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 Prezentační vrstva 11

4.2 Business vrstva 15

4.3 Datová vrstva 22

5. Uživatelská příručka 27

5.1 Administrátorská část 27

5.2 Uživatelská část 28

Závěr: 35

Conclusion: 36

Reference: 37

Příloha: 38

Obsah přiloženého CD: 39

Seznam obrázků:

Obrázek 1. Interface rozhraní v aplikaci…..……16
Obrázek 2. Class diagram……18
Obrázek 3. ER Diagram……..…….26

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. spagetti codu. Logika vlastní aplikace je v takovém případě prolnuta 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 vetší 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.
- Lucene.NET - .net knihovna pro fulltextove vyhledávání

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 [1], 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

[1] „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 – příchozí 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 hierachii 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[2]. Na tyto objekty lze nahlížet jako na objekty business vrstvy, ale upravené právě pro potřeby prezentační vrstvy. Tudíž 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 webservices, 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 nenavrhoval 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. Přímo 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).

[2] 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)

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 dostupném 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 zpracovaní 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 extérní 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 – standartní 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 překresleni 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 uživatelovo 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[3]. 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 přestavuje odkaz, který bude uveden vedle sdělení, že se uživateli nějaký článek líbí. Druhou možností je použití JavaScriptového

[3] 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 JavaSrcipt API, a proto je na Facebooku zaregistrována a používá jedinečný klíč, který pro každou aplikaci Facebook vygenereuje. V master page webové aplikace ePortfolia je reference na JavaSriptový 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>.

4.1.5 CSS - Použití knihovny Bootstrap

Pro kaskádové styly jsem použil knihovnu s názvem Bootstrap. Jedná se o populární knihovnu, kterou vyvinul Twitter. Tato knihovna implementuje nejenom CSS styly pro často používané HTML tagy (nadpisy, formuláře, tlačítka…), ale přináší i mnoho nových vlastních stylů pro specifické elementy, jako je například navigační lišta s menu, speciální tlačítka tzv. drop down, kdy se po kliku rozbalí nabídka s dalšími možnostmi. Velkou výhodou je, že s sebou přináší i JavaScriptovou knihovnu, která ve spolupráci s CSS knihovnou dokáže realizovat tzv. Responsive design, což znamená, že prvky se dokáží přizpůsobit rozlišení displeje a v případě nutnosti se layout stránky přeskládá pro každý displej jinak. Například lišta menu, pokud se na ní nevejdou všechny položky menu vedle sebe se změní na tlačítko, které po klinutí menu zobrazí ve vertikální poloze. Tento design je výhodný především pro zobrazeni na mobilních telefonech a tabletech.

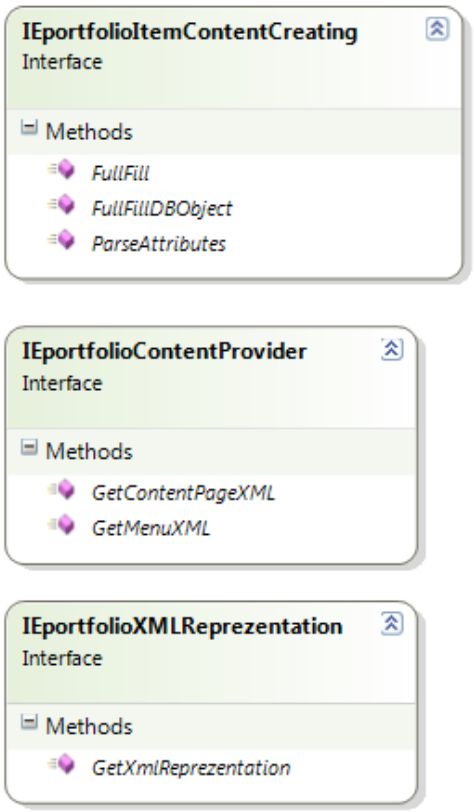
4.2 Business vrstva

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, oznámkované 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 IEportfolioXMLReprezentation pro podporu XML formátu, které musí implementovat každý objekt, který je poslán zobrazovací vrstvě.



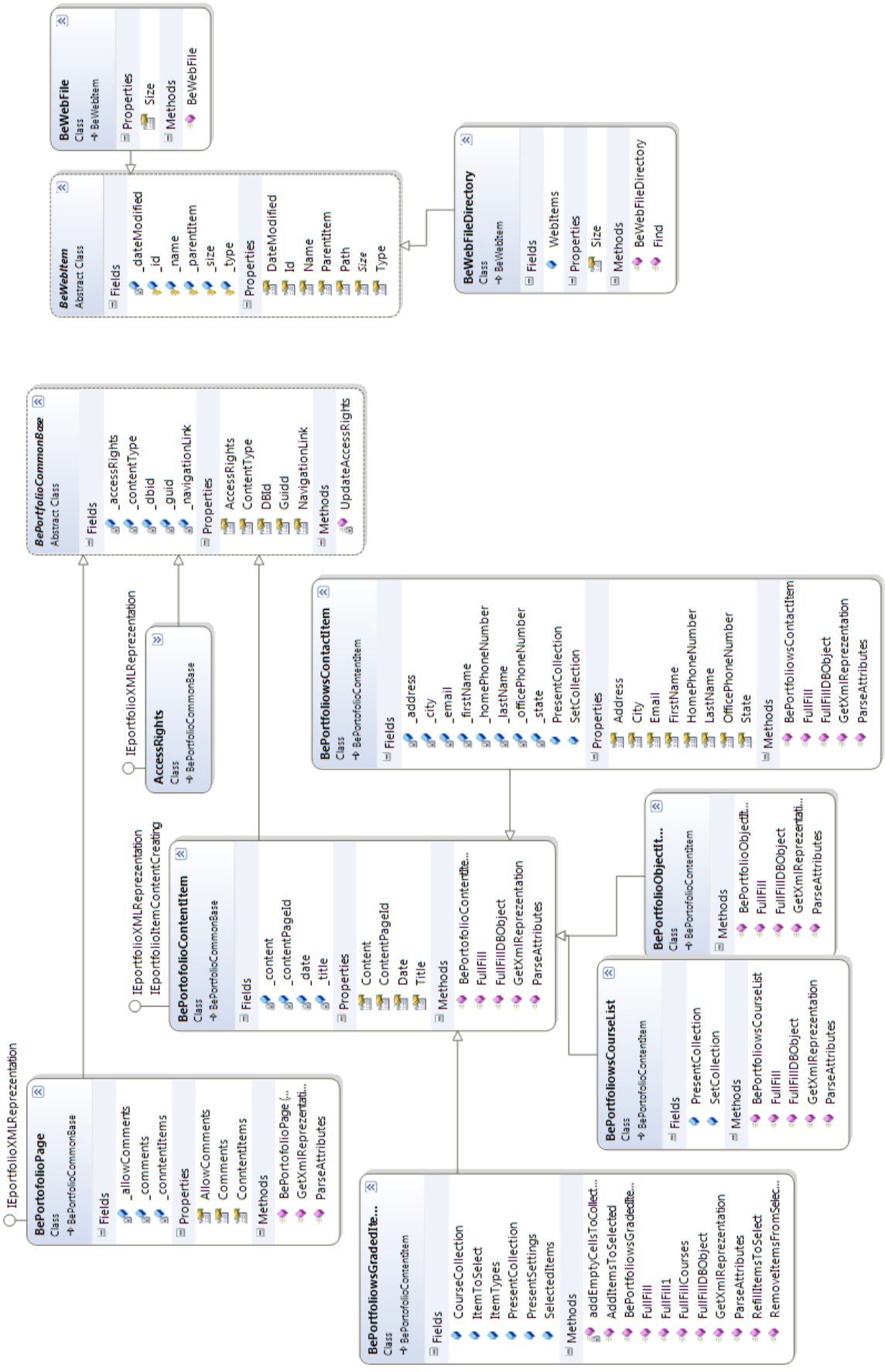
Obrázek 1. Interface rozhraní v aplikaci.

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 BePortofolioContentItem , který implementuje společné atributy pro položky na stránce jako jsou název položky, datum poslední aktualizace, příslušnost k určité stránce.

Bázová třída pro položky BePortofolioContentItem nachází využití hlavně při generování stránky ePortfolia. Hlavní cyklus pracuje s proměnnými typu BePortofolioContentItem 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 napněna 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 čí 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 IEportfolioXMLReprezentation převedou na XML a odesílají se prezentační vrstvě.

4.2.3 Spolupráce s datovou vrstvou

Hlavním rysem této spolupráce je technologie LINQ[4]. 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

[4] 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 jakýmikoliv daty, usnadňuje jejich tvorbu, třídění, jejich propojování i vyhledávání v nich. (přeloženo z www.wikipedia.com – staženo dne 23.4.2012)

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

4.2.4 Implementace vyhledávání

Pro potřeby vyhledávání je v aplikaci implementována knihovna SimpleLucene, která obaluje fulltextový vyhledávácí engine Lucene.NET. Implementace této knihovny zajišťují metody mojí třídy Indexer. Metody této třídy se volají vždy, když dojde ke změně obsahu ePortfolia a musí se změnit index, který Lucene.NET vytvořil. Aby Lucene.NET vytvořil index pro ePortfolio, bylo potřeba naprogramovat třídu EpItemIndexDefinition, která implementuje rozhraní Lucene.NET IIndexDefinition. Jedná se o generické rozhraní, které operuje nad třídou ePortfolioItem – představitelem jedné obsahové položky ePortfolia. Tudíž EpItemIndexDefinition představuje, jaké položky ePortfolioItem se mají zaindexovat a jak se s nimi má zacházet.

Aby aplikace při změně obsahu ePortfolia nemusela čekat, až se zaktualizuje index, jednotlivé požadavky na aktualizaci se ukládají do fronty, která se zpracovává asynchronně.

Pro potřeby samotného hledání nad vytvořeným indexem se používá třída EpItemResultDefinition, která implementuje generické rozhraní Lucene.NET IResultDefinition, pracující nad třídou ResDocument. Ve třídě ResDocument jsou naimplementovany položky, které nás z indexu zajímají, a tudíž třída EpItemResultDefinition dokáže transformovat obecné výsledky do ResDocumentu, který je přizpůsoben pro potřeby ePortfolia. Třída EpItemResultDefinition se uplatňuje při každém vyhledávání. Stejně tak, jak při každé změně v indexu se uplatňuje třída EpItemIndexDefinition.

4.2.5 Implementace API BigBlueButton

Jedním z požadavků na aplikaci bylo, aby dokázala spolupracovat konferenčním systémem BigBlueButton, který ve firmě celkem nově používáme a máme s ním dobré zkušenosti.

BigBlueButton beží na linuxové distribuci Ubuntu a poskytuje http rozhraní pro zakládní , připojování a zjišťování stavu konferencí. Jako výsledek volání tohoto API je často XML.

Implementace BigBlueButton API se opírá o třídu BBB. Tato třída přijímá požadavky na zakódování API volání od zobrazovací vrstvy, tento požadavek přepošle do datové vrstvy. Tam se celý požadavek validuje a vypočte se hash pomocí algoritmu MD5. Následně je výsledek vrácen do webové aplikace, která provede samotné HTTP volání. Například zda je pro dané ePortfolio konference aktivní a na základě XML výsledku od serveru BigBlueButton pomocí JavaScriptu vykreslí na stránce správné tlačítko.

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 přenese 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 konkretní 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 neumožnuje 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 zapomene se datacontext, uživatel objekt změní 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[5]: „ 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 objetky 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 bázová 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 bázových tříd do datové vrstvy bylo celkem snadné. Kódové soubory, kde je třída implementována se přidaly do projektu a bylo pořeba 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.

[5] 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 cílém navrhnout jednoduchou strukturu, která by se snadno rozšiřovala. Zvlášť 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 umožnuje 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 udržovaní 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 AlowComments.

4.3.3.3 ePortfolioItem

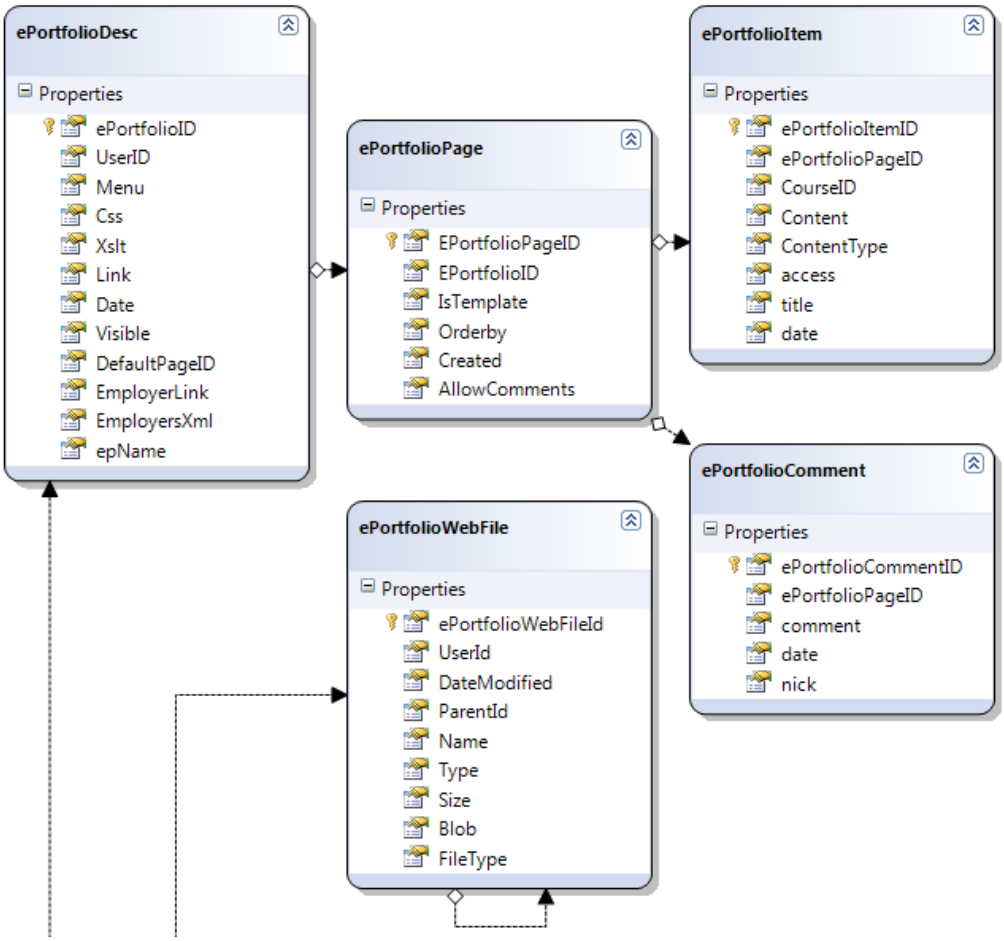
Je úložiště pro všechny položky na stránce ePortfolia, referencuje tabulku ePortfolioPage pomocí cizího klíče ePortfolioPageID. Duležitým atributem je zde sloupec ContentType a Content. První atribut popisuje o jaký typ položky se jedná. Zda je to například článek či kontaktní informace. A druhý atribut 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

5.1 Administrátorská část

5.1.1 Pravomoce

Administrátor aplikace ePortfolio je oprávněn zakládat nové uživatele (a tím pádem i nová ePortfolioa), vyhledávat a editovat existující uživatele. Je také možné uživatelům editovat jejich vlastní ePortfolia – například v situaci, kdy uživatel vystaví na svůj web nevhodný obsah.

5.1.2 Přihlášení / odhlášení do aplikace

Pro přihlášení do aplikace jako administrátor je nutné zadat do prohlížeče adresu „epweb/admin.aspx“. V pravém horním rohu stránky se nachází tlačítko „log in“. Po kliknutí na toto tlačítko se zobrazí přihlašovací dialog. Zde je možné použít předpřipravené uživatelské jméno “admin“ a heslo „admin“. Po úspěšném přihlášení se na místě tlačítka „log in“ objeví jméno a příjmení právě přihlášeného uživatele. Odhlášení probíhá klikem na uživatelovo jméno a následným výběrem položky „log out“ z rozbalovacího menu.

5.1.3 Vyhledávání a editace uživatelů

Ihned po přihlášení se objeví seznam všech existujících uživatelů – ePortfolií.
Pro procházení tohoto seznamu slouží stránkování, které se nachází na samotným seznamem. Na jedné stránce je zobrazeno maximálně 20 záznamů. Pro přechod na další stránku stačí kliknout na číslo požadované stránky ve stránkovači. Pro efektivní vyhledání uživatele slouží vyhledávací políčko „search“, které je umístěno vpravo na horní liště administrátorské stránky. Vyhledávání probíhá nad údaji uživatelské jméno, jméno uživatele a příjmení uživatele. Pro editaci uživatele slouží tlačítko „edit“, které je umístěno na konci každého řádku tabulky se seznamem uživatelů. Dialog pro editace uživatele má stejné vlastnosti jako dialog pro vytváření uživatele viz. 5.1.4.

5.1.4 Založení nového uživatele

Pro založení nového uživatele slouží tlačítko „Create a new user“. V následném dialogu administrátor vyplní přihlašovací údaje uživatele a další informace o uživateli.

Povinné údaje jsou „User name“ a „Password“. Pokud administrátor tyto povinné údaje nevyplní nebo použije již existující uživatelské jméno, systém mu nepovolí takového uživatele uložit a oznámí mu chybový stav spolu s popisem chyby pomocí výstražné červené oblasti, která se zobrazí nad dialogem. Po opravení formuláře je uživatele uložit – vytvořit.

5.1.5 Editace ePortfolia uživatele

Pro přechod na ePortfolio jakéhokoliv exitujícího uživatele slouží tlačítko „see ePortfolio“ umístěné na řádku uživatele v seznamu uživatelů. Po kliknutí se otevře příslušné ePortfolio a administrátor může provádět stejné úpravy jako samotný vlastník ePoportfolioa.

5.2 Uživatelská část

5.2.1 Přihlášení / odhlášení do aplikace

Pro přihlášení do aplikace je nutné zadat do prohlížeče adresu již existujícího ePortfolia např. „/epweb/people/pavel sanca/home.aspx” V pravém horním rohu stránky se nachází tlačítko „log in“. Po kliknutí na toto tlačítko se zobrazí přihlašovací dialog. Zde je možné použít předpřipravené uživatelské jméno “psanca“ a heslo „psanca“. Po úspěšném přihlášení se na místě tlačítka „log in“ objeví jméno a příjmení právě přihlášeného uživatele. Odhlášení probíhá klikem na uživatelovo jméno a následným výběrem položky „log out“ z rozbalovacího menu.

5.2.2 Rozložení stránky

Základní rozložení stránky pro přihlášeného uživatele se skládá:

- Z horní lišty s logem aplikace
- Menu aplikace s vyhledávacím políčkem
- Levého sloupce, který obsahuje:

- Sekci „Conference“ pro spouštění a připojování se do živých konferencí.
- Sekci „Tools“ s tlačítky pro editaci uživatelského profilu a tlačítkem pro tvorbu menu. Tato sekce je dostupná jenom pro vlastníka ePortfolia.
- Sekci „WebStudy email“ s přehledem posledních tří emailům, které se nacházejí v příchozí poště připojeného LMS systému.
- Sekci „Link for employers“ kde se nachází speciální link, který mohou použít potenciální zaměstnavatelé pro prohlížení ePortfolia.

- Pravého obsahu aktivní stránky. Zde se zobrazuje stránka podle zvolené položky menu, nebo editační dialogy.

5.2.3 Tvorba menu

Pro vytvoření obsahu ePortfolia je nejprve nutné vytvořit menu, které se zobrazuje na hlavní liště aplikace. Každá položka menu představuje jednu obsahovou stránku ePortfolia. Editace se provádí tlačítkem „Edit menu“ v pravé sekci „Tools“
Menu může mít až dvojúrovňovou strukturu.
Seznam dostupných akcí při správě menu ePortfolia:

- „Add new menu“ - přidává nové menu na nejvyšší úrovni, po kliknutí uživatel vyplní název menu a položka se přídá na konec seznamu všech již vytvořených položek menu.
- „add submenu“ – přidává nové podmenu pro již existující hlavního položku menu
- „edit“ – edituje název již exitujícího menu nebo podmenu
- „delete“ – po potvrzení konfirmačního dialogu smaže existující menu nebo podmenu
- „move up“ – přesouvá existující položku menu na hlavní úrovni nebo podmenu o úroveň výše
- „move down“ přesouvá existující položku menu na hlavní úrovni nebo podmenu o úroveň níže

5.2.4 Editace uživatelského profilu

Uživatel může změnit svůj uživatelský profil. Editace se provádí tlačítkem „Edit user“ v pravé sekci „Tools“
Dostupné editovatelné položky jsou:

- Adresa
- Město
- Stát
- Telefon do kanceláře
- Telefon domů

Needitovatelné položky jsou:

- Jméno uživatele
- Příjmení uživatele
- Email

5.2.5 Zakládání / připojování do živých konferencí

Pro „živou“ komunikaci s ostatními přihlášenými uživateli má vlastník ePortfolia možnost založit konferenci v sekci „Conference“ pomocí tlačítka „Start conference“. Po klinutí na toto tlačítko se otevře nové okno s konferencí a uživatel se automaticky stane přihlášeným moderátorem této konference.

Hlavní možnosti a nástroje ve spuštěné konferenci jsou:

- Sdílení videa a audia všech přihlášených uživatelů
- Sdílení celé nebo části obrazovky moderátora
- Veřejný a soukromý chat pro všechny přihlášené uživatele.
- Nahrát slajdy na konferenční tabuli ve formátu ppt, pptx
- Kreslení na tabuli
- „Zvednutí ruky“ přihlášeného uživatele
- Předat funkci moderátora na dalšího uživatele

Kompletní popis konferenčního softwaru BigBlueButton je dostupná na adrese: http://www.bigbluebutton.org/.

Po spuštění konference se přihlášeným návštěvníkům ePortfolia se v sekci „Conference“ zobrazí tlačítko „Join the conference“ a mohou se do konference přihlásit a aktivně se na ní podílet. Pokud žádná konference neběží přihlášeným návštěvníkům se zobrazí v sekci „Conference“ zpráva „No conference is running here“.

5.2.6 Tvorba obsahu na stránce ePortfolia.

Každá stránka ePortfolia představuje prostor, kde může uživatel vložit různé typy položek. Pro přidání obsahu na stránku slouží zelené tlačítko „Page actions“. Zde uživatel vybere, který typ obsahu si přeje vložit anebo může povolit/ zakázat komentáře návštěvníků pro danou stránku pomocí volby „Page properties“.

Každá položka stránky má svůj název, specifický obsah a oprávnění, které se uplatňuje při procházení stránky návštěvníky.
Pokud se uživatel rozhodne pro přidání položky na stránku, může zvolit z následujících možností typů položek:

- Article – představuje klasický článek, kde uživatel pomocí RTF editoru vkládá text či HTML a upravuje jeho zobrazení. Podrobné možnosti práce se RTF editorem jsou popsány v kapitole 5.2.7.
- Object – slouží jako obecný kontejner pro HTML objekty typu iframe nebo object. Typické použití je, že uživatel si například okopíruje link od youtube.com videa a vloží do ePortfolia. Je zde jistota, že editor nepřidá žádné nepatřičné HTML tagy.
- Contact – uživateli umožnuje publikovat na ePortfoliu své kontaktní informace ze svého uživatelského profilu. 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.
- My courses – protože je ePortfolio napojeno na hlavní LMS systém, má uživatel kromě svých kontaktních informací také možnost prezentovat kurzy, které v současné době studuje – volba “Current courses” nebo studoval – volba „Passed courses“. A dále může specifikovat, jak podrobné informace se o jeho kurzech zobrazí. Dostupné volby jsou: semestr, kód kurzu, název kurzu, získaná známka, datum oznámkování a či komentář instruktora.
- My graded items – pokud již uživatel v LMS absolvoval nebo absolvuje kurzy a již získal za nějakou položku kurzu (zkouška, úloha, fórum) známku, může tyto oznámkované položky kurzu vystavit na svém ePortfoliu. V editoru tohoto nástroje nejprve zvolíme kurz, ze kterého chceme čerpat pomocí rozbalovací nabídky „Select your course and type of course work...“ a následně typ položky, kterou chceme vystavit. Pokud taková položka ve zvoleném kurzu existuje, objeví se v sekci „Items

to select“ a pomocí tlačítka „Add to list“ přeneseme tuto položku do sekce vybraných položek „Selected items“. Odtud je možno položku odebrat pomocí tlačítka „Remove from list“. Po umístění položky do „Selected items“ můžeme vybrat jiný kurz a ve výběru pokračovat.

Je možno se rozhodnout, jak moc podrobné informace o kurzovních položkách se budou zobrazovat: semestr, kód kurzu, název kurzu, získaná známka, datum oznámkování a komentář instruktora.

Všechny vytvořené položky se zobrazí na stránce společné s datem vytvoření a také tlačítkem „edit“, které má následující volby:

- „edit item“ – pro editaci obsahu a názvu položky – spustí se příslušný editor podle typu položky
- „delete item“ – zobrazí konfirmační dialog pro smazání položky
- „access right“ – nastavuje pro které typy návštěvníků bude položka zobrazena – rozlišuje se mezi přihlášeným uživatelem „student“, nepřihlášeným návštěvníkem „guest“ a potencionálním zaměstnavatelem „employer“

5.2.7 Použití RTF editoru

RTF editor slouží k zadávání textu a jeho stylovaní, ale také k vkládání objektů typu video/audio, obrázek nebo jakýkoliv soubor , který již uživatel má na svém disku a po nahrátí do aplikace, může vystavit na webu.

RTF editor má dvě hlavní záložky „home“ a „objects“

Na záložce „home“ se kromě dobře známých tlačítek pro formátování textu vyskytují:

- Fullscreen – přepíná editor do celoobrazovkového režimu pro snadnější editaci
- Hyperlink – umožňuje vytvořit odkaz na webovou stránku nebo na nahraný soubor do aplikace (viz práce se soubory)
- Image – umožňuje vložit obrázek, jehož zdroj je na internetu nebo se jedná o nahraný soubor do aplikace (viz práce se soubory)
- View/edit source – otevře okno s editorem, kde uživatel vidí HTML zdrojový text

Na záložce „object“ jsou tyto tlačítka:

- Media – umožňuje uživateli vybrat nahraný multimediální soubor, který se následně bude přehrávat ve vestavěném přehrávači přímo na stránce ePortfolia – podpora formátů wmv, avi.
- Flash – umožňuje uživateli vybrat nahraný flashovy soubor ve formátu swf, který se následně bude přehrávat ve flashovém přehrávači přímo na stránce ePortfolia
- Line – vloží dělící čáru do RTF boxu
- Special characters – nabízí možnost vložení speciálních znaků

5.2.7.1 Práce se soubory

Při volbě „Media“, „Flash“, „Image“, „Hyperlink“ se editor v dialogu zeptá, kde se nachází zdroj elementu, který uživatel hodlá vytvořit. Ve všech těchto dialozích se nachází ikona složky, která otevírá okno file manageru, kde má uživatel možnost pracovat se soubory.

Ve file manageru jsou dostupné následující možnosti:

- Vybrat již existující soubor pomocí kliknutí na název soubor nebo přejít do jiné složky kliknutí na název složky. Při vybrání souboru kliknutím se odkaz na soubor přenese do dolního políčka „source“ a tlačítkem OK lze potvrdit výběr tohoto souboru jako zdroj pro právě tvořený element.
- Vymazat celou složku odkazem „delete“ vedle názvu složky.
- Vybrat soubor zatržením zatržítka a po následném kliku na odkaz „show manager“ se objeví nabídka s možností vymazání jednoho nebo vice souboru – „Delete selected file“.
- Vytvořit složku v aktuálním umístění – po kliknutí na odkaz „show manager“ stačí zadat název složky do políčka a stisknout tlačítko „create folder“.
- Nahrát soubor do aktuálního umístění – po kliknutí na odkaz „show manager“ a pomocí tlačítka „vybrat soubor“ uživatele vybere soubor na svém disku volbou „Upload file“ soubor nahraje do aplikace do aktuálního umístění.

5.2.8 Vyhledávání obsahu

Pro vyhledání článku na svém nebo cizím ePortfoliu je k dispozici políčko „search“ na hlavní liště. Vyhledávání probíhá nad všemi ePortfolii a porovnává vložený text s názvem a obsahem všech článků. Je možné zadávat i složitější výrazy do vyhledávácího políčka:

- Výrazy AND, NOT, OR
- Wildcards *, ?
- title: hledaný výraz – prohledává jenom názvy článků
- content: hledaný výraz – prohledává jenom obsah článků

Výsledky vyhledávání jsou zobrazeny ve formátu:

- Vlastník ePortfolia
- Název článku
- Prvních 300 znaků obsahu – (HTML tagy jsou odstraněny)
- Datum vytvoření článku

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 nepřeberného 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 neccessary 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
5. http://twitter.github.com/bootstrap/
6. http://simplelucene.codeplex.com/

Příloha:

CD – aplikace ePortfolio

Obsah přiloženého 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.