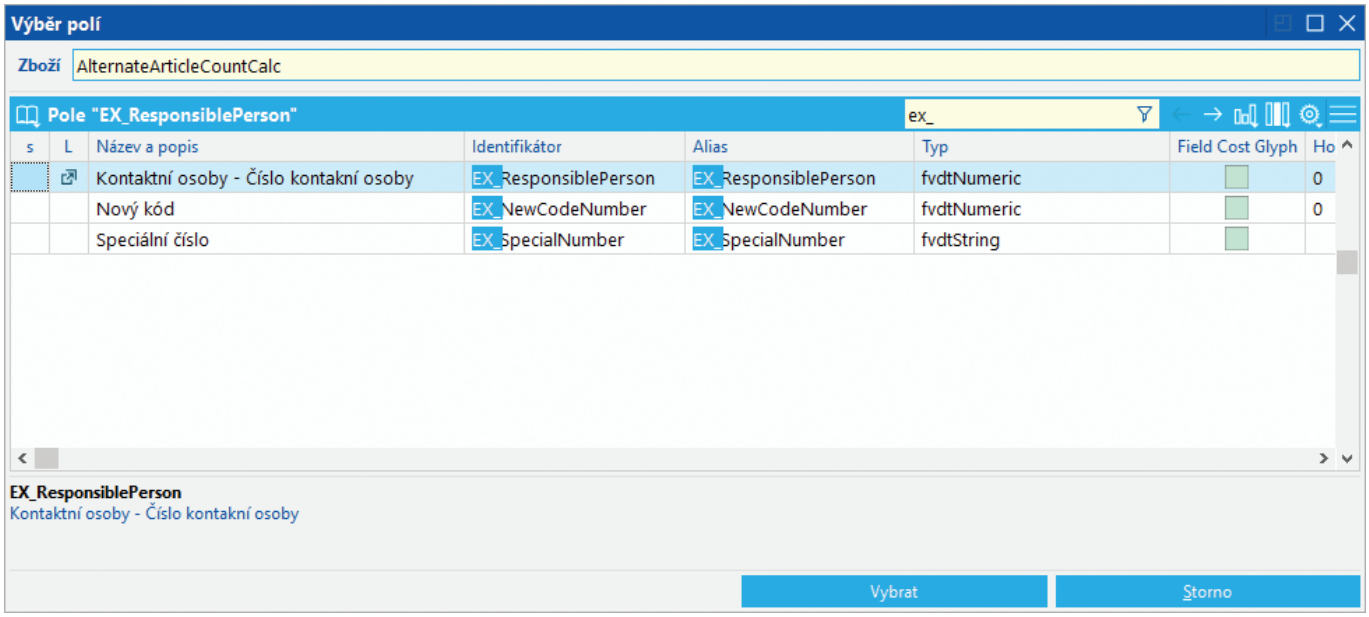
Obrázek 306 - Založení pole SpecialNumber

Při zakládání polí nevybíráme žádné právo ani výchozí hodnotu. U textového pole nesmíme zapomenout vyplnit jeho délku 30 znaků. Po založení polí byste měli mít výsledek, jako je na obrázku Obrázek 307 - Založení polí v tabulce Article. Následně potvrdíme a spustíme akci „Deploy“.



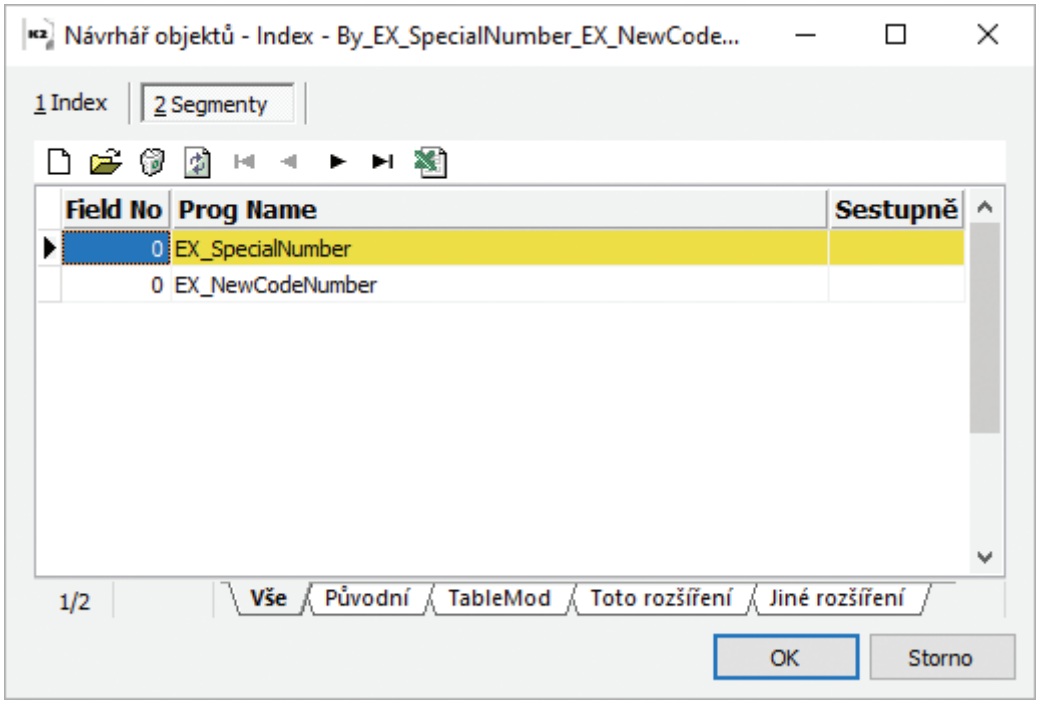
Obrázek 307 - Založení polí v tabulce Article

V IS K2 zkontrolujeme, že se nám opravdu nové políčka založily do tabulky Article tím, že si otevřeme zboží a přes přidání nových sloupečků vidíme, že nám IS K2 již tyto pole nabízí (Obrázek 308 - Kontrola založených polí).



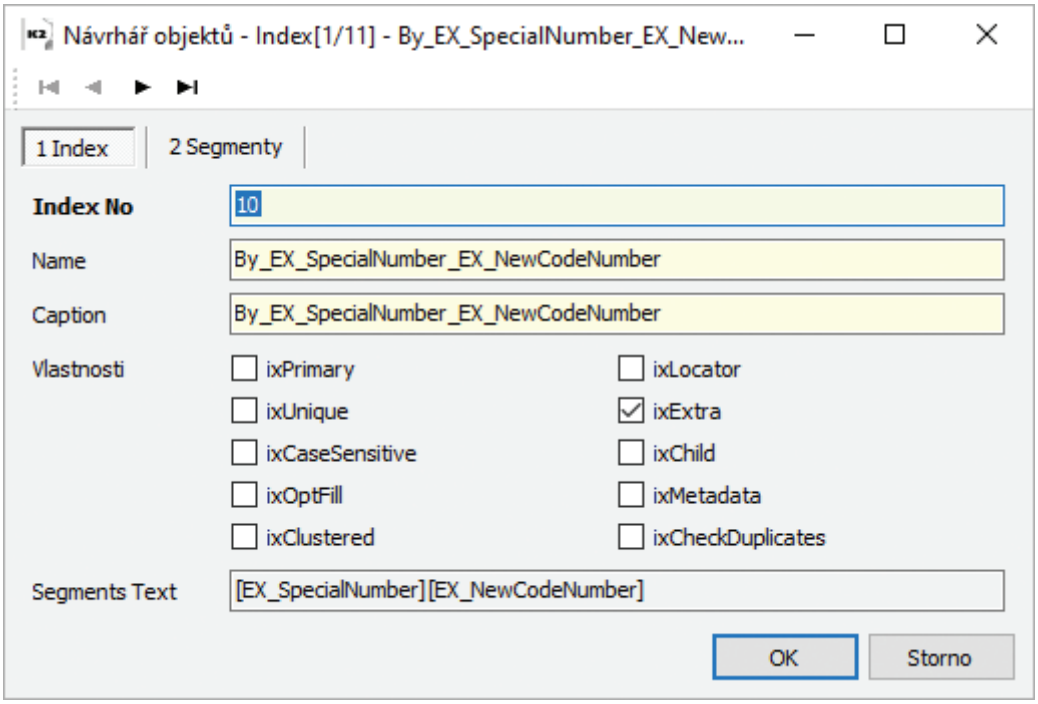
Obrázek 308 - Kontrola založených polí

Jako druhý krok si vyzkoušíme vytvořit index. Otevřeme opět NO, přepneme se na stranu tabulky, najdeme tabulku Article(zboží) a v ní se přepneme do záložky ***„Indexy“***. Vytvoříme nový index např. pomocí klávesy Insert. Jako segmenty vybereme námi vytvořené dvě pole EX_SpecialNumber a EX_NewCodeNumber (Obrázek 309 - Vytvoření vlastního klíče - vložení segmentů).



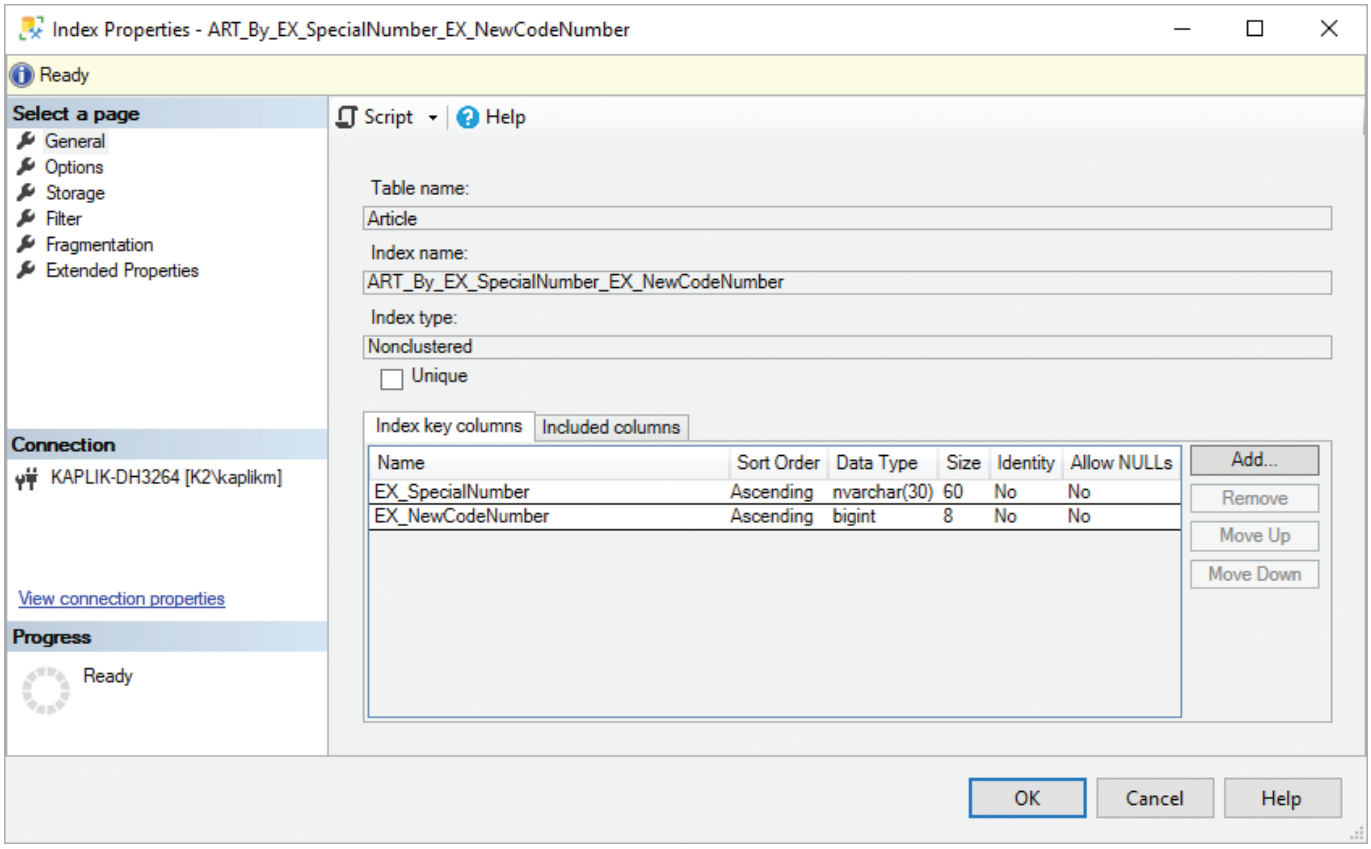
Obrázek 309 - Vytvoření vlastního klíče - vložení segmentů

Můžeme si všimnout, že na 1. straně se nám automaticky vyplnilo jméno indexu a zaškrtla se vlastnost ixExtra. Potvrdíme a spustíme ***„Deploy“***.



Obrázek 310 - Vytvoření vlastního klíče

Po ***„Deploy“*** si můžeme zkontrolovat v SQL nově vytvořený index (Obrázek 311 - Kontrola založeného indexu).



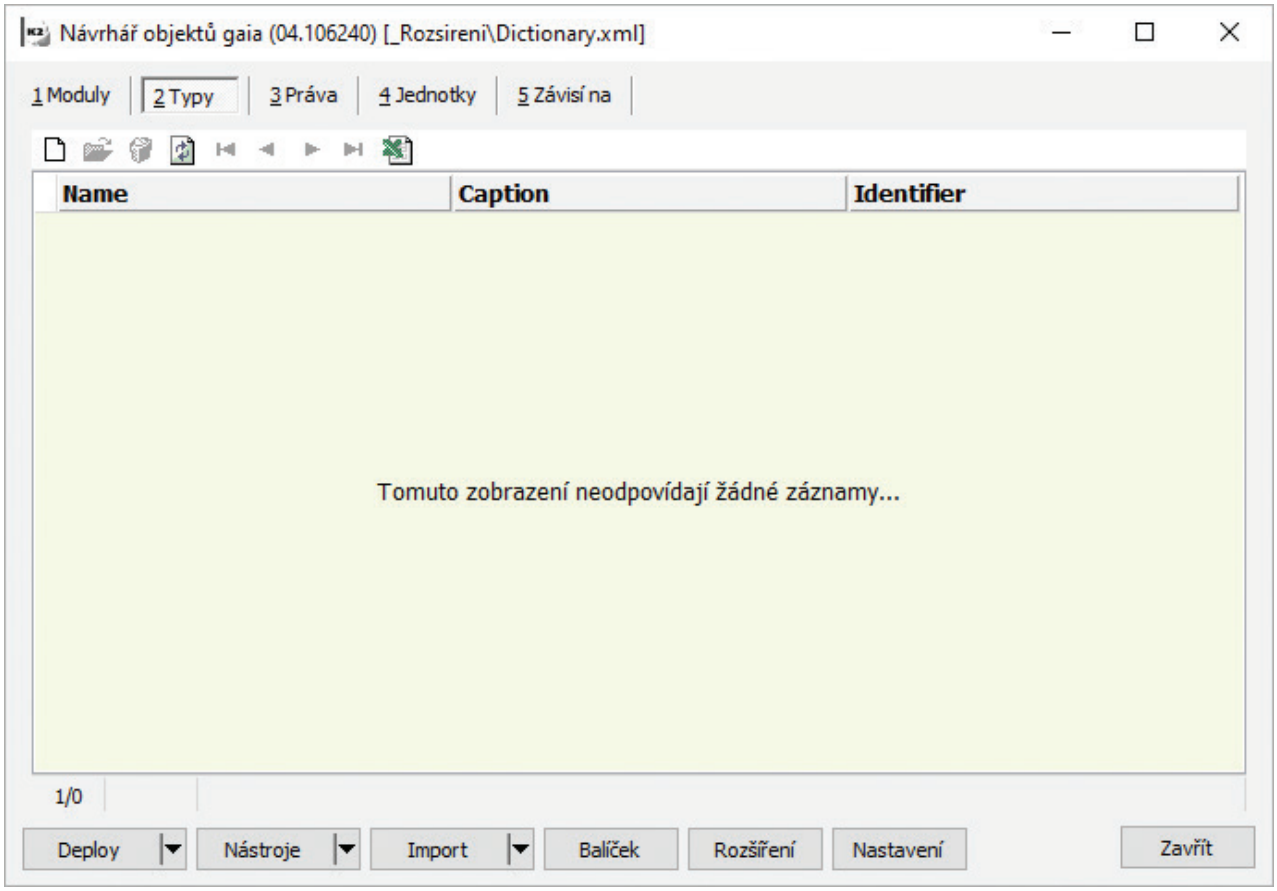
Obrázek 311 - Kontrola založeného indexu

9. ZÁLOŽKA „TYPY“

V této části Návrháře objektů můžeme definovat tzv. výčtové typy. Jedná se o množinu konstant, které mají svůj název (identifikátor) a také číslo, které mají přiřazeno. Výčtové typy se používají v případech, kdy máme pevnou množinu hodnot, u které jsme si jisti, že se nebude měnit, případně se mění velmi zřídka kdy. Jako příklad si můžeme představit výčtový typ, který slouží k rozlišení pohlaví člověka - ***„muž“*** / ***„žena“***, nebo výsledný stav hry pro jednotlivé hráče – ***„prohra, výhra, remíza“***.

Jak se definují a používají výčtové typy v Návrháři objektů, si ukážeme na následujícím popisu této funkčnosti.

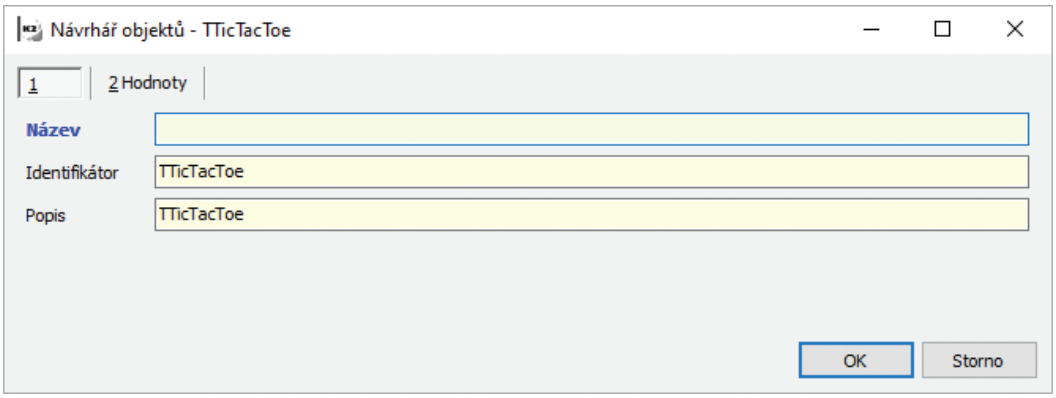
Nový typ založíme stisknutím klávesy Insert nebo stisknutím tlačítka Nový v nástrojové liště v záložce ***„Typy“***, viz Obrázek 312 - Seznam výčtových typů v rozšíření Návrháře objektů.



Obrázek 312 - Seznam výčtových typů v rozšíření Návrháře objektů

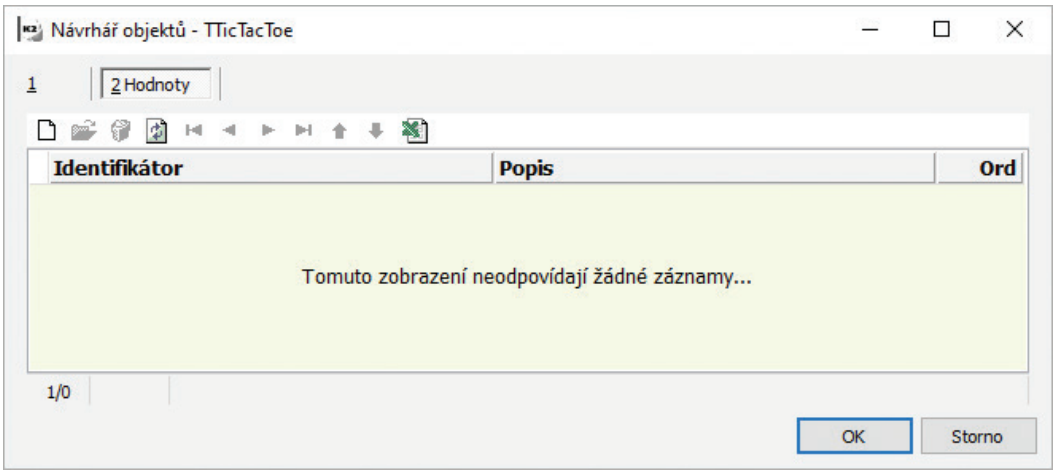
Zobrazí se formulář pro zadání nového typu. Na první záložce je potřeba vyplnit následující údaje. Je potřeba uvést název a popis pro typ a hlavně identifikátor, kterým se k výčtovému typu přistupuje ve skriptech K2, viz Obrázek 313 - Nový výčtový typ.

Identifikátor je doplněn o namespace rozšíření, který je definován a je ve tvaru ***„T{namespace}“***. Po určení názvu typ, je doplněn identifikátor i o tuto hodnotu, tedy ***„T{namespace}{název typu}“***. Doporučujeme tento identifikátor ponechat, aby byla zajištěna unikátnost napříč celou instalací K2.



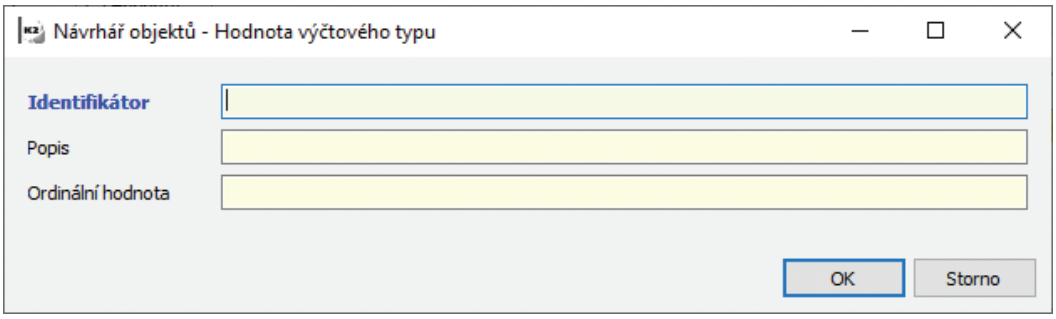
Obrázek 313 - Nový výčtový typ

Na druhé záložce se definují jednotlivé hodnoty pro vytvořený výčtový typ. Nový vložíme stisknutím klávesy Insert nebo tlačítka Nový v nástrojové liště nad seznamem, viz Obrázek 314 - Definice hodnot výčtového typu.



Obrázek 314 - Definice hodnot výčtového typu

Zobrazí se formulář pro založení nové hodnoty výčtového typu. Je nutné zadat identifikátor hodnoty výčtového typu a také popis. Ordinální hodnotu Návrhář objektů doplňuje automaticky postupným navyšováním. V případě potřeby je možné hodnotu změnit, viz Obrázek 315 - Definice hodnoty výčtového typu.

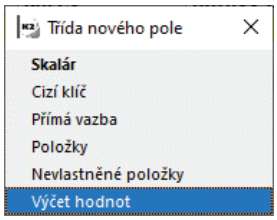


Obrázek 315 - Definice hodnoty výčtového typu

9.1. POUŽITÍ VÝČTOVÉHO TYPU V DATOVÉM MODULU

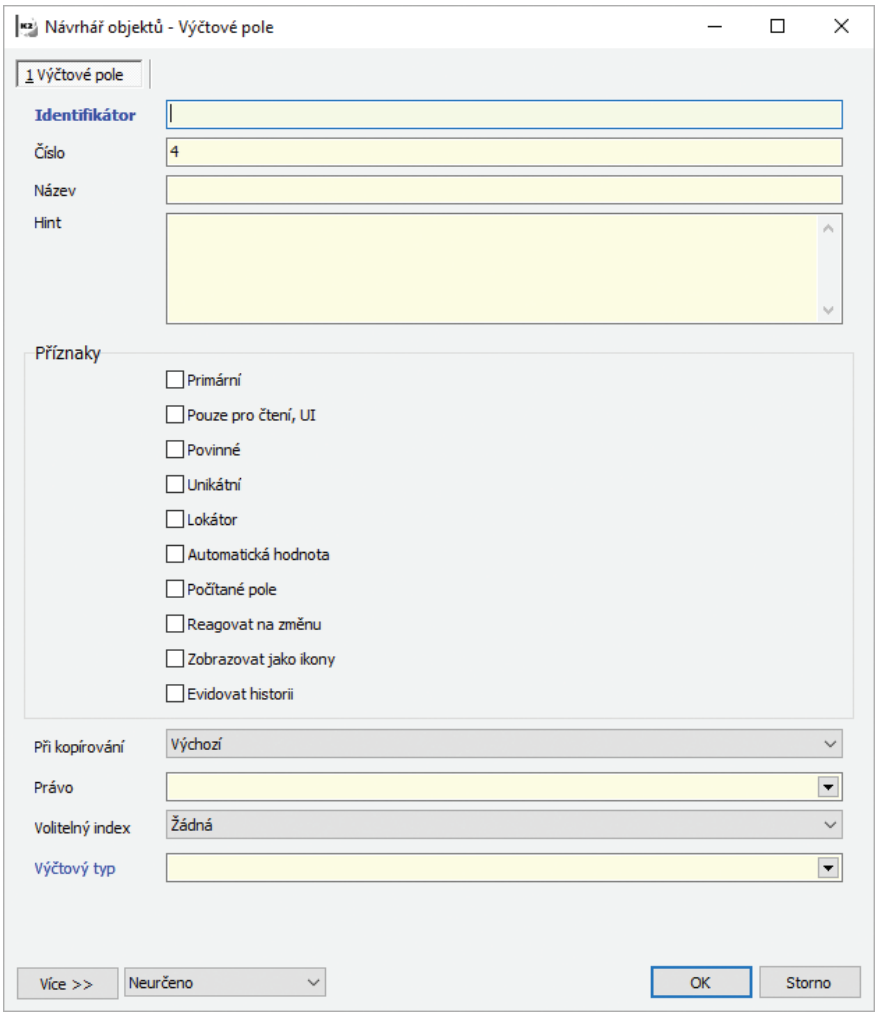
Definované výčtové typy se následně používají v datových modulech jako pole, která mají nastaven výčtový typ.

Pole vložíme do datového modulu stisknutím tlačítka Nový v nástrojové liště nebo stiskem klávesy Insert nad seznamem polí v datovém modulu. Zobrazí se výběr pro založení nového pole, kde vybereme hodnotu ***„Výčet hodnot“***, viz Obrázek 316 - Použití výčtového typu v datovém modulu.



Obrázek 316 - Použití výčtového typu v datovém modulu

Zobrazí se formulář pro vytvoření pole, které je založené na výčtovém typu. Formulář je téměř identický jako pro klasické pole datového modulu. Rozdílný je ve výběru datového typu, který v tomto případě není nutné zadávat. Vždy se jedná o celočíselnou hodnotu. Novým nastavením je hodnota ***„Výčtový typ“***. Zde vybereme ze seznamu výčtových typů, ten který požadujeme nastavit pro nové pole, viz Obrázek 317 - Pole typu výčtový typ v datovém modulu.

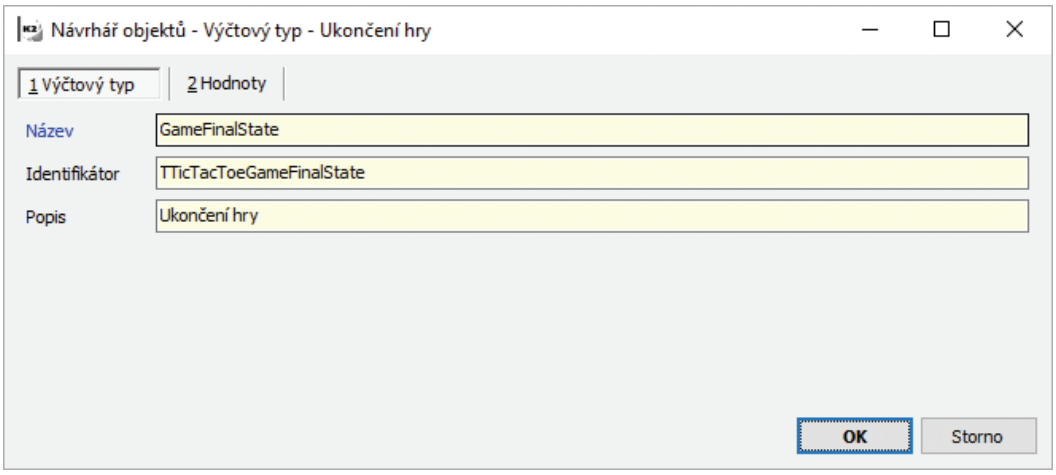


Obrázek 317 - Pole typu výčtový typ v datovém modulu

PŘÍKLAD: Na příkladu si předvedeme, jak lze výčtové typy vytvořit a použít v datovém modulu.

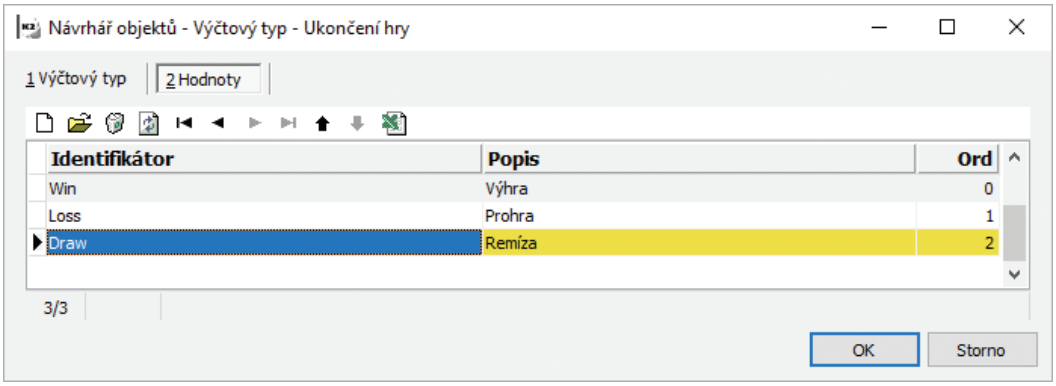
Vytvoříme výčtový typ, který bude sloužit k rozlišení stavu výsledku hry Piškvorky. Založíme typ, který bude mít tři hodnoty – výhra, prohra, remíza. Protože se tyto hodnoty nebudou měnit, nepotřebujeme kvůli tomu zakládat číselník, který by měl tyto záznamy uloženy v databázové tabulce.

Na záložce ***„Typy“*** založíme nový záznam ***„GameFinalState“***, který popíšeme jako ***„Ukončení hry“***. Identifikátor výčtového typu se automaticky založí jako ***„TTicTacToeGameFinalState“***, viz Obrázek 318 - Příklad založení nového výčtového typu.



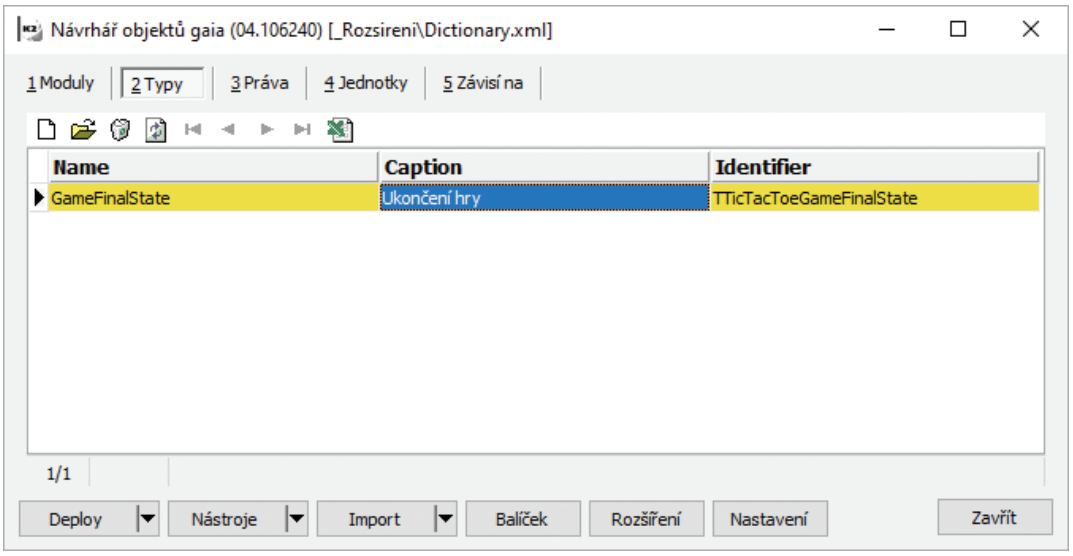
Obrázek 318 - Příklad založení nového výčtového typu

Do seznamu hodnot postupně vložíme záznamy pro jednotlivé hodnoty zmíněné výše. Pro vítězství založíme identifikátor ***„Win“*** s popisem ***„Výhra“*** a hodnotu necháme jako ***„0“***, pro prohru založíme ***„Loss“*** s popisem ***„Prohra“*** a hodnotou 1, nakonec pro remízu založíme ***„Draw“*** s popisem ***„Remíza“*** a hodnotou 2, viz Obrázek 319 - Příklad založení záznamů v novém výčtovém typu.



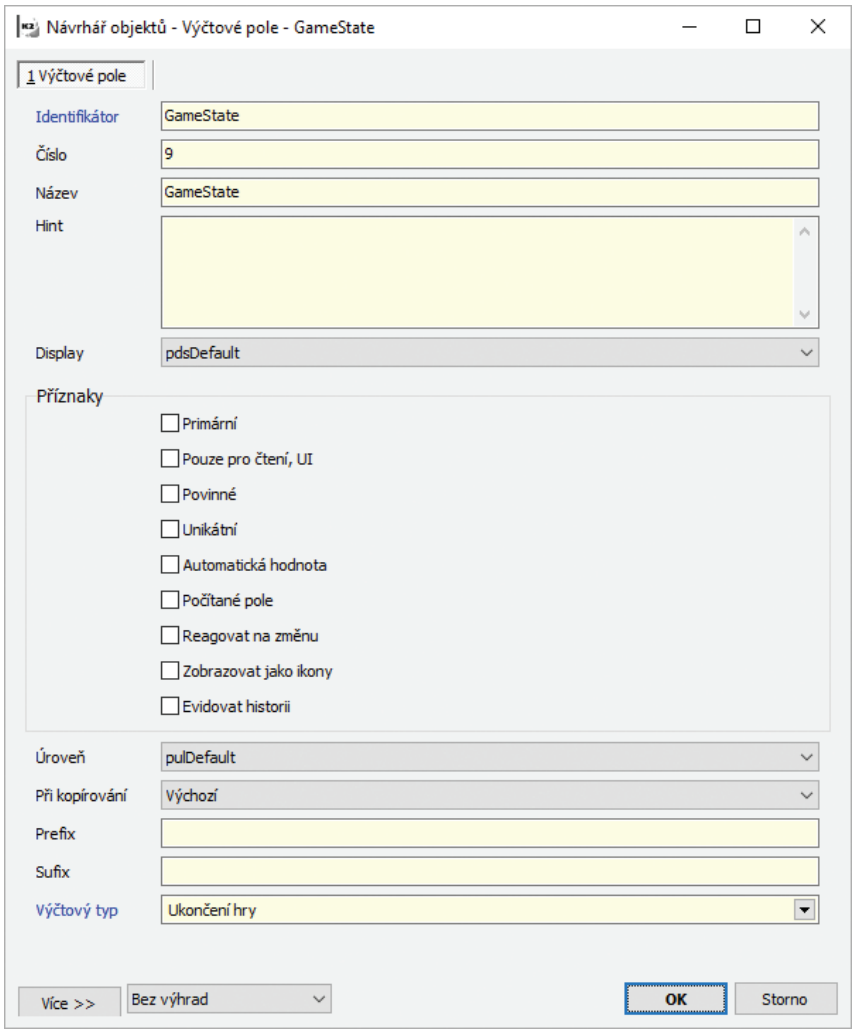
Obrázek 319 - Příklad založení záznamů v novém výčtovém typu

Výsledný typ je pak vidět v seznamu jako ***„GameFinalState“*** s popisem ***„Ukončení hry“***, viz Obrázek 320 - Příklad - Vytvořený výčtový typ.



Obrázek 320 - Příklad - Vytvořený výčtový typ

Následně potřebujeme typ využít jako pole v datovém modulu. Založíme si modul pro evidenci her pro Piškorky. V modulu založíme pole ***„GameState“***, které bude typu ***„Výčet hodnot“***. Pole bude mít popis ***„Stav hry“*** a do hodnoty ***„Výčtový typ“*** nastavíme vytvořený typ z předchozích bodů – ***„Ukončení hry“***, viz Obrázek 321 - Příklad - Vytvoření pole s výčtovým typem.



Obrázek 321 - Příklad - Vytvoření pole s výčtovým typem

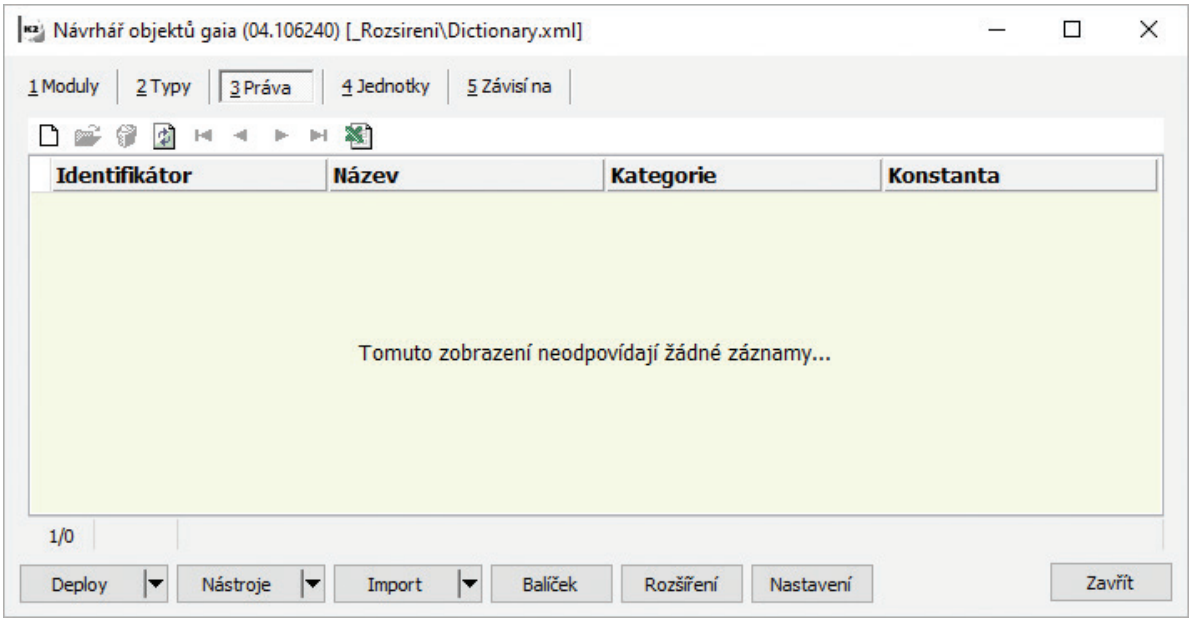
Následně můžeme ve skriptu používat výčtový typ například v konstrukcích pro větvení jako „if“ nebo „switch“.

Například:

```
begin
  if GameState = Loss then
  begin

  end
  else if GameState = Win then
  begin

  end
  else begin

  end;
end;
```

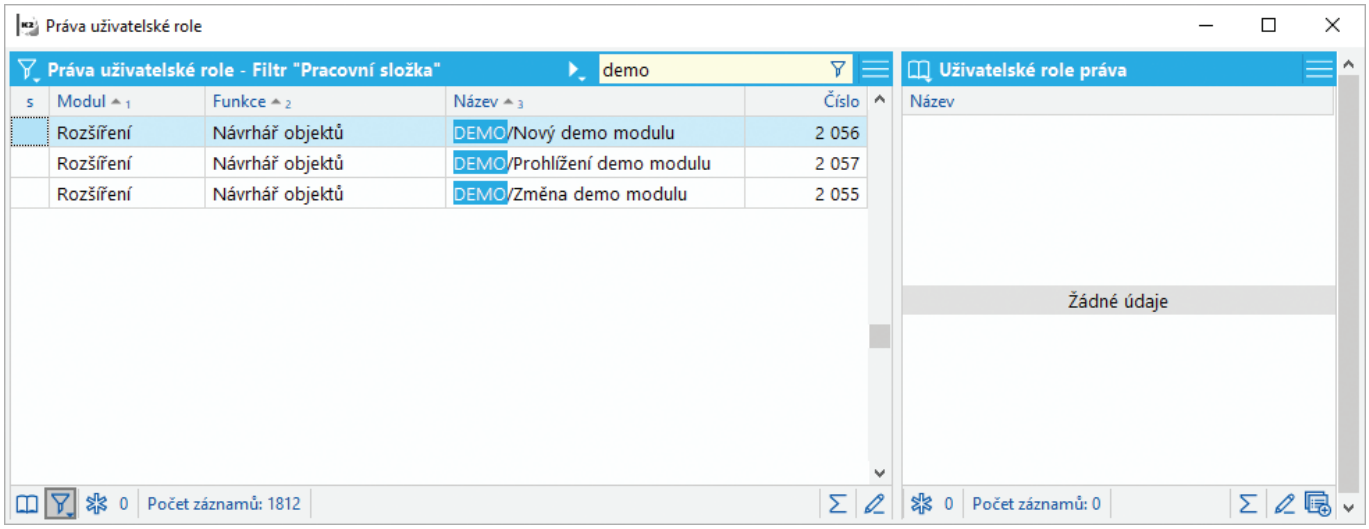
10. ZÁLOŽKA „PRÁVA“

Na této záložce návrháře můžeme definovat vlastní práva, viz Obrázek 322 - Záložka "Práva".



Obrázek 322 - Záložka "Práva"

Nově přidaná vlastní práva lze přiřazovat stejně jako klasická práva K2 přes role. Po vytvoření nového práva přes návrhář objektů se právo objeví v nabídce práv viz Obrázek 323 - Nová práva.



Obrázek 323 - Nová práva

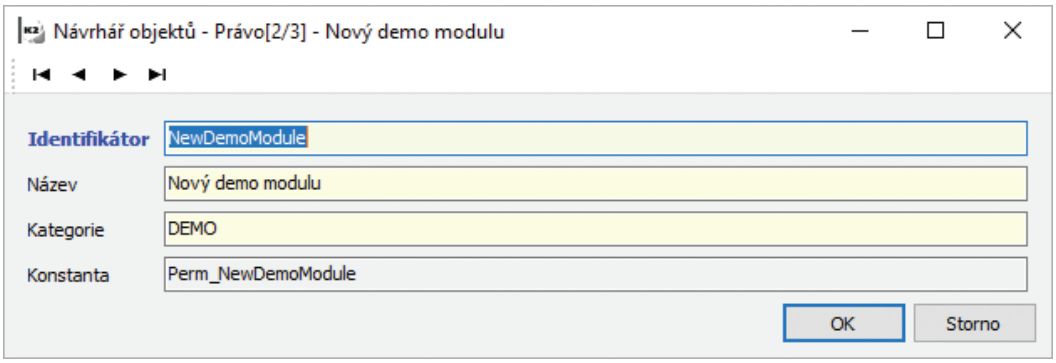
Nové právo založíme buď to pomocí tlačítka Insert, nebo pomocí ikony pro založení nového záznamu (Nový záznam). Pro vytvoření práva stačí vyplnit pouze identifikátor, název a kategorii viz Obrázek 324 - Založení nového práva přes návrhář objektu.

Identifikátor
Slouží pro identifikaci daného práva. Nejedná se o jedinečnou identifikaci, k tomu slouží hodnota GUID, která se vyplní automaticky. Duplicita se kontroluje pouze na úrovni projektu návrháře projektů a Case Sensitive (kontroluje i velká a malí písmena). Např. si pojmenuji právo NewDemoModule, abych věděl, čeho se na první pohled týká.

Název
Slouží k detailnějšímu popisu práva.

Kategorie
Pole ***„kategorie“*** slouží k rozdělení práv do určité kategorie. Např. v našem ***„DemoModul“*** si vytvoříme vlastní práva a chceme tyto práva umět rychle identifikovat, tak si ke všem těmto právům do ***„kategorie“*** napíšeme ***„DEMO“*** a při přiřazování nám bude hned jasné k čemu tyto práva patří.

Konstanta
Hodnota se zde vyplní automaticky, pří vyplnění ***„Identifikace“***. Složí se z konstanty ***„Perm_“*** a pole ***„Identifikátor“*** práva. Např. Perm_NewDemoModule.



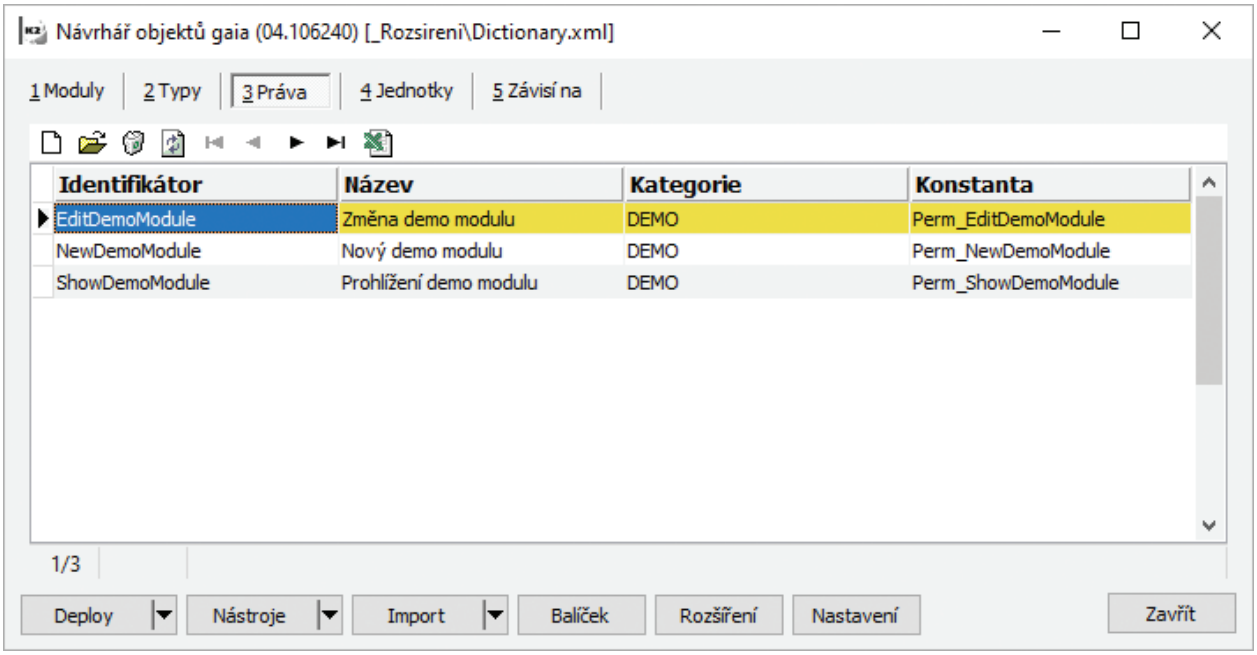
Obrázek 324 - Založení nového práva přes návrhář objektu

POZNÁMKA: Návrhář objektů nekontroluje, zda je při použití práva z jiného rozšíření toto rozšíření uvedeno v závislostech.

POZNÁMKA: Dané práva lze použít také i na právo datového pole, commandu. S vlastními právy vytvořenými přes návrhář objektů lze také dále pracovat i ve skriptu (GetPermission a GetPermissionWithMessage).

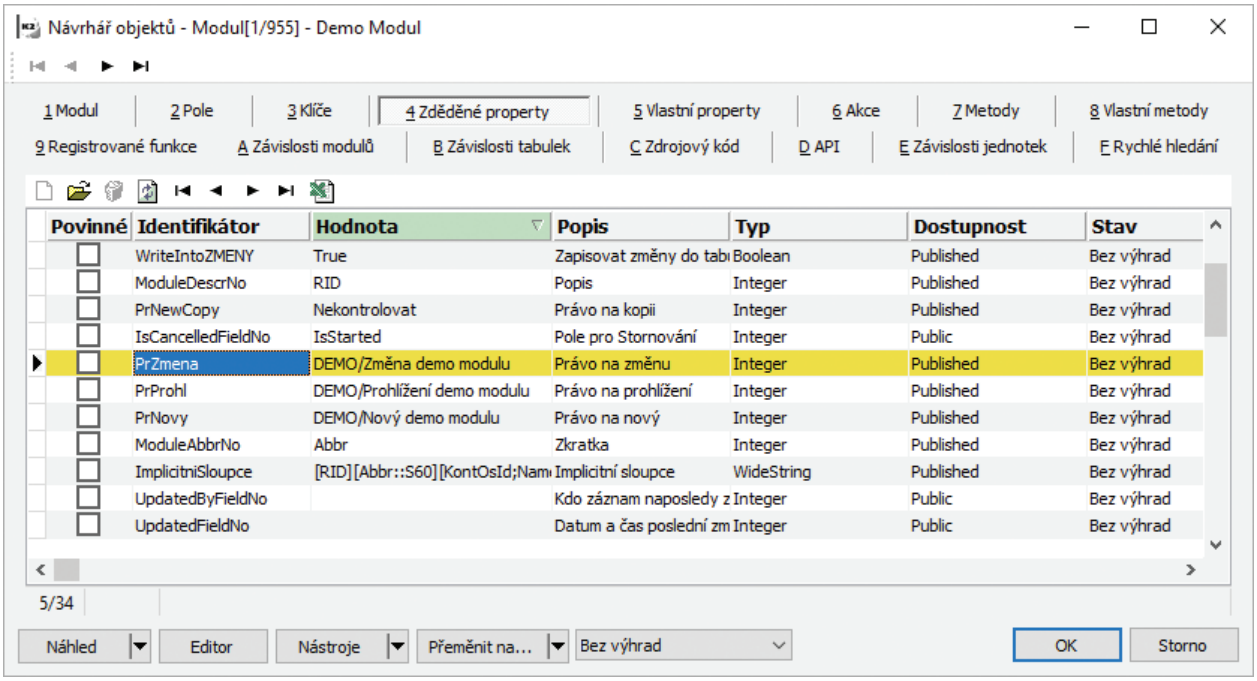
PŘÍKLAD: Vytvořte vlastní práva pomocí návrháře objektů, které budou ovlivňovat nový záznam, změnu záznamu a prohlížení ve Vašem modulu. Tyto práva přiřaďte k vašemu modulu na práva PrNovy, PrProhl a PrZmena.

Nejprve si vytvoříme 3 nová práva na záložce ***„Práva“***, viz Obrázek 325 - Příklad na vlastní práva.



Obrázek 325 - Příklad na vlastní práva

Po vytvoření práv, si najdeme náš ***„DemoModul“***, kde klikneme na záložku ***„Zděděné property“*** a do property ***„PrZmena“***, ***„PrNovy“*** a ***„PrProhl“*** si nastavíme naše nově vytvořená práva (Obrázek 330 - Nová jednotka - implementace jednotky).



Obrázek 326 - Příklad na vlastní práva – aplikování v modulu

Po akci ***„Deploy“*** si ještě budeme muset daná práva přiřadit v K2, jinak nebudeme schopni daný modul ani prohlížet, nebo editovat, či přidávat nové záznamy.

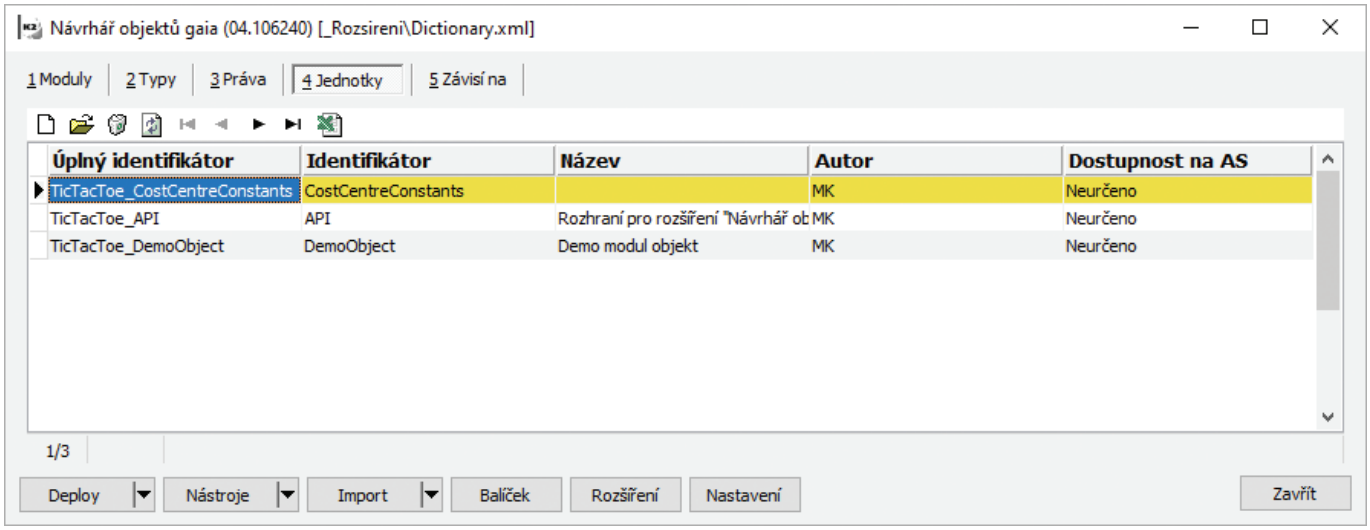
11. ZÁLOŽKA „JEDNOTKY“

Další důležitou záložkou po modulech jsou ***„Jednotky“*** Obrázek 327 - Seznam jednotek v návrháři objektů. Jedná se o seznam všech dostupných jednotek v aktuální definici rozšíření. Pomocí této evidence můžete vytvářet nové nebo upravovat stávající skriptové jednotky, které mohou využívat množinu datových modulů, objektů a dalších jednotek vytvořených v aktuální nebo nainstalované definici rozšíření.

Smyslem této evidence, je mít možnost vytvářet jednotky, ve kterých můžeme implementovat procedury a funkce, které následně mohou komunikovat se všemi datovými moduly v K2 včetně těch vytvořených v návrháři objektů a také se všemi jednotkami, které máme vytvořené v rámci definice rozšíření v návrháři objektů. Toto chování platí i opačně, tedy můžeme v datových modulech vytvořených v návrháři objektů, používat jednotky vytvořené a evidované na záložce ***„Jednotky“***. V případě, že nadefinujeme závislosti mezi jednotlivými definicemi rozšíření, pak můžeme využívat i jednotky a datové modulyz jiných nainstalovaných definic rozšíření, více v kapitole 12. Záložka „Závisí na“. V případě, že pak vytváříme z definice instalační balíček, tento bude mít v sobě zahrnuty všechny jednotky, které jsou s ním spjaty, včetně definic rozšíření, na kterých závisí, o tom více v kapitole 13.3. „Balíček“.

V následujícím textu si popíšeme význam a možnosti oblasti ***„Jednotky“***.

POZNÁMKA: V jednotkách a datových modulech není možné používat procedury a funkce (obecně jednotky) z jiných zdrojů než z těch, které máme nadefinovány na záložce ***„Jednotky“***.



Obrázek 327 - Seznam jednotek v návrháři objektů

11.1. SEZNAM DOSTUPNÝCH JEDNOTEK

Na úvodní obrazovce záložky ***„Jednotky“*** je k dispozici seznam všech dostupných jednotek. Ten vždy obsahuje jednotku typu ***„API“***, která obsahuje konstanty pro všechny struktury v rámci jedné definice rozšíření. Tato jednotka má vždy identifikátor ve tvaru ***„Název Rozšíření_API“*** a název ***„API“***. Dále existuje jednotka pro každou definovanou skriptovou třídu (objekt). Dalšími jednotkami jsou pak již nové, definované přímo v této sekci, více v 11.2. Nová jednotka.

V následující sekci si popíšeme význam jednotlivých sloupců v seznamu, kde jejich význam budeme potřebovat i dále při vytváření nových jednotek.

Úplný identifikátor
Jednoznačný identifikátor jednotky. Vždy je ve tvaru ***„Název rozšíření_Identifikátor“***. V případě, že využíváme závislostí mezi definicemi, dle identifikátorů pak poznáme, do které definice rozšíření jednotlivé jednotky patří.

POZNÁMKA: Úplný identifikátor je generovaný automaticky návrhářem objektů a nemůžeme ho měnit.

Identifikátor
Je identifikátor jednotky, který si zvolíme sami. Tento identifikátor je jedinečný v rámci rozšíření návrháře objektů.

Název
Název jednotky. Definuje autor jednotky. Pokud máme v seznamu jednotky, které jsou z jiných definic rozšíření, tyto jsou pouze pro čtení a není možné název, ani žádný jiný údaj měnit.

Popis
Popis definovaný autorem jednotky.

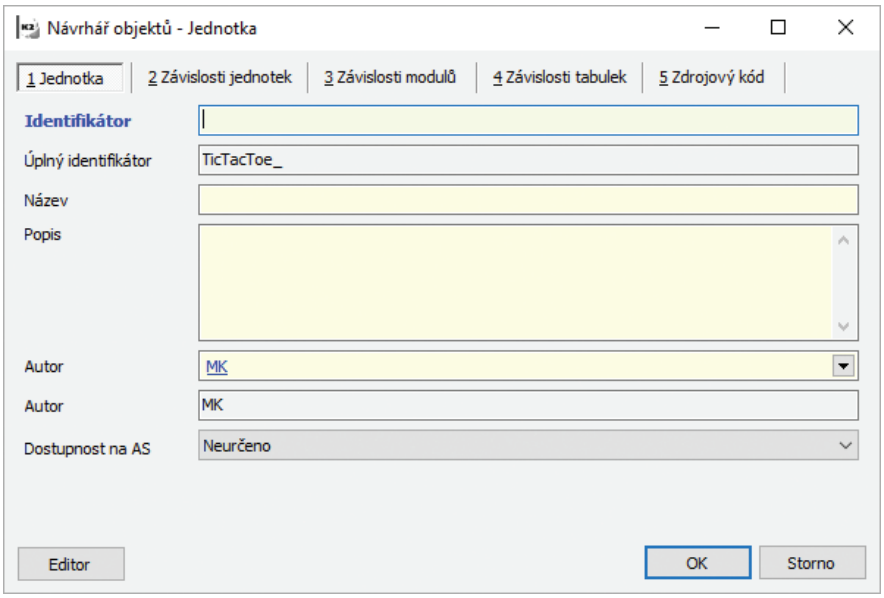
Autor
Autor jednotky.

Dostupnost na AS
Stejně jako u datových modulů, je možné nastavit chování jednotek na aplikačním serveru. Je možné nastavit ***„Pouze pro čtení“***, ***„Pro čtení a zápis“*** a ***„Nepodporováno“***.

11.2. NOVÁ JEDNOTKA

Jak již bylo uvedeno v úvodu této kapitoly, můžeme aktuální definici rozšiřovat o nové jednotky. V těch pak můžeme implementovat vlastní funkce a procedury, které mohou využívat datové moduly, objekty a další jednotky, které existují v aktuální definici rozšíření nebo jsou dostupné ze závislých definic.

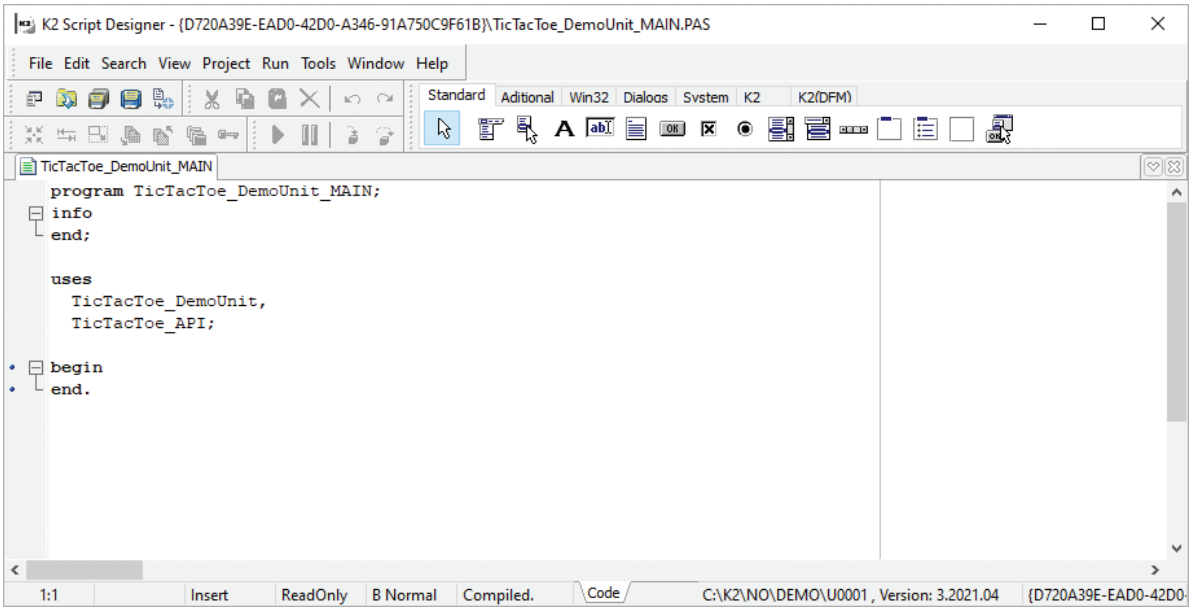
Pokud budeme chtít v aktuálním rozšíření vytvořit jednotku, stiskneme tlačítko Nový nebo stiskneme klávesu Insert. Zobrazí se nám formulář, který můžeme vidět na Obrázek 328 - Nová jednotka.



Obrázek 328 - Nová jednotka

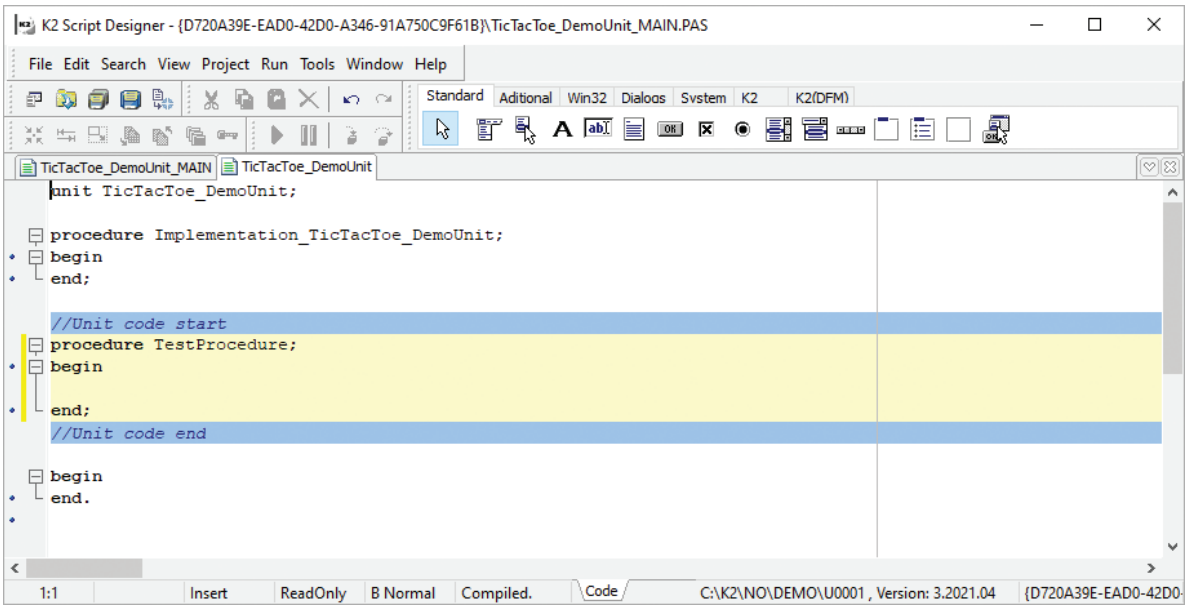
Popis jednotlivých vstupů již nebudeme popisovat, je shodný se sloupci, které jsou popsány v předchozí podkapitole. Po vyplnění všech hodnot můžeme přejít k samotné implementaci jednotky. K tomu využijeme editor skriptu, který je dostupný pod tlačítkem Editor v dolní levé části formuláře viz Obrázek 328 - Nová jednotka.

PŘÍKLAD: V případě, že novou jednotku nazveme například ***„DemoUnit“*** a spustíme editor, zobrazí se nám prostředí pro implementaci a ladění skriptu s vygenerovaným skriptovým programem, který nám umožní skript překládat, viz Obrázek 329 - Nová jednotka – editor skriptu.



Obrázek 329 - Nová jednotka – editor skriptu

V sekci ***„uses“*** pak můžeme vidět naší definovanou jednotku ve tvaru ***„název rozšíření_název jednotky“***, tedy v našem případě ***„TicTacToe_DemoUnit“***. Po otevření jednotky v editoru můžeme vidět sekci pro implementaci našeho kódů. Jedná se o žlutě podbarvenou část ohraničenou textem ***„Unit code start“*** a ***„Unit code end“***. V našem příkladu máme implementovánu prázdnou proceduru ***„TestProcedure“***, viz Obrázek 330 - Nová jednotka - implementace jednotky.



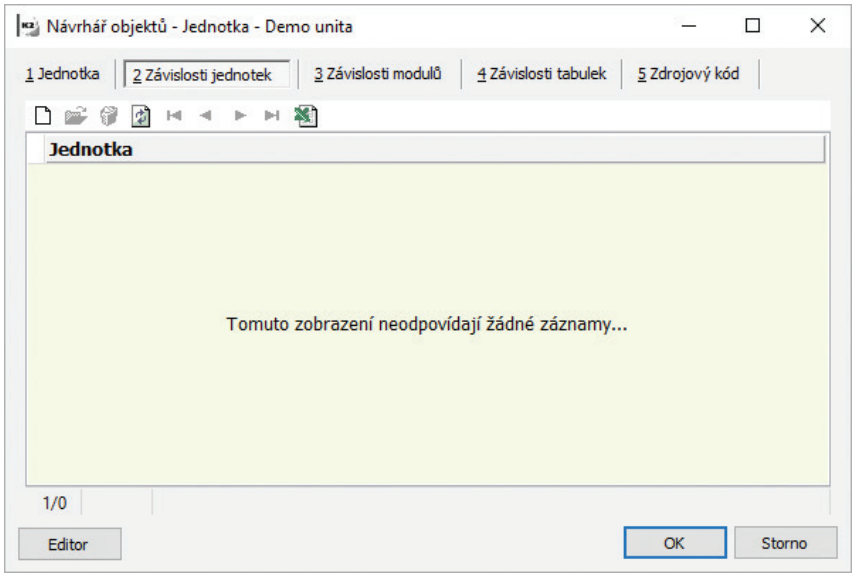
Obrázek 330 - Nová jednotka - implementace jednotky

11.2.1. ZÁVISLOSTI JEDNOTEK

Ve formuláři pro definici nové jednotky, viz Obrázek 328 - Nová jednotka, je vidět, že v záhlaví formuláře existují další tři záložky, které upřesňují chování jednotky. V této kapitole si popíšeme ***„Závislosti jednotek“***.

V případě, že potřebujeme v implementaci jednotky použít kód z jiné jednotky, je nutné ji nejprve připojit. K tomu nám slouží právě tato záložka, viz Obrázek 331 - Jednotky – závislosti jednotek. Jednotku připojíme stisknutím tlačítka Nový nebo stisknutím klávesy Insert. Zobrazí se formulář s dostupnými jednotkami, kde si vybereme požadovanou a potvrdíme výběr. Tyto jednotky se nám objeví v sekci ***„uses“*** naší jednotky, díky čemuž můžeme využívat její procedury a funkce.

POZNÁMKA: Do závislostí můžeme připojit pouze jednotky, které máme dostupné na záložce ***„Jednotka“***, tedy vlastní z aktuální definice rozšíření nebo z jiné definice, kterou máme nainstalovánu a připojenu v závislostech, více v kapitole 12. Záložka „Závisí na“.

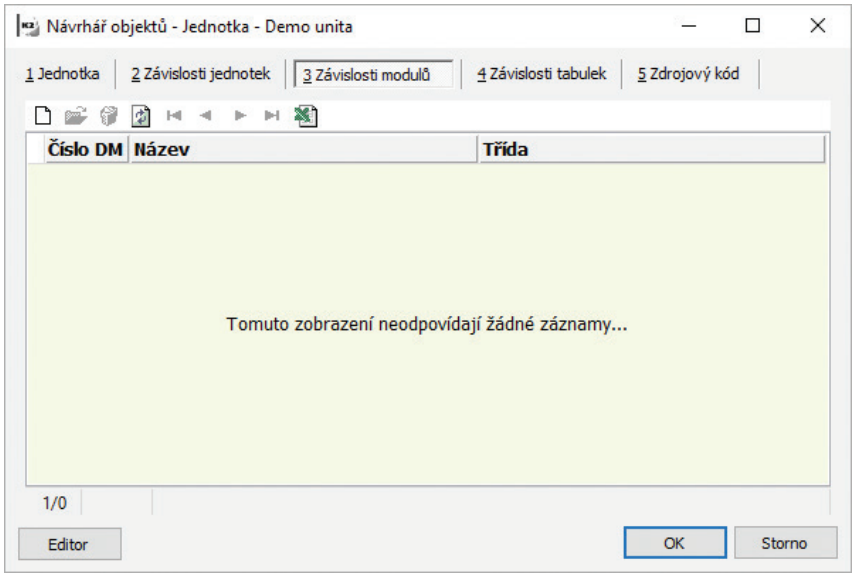


Obrázek 331 - Jednotky – závislosti jednotek

11.2.2. ZÁVISLOSTI MODULŮ

V případě, že potřebujeme připojit k jednotce datové moduly, abychom mohli využívat jejich funkčností, je potřeba, stejně jako u jednotek, tyto moduly připojit. Pro tuto definici existuje záložka ***„Závislosti modulů“***, viz Obrázek 332 - Jednotka – závislosti modulů. Připojení modulu probíhá stejným způsobem jako u jednotek. Stiskneme tlačítko Nový nebo stiskneme klávesu Insert a vybereme datový modul z dostupného seznamu. Připojené moduly se nám pak zobrazí v sekci ***„modules“*** naší jednotky, díky čemuž můžeme využívat jejich funkčností.

POZNÁMKA: Do závislostí můžeme připojit pouze datové moduly, které máme dostupné z aktuální definice rozšíření nebo z jiné definice, kterou máme nainstalovánu a připojenu v závislostech, více v kapitole 12. Záložka „Závisí na“.

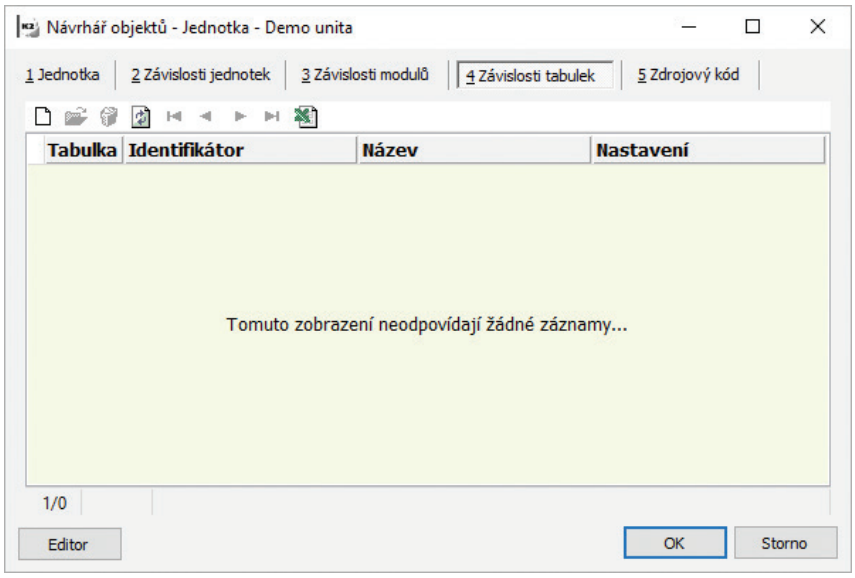


Obrázek 332 - Jednotka – závislosti modulů

11.2.3. ZÁVISLOSTI TABULEK

V IS K2 existují skriptové jednotky, které začínají prefixem ***„FM_“***. Tyto jednotky existují pro každou databázovou tabulku K2 a jejich obsahem je seznam všech polí a indexů, které daná tabulka obsahuje. Seznam je ve formě konstant, které je možné využít v případě přístupu ke standardním modulům K2. Připojené tabulky se nám pak zobrazí v sekci ***„uses“*** naší jednotky, díky čemuž můžeme využívat jejich konstant.

POZNÁMKA: Do návrháře objektů nelze do závislostí připojit žádné standardní skripty K2. Lze použít pouze skripty, které vznikly v NO. Tato vlastnost neumožňovala používat ***„FM“*** soubory, které v K2 existovaly v podobě fyzických skriptových souborů. Soubory proto byly zrušeny a převedeny na generované v paměti K2. Díky tomuto je možné je použít i v návrháři objektů.

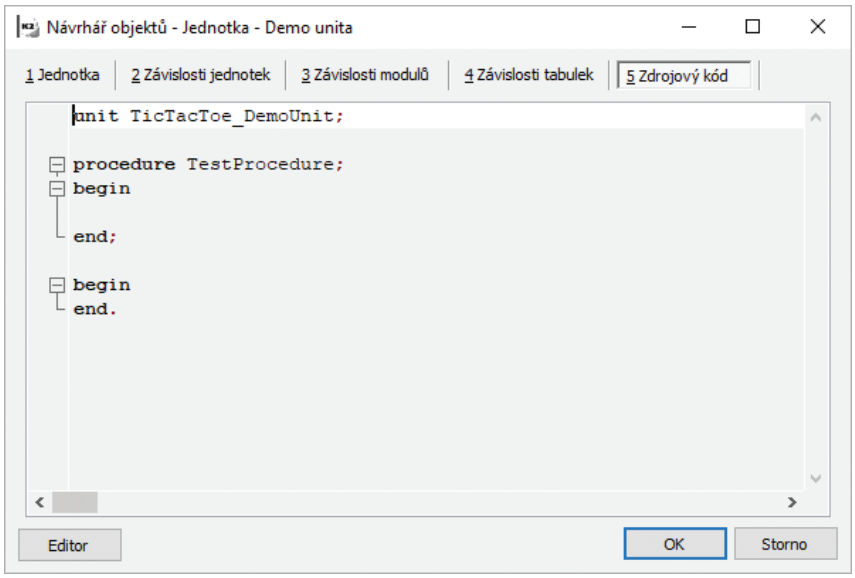


Obrázek 333 - Jednotka – závislosti tabulek

11.2.4. ZDROJOVÝ KÓD

V poslední záložce ***„Zdrojový kód“*** můžeme vidět kompletní implementaci naší jednotky, viz Obrázek 334 - Jednotka – zdrojový kód.

POZNÁMKA: V případě jednotek, které jsou dostupné z jiných definic rozšíření, které máme nainstalovány, máme zdrojový kód takových jednotek k dispozici pouze k náhledu, jsou pouze pro čtení.



Obrázek 334 - Jednotka – zdrojový kód

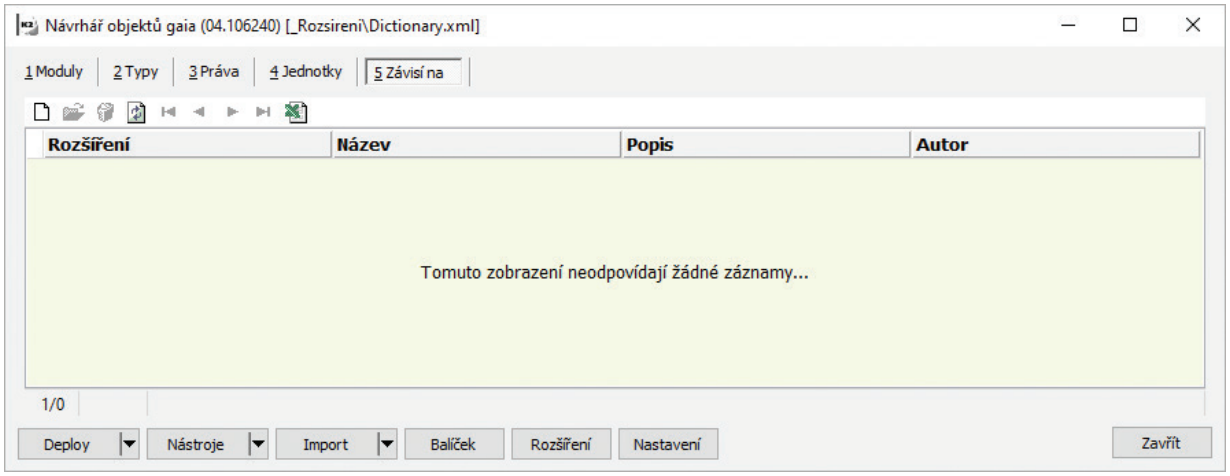
POZNÁMKA: V návrháři objektů není možné používat případné existující zákaznické úpravy (speciální skripty), které jsou definovány mimo návrhář objektů. Pokud bychom je chtěli použít, je nutné pro každou skriptovou jednotku vytvořit jednotku v návrháři objektů a obsah skriptu do této jednotky přesunout, včetně všech jednotek, které se v jednotlivých skriptech používají. Po přesunu a provedení akce ***„Deploy“*** můžeme v návrháři objektů tyto jednotky a jejich logiku používat.

POZNÁMKA: V další kapitole bude uveden komplexnější příklad na definici jednotek, který nejprve vyžaduje seznámení s další částí návrháře objektů – sekce ***„Závisí na“***.

12. ZÁLOŽKA „ZÁVISÍ NA“

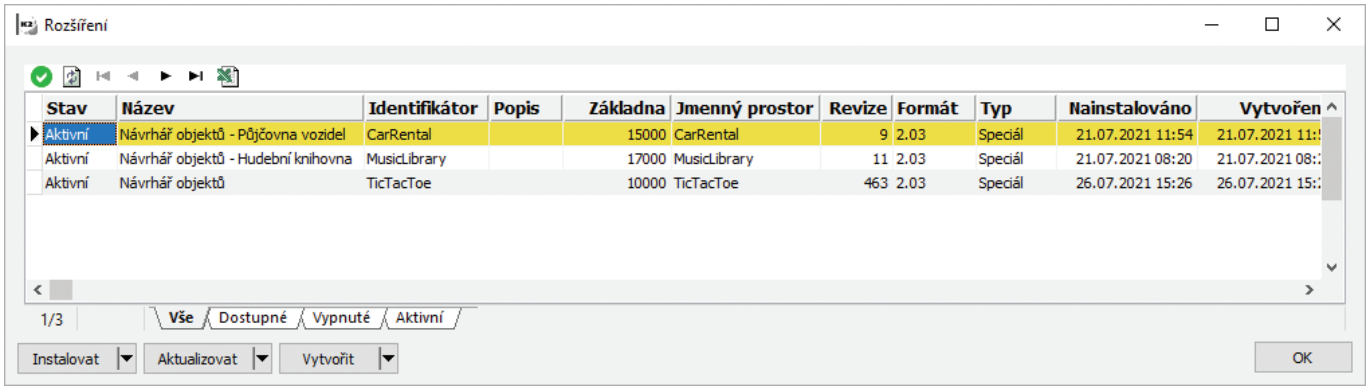
Na třetí záložce ***„Závisí na“*** můžeme nadefinovat, na kterých instalovaných definicích rozšíření závisí aktuální definice. Díky nadefinování závislostí pak můžeme z připojených definic používat datové moduly, jednotky a jejich funkce a procedury. Připojit k závislostem je možné pouze nainstalované definice rozšíření, které jsou distribuovány formou instalačních balíčků. Seznam všech připojených definic můžeme vidět po přepnutí na záložku ***„Závisí na“***, viz Obrázek 335 - Seznam závislostí – záložka „Závisí na“.

Připojit definici rozšíření do závislostí provedeme tak, že v seznamu stiskneme klávesu Insert nebo stiskneme tlačítko Nový.



Obrázek 335 - Seznam závislostí – záložka „Závisí na“

Zobrazí se formulář, ve kterém máme na výběr všechny dostupné definice rozšíření. Výběrem definici zařadíme do seznamu závislostí. Po zařazení se automaticky do seznamu dostupných jednotek zařadí všechny jednotky a skriptové třídy z připojené definice. Ty pak můžeme využívat v aktuální definici v datových modulech, jednotkách a skriptových třídách.



Obrázek 336 - Seznam nainstalovaných rozšíření

POZNÁMKA: V případě, že vytváříme instalační balíček pro aktuální definici rozšíření, jsou do balíčku zahrnuty všechny struktury ze všech balíčků, které jsou připojeny v závislostech. Máme tak díky tomu zajištěno, že bude balíček fungovat po přenosu a instalaci do jiné instalace IS K2.

POZNÁMKA: V případě, že potřebujeme připojit do závislostí definici rozšíření, kterou nemáme nainstalovánu, můžeme přímo ve formuláři dostupných rozšíření, Obrázek 336 - Seznam nainstalovaných rozšíření provést její instalaci a následně použít.

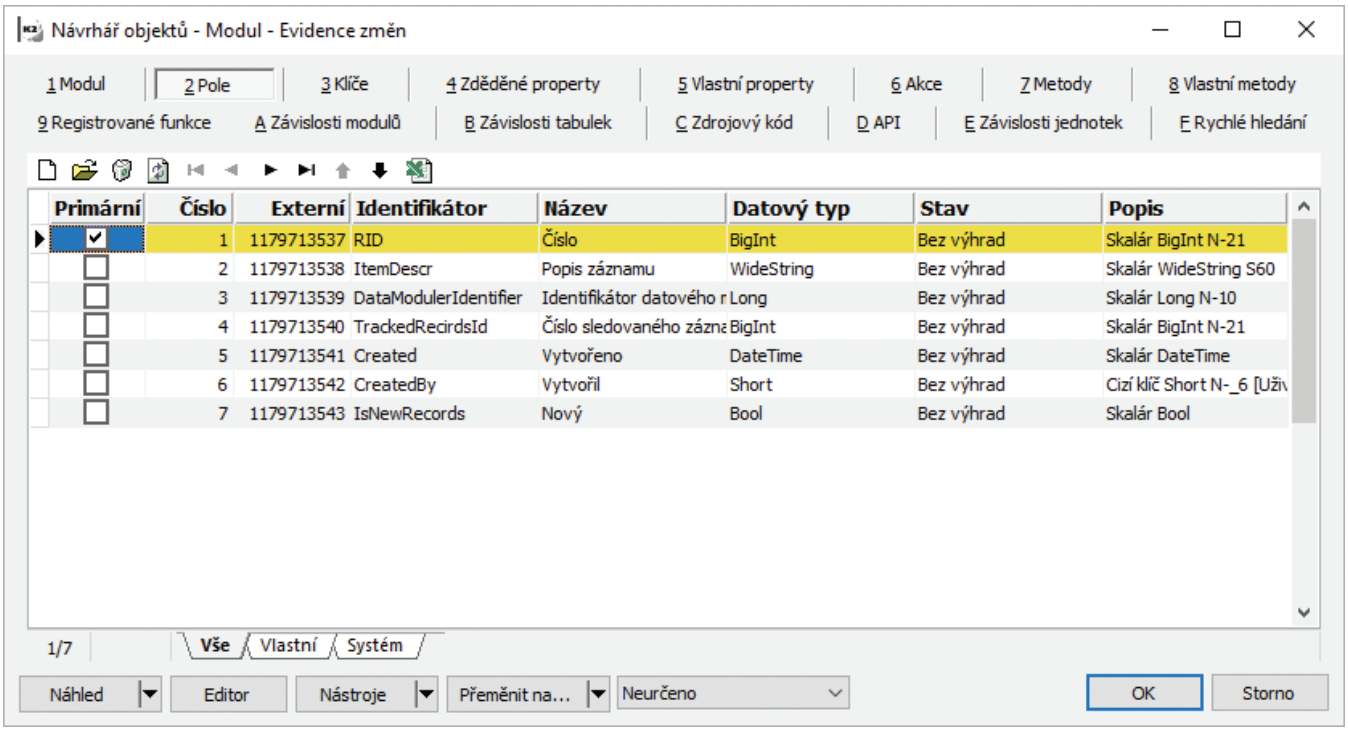
TIP: Jednoduchou instalaci si můžeme provést přetažením instalačního souboru principem ***„Drag&Drop“*** přímo z disku počítače do formuláře dostupných balíčků. Přetažený balíček se automaticky nainstaluje a je připravený k použití. Detailnější popis instalace a vše kolem této problematiky nalezneme v kapitole 13.4. „Rozšíření“.

Pro lepší pochopení využití záložky ***„Jednotky“*** a „Závisí na“ si použití popíšeme na detailnějším příkladu.

PŘÍKLAD: Představme si dvě definice rozšíření v návrháři objektů. První definice, kterou nazveme ***„ChangeTracker“*** bude sloužit k evidencí změn v libovolném datovém modulu. Pro jednoduchost nás bude zajímat pouze informace o tom, že uživatel vytvořil či změnil záznam. Tato definice bude ve výsledku sloužit jako knihovna, kterou můžeme připojit k libovolné jiné definici rozšíření za účelem sledování změn v jejich datových modulech.

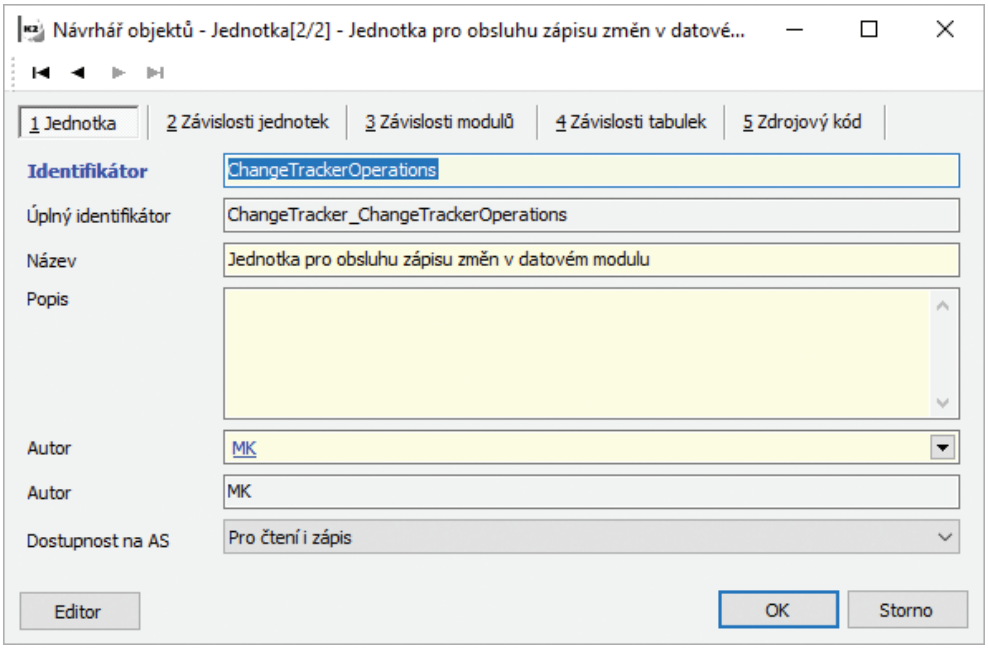
Druhá definice, nazveme ji ***„CarEvidence“***, bude sloužit k evidenci vozového parku. K této definici nainstalujeme balíček s ***„ChangeTracker“*** definicí a použijeme její logiku k záznamu změn v datových modulech evidence vozů.

Pojďme si teď nadefinovat první definici na sledování změn. Zde vytvoříme pouze jeden datový modul, který nazveme ***„ChangesEvidence“***. Vytvoříme pole ***„RID“*** jako primární klíč, pole s popisem záznamu ***„ItemDescr“***, identifikátor datového modulu, ve kterém došlo ke změně ***„DataModuleIdentifier“***, identifikátor sledovaného záznamu ***„TrackedRecordId“***, příznak, zda se jedná o nový záznam či editovaný ***„IsNewRecord“*** a nakonec pole ***„Created“*** a ***„CreatedBy“*** k uložení kdy se změna udála a kdo ji provedl. Více můžeme vidět na Obrázek 337 - Datový modul „ChangesEvidence“.



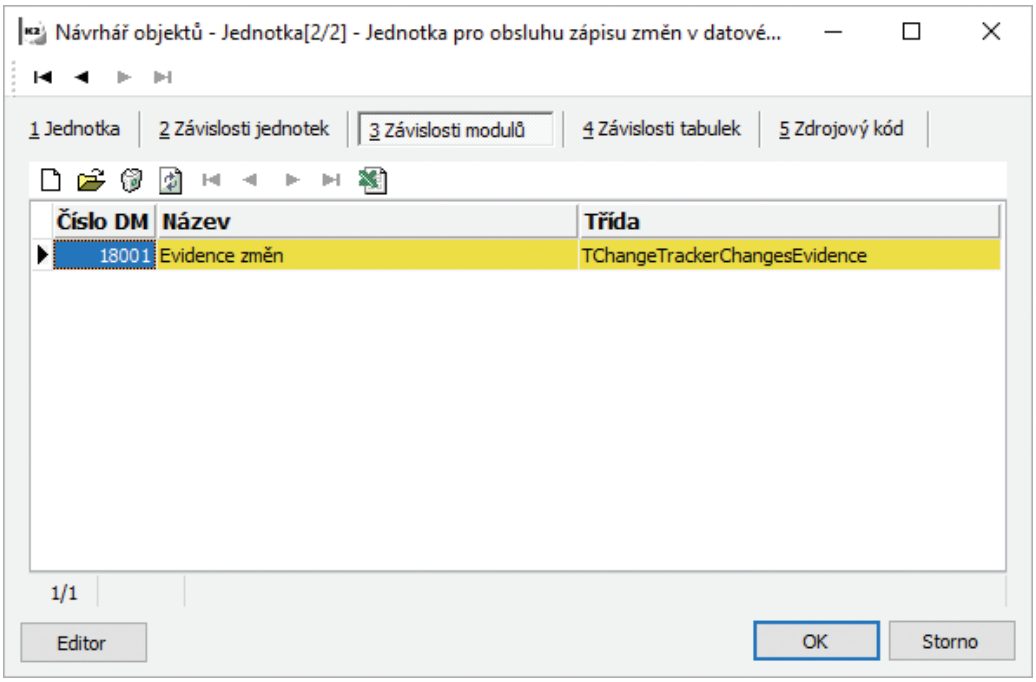
Obrázek 337 - Datový modul „ChangesEvidence“

Datový modul pro ukládání dat máme hotový. Dále se přepneme na záložku ***„Jednotky“***, kde založíme novou jednotku. Její funkcí bude plnění datového modulu pomocí procedury, která bude mít potřebné vstupní parametry. Jednotku nazveme ***„ChangeTrackerOperations“***. Základní definici můžeme vidět na Obrázek 338 - Definice jednotky "ChangeTrackerOperations".



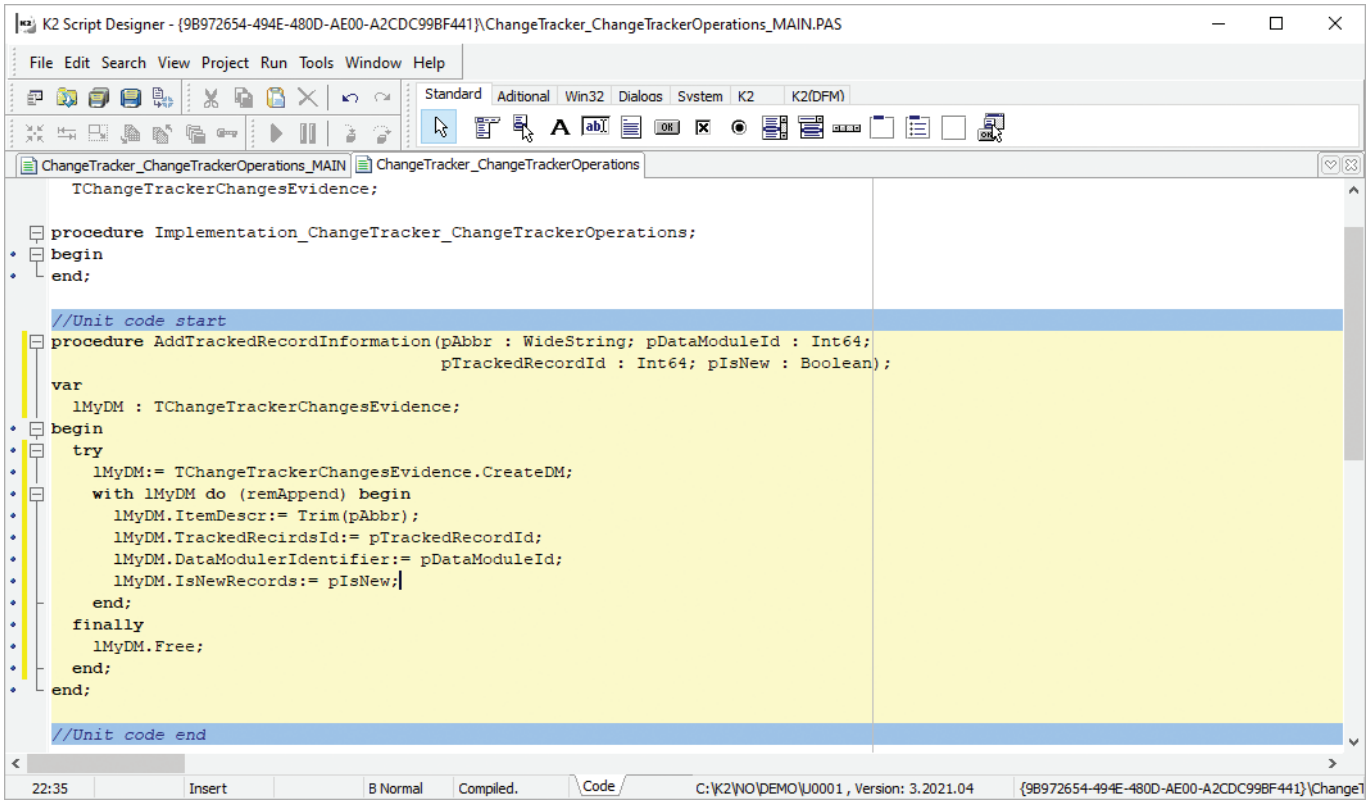
Obrázek 338 - Definice jednotky "ChangeTrackerOperations"

Protože potřebujeme v implementaci používat datový modul ***„ChangesEvidence“***, který jsme si vytvořili v předchozím kroku, je nutné ho připojit do ***„Závislostí modulů“***. Přepneme se tedy na tuto záložku a vložíme datový modul, viz Obrázek 339 - Závislosti v jednotce "ChangesTrackerOperations". Díky tomuto budeme mít ve skriptu pro jednotku definovanou sekci ***„modules“***, ve které bude připojen náš datový modul, viz Obrázek 340 - Implementace jednotky "ChangestTrackerOperations".



Obrázek 339 - Závislosti v jednotce "ChangesTrackerOperations"

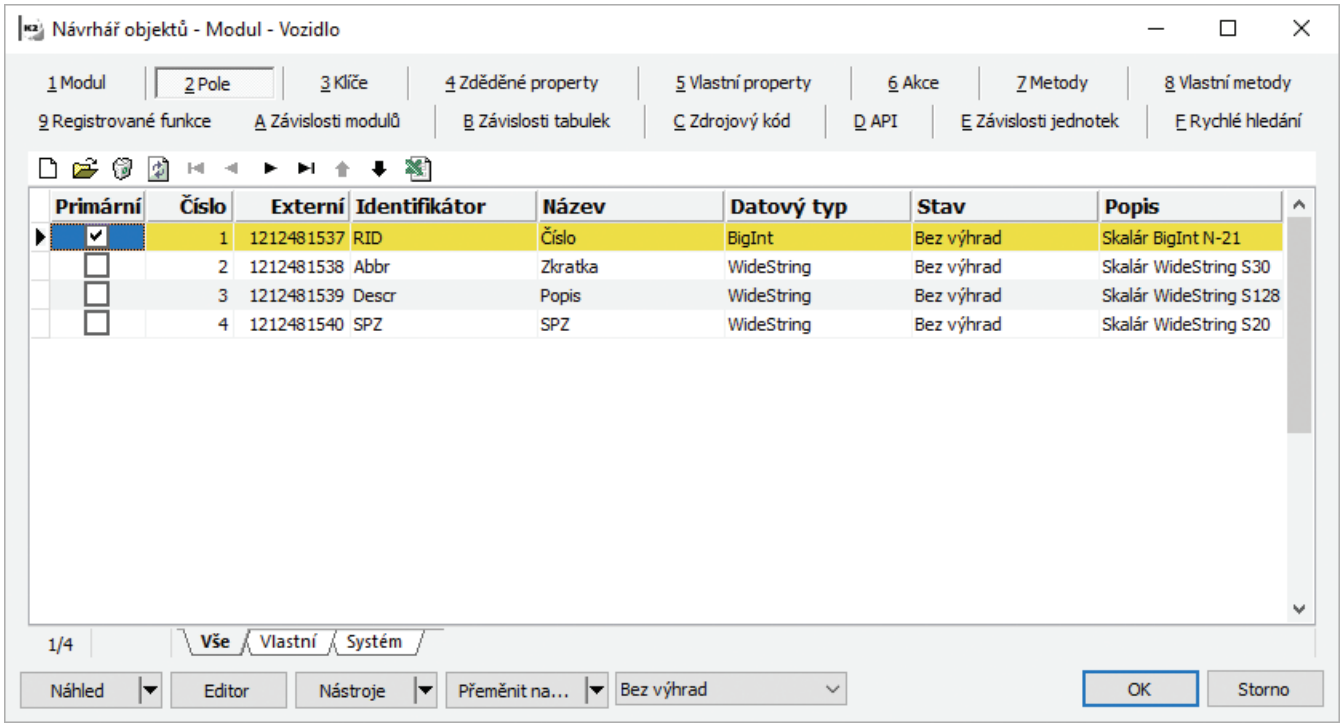
Dalším krokem bude implementace samotné procedury pro ukládání záznamů. Z jednotky spustíme editor. Do sekce určené k implementaci vytvoříme proceduru, kterou nazveme ***„AddTrackedRecordInformation“***, která bude mít jako vstupní parametry popis vkládaného záznamu jako ***„pAbbr“*** typu řetezec (widestring), dále číslo datového modulu jako ***„pDataModuleId“*** typu číslo (Int64), číslo měněného záznamu jako ***„pTrackedRecordId“*** typu velké číslo (Int64) a nakonec informaci o tom, zda se jedná o nový záznam či ne, tedy ***„pIsNew“*** typu ***„ano/ne“*** (boolean). Uvnitř procedury je kód, který vloží záznam do datového modulu ***„ChangesEvidence“*** dle vstupních parametrů, víz Obrázek 340 - Implementace jednotky "ChangestTrackerOperations".



Obrázek 340 - Implementace jednotky "ChangestTrackerOperations"

Knihovnu pro sledování změn máme hotovou. Vytvoříme instalační balíček pomocí funkce ***„Balíček“***. Následně ho použijeme v jiné instalaci IS K2, kde bude vytvářet další definici rozšíření.

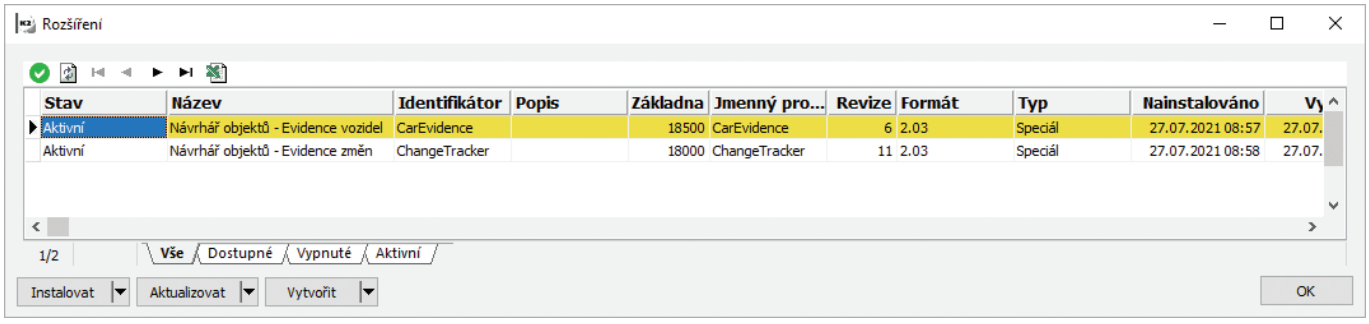
V jiné K2 než jsme používali do teď, založíme novou definici rozšíření, kterou nazveme ***„CarEvidence“***. V této definici založíme nový datový modul ***„Car“***, který bude sloužit k evidenci automobilů. V modulu vytvoříme pole ***„RID“***, které bude sloužit jako primární klíč záznamu, pole ***„Abbr“*** jako zkratka názvu vozidla, pole ***„Descr“*** jako popis vozidla, a nakonec pole SPZ jako uložení státní poznávací značky vozidla, viz Obrázek 341 - Datový modul "Car".



Obrázek 341 - Datový modul "Car"

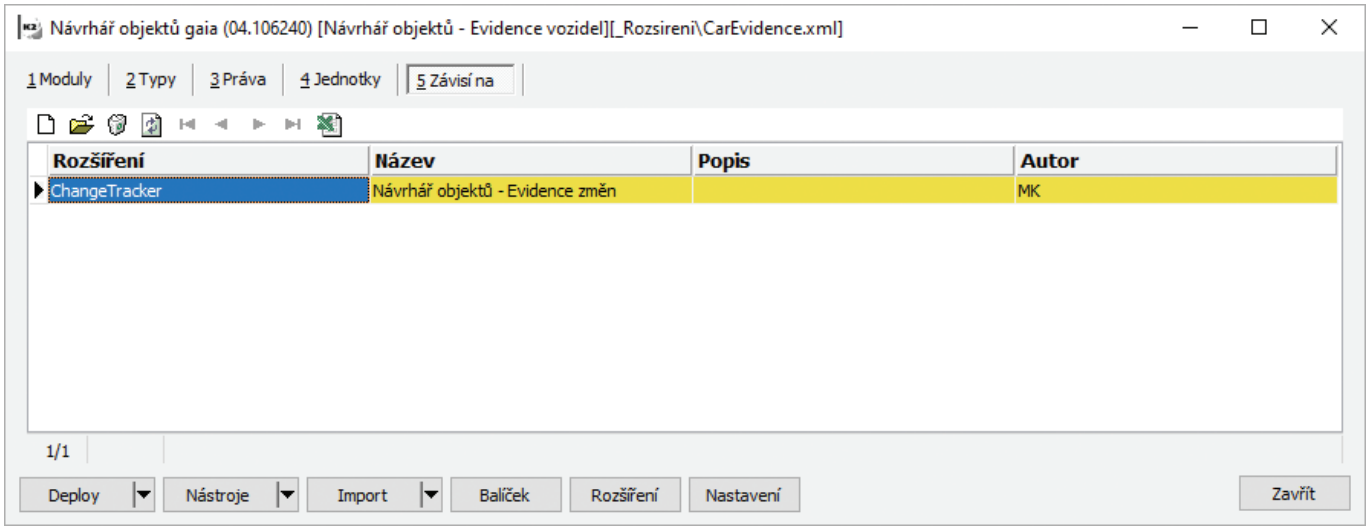
Máme definován modul pro ukládání vozidel. Teď bychom chtěli připojit knihovnu pro sledování změn záznamů, kterou jsme si připravili v předchozí části příkladu. Tuto knihovnu musíme nainstalovat a následně použít její proceduru k záznamu změn.

Přepneme se do záložky ***„Závisí na“***, kterou jsme popisovali v aktuální kapitole, a stiskneme tlačítko pro vložení závislosti ze seznamu rozšíření, která jsou nainstalována v aktuální IS K2. Naše rozšíření zde pochopitelně nenajdeme, musíme ho nejprve nainstalovat. Jednoduše chytíme myší soubor s instalačním balíčkem ***„ChangesTracker“***, který jsme si vytvořili výše v příkladu a máme ho připravený na pevném disku počítače, a přetáhneme ho způsobem ***„Drag&Drop“*** do formuláře ***„Rozšíření“***, který máme aktuálně otevřený. Pokud nenastane nějaký problém, balíček se nainstaluje a aktivuje, viz Obrázek 342 - Instalace balíčku "ChangesTracker". Nainstalovaný balíček vybereme a vložíme do našeho aktuálního rozšíření, viz Obrázek 343 - Definice závislosti "CarEvidence" na „ChangesTracker“. Od teď ho můžeme začít používat v našem rozšíření.



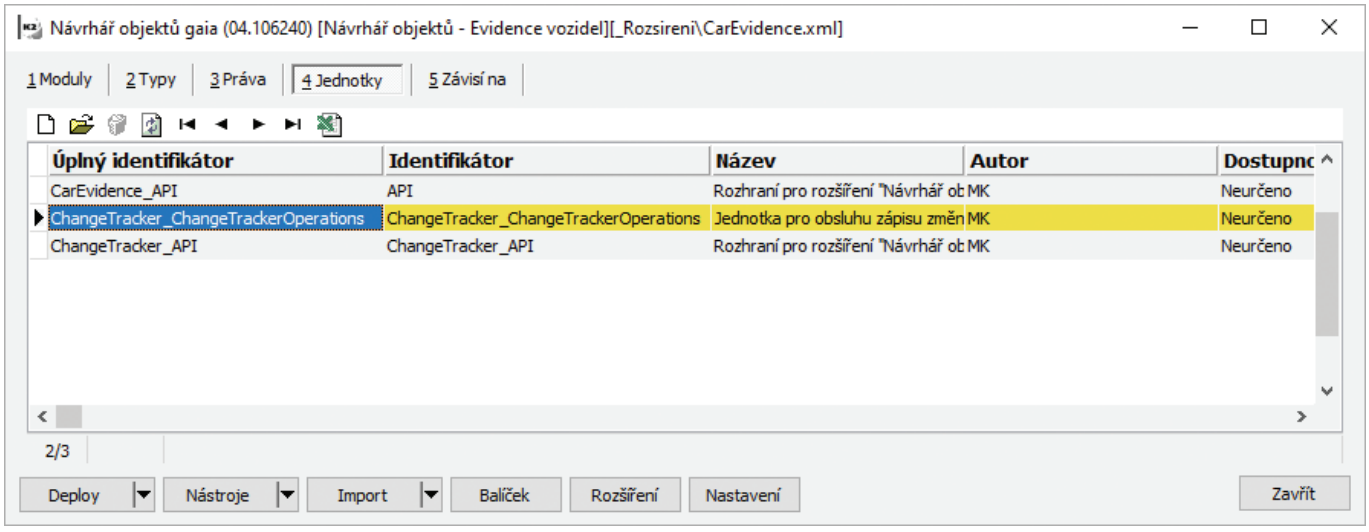
Obrázek 342 - Instalace balíčku "ChangesTracker"

POZNÁMKA: V případě, že při instalaci balíčku dojde k nějakému problému, doporučuji si nastudovat kapitolu 13.4. „Rozšíření“, která pojednává o instalaci balíčků rozšíření.



Obrázek 343 - Definice závislosti "CarEvidence" na „ChangesTracker“

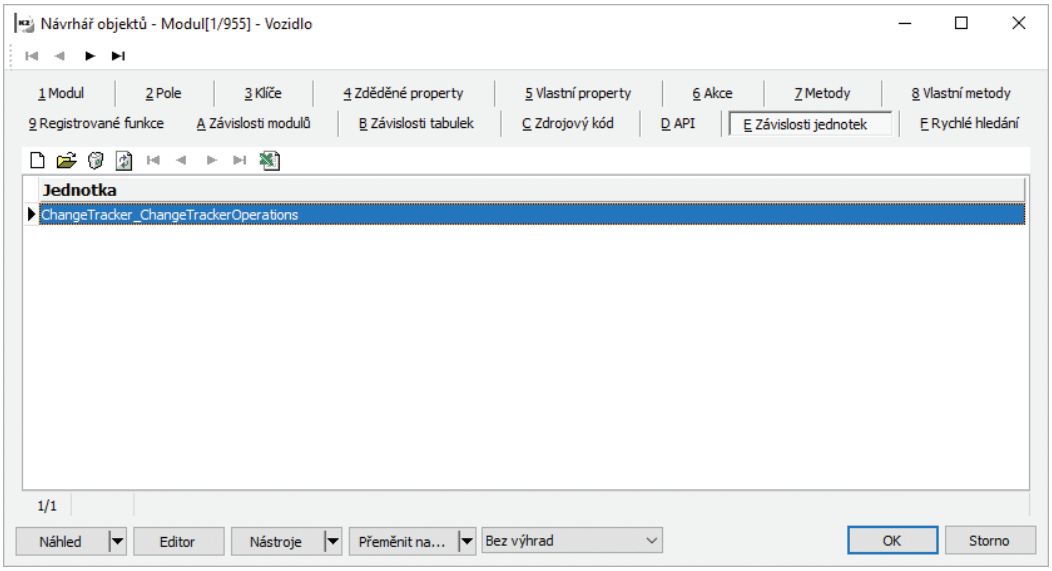
Po přepnutí na záložku ***„Jednotky“*** můžeme vidět všechny dostupné struktury z připojeného a nainstalovaného balíčku ***„ChangesTracker“***. Jedná se o jednotku ***„ChangesTrackerOperations“***, kterou jsme si implementovali v první části příkladu a také o aplikační rozhraní balíčku – ***„ChangesTracker_API“***. Zde se můžeme v detailu jednotek podívat na její implementaci. Měnit ji, ale nemůžeme, všechny struktury z nainstalovaných balíčků jsou pouze pro čtení, viz Obrázek 344 - Dostupné jednotky z "ChangesTracker" po připojení definice.



Obrázek 344 - Dostupné jednotky z "ChangesTracker" po připojení definice

Balíček pro podporu sledování změn máme nainstalovaný a připojený k naší definici rozšíření. Zbývá nám doplnit implementaci, která naši knihovnu využije.

Otevřeme si detail datového modulu ***„Car“***, který slouží k evidenci vozidel a musíme připojit jednotku ***„ChangesTrackerOperations“*** k datovému modulu, abychom ji mohli používat v jeho funkcích. Připojení závislé jednotky provedeme na záložce ***„Závislosti jednotek“*** v definici datového modulu, viz Obrázek 345 - Definice závislosti na datovém modulu "CarEvidence".

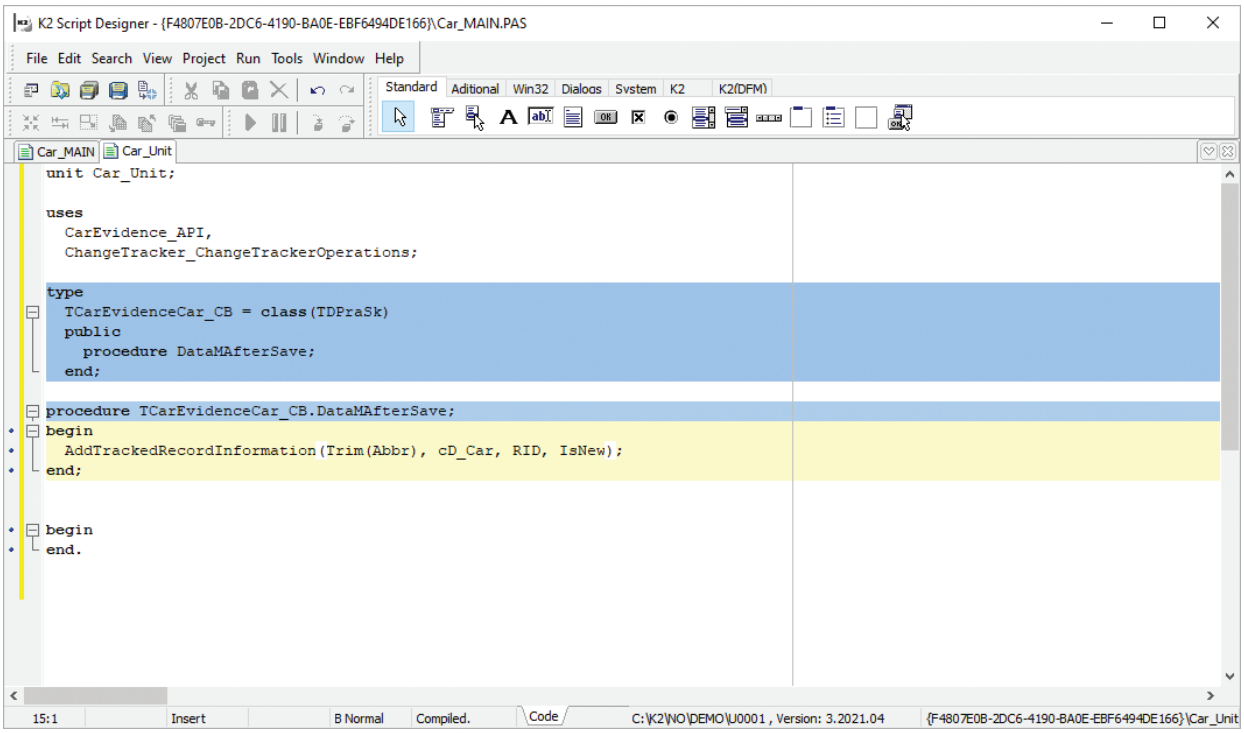


Obrázek 345 - Definice závislosti na datovém modulu "CarEvidence"

Nakonec otevřeme editor pro implementaci logiky datového modulu a implementujeme metodu ***„AfterSave“***. V této metodě zavoláme proceduru ***„AddTrackedRecordInformation“*** z naší připojené jednotky a předáme ji patřičné informace. Jako popis záznamu předáme jeho zkratku, dále předáme číslo datové modulu, číslo měněného záznamu a nakonec informaci, zda se jedná o nový záznam či změnu, viz Obrázek 346 - Implementace uložení záznamu do "ChangesTracker" z "CarEvidence".

Tato implementace nám zajistí, že potom co se vloží či edituje záznam v datovém modulu ***„Car“***, zavolá se procedura ***„AfterSave“***, která převolá uložení záznamu o změně do datového modulu ***„ChangesTracker“***.

Potom co celý projekt promítneme do K2, provedeme operaci ***„Deploy“***, a můžeme datový modul včetně připojené knihovny používat.



Obrázek 346 - Implementace uložení záznamu do "ChangesTracker" z "CarEvidence"

13. GLOBÁLNÍ FUNKCE NÁVRHÁŘE OBJEKTŮ

Ve spodní části základního formuláře návrháře objektů se nachází několik tlačítek. Jedná se o globální funkce, které provádějí operace nad celou definicí v návrháři objektů. V následující části si popíšeme jednotlivé funkce podrobněji.

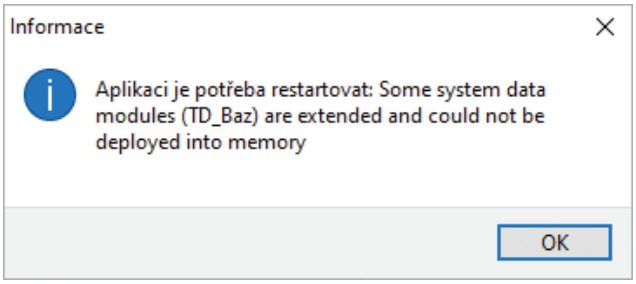
13.1. „DEPLOY" – APLIKOVÁNÍ ÚPRAV DO K2

Funkcí tohoto tlačítka, je promítnout definici rozšíření v návrháři objektů do K2. Tedy aplikovat návrh objektů, vytvořených v návrháři objektů do K2, tak aby je mohlo používat. S touto akcí jsou spojené i databázové operace. Vytváření tabulek, polí, klíčů a všech možných struktur.

Při spuštění se provede kontrola, zda je vše definováno správně. Pokud ano, vytvoří se z definice rozšíření soubory ve formátu ***„xml"***, které popisují struktury. Pro každý datový modul vzniknou dva soubory. Jeden popisuje datový modul a druhý databázovou tabulku k modulu, respektive ***„xFile"***. Soubory vznikají v adresáři ***„Extensions"***, který je v adresáři instalace IS K2. Více o souborech také v kapitole 13.4.1. Instalace rozšíření. Dále se provedou databázové operace, pokud jsou vyžadovány definicí rozšíření. Pokud není nalezena databázová tabulka, pak se vytváří. V případě, že tabulka již existuje, pokusí se návrhář objektů napravit její strukturu, v případě změny, databázovou operací ***„Alter table"***. Vždy to probíhá tak, že se data zálohují do souborů DES/HES. Ty se vytvářejí v adresáři ***„extensions"***, ***„backup"*** v uživatelském adresáři uživatele. Následně se tabulka modifikuje a data z DES/HES souborů se naimportují zpět.

POZNÁMKA: Může nastat situace, kdy se import nepovede. Například v situaci, kdy přidáme pole, na kterém bude existovat unikátní index. Pak na tuto skutečnost návrhář upozorní a je potřeba data ručně naimportovat.

POZNÁMKA: Většina úprav v Návrháři objektů nevyžaduje restart IS K2 po operaci ***„Deploy"***. V případě, že je nutné restart provést, pak se zobrazí po operaci ***„Deploy"*** hlášení s touto informací jak můžete vidět na Obrázek 347 - Hlášení, kdy je nutné provést restart K2 po "Deploy". Pokud je zapotřebí provést po provedení operace ***„Deploy"*** restart K2, další ***„Deploy"*** nepůjde provést, dokud se neprovede restart K2.



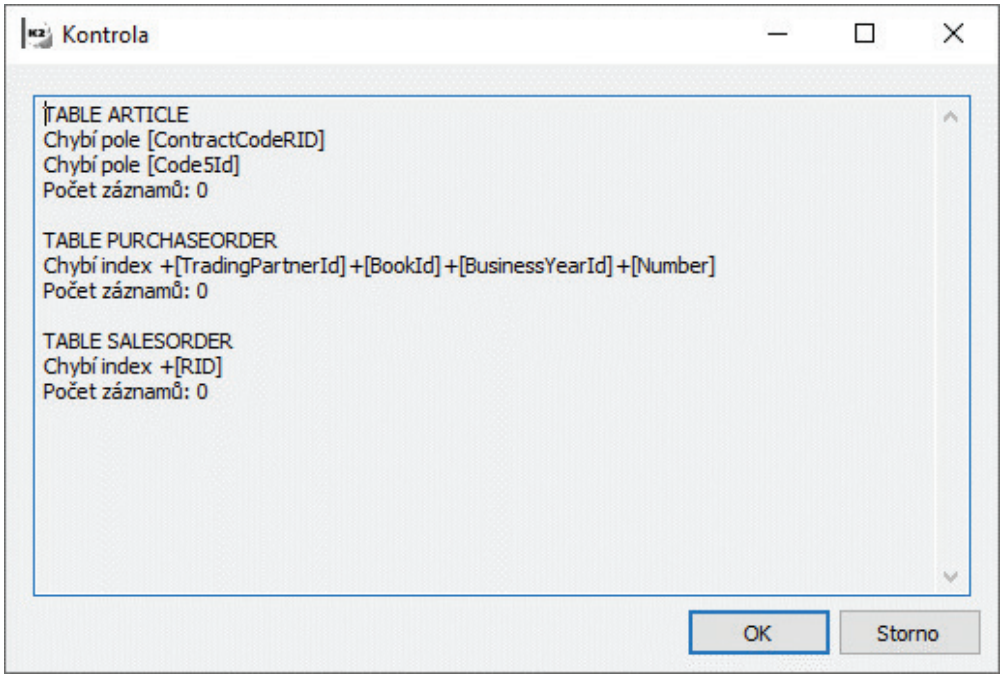
Obrázek 347 - Hlášení, kdy je nutné provést restart K2 po "Deploy"

13.2. „NÁSTROJE“

Pod tlačítkem Nástroje se skrývá seznam nástrojů, které nám pomohou se správou definice rozšíření v návrháři objektů.

13.2.1. KONTROLA

Pomocí tohoto nástroje provedete kontrolu struktur všech datových modulů, které existují v K2, tedy včetně továrních modulů pro mandanta, ve kterém kontrolu spouštíme. Kontrola zjistí, zda se definice shoduje s reálnými objekty v databázi. Případné problémy jsou vypisovány do výstupního okna. Díky tomuto nástroji můžete detekovat například neexistující indexy v databázi a tím zjistit příčinu nízkého výkonu postižených datových modulů. Možný výstup je vidět na Obrázek 348 - Výstup z nástroje "Kontrola".



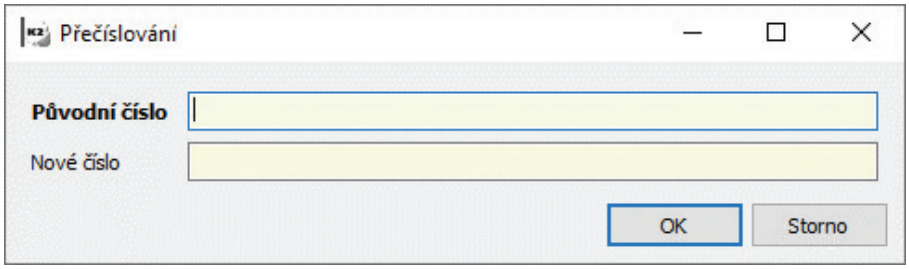
Obrázek 348 - Výstup z nástroje "Kontrola"

13.2.2. API

Funkce ***„API“*** slouží k vygenerování ***„aplikačního rozhraní“*** pro všechny uživatelem definované moduly v návrháři objektů. V kapitole 4.14. API jsme si popsali záložku ***„API“***, která je dostupná u každého datového modulu a zobrazuje aplikační rozhraní pro aktuální datový modul. Pomocí tohoto nástroje získáme aplikační rozhraní pro všechny datové moduly souhrnně.

13.2.3. PŘEČÍSLOVÁNÍ

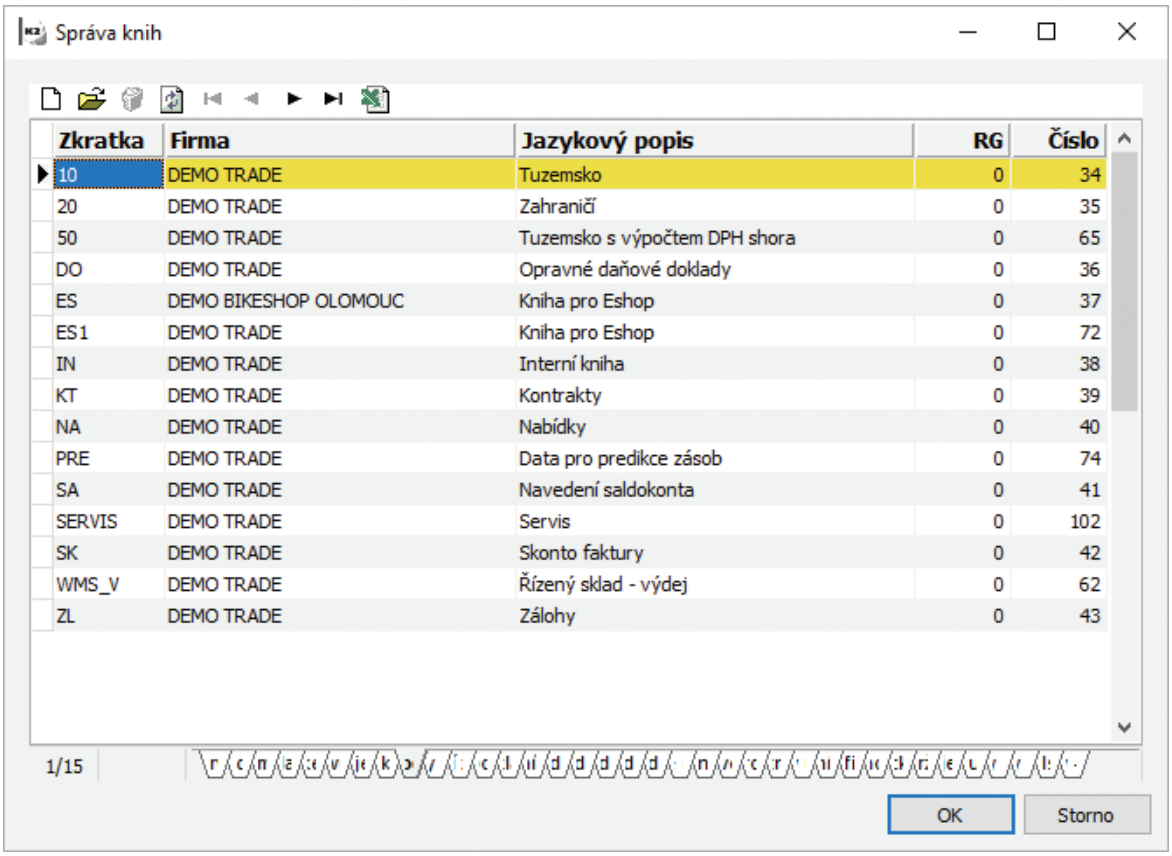
Může nastat situace kdy je potřeba změnit číslování struktur vytvořených v návrháři objektů. Číslo modulu, lze změnit přímo v definici, ale není tak zaručeno, že se změna promítne do všech provázaných struktur, které daný modul používají. Je tedy doporučeno v takovém případě raději využít funkce pro přečíslování záznamů. Ta je k dispozici na hlavní straně pod tlačítkem Nástroje / Přečíslování. Po spuštění se zobrazí formulář, viz Obrázek 349 - Obrázek 348 - Nástroj "Přečíslování". Do hodnoty ***„Původníčíslo“*** vložme původní hodnotu struktury a do pole ***“Novéčíslo“*** novou hodnotu. Po potvrzení dojde k přečíslování ve všech strukturách v návrháři objektů.



Obrázek 349 - Obrázek 348 - Nástroj "Přečíslování"

13.2.4. SPRÁVA KNIH

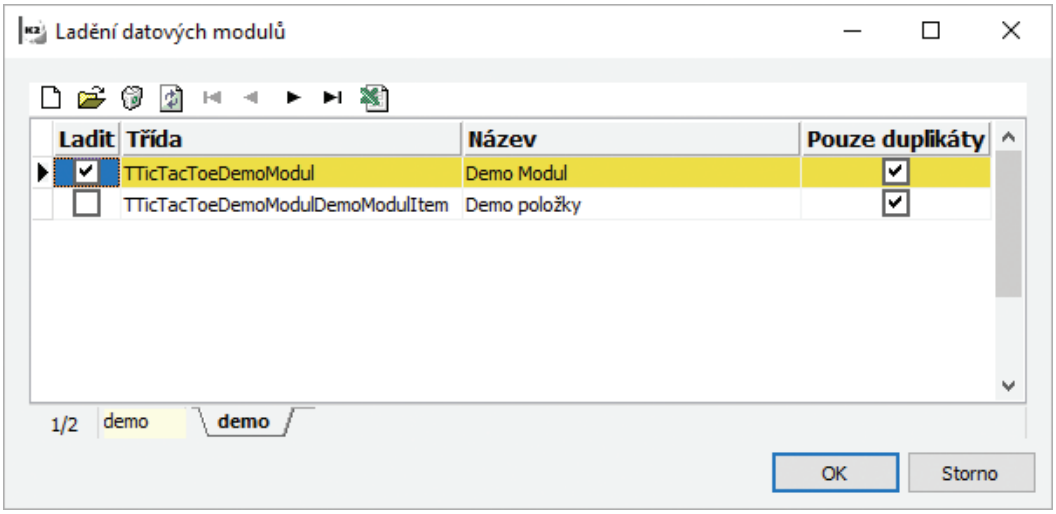
Tento nástroj slouží ke správě všech knih v K2. Je standardně dostupný v K2 ve stromu v sekci *„Správce/ Systém/Správa knih"*. Zde je k dispozici pro případ, že uživatel potřebuje upravit nastavení knih v návaznosti na změny v návrháři objektů. Nemusí tak NO vypínat, ale použije funkci přímo zde.



Obrázek 350 - Správa knih

13.2.5. LADĚNÍ

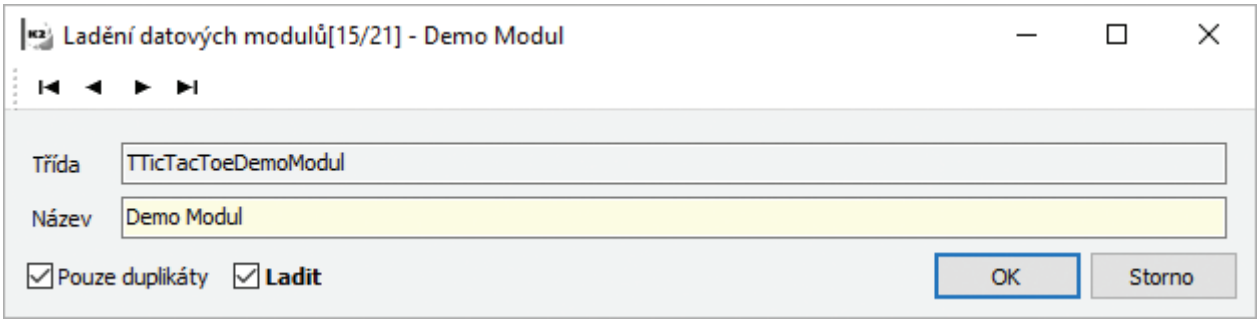
Tato funkce slouží k nastavení ladění jednotlivých datových modulů. Po jejím spuštění se nám zobrazí formulář, který vidíme na Obrázek 351 - Nastavení ladění datových modulů. Jedná se o seznam všech datových modulů, které jsou vytvořené v rámci definice rozšíření v návrháři objektů. U modulů, které budeme chtít ladit, zatrhneme volbu ***„Ladit"***. Výběr můžeme provést přímo v seznamu kliknutím na buňku s volbou ***„Ladit"***.



Obrázek 351 - Nastavení ladění datových modulů

Volbu ***„Ladit“*** můžeme zatrhnout i v detailu modulu v seznamu pro ladění. Detail zobrazíme dvojitým kliknutím myší na konkrétní záznam nebo stiskem tlačítka Změna v nástrojové liště nad seznamem. Otevře se nám formulář, který vidíme na Obrázek 352 - Detail nastavení ladění jednoho datového modulu.

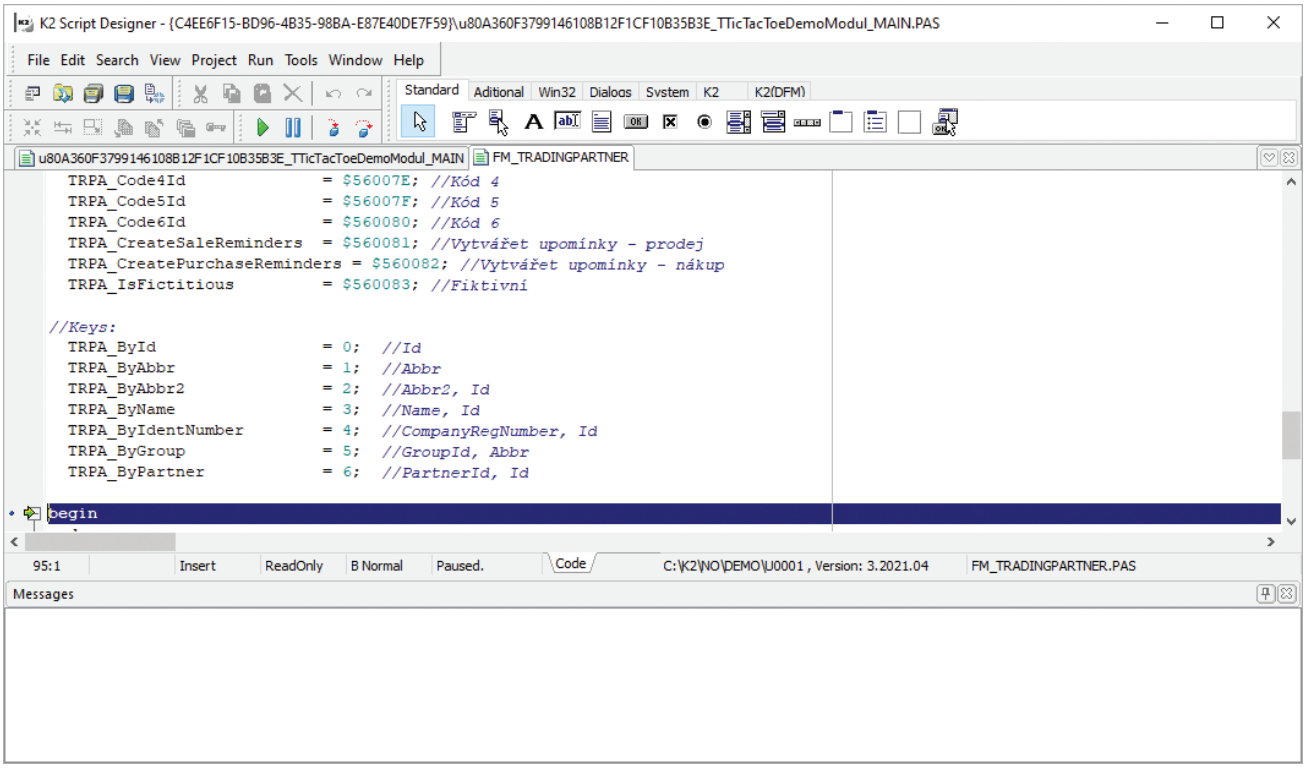
Další volbou, je možnost nastavit, zda budeme ladit originál nebo duplikát. Výchozí hodnota je nastavena na ***„Pouze duplikáty“***. Tato varianta je doporučená.



Obrázek 352 - Detail nastavení ladění jednoho datového modulu

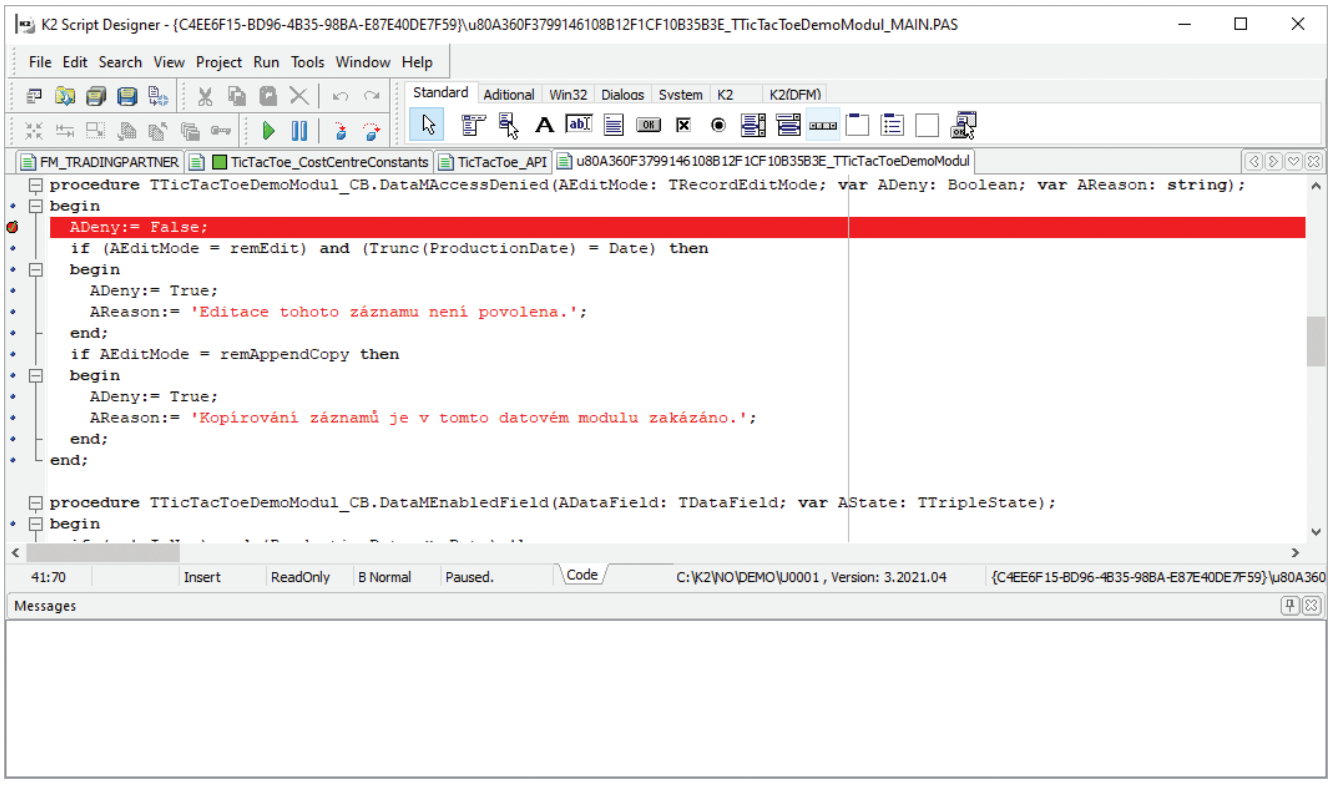
POZNÁMKA: V seznamu datových modulů pro ladění, se nacházejí pouze ty datové moduly, které jsou publikovány do K2. Tedy byla spuštěna funkce ***„Deploy“***, která celou definici v návrháři objektů promítne do K2. Více o této funkci se dozvíme v 13.1. „Deploy“ – aplikování úprav do K2.

Tímto jsme nastavili, které datové moduly jsou připraveny pro ladění. Jakmile spustíme v K2 takto označený datový modul, okamžitě se nám otevírá editor skriptu, o kterém jsme blíže mluvili v kapitole 4.18. Editor, a je ve stavu ladění. Vykonávání kódu se nám zastaví v sekci, kde začíná spouštění datového modulu, viz Obrázek 353 - Ladění skriptu v editoru skriptů.



Obrázek 353 - Ladění skriptu v editoru skriptů

V tomto režimu si otevřeme skriptovou unitu, která je vygenerována návrhářem objektů pro datový modul, který chceme ladit. V místech ladění si nastavíme breakpointy, na kterých se nám aplikace zastaví, a můžeme postupně ladit. Například na Obrázek 354 - Nastavení breakpointu pro ladění v editoru skriptu, máme nastavený breakpoint v proceduře, která je vyvolána při spuštění metody ***„AccessDenied"***, kterou jsme si upravili.

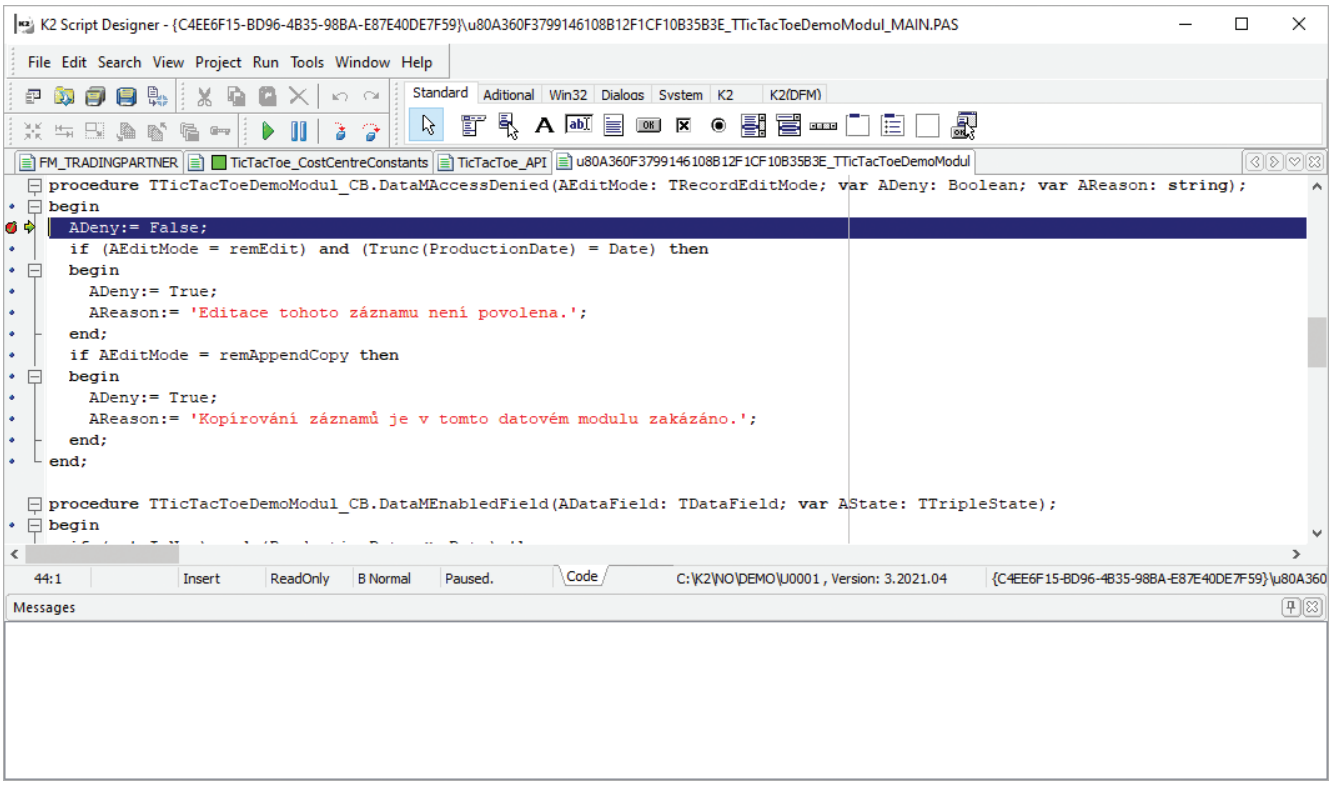


Obrázek 354 - Nastavení breakpointu pro ladění v editoru skriptu

Po nastavení breakpointu stiskneme zelené tlačítko Run v nástrojové liště nebo stiskneme klávesu F9. Tím povolíme zastavenému skriptu, aby pokračoval dále ve spouštění. Dokončí se nájezd datového modulu a zobrazí se nám formulář. V datovém modulu vyvoláme akci, kterou chceme ladit. V našem příkladu se jedná vstupu modulu do editačního režimu. Na Obrázek 355 - Zastavení skriptu na breakpointu v editoru skriptů, můžeme vidět zastavení v místě definovaného breakpointu. Dále již můžeme skript krokovat a ladit problémové části.

POZNÁMKA: Pokud spouštíme datový modul, který je označený k ladění, editor skriptů se spustí nezávisle na způsobů spouštění. Tedy v případě spouštění formou náhledu, který je popsán v 4.17. Náhled, kdy se jedná o standardní formuláře K2, tak i formou univerzálních formulářů.

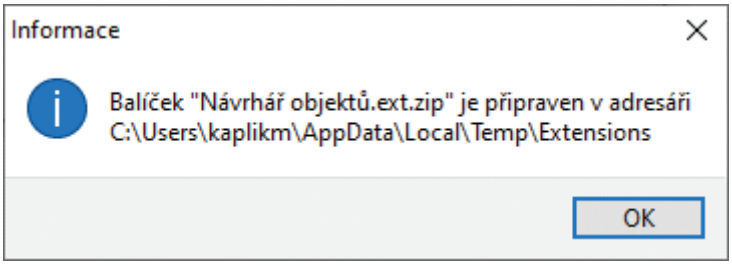
POZNÁMKA: Režim ladění se u vybraných datových modulů zruší automaticky po ukončení K2 nebo ručním zrušením nastavení ladění. Dokud je zatrženo, vždy se spustí, takto nastavený datový modul, v režimu ladění.



Obrázek 355 - Zastavení skriptu na breakpointu v editoru skriptů

13.3. „BALÍČEK“

Pomocí této funkce je možné vytvořit instalační balíček se všemi objekty, které jsou vytvořené v rámci definice rozšíření v návrháři objektů. Po spuštění se vytvoří balíček a uloží se v uživatelském adresáři Windows, viz hlášení Obrázek 356 - Hlášení o vytvoření instalačního balíčků, které se objeví na konci této operace.



Obrázek 356 - Hlášení o vytvoření instalačního balíčků

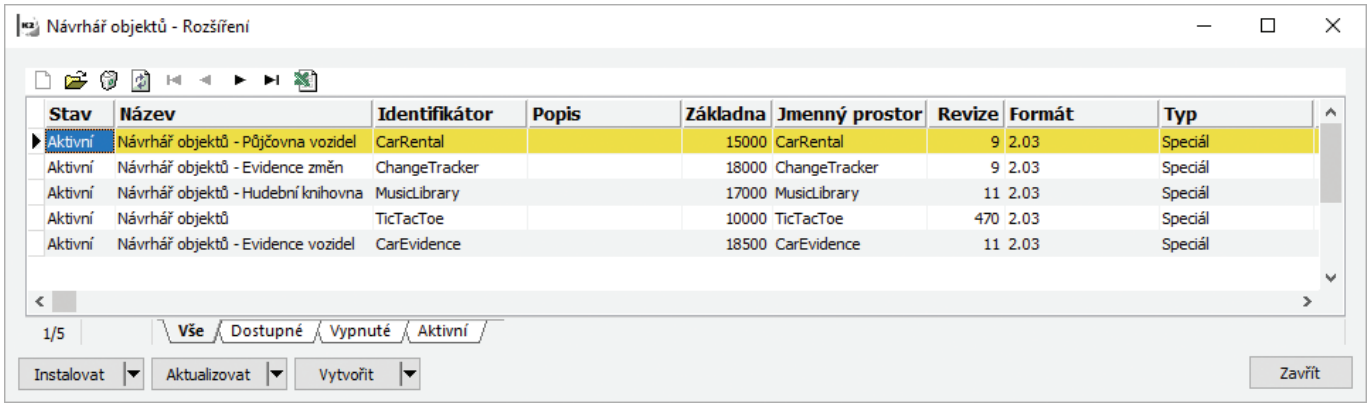
POZNÁMKA: V případě, že máme k aktuální definici připojené další balíčky skrz záložku ***„Závisí na“***, viz kapitola 12. Záložka „Závisí na“, přibalí se do vytvářeného balíčku i tyto, na kterých jsme závislí. Máme tak konzistentní instalační balíček, který obsahuje vše, co je potřeba k tomu, abychom ho mohli kdekoliv nainstalovat a použít.

Takto vytvořený balíček pak můžeme použít k instalaci do jiné instalace IS K2 nebo třeba k záloze. Jak se dále pracuje s balíčkem, si ukážeme v následující kapitole 13.4. „Rozšíření“.

13.4. „ROZŠÍŘENÍ“

Tato funkce slouží ke správě definic (rozšíření) návrháře objektů. Pomocí tohoto nástroje můžeme instalovat do IS K2 různá rozšíření, která jsou připravena v podobě balíčků. Instalované balíčky můžeme instalovat, odinstalovat, aktivovat a deaktivovat. Vše si popíšeme v následujícím textu.

Po spuštění nástroje ***„Rozšíření“*** se zobrazí formulář, který je vidět na Obrázek 357 - Seznam rozšíření a přidání nového rozšíření. V první záložce ***„Vše“*** vidíme všechny importované, instalované, aktivní i neaktivní definice rozšíření K2. Další záložky jsou pak již podmnožiny definic rozšíření dle jejich aktuálního stavu.

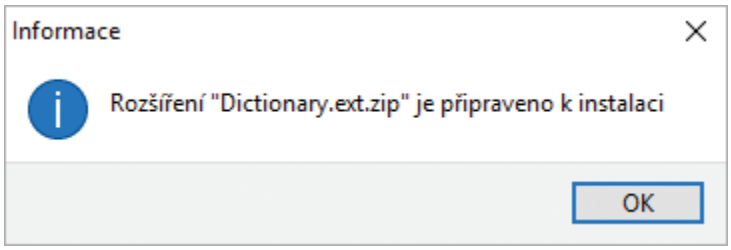


Obrázek 357 - Seznam rozšíření a přidání nového rozšíření

13.4.1. INSTALACE ROZŠÍŘENÍ

Instalaci rozšíření můžeme provést dvěma způsoby.

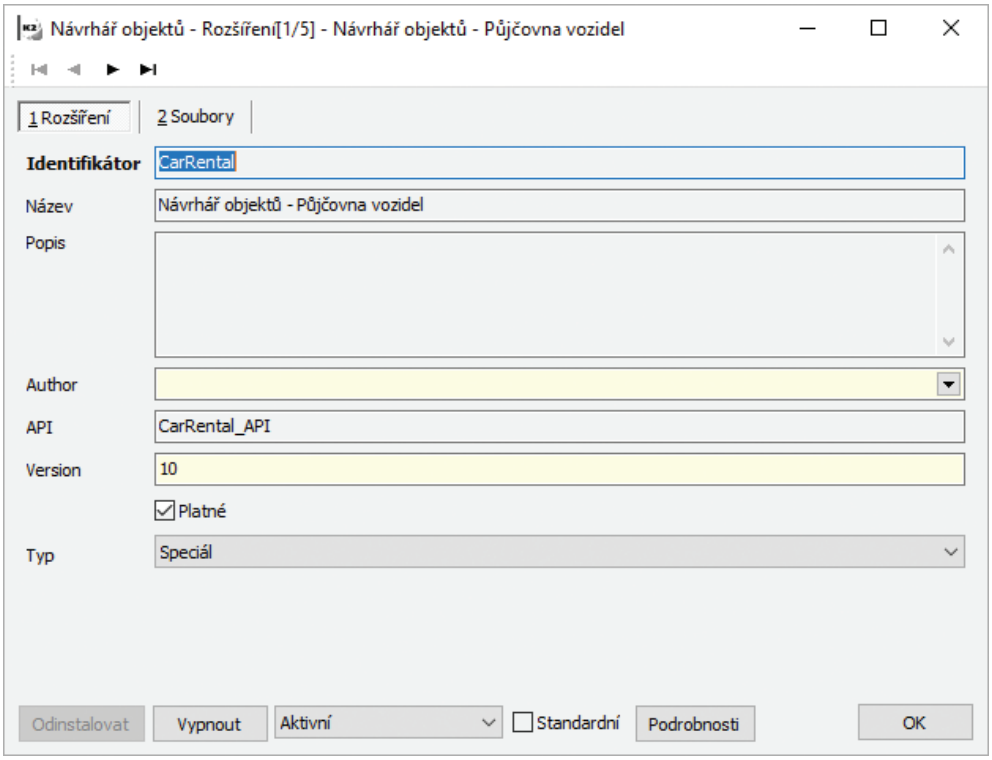
První způsob je pomocí stisknutí tlačítka **Instalovat**, které je vidět na Obrázek 357 - Seznam rozšíření a přidání nového rozšíření. Je zde na výběr několik způsobů, buď pomocí standardu, převzít, nebo z konkrétního adresáře, který se používá nejčastěji. Po stisku se zobrazí dialog na výběr souboru s definici rozšíření, kterou požadujeme vložit. Typický název souboru pro balíček, je ve tvaru ***„název.ext.zip“***. Soubor nalezneme a stiskneme tlačítko pro potvrzení. Návrhář objektů zobrazí hlášení, že se bude balíček instalovat, viz Obrázek 358 - Dialog pro potvrzení přidání rozšíření. Po odklepnutí tlačítkem **OK** se balíček nainstaluje a zároveň se nastaví jako ***„Aktivní“***. Po úspěšné instalaci a aktivaci balíčku můžeme struktury již používat při implementaci aktuální definice rozšíření. V případě, že požadujeme přímo používat datové moduly a jejich funkce, musíme nejprve K2 restartovat.



Obrázek 358 - Dialog pro potvrzení přidání rozšíření

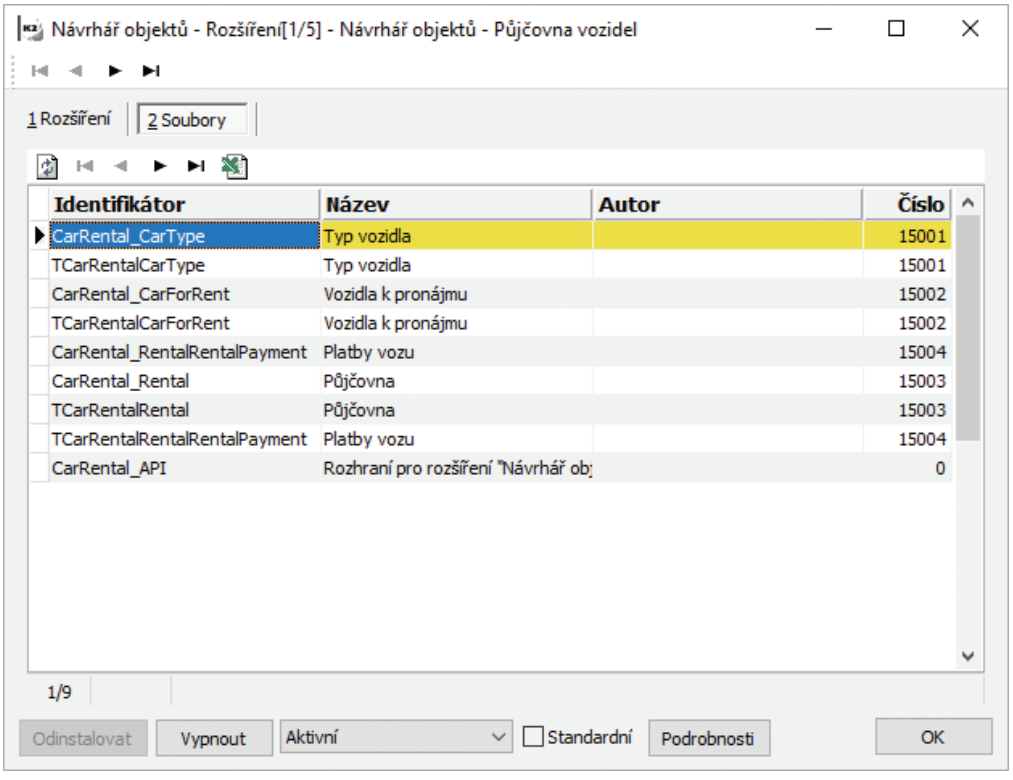
Druhou možností instalace je využití funkce ***„Drag&Drop“***. Jednoduše chytíme myší soubor s instalačním balíčkem a přetáhneme ho do okna rozšíření v návrháři objektů. Návrhář automaticky tento balíček nainstaluje a provede jeho aktivaci.

V případě, že bychom chtěli zjistit detailní údaje o balíčku, případně s ním provádět další operace (vypnutí, od instalace apod.) můžeme otevřít jeho detail, viz Obrázek 359 - Detail nainstalovaného a aktivního rozšíření. V detailu můžeme vidět dvě záložky. V první záložce ***„Rozšíření“*** je identifikátor, tedy jmenný prostor definice rozšíření, dále název, popis a název pro API - ***„aplikační rozhraní“*** atd. Ve spodní části se nacházejí tlačítka pro odinstalování, vypnutí/zapnutí, informace o tom, zda je aktivní dané rozšíření, zda se jedná o standardní rozšíření, tlačítko na podrobnosti (kde se nacházejí informace o tom, kdo kdy vytvořil nebo upravil rané rozšíření) a OK.



Obrázek 359 - Detail nainstalovaného a aktivního rozšíření

V druhé záložce ***„Soubory"*** pak můžeme vidět všechny soubory, které instalační balíček obsahuje. Soubory mohou být různých typů. Zde například na Obrázek 360 - Záložka "Soubory" na rozšíření můžeme vidět pro každý datový modul dva soubory, kdy jeden je typu ***„TDataMInstaller"***, představuje popis datového modulu, a druhý je typu ***„TxFileInstaller"***, který představuje popis databázové tabulky pro datový modul.



Obrázek 360 - Záložka "Soubory" na rozšíření

POZNÁMKA: Všechny soubory se při instalaci balíčku kopírují do složky ***„Extensions“*** v adresáři K2. Více o funkčnosti této složky je popsáno v kapitole 13.1. „Deploy“ – aplikování úprav do K2.

POZNÁMKA: Po úspěšné instalaci a aktivaci můžeme ihned začít používat struktury z nainstalovaného balíčku v naší definici rozšíření. Pokud bychom, ale chtěli začít používat datové moduly v K2, tak je nejprve nutné provést restart K2, aby došlo k načtení těchto nových struktur.

13.4.1.1. Problémy při instalaci a stav balíčku „Dostupné“

Může nastat situace, kdy se nepodaří balíček nainstalovat. Například dojde k chybě při instalaci. Pak je chyba zobrazena a po potvrzení hlášení je balíček zaveden do rozšíření, ale není aktivní, má stav ***„Dostupné“***.

Tento stav značí, že je definice rozšíření připravena, ale není nainstalována. Do stavu ***„Dostupné“*** se může balíček dostat buď ručním nastavením, v případě, že ho odinstalujeme nebo chybou při instalaci, viz text výše.

Balíček v tomto stavu můžeme buď nainstalovat, nebo smazat.

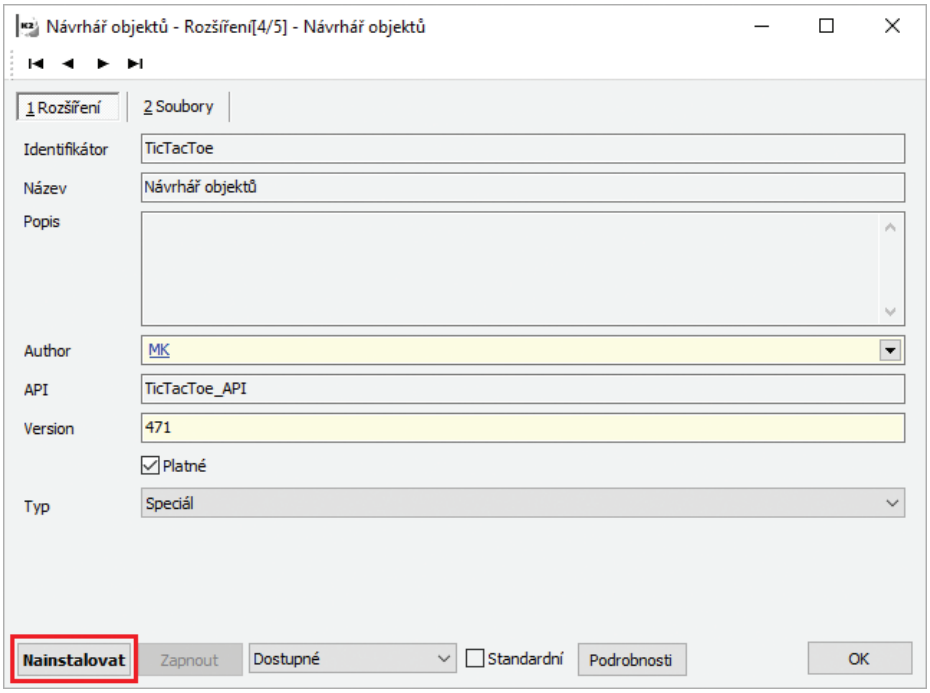
13.4.1.2. Výmaz balíčku

Výmaz provedeme pomocí tlačítka Odstranit záznam v nástrojové liště nad seznamem nebo stisknutím klávesy Delete. Záznam zmizí ze seznamu.

POZNÁMKA: Abychom balíček mohli smazat z rozšíření, musí být nejprve ve stavu ***„Dostupné“***.

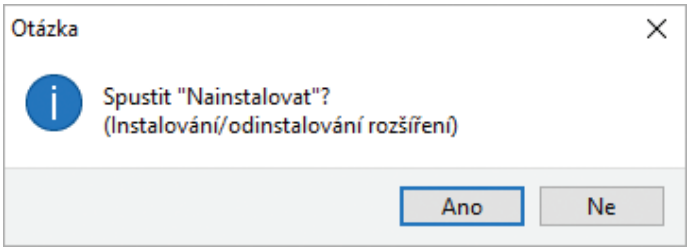
13.4.1.3. Instalace balíčku ze stavu „Dostupné“

V případě, že máme balíček ve stavu ***„Dostupné“*** je možné ho nainstalovat. Před samotnou instalací musíme nejprve přejít do detailu definice rozšíření. To provedeme stisknutím tlačítka Upravit záznam v nástrojové liště nad seznamem nebo stisknutím klávesy F5. Zobrazí se nám formulář, který můžeme vidět na Obrázek 361 - Detail rozšíření a instalace.



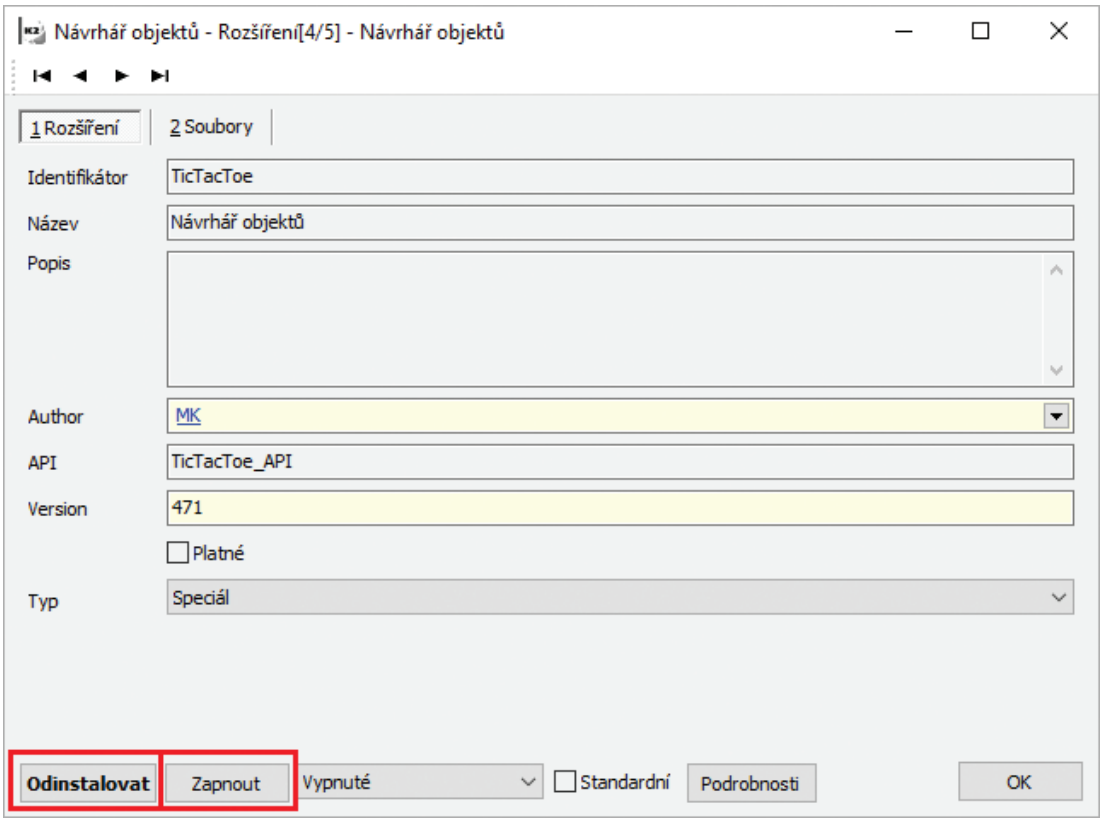
Obrázek 361 - Detail rozšíření a instalace

Samotnou instalaci provedeme pomocí stisknutí tlačítka Nainstalovat. Návrhář objektů si vyžádá potvrzení, zda opravdu chceme balíček instalovat, viz Obrázek 362 - Potvrzení instalace rozšíření.



Obrázek 362 - Potvrzení instalace rozšíření

Po potvrzení se balíček nainstaluje, což poznáme tak, že se jeho stav přepne na hodnotu ***„Vypnuté“*** a zobrazí se tlačítka Odinstalova a Zapnout, viz Obrázek 363 - Aktivace a odinstalace rozšíření.



Obrázek 363 - Aktivace a odinstalace rozšíření

13.4.1.4. Odinstalace balíčku

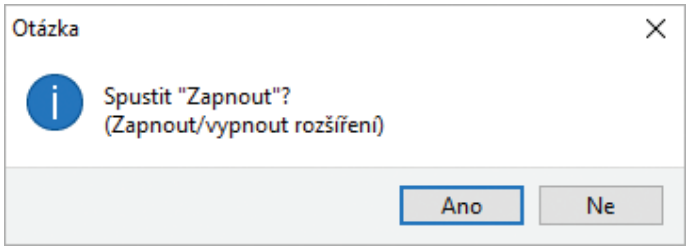
Pokud bychom chtěli balíček odinstalovat, musí být nejprve ve stavu ***„Vypnuté“***, tedy neaktivní. Odinstalaci definice rozšíření provedeme stisknutím tlačítka Odinstalovat. V ten moment se definice odinstaluje a přepíná se zpět do stavu ***„Dostupné“***.

POZNÁMKA: V případě, že chceme instalovanou definici úplně odebrat z nástroje ***„Rozšíření“***, je potřeba ji nejprve odinstalovat, tím se dostane do stavu ***„Dostupné“***, a následně můžeme definici odstranit. Pokud by definice rozšíření byla aktivní, pak navíc musíme provést deaktivaci, následně odinstalaci a až poté odebrání. Aktivaci a deaktivaci si probereme v následující kapitole.

13.4.2. AKTIVACE / DEAKTIVACE ROZŠÍŘENÍ

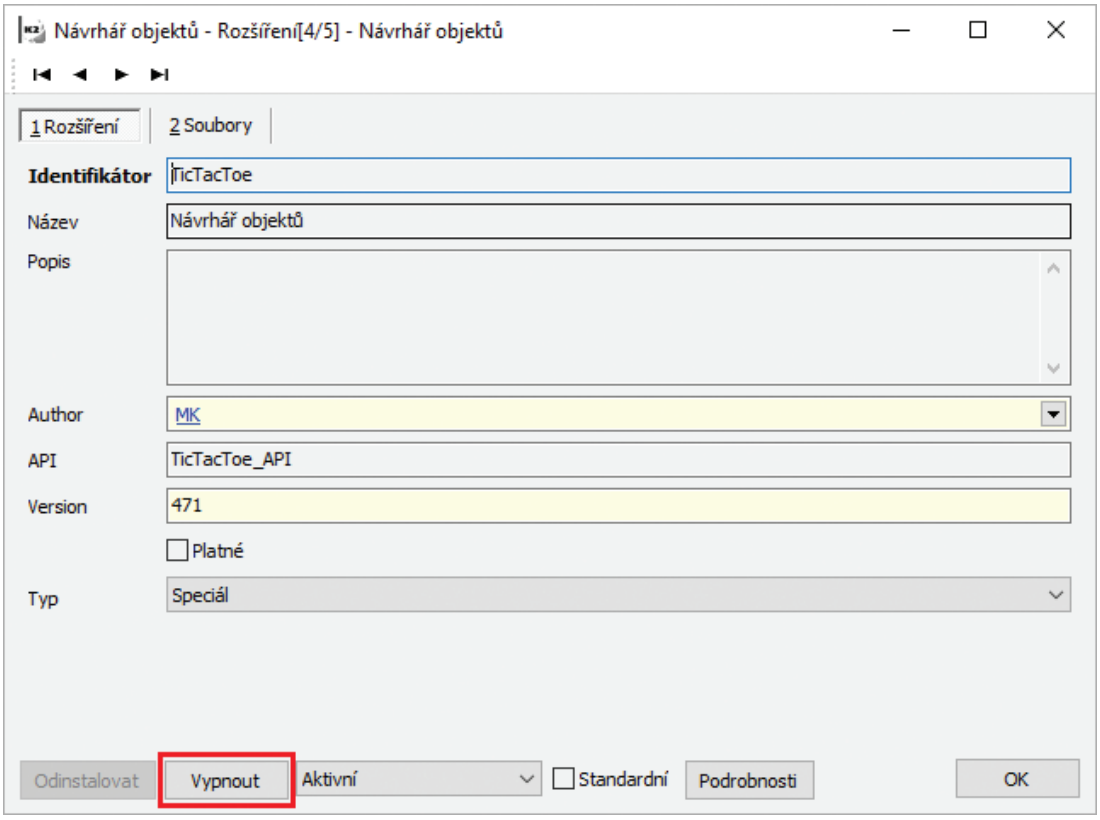
Nainstalovaný balíček je automaticky nastaven do stavu ***„Aktivní“***. V případě, že bychom chtěli balíček deaktivovat, stiskneme tlačítko Vypnout, viz Obrázek 365 - Deaktivace rozšíření.

V případě, že je definice rozšíření neaktivní, je ve stavu ***„Vypnuté“***. Abychom ji mohli používat, je potřeba ji aktivovat neboli zapnout. To provedeme pomocí tlačítka Zapnout, které je vidět na Obrázek 363 - Aktivace a odinstalace rozšíření. Nástroj se nás zeptá, zda opravdu chceme provést aktivaci, viz dialogové okno Obrázek 364 - Potvrzení aktivace rozšíření.



Obrázek 364 - Potvrzení aktivace rozšíření

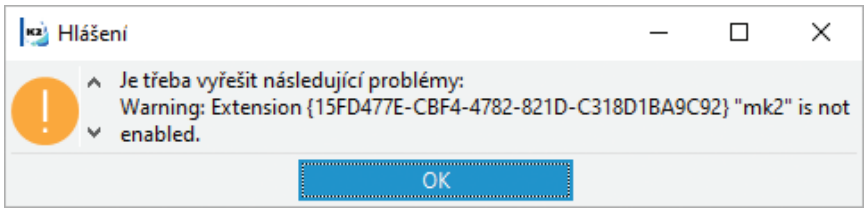
Po potvrzení se definice rozšíření aktivuje. Poznáme to tak, že se přepne do stavu ***„Aktivní“*** a tlačítko bude přepnuto do stavu ***„Vypnout“***, jak můžeme vidět na Obrázek 365 - Deaktivace rozšíření. Stisknutím tlačítka Vypnout definici samozřejmě zpět deaktivujeme.



*Obrázek 365 - **Deaktivace rozšíření***

POZNÁMKA: Pro úplné odebrání definice rozšíření musíme nejprve aktivní definici vypnout, odinstalovat, a nakonec můžeme úplně odebrat, tak jak bylo popsáno již výše v textu.

POZNÁMKA: Pokud máme v K2 rozšíření, které není aktivní, dojde při spuštění K2 k informativnímu oznámení (pouze uživatelům s právy správce, viz Obrázek 366 - Hlášení o neaktivní rozšíření), že se nachází neaktivní rozšíření v adresáři Extensions. Takovéto rozšíření doporučujeme odstranit.



Obrázek 366 - Hlášení o neaktivní rozšíření

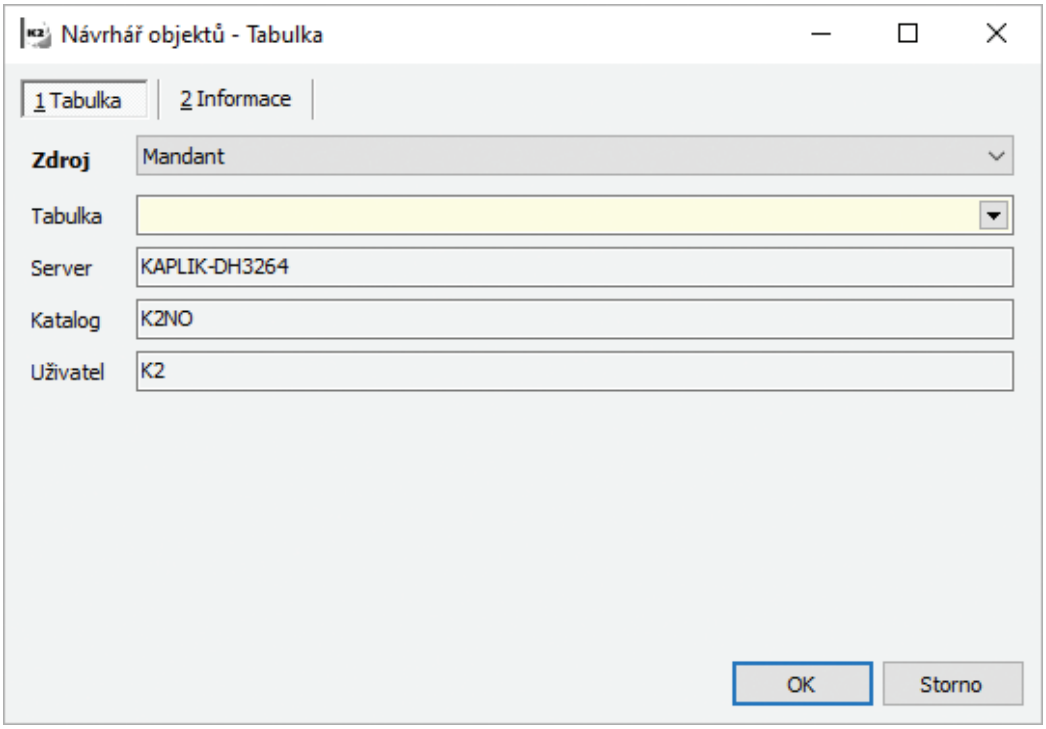
13.5. „NASTAVENÍ“

S funkcí nastavení návrháře objektů jsme se již seznámili v kapitole 2.2. Nastavení návrháře objektů. Nebudeme se zde tedy nastavením dále zabývat.

13.6. IMPORT

Dalším nástrojem v návrháři objektů, je možnost importovat databázové tabulky. Využít této funkce můžeme například, v případě, že máme implementovánu speciální úpravu mimo návrhář objektů a chtěli bychom funkčnost převést do návrháře. Místo ruční definice všech tabulek, tak můžeme využít této funkce, která nám vytvoří v definici rozšíření datové moduly dle struktury vstupních databázových tabulek. V následujícím textu popíšeme tuto funkci.

Po spuštění importu se zobrazí formulář, viz Obrázek 367 - Vstupní formulář pro import tabulky, pro zadání vstupních informací. Návrhář objektů umožňuje importovat tabulky pouze z databází dostupných v K2, tedy databáze CONF a databáze aktuálního mandanta. Některé vstupní údaje jsou proto vyplněny dle aktuální K2.



Obrázek 367 - Vstupní formulář pro import tabulky

Formulář se skládá ze dvou záložek. První je nazvána *„Tabulka"* a obsahuje následující informace.

Zdroj
Zde je potřeba nastavit z jakého zdroje bude načten seznam dostupných tabulek, a tedy i zdrojová pro import. K výběru jsou k dispozici následující možnosti.

Conf – vstupem je databáze CONF z aktuální instalace K2.

Mandant – vstupem je databáze aktuálního mandanta K2.

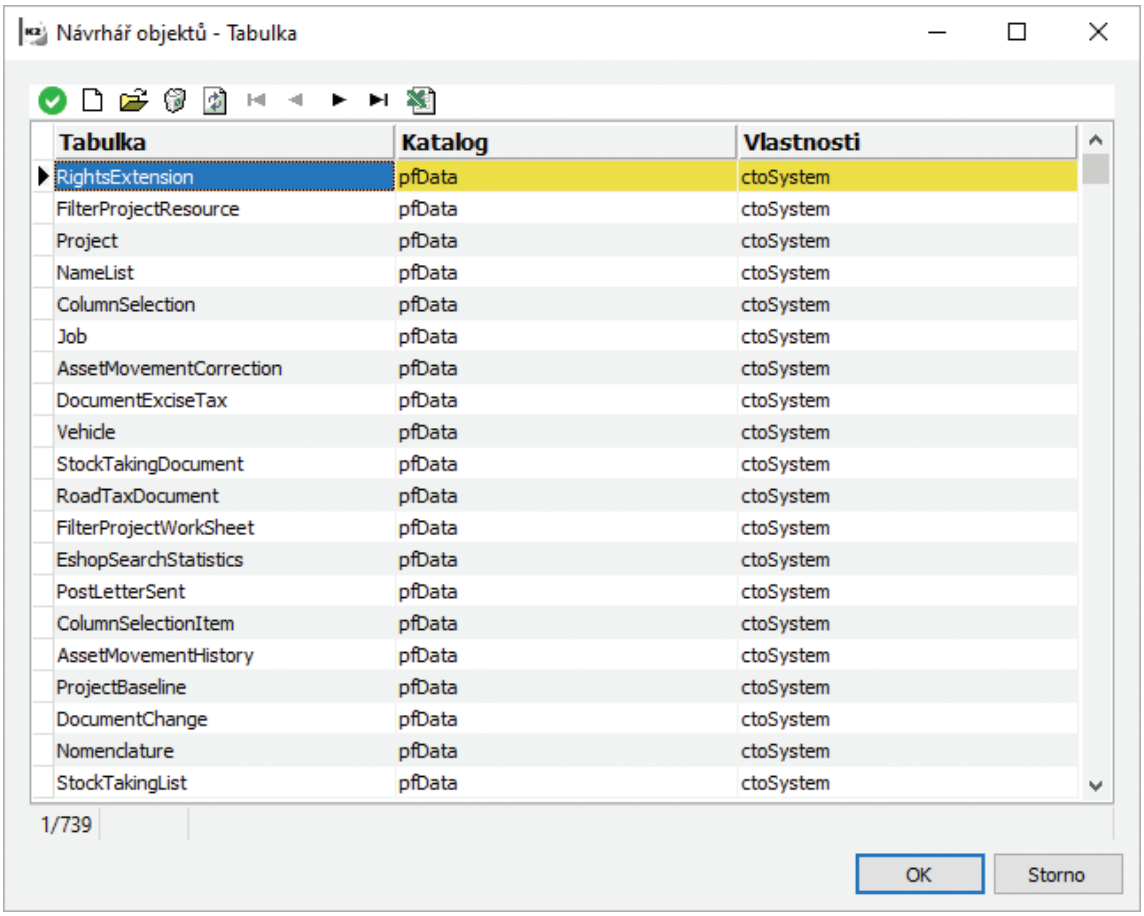
Neurčeno – jedná se o systémovou záležitost, nelze vybrat.

Server
Název databázové serveru, na kterém je umístěna databáze. Automaticky je vyplněno dle databázového serveru, na kterém je provozována aktuální K2.

Katalog
Název katalogu K2, tedy databáze CONF, v případě výběru typu databáze mandant, se pak bere katalog, jako prefix pro databázi aktuálního mandanta, ze kterého se budou nabízet tabulky pro import.

Uživatel
Zde je uvedeno přihlašovací jméno, přesněji *„SQL login"*, pod kterým se K2 přihlašuje k databázovému serveru a katalogu, který je výše nastaven. Pod tímto uživatelem proběhne import tabulky.

Tabulka
V této části máme k dispozici rozevírací nabídku, kde se nám po otevření zobrazí všechny dostupné databázové tabulky z definovaného zdroje, viz Obrázek 368 - Seznam tabulek k importu.



Obrázek 368 - Seznam tabulek k importu

Dle hodnoty ve sloupci ***„Katalog“*** můžeme rozpoznat, zda se jedná o databázi mandanta – ***„pfData“ nebo CONF – „pfConf“***.

Import tabulky si popíšeme na příkladu konkrétní tabulky.

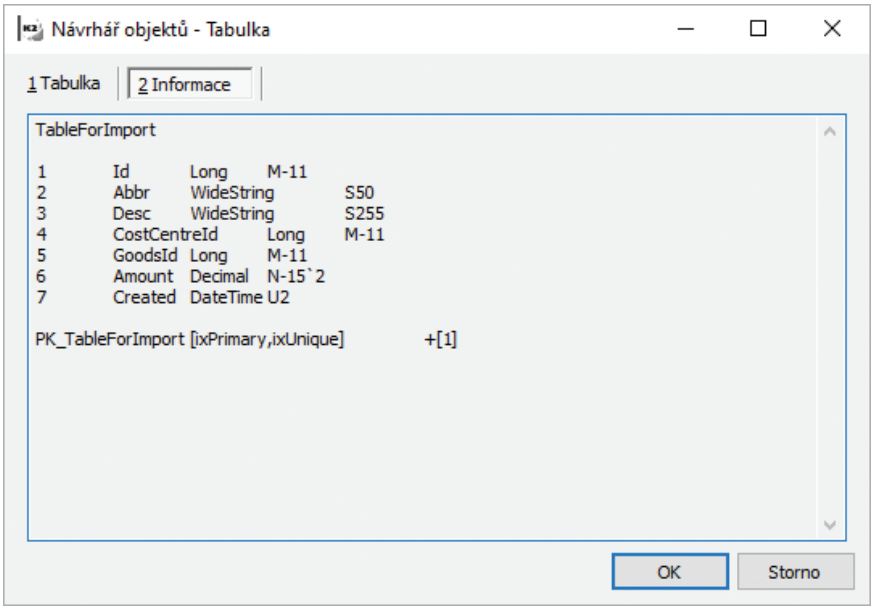
PŘÍKLAD: V databázi mandanta K2 máme k dispozici tabulku s názvem ***„TableForImport“***. Na Obrázek 369 - Struktura tabulky na MSSQL serveru je vidět struktura tabulky na MS SQL serveru. Můžeme vidět názvy sloupců, včetně jejich datových typů.

KAPLIK-DH3264.GAI...bo.TableForImport

Column Name	Data Type	Allow Nulls
Id	int	☐
Abbr	nvarchar(50)	☐
[Desc]	nvarchar(255)	☐
CostCentreId	bigint	☐
GoodsId	int	☐
Amount	money	☐
Created	datetime	☐
		☐

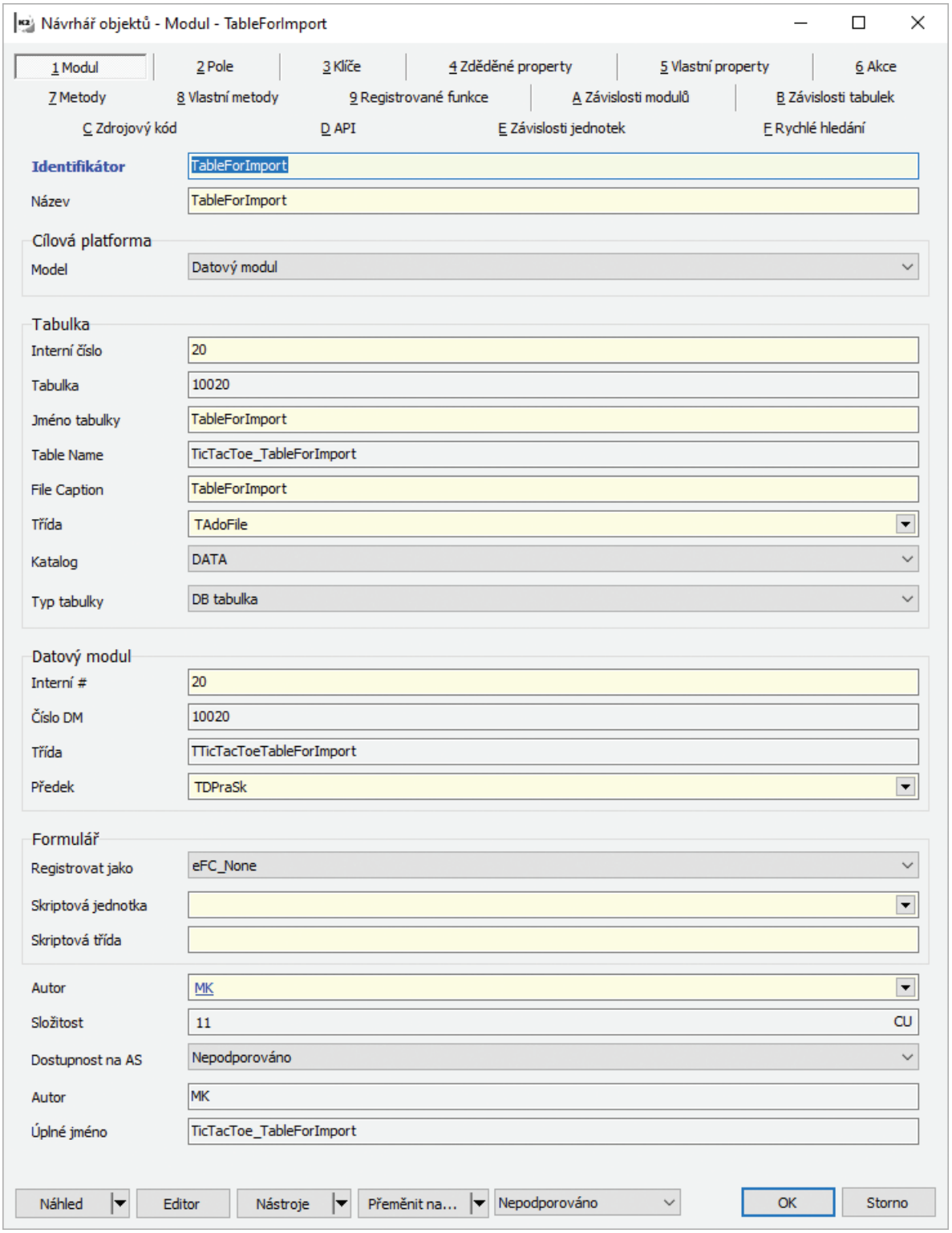
Obrázek 369 - Struktura tabulky na MSSQL serveru

Po výběru tabulky ***„TableForImport“*** se načtou informace o tabulce do záložky ***„Informace“***, které můžeme vidět na Obrázek 370 - Načtené informace o tabulce pro import. Z informací můžeme vyčíst názvy sloupců s datovými typy, které jsou převedeny na datové typy, které používá K2. Popsány jsou i indexy tabulky.



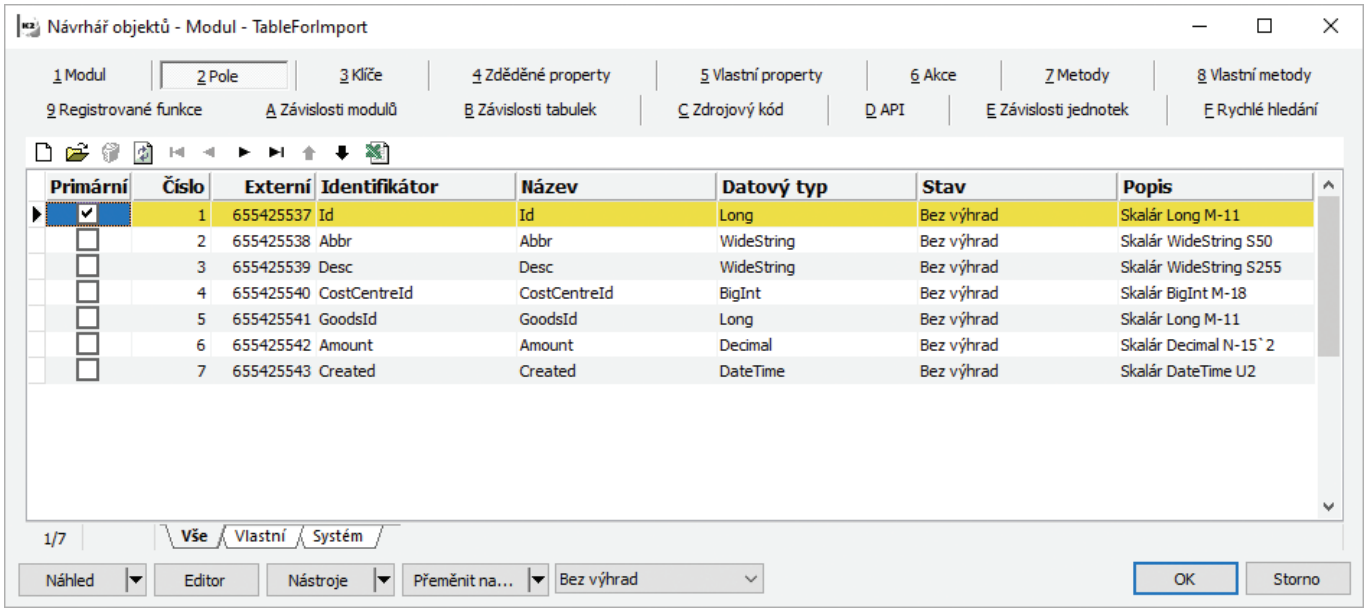
Obrázek 370 - Načtené informace o tabulce pro import

Po stisknutí tlačítka OK na formuláři, potvrdíme výběr tabulky k importu. Návrhář objektů vytvoří nový datový modul, který pojmenuje podle vstupní tabulky, dle sloupců vytvoří jednotlivá pole datového modulu a podle indexů tabulky také klíče. Základní informace o datovém modulu můžeme vidět na Obrázek 371 - Importovaný datový modul.



Obrázek 371 - Importovaný datový modul

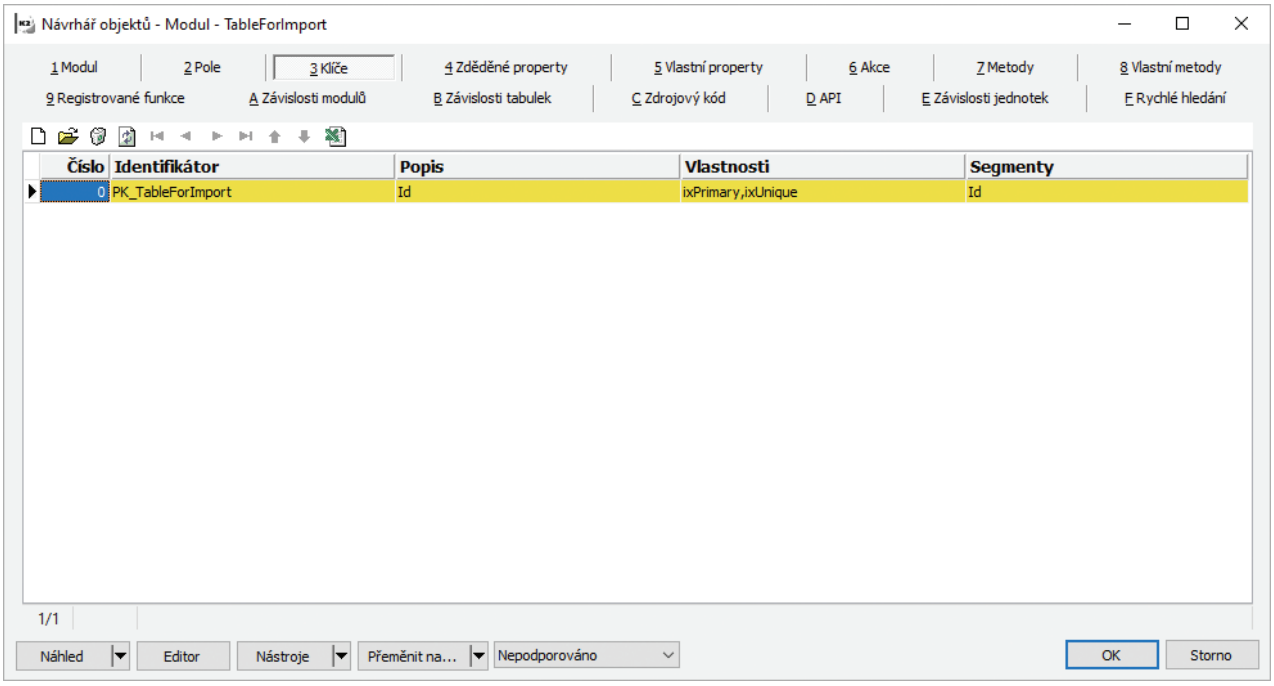
Seznam polí, která jsou vytvořená dle importované tabulky, můžeme vidět na Obrázek 372 - Seznam polí importovaného datového modulu.



Obrázek 372 - Seznam polí importovaného datového modulu

V případě, že některé z polí ve zdrojové databázové tabulce je cizím klíčem, v návrháři objektů takto načteno nebude. Všechna pole jsou vždy vytvořena jako skaláry. Návrhář objektů není schopen rozpoznat, že existuje nad tabulkou taková definice. Pole, která jsou cizími klíči, případně přímými vazbami, můžeme jednoduše na tyto typy převést pomocí funkce ***„Přeměnit na“***, která je popsána v kapitole 4.2.4. Rychlá změna typu pole – funkce „Přeměnit na“.

Seznam klíčů, které jsou naimportovány do datového modulu je vidět na Obrázek 373 - Seznam klíčů importovaného datového modulu.



Obrázek 373 - Seznam klíčů importovaného datového modulu

POZNÁMKA: Datový modul importovaný z databázové tabulky má nastaveného předka jako *„TDPraSk“*.

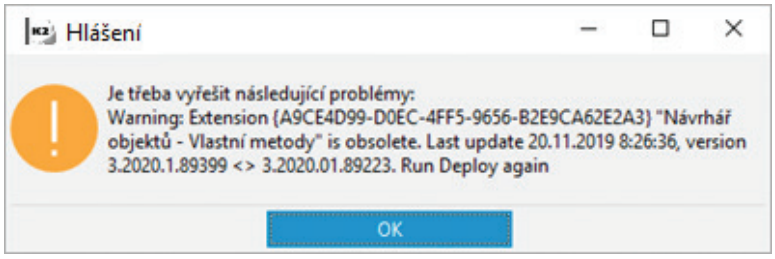
POZNÁMKA: Po importu tabulky vznikne nový datový modul, který má jako datový zdroj novou databázovou tabulku, která má v názvu navíc prefix dle nastavení definice rozšíření. Tímto máme zajištěno, že nedojde ke smazání či jakémukoliv poškození zdrojové tabulky.

13.7. „ZAVŘÍT“

Pomocí tohoto tlačítka ukončíme práci v návrháři objektů. Po jeho stisknutí se formulář zavře a probíhá uvolňování držených záznamů, které popisují definici v návrháři objektů. Tato operace může trvat 1–2 vteřiny v závislosti na rychlosti počítače. Návrhář objektů, můžeme samozřejmě uzavřít i pomocí tlačítka ☒ v záhlaví formuláře, jak je to běžné u všech ostatních formulářů.

14. AKTUALIZACE ROZŠÍŘENÍ PO REINSTALACI

Po každé reinstalaci K2 je nutné provést i aktualizaci všech rozšíření, které jsou na K2 nainstalovány. V případě, že neproběhne aktualizace, je při spouštění K2, uživatelům, kteří mají právo na servisní zásahy, zobrazeno hlášení o rozšířeních, které nejsou aktuální, viz Obrázek 374 - Hlášení - zastaralé rozšíření v K2.

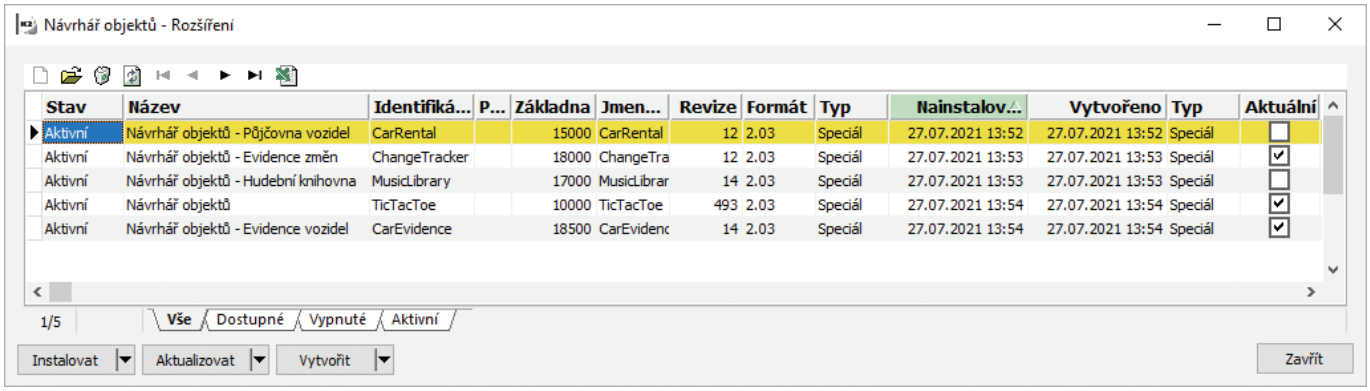


Obrázek 374 - Hlášení - zastaralé rozšíření v K2

Ve verzi K2 luna je v reinstalačním procesu K2 zaveden mechanismus, který zajistí automatickou aktualizaci všech rozšíření instalovaných v K2. Tedy, v případě, že vše proběhne bez problémů, nemělo by se po reinstalaci hlášení zobrazovat.

V případě, že je rozšíření zastaralé (zobrazuje se hlášení), je nutné provést aktualizaci ručně. K2 se musí spustit se speciálním parametrem EXTENSIONS=UPDATE a doporučuje se i parametr RSOFF=1. V tomto momentě K2 začne aktualizovat všechny rozšíření, které jsou na dané K2 zastaralé. V závislosti na rozšířeních je možné, že bude během aktualizace vyžadován restart K2. Poznáme to tak, že se K2 ukončí. V tomto momentě je potřeba spustit aktualizaci znovu se zmíněným parametrem. Jakmile K2 zůstane po aktualizaci spuštěná, je vše v pořádku aktualizováno a hlášení by se již nemělo zobrazovat.

POZNÁMKA: Stav rozšíření je možné ověřit ve formuláři pod funkcí ***„Rozšíření“*** v Návrháři objektů, kde je u každého informace o stavu. Jedná se o sloupec ***„Aktuální“***, viz Obrázek 375 - Stav rozšíření na K2.



Obrázek 375 - Stav rozšíření na K2

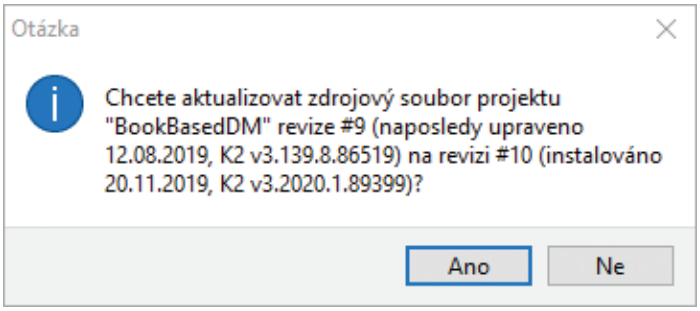
POZNÁMKA: Rozšíření bude na K2 fungovat i v případě, že bude zastaralé. Není ale vhodné v tomto stavu K2 ponechat. Při operaci aktualizace dochází ke konverzi struktur, doplňují se nové, případně zrušené vlastnosti, které jsou zohledněny až po aktualizaci a je tedy doporučené udržovat zkompilované soubory rozšíření ve stejné verzi jako je verze K2.

POZNÁMKA: V případě přechodu na verzi K2 mia.03 je nutné provést konverzi formátu rozšíření z verze 1.0 na verzi 2.01. Tímto tématem se zabývá následující text.

14.1. KONVERZE STRUKTURY ROZŠÍŘENÍ DO NOVÉHO FORMÁTU

Ve verzi K2 mia (přesněji K2 mia.03) došlo ke změně formátu souborů pro rozšíření Návrháře objektů. Jedná se jak o definiční xml soubor (rozšíření), tak o soubory, které vznikají pro běh K2 (extensions). Struktury byly povýšeny na verzi 2.02. Informaci lze vidět na Obrázek 375 - Stav rozšíření na K2, ve sloupci ***„Formát“***. Pokud proběhne reinstalace na tuto verzi K2, rozšíření budou fungovat dále, ale jen do doby, kdy je potřeba provést nějakou úpravu. Před jakoukoliv modifikací je nutné nejprve provést konverzi na novou verzi 2.02.

Konverze se spouští úplně stejně jako aktualizace, tedy parametrem EXTENSIONS=UPDATE. Dokud tato operace neproběhne, není možné modifikovat rozšíření v Návrháři objektů. Po konverzi je v adresáři ***„extensions“*** již nová struktura. Pro definici rozšíření, ale zůstala stará varianta xml souboru. K2 nezná umístění zdroje pro adresář ***„extensions“***, nelze proto provést konverzi definice automaticky. Následně pokud se uživatel pokusí otevřít (po konverzi) zastaralou definici rozšíření, K2 pozná, že již existuje v adresáři ***„extensions“*** novější rozšíření a automaticky uživateli nabídne převod definice na novou strukturu v místě, ze kterého definici otevírá. Převod je nutné potvrdit, jinak Návrhář objektů nedovolí dále s touto definicí pracovat, viz Obrázek 376 - Informace o zastaralé definici rozšíření.



Obrázek 376 - Informace o zastaralé definici rozšíření

POZNÁMKA: V textu je zmíněno, že K2 bude fungovat na staré struktuře, dokud nebude potřeba její modifikace. Tento režim samozřejmě nedoporučujeme. Správce by měl provést konverzi na novou strukturu, aby bylo zajištěno, že verze zdrojů rozšíření je vytvořena na stejné verzi K2, na které je spuštěno.

14.2. MOŽNOSTI PARAMETRU EXTENSIONS

Parametr EXTENSIONS nabývá několika hodnot, které je možné použít. Jedná se o následující varianty:

PARAMETR	POPIS
EXTENSIONS=OFF	K2 se spustí bez rozšíření
EXTENSIONS=UPDATE	Při spuštění se provede aktualizace všech rozšíření instalovaných v K2, která jsou zastaralá.
EXTENSIONS=UPDATE_ALL	Při spuštění se provede aktualizace všech rozšíření instalovaných v K2 bez ohledu na to, zda jsou zastaralá či nikoli. Je vhodné využít při přenosu adresáře ***„extensions“*** z jiné, testovací K2, která má shodnou verzi jako aktuální K2.

Poznámky

Poznámky

Poznámky

Poznámky

Poznámky

Poznámky

Poznámky