Reportovací nástroj pro .NET aplikace

Reporting tool for .NET applications

Bc. Tomáš Mořkovský

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš MOŘKOVSKÝ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Reportovací nástroj pro .NET aplikace**

Zásady pro vypracování:

1. **Proveďte průzkum informačních zdrojů k danému tématu.**
2. **Analyzujte a popište obecnou funkčnost stávajících řešení a porovnejte je.**
3. **Vypracujte obecný návrh univerzálního reportovacího nástroje.**
4. **Realizujte reportovací nástroj dle obecného návrhu schopný všech základních funkcí v prostředí .NET.**
5. **Vypracujte stručnou uživatelskou dokumentaci.**
6. **Realizujte sadu ukázkových aplikací využívající reportovač.**

Rozsah práce: **59 stran**

Rozsah příloh: **bez příloh**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KANISOVÁ, Hana, MÜLLER, Miroslav, UML srozumitelně : Computer Press, 2004. 158 s. ISBN 80-251-0231-9**
2. **Design patterns : Data & Object Factory [online]. [cit. 2009-01-23]. Dostupný z WWW: http://www.dofactory.com/Patterns/Patterns.aspx**
3. **TROELSEN, Andrew, Pro C-SHARP 2008 and the .NET 3.5 Platform : Apress, 2007. 1370 s. ISBN 1-59059-884-9**
4. **NAGEL, Christian, EVJEN, Bill, GLYNN, Jay, SKINNER, Morgan, WATSON, Karli. Professional C-SHARP 2008 : Wiley Publishing, Inc. (Wrox), 2008. 1782 s. ISBN 978-0-470-19137-8**
5. **Contoli, A., Simple Vector Shapes [online]. naposledy editováno 2008-03-27. Dostupný z WWW: http://www.codeproject.com/KB/graphics/SimpleVectorShapes.aspx**
6. **VUGT, Wouter van. Open XML The markup explained [online]. 2007 [cit. 2009-01-23]. Dostupný z WWW: http://openxmldeveloper.org/attachment/1970.ashx**

Vedoucí diplomové práce: **doc. Mgr. Roman Jašek, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Reportovacím nástrojem nazýváme aplikaci nebo službu schopnou z velkého množství dat v datovém zdroji vytvořit přehlednou sestavu, která bude mít pro uživatele skutečnou informační hodnotu. Hlavním cílem této práce je takový nástroj navrhnout a vytvořit, konkrétně pro platformu Microsoft .NET. Nebude to v žádném případě první produkt tohoto druhu, ale bude to otevřené řešení pro drobné vývojáře, pro které jsou komerční nástroje zbytečné komplexní a hlavně drahé. V teoretické části práce bude popsána obecná koncepce a možnosti těchto nástrojů. Zejména jejich funkční analýzou je navrhnuto vlastní řešení. Praktická část práce je věnována vývoji aplikace, od specifikace funkčnosti, přes analytický návrh k finální aplikaci včetně ukázkového použití.

Klíčová slova: reportovací nástroj, sestava, šablona, dokument

ABSTRACT

Reporting tool called an application or service capable of a large amount of data in the data sources to create a clear set of which will have information for users of genuine value. The main objective of this thesis is a tool to design and create a platform specifically for Microsoft .NET. This will not in any case, the first product of this kind, but it will be open solutions for small developers, which are quite complex and commercial tools and mostly unnecessary expensive. In the theoretical part of the work will be described the general concepts and possibilities of these instruments. In particular, their functional analysis is proposed its own solution. Practical work is devoted to the development of applications, from the specification of functionality, through an analytical proposal for the final application, including a sample application.

Keywords: reporting tool, report, template, document

Poděkování, motto

Velmi rád bych poděkoval a vyslovil uznání všem, kteří mi pomáhali při vzniku této práce. Především panu doc. Mgr. Romanu Jaškovi, Ph.D., vedoucímu mé diplomové práce, za vstřícné vedení, cenné rady a připomínky.

Také bych chtěl poděkovat rodičům za poskytnuté zázemí a podporu po celou dobu studia.

„Každá lidská činnost se nakonec musí nějak projevit v číslech.“

Tomáš Baťa

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval.

V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně 27. května 2009 …….

Podpis diplomanta

OBSAH

ÚVOD

Data a informace, to jsou slova, která slýcháme v posledních letech stále častěji a nejinak tomu bude i v letech následujících. Shromažďování dat a informací člověka provází od pradávna, aniž by si to uvědomoval. V době kamenné lidé kreslili své poznatky na zeď v jeskyni či na pláži do písku, později své znalosti předávali v psané formě na papyru a papíru, a není tomu tak dávno co přenos informací začala zprostředkovávat technika. Se stále rychlejší možností pořizování, přístupnější formou přenosu a masového šíření také roste množství shromažďovaných, přenášených a zpracovávaných informací. Je to nezastavitelný pokrok a můžeme být na něj patřičně hrdí, ovšem nadměrná produkce dat vnáší do celé věci i stinné stránky. Čím více dat máme, tím je složitější s nimi pracovat a také z nich vyvozovat analýzy a závěry.

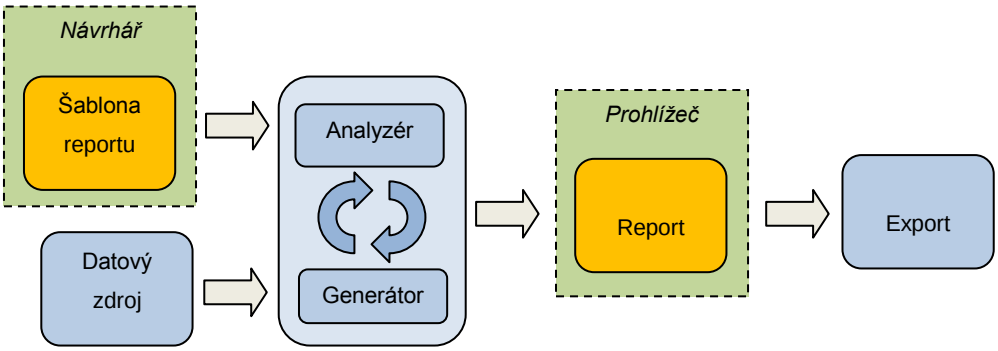
A jak s daty nakládá obyčejný člověk? Před nákupem cizí měny se většina z nás podívá, jaký je kurz a jak vypadá trend vývoje v grafu, zda se třeba nákup valut nevyplatí až těsně před odjezdem. Pokud by nám někdo ukázal tabulku s denními kurzy za poslední 3 měsíce, asi bychom z toho mnoho nevyčetli, ale informace z dat vynesených do grafu je pro nás zcela jasná. Jako další příklad můžeme použít měsíční výpis z účtu. Na tomto výpisu jsme schopni od oka sečíst příchozí částky a odečíst od nich platby, případně si dopočítat úrok. Ale proč? Na konci stránky máme sečtené odchozí platby, příchozí platby včetně dopočteného úroku a ještě je tam uveden rozdíl mezi příchozími a ochozími platbami. A nakonec něco komplexnějšího z prostředí prodeje automobilů. Výrobce má tabulku prodaných vozů za poslední rok s adresami zákazníků a zajímá ho, jak si v jednotlivých krajích vede prodej různých modelů. V uvažovaném množství zakázek už není téměř reálné takovou úlohu analyzovat vlastními silami. A tak použije kontingenční tabulku nebo sestavu.

Z výše uvedeného plyne, že člověk je zcela obklopen daty. O datech mluvím zcela záměrně, protože data jsou definována jen jako znaky, v počítačích dokonce v binární podobě, a teprve jejich uspořádání z nich činí informaci. Ale aby tyto data, případně informace byl schopen lidský mozek vstřebat a zpracovat, potřebuje nástroj, který je přinejmenším vizualizuje či seskupí a zesumuje, nebo dokonce obojí.

I. TEORETICKÁ ČÁST

1 REPORTOVACÍ NÁSTROJE

Reportovacím nástrojem nazýváme aplikaci nebo službu schopnou z velkého množství dat v datovém zdroji vytvořit přehlednou sestavu, která bude mít pro uživatele skutečnou informační hodnotu.



Obr. 1: Obecné schéma generování reportu a reportovacích nástrojů

1.1 Užití v praxi

Kdo potřebuje reportovací nástroje:

- analytici
- uživatelé aplikace

Většině analytiků dat, u kterých můžeme předpokládat znalost práce s databází, vyhovuje prostředí pro dotazy pro přímou práci s daty. Tyto nástroje jsou rychlejší, flexibilnější, a lze říci, že i levnější, protože analytik od svého dotazu očekává určitou odpověď a není tedy potřeba výsledná data popisovat.

Od ostatních uživatelů toto v žádném případě očekávat nemůžeme. Uživatel je zvyklý jen data do specifických aplikací zadávat, většinou ani nemá ponětí, co aplikace se zadanými daty přesně provádí, ale ve výsledku očekává konkrétní výpočty, součty, tabulky a grafy a to v podobě která je obecně zažitá nebo si ji sám definoval u výrobce aplikace.

1.1.1 Data a datová úložiště

Data máme tedy uložena v datovém úložišti, které má formu databáze nebo datového souboru. Databáze je organizované úložiště, kde se příslušné nástroje mohou v datech rychle a efektivně orientovat. S daty v databázi nebo s datovými soubory pracují běžné

aplikace. Aplikace na rozdíl od databázových nástrojů data načítají do objektové formy v paměti a v ní s nimi vykonávají většinu operací. Výsledky operací se buďto zpětně ukládají do původního úložiště nebo se jen prezentují uživateli např. ve formě seznamu.

Typy datových úložišť:

- databáze
- datové soubory / objekty

1.1.2 Data ve formě výsledků

Výstup dat je pro nás důležitý, ať již aktuální v aplikaci nebo specifický pro tisk a další zpracování. Zobrazování dat v aplikaci uživatel většinou neovlivní – to je přesně dané programem. Navíc s programem pracuje uživatel nebo celá firma a prostředí je jim známé. Informace, které chceme poskytnout dalším lidem, už musejí mít běžnější formu, rozšířenou o dodatečné informace a často poznámky či vysvětlení k daným údajům. Nezřídka je takový výstup navíc doplněn logem společnosti. Tedy formát a vzhled výstupních dokumentů (tzv. reportů) už své opodstatnění má a pro uživatele aplikace je tato možnost vítána. Pro tyto účely byly zavedeny šablony dokumentů.

1.1.3 Zpracování dat

Pokud máme datový zdroj (úložiště) a dle šablony víme, jak má vypadat výstupní formát, můžeme přistoupit ke zpracování dat.

Samotné zpracování můžeme popsat několika kroky:

- načtení šablony
- načtení datového zdroje
- analýza vazeb prvků na šabloně
- ověření zda jsou vazby realizovatelné v daném datovém zdroji
- postupné načítání primitivních hodnot a dat potřebných pro vyhodnocení šablony
- vyhodnocení kódu definovaného na šabloně na základě načtených dat
- postupné doplňovaní vypočtených hodnot do šablony
- výpočty na šabloně (počty, sumace, průměry, minima, maxima,…)

Nelze obecně říci, že všechny kroky probíhají v tomto pořadí, dle typu šablon nejsou některé vůbec provedeny, agregační funkce se provádějí paralelně.

1.1.4 Vizualizace dat

V případě, že máme z dat získat nějaké statistické hodnoty nebo data předat dalšímu uživateli volíme nejčastěji náhled a tisk dat v přehledné formě, uzpůsobené pro příjemce informace. Výstupem nemusí být vždy všechna data – často je část dat důvěrná a tak požadujeme jen tisk určitých sloupců nebo hodnot. Výstupy mohou být velice jednoduché - např. stvrzenka o zaplacení, kde je uveden fixní text s názvem a adresou prodejce případně s logem firmy, do dalších polí program doplní částku a jméno s adresou příjemce. Výpis může také obsahovat generovaný výčet různých položek na faktury zákazníka, kde je ve sloupcích uveden i počet jednotlivých kusů toho či onoho zboží a na konci nalezneme sumaci ceny. Mimo to má snad každá faktura rámeček a také plno rámečků kolem důležitých položek. Zda již můžeme přesně rozeznávat určité prvky, ze kterých je takový dokument složen.

1.1.5 Tisk a export dat

Se zobrazením na monitoru se většina uživatelů nespokojí. Přinejmenším je potřeba takový dokument vytisknout, protože dnes je stále nejčastější známkou autorizace dokumentu vlastnoruční podpis, popř. razítko firmy. Pokud není ruční podpis vyžadován, dokument je třeba odeslat emailem. To znamená, že data z aplikace musí být možné vyexportovat do formátu, který je akceptovatelný pro příjemce nebo pro určený cíl. Může to být obrázek nebo PDF dokument pokud trváme na přesném zobrazení u protistrany. Nebo naopak můžeme mít požadavek na dodatečnou editaci informací, to znamená výstup do formátu dokumentu či tabulek kancelářských aplikací (RTF, MS Office, Open Office). Specifickou výjimkou může být export dat do strohého textového formátu, například pro import do další aplikace. Taktéž je dobře využitelný formát HTML například pro sestavy zobrazené na internetu.

1.2 Existující nástroje

Na trhu existuje mnoho nástrojů, které tuto funkčnost poskytují a většina uživatelů a často ani vývojářů bez praxe si důvod jejich existence neuvědomují. O samotném vývoji aplikací

je k dispozici mnoho literatury, kurzů, předmětů ve škole, ale téměř všude se naučí vývojář aplikaci navrhnout a implementovat. Bohužel se ji naučí implementovat do stavu, kdy aplikace data načítá, popř. je zadává uživatel, a zpracovaná data vypisuje nebo ukládá. Zbývá ji otestovat a napsat příručku. A v tom přijde šok - co tisk? Vynecháme-li éru děrných štítků tak dnes žádná aplikace sama data netiskne. V rámci úzkého profilu použití je sice možné implementovat tisk pro konkrétní situaci – i komplexní textový editor má v jádru jednoduchý tisk. Ale pokud se má tisknout více informací z různých míst v informačním systému, stává se takové řešení noční můrou ve formě duplicitního a nepřehledného kódu. Naštěstí vznikly reportovací nástroje, které tisk a mnohem více funkčnosti hravě zvládají.

1.2.1 Zdarma nebo za peníze

Většina informačních systémů není zdarma a potřeba kvalitní reprezentace dat a tisku je pak velice důležitá. Tento stav je tedy i živnou půdou pro další komerční programy, které vám celou prezentaci a tisk obstarají. Samozřejmě najdou se i alternativní volně dostupná řešení.

Zatímco velké firmy zabývající se touto problematikou profesionálně mají ve svém portfoliu nástroje, které snadno zapadnou do prostřední vyvíjené aplikace. Drobná sdružení a dobrovolníci tyto aplikace vyvíjejí do prostředí, které je samo o sobě často volně dostupné, např. multiplatformní C++, JAVA. Nelze tím obecně říci, že volně dostupná řešení jsou špatná, ale napojení tohoto nástroje např. do .NET aplikace je buď nemožné, nebo není úplně přímé. Taktéž není úplně ideální, když vzhled samotné aplikace je graficky a uživatelsky odlišný od reportovacího nástroje.

Ceny komerčních nástrojů sice lákavě začínají třeba na krásných 99 USD. Tyto verze podporují buď jen tu nejzákladnější funkčnost, nebo mají mnohá omezení a jsou spíše koncipovány jako verze k vyzkoušení a jako první krůček k verzím plnohodnotným. Taktéž se ceny často odvíjejí podle počtu licencí pro vývojáře, nebo také, zda možnost editovat šablonu, bude mít i koncový uživatel aplikace. V takovém případě už ale ceny letí do závratných výšek, přesněji do stovek až tisíců USD. Reálně použitelnou verzi pro jednoho vývojáře lze pořídit okolo 500 USD.

Zbývající část práce bude omezena použitím .NET platformy.

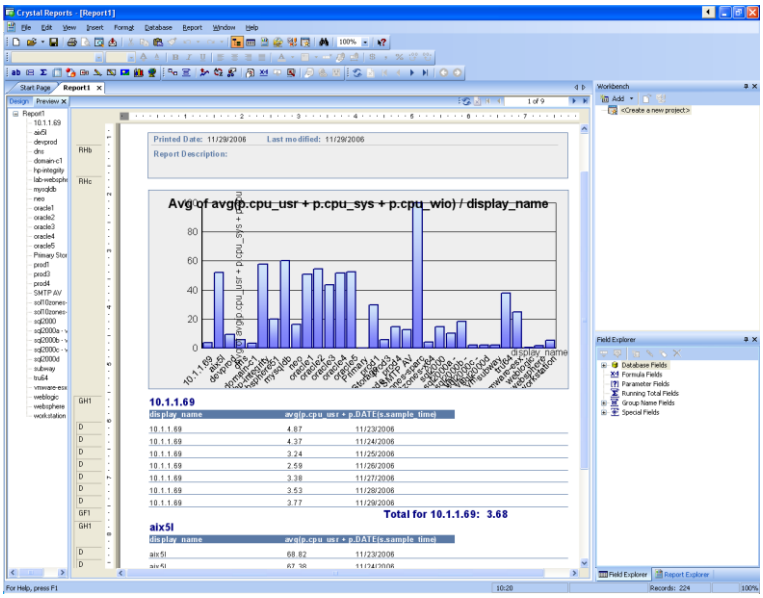
1.2.2 Komerční nástroje pro .NET

Komerčních nástrojů nalezneme několik. Konkrétní produkty jsem hledal v diskusích a mezi kolegy programátory v jiných firmách. Těžko lze říci, který produkt je lepší nebo který umí více, protože způsob prezentace dat v podobných prvcích vede k podobným výsledkům. Zde uživatelé volí spíše výhodnost cen edicí, pro konkrétní potřeby. Uvedené ceny jsou orientační, většinou jde o cenu za jednu licenci.

Pokud si prohlédneme alespoň popis jejich funkcí a zachycené obrazovky v praxi, zjistíme, že jsou si velice podobné. Mírné odlišnosti nalezneme v prvcích při editaci šablony, a tedy mají i jinak uzpůsobené generování sestav. Většinou jsou nabízeny ve formě balíčků pro klientské aplikace WinForms a pro webová řešení ASP.

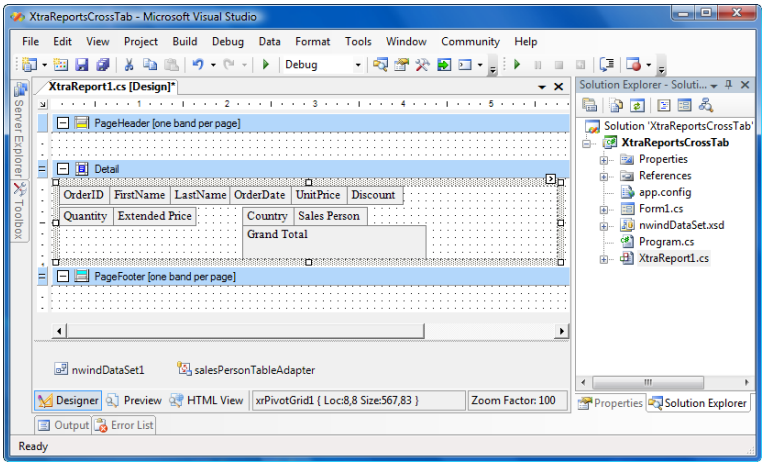
Obecně lze říci, že tyto programy zaznamenaly v posledních letech velice strmý růst v možnostech, možnostech ovládání i kvalitě výstupů.

První z nich, nástroj **Business Objects Crystal Reports**, lze nalézt už při instalaci vývojového prostředí .NET. Nehledejte ho však ve volně šiřitelné verzi Visual Studia Express, ale ani v nejlevnější komerční verzi Visual Studio Standard, která je asi nejsnáze dostupná. Jako součást se distribuuje až od verze Professional, která výrazně dražší. Samotný produkt je k dispozici od cca 550 EUR.



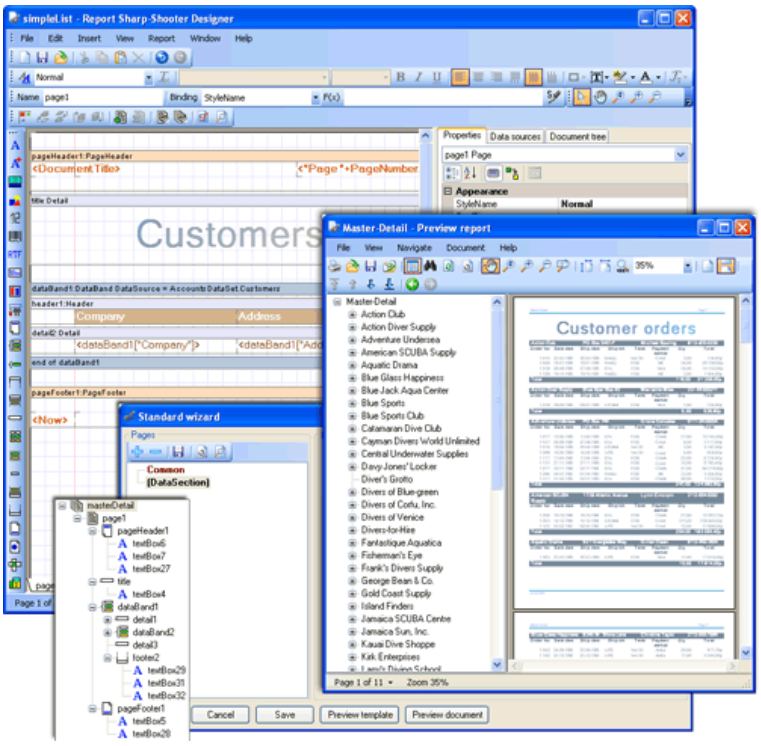
Obr. 2: Prostředí nástroje Business Objects Crystal Reports

Dalším zástupcem je velice uživatelsky příjemný **DevExpress Reporting**, protože tato firma má velké zkušenosti mimo jiné i s vývojem uživatelských prvků pro .NET aplikace. Cena začíná od cca 499 USD.



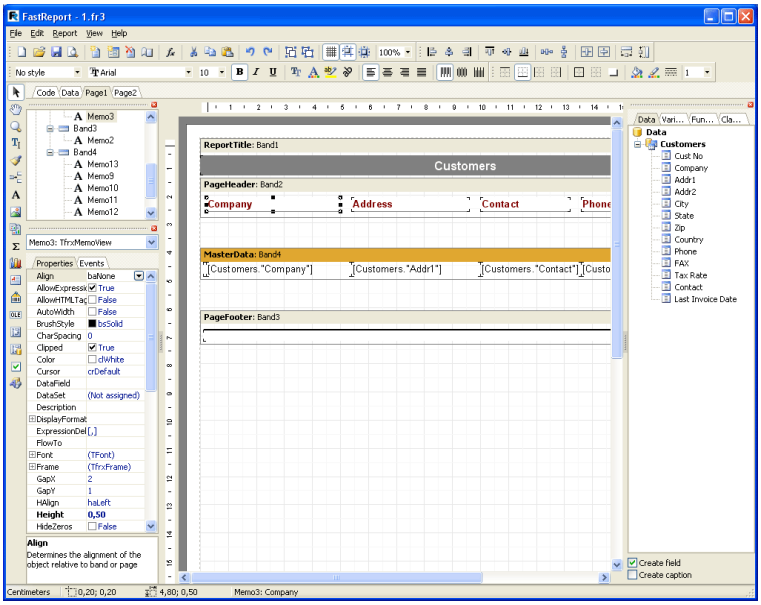
Obr. 3: Prostředí nástroje DevExpress Reporting

S nástrojem **Perpetuumsoft Report SharpShooter** se setkávám denně ve svém zaměstnání. Cena od 333 USD. Existuje také varianta „Express“, která je zdarma, ovšem s mnoha omezeními pro možnosti editace koncovým uživatelem a omezenějších exportů.



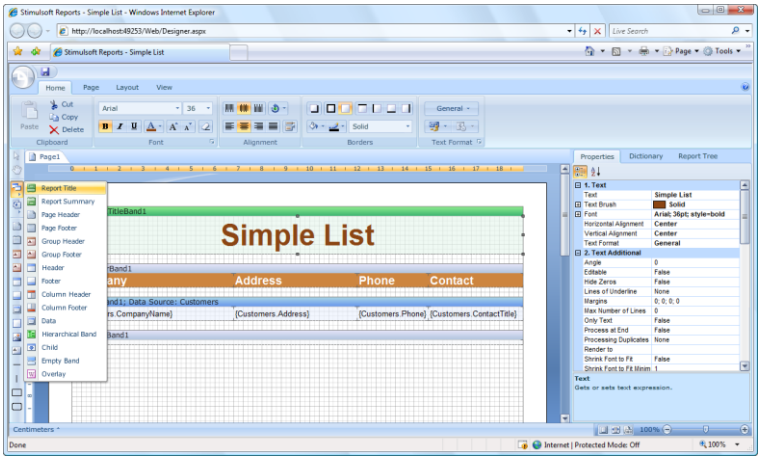
Obr. 4: Prostředí nástroje Perpetuumsoft Report SharpShooter

Mezi jednodušší nástroje můžeme zařadit **Fast Reports Inc. FastReport** Jde o jeden o trošku jednodušší nástroj než předešlé. Cena od 299 USD. I zde je možné pořídit levnější provedení, za 99 USD získáte produkt s možností exportu jen do obrázku.



Obr. 5: Prostředí nástroje Fast Reports Inc. FastReport

Jako poslední komerční nástroj pro .Net zmíním Stimulsoft Reports .Net. Cena tohoto nástroje začíná na 599 USD.

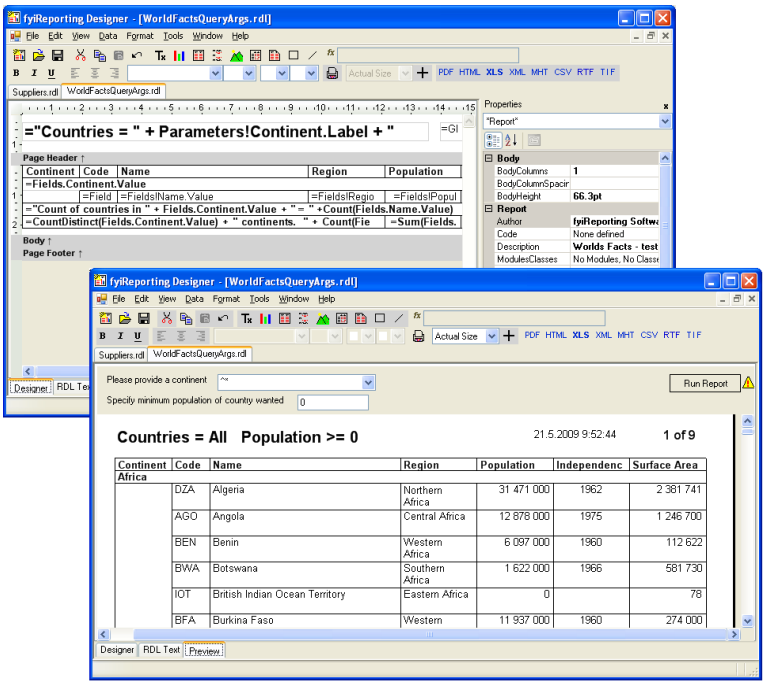


Obr. 6: Prostředí nástroje Stimulsoft Reports .Net

1.2.3 Volně dostupné nástroje pro .NET

Tento odstavec by ještě minulý rok obsahoval asi jedinou větu o neexistenci reálně použitelného nástroje. Dnes již ale existuje docela zajímavý počin – Fyireporting software RDL Project, který je založen na formátu RDL (Report Definition Language) a poskytuje již docela dobré možnosti použití. Tento formát využívají i Crystal Reports.

A nejspíše z tohoto důvodu také v poslední době vznikly i různé „lite“ a „free“ verze komerčních nástrojů. Je ale také nutné poznamenat, že produkt zatím trošku zaostává, protože není vyvíjen tak dlouhou dobu a asi na něm nepracuje velký tým lidí.



Obr. 7: Prostředí nástroje FyiReporting Software RDL Project

Existují také další nástroje, ale jsou to spíše takové nedokončené pokusy vývojářů, kde navíc licence nebo nedostupnost zdrojových kódů brání jejich dodělání a uzpůsobení do vlastní aplikace:

Fabio Zanetta MyNeoReport (nejsou dostupné zdrojové kódy a produkt není prakticky příliš použitelný)

Daniel Zaharia Report Builder (neanalyzuje data, jen zobrazuje tabulky)

Tyto produkty také nejsou nijak nabízeny, ale podařilo se mi je vyhledat.

2 EDITOR ŠABLON, PRVKY, A GENEROVÁNÍ SESTAVY

Každý reportovací nástroj je složen ze dvou téměř nezávisle pracujících aplikací nebo jedné aplikace a služby. Vždy je součástí editor šablon a potom dle použití buďto aplikace pro generování sestav nebo reportovací služba, která sestavy generuje a ukládá.

2.1 Editor šablon

Jde o aplikaci podobnou editoru vektorových obrazců, ve které si uživatel vytváří šablony sestav.

Šablona je datový soubor, obsahující informace o pozicích a vlastnostech specifických prvků. Jednotlivé šablony umí editor načítat a ukládat.

Součástí editoru většinou bývá i editor skriptů, které dynamicky ovlivňují vlastnosti prvků.

2.2 Prvky šablony

Prvky na šabloně graficky reprezentují obdélníky a jejich vlastnosti je možno měnit nejčastěji prostřednictvím komponenty nazývané „PropertyGrid“ (jde o dvousloupcovou tabulku, kde v prvním sloupci jsou názvy vlastností, a v druhém, editovatelném sloupci, uživatel vlastnosti modifikuje). Prvky mají dva druhy vlastností – fixní a dynamické. Fixní vlastnosti mohou být například: Text = „Nadpis“, Barva textu = „Modrá“. Dynamické vlastnosti naopak vycházejí z reálných dat v době generování sestavy, tedy do editoru zadáme například takové vlastnosti Text = „Zdroj.Faktura.Číslo“, Barva textu = „Když Zdroj.Faktura.Suma > 0 tak „Černá” jinak „Červená“. Většina prvků má vlastnost „Datový zdroj“, která definuje propojen do zdrojových dat. Název této vlastnosti mže být různý pro konkrétní prvky.

2.2.1 Dokument a stránka

Dokument a stránku můžeme nazvat abstraktními prvky, protože jsou nedílnou součástí šablony. Mají své vlastnosti, ale ve výsledné sestavě nejsou přímo vidět. Vlastnosti dokumentu jsou orientovány spíše na nastavení editoru. Dokument zastřešuje celou šablonu a je složen z jednotlivých stránek. Stránku nemůžeme chápat doslovně. Stránka slouží pro vizuální představu o výsledném reportu v době editace. Skutečné umístění a velikost prvků umístěných na stránce je dopředu obecně neznámá. Při generování

se prvky plní reálnými daty a prvek se vertikálně roztahuje podle plněných dat. Nezřídka je i prvek, zvláště cyklický, roztáhnutý na několik stránek. Ostatní prvky na šabloně svou pozicí mění dle předchozích prvků. Reálné využití další stránky nalezneme v případě, že následující data budou vždy začínat na nové stránce. Některé reportovače jsou dokonce jednostránkové a používají prvek umožňující zalomení stránky.

2.2.2 Textové pole

Textové pole je nejprimitivnějším datovým prvkem, ale hlavním stavebním kamenem celé sestavy. Text může být zadán fixně. Toto využití nejčastěji nalezneme v nadpisech, v textových poznámkách, v komentářích apod. Nebo dynamicky přes vlastnost „Datový zdroj“, kde definujeme se kterou proměnnou v datovém zdroji je prvek propojen a tedy jaká hodnota se do něj bude vepisovat ve výsledné sestavě.

2.2.3 Cyklický prvek

Ačkoli tento prvek ve výsledné sestavě vůbec neuvidíme tak z celé sestavy odvádí nejvíce práce. Cyklický prvek je kontejnerem pro další součásti cyklického prvku a je jej nutno vždy navázat na datový zdroj. Zadaný datový zdroj se musí odkazovat na typ pole nebo typ kolekce a implementovat patřičné rozhraní pro průchod touto skupinou prvků. Obecně můžeme říci, že zdroj může být i pole řádků nebo sloupců tabulky. Cyklický prvek při generování sestavy inkrementuje vnitřní ukazatel a hodnotu na dané pozici poskytuje prvkům, které jsou v něm vloženy. Vložit lze do něj prvky: záhlaví, tělo a zápatí cyklického prvku

2.2.4 Záhlaví cyklického prvku

Záhlaví lze vložit do cyklického prvku. Je kontejnerem pro další prvky (mimo cyklických). Ačkoli je umístěn v cyklu provede se jen jednou, resp. stále nabízí první hodnotou z datového zdroje. Využití tohoto pole nalezneme v umístění textových polí s nadpisy sloupců.

2.2.5 Tělo cyklického prvku

Tělo cyklického prvku má největší funkční podíl na celé sestavě. Jde opět o kontejner pro další prvky. Při generování sestavy cyklicky prochází všechny prvky v datovém zdroji

a poskytuje je vnořeným prvkům. Tento kontejner může obsahovat ale i další cyklické prvky. V těchto dvou bodech se podstatně liší oproti záhlaví. V praxi se tak datovým zdrojem stává například kolekce faktur. V těle se pak vypisují informace z hlavičky faktury. Do tohoto těla se napojí další cyklický prvek, který jako datový zdroj obsahuje kolekci položek faktur pro právě procházenou fakturu a ve svém těle vypisuje jednotlivé položky.

2.2.6 Zápatí cyklického prvku

Zápatí cyklického prvku je funkčně velmi podobné záhlaví. Jediný rozdíl je v poskytované hodnotě z datového zdroje, která je vždy hodnotou posledního prvku.

2.2.7 Záhlaví stránky

Kontejner pro prvky, který se objevuje v horní části stránky. Lze jej nastavit, zda se má být generován právě pro dané stránky vycházející z návrhu konkrétní strany nebo úplně pro všechny strany.

2.2.8 Zápatí stránky

Má obdobnou funkci jako záhlaví stránky, akorát se zobrazuje na koncích stránek.

2.2.9 Tvary

Tvary jsou primitivní vektorové obrazce, slouží ke grafickému zvýraznění či oddělení.

2.2.10 Obrázek

Obrázek je velice jednoduchý prvek a spíše doplňkový. Ačkoli existuje možnost navázání datového zdroje, nejčastěji se využívá fixní hodnota např. pro logo společnosti. Jediné rozumné využití datového zdroje je možno hledat v zobrazení například grafu, který nám jiná aplikace poskytuje jen ve formě obrázku.

2.2.11 Graf

Funkčnost tohoto prvků je zřejmá. Graf je ovšem specifický na data v datovém zdroji. Data musí typ pole nebo kolekce, a jednotlivé prvky musejí obsahovat proměnné se souřadnicemi. Tedy prvky musejí mít jednotnou strukturu.

2.2.12 Čárové kódy

Velkou roli v tiskových sestavách určených zejména pro evidenci skladových zásob hrají čárové kódy, které umožňují elektronické čtení například čísla faktury, dodacího listu. Prvek čárového kódu potřebuje ke své funkčnosti dvě informace – typ čárového kódu a zobrazované číslo.

2.2.13 Kontingenční tabulka

V reportovacích nástrojích si kontingenční tabulka našla své místo zejména pro přehledné zobrazení statistických dat. Tento prvek se však objevil až v pozdějších verzích většiny nástrojů, protože komplexní funkčnost tohoto prvku se již trošku odklání od původních záměrů těchto nástrojů.

2.2.14 Volný prvek

Volný prvek je v této oblasti celkem novinkou, avšak velice potřebnou. Jde o kódem modifikovatelnou komponentu, která do jisté míry pokrývá individuální potřeby uživatelů. Dosud bylo přidání nového prvku možné jen modifikací zdrojového kódu celého nástroje, který nemusí být možné získat nebo jeho pořízení vyžaduje další investice. Některé úlohy není ani s volným prvkem možné vyřešit a editaci zdrojových kódů se nevyhneme.

2.3 Generování sestavy

Pokud máme k dispozici šablonu a datový zdroj, můžeme přistoupit ke generování sestavy.

Samotné generování je spojené se dvěma hlavními kroky:

- umístění prvku
- vyhodnocení vlastností prvku

V závěru generování je sestava zobrazena uživateli nebo uložena na disk.

2.3.1 Datový zdroj a zdrojová data

Obecně existuje pouze jeden datový zdroj a jednotlivé části sestavy v něm vyhledávají konkrétní zdrojová data. Napojení datového zdroje by měl umožňovat adaptér, který by pro konkrétní formát uložených dat umožňoval jednotný přístup. Existují tedy adaptéry pro

přístup k databázím, datovým souborům, objektům či k rozhraním dalších aplikací. Adaptéry jsou často řešeny jako samostatné knihovny, což do jisté míry značně umožňuje licencování.

2.3.2 Umístění prvku

Při umístění hodnot prvku je přihlíženo ke dvěma kritériím:

- pozice prvku
- hierarchie prvku

Vertikální pozice prvku je při generování relativní vzhledem k jeho souřadnicím v době návrhu. Tato relativita je dána proměnnou výškou skutečně zobrazených hodnot v předchozích prvcích. Vertikální pozici lze tedy určit sumou všech výšek předchozích prvků. Horizontální pozice zůstává zachována.

U prvků, které jsou umístěny v cyklických prvcích, se navíc kalkuluje s hierarchií. Nejprve se vykreslují cyklické prvky, které jsou umístěny přímo na stránce. V jejich rámci se vykreslí záhlaví, tělo a nakonec zápatí. Pokud tělo obsahuje další cyklické prvky, vykreslování se provádí obdobně na nižší úrovni.

2.3.3 Vyhodnocení vlastností

Většina vlastnosti je definována duplicitně jako:

- fixní
- dynamické

Fixní vlastnosti se vyhodnocují okamžitě a jsou známy již v době návrhu. Mezi typicky fixní vlastnosti patří například výběr fontu a dalšího nastavení písma nebo formát masky pro zobrazení čísel. Výsledkem vyhodnocení je tedy přímo zadaná hodnota.

Dynamicky vyhodnocované vlastnosti jsou přímo závislé na datovém zdroji. Nejčastěji se využívá vlastnost, která vrací hodnotu proměnné z datového zdroje. Navíc není vždy zcela vhodné zobrazovat hodnotu tak jak byla uvedena v datovém zdroji. Například booleovské operátory by nám do sestavy vypisovaly „1“ nebo „0“ případně „true“ a „false“. Optimální by tedy bylo zadat výraz, který vstupní hodnotu vyhodnotí a vypíše smysluplnější hodnotu, např. „ano“ či „ne“. Uplatnění si najdou i vlastnosti určující

viditelnost nebo barvu. Viditelnost prvku může zpřehlednit zobrazení částek – ve sloupcích není nutné zobrazovat hodnotu, pokud je rovna nule. Barvou můžeme zvýrazňovat záporná čísla.

2.3.4 Zobrazení, tisk a export dat

Výsledný report je buďto prostřednictvím komponenty zobrazen uživateli nebo v případě, že reportovací nástroj běží jako služba ihned uložen do daného výstupního formátu. Možnost tisku či exportu má i uživatel po náhledu v komponentě.

Formát zobrazení nebo exportu určuje zvolený renderovací aparát.

Pro úlohy zobrazení, tisk a export do obrázku je vždy použito s malými úpravami kreslení do bitmapového formátu, tento se buď zobrazí, odešle na tiskárnu nebo po konverzi uloží do formátu obrázku.

Pro export do formátu kancelářských balíků, HTML, XML a prostého textového formátu se již používají individuální aparáty, distribuované většinou v samostatných doplňkových knihovnách.

II. PRAKTICKÁ ČÁST

3 ANALÝZA

Aplikaci i analýzu můžeme rozdělit do dvou rovin řešení.

Zaprvé bylo třeba navrhnout vhodnou strukturu objektů tvořící šablonu a výsledný dokument. Se šablonou bude pracovat aplikace návrháře, poskytovatelé importu a exportu. Nyní, když podobné aplikace existují, lze mnohem lépe usoudit, co můžeme od struktury šablony očekávat:

- struktura šablony musí být hierarchická,
- prvky budou mít společného předka nebo implementovat stejný interface,
- některé prvky budou hrát roli kontejnerů pro jiné prvky,
- metody kontejnerových prvků budou rekurzivní,
- pro zjednodušení by struktura dokumentu měla vycházet ze struktury šablony.

Návrhář šablon, prohlížeč dokumentů, export do obrázků, tisk mají obdobnou vizuální podobu, tedy měly by mít společný základ.

Druhým úkolem bylo nutné vymyslet způsob, jakým by bylo možné vyhodnotit výrazy zadané uživatelem. V průběhu další části zjistíme, že se nejedná o pouhé vyhodnocení.

Většina specifických tříd začíná písmenem „M“, protože aplikace byla nazvána MReporter. Z tohoto názvu vycházejí přípony souborů MRT pro šablonu a MRD pro dokument.

3.1 Objektový návrh

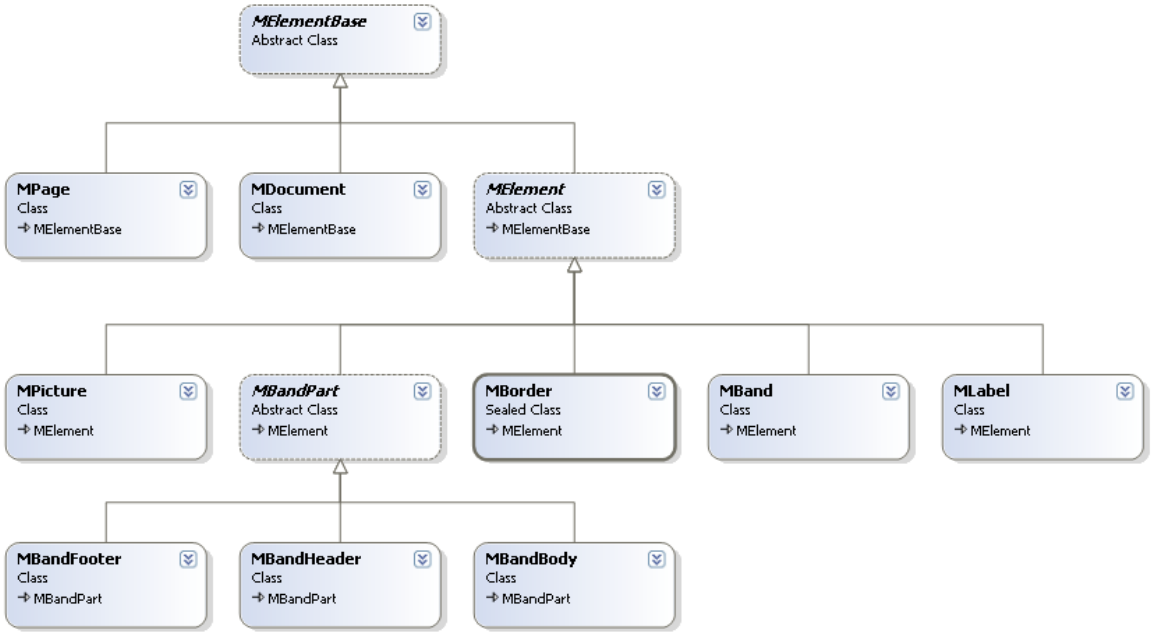
3.1.1 Hierarchická struktura šablony a dokumentu

Prvky mají mnoho společných vlastností:

- jméno (bude v rámci šablony unikátní),
- pozici prvku (pozici relativní vůči kontejneru pokud jej má a absolutní ke stránce),
- velikost prvku,

proto vznikl předek *MElementBase*, který mimo těchto vlastností zajišťuje unikátnost jmen prvků. Unikátní jméno si prvky odvozují od svého typu přidáním číselného postfixu. Z tohoto předka dědí i prvky, se kterými nemůžeme manipulovat. Jedná se o dokument

a stránku. Stránka je první kontejnerový objekt. Ostatní prvky jsou odvozeny od potomka *MElement*. Ten přidává specifickou vlastnost „viditelnost prvku“ a objekt doplňují dopočitatelné informace o ploše, kterou prvek na plátně zabírá. Zvláštním prvkem je cyklický prvek *MBand*, jenž je kontejnerem pro *MBandHeader* (záhlaví), *MBandBody* (těla), a *MBandFooter* (zápatí). Tyto části jsou rovněž kontejnery pro další prvky, a protože mají více společných vlastností, mají i společného předka jménem *MBandPart*. Kontejnerové prvky nejsou samy o sobě mimo návrh viditelné. Skutečné informace nesou prvky typu *MLabel* (popisek), *MBorder* (rámeček), *MPicture* (obrázek).



Obr. 8: Diagram tříd hierarchie elementů

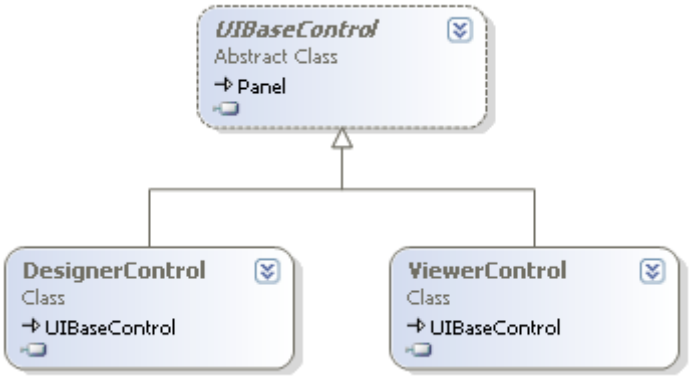
Správnou úvahou byl zcela jistě návrh jedinečné struktury pro šablonu a výsledný dokument. Dokument se od šablony liší pouze absencí kontejnerových prvků a připravený dokument také nepotřebuje k zobrazení uživatelem zadané výrazy, protože ty již byly nahrazeny skutečnou hodnotou během generování.

3.1.2 Komponenty pro návrhář a prohlížeč

Komponentou na uživatelském rozhraní nazýváme prvek se specifickou úlohou. V našem případě bude komponenta představovat vystředěný list papíru (plátno), reagující na pokyny

uživatele. Inspiraci jsem našel v editoru vektorových obrazců od A. Contoli [5]. Pokud si je podobná struktura šablony i dokumentu mohla by vzniknout jediná komponenta i pro vizualizaci. V praxi však návrhář obsahuje mnoho funkcí a prohlížeč je stavěn jen do role pasivního zobrazení. Z toho plyne že, komponenty budou dvě a budou mít pouze společného předka, který bude mít metody použitelné v obou režimech práce.

Základní komponenta obsahuje plátno s okrajem, odchytává základní události vyvolané myší. Potomek pro návrhář doplňuje funkčnost o možnost vkládání, editaci a mazání prvků. Komponenta prohlížeče naopak rozšiřuje základní funkčnost o tisk a export dokumentu.



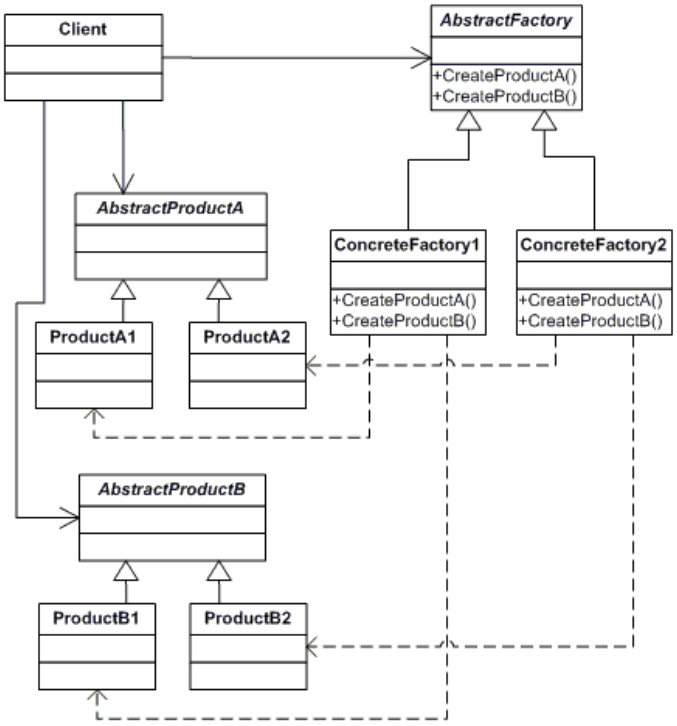
Obr. 9: Diagram tříd uživatelských komponent

3.1.3 Projekce struktury

Slovo projekce bylo zvoleno záměrně, protože dokument je nutné a možné promítnout do několika různých podob:

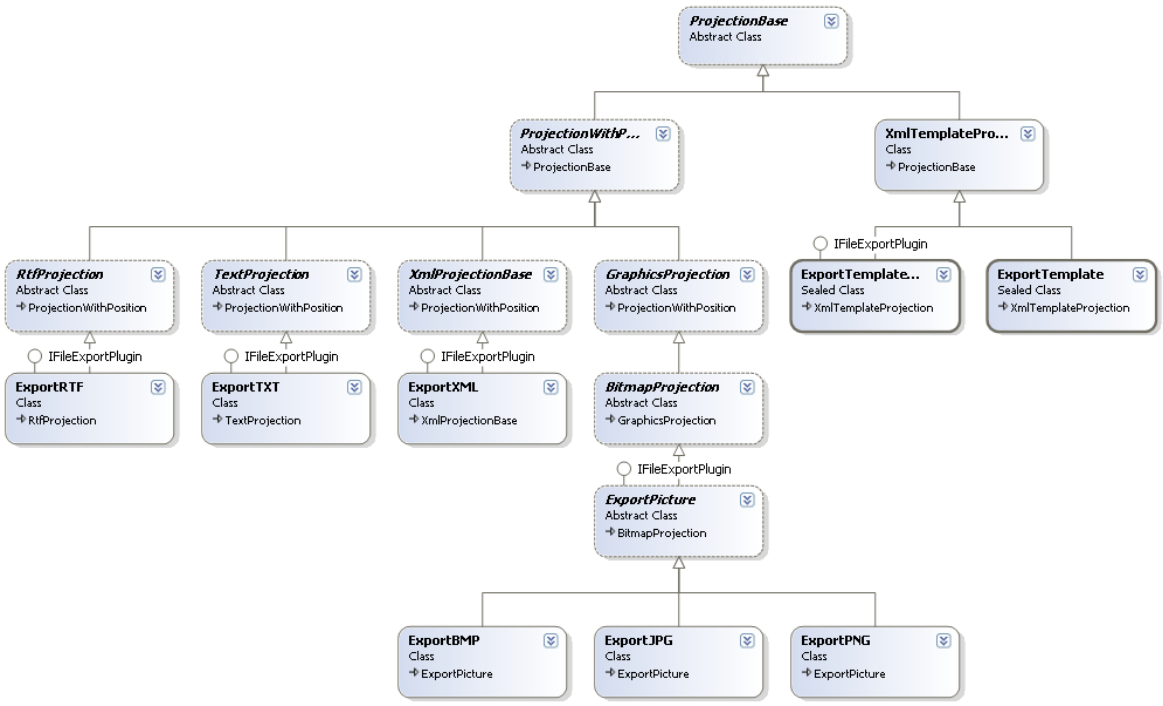
- zobrazení šablony v návrháři,
- zobrazení dokumentu v prohlížeči,
- forma pro náhled tisku a tisk samotný,
- struktura v souborovém formátu pro uložení návrhu,
- struktura v souborovém formátu pro uložení dokumentu,
- export do různých formátů obrázků,
- export do textového souboru,
- export do RTF využitelný dále v kancelářských balících.

Vzhledem k tolika možnostem byl vybrán návrhový vzor GOF „Abstract Factory“ [4], který se přesně na tuto situaci hodí.



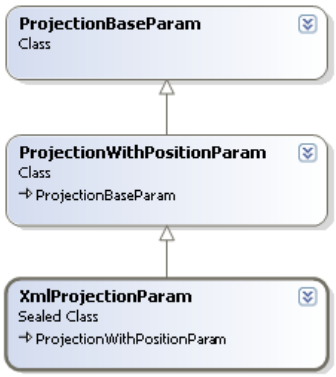
Obr. 10: Diagram tříd obecného návrhového vzoru Abstract Factory

V našem případě hraje roli AbstractFactory předek projekcí třída *ProjectionBase*, ConcreteFactory jsou projekce v hierarchii uvedené níže, produktem jsou výstupy dle konkrétních factory např. XML dokument, obrázek, Textový soubor atd.



Obr. 11: Diagram tříd použitých pro projekci dat

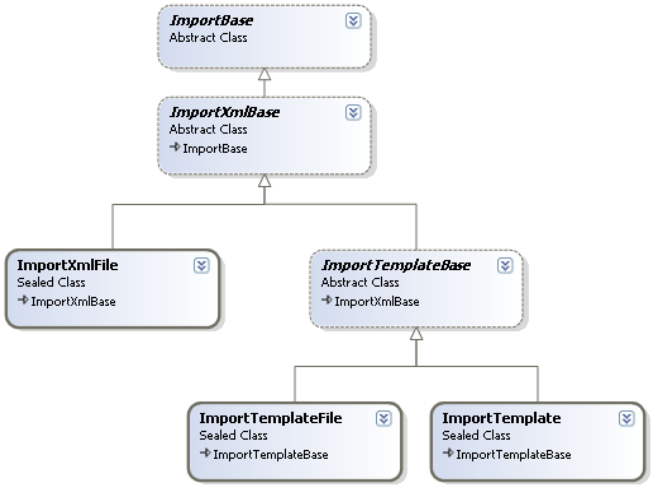
Jednotlivé projekce ještě mají dále definovaný typ parametrické třídy, která udržuje podstatné informace (vertikální posuv, viditelnost, XML uzel, …) v rámci hierarchie v době generování. Vzhledem k menší využitelnosti byly tyto parametry nakonec zredukovány do následující postačující podoby.



Obr. 12: Diagram tříd parametrů pro projekci

3.1.4 Import struktury

Projekcí jsme mohli uložit šablonu nebo dokument, pro načtení použijeme importní funkce. Import je založen na podobném návrhu jako projekce. Importují se zvlášť soubory pro šablonu a pro dokument. Další vnitřní import slouží pro funkce zpět a opakovat v návrháři, případně pro náhled po generování.



Obr. 13: Diagram tříd pro import dat

3.2 Návrhář a prvky

Návrh funkčnosti byl převzat z funkčností stávajících aplikací.

3.3 Řešení problémů generování spojených s vizuálním návrhem

3.3.1 Vizuální kolize prvků

V době návrhu šablony uživatel vkládá prvky na plátno a následně s nimi pohybuje tak, aby docílil požadovaný vzhled šablony. Omylem se však může dopustit chyby a některé prvky by se mohly překrývat, což by vedlo k následné nečitelnosti textu. Jindy však může být toto překrytí žádané. Test kolizí lze spustit přes menu.

V návrháři proto existuje funkce pro kolizní test dokumentu, které může uživatel využít, aby předešel zmíněné komplikaci.

Kolize se zjišťují na základě průniků ploch, které prvky zabírají.

3.3.2 Kolize pořadí prvků při generování

Podstatně vážnější problém s umístěním prvků nastává v době generování. V praxi velice snadno vznikne situace, kdy vizuální pozice prvku vloženého dříve je relativně níž než následně vkládaný prvek. Pořadí prvků v kolekci nadřízeného objektu tedy neodpovídá pořadí při vykreslování. Tento problém je třeba vyřešit před spuštěním generování, protože při vykreslování prvků je třeba udržovat informaci o rostoucím cyklickém prvku, který může svým plněním vše posunout níže a dokonce založit následující stránku. Případný následný návrat o stránku i více zpět, vykreslení prvku s vyšší pozicí, a posun na relativní konec dokumentu, je natolik komplikovaný, že nad ním nemůžeme dále uvažovat.

Tento test s výhodou využívá hierarchického modelu dokumentu, protože je třeba třídit dle vertikální pozice především prvky ve stejných úrovních vlastnících prvků.

Samotné třídění zajišťuje metoda *Sort* objektu *ElementList*. Tato metoda implementuje QuickSort třídění dle *VerticalComparator*. Tento komparátor srovnává vertikální pozici prvku.

3.4 Vyhodnocování výrazů v dynamickém assembly

V reportech je často potřeba některé věci měnit podle aktuálního stavu vypisovaného objektu. Nebo vypisované hodnoty ještě upravovat. Například z důvodu přehlednosti nechceme zobrazovat hodnoty, které jsou rovny nule. Nebo je třeba vizuálně odlišit záporná čísla červenou barvou. Nebo chceme na šabloně sečíst vypisované hodnoty. Zvláštním požadavkem plynoucím z reportu je závislost určité vlastnosti prvku na hodnotě jeho vlastníka. A k tomu všemu potřebujeme vyhodnocovat výrazy, které uživatel definuje.

Uživatel tyto výrazy definuje textovým zadáním. Pro jednoduchost a praktičnost v jazyce v jakém programuje, což je v našem případě C#, případně Visual Basic Net.

Prostředky, které má dnes programátor .Net k dispozici, však přímé vyhodnocení kódu v textu neumožňují. A tak se nabízí buďto námi definovaná jednoduchá syntaxe, kterou budeme následně ručně zkoumat. Toto řešení má velice omezené možnosti použití. Bylo by také možné napsat si na toto vyhodnocení vlastní aparát založený na specifické gramatice, ovšem toto řešení je už implementačně samo o sobě náročnější a opět by nepokrylo všechny možnosti původního programovacího jazyka. Pokud tedy nepůjdeme úplně přímou cestou, dojdeme k závěru, že vyhodnocení lze provést také díky JIT (Just-In-Time)

kompilaci, kterou .Net framework umožňuje. Přesněji můžeme požadovaný kód k vyhodnocení zakomponovat do textu nově tvořeného kódu, ve kterém jej obalíme definicí metody s návratovou hodnotou a definicí třídy a jmenného prostoru. Z tohoto textu se pokusíme vytvořit dynamické sestavení v paměti a nechat si výslednou hodnotu vrátit. Pro toto sestavení budu používat anglický termín assembly.

Ještě před započetím vlastního generování assembly, je třeba vědět, jaké výrazy budeme potřebovat vyhodnotit. K tomuto musíme celý dokument, resp. prvek po prvku projít a všechny vyhodnotitelné hodnoty, si zapamatovat.

3.4.1 Analýza prvku s definovaným výrazem leží přímo na stránce

Tato varianta je nejjednodušší, protože bude stačit pomocí reflexe najít danou vlastnost objektu, načíst její hodnotu a tu vrátit.

3.4.2 Analýza prvku s definovaným výrazem na části cyklického prvku

Oproti předchozímu zadání zde vznikly dvě zásadní komplikace:

- jde o kontejnerový prvek, který lze navíc rekurzivně vkládat, tj. hledaná vlastnost objektu není přímo dostupná bez předchozí analýzy nadřízených prvků,
- jde o cyklický prvek, tedy uživatel na šabloně definuje jedno pole s jedním výrazem, ale ve skutečnosti je těchto polí právě tolik jako je prvků v kolekci.

Už z principu je jasné, že se v drtivé většině případů setkáme právě s kombinací obou předchozích. Například struktura šablony zobrazující výčet zákazníků, jejich faktur a položek faktur by mohla vypadat takto:

```
[Zakaznik(Jmeno)]
    [Zakaznik::Faktura(Cislo)]
        [Zakaznik::Faktura::Polozka(Popis)] [Zakaznik::Faktura::Polozka(Cena)]
```

Uživatel na třech řádcích velice snadno definuje požadavky pro zobrazení. Reálný výstup však bude obsahovat desítky zákazníků, každý zákazník má určitý počet faktur, a každá faktura má určitý počet položek.

Tedy první zmíněnou komplikací je hierarchie Zakaznik -> Faktura -> Polozka faktury, druhou komplikací je fakt, že se jedná o seznam zákazníků, se seznamem faktur a se seznamem položek faktury.

3.4.3 Agregační funkce

Agregací je myšlena funkce, která je aplikována na určitou skupinu položek. V praxi jde nejčastěji o součtování, minimum, maximum a průměr. Většinou je třeba součtovat pouze prvky, které mají společnou vlastnost – například součet cen položek faktury pro konkrétní fakturu. I zde situaci komplikuje rozpor jednoduchosti zadání uživatele a reálné organizace dat.

3.4.4 Princip použití dynamického assembly

Dynamické assembly, jak již bylo zmíněno je sestavený kód z textového zadání, které může vzniknout i postupným generováním. Textové zadání samozřejmě musí splňovat všechny náležitosti a gramatickou správnost kódu, aby mohlo být úspěšně zkompilováno.

Příkladem takového textu kódu může být například:

```
 1  namespace TestAssembly
 2  {
 3     public sealed class TestClass
 4     {
 5        public int GetSum()
 6        {
 7           return [A] + [B];
 8        }
 9     }
10  }
```

Pokud před sestavením nahradíme v textu úseky „[A]“ a „[B]” konkrétními čísly, pak po sestavení a zavolání funkce, resp. metody *GetSum()* obdržíme součet těchto hodnot. Pro případy, které uvažujeme, bychom mohli, takových celých metod do textu vložit právě tolik kolik potřebujeme vyhodnotit výrazů. Toto řešení by zřejmě bylo funkční, ovšem zbytečně by rostla režie spojená se vznikem tolika metod a taktéž by asi neřešila vnitřní závislosti výrazů. Příkladem takové závislosti může být například doplnění zkratky měny za cenu u každé položky faktury, ovšem měna je uložena na vlastníkovi položky, tj. objektu faktury.

3.4.5 Předpříprava výrazů pro dynamické assembly

Před samotným sestavením tedy musíme referenční tabulku všech výrazů, které se můžou provázat, aby během vyhodnocení už byly vazby zřejmé a stačilo všechny výrazy vyhodnotit v jednom kroku. Tento problém je následkem hierarchie objektů. Druhou dříve zmíněnou komplikací je „cyklické bobtnání“ jednoho prvku dle určitého počtu prvků

v navázaném seznamu objektů. Vzhledem k tomu, že bude běžně docházet k současnému řešení obou problémů, je proto nutné toto řešit komplexně.

Cílem této fáze přípravy, o níž se stará instance třídy *ReportPrecompiler*, bude vytvoření hashovací tabulky. Přesněji půjde o slovník, kde klíčem bude složenina jednoznačné identifikace prvku dle hierarchie definované na šabloně a označení výrazu. Pro cyklický prvek bude klíč navíc obsahovat pořadové indexy, které pokryjí celý seznam prvků v kolekci navázané na cyklický prvek.

Příklad záznamu ve slovníku pro přípravu:

```
[<Popisek3|TextValue>,<”Dnes je ” + DateTime.Today.ToShortDateString()>]
[<BandZakaznik(4).BandFaktura(3)|VisibleVal>,<”(int)BandFaktura[“Year”]> 2008”>]
```

V rámci této přípravy je tedy nutné pomocí reflexe rekurzivně projít všechny dostupné kombinace průchodů kolekcemi, abychom zajistili úplné pokrytí hodnot, na které se můžeme v době generování dotázat. Toto zajišťuje metoda *ReportPrecompiler .Run(...)*, resp. pro rekurzi metoda *ReportPrecompiler .Generate(...)*.

Samotný uživatelský zápis pro získávání hodnoty z cyklického prvku např. BandFaktura[“Year”] je pomocí regulárních výrazů převeden na zápis volání metody *ReportCompilerBase.GetBandValue(...)*. Tento zápis, stejně jako ostatní výrazy ukládám do příslušných hodnot ve slovníku.

Vstupní i výstupní hodnoty v době kompilace šablony se přenášejí pomocí struktury *ReportCompilerData*.

3.4.6 Kompilace dynamického assembly a vyhodnocení výrazů

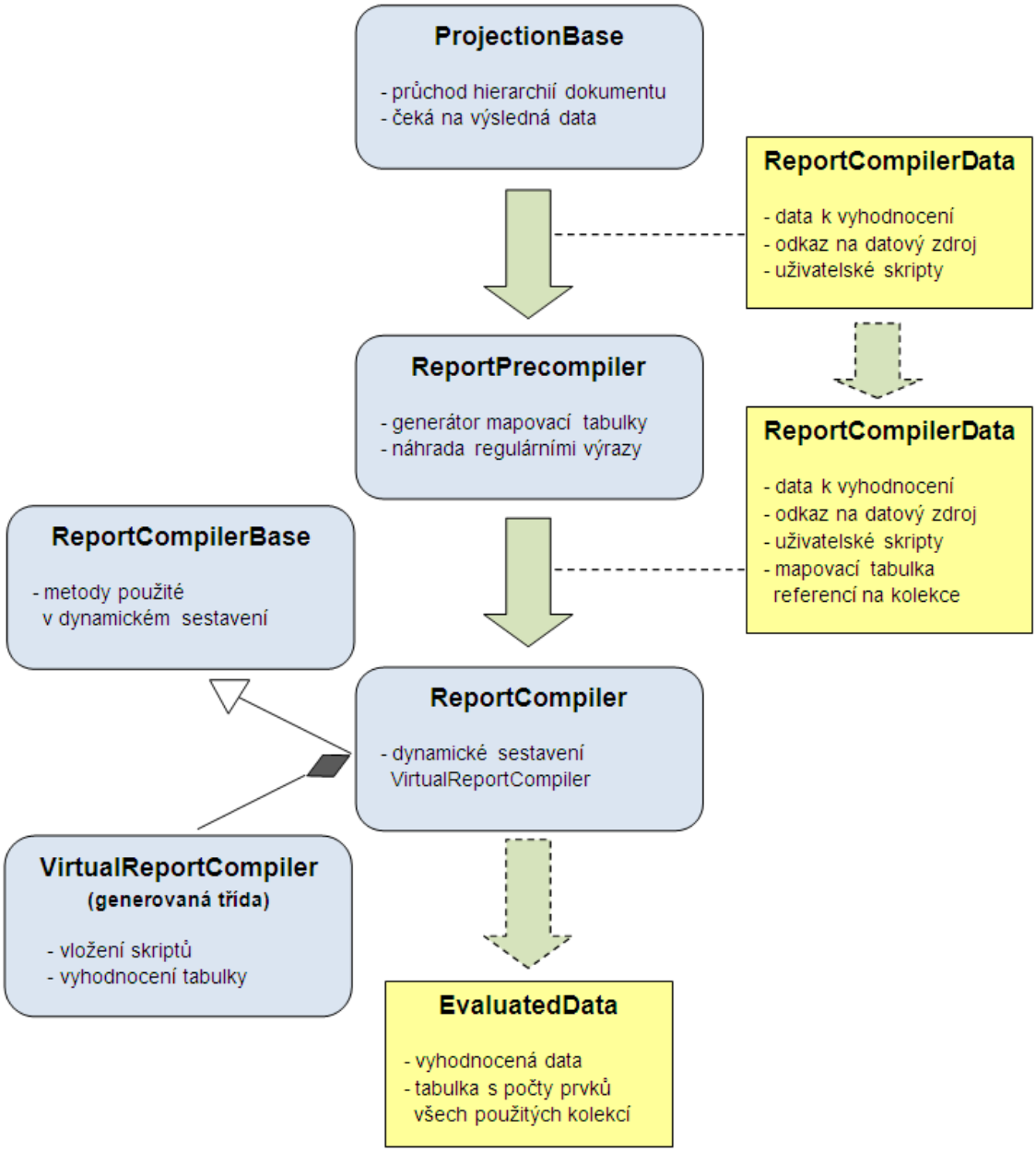
Samotné sestavení provede vyhodnocení hodnot u příslušných klíčů ve slovníku. Stejně tak se zkompiluje uživatelsky definovaný kód, který vložen do těla třídy a kód vložený do těla metody *RunStartupScript()*.

Příklad téměř reálného kódu pro sestavení:

```
 1 using System;
 2 using MReporterCore;

 3 namespace MReporterCompiler
 4 {
 5   public sealed class VirtualReportCompiler : ReportCompilerBase
 6   {
 7      … <kod definovany uzivatelem> …

 8      public void RunStartupScript()
 9      {
10         … <kod, ktery si uzivatel zvolil pri startu generovani> …
11      }

12      public override EvaluatedData PrepareData(
13        ReportCompilerData compilerData, EvaluatedData evaluatedData)
14      {
15         this.CompilerData = compilerData;

16         RunStartupScript();

17          evaluatedData.ValueDictionary.Add(
             "BandInvoice(1).BandInvoiceItem(1).Popisek2|TextValue",
             GetBandValue("BandInvoiceItem(1)", " BandInvoiceItem ", "Name"));

18          evaluatedData.ValueDictionary.Add(
             "BandInvoice(1).BandInvoiceItem(2). Popisek5|VisibleValue",
             (int)GetBandValue("BandInvoiceItem(2)","BandInvoiceItem","Price")>5);

19         return evaluatedData;
20      }
21   }
22 }
```

V textu kódu je možné si všimnout volání metody *GetBandValue(...)*, která tu není implementována. Ovšem tato metoda se nachází na předkovi této sestavované třídy. Toto řešení bylo zvoleno z důvodu minimalizace psaného kódu, který se bude zpracovávat a také pro lepší práci v době vývoje.

V tuto chvíli bych rád upozornil na možnou nestabilitu aplikace, kterou nelze zaručit, jelikož kód přímo ovlivňuje uživatel a tiché zachycení této výjimky rozhodně nepomůže při hledání problému. Samozřejmě, že se dá mnoho věci zkontrolovat a průběžně ověřovat, ale není to možné realizovat v rozsahu této práce. Výjimky jsou zobrazovány i v komerčních nástrojích. Většinu výjimek, ke kterým mohlo dojít, jsem se snažil vhodně ošetřit, dohledat jejich příčinu a problém v dialogovém okně vhodně prezentovat uživateli. Bohužel v dynamickém assembly jsou závažné výjimky zabaleny do vyjímky frameworku a ta většinou nepodává informace o konkrétní vnitřní výjimce. Ošetření tohoto případu by znamenalo vytvořit robustnější aparát, který by kód vnitřně hlídal.



Obr. 14: Schématické znázornění průběhu zpracování a vyhodnocení dat

3.4.7 Párování prvků s vyhodnocenými výrazy

Pokud úspěšně proběhne předpříprava dat a sestavení assembly, zbývá poslední krok a to vyrenderovat výsledný dokument. Renderování opět probíhá v projekčních třídách, ovšem tentokráte s příznakem že jsou hodnoty vyhodnocené a prvky se skutečně zobrazují, pokud mají být viditelné. Samotné párování probíhá podle klíče, který je opět vypočten a dohledán ve slovníku.

U cyklických prvků je navíc potřeba dle prototypu prvku v šabloně vygenerovat tolik prvků, kolik jich má navázaný seznam datového objektu. Reflexe objektů a mapování kolekcí. Toto nese mnoho problémů protože, prvky je třeba vertikálně posunovat, včas zakládat nové stránky atd. I tato problematika vyžaduje hlubší analýzu.

3.5 Projekce výsledků a export dat

Samotná projekce vychází z konkrétních požadavků, pro specifické cílové místo.

3.5.1 Projekce výsledků do XML

Tato projekce má základní význam, jelikož dle struktury dokumentu je prohlížeč opět schopen sestavit dokument, který můžeme prohlížet, tisknout nebo také exportovat do jiného formátu jinou projekcí.

3.5.2 Projekce výsledků do grafického kontextu zařízení

Vlastnost Graphics kontextu zařízení obsahuje referenci na konkrétní zařízení, kterým může být obrazovka, tiskárna apod. Proto této projekce využívám pro prezentaci na obrazovce v době návrhu, i při prohlížení dokumentu. Stejně tak je využita pro náhled před tiskem a tisk samotný.

Drobnou úpravou lze Graphics převádět na bitmapový obrázek, tedy i projekce obrázků je v jádru tímto realizována.

3.5.3 Projekce výsledků do textového souboru

Textový soubor slouží prakticky jen pro export textových popisků, o pozicování se tu také moc snažit nemusíme, protože prostý text nemá žádnou jednotku velikosti písma ani omezující velikosti stránky.

3.5.4 Projekce výsledků do formátu RTF

Pro tuto projekci jsem využil rozšířené komponenty RichTextBox od Khendyse Gordona [8]. Nalezneme tu velkou podobnost s textovým souborem. Navíc můžeme aplikovat font písma a vkládat obrázky.

3.6 Lokalizace

Uživatelská aplikace, u které lze předpokládat, že bude užívána ve více zemích, by měla být lokalizována.

3.6.1 Lokalizace aplikace

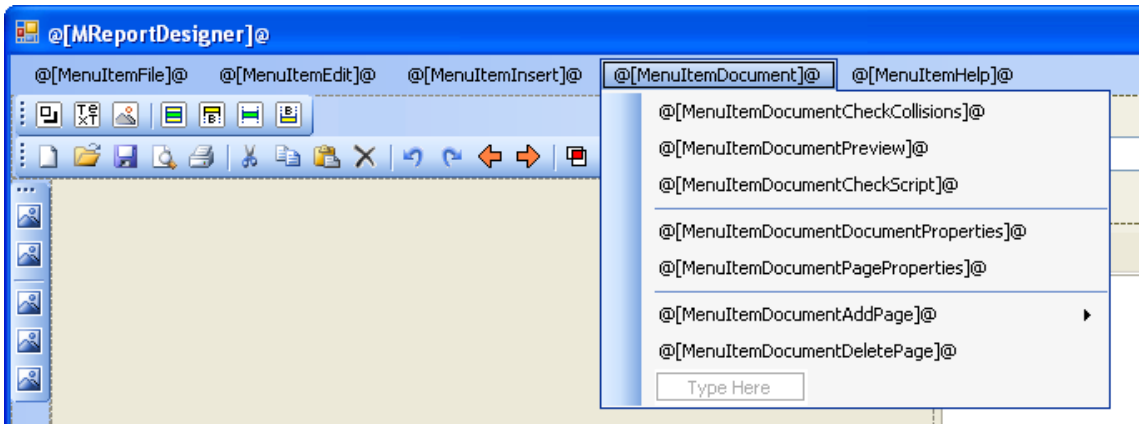
Aplikace je plně lokalizovatelná (mimo komponenty PropertyGrid použité v návrháři). Lokalizace je provedena těsně před načtením formuláře, kdy se přeložitelné názvy zamění za příslušné překlady. Formát @[TextToTranslate]@ je přeložen prostřednictvím třídy *Translator* na skutečný „Text k přeložení“. Slovník překladů se edituje prostřednictvím přiložené aplikace MReporterLang. Překlady aplikace musejí být v souboru MReporter.lng.

Výchozím jazykem je čeština. Toto nastavení lze změnit v souboru app.config.

Obsah souboru app.config:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
        <add key="Language" value="CZ" />
    </appSettings>
</configuration>
```

Přípustné parametry jsou v tuto chvíli: CZ, EN, DE (DE slovíčka však nejsou doplněna).



Obr. 15: Ukázka nepřeložené aplikace v době návrhu

Do jisté míry jsou lokalizovány i ovlivnitelné části případných chybových hlášek a výjimek.

3.6.2 Lokalizace reportů

Stejně jako aplikaci lze překládat i samotné reporty. Texty k přeložení opět stačí vložit mezi @[…]@ a u reportu ponechat soubor stejného jména s příponou *.lng, který bude příslušné překlady obsahovat. Není-li tento soubor dostupný nebo nejsou-li v něm potřebné překlady definovány, zobrazí se text včetně @[…]@. Samotný proces lokalizace se provádí vždy až při závěrečném generování reportu.

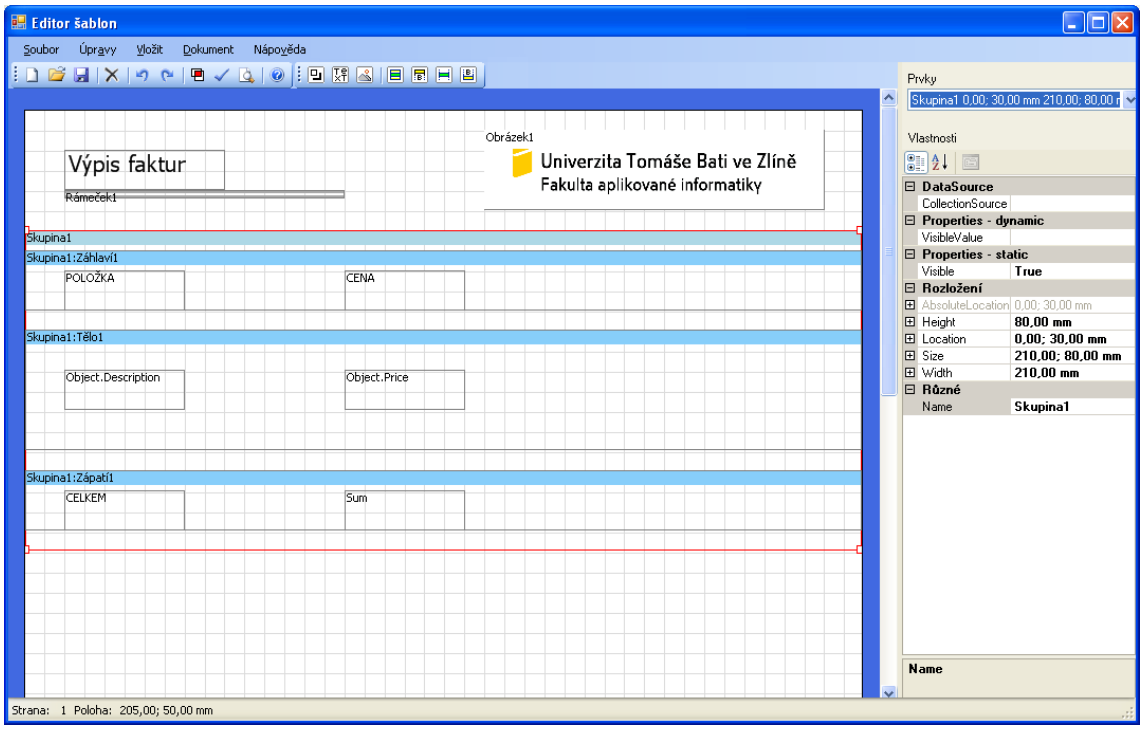
4 APLIKACE

Výsledná aplikace je rozdělena na 3 samostatné celky:

- návrhář šablon,
- prohlížeč dokumentů
- doplňková aplikace pro editaci jazykový překladů aplikace a reportů.

4.1 Popis použití návrháře šablon

Návrhář šablon je nástroj, ve kterém uživatel definuje umístění prvků a jejich vlastnosti. Výsledné šablony vytváří buďto autor aplikace nebo zkušenější a proškolený uživatel. Zaškolení je nezbytně nutné, protože se pracuje s objekty aplikace. Při nesprávné manipulaci by mohlo dojít k pádu aplikace (k tomuto může v těchto aplikacích dojít i tak), ale uživatel by mohl narušit i reálná data.



Obr. 16: Ukázka vlastního nástroje pro návrh šablon

4.1.1 Práce se soubory

Založení nové šablony

Založí novou prázdnou šablonu.

Postup:

- Výběr akce  (Menu: Soubor => Nový)

Otevření existující šablony

Otevře existující šablonu uloženou v souboru.

Postup:

- Výběr akce  (Menu: Soubor => Otevřít)

Uložení editované šablony

Uloží editovanou šablonu do souboru.

Postup:

- Výběr akce  (Menu: Soubor => Uložit)

V menu je dostupná i funkce Uložit jako.

4.1.2 Práce v editoru

Krok zpět

Vrací zpět poslední provedenou operaci.

Postup:

- Výběr akce  (Menu: Úpravy => Zpět)

Opakovat krok

Zopakuje dříve vrácenou provedenou operaci. Opakovat krok nelze použít, pokud již byla provedena jiná editační akce.

Postup:

- Výběr akce  (Menu: Úpravy => Opakovat)

Vymazat

Smaže označené prvky.

Postup:

- Výběr akce  (Menu: Úpravy => Vymazat)

4.1.3 Práce s prvky

Textové pole - Popisek

Textové pole je nosným prvkem informace, lze jej tedy umístit na stránku nebo do záhlaví, těla, zápatí cyklického prvku.

Postup:

- Výběr akce  (Menu: Vložit => Popisek)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku
  - Text / hodnotu textu
  - Barvu písma, velikost písma, font písma
  - Viditelnost prvku, pozici a velikost prvku (lze měnit i myší)

Vložení rámečku

Rámeček slouží k vizuálnímu oddělení informací či prvků, lze jej umístit na stránku nebo do záhlaví, těla, zápatí cyklického prvku.

Postup:

- Výběr akce  (Menu: Vložit => Rámeček)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku
  - Tloušťka čáry pro jednotlivé strany, barva a styl rámečku
  - Viditelnost prvku, pozici a velikost prvku (lze měnit i myší)

Vložení obrázku

Prostřednictvím tohoto prvku, je možné definovat prostor, ve kterém se zobrazí obrázek. Do šablony se vkládá pouze relativní cesta k obrázku.

Postup:

- Výběr akce  (Menu: Vložit => Obrázek)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku
  - Cestu k obrázku
  - Viditelnost prvku, pozici a velikost prvku (lze měnit i myší)

Vložení cyklického prvku - Skupina

Cyklický prvek není ve výsledku viditelný, slouží pouze zde v návrháři jako kontejner pro další prvky, které budou zobrazeny. Prvek automaticky přizpůsobuje svou šířku šířce stránky.

Postup:

- Výběr akce  (Menu: Vložit => Skupina)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku
  - Cestu k datovému zdroji kolekce
  - Viditelnost prvku, vertikální pozici a velikost prvku (lze měnit i myší)

Vložení záhlaví cyklického prvku – Záhlaví skupiny

Záhlaví cyklického prvku není ve výsledku viditelné, slouží pouze zde v návrháři jako kontejner pro další prvky, které budou zobrazeny. Prvek automaticky přizpůsobuje svou šířku šířce stránky.

Postup:

- Výběr akce  (Menu: Vložit => Záhlaví skupiny)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku
  - Viditelnost prvku, vertikální pozici a velikost prvku (lze měnit i myší)

Vložení těla cyklického prvku – Tělo skupiny

Tělo cyklického prvku není ve výsledku viditelné, slouží pouze zde v návrháři jako kontejner pro další prvky, které budou zobrazeny. Prvek automaticky přizpůsobuje svou šířku šířce stránky.

Postup:

- Výběr akce  (Menu: Vložit => Tělo skupiny)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku

Viditelnost prvku, vertikální pozici a velikost prvku (lze měnit i myší)

Vložení zápatí cyklického prvku – Zápatí skupiny

Zápatí cyklického prvku není ve výsledku viditelné, slouží pouze zde v návrháři jako kontejner pro další prvky, které budou zobrazeny. Prvek automaticky přizpůsobuje svou šířku šířce stránky.

Postup:

- Výběr akce  (Menu: Vložit => Zápatí skupiny)
- Kliknutí na plátno šablony (nebo také kliknutím a potažením myší) umístíme prvek
- V editoru vlastností můžeme změnit
  - Jméno prvku

Viditelnost prvku, vertikální pozici a velikost prvku (lze měnit i myší)

4.1.4 Práce s dokumentem šablony

Detekce kolizí

Spustí detekci kolizí objektů (kontrola překrývání). Výsledek je zobrazen prostřednictvím okna se zprávou, které prvky spolu kolidují. Kolizní chyba není závažná, může však způsobit nechtěný posun prvků nebo překrytí textů ve výsledném generování.

Postup:

Výběr akce  (Menu: Dokument => Detekce kolizí)

Ověření skriptu

Spustí ověření zadaného skriptu Výsledek je zobrazen prostřednictvím okna se zprávou. Chyba skriptu je závažná a bez její nápravy nejspíše nebude dokument vygenerován. Ani úspěšné prověření skriptu však nemůže zaručit úspěšné vygenerování reportu.

Postup:

- Výběr akce  (Menu: Dokument => Ověřit skript)

Náhled

Zobrazí náhled na hrubě vygenerovaný report (bez překladů). Pokud je na šabloně odkaz do datového objektu a návrhář není spouštěn z konkrétní aplikace, tj. nemá datový objekt přiřazen může dojít k chybě.

- Výběr akce  (Menu: Dokument => Náhled)

4.1.5 Práce s nápovědou

O programu

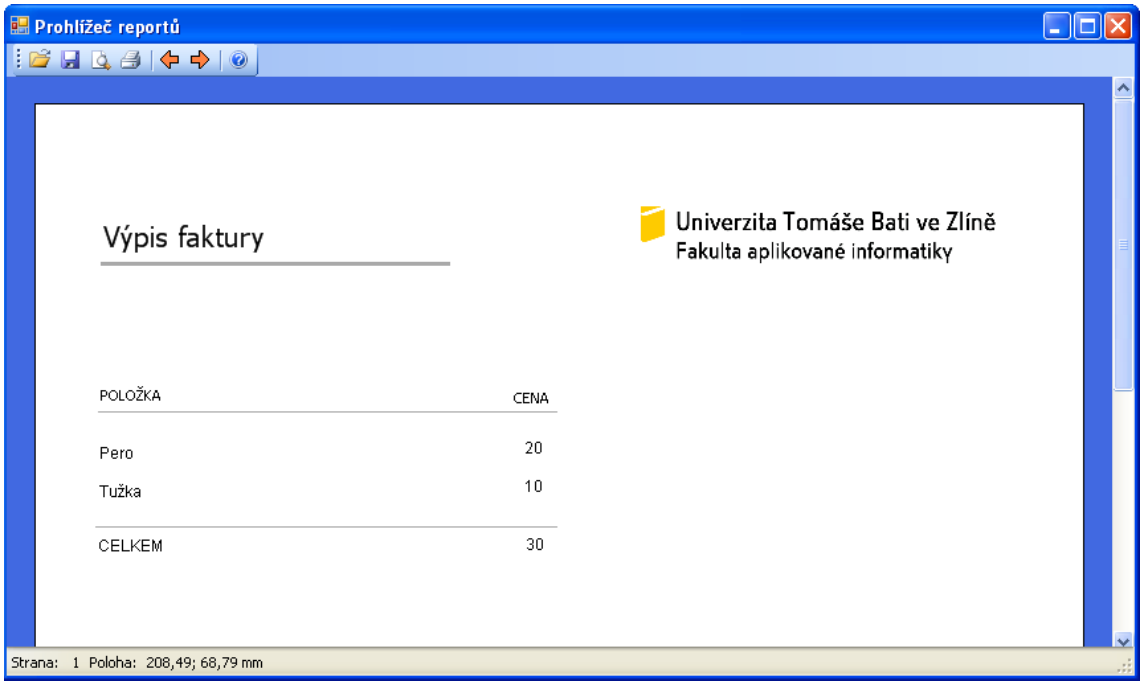
Zobrazí informaci o programu a autorovi.

Postup:

- Výběr akce (Menu: Nápověda => O programu)

4.2 Popis použití prohlížeče

Prohlížeč slouží už pouze k otevírání vygenerovaných reportů v XML, a je z něj možné tyto reporty dále tisknout nebo exportovat do různých jiných formátů.



Obr. 17: Ukázka vlastního nástroje pro prohlížení dokumentů

4.2.1 Práce s nabídkou panelu

Otevření vygenerovaného reportu

Otevře vygenerovaný report uložený v souboru.

Postup:

- Výběr akce 

Náhled před tiskem

Zobrazí náhled stránky, ze kterého je také možný tisk.

Postup:

- Výběr akce 

Tisk reportu

Zobrazí tiskový dialog pro tisk.

Postup:

- Výběr akce 

Předchozí strana

Zobrazí předchozí stranu v reportu.

Postup:

- Výběr akce 

Následující strana

Zobrazí následující stranu v reportu.

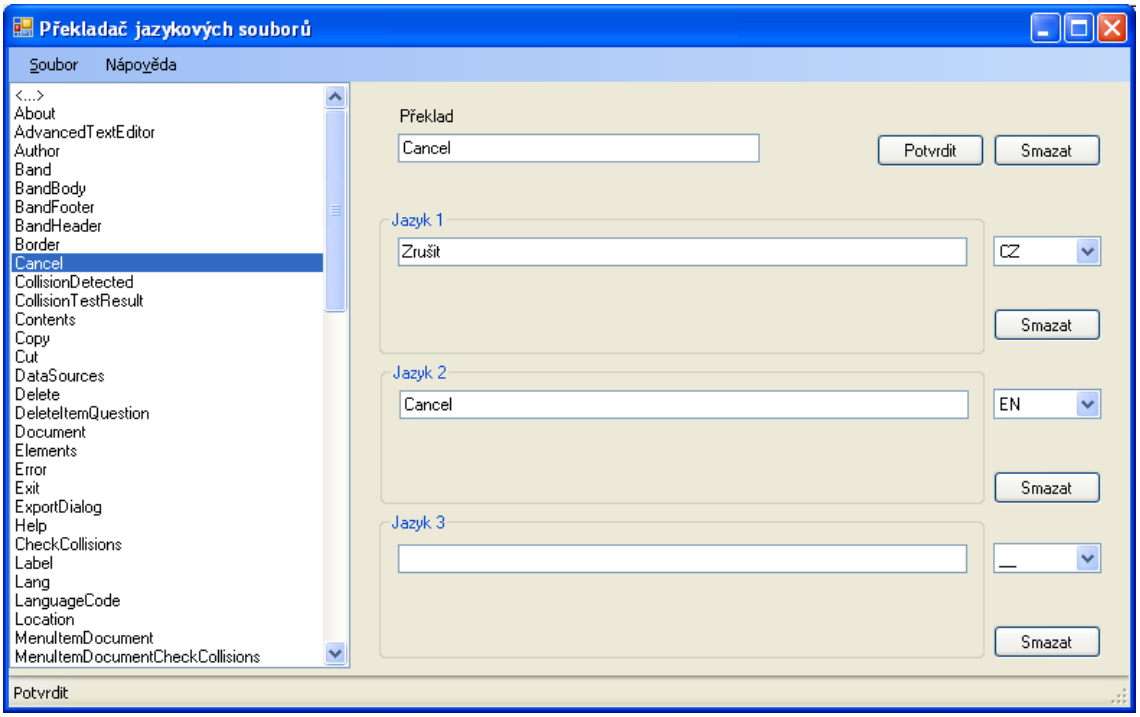
Postup:

- Výběr akce 

4.3 Popis použití aplikace pro překlady

Tato pomocná aplikace slouží k editaci překladových variant pro zástupný název použitý v aplikaci nebo na reportech.

Aplikace *MReporter* vyžaduje překlady uložené v souboru MReporter.lng. Překlady pro report musejí mít název souboru identický s názvem souboru reportu,



Obr. 18: Ukázka vlastního nástroje pro editací překladů

Aplikace je složena z těchto hlavních částí

- Seznam překladů – seznam v levé části
- Vybraný překlad – textové pole nahoře
- Potvrdit a smazat – ukládá zadaný překlad, maže celý aktuální překlad
- Jazykové varianty 1 až 3 – zde se zadávají překlady do konkrétních jazyků (tlačítko smazat vedle daného jazyku smaže všechny překlady pro daný jazyk)

Po otevření nebo založení nových překladů se můžeme pustit do jejich vkládání nebo oprav. Chceme-li založit nový překlad klikneme v levém seznamu na první řádek „<...>". V pravé části pak v poli „Překlad" doplníme zástupný text (bez mezer). V polích jednotlivých jazyků vypíšeme jazykové varianty a vždy potvrdíme tlačítkem „Potvrdit". Pokud z nějakého důvodu chceme překlad smazat, tak toto provedeme tlačítkem „Smazat" vedle tlačítka „Potvrdit". Tlačítka „Smazat" vedle překladových polí slouží k úplnému smazaní všech překladů pro konkrétní jazyk.

4.3.1 Práce se soubory

Založení nového překladového souboru

Založí novou databázi překladů.

Postup:

- Výběr akce  (Menu: Soubor => Nový)

Otevření existujícího jazykového souboru

Otevře existující překlady uložené v souboru.

Postup:

- Výběr akce  (Menu: Soubor => Otevřít)

Uložení editovaného jazykového souboru

Uloží editované překlady do souboru.

Postup:

- Výběr akce  (Menu: Soubor => Uložit)

V menu je dostupná i funkce Uložit jako.

Popis použití prezentační aplikace

4.4 Ukázkové aplikace

4.4.1 Popis

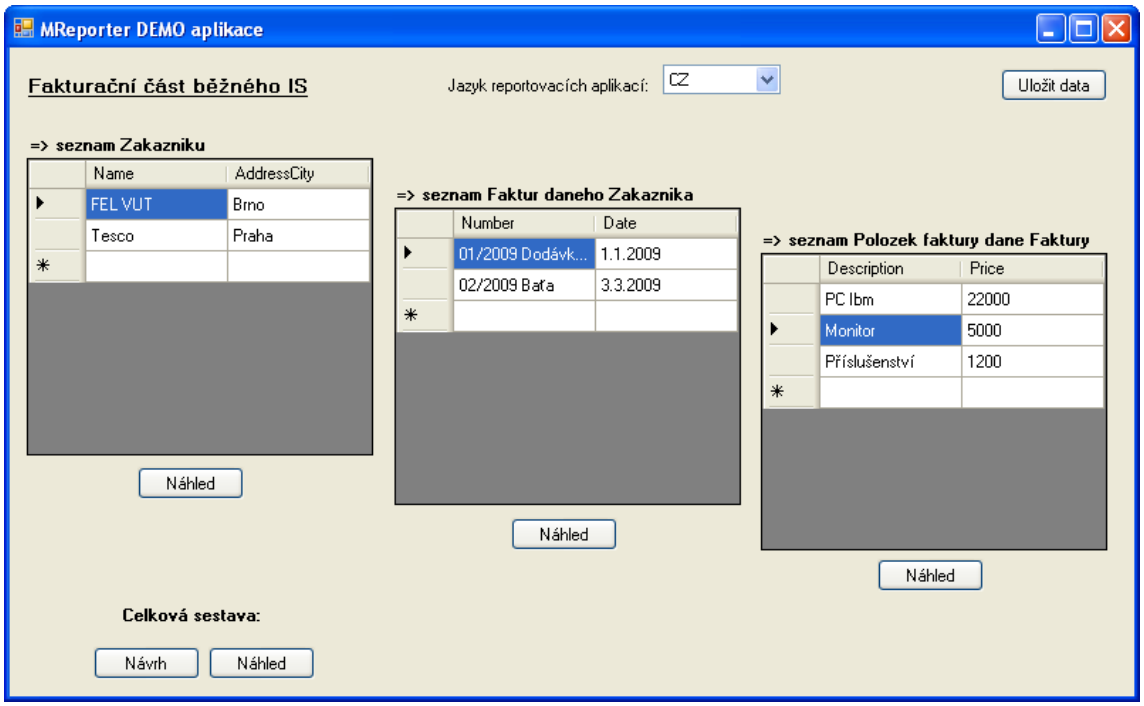
Aplikace, na které je výsledná aplikace testována simuluje fakturační část informačního systému. V systému existuje databáze zákazníků, jejich faktur, a příslušných položek těchto faktur. Je zde možno libovolně editovat a přidávat záznamy, pro jejich trvalé uložení je určeno tlačítko *Uložit data*. Od praxe se tento systém liší snad jen tím, že je vše na jedné

obrazovce. Tedy, po výběru zákazníka se aktualizuje seznam faktur, po výběru faktury se aktualizuje seznam jejích položek.

V horní části aplikace je přidán prvek pro volbu jazyka reportovacích nástrojů (doplněny jsou však jen překlady pro češtinu a angličtinu).

Typicky je každý seznam v informačním systému po náhledu možno vytisknout. To je možné i zde prostřednictvím tlačítek *Náhled* pod každým seznamem. Náhledy nemají definovanou šablonu. Šablona vzniká dynamicky, viz dále uvedená ukázka kódu pro náhled položek faktury. Dynamicky se toto provádí z důvodu jednotné úpravy.

Ve spodní části se nacházejí ještě dvě tlačítka. Ta jsou určena jednomu ukázkovému reportu se šablonou v souboru, kterou je možné tlačítkem *Návrh* měnit. Tlačítko *Náhled* zobrazí výslednou sestavu.



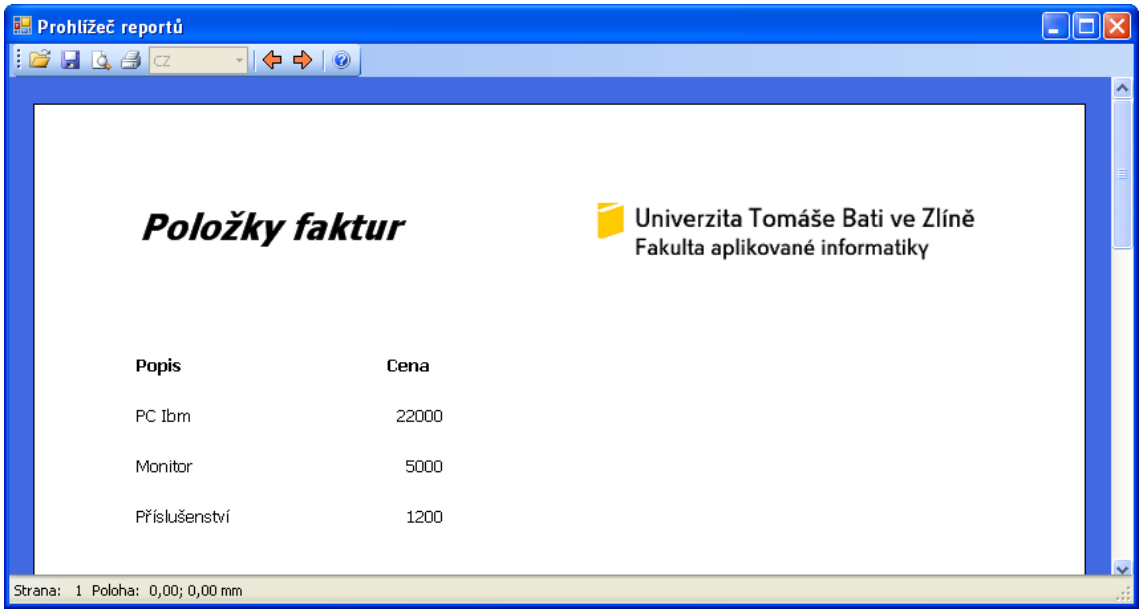
Obr. 19: Ukázková aplikace

4.4.2 Ukázka dynamicky tvořené šablony

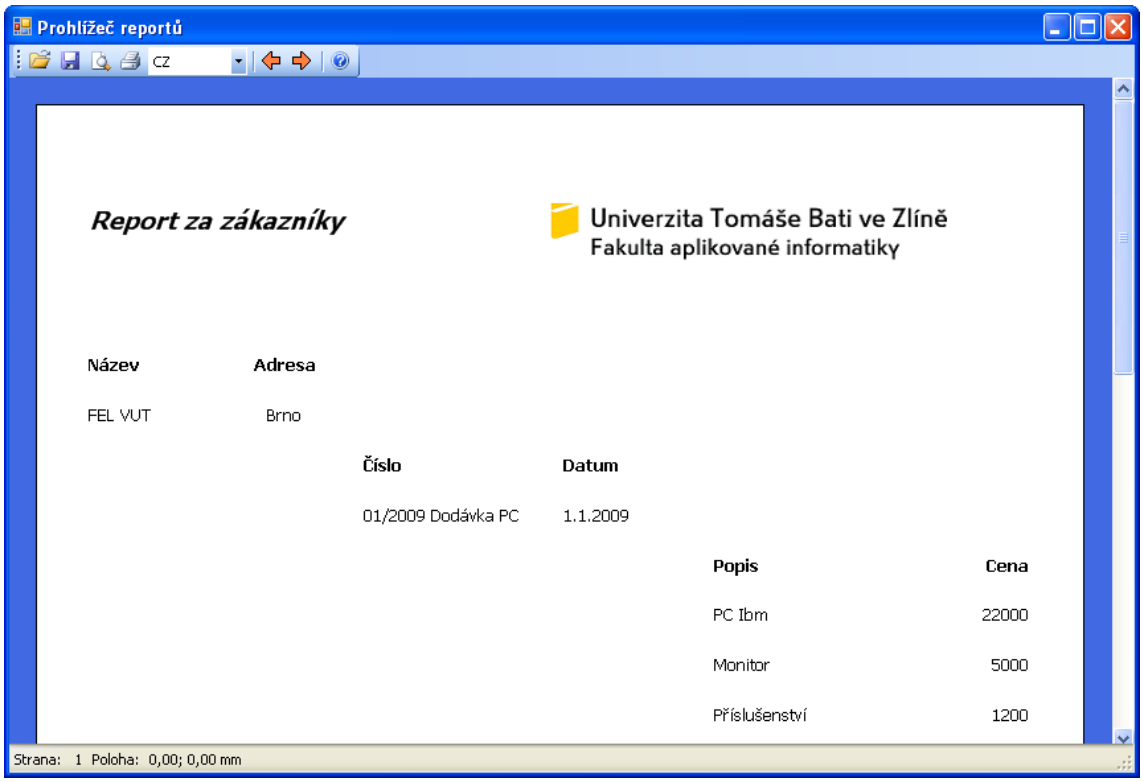
Následující kód založí dokument šablony a na ní definuje prvky, konkrétně se jedná o definici šablony pro „výpis položek faktur“ z ukázkové aplikace. Kód je složený jako celek, jeho části generují různé metody, aby byla zajištěna univerzálnost použití pro různé seznamy. V první části je napojen datový zdroj (řádek 4), dále je definována stránka

(řádek 5) a jednotlivé prvky stránky. Cyklický prvek (řádek 18) je naplněn záhlavím (řádek 23) a tělem (řádek 38).

```
1     MDocument document = new MDocument();

2     DataSourceObject dataSource = new DataSourceObject();
3     dataSource.DATA = this;
4     document.DataSourceObject = dataSource;

5     MPage page = new MPage();
6     document.Pages.Add(page);

7     MLabel labelTitle = new MLabel();
8     labelTitle.Location = new UnitValueXY(2, 2, MeassureUnitType.cm);
9     labelTitle.Text = title;
10    labelTitle.FontStyle = FontStyle.Bold | FontStyle.Italic;
11    labelTitle.FontSize = 20;
12    page.Elements.Add(labelTitle);

13    MPicture logo = new MPicture();
14    logo.Location = new UnitValueXY(10.5f, 1.5f, MeassureUnitType.cm);
15    logo.Size = new UnitValueXY(9, 2, MeassureUnitType.cm);
16    logo.ImageFileName = "logo_fai.png";
17    page.Elements.Add(logo);

18    MBand band = new MBand();
19    band.Name = "listband";
20    band.DataSource = bandDataSource;
21    band.Location = new UnitValueXY(0, 5, MeassureUnitType.cm);
22    page.Elements.Add(band);

23    MBandHeader header = new MBandHeader();
24    header.Location = new UnitValueXY(0, 0, MeassureUnitType.cm);
25    band.Header = header;

26    MLabel labelHead1 = new MLabel();
27    labelHead1.Location = new UnitValueXY(2, 0, MeassureUnitType.cm);
28    labelHead1.Text = header1;
29    labelHead1.FontStyle = FontStyle.Bold;
30    labelHead1.FontSize = 10;
31    header.Elements.Add(labelHead1);

32    MLabel labelHead2 = new MLabel();
33    labelHead2.Location = new UnitValueXY(7, 0, MeassureUnitType.cm);
34    labelHead2.Text = header2;
35    labelHead2.FontStyle = FontStyle.Bold;
36    labelHead2.FontSize = 10;
37    header.Elements.Add(labelHead2);

38    MBandBody body = new MBandBody();
39    body.Location = new UnitValueXY(0, 1, MeassureUnitType.cm);
40    band.Bodies.Add(body);

41    MLabel labelBody1 = new MLabel();
42    labelBody1.Location = new UnitValueXY(2, 0, MeassureUnitType.cm);
43    labelBody1.TextValue = @"listband[""Description""]";
44    labelBody1.FontSize = 10;
45    body.Elements.Add(labelBody1);

46    MLabel labelBody2 = new MLabel();
47    labelBody2.Location = new UnitValueXY(7, 0, MeassureUnitType.cm);
48    labelBody2.Size = new UnitValueXY(2, 0.5f, MeassureUnitType.cm);
49    labelBody2.TextValue = @"listband[""Price""]";
50    labelBody2.TextAlign = TextAlignTypes.TopRight;
51    labelBody2.FontSize = 10;
52    body.Elements.Add(labelBody2);
```



Obr. 20: Vygenerovaný náhled nad seznamem z ukázkové aplikace



Obr. 21: Vygenerovaný náhled sestavy z ukázkové aplikace

ZÁVĚR

Cílem práce bylo vytvořit jednoduchý reportovací nástroj pro platformu .NET. Po širší analýze bylo tohoto cíle dosaženo. Přestože výsledek nedosahuje kvalit komerčních řešení, je reálně použitelný. Popsaná i realizovaná funkčnost může být navíc i dobrým základem pro jiné aplikace, které by měly data v externích objektech vzájemně vyhodnocovat.

V přípravě na diplomovou práci jsem vycházel hlavně ze zkušeností, které jsem získal používáním reportovacího nástroje Perpetuum Report Sharpshooter. Srovnání dalších nástrojů jsem provedl dle konzultací s kolegy vývojáři v jiných firmách, kde tyto nástroje mají a taktéž jsem vycházel z popisů a specifikací funkčnosti na jednotlivých stránkách produktů. Co do funkčnosti i stylu prostředí jsou zmíněné nástroje přibližně srovnatelné, přihlédneme-li v porovnání na různé produktové řady a balíčky. Nelze však říci, že jsou stejné a výběr konkrétního nástroje bude vždy záležet na cílovém použití, osobních preferencí uživatelů a ceně.

Použitelnost těchto nástrojů je založena na několika málo prvcích, které v těchto produktech nalezneme pod různými názvy, ovšem téměř všechny oplývají stejnou funkčností. Analýzou těchto prvků a jejich vlastností jsem se dostal až k samotnému návrhu a tvorbě návrháře šablon a prohlížeče dokumentů. Součástí realizace je také proces zpracování zadaných výrazů, mapování výrazů na datový objekt, vyhodnocení výrazů na základě datového objektu a opětovné spárování výsledků s prvky šablony.

V rámci této práce se podařilo prověřit možnosti spolupráce reflexe a dynamického sestavení kódu. Vhodnou optimalizací mapování hodnot bylo následně dosaženo velmi efektivního a rychlého zpracování dat. Oproti exitujícím řešením byla přidána vestavěná podpora pro vícejazyčné šablony.

Výsledkem je funkční aplikace plnící základní funkčnost těchto nástrojů. Návrh byl však proveden dostatečně robustně, aby bylo možné rozšiřování funkčnosti, bez větších zásahů do jádra samotné aplikace.

ZÁVĚR V ANGLIČTINĚ

The goal of this work was to create a simple reporting tool for the .NET platform. That was achieved after a broader analysis. Although the result does not reach the quality of commercial solutions, it is realistically applicable. Described and implemented functionality can be a good basis for other applications evaluating data of external data objects.

I made the comparison of the other tools according to the consultations with colleagues and developers in other companies, where they have the tools and I also proceeded from the descriptions and specifications of the functionality of each product page. The functionality and the environment style of the mentioned tools are approximately comparable, if taken into account in relation to different product lines and packages. There is, however, say that they are the same and the choice of a particular instrument will always depend on the target application, personal preferences of users and price.

The applicability of these instruments is based on a few elements that can be found in these products under different names, but almost all have the same functionality. Analysis of these elements and their properties, I get to the actual design and creation of designer templates and browser documents. As part of the implementation process is also specified expressions, mapping expressions to the data object, the evaluation of the expressions of data object and re-pair the results with the elements of the template.

Part of this work was examines the possibilities for cooperation of reflection and dynamic code compilation. The appropriate mapping optimization were subsequently achieved a very efficient and rapid data processing. Compared to the existing solution built-in support for multilingual templates was added.

The result is a functioning application of fulfilling the basic functionality of these tools. The proposal was made sufficiently robust to allow the expansion of functionality without too much interference in the application itself.

SEZNAM POUŽITÉ LITERATURY

[1] KANISOVÁ, Hana, MÜLLER, Miroslav, *UML srozumitelně*. Computer Press, 2004. 158 s. ISBN 80-251-0231-9

[2] TROELSEN, Andrew, *Pro C-SHARP 2008 and the .NET 3.5 Platform*. Apress, 2007. 1370 s. ISBN 1-59059-884-9

[3] NAGEL, Christian, EVJEN, Bill, GLYNN, Jay, SKINNER, Morgan, WATSON, Karli. *Professional C-SHARP 2008*. Wiley Publishing, Inc. (Wrox), 2008. 1782 s. ISBN 978-0-470-19137-8

[4] *Design patterns: Data & Object Factory* [online]. [cit. 2009-01-23]. Dostupný z WWW: http://www.dofactory.com/Patterns/Patterns.aspx

[5] CONTOLI, A., *Simple Vector Shapes* [online]. Naposledy editováno 2008-03-27. Dostupný z WWW: <http://www.codeproject.com/KB/graphics/SimpleVectorShapes.aspx>

[6] VUGT, Wouter van. *Open XML The markup explained* [online]. 2007 [cit. 2009-01-23]. Dostupný z WWW: <http://openxmldeveloper.org/attachment/1970.ashx>

[7] STRAHL, Rick, *Dynamically executing code in .Net* [online] 2002 [cit. 2009-01-23]. Dostupný z WWW: <http://www.west-wind.com/presentations/dynamicCode/DynamicCode.htm>

[8] GORDON, Khendys, *Insert Plain Text and Images in RichTextBox* [online] 2006 [cit. 2009-01-23]. Dostupný z WWW: <http://www.codeproject.com/KB/edit/csexrichtextbox.aspx>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

.NET	Soubor technologií (platforma) společnosti Microsoft
ASP	Active Server Pages – webová skriptovací platforma
BMP	Microsoft windows BitMap – formát obrázku
GOF	Gang of Four – označení 4 autorů návrhových vzorů
HTML	HyperText Markup Language – značkovací jazyk použitý pro webové stránky
JPEG	též JPG, Joint Photographic Experts Group – formát obrázku
MRD	MReport Document – soubor s vygenerovaným dokumentem
MRT	MReport Template – soubor se šablonou
PNG	Portable Network Graphics – formát obrázku
RDL	Report Definition Language – XML jazyk pro reportovací nástroje
RTF	Rich Text Formát – formát textového dokumentu
XML	eXtensible Markup Language – značkovací jazyk

SEZNAM OBRÁZKŮ