VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

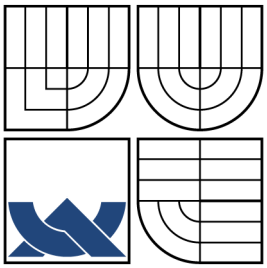
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

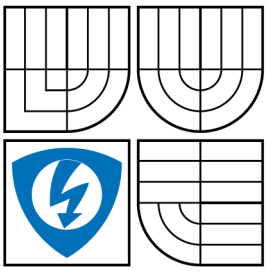
VYUŽITÍ NÁVRHOVÉHO VZORU MODEL-VIEW-CONTROLLER VE WEBOVÝCH APLIKACÍCH

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE LADISLAV MARGAI
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ NÁVRHOVÉHO VZORU MODEL-VIEW-CONTROLLER VE WEBOVÝCH APLIKACÍCH

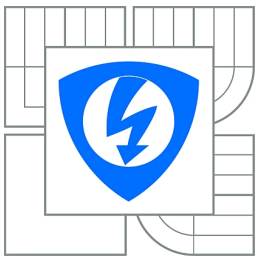
USAGE OF SOFTWARE PATTERN MODEL-VIEW-CONTROLLER IN WEB APPLICATIONS

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE — LADISLAV MARGAI
AUTHOR

VEDÚCI PRÁCE — doc. Ing. IVO LATTENBERG, Ph.D.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Ladislav Margai	***ID:***	134551
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Využití návrhového vzoru Model-View-Controller ve webových aplikacích

POKYNY PRO VYPRACOVÁNÍ:

S využitím programovacího jazyka C#, vývojového prostředí Microsoft Visual Studio 2012 a technologie Model-View-Controller realizujte webovou aplikaci představující rezervační systém pro rezervaci učeben a laboratoří. Uvažujte několik rolí uživatelů, jednorázové časové rezervace (zkoušky) i pravidelně se opakující rezervace po určitou dobu (výuka v semestru). Aplikaci řádně okomentujte a pokuste se ji rozdělit do jednotlivých etap, ze kterých bude zřejmý postup při návrhu a realizaci.

DOPORUČENÁ LITERATURA:

[1] SHARP, J. Microsoft Visual C# 2010, Nakladatelství Computer Press, a.s. 2010, 696 s., ISBN 978-80-251-3147-3
[2] MACDONALD, M., FREEMAN, A., SZPUSZTA, A. ASP.NET 4 a C# 2010 - KNIHA 1 - tvorba dynamických stránek profesionálně, Zoner press, 2011, 880 s., ISBN 978-80-7413-131-8
[3] MACDONALD, M., FREEMAN, A., SZPUSZTA, A. ASP.NET 4 a C# 2010 - KNIHA 2 - tvorba dynamických stránek profesionálně, Zoner press, 2011, 704 s., ISBN 978-80-7413-145-5

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: doc. Ing. Ivo Lattenberg, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Obsah práce je zameraný na využitie návrhového vzoru Model-View-Controller vo webových aplikáciach. Práca je rozdelená do niekoľkých kapitol. V prvej kapitole je vysvetlený model MVC ako taký a nutné úpravy interpretácie pre použitie vo webových aplikáciach. V druhej kapitole je popísaný vývoj od konceptu webových formulárov až po MVC a porovnanie týchto dvoch technológií. Tretia kapitola sa zameriava na prácu s Razor engine. Posledné tri kapitoly popisujú samostatný vývoj aplikácie.

KĽÚČOVÉ SLOVÁ

MVC, Model, Zobrazenie, Radič, Webové formuláre, C#, Visual Studio, Razor engine, ASP.NET, ASP, HTTP, AJAX, jQuerry, HTML, xHTML, Entity Framework

ABSTRACT

Content of this thesis is focused on usage of design pattern Model-View-Controller in web applications. The thesis is divided into several chapters. In first chapter is explained how MVC model works and necessary edits for usage in web applications. In second chapter is described development from web forms to MVC and comparison of these two concepts. Third chapter is focused on work with Razor engine. Last three chapters describe development of the application.

KEYWORDS

MVC, Model, View, Controller, Web Forms, C#, Visual Studio, Razor engine, ASP.NET, ASP, HTTP, AJAX, jQuerry, HTML, xHTML, Entity Framework

MARGAI, Ladislav *Využití návrhového vzoru Model-View-Controller ve webových aplikacích*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 46 s. Vedúci práce bol doc. Ing. Ivo Lattenberg, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Využití návrhového vzoru Model-View-Controller ve webových aplikacích“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu semestrálnej práce pánovi doc. Ing. Ivovi Lattenbergovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Táto práca sa venuje možnostiam použitia návrhového vzoru Model-View-Controller vo webových aplikáciach. Zameriava sa na praktické využitie a porovnanie so staršou architektúrou webových formulárov (web forms). Keďže MVC model je pomerne komplexný, predpokladá sa, že čitateľ má základné vedomosti o frameworku ASP.NET, ktorý je tu vysvetlený pomerne stručne. Samostatná práca je rozdelená na teoretický rozbor, a popis vyhotovenia aplikácie. Kód samostatnej aplikácie bol písaný podľa štandardov a doporučení [11] a [12], preto sú všetky premenné, názvy funkcií a komentáre v angličtine.

Jednotlivé kapitoly práce:

1. **NÁVRHOVÝ VZOR MVC** – Vysvetlenie návrhového vzoru MVC vo všeobecnej rovine a interpretácia návrhového vzoru pre použitie vo webových aplikáciach.
2. **VÝVOJ ASP.NET MVC** – Postupný vývoj webových formulárov, ich nevýhody, dôvody prečo vznikol ASP.NET MVC framework a ich vzájomné porovnanie.
3. **RAZOR ENGINE** – Nástupca ASPX enginu, popis používania a implementácie jednotlivých Zobrazení (View) pomocou tejto technológie.
4. **UI NÁVRH** – Popis základného užívateľského rozhrania aplikácie.
5. **VYTVORENIE DÁTOVÉHO MODELU** – Vnútorná logika a štruktúra dátového modelu.
6. **WEB APLIKÁCIA** – Vyhotovenie samostatnej web aplikácie.
7. **ROZŠÍRITEĽNOSŤ** – Možnosti rozšírenia aktuálneho riešenia.

Aplikácia dostupná online na `http://bakalarka.datagrid.sk`, prihlasovacie údaje jednotlivých persón sú uvedené v sekcii UI návrh 4.1.

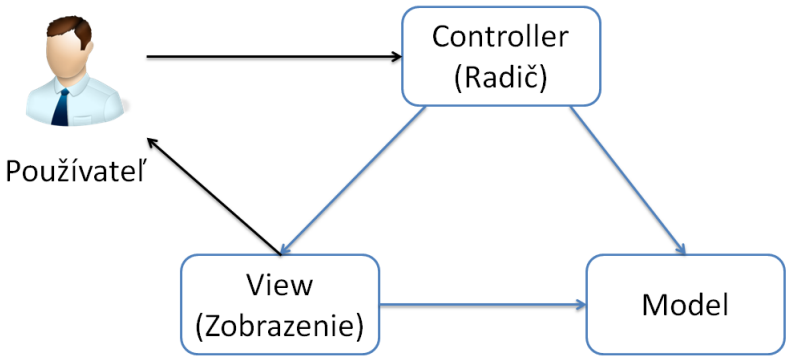
1 NÁVRHOVÝ VZOR MVC

MVC je skratka zložená z troch začiatočných písmen slov **M**odel-**V**iew-**C**ontroller, v preklade je to Model-Zobrazenie-Radič. Je to softvérová architektúra, ktorá rozdeľuje dátový model aplikácie, užívateľské rozhranie a riadiacu logiku do troch nezávislých komponentov, tak, že modifikácia ktorejkoľvek z nich má minimálny vplyv na ostatné.

V blokovej schéme (Obr. 1.1) je zobrazený základný tok udalostí v aplikácií:

1. Užívateľ vykoná nejakú akciu na užívateľskom rozhraní – Zobrazenie (View)
2. Táto akcia je zachytená Radičom (Controller)
3. Radič (Controller) rozhodne, ako na akciu reagovať, a typicky zmení nejaké hodnoty v Modeli, alebo priamo ovplyvní Zobrazenie (View)
4. Zobrazenie (View) zobrazí zmeny užívateľovi.

Tento kolobeh sa potom opakuje. Na schéme je ešte vidno, že Radič (Controller) obsahuje väzbu na Zobrazenie (View), čo je v MVC frameworkoch dosť typické, ale táto väzba je menej dôležitá ako schopnosť Radiča (Controller) upraviť Model. Vzťah Radiča (Controller) a Zobrazenia (View) je zvyčajne taký, že Radič (Controller) len rozhoduje ktoré Zobrazenie (View) sa má zobraziť.



Obr. 1.1: Bloková schéma návrhového vzoru MVC

1.1 MVC v prostredí webu

S MVC sa dnes stretneme najčastejšie vo webových technológiách. Pretože ale takmer všetky prezentačné vzory zišli z desktopových technológií, je potrebné niektoré veci upraviť.

V prostredí webu vyzerajú jednotlivé komponenty následovne:

- **Model** je v tomto prípade najjednoduchšia časť, pretože je identický s Modelom v desktopových technológiách. Obsahuje dáta a biznis logiku, s konkrétnou prezentáciu nemá nič spoločné.
- **Zobrazenie (View)** je serverový kód, ktorý sa stará o generovanie HTML, avšak ani u webovej aplikácií nie je striktne dané, že sa musí jednať práve o HTML. Ako výstup môže byť aj XML alebo JSON.
- **Radič (Controller)** sa v prostredí webu najčastejšie skladá z dvoch hlavných častí. Prvá je tzv. `Front Controller`, ktorý zachytáva všetky HTTP požiadavky a tie následne spracuje a prepošle konkrétnym Radičom (Controller) – čo je tá druhá časť. Konkrétny Radič (Controller) potom typicky príjme dáta pôvodne pochádzajúce z HTTP požiadavky, uloží ich do Modelu a ten ich previaže s konkrétnym Zobrazením (View), ktoré sa už vie o vyrenderovanie HTML alebo iného formátu postarať.

2 VÝVOJ ASP.NET MVC

Prvé interaktívne webové aplikácie, ktoré vznikli na Microsoft platforme boli známe ako CGI(Common gateway Interface) skripty, ktoré boli zodpovedné za vytváranie HTTP výstupov. Šablóny na formátovanie obsahu sa vytvárali nesystematicky podľa potreby, čo malo za následok, že programy sa veľmi ťažko a zle písali, ladili, opravovali alebo aj testovali.

Ku koncu 90-tych rokov 20. storočia Microsoft predstavil HTX (HTML Extension Template) šablóny a IDC (Internet Database Connector) konektory. Krátko na to bola uvedená aj Active Server Pages technológia známa pod skratkou ASP. ASP prinieslo do sveta webových technológií šablónovanie – stránka na serveri bol mix HTML syntaxe a dynamického skriptovania. Bol to obrovský krok do predu z CGI skriptov, ale postupom času ako vznikali rôzne rozsiahle aplikácie, stávalo sa ASP veľmi neprehľadné.

Začiatkom roku 2002 Microsoft uvoľnil prvú verziu frameworku .NET, ktorá úplne zmenila vývoj webových aplikácii pod platformou windows.

2.1 ASP.NET Web Forms

Keď spoločnosť Microsoft vydala ASP.NET 1.0 dokonca ani oni netušili, ako nadšene bude táto technológia prijatá. Rýchlo sa stala štandardom pre vývoj webových aplikácií s technológiami od Microsoftu. Postupom času sa vyvíjala až do dnešnej podoby.

2.1.1 ASP.NET 1.0 a 1.1

Kľúčová myšlienka spočívala v návrhovom modeli zvanom webové formuláre . Je to abstrakcia, ktorá modeluje stránku ako kombináciu objektov. Keď klient požiada konkrétnu stránku, ASP.NET najprv vytvorí inštanciu objektu stránky a potom vytvorí objekty pre všetky ovládacie prvky ASP.NET, ktoré sa nachádzajú vo vnútri stránky. Celé jej riadenie prejde postupnosťou udalostí jej životného cyklu – potom sa jej spracovanie skončí – realizuje sa finálne HTML a následne sa uvoľní z pamäti. Ťažiskom programovania v ASP.NET je zaplniť to, čo sa má diať medzi tým.

2.1.2 ASP.NET 2.0

Nová verzia si ponechala to isté jadro abstrakcie (model webových formulárov) a koncentrovala sa na pridávanie nových vysokoúrovňových funkcií ako mater pages

(vzory stránok), themes (motívy), navigation controls (nástroje na vytváranie navigácie napr. stromová štruktúra), profiles & credentials (bezpečnosť a členstvo), web parts (portálový framework), ovládacie prvky pre zdroje dát a mnoho iných.

2.1.3 ASP.NET 3.5

V roku 2007 bola nová ďalšia nová verzia ASP.NET. Keďže vyšla spolu až .NET frameworkom 3.5, to vysvetľuje, prečo neexistuje ASP.NET 3.0. V porovnaní s verziu 2.0 sa jedná skôr o postupnú evolúciu. Zmeny by sa dali zhrnúť do dvoch oblastí: implementácia LINQ (Language Integrated Query) a AJAX (Asynchronous JavaScript and XML).

V tomto istom roku predstavil Microsoft prvú verziu ASP.NET MVC. Obsahovala pomerne jednoduchý HTML helper a ako Zobrazenie (View) používala ASPX engine.

2.1.4 ASP.NET 4.0

Nová verzia naďalej pokračovala v rovnakom duchu – integrovala do seba viaceré vylepšenia, medzi tie najvýznamnejšie patrili: konzistencia XHTML, automatická detekcia prehliadača, komprimovanie stavu relácie, stav zobrazenia na vyžiadanie (opt-in view state), rozšírená možnosť cachovania a smerovanie (routing).

Medzičasom bol uverejnený ASP.NET MVC2 framework – prepracovanejšie Helpery (HTML Helper, Form Helper), asynchrónne kontroléry, avšak stále používal ASPX engine. S príchodom ASP.NET MVC3 boli implementované do frameworku viaceré zmeny, medzi najpodstatnejšie patria: Razor view engine, podpora validácie na strane klienta cez knižnice jQuery a JSON Model binding.

2.1.5 ASP.NET 4.5

Aktuálna verzia ASP.NET vyšla len prednedávnom spolu s Visual Studiom 2012. Priniesla podporu HTML5, zlepšenú validáciu a model binding (vychádza s implementácie ovládacieho prvku ObjectDataSource).

Spolu s inštaláciu Visual Studia 2012 sa automaticky nainštaluje aj ASP.NET MVC4, kde bola zlepšená podpora Windows Azure a dodané knižnice pre návrh aplikácií pre mobilné zariadenia.

2.2 ASP.NET MVC

ASP.NET MVC framework ponúka úplne odlišný spôsob budovania webových aplikácií. Je uvedený ako alternatíva k štandardným modelom webových formulárov.

Avšak keďže je MVC založené na ASP.NET, pre správne porozumenie ukážok treba mať túto technológiu aspoň z časti zvládnutú.

Implementácia návrhového vzoru MVC v podaní ASP.NET:

- **Model** odkazuje na datový model, inými slovami obchodný kód špecifický pre danú aplikáciu – napríklad logika k prístupu k dátam a overovacie pravidlá, čo sa dá pochopiť aj tak, že vykonáva základné operácie nad dátami aplikácie ako čítanie, aktualizácia a odstraňovanie. Často sa používa na to skratka CRUD, ktorá je zostavená zo začiatočných písmen slov **C**reate-**R**ead-**U**pdate-**D**elete.
- **Zobrazenie (View)** je to, čo sa prezentuje užívateľovi. Zvyčajne je to HTML výstup ktorý generuje buď ASPX engine (`.aspx` súbor) alebo Razor engine (`.cshtml` súbor). Obsah stránky sa obvykle vzťahuje k operáciám CRUD, ktoré sa bude snažiť užívateľ vykonávať, takže bude obsahovať podrobnosti jedného alebo viacerých dátových prvkov, alebo prostriedky k editovaniu či odstraňovaniu dátových prvkov.
- **Radiče (Controllers)** sú súbory `C#`, ktoré tvoria mosty medzi jednotlivými zobrazeniami a modelmi. Kontrolér obdrží od klienta nejakú požiadavku a vyberie Zobrazenie (View), ktoré má obslúžiť túto požiadavku.

2.3 MVC vs. Web Forms

Vzor MVC zatlačuje do ústrania niektoré tradičné prvky ASP.NET ako sú webové formuláre , webové ovládacie prvky (GridView, ListView,. . .), stav zobrazenia (view state), odosielanie späť na server (postbacks) a stav relácie (session state). Dôsledkom toho je fakt, že vývojári sú nútení prijať nový spôsob premýšľania a akceptovali dočasný pokles produktivity. MVC vzor je čistejší a pre web vhodnejší.

MVC má od webových formulárov veľa výhod. Základná myšlienka spočíva v tom, že sú Model, Zobrazenie (View) a Radič (Controller) od seba oddelené. To má za následok, že aplikácia je prehľadná a ľahko sa testuje. Taktiež má vývojár pod kontrolou celý proces generovania HTTP výstupu a môže doň kedykoľvek zasiahnuť. Dôležitým aspektom je aj fakt, že nedochádza prenášaniu obrovského objemu stavových dát (view state), čo u webových formulároch patrí k najväčším slabinám tejto technológie.

Tri základné piliere, prečo vývojári odchádzajú k MVC:

- **Vývoj riadený testami** – Vďaka tomu, že všetky časti aplikácie sú od seba čisto oddelené, je ľahké vytvoriť unit testy, ktoré preveria, či fungujú správne. S webovými formulármi je automatizované testovanie pracné a veľmi často aj nemožné.

- **Kontrola nad značkovaním HTML** – S webovými formulármi sa programuje s bohatými sadami objektami, ktoré majú na starosť správu a realizáciu HTML. S ASP.NET MVC je obsah stránok vkladaný podobne ako pri viazaní dát. Znamená to, že návrh zložito naformátovaných stránok môže dať viacej práce, no napriek tejto nevýhode vývojár získa úplnú kontrolu nad všetkými detailami značkovania. Táto kontrola je veľmi užitočná, pokiaľ vývojár plánuje písať JavaScript na strane klienta, alebo používať javascriptové knižnice tretích strán ako jQuery. Na druhej strane, vyžaduje to značnú znalosť HTML a JavaScriptu, takže pokiaľ tieto znalosti chýbajú, je lepšie zostať pri webových formulároch , ktoré to vygenerujú samy.
- **Kontrola nad URL** – Aj keď webové formuláre dávajú vývojárom možnosť smerovania URL (URL routing, od .NET frameworku 4.0), ASP.NET MVC má túto funkcionalitu a nástroje priamo integrované v sebe.

Na druhej strane webové formuláre sú veľmi silná platforma. Pokiaľ vývojár dáva prednosť čo najrýchlejšiemu návrhu aplikácie, vysokoúrovňovému modelu, ktorý bude riadiť stav aplikácie za neho a rád využíva bohaté knižnice ovládacích prvkov, ktoré umožňujú vytvárať rýchlo a jednoducho rozsiahle aplikácie (známe aj pod skratkou RAD – **R**apid **A**pplication **D**evelopment), budú webové formuláre lepšia alternatíva.

Očakáva sa, že v najbližších pár rokov webové formuláre ostanú naďalej dominantné v intranetových a backend aplikáciach, zatiaľ čo internetové frontend aplikácie budú smerovať k MVC.

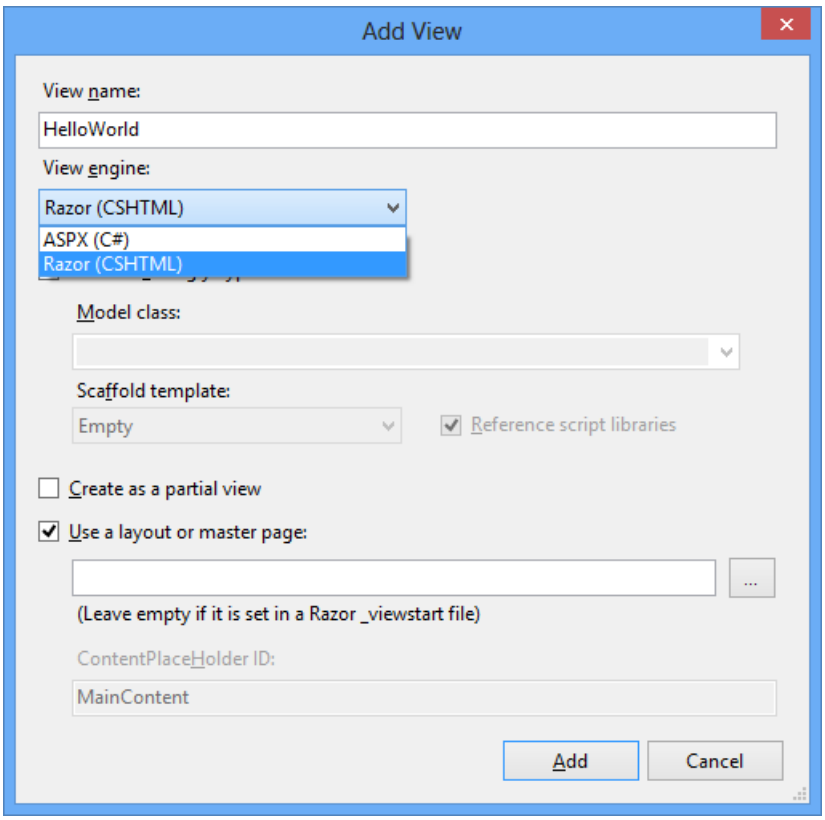
3 RAZOR ENGINE

Bol predstavený ako jedna z hlavných noviniek ASP.NET MVC3. Jedná sa o ASP.NET syntaktický analyzátor používaný na vytváranie dynamických Zobrazení (Views), ktorý čiastočne vychádza z predchádzajúcej architektúry ASPX enginu.

Výhody, prečo používať Razor engine sa dajú zhrnúť do troch bodov:

- Nie je to nový jazyk (Vychádza zo syntaxe `C#` alebo `VB`)
- Podporuje IntelliSence (Automatické doplňanie príkazov)
- Ľahko testovateľný
- Podpora „Layouts“

Súbory, ktoré spracováva Razor engine majú príponu `.cshtml` (ak by bol projekt písaný vo Visual Basicu, mali by príponu `.vbhtml`) a sú umiestnené v zložke `Views`, kedže výstup Razor enginu je zvyčajne HTML, a to má na starosti Zobrazovacia (View) vrstva modelu MVC. Pri pridávaní nového Zobrazenia (View) do projektu sa Visual Studio vždy opýta, aký syntaktický analyzátor má použiť – ASPX engine alebo Razor engine (Obr. 3.1).



Obr. 3.1: Pridávanie nového Zobrazenia (View)

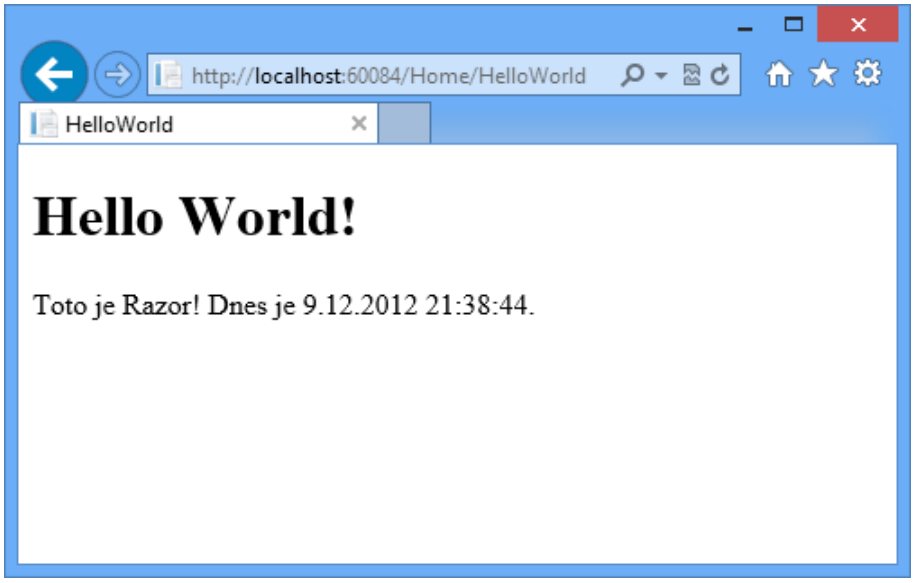
3.1 Hello Razor!

Prvá ukážka práce s Razor Enginom bude klasická – `Hello world`, v preklade `Ahoj svet`. Slúži pre predstavu základnej štruktúry a syntaxu kódu.

Ako je vidieť z následujúcej ukážky, pre označenie začiatku kódu pre Razor sa používa znak `@`.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>HelloWorld</title>
</head>
<body>
    <div>
        <h1>Hello World!</h1>
        Toto je Razor! Dnes je @DateTime.Now.ToString().
    </div>
</body>
</html>
```

Dané Zobrazenie (View) vypíše aktuálny dátum, použije na to systémovú knižnicu `mscorlib.dll`.



Obr. 3.2: Zobrazenie (View) HelloWorld.cshtml

3.2 Podmienky a cykly

Tak ako ASPX engine, tak aj Razor engine podporuje podmienky a cykly. Nie je potrebné určovať kde končí syntax `C#` kódu a kde začína HTML, Razor to spraví za vývojára.

V následujúcej ukážke bude použitý ako Model trieda `Process`, ktorý sa dostane do Zobrazenia (View) importovaním knižnice `System.dll` pomocou príkazu `@using System.Diagnostics`.

```
@using System.Diagnostics;
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Podmienky a Cykly</title>
</head>
<body>
    <div>
    @if (DateTime.Now.DayOfWeek == DayOfWeek.Saturday ||
     DateTime.Now.DayOfWeek == DayOfWeek.Sunday)
    {
      <h1>Dnes nemusím ísť do práce!</h1>
    }
    else
    {
        <h1>Achjaj, zase práca!</h1>
    }

    @foreach (Process process in Process.GetProcesses())
    {
        <div>@process.ProcessName</div>
    }
    </div>
</body>
</html>
```

Dané Zobrazenie (View) porovná aktuálny deň v týždni, a ak je sobota alebo nedeľa, vypíše hlášku, ak je pracovný deň vypíše hlášku inú. Na záver cyklus `foreach` prejde všetky prvky poľa, ktoré vracia statická metóda `Process.GetProcesses()` a každý zvlásť vypíše do HTML elementu `div`. Jeden z možných výstupov je vidieť na Obr. 3.3.

Obr. 3.3: Zobrazenie (View) PodmienkyCykly.cshtml

3.3 Viacriadkové zápisy, rozpoznávanie obsahu

Občas je potrebné vykonať celý blok kódu v danom Zobrazení (View). Razor engine má túto funkcionalitu zabudovanú, stačí dodržať následovnú syntax:

```
@{
  blok kódu
}
```

Ukážka viacriadkového zápisu:

```
@{
  int cislo = 1;
  string sprava = "Číslo je " + cislo;
}
<p>Vaša správa: @sprava</p>
```

Pri analyzovaní Zobrazenia (View) Razor engine pomerne presne vie, kedy sa jedná o `C#` príkazy a kedy len o prostý text. Rozlišuje to len na základe HTML tagov. V prípade, že sa jedná o znak `@`, pozrie sa čo sa nachádza pred a za týmto znakom a na základe toho určí ďalšie spracovanie.

V prípade, že by Razor zle interpretoval vývojárov kód, dá sa to manuálne vynútiť pomocou jednoduchých príkazov.

Prvý scenár – pokiaľ vývojár jasne nedefinuje čo je HTML a čo C# príkazy, môže nastať problém.

```
@if (DateTime.Now.Year == 2012)
{
  Chybný zápis, vypíše error.
}
```

Oprava je jednoduchá – stačí daný text uzatvoriť medzi dva ľubovolné HTML tagy. Pokiaľ však z určitých dôvodov nechce mať žiadny HTML tag na výstupe, môžu sa použiť interné tagy <text> a </text>, ktoré budú z výstupu odstránené – slúžia len na oddelenie C# príkazov a obyčajného textu.

```
@if (DateTime.Now.Year == 2012)
{
  <text>
    Opravené, nevypíše error.
  </text>
}
```

Druhý scenár – Razor automaticky rozlišuje v prípade znaku @ či sa jedná o e-mailovú adresu alebo C# príkaz. Môže sa stať, ako je v následujúcej ukážke, že sa rozhodne nesprávne.

```
@if (User.IsInRole("Administrators"))
{
  <strong>Prednastavené heslo do účtu: @ah0j$</strong>
}
```

Oprava je jednoduchá stačí použiť zdvojený zápis @@.

```
@if (User.IsInRole("Administrators"))
{
  <strong>Prednastavené heslo do účtu: @@ah0j$</strong>
}
```

3.4 Použitie Layouts/Master Pages

Pri uvedení ASP.NET 2.0 bol predstavený nový koncept tvorenia obsahu – Master Pages (Vzory Stránok). Pri vytváraní Zobrazenia (View) cez ASPX engine sa tento koncept používa doteraz, pretože vývojarovi umožnuje:

- Definovať časť stránok samostatne a opätovne ich využívať na viacerých stránkach súčasne.

- Vytvoriť uzamknuté rozloženie špecifikujúce konkrétne regióny, ktoré bude možné editovať. Stránky, ktoré budú využívať šablónu tohto druhu, budú môcť pridávať (alebo modifikovať) obsah len v príslušných regiónoch.
- Povoliť prispôsobovanie prvkov, ktoré sa opätovne využívajú na stránkach.
- Zviazať webovú stránku deklaratívne (t.j. bez kódu), či programátorsky pri behu.
- Navrhovať webovú stránku , ktorá používa šablónu stránky, v nejakom návrhárskom nástroji ako je Visual Studio.

Razor Engine používa na vytváranie rozloženia stránok `Layouts` – vychádza z konceptu `Vzorov Stránok (Master Pages)`.

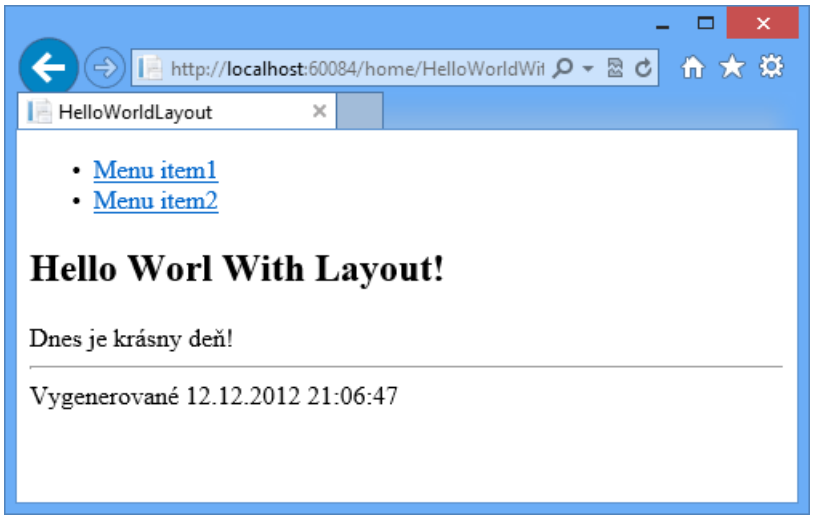
V následujúcej ukážke je Layout stránky `HelloWorldLayout.cshtml`. Obsahuje statický obsah a dve metódy HTML helpera `@RenderBody()` a `@RenderSection( string name, bool required)` – obidve majú na starosti dynamicky napĺňať špecifický obsah na základe požadovanej URL adresy. Rozdiel je v tom, že metóda `@RenderSection(string name, bool required)` je v tomto prípade nepovinná a nemusíme ju definovať.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>HelloWorldLayout</title>
</head>
<body>
  <div>
    <ul>
      <li><a href="#">Menu item1</a></li>
      <li><a href="#">Menu item2</a></li>
    </ul>
  </div>
  <div>
    @RenderBody()
  </div>
  <hr />
    @RenderSection("footer", false)
</body>
</html>
```

Súbor `HelloWorldWithLayout.cshtml` dedí z rodiča celý layout. Telo tohto súboru sa berie ako implementácia metódy `@RenderBody()`. Pokiaľ je potrebné implementovať jednotlivé sekcie, je potrebné to definovať pomocou príkazu `@section`.

```
@{
  ViewBag.Title = "Hello World With Layout";
  Layout = "~/Views/Home/HelloWorldLayout.cshtml";
}

<h2>Hello Worl With Layout!</h2>
Dnes je krásny deň!

@section footer
{
  Vygenerované @DateTime.Now
}
```

Výstup zobrazenia je možný vidieť na Obr. 3.4.



Obr. 3.4: Zobrazenie (View) HelloWorldLayout.cshtml

4 UI NÁVRH

Pre správne fungovanie ktorejkoľvek aplikácie, je potrebné navrhnúť UI (User Inteface – užívateľské rozhranie), ktoré by malo dodržiavať zásady správneho UX (User Experience – zameranie na potreby užívateľa).

4.1 Persóny

Persóny z pohľadu UX sú fiktívne osoby, ktoré predstavujú finálnych zákazníkov. Každá je špecifická určitými parametrami: vek, pohlavie, vzdelanie, záľuby...
Tieto persóny v aplikácií zvyčajne ostávajú aj po vydaní – budúci zákazník sa potom vie ľahko vžiť do týchto osôb a ľahko zistí čo všetko mu daná aplikácia ponúka.

Rezervačný systém obsahuje tieto persóny:

- `Ján Vysoký` je mladý 32-ročný muž, absolvent školy technického smeru, pracuje na polovičný úväzok ako správca systému.
- `Alfréd Neumayer` sa volá 58-ročný profesor fyziky, ktorý učí na škole už 28 rokov, získal niekoľko významných ocenení za svoje objavy.
- `Robert Šamaj` študuje obor biomedicínu, je v druhom ročníku a má 20 rokov.

Meno	**Login**	**Heslo**
Robert Šamaj	student	student
Alfréd Neumayer	ucitel	ucitel
Ján Vysoký	admin	admin

Tab. 4.1: Prihlasovacie údaje pre jednotlivé persóny

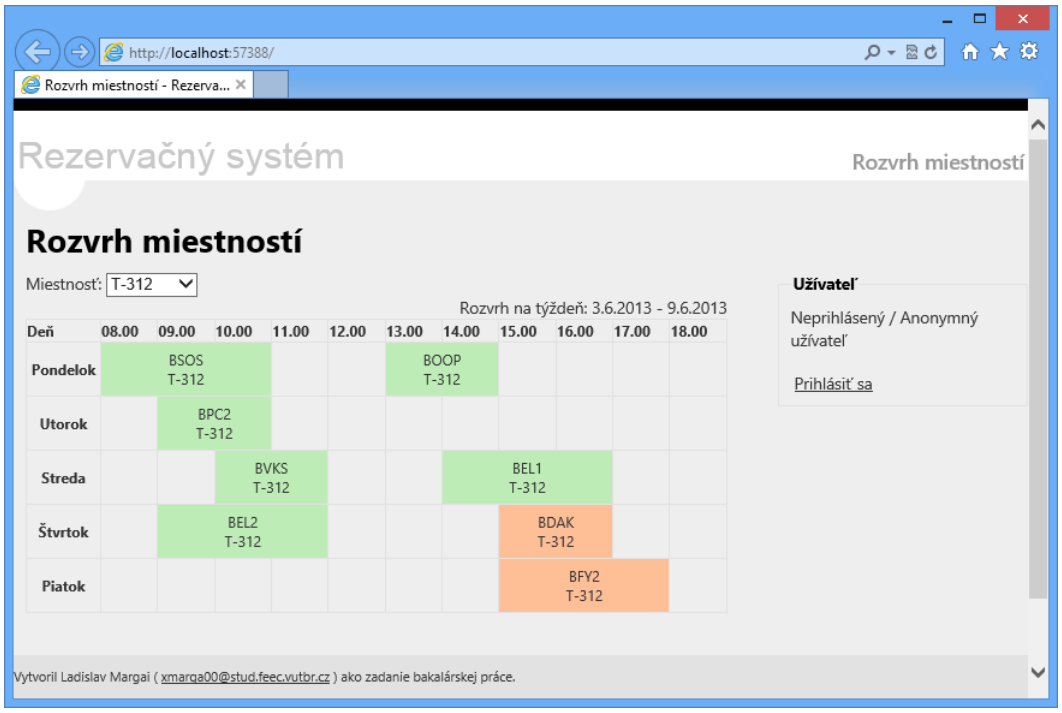
4.2 Role

Aby bolo jednoduchšie pridávanie a odoberanie právomocí užívateľom boli vytvorené následovné role:

- `Administrátori`
- `Učitelia`
- `Študenti`
- `Anonymní používatelia`

4.2.1 Anonymní používatelia

Predstavujú neprihlásených používateľov. Jediná povolená sekcia je `Rozvrh miestností` (Obr. 4.1). Po vybratí miestnosti sa zobrazí jej rozvrh na aktuálny týždeň. V rozvrhu je zobrazená pravidelná výuka ako aj jednorazové registrácie na aktuálny týždeň. Jednorazové registrácie, ktoré nie sú aktuálne pre daný týždeň sú zobrazené oranžovou farbou. Kdekoľvek inde bude anonymnému používateľovi prístup odoprený a zobrazená výzva na prihlásenie.



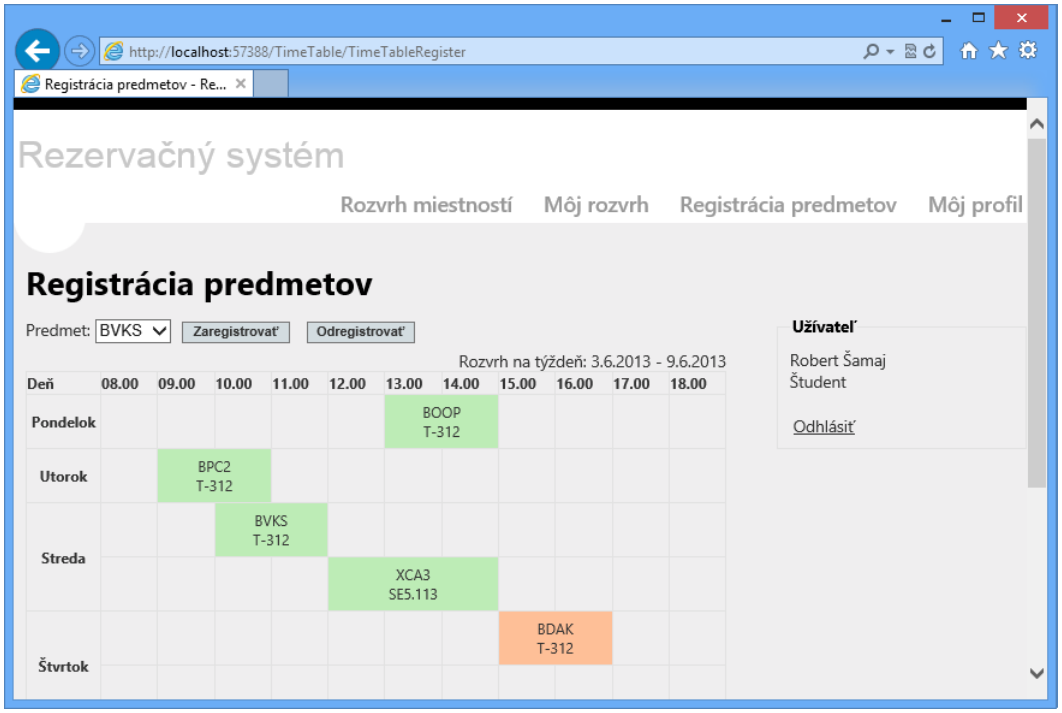
Obr. 4.1: Rozvrh miestností

4.2.2 Študenti

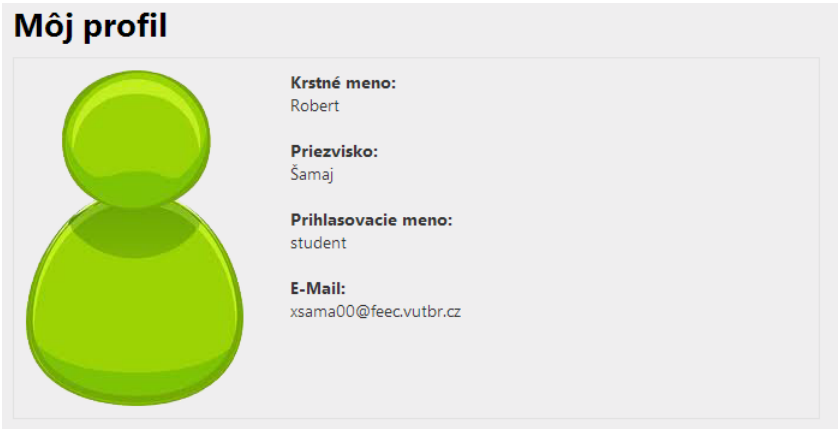
Majú okrem sekcie `Rozvrh miestností` prístup aj do sekcií:

- `Môj rozvrh` – V tejto sekcii sa zobrazí aktuálny rozvrh študenta na tento týždeň. Zobrazuje sa len vyučovanie, ktoré si študent zapísal v registrácií predmetov.
- `Registrácia predmetov` – Študent má možnosť zapísať si ľubovoľný predmet, ktorý má pravidelnú ale aj nepravidelnú výuku. Nerieši sa otázka, či má študent skutočne navštevovať predmet, alebo nie (Obr. 4.2). Študent si môže zapísať aj predmet, ktorý nemá registrovanú výuku, potom sa však nezobrazí v rozvrhu.

- `Môj profil` – Základné informácie o prihlásenom používateľovi bez možnosti ich zmeniť (Obr. 4.3).



Obr. 4.2: Registrácia predmetov

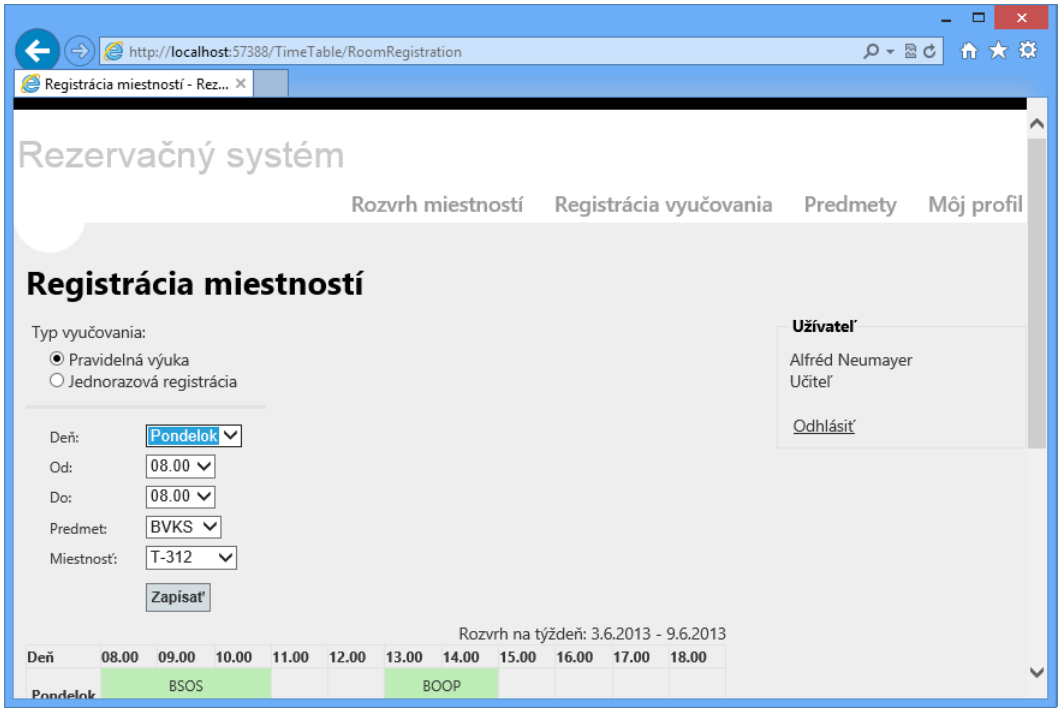


Obr. 4.3: Môj profil

4.2.3 Učitelia

Majú prístup do tých istých sekcií ako študenti, okrem sekcie `Registrácia predmetov`. Oproti študentom majú ešte prístupnú `Registráciu miestností` (Obr. 4.4), kde

si môžu zvoliť registráciu pre pravidelnú výuku – je potrebné určiť deň opakovania, alebo jednorazovú výuku – je potrebné zvoliť konkrétny dátum. V obidvoch prípadoch bude aplikácia kontrolovať, či nedochádza ku kolíziam.



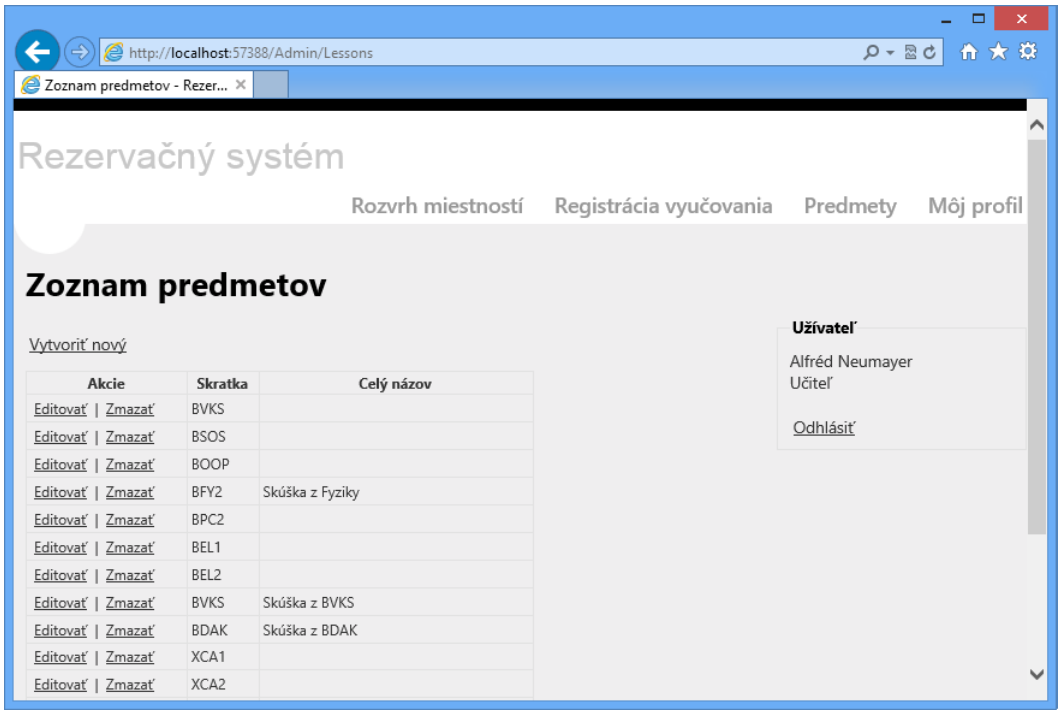
Obr. 4.4: Registrácia miestností

4.2.4 Administrátori

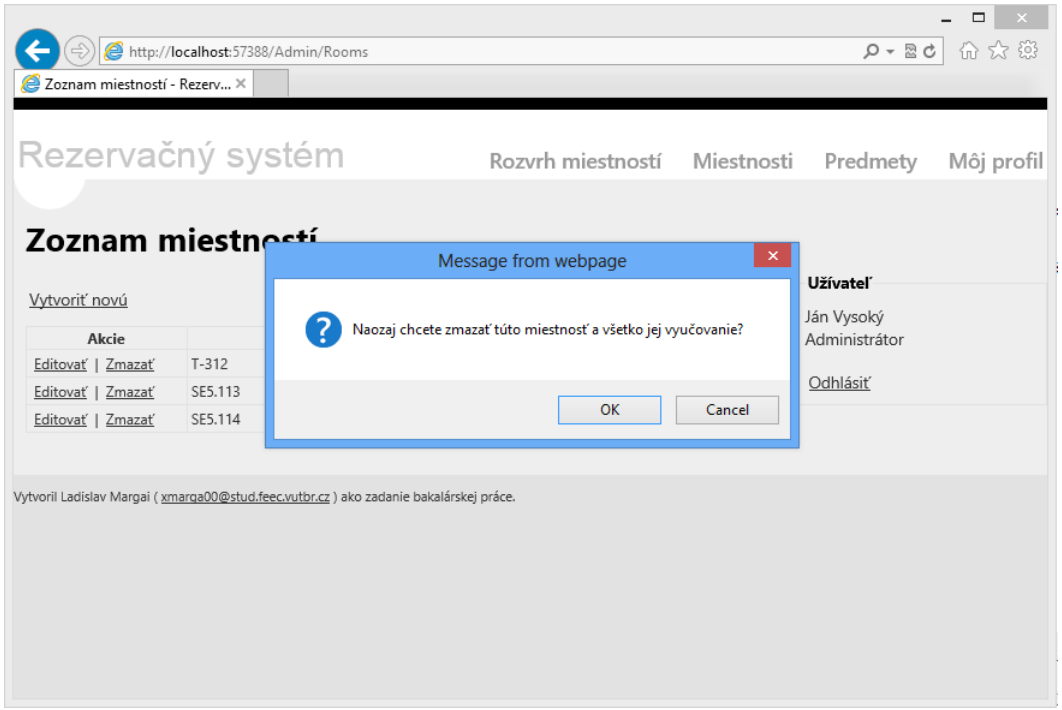
Majú absolútnu kontrolu nad jednotlivými vyučovaniami vo všetkých miestnostiach. Po kliknutí na sekciu `Miestnosti` sa zobrazí zoznam všetkých miestností – je možné vytvoriť novú, editovať alebo zmazať existujúcu(Obr. 4.6).

Po kliknutí na editáciu vybranej položky sa otvorí formulár pre zmenu názvu miestnosti. Zmazaním miestnosti sa odregistruje celé jej vyučovanie!

Učitelia aj Administrátori majú prístup ešte k editácií predmetov (Obr. 4.5), kde ich môžu editovať alebo vymazať.



Obr. 4.5: Zoznam predmetov



Obr. 4.6: Zmazanie miestnosti

5 VYTVORENIE DÁTOVÉHO MODELU

Prvé písmeno skratky MVC znamená **M**odel. Model obsahuje logiku a všetko, čo do nej spadá. Sú to napríklad výpočty, databázové dotazy, validácia a podobne. Model vôbec nevie o výstupe. Jeho funkcia spočíva v prijatí parametrov zvonku a vydanie dát von. Parametrami samozrejme nie je myslená URL adresa ani žiadne iné parametre od užívateľa. Model nevie, odkiaľ dáta v parametroch prišli a ani ako budú výstupné dáta sformátované a vypísané.

5.1 Entity Framework

V rezervačnom systéme bude slúžiť pre vytvorenie dátového modelu Entity Framework – jeden z najznámejších nástrojov pre **O**bjektovo **R**elačné **M**apovanie (ORM) v .NET prostredí. Bude nám zaisťovať automatickú konverziu dát medzi relačnou databázou (v tomto prípade MS SQL) a objektovo orientovaným programovacím jazykom (C#).

Možnosti modelovania a mapovania v EF:

1. `Model-First` – vývojár vytvára entity, vzťahy, dedičnosť a hierarchie medzi entitami priamo v Model dizajnérovi EDMX. Následne je vygenerovaná databáza ako aj objektový kód dátového modelu.
2. `Database-First` – Z vytvorenej databáze Model dizajnér EDMX vytvorí objektový kód dátového modelu
3. `Code-First` – vývojár sa úplne vyhne Model dizajnérovi EDMX a sám napíše objektový kód modelu, a Entity Framework vygeneruje databázu.

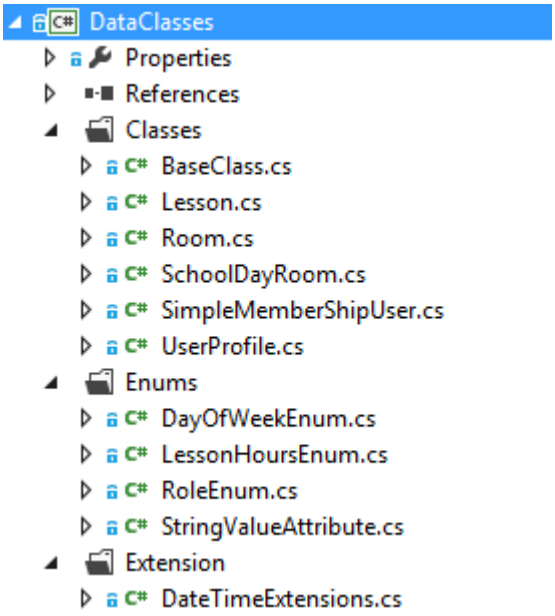
V rezervačnom systéme bude použitá možnosť `Code-First` pre modelovanie a mapovanie. Detailne rozpísané všetky možnosti Entity Framework-u sú k dispozícií na [8].

5.2 Vytvorenie modelov vo Visual Studiu

Jednou zo základných myšlienok návrhového vzoru MVC je nezávislosť modelu na prezentačnej vrstve (zobrazenie – View). Model má byť prenositeľný aby s ním mohli pracovať viaceré aplikácie – v tomto prípade bude s ním pracovať iba web aplikácia. Docieli sa to tým, že sa vo Visual Studiu vytvorí v `Solution` samostatný projekt, na ktorý sa následne odkazujú ostatné aplikácie (referencie).

5.2.1 Projekt pre dátové triedy – DataClasses

Po otvorení solution sa v `solution explorer` objaví niekoľko projektov, medzi nimi je aj projekt s názvom DataClasses (Obr. 5.1).



Obr. 5.1: Solution explorer – projekt DataClasses

V zložke `Classes` sú jednotlivé dátové triedy reprezentujúce entity. Každá trieda potrebuje vlastnosť (property), ktorá slúži ako jednoznačný identifikátor. Aby sa predišlo redundantnému kódu, bola vytvorená trieda `BaseClass.cs`, ktorá implementuje túto vlastnosť.

Zdrojový kód súboru `DataClasses/Classes/BaseClass.cs`.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DataClasses
{
    public class BaseClass<T>
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public T Id { get; set; }
    }
}
```

Ako je vidieť z ukážky, bol použitý generický typ triedy. Generické typy poskytujú možnosť k vytvoreniu výkonných dátových štruktúr, ktoré použitie bude schopný

kontrolovať už kompilátor. Tieto takzvané parametrizované typy sú vytvorené tak, že ich vnútorné algoritmy zostávajú rovnaké, ale vlastný typ dát je špecifikovaný vývojárom. Triedy a štruktúry sa vytvárajú rovnako ako pred tým a medzi znaky < a > se uvádzajú typové parametre. Keď sú potom použité, každý tento parameter je nahradený príslušným dátovým typom, ktorý zadal vývojár.

Z tejto triedy dedia všetky ostatné triedy, ako príklad je možné uviesť entitu, ktorá reprezentuje miestnosť v rezevačnom systéme.

Zdrojový kód súboru `DataClasses/Classes/Room.cs`.

```
using System.ComponentModel.DataAnnotations;

namespace DataClasses
{
    public class Room : BaseClass<int>
    {
        [Display(Name = "Lokalita")]
        [Required(ErrorMessage = "Povinné vyplniť!")]
        public string Location { get; set; }
    }
}
```

Trieda `Room` dedí z `BaseClass` a predáva do tejto triedy typový parameter `int`. Z toho vyplýva, že zdedená vlastnosť `Id` bude typu `int`.

Za povšimnutie stoja upresňujúce atribúty, ktoré sú pred jednotlivými vlastnosťami. Nazývajú sa `DataAnnotations`, a slúžia na to, aby Entity Framework vedel objekty správne serializovať do SQL, t.j. správne vygenerovať príslušné SQL tabuľky, a v nich jednotlivé stĺpce. Najpoužívanejšie sú uvedené v tabuľke.

Atribút	**Význam**
Key	Stĺpce, ktoré definujú primárny kľúč v tabuľke
MaxLength	Maximálna dĺžka stĺpca v databáze
Required	Hodnota stĺpca je nenulová (*NOT NULL*)
DatabaseGenerated	Označenie, ktoré indkuje, že hodnota je vyplnená databázou (možnosti: `Computed`, `Identity`, `None`)
ForeignKey and InverseProperty	Stĺpce, ktoré sú označené ako cuzdí kľúč

Tab. 5.1: Najpouživanešie atribúty

Ostatné typy sú vysvetlené na oficiálnych blogoch Microsoftu [9].

V zložke `Classes` je ešte jedna trieda, ktorá nededí z `BaseClass` a to `SimpleMemberShipUser`, ktorá implementuje `SimpleMembership Provider`. Je to nové rozhranie v .NET 4.5. ktoré je v základe veľmi štíhle, a dáva oproti staršiemu `MembershipProvider` vývojárovi obrovskú možnosť rozšírenia.

Keďže Entity Framework od verzie 5.0 podporuje ukladanie do databáze aj typ `Enum`, tak v zložke `Enums`, je vytvorených niekoľko výčtových typov.

Zdrojový kód súboru `DataClasses/Enums/RoleEnum.cs`.

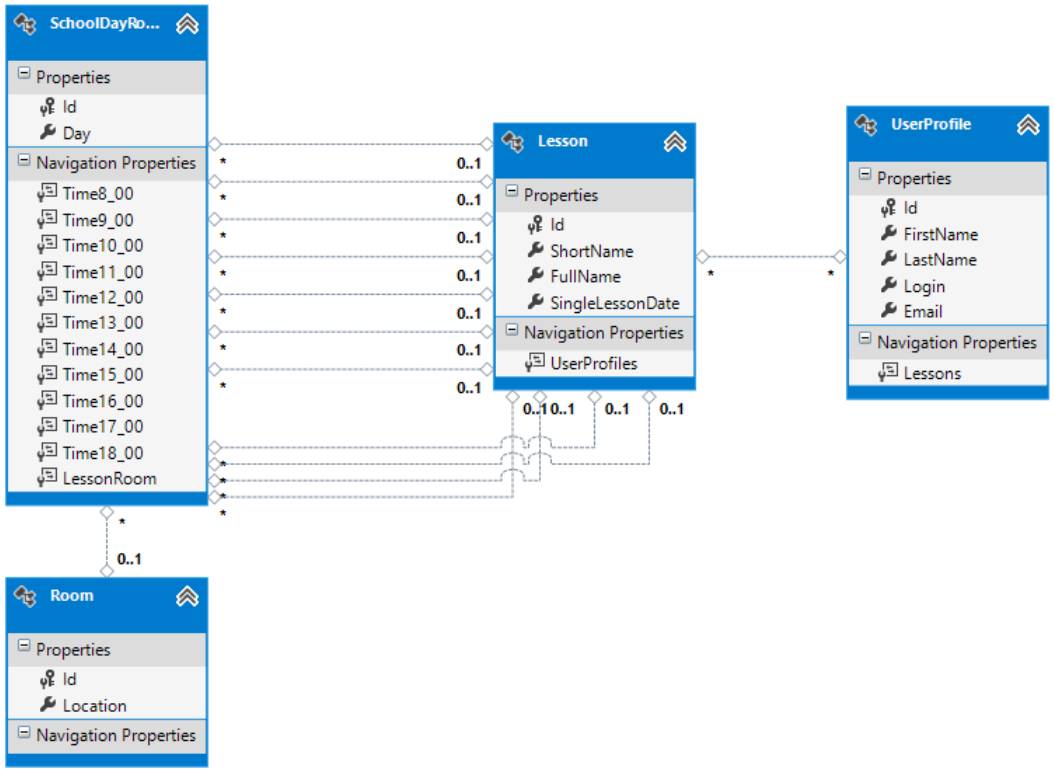
```
namespace DataClasses
{
    public enum RoleEnum
    {
        [StringValue("Študent")]
        Students = 1,
        [StringValue("Učiteľ")]
        Teachers = 2,
        [StringValue("Administrátor")]
        Administrators = 3
    }
}
```

Pre potreby aplikácie, je potrebné definovať aj textovú identifikáciu jednotlivých položiek výčtového typu. Preto v súbore `DataClasses/Enums/StringValueAttribute.cs` je napísaných niekoľko rozširujúcich (extensions) metód, pre prácu s výčtovými typmi. Autorom je Matt Simner [10], zdrojový kód je pod licenciou CPOL - The Code Project Open License, `http://www.codeproject.com/info/cpol10.aspx`.

Tieto rozširujúce metódy umožňujú aplikovanie reťazového atribútu na jednotlivé položky, ako aj podporu pre správne zobrazovanie týchto hodnôt v aplikácií.

Posledná zložka `Extensions` obsahuje vlastné rozširujúce metódy, ako napríklad `DateTimeExtensions.cs`, ktorá rozširuje triedu `DateTime`, ktorá sa nachádza v systémovej knižnici `mscorlib.dll`. Sú tam pridané dve metódy, ktoré zistia dátum začiatku a konca týždňa.

5.2.2 Vnútorná logika modelu



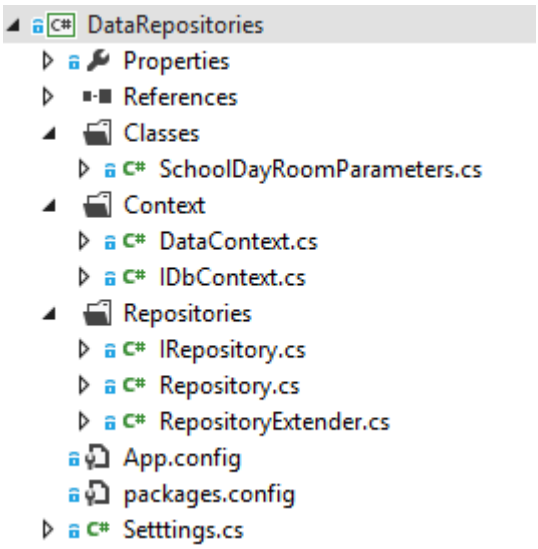
Obr. 5.2: Vygerovaný objektový model

Z vygenerovaného objektového modelu vo Visual Studiu, vyplýva, že každá miestnosť má vytvorenú entitu `Room`, kde sú uložené údaje o miestnosti. Každý školský deň je charakterizovaný práve jednou miestnosťou, na ktorú odkazuje. Ďalej má vytvorených presný počet vlastností, ktoré reprezentujú čas výuky, a na každú jednu odkazuje presne jedna hodina, ktorá je reprezentovaná entitou `Lesson`. Každá hodina odkazuje na viacerých používateľov (študentov),t.j. ľudí ktorí si ju zapísali.

To, že deň má vytvorených presný počet vlastností, ktoré reprezentujú čas výuky nie je to ideálne riešenie, ale bolo zvolené zámerne, aby bolo vidieť, že pomocou Entity Frameworku veľmi ľahko sa dá modifikovať definícia jednotlivých entít, s regeneráciu celej SQL databázy bez straty dát. Viac o tom je rozpísané v kapitole Rozšíriteľnosť[7].

5.2.3 Projekt dátového repozitára – DataRepositories

Keďže každá jedná dátová entita, potrebuje implementovať základné CRUD(Create, Read, Update, Delete) operácie, tak padlo rozhodnutie na vyžitie vzoru takzvaného `repozitára objektov`. Opať bol zvolený nový projekt – `DataRepositories` (Obr. 5.3)



Obr. 5.3: Solution explorer – projekt DataRepositories

V zložke `Repositories` sa nachádza rozhranie `IRepository`, kde sú špecifikované základné CRUD operácie.

Zdrojový kód súboru `DataRepositories/Repositories/IRepository.cs`.

```
using System.Linq;

namespace DataRepositories
{
    public interface IRepository<TEntity> where TEntity : class
    {
        IQueryable<TEntity> GetAll();
        TEntity GetById(object id);
        void Insert(TEntity entity);
        void Update(TEntity entity);
        void Delete(TEntity entity);
    }
}
```

Trieda `Repository` toto rozhranie implementuje. Ďalej sa tu nachádza súbor `RepositoryExtender.cs` v ktorom sú rozširujúce (extensions) metódy, pre parametrizované typy repozitárov.

Extension metóda z `DataRepositories/Repositories/RepositoryExtender.cs`

```
/// <summary>
/// Gets current user full user name.
/// </summary>
/// <param name="repository">this repository</param>
/// <param name="login">login name</param>
/// <returns>Full user name</returns>
public static string GetFullUserName(this Repository<UserProfile> repository,
                                     string login)
{
    var userProfile = from u in repository.GetAll()
                      where u.Login.Equals(login)
                      select (u.FirstName + " " + u.LastName);

    return userProfile.Single();
}
```

Kontextová trieda `DataContext`, a rozhranie `IDbContext`, ktorá táto trieda implementuje, poskytujú prostriedok pre Entity Framework pre dotazovanie a pracovanie s jednotlivými dátovými entitami ako s objektami.

V zložke `Classes` sa nachádza trieda `SchoolDayRoomParameters`, ktorá slúži ako komunikačný most medzi dátovým modelom a MVC aplikáciu. Má na starosti validáciu a analyzovanie dynamických vstupných parametrov z prezentačnej vrstvy aplikácie.

Ukážka validačnej metódy zo súboru
`DataRepositories/Repositories/SchoolDayRoomParameters.cs`.

```
/// <summary>
/// Validates From and To selected hours.
/// </summary>
private void ValidateFromToHour()
{
    if (((int)hourTo - (int)hourFrom) <= 0)
    {
        isValid = false;
        erroMsg.Add("Čas od-do sa nesmie prekrývať!");
    }
}
```

6 WEB APLIKÁCIA

Tretí projekt, ktorý sa nachádza v `Solution` je samotná web aplikácia.

6.1 Príprava a základné nastavenia

6.1.1 Visual Studio

Na tvorbu aplikácií pre ASP.NET MVC4 je potrebné mať nainštalované vývojové prostredie Visual Studio verzie 2010 alebo 2012. Sú voľne dostupné na ofiálnych stránkach Microsoftu <`http://www.microsoft.com/visualstudio/en-us/try`>, kde sa dá stiahnuť buď 30-dňová skúšobná verzia, alebo značne orezaná express edícia (Visual Web Developer).

Visual Studio 2012 už obsahuje v sebe MVC4 Framework integrovaný, do Visual Studia 2010 ho treba manuálne doinštalovať. Aktuálna verzia MVC Frameworku sa nachádza na <`http://www.asp.net/mvc`>.

Všetky aplikácie a ukážky publikované v tejto práci sú písané pod Visual Studiom 2012, niektoré postupy sa môžu líšiť vo Visual Studiu 2010.

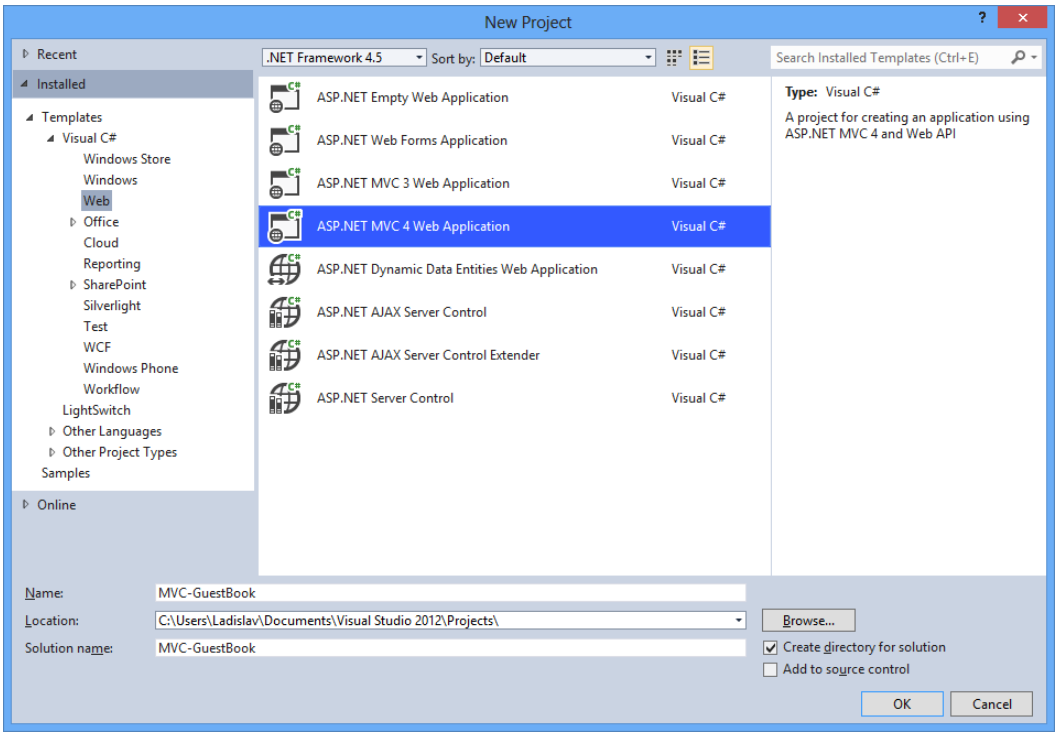
6.1.2 Vytvorenie nového projektu

Nový projekt sa založí kliknutím na `File -> New -> Project` v kontextovom menu. Po otvorení záložky `Web` v ľavom vertikálnom menu je na výber niekoľko možností, je potrebné vybrať `ASP.NET MVC 4 Web Application`, zadať názov a lokalitu projektu. Po potvrdení formulára sa objaví ešte jedno dialógové okno, kde treba upresniť následujúce údaje:

- `Template` – Typ šablóny, ktorá sa má použiť na predgenerovanie projektu. V závislosti na predpokladanom nasadení aplikácie sa modifikujú niektoré časti konfigurácie (napr. typ použitej autentifikácie), alebo sa predgenerujú celé bloky kódu.
- `View engine` – Syntaktický analyzátor, ktorý transformuje jednlotlivé Zobrazenia (View). S príchodom MVC3 bola zabudovaná podpora pre `Razor engine`. Z dôvodov spätnej kompatibility je ponechaný aj `ASPX engine`.
- `Create a unit test project` – Pri zaškrtnutí tejto možnosti je automatický vytvorený nový projekt na pracovanie s unit testami.

6.1.3 Prehľad adresárovej štruktúry aplikácie

Po vytvorení projektu sa na disku vytvorilo niekoľko adresárov, podadresárov a rôznych iných súborov. Každý súbor a adresár má svoj špecifický účel a odporúča

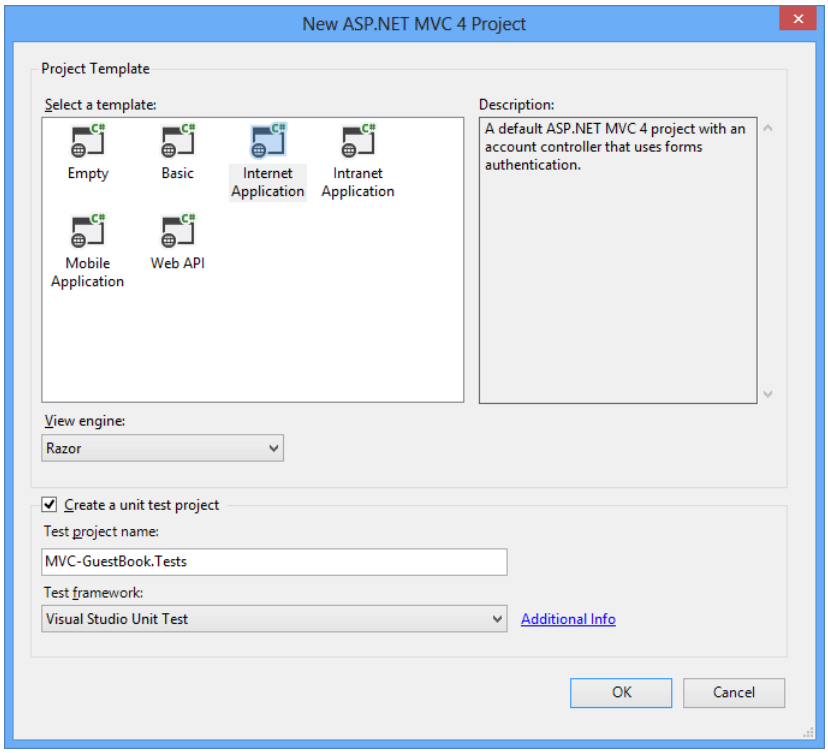


Obr. 6.1: Vytvorenie nového projektu

sa dodržiavať danú hierarchiu. Zobraziť a prechádzať adresárovú štruktúru projektu je možné cez `Solution Explorer` (Obr. 6.3).

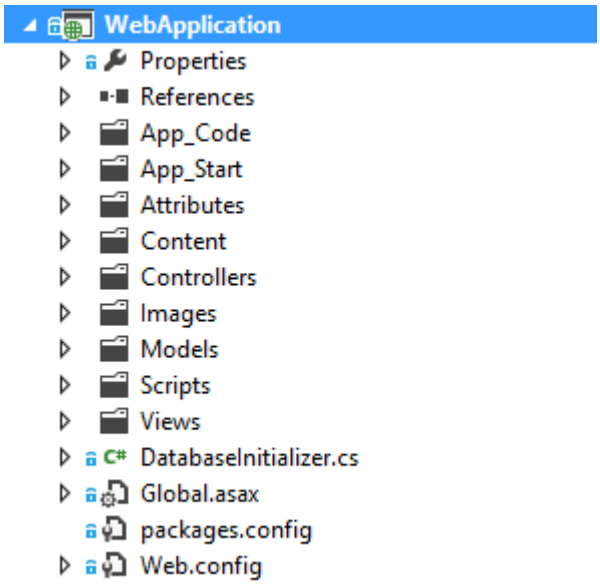
Popis najdôležitejších súborov a priečinkov:

- `App_Data` – Slúži na ukladanie databáz, XML súborov a akýchkoľvek iných dát, potrebné pre beh ASP.NET aplikácie. Do tohto adresára smie čítať a zapisovať len ASP.NET aplikácia. Rezervačný systém používa SQL databázu, čiže táto zložka bola zbytočná a preto zmazaná.
- `App_Code` – súbory v tejto zložke sa kompilujú až za behu aplikácie.
- `Content` – Tu sa uskladňuje statický obsah ako sú obrázky, kaskádové štýly a témy – šablóny.
- `Controllers` – Obsahuje triedy kontrolérov. Pri vytváraní kontrolérov stojí za povšimnutie , že sa v tomto priečinku dodržuje špecifická štruktúra, ktorú zaisťuje samotné Visual Studio 2012.
- `Models` – Nachádzajú sa tu dátové modely, ktoré používa aplikácia.



Obr. 6.2: Prednastavenie nového projektu

- `Scripts` – Sem sa odporúča umiestniť JavaScriptové skripty. Po vytvorení MVC Guestbook projektu sa do tejto lokality implicitne nahrali jQuerry knižnice používané na vstupnú validáciu údajov na strane klienta.
- `Views` – Obsahuje súbory s príponou `.aspx` alebo `.cshtml`. Sú zodpovedné za realizovanie Zobrazenia (View).
- `Global.asax` – Tento súbor sa nachádza v koreňovom adresári projektu. Obsahuje kód, ktorý sa vykonáva na základe životného cyklu aplikácie, napr. registrácia virtuálnych ciest pri štarte.
- `Web.config` – Konfiguračný súbor, ktorý sa tak isto sa nachádza v koreňovom adresári aplikácie, obsahuje všetky potrebné nastavenia, ktoré sú nevyhnutné pre bezproblémový beh ASP.NET MVC aplikácie. Niektoré vykonané zmeny v tomto súbore vyžadujú reštart aplikácie.



Obr. 6.3: Solution Explorer - web aplikácia

6.2 Kontroléry použité vo webovej aplikácií

Kontroléry alebo inak povedané radiče reagujú na udalosti (typicky pochádzajúce od užívateľa) a zaisťuje zmeny v modeli nebo v zobrazení.

V rezervačnom systéme sa používajú tri:

- `AccountController` – slúži na správu účtu, prihlásenie a odhlásenie užívateľa, vypísanie základných informácií o ňom.
- `AdminController` – administratorske nástroje na vytvorenie / editáciu / zmazanie miestonosti a učební.
- `TimeTableController` – má na starosti všetku prácu ohľadom registrovania, zobrazovania a odregistrovania predmetov.

Ukážka z `WebApplication/Controllers/AdminController.cs`

```
//
// GET: /Admin/Rooms
[RoleAuthorize(Role=RoleEnum.Administrators)]
public ActionResult Rooms()
{
    var lessonRepository = new Repository<Room>(context);
    var model = lessonRepository.GetAll().ToList();
    return View(model);
}
```

Ako je vidieť z ukážky, kontrolér využíva atribút `RoleAuthorize`, ktorému sa nastavuje ako parameter `Role` z `RoleEnum`. Tento atribút nie ju štandardne obsiahnutý v `SimpleMemberhip provider`, bol vytvorený na základe úpravy `AuthorizeAttribute`.

Zdrojový kód `WebApplication/Attributes/RoleAuthorizeAttribute.cs`

```
using System;
using System.Web.Mvc;
using DataClasses;

namespace WebApplication
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method)]
    public class RoleAuthorizeAttribute : AuthorizeAttribute
    {
        public RoleEnum Role { get; set; }

        public override void OnAuthorization(AuthorizationContext filterContext)
        {
            if (Role != 0)
                Roles = Role.ToString();

            base.OnAuthorization(filterContext);
        }
    }
}
```

6.3 Zobrazenia (Views) v Rezervačnom systéme

Každý kontrolér má k sebe vytvorených niekoľko zobrazení, ktoré zodpovedajú ich akciám. Avšak, nie každá akcia kontroléru musí mať vlastné zobrazenie – napríklad akcia `RoomDelete` v `AdminController` po úspešnom zmazaní len presmeruje užívateľa na inú akciu – `RedirectToAction("Rooms")`.

Za zmienku stojí vlastný Razor helper, ktorý sa nachádza v `App_Code` (každý Razor helper musí byť tam, aby bol zbuildovaný až samotnou ASP.NET aplikáciu a nie Visual Studiom). Jeho úlohou je vykreslovať deň za dňom v rozvrhu, a na základe toho, či sa jedná o pravidelnú výuku alebo len jednorazovú registráciu korektne zobrazovať dáta.

Za zmienku stojí ešte zložka `Models`, v tomto prípade neobsahuje skutočné modely ktoré predstavujú dáta, ale len jeden pomocný model pre zobrazovanie. Nemá žiadnu zložitú vnútornú logiku.

7 ROZŠÍRITEĽNOSŤ APLIKÁCIE

Napriek tomu, že je počet vyučovacích hodín pevne daný vlastnosťami objektu `SchoolDayRoom`, veľmi ľahko je možné rozšíriť ho o nové hodiny. V klasickej databáze by bolo veľmi obtiažne a náročné pridať novú položku, ale Entity Framework to zvládne za vývojára doslova za behu, s tým, že stávajúce dáta ostanú zachované.

Ako prvý krok je potrebné rozšíriť dátovú triedu, ktorá predstavuje entitu o dané vlastnosti. V prípade, že je potrebné napr. rozšíriť vyučovanie o 2 hodiny, tak ich stačí len pridať.

Zdrojový kód `DataClasses/Classes/SchoolDayRoom.cs`

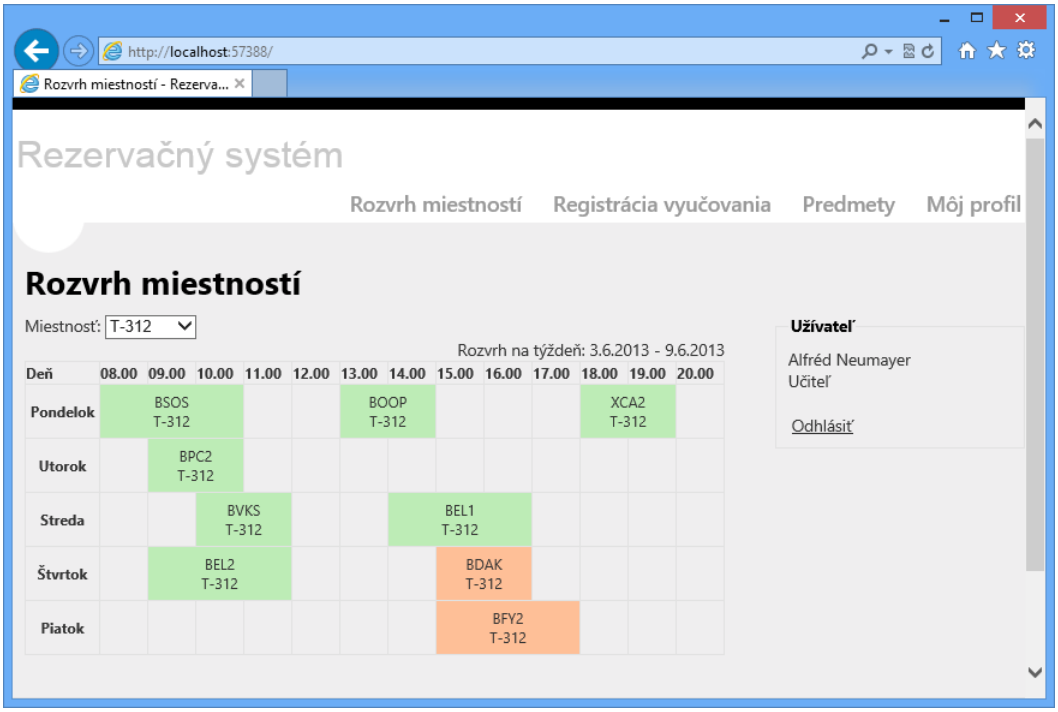
```
namespace DataClasses
{
    public class SchoolDayRoom : BaseClass<int>
    {
        public DayOfWeekEnum Day { get; set; }
        public virtual Lesson Time8_00 { get; set; }
        public virtual Lesson Time9_00 { get; set; }
        public virtual Lesson Time10_00 { get; set; }
        public virtual Lesson Time11_00 { get; set; }
        public virtual Lesson Time12_00 { get; set; }
        public virtual Lesson Time13_00 { get; set; }
        public virtual Lesson Time14_00 { get; set; }
        public virtual Lesson Time15_00 { get; set; }
        public virtual Lesson Time16_00 { get; set; }
        public virtual Lesson Time17_00 { get; set; }
        public virtual Lesson Time18_00 { get; set; }

        // Added properties
        public virtual Lesson Time19_00 { get; set; }
        public virtual Lesson Time20_00 { get; set; }

        public virtual Room LessonRoom { get; set; }
    }
}
```

Zároveň treba upraviť aj závislosti, ak nejaké existujú, v tomto prípade ešte aj výčtový typ hodín pre správne zobrazovanie.

Zdrojový kód `DataClasses/Classes/LessonHoursEnum.cs`

```
namespace DataClasses
{
    public enum LessonHoursEnum
    {            .
                 .
                 .
        [StringValue("16.00")]
        Time16_00 = 9,

        [StringValue("17.00")]
        Time17_00 = 10,

        [StringValue("18.00")]
        Time18_00 = 11,

        // Added values
        [StringValue("19.00")]
        Time19_00 = 12,

        [StringValue("20.00")]
        Time20_00 = 13

    }
}
```

Po týchto úpravách je potrebné ešte zbuidlovať aplikáciu. Po úspešnom builde by sa mal zobraziť rozvrh rozšírený o dve hodiny, s tým, že do týchto nových dvoch hodín je možné aj registrovať výuku.



Obr. 7.1: Rozšírený rozvrh o dve hodiny

8 ZÁVER

V práci bol popísaný návrhový vzor Model-View-Controller, a jeho použitie vo webových aplikáciach. V práci je rozpísané porovnanie so starším konceptom návrhu – webové formuláre. Pomocou jazyka C# a Visual Studia 2012 bola navrhnutá vzorová aplikácia Rezervačný systém, na ktorej boli ukázané možnosti customizácie MVC frameworku (vlastný Razor helper pre zobrazenia, vlastný atribút pre kontroléry).

LITERATÚRA

[1] SHARP, J. *Microsoft Visual C# 2010*, Nakladatelství Computer Press, a.s. 2010, 696 s., ISBN 978-80-251-3147-3

[2] MACDONALD, M., FREEMAN, A., SZPUSZTA, A. *ASP.NET 4 a C# 2010 – KNIHA 1 - tvorba dynamických stránek profesionálně*, Zoner press, 2011, 880 s., ISBN 978-80-7413-131-8

[3] MACDONALD, M., FREEMAN, A., SZPUSZTA, A. *ASP.NET 4 a C# 2010 – KNIHA 2 – tvorba dynamických stránek profesionálně*, Zoner press, 2011, 704 s., ISBN 978-80-7413-145-5

[4] PALERMO, J., BOGARD, J., HEXTER, E., HINZE, M. a SKINNER, J. *ASP.NET MVC 4 in Action*, Manning, 2012, 496 s., ISBN 978-1617290411.

[5] SANDERSON, S. a FREEMAN, A. *Pro ASP.NET MVC 3 Framework*, Apress, 2011, 824 s., ISBN 978-1430234043.

[6] GUTHRIE, S. *Introducing „Razor" – a new view engine for ASP.NET* [online]. 2010, poslední aktualizace 02. 07. 2010 [cit. 10. 12. 2012]. Dostupné z URL: <`http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx`>.

[7] BOREK, B. *Prezentační vzory z rodiny MVC* [online]. 2009, poslední aktualizace 11. 05. 2009 [cit. 12. 12. 2012]. Dostupné z URL: <`http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/`>.

[8] JOHNESS, J. *Entity Framework tutorial* [online]. 2013, poslední aktualizace 10. 05. 2013 [cit. 5. 6. 2013]. Dostupné z URL: <`http://www.entityframeworktutorial.net/`>.

[9] LERMAN, J. *Code First DataAnnotations* [online]. 2011, poslední aktualizace 01. 04. 2011 [cit. 5. 6. 2013]. Dostupné z URL: <`http://msdn.microsoft.com/en-us/data/gg193958.aspx`>.

[10] SIMMER, M. *String Enumerations in C#* [online]. 2005, poslední aktualizace 07. 06. 2005 [cit. 5. 6. 2013]. Dostupné z URL: <`http://www.codeproject.com/Articles/11130/String-Enumerations-in-C`>.

[11] MICROSOFT, Inc. *C# Coding Conventions (C# Programming Guide)* [online]. 2012, poslední aktualizace 02. 12. 2012 [cit. 5. 6. 2013]. Dostupné z URL: <`http://msdn.microsoft.com/en-us/library/vstudio/ff926074.aspx`>.

[12] Doomen, D. *C# 3.0, C# 4.0 and C# 5.0 Coding Guidelines* [online]. 2010, poslední aktualizace 03. 05. 2012 [cit. 5. 6. 2013]. Dostupné z URL: <http://www.codeproject.com/Articles/11130/String-Enumerations-in-C>.

[13] GALLOWAY, J. *SimpleMembership, Membership Providers, Universal Providers and the new ASP.NET 4.5 Web Forms and ASP.NET MVC 4 templates* [online]. 2012, poslední aktualizace 29. 08. 2012 [cit. 2. 6. 2013]. Dostupné z URL: <http://weblogs.asp.net/jgalloway/archive/2012/08/29/simplemembership-membership-providers-universal-providers-and-the-new-asp-net-4-5-web-forms-and-asp-net-mvc-4-templates.aspx>.

[14] JEESHENLEE, I. *How to create a simple data access layer using Repository Pattern with Entity Framework* [online]. 2012, poslední aktualizace 28. 11. 2012 [cit. 25. 5. 2013]. Dostupné z URL: <http://bizvise.com/2012/11/28/how-to-create-a-simple-data-access-layer-using-repository-pattern-with-entity-framework/>.

[15] MICROSOFT, INC. *The Entity Framework Configuration Section* [online]. 2012, poslední aktualizace 10. 10. 2012 [cit. 5. 3. 2013]. Dostupné z URL: <http://msdn.microsoft.com/en-us/data/jj556606.aspx/>.

[16] MICROSOFT, INC. *Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application* [online]. 2012, poslední aktualizace 01. 11. 2012 [cit. 5. 3. 2013]. Dostupné z URL: <http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>.

[17] MICROSOFT, INC. *Code First to a New Database* [online]. 2012, poslední aktualizace 18. 04. 2012 [cit. 5. 3. 2013]. Dostupné z URL: <http://msdn.microsoft.com/en-us/data/jj193542>.

[18] ANDERS, A. *EF Code First Navigation Properties and Foreign Keys* [online]. 2012, poslední aktualizace 23. 06. 2012 [cit. 5. 6. 2013]. Dostupné z URL: <http://coding.abel.nu/2012/03/ef-code-first-navigation-properties-and-foreign-keys/>.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

MVC Model-View-Controller (Model-Zobrazenie-Radič)

ASP Active Server Pages (skriptovacia platforma)

HTML HyperText Markup Language (značkovací jazyk pre hypertext)

JSON JavaScript Object Notation (javascriptový objektový zápis)

XML Extensible Markup Language (rozšíriteľný značkovací jazyk)

HTTP Hypertext Transfer Protocol (internetový protokol)

CGI Common Gateway Interface (protokol pre pripojenie externých aplikácií)

HTX HTML Extension Template (rozšíriteľné HTML šablóny)

IDC Internet Database Connector (protokol pre pripojenie databáz)

LINQ Language Integrated Query (integrovaný jazyk pre dotazovanie)

AJAX Asynchronous JavaScript and XML (obecné označenie pre technológie vývoja interaktívnych webových aplikácií)

XHTML Extensible Hypertext Markup Language (rozšiřiteľný hypertextový značkovací jazyk)

jQuerry javascriptová knižnica

CRUD Create-Read-Update-Delete (Vytváranie-Čítanie-Aktualizovanie-Mazanie)

RAD Rapid Application Development (extrémne rýchly vývoj aplikácií)