

PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO  
KATEDRA INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

ePortfolio



2012

Pavel Šanca

## **Prohlášení**

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a souhlasím, aby práce byla zveřejněna v elektronické knihovně Přírodovědecké fakulty Univerzity Palackého a zpřístupněna ke studijním účelům.

.....  
Pavel Šanca

Děkuji panu Ing. Jiřímu Hronkovi za vedení při zpracování této práce. Dále děkuji své rodině a zaměstnavateli za jejich trpělivost.

## **Anotace**

Cílem webové aplikace ePortfolio je poskytnout studentům a instruktorům prostor pro publikování a průběžné aktualizování svých profesních a studijních zkušeností. Je napojena na e-learningový systém, ze kterého uživatel může čerpat data pro prezentování výsledků a průběhu studia.

## Obsah:

1. Úvod.....	7
1.1 Struktura práce .....	7
2. Použité technologie .....	8
3. Popis aplikace ePortfolio .....	9
3.1 Motivace k vytvoření aplikace.....	9
3.2 Tvorba obsahu.....	9
3.3 Oprávnění.....	10
4. Architektura aplikace .....	11
4.1 Prezentáční vrstva .....	11
4.2. Business vrstva.....	15
4.3 Datová vrstva .....	21
5. Uživatelská příručka .....	26
Závěr: .....	26
Conclusion: .....	29
Reference: .....	30
Příloha: .....	31
Obsah přiloženého DVD:.....	32

## **Seznam obrázků:**

Obrázek 1. Interface rozhraní v aplikaci.....	16
Obrázek 2. Class diagram.....	18
Obrázek 3. ER Diagram.....	25

# 1. Úvod

Jako téma bakalářské práce jsem si zvolil vývoj aplikace ePortfolio, která si klade za cíl poskytnout studentům a instruktorům prostor pro zveřejňování a průběžnou aktualizaci svých studijních zkušeností. Pokud si vytvoří takovéto ePortfolio, mohou ho následně využít nejen pro studijní účely, ale také je nabídnout svým potenciálním zaměstnavatelům.

Moje motivace vychází z potřeb mé dosavadní programátorské praxe, kde byly webové aplikace doposud implementovány v klasickém ASP (Active server pages), jejichž architektura byla řešena nebo neřešena pomocí tzv. spaghetti codu. Logika vlastní aplikace je v takovém případě prolnta s prezentační a ukládací logikou. Důsledkem tohoto řešení je velmi nízká možnost takovouto aplikaci jednoduše rozšiřovat. Každý zásah může způsobit problém v jiné části aplikace, a tudíž se rozvoj takovéto aplikace prodražuje a vzniká větší množství neočekávaných chyb.

Cílem této práce je tedy realizace vícevrstvé jasně strukturované webové aplikace s jednoduchou škálovatelností, kde každá vrstva se zaměřuje pouze na úkoly, které spadají do její oblasti. Databáze bude sloužit jen jako úložiště relačních dat a neměla by obsahovat jakoukoliv část implementace hlavní logiky aplikace. O to se bude starat výhradně jenom jedna z vrstev.

## *1.1 Struktura práce*

Počátek práce je věnován použitým technologiím a postupům. Další kapitola se týká samotné aplikace ePortfolio. Zde bude popsána stručně její funkčnost a oprávnění. Podrobněji bude aplikace představena ke konci práce v uživatelské příručce. Klíčová kapitola pojednává o architektuře aplikace. Podrobněji se věnuje jednotlivým vrstvám a jejich vzájemné spolupráci. V závěru práce zhodnotím zkušenost s programování ePortfolia, jeho přínos a případné nedostatky.

## 2. Použité technologie

Aplikace je napsána v jazyce C# využívající ASP.NET framework běžící pod IIS 7.0. Jako úložiště slouží SQL Server 2008. Další použité technologie jsou LINQ to SQL, LINQ to objects. Jako editor kódu je použito Visual Studio 2010. Jazyk C# a SQL jsem již ovládal, částečně i ASP.NET, hlavní výzvou pro mě byly technologie týkající se LINQu.

- ASP.NET – framework pro tvorbu webových aplikací od firmy Microsoft.
- Jazyk C# - objektově orientovaný jazyk od firmy Microsoft.
- IIS 7.0 - webový server od firmy Microsoft.
- SQL Server 2008 – databázový server od firmy Microsoft.
- LINQ to SQL – implementace obecného dotazovacího jazyka LINQ nad databázemi využívající rozhraní MS SQL.
- LINQ to Objects – implementace obecného dotazovacího jazyka LINQ nad objekty.
- JavaScript – objektově orientovaný scriptovací jazyk, běžící v klientském prohlížeči.
- JQUERY – JavaScriptová knihovna zjednodušující práci s DOM objekty prohlížece.



## **3. Popis aplikace ePortfolio**

### ***3.1 Motivace k vytvoření aplikace***

Aplikace ePortfolio byla vytvořena na základě konkrétního požadavku vycházejícího z praxe. Zadavatelem byla společnost WebStudy EU s.r.o., která se zabývá vývojem a údržbou systému LMS (learning management system). Cílem aplikace ePortfolio je umožnit studentům a instruktorům e-learningového systému publikovat a průběžně aktualizovat svoje studijní a profesní zájmy a zároveň čerpat již jednou vložená data z e-learningového systému. Ústřední myšlenkou byla potřeba studentů prezentovat svoje studijní a profesní zkušenosti a průběžně je doplňovat s cílem je využít při oslovování případných budoucích zaměstnavatelů.

### ***3.2 Tvorba obsahu***

Prostředky pro uživatele nabízí flexibilitu při tvorbě obsahu uživatelského portfolia. Uživatel má volnost vytvořit si svoje vlastní dvojdimenzionální menu. Každé menu představuje jednu stránku ePortfolia, které může mít neomezeně mnoho položek. Jedním ze základních požadavků od uživatelů byla možnost nastavení různých typů oprávnění na jednotlivé položky stránky ePortfolia tak, aby host a člen fakulty nebo potencionální zaměstnavatel si mohl prohlédnout takové informace, které pro něj tvůrce ePortfolia zamýšlel zveřejnit.

Při tvorbě obsahu ePorfolia má uživatel k dispozici různé nástroje. Hlavní publikační nástroj je položka typu článek. Pro snadnou tvorbu článků má uživatel k dispozici WYSIWYG editor <sup>1</sup>, který nabízí formátování písma, odstavců atd. přímo v prohlížeči uživatele, aniž by uživatel musel znát HTML kód. Další možností, jak dostat obsah do svého ePorfolia je využití uložených informací v LMS systému. Jsou to především informace o kurzech, které student právě studuje nebo v minulosti studoval. Tyto položky čerpané z LMS systému mají bohaté možnosti nastavení. Uživatel se tak může rozhodnout, jestli například k

---

<sup>1</sup> „co vidíš, to dostaneš“ (<http://cs.wikipedia.org/wiki/WYSIWYG>)

již absolvovaným kurzům připojí i informace o dosažené známce nebo komentáře instruktora.

Grafické zpracování kopíruje logiku aplikace. Snaží se maximálně zachovat jednoduchost ovládání, ergonomii při využití moderního a přívětivého designu.

Detailní popis funkčnosti aplikace je součástí Uživatelské příručky.

### **3.3 Oprávnění**

Protože jednotlivé položky ePortfolia mohou sebou nést různá nastavení, tak za použití oprávnění může uživatel použít položku opětovně, jinak tuto položku nastavit a přiřadit jiné oprávnění pro prohlížení. Systém rozeznává čtyři základní role:

- Vlastník – majitel a tvůrce ePorfolia;
- Člen instituce – vlastní login do LMS systému;
- Host – přichozí uživatel, který nemá login do LMS systému;
- Zaměstnavatel – uživatel, který obdržel od vlastníka ePorfolia unikátní adresu pro přístup.

Každá stránka ePorfolia obsahuje nastavení, zda je možné přidávat komentáře od návštěvníků.

Vzhledem ke vzrůstající oblibě a využívání sociálních sítí je možné každou položku ePorfolia hodnotit pomocí tlačítka „Like“ od návštěvníků, kteří mají účet na Facebooku.

## 4. Architektura aplikace

Aplikace se sestává ze tří projektů. Každý projekt reprezentuje jednu vrstvu. Prezentační vrstva je implementována jako ASP.NET aplikace napsaná v jazyce C#. Business a datová vrstva jsou napsány ve formě C# knihovny. Nasazení této aplikace tedy znamená publikovat webovou aplikaci na server s IIS 7.0 spolu s těmito knihovnami a publikovat databázi na SQL server – pokud možno na samostatný stroj.

Vrstvy jsou navzájem postaveny v hierarchii, kde prezentační vrstva je na nejvyšší úrovni a datová na nejnižší. Této hierarchii odpovídá směr komunikace od nejvyšší po nejnižší. Každá vrstva může požadovat data, jenom od své nejbližší nižší vrstvy.

Pokud by se aplikace dále rozšiřovala a rostla by náročnost logiky aplikace a tím pádem i nároky na výpočetní výkon, bylo by vhodné prezentační vrstvu oddělit od zbytku aplikace a nasadit na samostatný server. Aby tento proces mohl být jednoduchý a mohlo dojít ke škálování aplikace, prezentační vrstva nepracuje přímo s objekty business vrstvy (BO – business objects), ale s objekty typu DTO<sup>2</sup>. Na tyto objekty lze nahlížet jako na objekty business vrstvy, ale upravené právě pro potřeby prezentační vrstvy. Tudiž neobsahují žádné metody a jejich součástí jsou jenom data, které prezentační vrstva potřebuje. V aplikaci jsou implementovány jako XML. Díky XML je možné prostřední vrstvu obalit mechanismem webservicem, kdy prostřední vrstva přijímá požadavky pomocí http volání a výsledkem jsou data právě ve formě XML, které jsou následně zpracována prezentační vrstvou.

Oproti tomu jsem aplikaci nenavrhol s tím, že by došlo k oddělení business a datové vrstvy v budoucnu, a proto jsou tyto dvě vrstvy těsněji spjaty a jako jejich DTO objekty slouží přímo objekty datové vrstvy.

### 4.1 Prezentační vrstva

Prezentační vrstva této webové aplikace (ASP.NET) zajišťuje dodání výstupu uživateli ve formě HTML. Přímou komunikuje pouze s business vrstvou. Tato vrstva se plně soustředí jen na otázky výstupu a obsluhy uživatelských akcí a je zodpovědná za udržování informací o přihlášeném uživateli (user session).

---

<sup>2</sup> Data transfer objects je návrhový vzor použitý pro přenos dat mezi jednotlivými komponentami systému. Objekty neobsahují žádné chování, jde pouze o data (volně přeloženo z [http://en.wikipedia.org/wiki/Data\\_transfer\\_object](http://en.wikipedia.org/wiki/Data_transfer_object))

### ***4.1.1 Vykreslování HTML***

Během návrhu webové aplikace jsem se rozhodl vyzkoušet trochu jiný přístup než je v typické ASP.NET aplikaci obvyklé. ASP.NET nabízí koncept vytváření HTML šablon v souborech typu ASPX, které čerpají data z připojených kódových souborů, kde jsou naimplementovány třídy. Výsledkem tohoto přístupu bývá pro každý webový formulář jedna ASPX stránka a připojený CS soubor.

V mém případě, ale webová aplikace obsahuje pouze dvě ASPX stránky, a to stránku která generuje HTML menu aplikace (MasterPage.master) a hlavní stránku (main.aspx), která generuje zbylý obsah a obsluhuje všechny uživatelské funkce. Obě tyto stránky jsou k sobě připojeny konceptem master page dostupným v ASP.NET.

Hlavní vykreslovací logika je řešena v aplikaci pomocí XSLT (EXtensible Stylesheet Language) transformací z XML na HTML. Jedná se tedy vlastně o přijetí DTO objektu z business vrstvy ve formě XML zavolání správné vykreslovací šablony a vygenerování HTML, které je posláno na výstup. Každá případná stránka je tedy nahrazena XSLT souborem, který daný objekt převede na HTML. Některé transformační šablony mohou odkazovat na jiné transformační šablony, které v sobě implementují transformace obecných zobrazovacích prvků vyskytujících se ve webových formulářích, tudíž se nemusí programovat vícekrát stejně. Jedná se například o společné vlastnosti různých typů článků jako jsou nadpis článku, datum vytvoření nebo generování menu pro položky na stránce. K tomuto přístupu jsem se rozhodl proto, abych se seznámil s XSLT transformacemi, vyzkoušel jak je možno šablony spojit dohromady a využil sílu jazyka pro zpracování složitějších XML struktur. Úskalím tohoto přístupu je složitější spolupráce transformačních šablon během stylování se zbytkem webové aplikace. Například editační menu pro jednotlivé články ePortfolia je dostupné jen pro přihlášené uživatele, šablona o této skutečnosti neví, a proto musí volat externí metodu webové aplikace pro rozhodnutí, zda-li editační menu vygenerovat, což způsobuje zpomalení celé transformace na HTML. Hlavní nevýhodou je ovšem nemožnost šablony vykreslovat ASP.NET web user controls, které mohou být pro výstup potřeba. V mém případě se jednalo zejména o RTF editor od firmy, který slouží pro zadávání textu článků uživatelem, kdy je možno přímo v prohlížeči bez použití HTML formátovat texty. Aby bylo možno tento editor použít, je nutné ho do výsledné stránky přidat mimo transformaci, čímž se narušuje princip, že vše co se týká HTML výstupu, je v šablonách.

### ***4.1.2 Použití update panelu***

Pro zpříjemnění práce uživatele při editaci článků na webu jsem zvolil použití update panelu – standardní součásti ASP.NET. Tento ovládací prvek slouží pro překreslení určité části obrazovky bez nutnosti znova vykreslit prvky, které zůstávají nezměněny.

Této funkcionality aplikace využívá při editování článků uživatelem. Po odeslání změny článku uživatelem aplikace změnu uloží a vykreslovací engine vygenerovanou pozměněnou stránku posílá do update panelu pro překreslení a například hlavní menu aplikace zůstává nezměněno.

### ***4.1.3 Routing***

Aby mohl mít každý uživatel k dispozici URL svého ePortfolio, které by neobsahovalo jeho userid, ale přímo jeho jméno v následujícím formátu EpWeb/people/username/, implementuje aplikace IHTTP handler pro všechny příchozí HTTP požadavky, kdy uživatelské jméno je následně přeloženo na userid a aplikace samotná již pracuje s userid.

### ***4.1.4 Použití facebook api***

Pro vytvoření napojení s populární sociální sítí Facebook, je v aplikaci použito facebook api<sup>3</sup>. Ke každé položce článku se může návštěvník, který má na Facebooku účet vyjádřit pomocí facebookového tlačítka “Like”. Pokud je tlačítko zmáčknuto a návštěvník není na Facebook přihlášen, spustí se malý dialog pro přihlášení a následně je do Facebooku odeslána zpráva o zmáčknutí tlačítka. To se projeví na návštěvníkově Facebook stránce tím, že je zde zobrazena zpráva, že uživateli se líbil nějaký článek a následuje odkaz na článek na ePortfolio.

Facebook nabízí dva způsoby, jak sociální pluginy integrovat do aplikace. Jedna možnost je v HTML stránce použít tag IFRAME, jehož zdroj je nasměrován na stránky Facebooku. Součástí URL zdroje je parametr , který představuje odkaz, který bude uveden vedle sdělení, že se uživateli nějaký článek líbí. Druhou možností je použití JavaScriptového

---

<sup>3</sup> Aplikační rozhraní zpřístupňující funkce aplikace Facebook.

rozhraní Facebooku, které je elegantnější a není potřeba v aplikaci generovat v souvislosti s integrací tolik kódu. Aplikace ePortofolio využívá Facebook JavaScript API, a proto je na Facebooku zaregistrována a používá jedinečný klíč, který pro každou aplikaci Facebook vygeneruje. V master page webové aplikace ePortfolia je reference na JavaScriptový soubor Facebook API a zavoláním funkce `init` společně s dodaným klíčem, probíhá generování všech Facebook prvků na stránce. Samotné vytvoření tlačítka pak znamená umístit do stránky speciální tag `<fb:like>`.

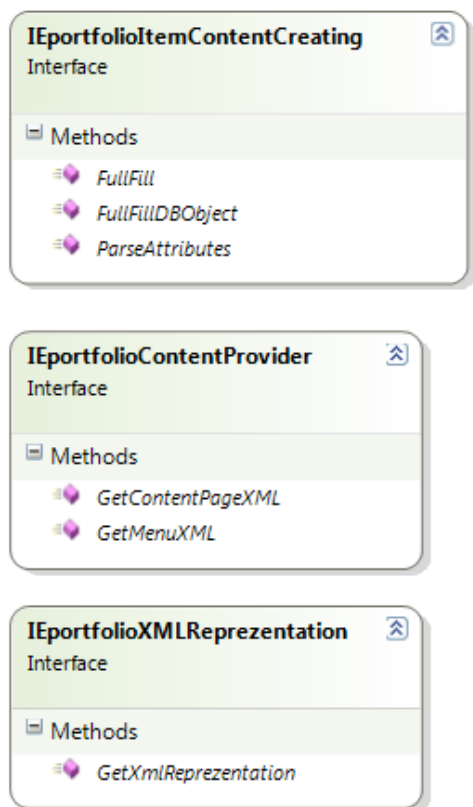
## **4.2 Business vrstva**

V business vrstvě se skrývá hlavní logika aplikace. Reaguje na požadavky prezentační vrstvy, získává objekty z datové vrstvy, žádá datovou vrstvu o uložení objektu v případě, že uživatel něco editoval, kontroluje uživatelská oprávnění a řeší jaké položky zobrazit podle přihlášeného uživatele. Každá položka na stránce může mít nastaveno oprávnění, jaký typ uživatele ji smí prohlížet. Ve výsledku převede BO na DTO objekty a posílá zobrazovací vrstvě.

### **4.2.1 Implementace business objektů**

Hlavní pojmy, se kterými aplikace ePortfolio pracuje je uživatel, vyžádaná stránka a její obsah. Obsah stránky je tvořen různými typy položek. Může to být článek, kontaktní informace, seznam kurzů uživatele, oznamované úlohy nebo zkoušky z jednotlivých kurzů, komentář ke stránce nebo položka typu embeded objekt, který udržuje a spouští obsah z jiného webu přímo na stránce uživatele (např. video z youtube.com)

Tuto skutečnost implementuje business vrstva a ke každému s těchto pojmů přiřazuje business objekt. Jejich konkrétní implementace využívá dědičnosti pro zachycení společných rysů objektů a také rozhraní IEportfolioXMLRepresentation pro podporu XML formátu, které musí implementovat každý objekt, který je poslán zobrazovací vrstvě.



Obrázek 1. Interface rozhraní v aplikaci.

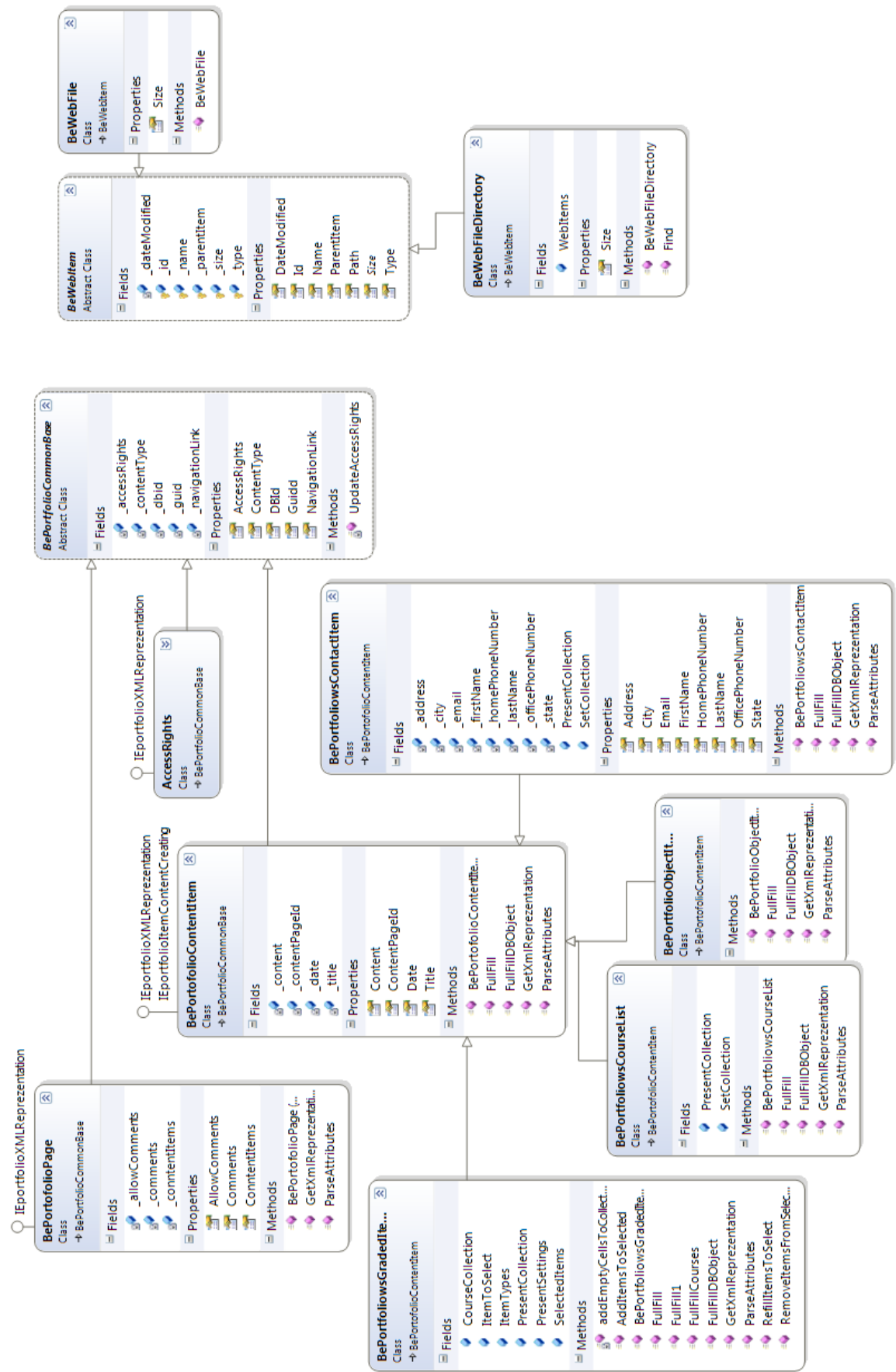
#### 4.2.1 Hierarchie objektů

V aplikaci jsou všechny objekty odvozené od základní abstraktní třídy *BePortfolioCommonBase*, která udržuje databázové ID objektu a nastavená uživatelská oprávnění. Obsahové položky stránky navíc dědí z třídy *BePortfolioContentItem*, který implementuje společné atributy pro položky na stránce jako jsou název položky, datum poslední aktualizace, příslušnost k určité stránce.

Bázová třída pro položky *BePortfolioContentItem* nachází využití hlavně při generování stránky ePortfolio. Hlavní cyklus pracuje s proměnnými typu *BePortfolioContentItem* a volá metody tohoto objektu, které jsou ale označené jako virtuální – tudíž se vykonají konkrétní metody potomka, které jsou implementovány pro každý typ položky jinak.



Aby proměnná typu `BePortofolioContentItem` byla naplněna správným konkrétním potomkem, je v aplikaci implementován návrhový vzor `Factory`. `Factory` představuje jednu konkrétní třídu, která ve své metodě `GetNewItem`, přijímá datový objekt položky a na základě jednoho atributu, který představuje typ položky vytvoří správný objekt a tím je proměnná typu `BePortofolioContentItem` naplněna konkrétním objektem.



Obrázek 2. Class diagram.

#### **4.2.2 Spolupráce s prezentační vrstvou**

Při navrhování typu spolupráce mezi prezentační a business vrstvou jsem vycházel z četných doporučení dostupných na internetu týkajících se návrhu aplikací. Proto jsem zavedl do business vrstvy rozhraní `IEportfolioContentProvider`. Toto rozhraní popisuje všechny dostupné metody, které může zobrazovací vrstva použít. Výhodou tohoto přístupu je, že prezentační vrstva zná právě jenom toto rozhraní, což odděluje prezentační vrstvu od aplikační a přispívá k možné škálovatelnosti aplikace. V konstruktoru tohoto rozhraní jsou specifikované atributy, které musí zobrazovací vrstva použít pro úspěšné volání metod business vrstvy. Jedná se zejména o ID přihlášeného uživatele a či `ePortfolio` je žádáno. Objekt, který rozhraní implementuje na straně business vrstvy je `EportfolioContentProvider`. V tomto objektu je naimplementována většina aplikační logiky, ta se odehrává nad BO objekty, které se filtrují podle vstupních parametrů prezentační logiky, kontrolují na oprávnění a na závěr se pomocí metod rozhraní `IEportfolioXMLRepresentation` převedou na XML a odesílají se prezentační vrstvě.

#### **4.2.3 Spolupráce s datovou vrstvou**

Hlavním rysem této spolupráce je technologie LINQ<sup>4</sup>. V tomto případě jsem použil konkrétně implementaci LINQ to Objects. Protože datová vrstva obsahuje objekty, které kopírují tabulky v databázi, business vrstva může pomocí LINQu tyto objekty prohledávat podle kritérii a následně získaná data převádět do BO pro svoji potřebu. Datové objekty obsahují vlastně kopii databázové struktury, takže je možno vyhledávat podle všech dostupných vlastností objektů. Výhodou použití LINQ to objects je, že business vrstva nepracuje s jazykem SQL, ale vlastně obecným vyhledávacím jazykem, takže při změně implementace datových objektů (místo DB by byl použit např. souborový systém) se nemusí kód business vrstvy změnit.

Pokud business vrstva potřebuje nějaký změněný nebo nový objekt uložit, naplní příslušný datový objekt, který je definován v datové vrstvě a požádá vrstvu o uložení. Datová vrstva, kromě samotného datového objektu např. `ePortfolioPage` (objekt reprezentující jednu

---

<sup>4</sup> LINQ (anglicky Language Integrated Query) je integrovaný jazyk pro dotazování, který byl představen spolu s jazyky C# 3.0 a Visual Basic 9 spolu s .NET Frameworkem 3.5. LINQ přináší nový způsob pro dotazování nad jakýmkoliv daty, usnadňuje jejich tvorbu, třídění, jejich propojování i vyhledávání v nich. (přeloženo z [www.wikipedia.com](http://www.wikipedia.com) – staženo dne 23.4.2012)

uživatelskou stránku ePortfolia) poskytuje také přidružený objekt např. `elsePortfolioPageRepository`, který právě používá business vrstva pro ukládání datových objektů.

## 4.3 Datová vrstva

Při návrhu aplikaci bylo mým cílem vytvořit co nejjednodušší datovou vrstvu, která neobsahuje žádnou aplikační logiku a stává se tedy velice jednoduchým uložištěm. Zároveň to byla možnost vyzkoušet si některý z ORM (Object-relational mapping) frameworků. ORM frameworky představují jednoduchou možnost, jak vytvořit z tabulek databáze propojené objekty, které kopírují databázovou strukturu a podporují jazyk LINQ to SQL.

### 4.3.1 Použití ORM

V projektu datové vrstvy je použit LINQ to SQL generátor, který po připojení do databáze načítá databázové schéma (tabulky, reference, vložené procedury, uživatelsky definované funkce atd.) a přenesení je do konfiguračního XML souboru s příponou dbml (DataBase Markup Language). Schéma je následně zkontrolováno, jestli neobsahuje chyby a soubor je předán generátoru kódu. Generátor vytvoří příslušné třídy tabulek. Atributy těchto tříd jsou automaticky vygenerovány podle sloupců příslušných tabulek. Vygenerované třídy obsahují další atributy, které zachycují provázanost tabulek z databáze. Například třída `ePortfolioPage` (uživatelská stránka `ePortfolia`), která reprezentuje stejnojmennou tabulku, implementuje také vlastnost s názvem `ePortfolioItems` – odkaz na tabulku `ePortfolioItem` (vložené položky na stránce). Použití tohoto konceptu je pak v aplikaci velice snadné. Stačí načíst konkrétní `ePortfolioPage` a hned má programátor k dispozici bez dalšího programování i položky stránky.

Dále tyto třídy automaticky implementují rozhraní `IQueryable`, takže je možné provádět LINQ dotazy. Generátor vytváří navíc ještě jednu speciální třídu `EpDataClassesDataContext`, která dědí třídu `System.Data.Linq.DataContext`. Tato třída představuje datový kontext, ve kterém se odehrávají případné změny v jednotlivých objektech a poskytuje hlavní metody pro uložení celého datového kontextu – metoda `submitchanges`.

Mým záměrem bylo použití přímo třídy `EpDataClassesDataContext` v business vrstvě pro uložení změn v datových objektech do databáze. Při prototypování tohoto řešení jsem ovšem narazil na problém, kdy jsem například vytvořil novou instanci objektu `ePortfolioPage` a rovnou do ní přiřadil několik položek typu `ePortfolioItems`. Při pokusu o uložení pomocí objektu `EpDataClassesDataContext`, se stránka uložila, ale již se neuložily podřízené

položky stránky. Je to proto, že LINQ to SQL neumožňuje v základu pracovat v tzv. disconnected módu. Disconnected mód nastává, když LINQ objekt opouští svůj datacontext, je změněn a pomocí jiného datacontextu je uložen. Tento scénář je typický pro vícevrstvé aplikace, kdy během jednoho webového požadavku je načten objekt a zapomeno se datacontext, uživatel objekt změni a během druhého požadavku je objekt pomocí nového datacontextu uložen. Tento problém popsal a vyřešil Adrian Grigore, když o použití LINQ v disconnected módu říká<sup>5</sup>: „LINQ to SQL je fantastický doplněk frameworku .NET! Představuje typově bezpečný, výkonný a extrémně flexibilní způsob jak implementovat přístup k datům v .NET aplikaci. Bohužel použití ve vícevrstvených aplikacích již není tak bezproblémové”. Grigore vyvinul generickou základní třídu RepositoryBase RepositoryBase<TEntityType, TContextType>, která implementuje základní operace Load, Save a SaveRecursively nad objekty TEntityType pomocí datacontextu TContextType. Jako TEntityType volíme LINQ objekty a TContextType je příslušný LINQ to sql datacontext. Pokud použijeme tuto třídu, objekty, které byly změněny v rámci disconnected módu budou v pořádku uloženy. Navíc metoda SaveRecursively dokáže správně uložit i vnořené objekty hlavního nadobjektu. Jako bonus bazová třída poskytuje výpis všech vygenerovaných SQL příkazů, které vznikly dotazováním nad LINQ objekty. Výstup příkazů může sledovat na console ve Visual Studiu během ladění aplikace.

Zavedení těchto bazových tříd do datové vrstvy bylo celkem snadné. Kódové soubory, kde je třída implementována se přidaly do projektu a bylo potřeba každý LINQ objekt rozšířit o statickou metodu CreateRepository(), která vrací RepositoryBase. Zavoláním této metody odkudkoliv z programu se vytvoří příslušné repository a již je možno používat funkce pro načítání a ukládání. Tyto metody potom využívá business vrstva, která vytváří instanci repository příslušného LINQ objektu a provádí dotazy pro vybrání správného objektu nebo požádá repository o uložení.

Protože aplikace ePortfolia pracuje nejenom se svými databázovými tabulkami, ale používá i tabulky připojeného LMS systému, představovalo použití technologie LINQ to SQL významnou úsporu času při tvorbě datových objektů, maximálně zjednodušilo dotazování se nad datovými objekty a procházení jejich hierarchické struktury.

---

<sup>5</sup> v článku <http://www.codeproject.com/Articles/29966/An-ASP-NET-Data-Layer-Base-Class-for-LINQ-Disconne>.

### **4.3.2 Databázová struktura**

Aplikace ePortfolia uchovává svoje data v databázových tabulkách a pracuje také s cizími tabulkami připojeného LMS systému.

Při návrhu databázového schématu nových tabulek aplikace bylo mým cílem navrhnout jednoduchou strukturu, která by se snadno rozšiřovala. Zvláště jsem bral v potaz možnost, že přibudou nové typy položek, které mohou být uloženy na stránku ePortfolia.

### **4.3.3 Přehled a hlavní rysy tabulek aplikace**

#### **4.3.3.1 ePortfolioDesc**

Každý uživatel, který vlastní ePortfolio má v této tabulce záznam, je zde uložen název ePortfolia, uživatelem vytvořená struktura jeho menu ve formátu XML a také vygenerované URL pro použití potencionálním zaměstnavatelem. Uložení menu ve formátu XML do sloupce Menu zjednodušilo tvorbu schématu a umožňuje velmi snadné rozšíření do budoucna. Datová vrstva vrátí informace o menu business vrstvě a ta již sama prochází strukturu a provádí následné akce. Nevýhodou tohoto přístupu je možné zpomalení při vykonávání aplikace oproti klasickému přístupu, kdy samotné menu a položky jsou většinou uloženy v samostatných tabulkách. Proto by se v budoucnu mohl zavést cache mechanismu, pro udržování málo měněných informací, pravděpodobně v business vrstvě aplikace. Business vrstva zajišťuje validnost XML menu a také při každé změně menu žádá datovou vrstvu o přidání nového záznamu do ePortfolioPage a zavede referenci na novu stránku do XML pomocí primárního klíče ePortfolioPageID. Primárním klíčem je zde sloupec EPortfolioID.

#### **4.3.3.2 ePortfolioPage**

Představuje úložiště pro metadata jednotlivých uživatelských stránek ePortfolia, referencuje tabulku ePortfoliDesc pomocí cizího klíče EPortfolioID. Primární klíč je ePortfolioPageID. Dále je tabulka zodpovědná o udržování, zda si přeje uživatel povolit pro tuto stránku komentáře pro návštěvníky či nikoliv. Pro tento účel slouží sloupec AllowComments.

#### **4.3.3.3 ePortfolioItem**

Je úložiště pro všechny položky na stránce ePortfolia, referencuje tabulku ePortfolioPage pomocí cizího klíče ePortfolioPageID. Důležitým atributem je zde sloupec ContentType a Content. První atribut popisuje o jaký typ položky se jedná. Zda je to například článek či kontaktní informace. A druhý atribut typu VARCHAR(max) obsahuje samotný obsah položky. V případě , že položka je článek, je zde přímo uložen obsah článku ve formě HTML, který uživatel vytvořil. Naproti tomu pokud se jedná o propojenou položku například kontaktní informace, je zde uloženo XML s nastavením, které informace o sobě si přál uživatel zveřejnit. Tyto informace čte business vrstva a zažádá datovou vrstvu o jinou tabulku z LMS, s kontaktními údaji uživatele a posílá je přefiltrované prezentační vrstvě.

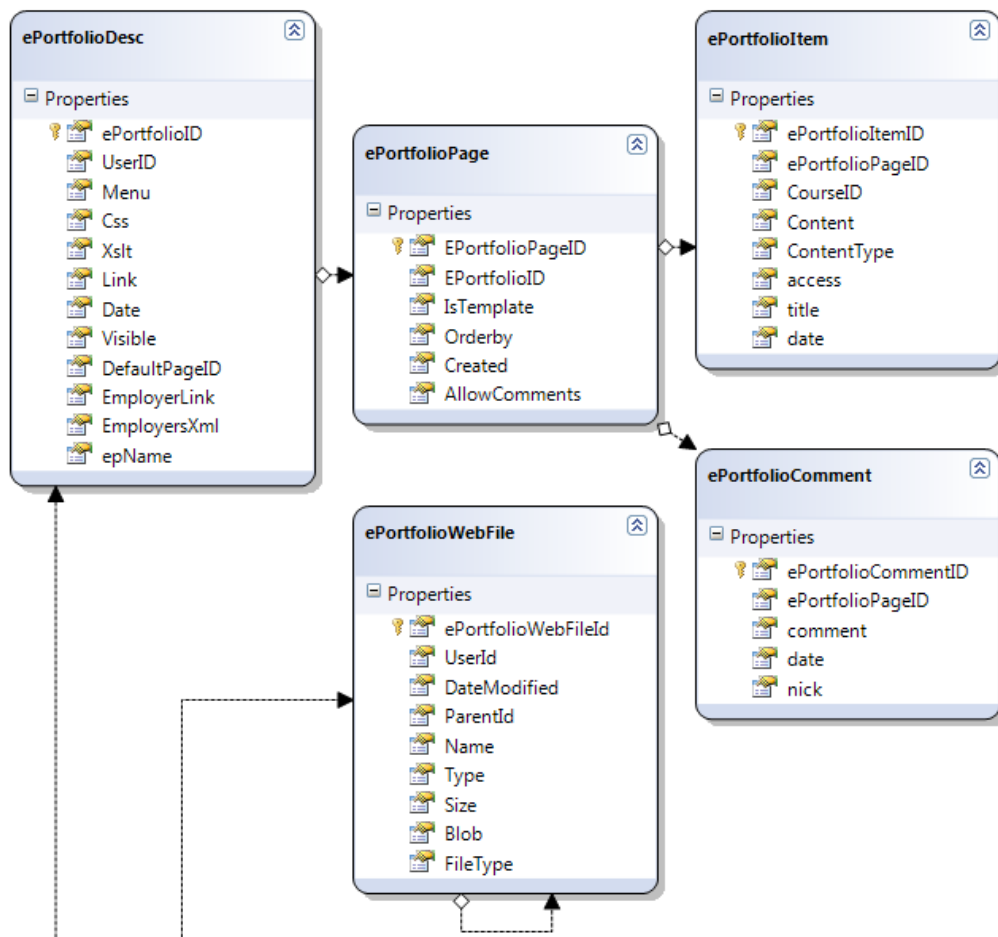
#### **4.3.3.4 ePortfolioComment**

Slouží jako úložiště pro komentáře vytvořené návštěvníky stránky, referencuje tabulku ePortfolioPage tabulku pomocí cizího klíče.

#### **4.3.3.5 ePortfolioWebFile**

Poskytuje úložiště pro obrázky a dokumenty, které uživatel může prostřednictvím RTF editoru nahrát jako součást své prezentace. Tabulka implementuje jednoduchou adresářovou strukturu pomocí odkazu sama na sebe ve sloupci ParentID. Samotný obsah souboru je potom uložen ve sloupci s názvem Blob, který je typu VarBinary(MAX).





Obrázek 3. ER Diagram

## 5. Uživatelská příručka

a) Přihlášení do systému se provádí v pravém horním rohu – odkaz „Log In“ - uživatelské jméno / heslo – jhroněk / jhroněk.

b) Odhlášení ze systému přes odkaz „Log Out“.

c) Editace menu portfolia – odkaz „Edit menu“ vpravo menu na hlavní liště.

Možné akce při editaci menu jsou (každé vytvořené menu nebo submenu automaticky vytvoří stránku, na kterou je možno přidávat obsah):

- přidávat menu na hlavní úrovni;
- přidat podmenu pod hlavní menu;
- smazat menu;
- editovat název menu;
- měnit pořadí hlavních menu;
- měnit pořadí podmenu.

d) Výběr stránky se provádí kliknutím na zvolenou položku z menu. Pokud menu obsahuje podmenu, zobrazí se po najetí myši nad hlavní menu. Pro snazší orientaci po vybrání stránky slouží odkazy v horní části stránky. Jedná se vlastně o přehled struktury menu, která patří vybrané stránce.

e) Přidání obsahu (článku, objektu) na stránku se provádí ozubeným kolečkem, které je umístěné v horní části stránky. Každá položka má svůj název – první podlouhlé políčko a potom svůj obsah. Obsahový editor se liší podle typu položky. Uložení se provádí tlačítkem submit, zrušení akce tlačítkem cancel.

Možné akce jsou:

- Add article – přidává článek, který používá HTML editor – je možné používat různé styly i nahrávat. Zde je možné, kromě stylování kliknout na ikonu “obrázek“ v horní liště HTML editoru. Kliknutím na ikonu složky vedle políčka source se otevře dialog pro nahrávání souboru a tvorby jednoduché adresářové struktury. Vybrání již nahraného souboru se provede kliknutím na link souboru a potvrzením tlačítkem OK

v dolní části dialogu. Pro tvorbu adresáře nebo nahrání obrázku je nutno kliknout na link “show manager”, kde se nachází tlačítka Create folder nebo upload file.

- Add object – přidává objekt do stránky, kde je možno vložit například objekt; zkopíruje se link na stránce youtube.com a vloží se do obsahového políčka na ePortfolio.
- Add my contacts – přidává kontaktní informace, které jsou součástí uživatelského profilu v napojeném LMS systému. Je možno zvolit, které konkrétní informace se budou prezentovat. Editor nabízí k zaškrtnutí následující volby jméno, příjmení, adresa, město, stát, email, telefon do kanceláře, telefon domů.
- Add my course – přidává do stránky informace o kurzech, které uživatel studuje / studoval. Volby jsou současné kurzy, minulé kurzy, zda zobrazit informaci o semestru, kód kurzu, název kurzu, získanou známku, datum oznámkování, či komentář instruktora.
- Add my graded items – přidává do stránky informace o oznámkovaných položkách z kurzu, které uživatel studuje / studoval (je možné vybrat kurz, typ položky z kurzu – úloha, zkouška, forum – a nastavit, co se bude přesně zobrazovat) , dostupné volby jsou: zda zobrazit informaci o semestru, kód kurzu, název kurzu, získanou známku, datum oznámkování, či komentář instruktora.
- Page properties – nastavuje, zda povolit / zakázat komentáře pro celou stránku;

f) Editace jednotlivých položek na stránce se provádí ozubeným kolečkem, které se zobrazuje přímo vedle článku, objektu atd. Možné akce jsou:

- Editovat obsah (i název);
- Smazat položku;
- Měnit oprávnění na položce;
  - Guest – položka je zobrazena komukoliv;
  - Student – položka je zobrazena jen přihlášeným studentům;
  - Employer – položka je zobrazena zaměstnavatelům (link pro otestování pohledu zaměstnavatele se nachází na pravé straně ePortfolia pod názvem „Link for employers“).
- Dále je možné pro příchozí návštěvníky:
  - Označit položku Facebookovým tlačítkem „Like“;
  - Okomentovat celou stránku.

## Závěr:

Vývoj kompletně celé aplikace od dat až po prezentační vrstvu je poměrně náročná práce, vyžadující znalosti různých technologií od SQL až po JavaScript. Dostupných zdrojů pro nastudování těchto technologií je poměrně dostatek, programátor může zvolit z nepřehledného množství přístupů a vybrat si ten nejvhodnější pro svoji aplikaci.

Během vývoje ePortfolia jsem se rozhodně přesvědčil, že některé již předpřipravené frameworky mohou ušetřit spoustu práce. Zvláště bych vyzdvihl přínos LINQ to SQL frameworku, který se dá celkem rychle naučit a jeho přínos je obrovský. Po zkušenosti s tímto frameworkem jsem se nebál v následujících projektech použít i komplexnější ORM frameworky jako je například NHibernate. Nejdůležitější poznatek, který jsem si odnesl při vývoji této aplikace je, že je potřeba zvážit dopředu v maximální míře možné funkce aplikace, aby zvolený postup, který se zdál býti dobrý při prototypování, byl vhodný i v pozdějších fázích projektu. Podle mě jsem zvolil nevhodně kombinaci ASP.NET, který je pro tento projekt zbytečně komplexní a mohl jsem použít ASP.NET MVC pro webovou část aplikace, které dobře zapadá do mé navržené vícevrstevné architektury.

Výhody vícevrstvé aplikace se ukázaly prakticky okamžitě během ladění aplikace, kdy bylo možno přesně problém izolovat a opravit.

Ze vzrůstajícího počtu vytvořených ePorfolií (v současnosti okolo 40 000) se dá usuzovat, že zadání aplikace stojí na reálné potřebě uživatelů.

## **Conclusion:**

Development of the full application, from data up to presentation layer, is relatively demanding work requiring knowledge of different technologies from SQL to JavaScript. There is a sufficient number of sources for studying these technologies and the programmer can choose from inexhaustible amount of approaches and use the most suitable one for his application.

During the development of the ePortfolio I definitively found out that some pre-prepared frameworks can save a lot of work. Especially I would like to underline the contribution of LINQ for SQL framework which can be learnt quite quickly and its contribution is huge. My experience with this framework allowed me to use the more complex ORM frameworks, such as NHibernate, in my following projects. The most important finding for me during the development of this application was that it is necessary to consider, to a maximum extent, all possible functions of application in advance so that the chosen approach, which seemed to be good in prototyping, was also suitable in later phases of the project. In my opinion, the combination ASP.NET was not chosen optimally, it seems to be unnecessarily complex for this project. For the web part of application I could use ASP.NET MVC, which well fits into my proposed multilayer architecture. In addition to that it offers solution for routing as well.

Advantages of the multilayer application showed themselves immediately during the adjusting of this application. It was possible to isolate the problem accurately and make necessary corrections.

The increasing number of created ePortfolios (currently about 40,000) shows that assignment of this application is based on a real need of users.

## Reference:

1. Young Michael C. XML Krok za krokem. Microsoft Press, 2002.
2. Groff James R., Weinberg Paul N. SQL – kompletní průvodce. Computer Press, 2005.
3. Bellinaso Marco. Webové programování v ASP.NET 2.0. Computer Press, 2007.

webové stránky:

1. <http://developers.facebook.com/docs/guides/web/>
2. <http://msdn.microsoft.com/en-us/library/ff650706>
3. <http://msdn.microsoft.com/en-us/library/bb399400.aspx>
4. <http://en.wikipedia.org/wiki/ASP.NET>

## **Příloha:**

CD – aplikace ePortfolio

## Obsah příloženého CD:

bin/

Verze webové aplikace určené pro nahrání na webový server. Obsahuje již všechny zkompilované knihovny a databázi, která se automaticky připojí k projektu po spuštění.

doc/

Pdf soubor diplomové práce.

src/

Adresář se zdrojovými soubory.

EpSln/

Adresář obsahující sln soubor pro spuštění solution ve Visual studiu. Je zde projekt EpDal datové vrstvy a EpBuss business vrstvy.

epweb/

Adresář obsahuje projekt webové aplikace

readme.txt

Soubor s instrukcemi pro instalaci.