Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program : Aplikovaná informatika

Obor: Informační systémy a technologie

Využití architektury MVC na platformě .NET

DIPLOMOVÁ PRÁCE

Student : David Ježek

Vedoucí : doc. Ing. Alena Buchalcevová, Ph.D.

Oponent : Ing. Václav Švec

2012

Prohlášení

Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 27. 6. 2012 .....

Jméno a příjmení studenta

Poděkování

Tímto bych rád poděkoval doc. Ing. Aleně Buchalcevové, Ph.D., vedoucí diplomové práce za všechnu pomoc, podněty, ochotu, vstřícnost a trpělivost při vypracování této diplomové práce.

Dále bych chtěl poděkovat rodině za podporu a také kolegům z práce za pomoc a konzultace této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá využitím technologie MVC (Model View Controller) v oblasti vývoje webových aplikací na platformě ASP.NET. Zejména pak využitím nejnovější verze ASP.NET MVC 3. Práce v první části popisuje architekturu MVC a v druhé částí se zabývá jejím využitím v různých částí webových aplikací a srovnáním s PHP.

Klíčová slova

Model View Controller (MVC), .NET, ASP.NET, ASP.NET MVC, Razor.

Abstract

This thesis deals with usage of MVC (Model View Controller) technology in web development on ASP.NET platform from Microsoft. Mainly it deals with latest version of framework ASP.NET MVC 3. First part describes MVC architecture and the second describes usage of MVC in certain parts of web application an comparing with PHP.

Keywords

Model View Controller (MVC), .NET, ASP.NET, ASP.NET MVC, Razor.

Obsah

1. Úvod

Softwarové aplikace představují silný nástroj pro organizace, především ovlivňují její výkonnost. Precizní doladění parametrů a funkcí aplikací, aby vyhovovaly požadavkům konkrétního byznysu, umožňuje organizacím se více soustředit na podstatné aktivity svého obchodování.

Dnešní vývoj softwaru se může spolehnout buď na model využívající samostatně stojící aplikace (tlustý klient), které si mohou vyměňovat informace prostřednictvím peer to peer (typ počítačové sítě ve které spolu komunikují přímo klienti), nebo se mohou spolehnout na aplikace fungující na architektuře klient/server (tenký klient), která odděluje klienta a server, kteří spolu komunikují přes počítačovou síť.

Klient/server popisuje vztah mezi dvěma počítačovými programy, tento vztah je založen na tom, že jeden program vystupuje jako klient, dotazuje se na informace a využití služeb na server. Klient je v této roli pouze tenká vrstva obsahující uživatelské rozhraní a prostředky pro komunikaci se serverem, zatím co sever obsahuje všechnu aplikační logiku. Tento model je základem pro dnešní webové aplikace, příkladem je internetové bankovnictví, přístup na e-mail atp. Výhodou webových aplikací je jejich reálná dostupnost odkudkoliv ze světa použitím pouze internetového připojení a prohlížeče. Můj názor je, že toto je budoucnost aplikací, jak je dnes známe. Největším zástupcem tlustých klientů (samostatně stojících aplikací) je v dnešní době herní průmysl. Avšak dnes už i v této oblasti lze hovořit o postupném přecházení na klient server architekturu. Kromě jednoduchých webových her se již tituly pro hraní více hráčů přesouvají na server, kde jsou uloženy veškeré informace, které aplikace potřebuje pro svoji funkci.

Pro tuto diplomovou práci je důležité, že se zabývá pouze vývojem webových aplikací a to zejména využitím nástrojů od společnosti Microsoft.

1.1 Technické předpoklady

Tato diplomová práce vyžaduje základní znalosti z oblasti internetových aplikací. Tato podkapitola popisuje základní termíny potřebné pro pochopení souvislostí v následujícím textu. Zavádí se zde některé obecně používané pojmy v oblasti vývoje aplikací. Diplomová práce je následně v jednotlivých kapitolách a podkapitolách rozebírá podrobněji.

- **Webform aplikace** - aplikace využívající připojení k internetu k přístupu k informacím a veškeré funkcionalitě.
- **Winform aplikace** - aplikace využívající k práci klasický přístup spustitelné aplikace pod operačním systémem.
- **Framework** - v této diplomové práci je použit pojem framework jako softwarový framework a je univerzální, znovupoužitelnou programovou platformou. (Wikipedia, 2012)
- **MVC** - architektura využívající rozdělení aplikace na Model, View a Controller (Data, UI, Aplikační část)
- **.NET** - framework z produkce společnosti Microsoft, který vznikl původně k využití ve Winform aplikacích.
- **ASP.NET** - framework vycházející ze základního .NET, který je rozšířen o podporu pro tvorbu Webform aplikací.
- **ASP.NET MVC** - framework vycházející z ASP.NET, který je rozšířen o podporu aplikací využívajících architekturu MVC
- **PHP** - Hypertext Preprocessor je programovacím jazykem umožňujícím vývoj webových stránek s dynamicky proměnlivým obsahem
- **Razor** - syntaxe využívaná v poslední verzi ASP.NET MVC frameworku, která má poskytovat jednoduchost a snadnost tvorby aplikací v PHP, programátorům využívajícím ASP.NET

Pro dobré pochopení práce, je také vhodné aby čtenář byl obeznámen s vývojem webových aplikací, alespoň se základním rámcem. Vhodné je také nainstalovat vývojové prostředí Visual Studio 2010, pro možnost vyzkoušení jednotlivých příkladů v diplomové práci.

1.2 Důvody výběru tématu

Tvorbou webových aplikací a stránek se již zabývám sedm let, z toho důvodu jsem se rozhodl vybrat si pro svoji diplomovou práci téma velmi blízké tomuto oboru. V této práci bych rád čtenáři přiblížil moderní trendy vývoje webových aplikací a to zejména MVC architekturou.

Původní námět na toto téma vzniknul ve společnosti Develion ART s.r.o, kde pracuji jako programátor a s kolegy jsem konzultoval, jaké téma by se pro moji diplomovou práci nejlépe hodilo. Kolegové z práce mi v počátku mého zájmu o ASP.NET MVC velmi pomohli a výrazně přispěli ke znalostem, které o této architektuře mám a budu v této práci prezentovat.

1.3 Cíle práce

Tématika diplomové práce vymezuje jednoznačně cíle, kterých se budu v této práci snažit dosáhnout. MVC jako architektura představuje současný trend vývoje webových aplikací a proto je toto téma ideální pro zpracování teoretické práce, která čtenáře seznámí s problematikou vývoje s využitím této architektury.

Hlavním cílem této diplomové práce je ukázání nové funkcionality, kterou umožňuje spojení ASP.NET s architekturou MVC. Zejména se pak práce bude zabývat poslední aktuální verzí frameworku ASP.NET MVC 3.0. Jednotlivé přínosy, které vznikly tímto spojením, jsou prezentovány na ukázkách kódu, které jsem pro tento účel vytvořil. Ke každému kódu, zaměřenému na jednotlivé funkcionality, je uveden popis, který má vysvětlit čtenáři jejich smysl a použití v praktickém světě. Čtenář se znalostmi s vývojem aplikací na platformě ASP.NET, by po přečtení této diplomové práce měl být schopen uplatnit uvedené příklady v praxi a vytvořit základní jednoduchou aplikaci využívající principy MVC.

Dílčím cílem práce je vytvoření souboru pravidel, která čtenáři pomůže při tvorbě webových aplikací s využitím ASP.NET MVC frameworku, zefektivnit práci a vyhnout se základním chybám. Soubor těchto chyb jsem vytvořil na základě vlastních zkušeností a také na základě preventivního studia podobných článků na internetu a ve zpravodajích.

Druhým dílčím cílem by pak mělo být porovnání intepretované syntaxe PHP a Razor za účelem přesvědčení čtenáře, že PHP není jedinou cestou jak jednoduše

vytvářet aplikace. Snažím se tak prakticky předvést, že využití ASP.NET je prostřednictvím syntaxe Razor výrazně podobná s PHP, což má za následek jednodušší změnu orientace stávajících PHP programátorů na ASP.NET.

Mým vlastním přínosem do této diplomové práce jsou zejména zkušenosti z oblasti programování v ASP.NET MVC 3 s využitím syntaxe Razor. Také zkušenosti s vývojem od malých webových prezentací až po velké projekty, na kterých jsem spolupracoval s týmem kolegů. Práce by tak měla přinést vypovídající výsledky a poradit začátečníkům i pokročilým, zda se vydat cestou MVC či nikoliv.

1.4 Rešerše prací na téma architektury MVC

Na téma architektury MVC již je možné na internetu nalézt několik bakalářských a i některé diplomové práce. Narazil jsem při svém hledání na diplomovu práci Jana Vondruše s názvem Renovace MVC Frameworků (Vondrouš, 2008).

Tato diplomová práce se zabývá možnostmi modernizace MVC frameworků. Kromě obecného popisu technologie a pojmů z oblasti MVC architektury, se zabývá konkrétními MVC frameworky od společnosti Apache, přesněji Struts 1 a 2, jež slouží jako reálná ukázka zlepšení frameworků, tedy jejich modernizaci.

Struts je open-source framework, sloužící k vytváření webových aplikací, podobně jako ASP.NET MVC, který ovšem není open-source, ale je licencován společností Microsoft. Struts je založen na technologiích z oblasti vývoje Java aplikací, tedy JSP (java server pages), JavaBeans a podobně.

Další práce, na kterou jsem narazil, se již zabývala MVC na platformě .NET. Autorem je Stanislav Kuznetsov, pod názvem Použití technologie Model View Controller (MVC) (Kuznetsov, 2009).

Tato bakalářská práce se již tematicky blíží předmětu mojí práce, významný rozdíl je v datování, bakalářská práce pana Kuznetsova se zabývá první verzi MVC od společnosti Microsoft, která měla dost problémů se stabilitou a ještě nebyla úplně připravena pro rozšířené využití. Teoretická část, podobným způsobem jako tato práce, rozebírá problematiku architektury MVC, má práce však vkládá více důrazu na modernější využití této technologie ve spojení s ASP.NET, která od doby vydání práce pana Kuznetsova proběhla. Moje práce také porovnává současné PHP s trendy Razor syntaxe na platformě .NET.

1.5 Literární zdroje diplomové práce

Důležitým zdrojem pro moji práci je kniha ASP.NET 3.5 a C# tvorba dynamických stránek Profesionálně od autorů Matthew MacDonald a Mario Szpuszta (MacDonald, a další, 2008).

Tato kniha shrnuje základní poznatky o platformě .NET a jejím webovém využití a je základním znalostním předpokladem pro využití dalších publikací týkajících se přímo ASP.NET MVC frameworků.

Tuto publikaci jsem již využil při psaní své bakalářské práce, je velmi přehledná a je možné ji sehnat i v přeloženém vydání od Zoner press, ale podobně jako další literární zdroj je v původním vydání od Apress.

Dalším literárním podkladem pro moji práci je Pro ASP.NET MVC 3 Framework od autorů Adam Freeman a Stever Sandreson (Freeman, a další, 2011).

Tato anglická publikace se zabývá výhradně tématikou ASP.NET MVC 3, které se chci v této diplomové práci věnovat a to od začátků programování a seznámení s frameworkem, až po nasazení hotové aplikace. Kniha je vydána nakladatelem APress a patří do série technicko-vzdělávacích publikací pro programátory.

Dále se budu přiklánět k webovému portálu microsoft MSDN, který velmi pečlivě dbá na dokumentaci a co nejpřesnější popis jejich technologií.

Dalším zdrojem jsou technologicky zaměřené články na internetu, kupříkladu Úvod do architektury MVC (Bernard, 2009) nebo podobné články přímo od společnosti Microsoft na webovém protálu www.asp.net (ASP.NET, 2011). Současný stav informačních technologií umožňuje výrazné využití vyhledávacích portálů k získáni relevantních informací.

2. Vývoj aplikací na platforme .NET

Záměrem této kapitoly je rozebrat základy a historické události, které vedly až ke vzniku dnešní architektury. Tato kapitola umožňuje pochopit, z čeho vznikly základy webového vývoje, jako předpokladu pro vznik architektury MVC.

2.1 Webové aplikace

První přenosy dat prostřednictvím protokolu HTTP (Hypertext Transfer Protocol) byly uskutečněny před více než 20lety a o přenos se zasloužili Tim Berners-Lee společně s Robertem Cailliauem. V dnešní době je zkratka HTTP už prakticky součásti běžného hovorového jazyka. Zpočátku nebylo lehké protokol nějak rozumně využít a vývojáři tak museli vytvářet složité aplikace, které se navzájem musely vyhledávat. Postupem čas vznikla za tímto účelem série standardů, jako HTML (Hypertext Markup Language) nebo XML (Extensible Markup Language). Oba zmíněné standardy měly za cíl pomoci vývojářům s tvorbou multiplatformních aplikací. HTML jako jednoduchý jazyk mělo za úkol definovat jak zobrazit dokument na prakticky jakýchkoliv počítačových platformách a XML mělo za cíl standardizovat formát dat pro různé platformy, jejichž využitím si mohli aplikace vyměňovat informace. (MacDonald, a další, 2008)

Výrobci softwaru tak měli k dispozici komunikační kanály a programovací jazyky pro vytváření aplikací, které by s webem komunikovaly. Bylo však zapotřebí vytvořit pracovní rámce, jež by vývojářům umožnily nejen návrh architektury, ale také cestu jakou aplikace vyvíjet a rozšiřovat co nejjednodušším způsobem. Tím se prakticky rozpoutaly závody ve zbrojení, přední softwaroví výrobci IBM,Sun Microsystems či Microsoft se začali předhánět o co nejlepší platformy. Microsoft vsadil v těchto závodech na kombinaci částí pro web (HTML, XML atp.) a osvědčenou objektově orientovanou metodologií. (MacDonald, a další, 2008)

Webovým aplikacím se využitím frameworku ASP.NET také jinak říká WebForm aplikace. Toto rozlišení je nutné zejména proto, že nástroje sady .NET využívají i aplikace desktopové jinak nazývané WinForm aplikace.

2.2 Desktopové aplikace

Desktopové aplikace představují starší přístup než aplikace webové, ale i přesto se stále udržely v některých aplikačních oblastech. Desktopové aplikace zůstaly tam kde je potřeba zvýšená bezpečnost, případně kde je zapotřebí výkonu, který není schopen server poskytnout každému jednotlivému uživateli zvlášť. Jako dobrý příklad je třeba Adobe Photoshop, či Visual Studio, zatím není v reálných silách možné převést tyto aplikační celky do webového prohlížeče, jako se tomu podařilo v případě office balíčků na službě Google docs. Nároky na výpočetní výkon těchto aplikací je totiž neúměrně vyšší. V koncepci se desktopové aplikace označují jako tlustý klient, protože hlavní funkcionalita se nachází na uživatelské stanici, kdežto webové aplikace jsou modelově provozovány jako klient tenký, veškerá funkcionalita je na serveru a uživatelské PC se na data dotazuje prostřednictvím webového prohlížeče, který zastupuje tenkého klienta.

2.3 HyperText Markup Language (HTML)

HTML je značkovací jazyk pro hypertext, tvoří základ prezentace dokumentů na internetu, vznikl v roce 1990, využitím principů univerzálního značkovacího jazyku SGML. Tento jazyk neustále prochází renovací a vývojem, je hodně závislý na webových prohlížečích a jejich možnostmi zobrazování dokumentů. V současné době posledním je standard HTML 5. Práce na této verzi HTML mají být, dle odhadů, ukončeny až v roce 2022, tak vzdálené datum je odhadováno pro vyřešení všech problémů a opravení všech chyb (Wikipedia, 2011).

Html charakterizují značky (tagy), ke kterým jsou přiřazeny různé atributy dle definice W3C (World Wide Web Consortium), což je mezinárodní konsorcium jež má na starost společně s veřejností vývoj webových standardů. Text uzavřený mezi jednotlivé tagy je obsahem dokumentu (jeho sématika). Jednotlivé tagy jsou uzavřený do úhlových závorek (`<div>`,`<a>`). Vnitřní text spolu se svými hraničními tagy je označován jako element dokumentu. Například `<p>Ahoj světe</p>` je elementem obsahujícím text v odstavci. Elementy tímto způsobem tvoří celou strukturu webových stránek. (Wikipedia, 2011)

2.4 Platforma .NET

Platforma .NET je relativně starý pojem, pro její upřesnění bych rád použil citaci Jozefa Vochozky z Ústavu výpočetní techniky při Masarykově univerzitě v brně:

".NET (dot net) je platforma vyvinutá firmou Microsoft. Cílem platformy je zjednodušit tvorbu programových aplikací a současně zpřehlednit jejich zdrojové kódy. V neposlední řadě pak zjednodušit správu softwarových instalací na konkrétním počítači. Současně by tato platforma měla posílit vnitřní i vnější bezpečnost SW systému jako takového. Nakolik Microsoft sází na tuto platformu ilustruje i fakt, že většina vývojářských nástrojů nabízených na webových stránkách Microsoftu je nabízena již pouze pro platformu .NET.

Při úvahách o platformě .NET je třeba mít na mysli i její mohutnou podporu ze strany Microsoftu, která spočívá mimo jiné i v množství dostupné literatury i tutoriálů. Navíc je možno získat kompletní vývojové prostředí i se školením v rámci programů podporujících zavedení této platformy. Díky komplexnosti vývojového prostředí vývojářské firmy přechází nebo už přešly na .NET. Všechny potřebné vrstvy této platformy jsou vestavěnou součástí Windows 2003 a vyšších verzí tohoto systému". (Vochozka, 2003)

Z této charakteristiky je jasné, proč se moje práce zaměřuje hlavně na tuto platformu. Mým osobním přesvědčením je, že má velkou budoucnost a stejně jako nepředpokládám zánik společnosti Microsoft, očekávám, že i podpora a vývoj platformy .NET bude pokračovat.

Ještě je pro tuto diplomovou práci důležité uvést základní nástin struktury platformy .NET:

"Zevrubný popis struktury platformy .NET rozhodně vyžaduje rozsáhlejší text než jen krátký článek ve Zpravodaji. Návštěvník knihkupectví snadno zjistí, kolik samostatných titulů mající ve svém názvu slovo .NET je dnes běžně nabízeno. V našem informativním textu se budeme velice stručně zabývat pouze dvěma nejzákladnějšími vrstvami - první z nich je Common Language layer (CL) a druhou .NET class library." (Vochozka, 2003)

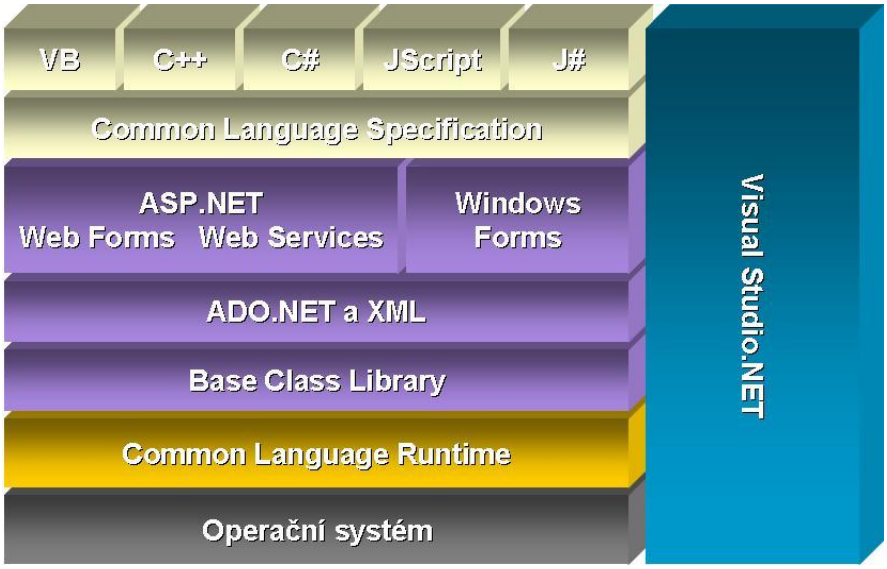
Rozdělení na CL a .NET class library je důležité k základnímu pochopení struktury, kterou obecně zavádí .NET. Jazyk CL vystupuje z hlediska platformy .NET jako interpretace kompilovaného kódu, jeho deklarace vznikla z návrhu společností Microsoft, Hewlet Packard a Intel. Tento krok měl snížit míru závislosti platformy .NET od společnosti Microsoft. Definice a omezení CL lze popsat následující citací:

"Autoři jazyka CL snad ani nepředpokládají, že by programátoři užívali jazyk CL přímo pro psaní zdrojového kódu (i když by to pravděpodobně bylo možné). Jazyk je primárně zamýšlen jako souhrn všech povolených operací, které může běžná aplikace potřebovat. Autoři jazyka CL vycházejí z předpokladu, že běžná aplikace nepotřebuje například formátovat disk nebo nepotřebuje nízkoúrovňové V/V operace, ale s velkou pravděpodobností bude tato aplikace potřebovat nástroje pro své zabezpečení (například šifrovací algoritmy), správu paměti, ošetření výjimek nebo například podporu svého ladění. Aplikace v .NETu by tak neměly mít přístup k těm funkcím, které mohou poškodit data jiných aplikací nebo aplikace samotné, ale na druhé straně by měly implicitně obsahovat vlastnosti zajišťující jejich vyšší bezpečnost a stabilitu." (Vochozka, 2003)

Druhou částí platformy .NET jsou takzvané Class Library, ty v zásadě splňují definici frameworku, kterou rozebírá tato práce v následující podkapitole, tedy poskytují programátorovi znovupoužitelnou programovou platformu. Pro kontinuitu definice platformy .NET je důležité zmínit citaci pana Vochozky z Masarykovi univerzity:

"Class library je knihovnou objektových tříd. Tyto knihovny jsou opět sdíleny všemi podporovanými jazyky. Pro snazší orientaci v knihovnách je pro volání jednotlivých objektů a jejich metod užito konceptu jmenných prostorů, jak jsou známy z xml. Zajímavý je i fakt, že všechny knihovny jsou sdíleny i mezi jednotlivými programovacími jazyky." (Vochozka, 2003)

Pro ilustraci uvádím jakou množinu prvků platforma .NET obsahuje, graficky znázorněno na Obr 1.



Obr 1 - Architektura .NET Frameworku (Běhálek, 2007)

2.5 Definice pojmu framework

Velmi často se v různých článcích a publikacích setkáváme s pojmem framework. Ačkoliv český překlad pro daný problém existuje (slovo "rámec"), není mnoho používán, proto jsem se i já v této práci přiklonil k využívání slova framework s českým skloňováním.

Anglická Wikipedie definuje framework jako abstrakci, ve které software poskytující určitou obecnou funkcionalitu, může být selektivně změn uživatelským kódem, poskytuje tak specifickou aplikaci softwaru. Softwarový framework je univerzální, znovupoužitelnou programovou platformou používanou k vývoji aplikací, produktů a celkových řešení. Softwarový framework obsahuje podpůrné programy, překladače kódu, knihovny, rozhraní pro programovací aplikace a soubor nástrojů, které spojují dohromady všemožné komponenty, aby umožnily vývoj řešení projektu. Přeloženo z (Wikipedia, 2012).

Pro základní pochopení pojmu je anglická verze Wikipedie dostačující, problém spočívá spíše ve formě samotného portálu, který funguje je otevřenou encyklopedií. Pro tuto diplomovou práci využívám framework jako pojem v souladu s následující definicí, zprostředkovaně získanou z diplomové práce Bc. Radima Hradeckého psanou na Českém vysokém učení technickém v roce 2009.

"Frameworky jsou znovupoužitelné softwarové struktury, založené na detailním pochopení aplikační domény, vyplývající většinou z opakované implementace daného zadání. Obsahují jak návrh, tak implementaci. Návrh reprezentuje model aplikační domény a implementace stanovuje alespoň částečně jeho použití. Zahrnuje tak nasbírané poznatky z předchozího vývoje - jakou softwarovou architekturu a její implementaci je vhodné zvolit. Zároveň ale nechává prostor pro úpravy, aby míra jeho použitelnosti byla co nejširší." (Hradecký, 2009)

Tato definice vychází ze studia disertační práce zabývající se tématikou frameworkových designů autora Drika Riehleho (Riehle, 2000). Tato definice nejlépe vypovídá o tom, jak budu přistupovat k frameworkům v této práci a dle mého názoru i nejlépe vystihuje daný pojem.

2.6 Historie frameworku pro vývoj webových aplikací od společnosti Microsoft

Pro správné pochopení smyslu vzniku ASP.NET MVC, je zapotřebí poznat kroky, které tomuto frameworku předcházely. Vývoj frameworku od společnosti Microsoft ilustruje tabulka v Obr 2.

Period	Technology	Strengths	Weaknesses
Jurassic	Common Gateway Interface (CGI)*	Simple Flexible Only option at the time	Runs outside the web server, so is resource-intensive (spawns a separate operating system process per request) Low-level
Bronze age	Microsoft Internet Database Connector (IDC)	Runs inside web server	Just a wrapper for SQL queries and templates for formatting result sets
1996	Active Server Pages (ASP)	General purpose	Interpreted at runtime Encourages "spaghetti code"
2002/03	ASP.NET Web Forms 1.0/1.1	Compiled "Stateful" UI Vast infrastructure Encourages object-oriented programming	Heavy on bandwidth Ugly HTML Untestable
2005	ASP.NET Web Forms 2.0		
2007	ASP.NET AJAX		
2008	ASP.NET Web Forms 3.5		
2009	ASP.NET MVC 1.0		
2010	ASP.NET MVC 2.0 ASP.NET Web Forms 4.0		
2011	ASP.NET MVC 3.0		

**CGI is a standard means of connecting a web server to an arbitrary executable program that returns dynamic content. The specification is maintained by the National Center for Supercomputing Applications (NCSA).*

Obr 2 - Microsoft's Lineage of Web Development Technologies (Freeman, a další, 2011)

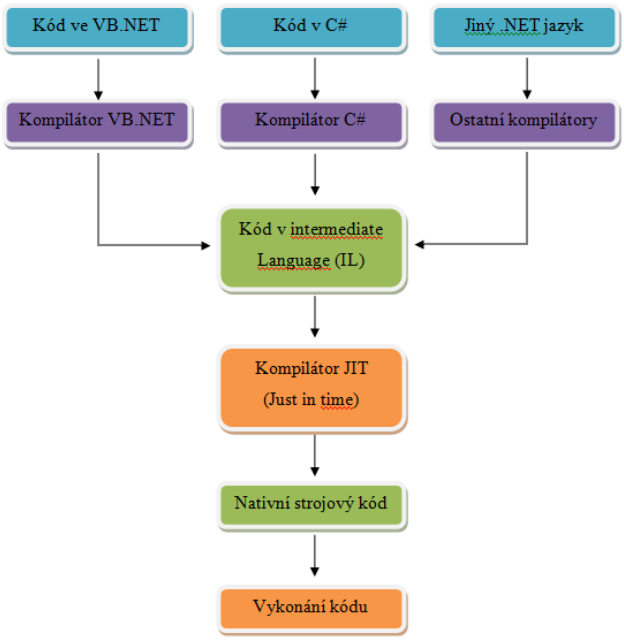
Jak je zde vidět, společnost Microsoft nezavrhuje vývoj standardních webform aplikací, mají stále své místo ve světě webových projektů, ale jsou zde vidět snahy o rychlý vývoj technologie MVC.

2.7 ASP.NET

Pro pochopení celé problematicky ASP.NET MVC, je nezbytné si na začátku projít skutečnosti, týkající se základního kamene webového vývoje společnosti Microsoft, tedy ASP.NET.

2.7.1 Kompilace kódu

Zdrojové kódy ASP.NET se na rozdíl od svých předchůdců kompilují a programátor tak vytvoří kód, který je procesem kompilace převeden a vykonán jak je znázorněno na Obr. 3 (MacDonald, a další, 2008).



Obr 3 - Kompilace kódu v ASP.NET (MacDonald, a další, 2008)

Kompilátor převede kód do tzv. "Intermidiate Language", tento kód je uložený v DLL (Dynamic-link library), který se obecně nazývá assembly. Kompilátor JIT (Just In Time) provádí kompilaci v době, kdy o něj aplikace zažádá (uživatel zavolá zkompilovanou funkci), přeloží jej do strojového kódu a ten se vykoná (MacDonald, a další, 2008).

2.7.2 ASP.NET a runtime prostředí CLR

Common Language Runtime (CLR) je virtuální prostředí, které jako součást Microsoft .NET framework je zodpovědné za spouštění a správu programů založených na platformě .NET. CLR se stará o spouštění zkompilovaného strojového kódu vzniklého z kompilace JIT (Just in time), které se přeloží do instrukcí a v návaznosti vykoná procesorem. Spojením CLR a ASP.NET webové aplikace získávají přístup k několika výhodám desktopových .NET aplikací. Rád bych jich pár citoval (MacDonald, a další, 2008):

- automatická správa paměti GC (Garbage Collection) – stará se o nakládání a průběžné využívání paměti, není proto nutné se o paměťové prostředky starat ručně,
- typová bezpečnost – při kompilaci ukládá .NET do assembly některé podrobnější informace o aplikaci (využité třídy, datové typy atd.) a tím urychlí jejich následné využívání, lze se tak vyvarovat chyb v nízkoúrovňové části aplikace,
- rozšiřitelná metadata – umožňují poskytování dodatečných informací pro runtime a jiné služby.

3. Softwarové vzory

Model View Controller je architektonickým vzorem používaným při vývoji softwaru, tato kapitola ukazuje rozdíl mezi architektonickým vzorem a návrhovým vzorem a také představuje základy architektonického vzoru MVC.

3.1 Architektonický vzor

Definice pojmu architektonický vzor není úplně jednoduchá, podobně jako u případu frameworku, existuje k dané problematice několik výkladů. Přitom vlastní definice již mohla existovat od poloviny 90. let minulého století. Již v té době se k dané problematice objevovaly publikace, takto je například definovaná v knize Pattern-Orriented Software Architecture:

Architektonický vzor vyjadřuje základní strukturální schéma organizace pro potřeby softwarových systémů. Poskytuje sestavu předdefinovaných subsystémů, upřesňuje jejich odpovědnosti a přidává pravidla a rady pro organizování vztahu mezi nimi. Přeloženo z (Frank Buschmann, 1996)

Anglická Wikipedie pojímá pojem takto:

Architektonický vzor je standardizovaný v oblasti softwarové architektury, v základní podobě má architektonický vzor širší pole působnosti než vzor designový, kromě vlastní koncepce aplikační, či datové logiky definuje i různé problematiky týkající se softwarového inženýrství. Patří sem například dostupnost aplikace, minimalizování obchodních rizik, výkonové omezení dostupného hardware a další. Přeloženo z (Wikipedia, 2009)

Definice z wikipedie vypovídá i o možnostech využití vzorů a více specifikuje význam architektonického vzoru v pojetí této práce.

3.2 Návrhový vzor

Problematika návrhového vzoru spočívá v rozdílu užití pojmu globálně nebo pro informační technologie, jak ukazuje definice z anglické Wikipedie:

Návrhový vzor je způsob jakým lze formálně dokumentovat řešení určitého designového problému. Původem vznikla tato myšlenka ve spojení s architekturou a to s architektem Christopherem Alexanderem. Od té doby byla adaptována do mnoha

odvětví, včetně počítačových věd. Návrhový vzor je využíván při samotném designování aplikace a také je velmi přínosný pro jednoduché nastínění a představení návrhu softwaru investorům, či komukoliv komu chceme svůj návrh představit. Přeloženo z (Wikipedia, 2012)

Pro potřeby této diplomové práce je zapotřebí tuto definici rozšířit, jelikož anglická wikipedie nevystihuje problematiku návrhových zdrojů v dostatečně širokém pojetí a zároveň dostatečně do hloubky. Proto tato práce čerpá z následující definice pocházející z diplomové práce Ing. Miloše Dvořáka, psané při VŠE Praha na tématiku Návrhových vzorů.

"*Dle* (Erich Gamma, 1995) *jsou návrhové vzory řešení, které se stále rozvíjí. Vymezení návrhových vzorů jako popisu řešení je velmi strohé. Jedná se spíše o jazyk, respektive o způsob komunikace. Jejich rozvoj je spojen s potřebou zachytit možnosti řešení netriviální problémů, které se opakovaně objevují při návrzích informačních systémů především v oblasti designu. Tyto dvě podmínky, tj. netriviální a opakující se situace, představují nutné podmínky pro vznik návrhových vzorů.*" (Dvořák, 2003)

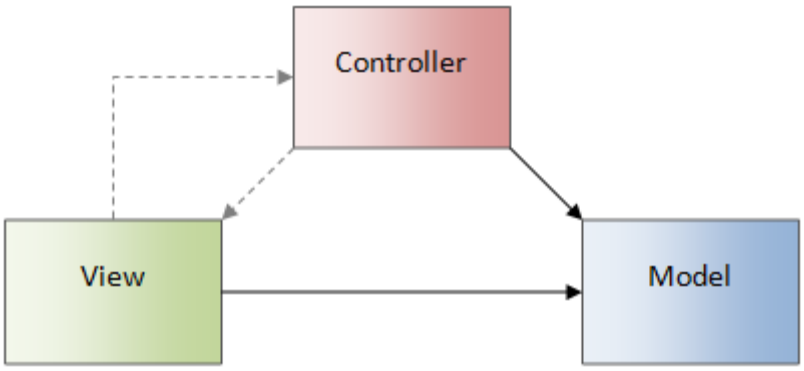
3.3 MVC

MVC (Model-View-Controller) v softwarové architektuře je považován za architektonický vzor. V softwarovém inženýrství není novou myšlenkou. MVC architektura vznikla roku 1978, byla využita firmou XEROX jako součást projektu Smalltalk. V dnešních dnech je však spojovaná zejména s technologií vývoje webových aplikací. Pro tuto skutečnost existují tyto důvody: (Freeman, a další, 2011)

MVC, ve spojení s HTTP požadavky, vytváří ideální spojení pro využití MVC architektury, tedy uživatel odešle požadavek (vykoná nějakou akci) a na základě toho aplikace vrátí nový aktualizovaný pohled (View), který se zobrazí a cyklus se může opakovat. (Freeman, a další, 2011)

Dnešní webové aplikace kombinují databázové aplikace, HTML kód a programový kód, přesně to je předpoklad pro účelné využití modelu MVC jako samotného. (Freeman, a další, 2011)

Na Obr 4 je ukázáno, jak vtah jednotlivých částí MVC funguje.



Obr 4 - Vztahy mezi vrstvami MVC modelu (Bernard, 2009)

- **Model (Model)**

Tato vrstva vytváří datovou logiku aplikace, vytváří přístup k datovým objektům v databázi a představuje tak výkonnou část aplikace. (Bernard, 2009)

- **Řadič (Controller)**

Funguje jako spojnice pohledu a modelu, jeho účelem je rozpoznávat a reagovat na impulzy, které prostřednictvím pohledu zadává uživatel. Rozhoduje, na základě volané akce, jaký pohled pro další zobrazení odeslat uživateli, a jaké události spustit pro aktualizaci či úpravy datového objektu modelu. Controller poskytuje veškerou aplikační logiku aplikace. (Bernard, 2009)

- **Pohled (View)**

Pohled představuje uživatelské rozhraní aplikace, na základě HTML struktury a zástupných datových znaků vytváří strukturu stránky, která se uživateli odešle a zobrazí v jeho webovém prohlížeči. Pro jeden model je možné vytvořit i více pohledů a ty pak deklarují různý vzhled a prezentaci dat v závislosti na okolnostech uživatele a jeho požadavcích. (Bernard, 2009)

Jsou tak odděleny všechny tři vrstvy webové aplikace standardizovaného modelu tří vrstev, a to Datová - Funkční - Prezentační.

Dnešní ASP.NET MVC vychází z těchto principů a využívá je k budování lépe optimalizovaných, rychlejších a přehlednějších aplikací. Toto rozdělení pak zásadně ulehčuje programátorům práci s vyhledáváním chyb.

ASP.NET MVC framework není zatím přímou součastí Visual Studia 2010. Do budoucna je plánované jej plně integrovat do nástupce vývojového prostředí od společností Microsoft. Prozatím je nezbytné si framework ve verzi 3.0 doinstalovat manuálně. Popis instalačních kroků lze v této diplomové práci nalézt v podkapitole 4.3.12.

4. ASP.NET MVC Framework

MVC spojené s ASP.NET vytváří velice mocný nástroj k budování webových aplikací. Limitem je pouze nutnost sdělenou informaci přenášet prostřednictvím HTML jazyka po HTTP protokolu, což znemožňuje tvorbu náročných aplikací jakými jsou například hry, či náročné desktopové aplikace. Vše je ve své podstatě omezeno pouze schopnostmi webových prohlížečů, což ale na druhou stranu umožňuje a usnadňuje tvorbu multiplatformních aplikací. (Freeman, a další, 2011)

ASP.NET MVC framework je k dispozici ve třech verzích, které jsou popsány v následujících podkapitolách. Základní programový balík pro tvorbu aplikací, použitím architektury MVC, je obsažen v ASP.NET MVC 1 následující verze 2.0 a 3.0, přinesly další podporu pro zefektivnění a zjednodušení práce programátorů. Jednotlivé přínosy verzí jsou proto zmíněny pod jednotlivými podkapitolami společně s ukázkami kódu využití některých z nich.

4.1 ASP.NET MVC 1

S příchodem frameworku ASP.NET MVC 1 v březnu roku 2009 bylo umožněno nástavbou na ASP.NET 3.5 využívat architektonické výhody MVC při vývoji webových aplikací, tím bylo dosaženo oddělení uživatelského rozhraní, aplikační logiky a datového základu. (Wikipedia, 2011)

4.2 ASP.NET MVC 2

Druhá verze frameworku ASP.NET MVC byla vydána téměř o rok později, v roce 2010, a přinesla vývojářům webových aplikaci novou funkcionalitu pro zjednodušení jejich práce.

Nová funkcionalita má pomoci ke zvýšení výkonnosti programátorů snížením náročnosti některých úkonů, které se často opakují. Upřesnění nových funkcionalit popisují následující podkapitoly, které čerpají přímo ze serveru společnosti Microsoft. (Microsoft, 2010)

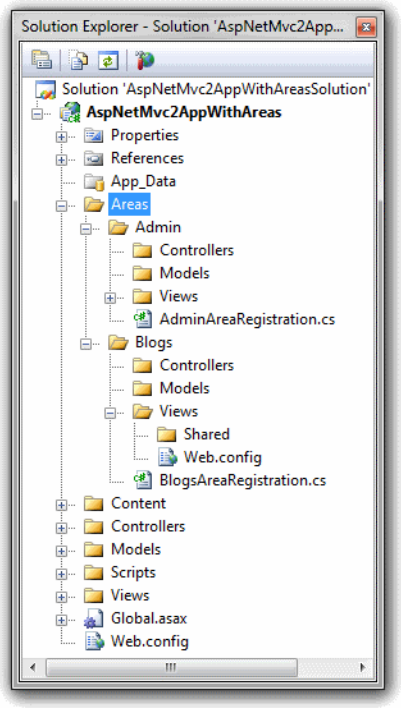
Následující podkapitoly popisují tzv. Template Helpers pro generování formulářů na základě šablon, vyšší členění aplikace pro zvládnutí komplexních aplikací, deklarování standardní hodnoty proměnných v modelu a ošetření typu požadavku HTTP události pro ošetření PUT, GET atd.

4.2.1 Generování formulářů na základě šablon

Generování formuláře na základě šablon, pomocí tzv. Template helpers, umožňuje automaticky spojovat HTML elementy s objekty, pro zobrazení s různými datovými typy. Například pokud máme data v typu System.DateTime, které chceme v pohledu zobrazit, Template Helper nám umožní vložit přímo do html objekt date-picker, což je v uživatelském rozhraní malý kalendář, který zobrazí datum z datového elementu. Takto je výrazně jednodušší prezentovat data daného pohledu pomocí jednoduchých objektů. (Microsoft, 2010)

4.2.2 Vyšší členění aplikace

Vyšší členění aplikace pomocí tzv. Areas pomáhají organizovat velké projekty do menších sekcí, za účelem zvládnutí komplexnosti velké webové aplikace. Každá sekce ("area") reprezentuje samostatnou sekci webové stránky a používá se k seskupení řadičů (controller) a pohledů (view) patřících z hlediska funkcionality k sobě. Pro vytvoření nové sekce (area), stačí kliknout pravým tlačítkem na projekt ,vybrat Add a zvolit Area, tento postup zobrazí dialogové okno, kde zadáme název sekce a Visual studio přidá sekci automaticky do složky jak ukazuje projekt s využitím dvou oddělených sekcí a to Admin a Blogs viz Obr 5. (Microsoft, 2010)



Obr 5 - Areas Admin a Blogs v projektu (Microsoft, 2010)

4.2.3 Podpora pro standardní hodnoty atributů volaných pohledů

Třída System.ComponentModel.DefaultValueAttribute umožňuje poskytovat metodě, která ovládá (controller) zobrazení dat v pohledu, standardní proměnné pro případ, že nejsou poskytnuty přímo voláním URL. Voláním URL je možné přímo jednotlivou akci řadiče (controller). (Microsoft, 2010)

Pro vložení standardního parametru využijeme DefaultValue jako v ukázce Kód 1.

```
public class ArticleController {
      public ActionResult View(int id, [DefaultValue(1)]int page)
      {
      // Kód metody
      }
}
```

Kód 1 - ukázka užití DefaultValue

Je tak přímo možné volat následující:

- /Article/View/123
- /Article/View/123?page=1 (Stejné jako předchozí požadavek)
- /Article/View/123?page=2

Aplikaci tak žádáme o zobrazení článku 123, v prvních dvou případech chceme zobrazit stránku 1 a v posledním stránku dvě. Na základě Kód 1 tak vidíme, že když je defaultní hodnota nastavená na 1, je volání prvních dvou opravdu stejné. Při volání třetího požadavku, ale nebude mít nastavení defaultní hodnoty žádný vliv a na základě kódu procedury se nám zobrazí druhá stránka článku.

4.2.4 Podpora pro omezení vlastnosti proměnných

DataAnnotations slouží k omezení vlastností jednotlivých datových typů proměnných, ASP.NET MVC 2 přináší omezení v RangeAttribute (omezení číselných proměnných například: 10 - 500), RequiredAttribute (stanoví, že pro uložení datového modelu je potřeba zadat proměnnou, touto vlastností označenou), StringLengthAttribute (omezení maximální délky řetězce) a RegexAttribute (omezení formy vloženého řetězce, používá se pro validaci e-mailových adres, telefonních čísel, umožňuje definovat formu jakéhokoliv řetězce na základě syntaxe regex, například `'[a-zA-Z0-9 ]*'` omezí řetězec pouze na malá a velká písmena a číslice). (Microsoft, 2010)

4.2.5 Ošetření metody HTTP

Za účelem zjištění různých metod požadavku HTTP, je možné označit jednotlivé akce řadiče (controller), pomocí označení kterou akci GET, PUT, POST nebo DELETE daná akce obsluhuje. V praxi je tak možné aby byla volána pouze jedna akce s různými výsledky zpracování. Akce bude stále mít stejný název, ale označením pomocí HttpGet, HttpPut, HttpPost nebo HttpDelete, můžeme rozlišit čtyři různé funkce, které budou volání ošetřovat jako v příkladě Kód 2. (Microsoft, 2010)

```
[HttpPost]
public ActionResult Edit(int id)
{
   // Kód metody pro POST
}
[HttpPut]
public ActionResult Edit(int id)
{
   // Kód metody pro PUT
}
```

Kód 2 - Ošetření HTTP metody

ASP.NET MVC 2 přinesl ještě několik dalších užitečných změn a zlepšení a hlavně výrazně zlepšil stabilitu aplikací založených na tomto frameworku. Zmíněné změny jsou těmi nejdůležitějšími pro zjednodušení práce s architekturou MVC na platformě .NET, pro potřeby této diplomové práce je ale nejdůležitější verze ASP.NET MVC 3 popsaná v následující podkapitole.

4.3 ASP.NET MVC 3

ASP.NET MVC 3 staví na předchozích verzích ASP.NET MVC 1 a 2, přidává mnoho zlepšení vycházejících z požadavků komunity programátorů na platformě .NET. Jednoznačným cílem je zjednodušení programového kódu a umožnění hlubšího rozšíření. Rád bych v podkapitolách 4.3.1 až 4.3.11 uvedl hlavní body, které programátorům zjednoduší jejich práci v podkapitole 4.3.12 je pak uveden postup instalace frameworku ASP.NET MVC 3 do programu Visual Studio 2010.

Následné kapitoly prakticky ukazují novinky, které přináší verze 3.0 frameworku ASP.NET MVC. Jednotlivé příklady ukazují jakým způsobem lze těchto rozšíření a úprav využít k zefektivnění a zlepšení práce a vysvětlují na prostých ukázkách, jak lze začít s jejich využíváním.

4.3.1 Rozšíření nástrojů pro automatické generování kódů

Nástroje pro automatické generování kódu (Anglicky tzv. Scaffolding), umožňují programátorovi, na základě daného modelu, předgenerovat datová a editační pole do vytvořených pohledů, který je pak možné podle vlastních potřeb upravit a přeprogramovat, aby odpovídal specifikacím a požadavkům aplikace. Tato funkce tak napomáhá hlavně v ASP.NET MVC začínajícím programátorům, kteří se na základě vygenerovaných základních formulářů mohou inspirovat a vytvářet své vlastní. Tato vlastnost není pouze pro začátečníky, i profesionálům může poskytnout rychlejší start při generování nových pohledů k modelům. (ASP.NET, 2011)

Pro MVC s příchodem verze 3.0 vznikl balíček instalovatelný přímo do prostředí Visual Studia 2010 (tzv. NuGet) nazvaný MvcScaffolding. Balíček obsahuje generované kódy pro několik příležitostí: (ASP.NET, 2011)

- Pro začátečníky s ASP.NET MVC - poskytuje kódy, které dokážou funkčně ukázat, jak se s MVC pracuje a umožní projít si první krůčky úprav aplikace.
- Pro pokročilé, zkoumající nové další ASP.NET MVC technologie.
- Pro znovu se opakující práci, kupříkladu vytváření podobných tříd či souborů různého druhu - MVC 3 umožňuje i tvorbu vlastních generovaných kódů, které pak za vás odvedou potřebnou rutinní práci, navíc existuje možnost je rozšířit i do celého pracovního týmu a usnadnit tak práci i dalším kolegům.

4.3.2 Šablony pro HTML 5

Obecně známý je mezi programátory fakt, že na starších prohlížečích vznikají velké problémy s používáním nových HTML 5 a CSS 3 elementů, čímž vzniká problém, zda se omezit na jednodušší a hůře vypadající výstup aplikace, zavrhnout staré prohlížeče s dodatkem, že si uživatel má zvolit a nainstalovat verzi novější, či věnovat nemalé úsilí, nehledě na finanční náročnost na optimalizaci a detekci verzí prohlížečů a omezení vzhledu na verzi prohlížeče, která zrovna stránky požaduje. (ASP.NET, 2011)

Tento problém, v současné době pomocí jQuery knihovny Modernizr 1.7, řeší nové šablony pro HTML 5 obsažené v ASP.NET MVC 3 Frameworku.

4.3.3 Multiple View Engines

ASP.NET MVC 3 podporuje pro jeden projekt tvorbu pohledů v rozdílných View Engines, který je odpovědný za převedení naprogramovaných pohledů do HTML. Pohledy využívají směsici HTML a části programového kódu sloužící k vyjádření datových proměnných, generování řádků tabulky z listu položek z modelu atp. (ASP.NET, 2011) Například:

```
<div class="inner">
  <%
  string BookmarkUrl = Server.UrlEncode("http://" + Request.Url.Authority +
Model.CurrentItem.Url);
  string BookmarkTitle = Server.UrlEncode(Model.CurrentItem.Title);
  %>
  <%foreach(var bookmark in Model.Bookmarks){%>
    <a href="<%=String.Format(bookmark.UrlFormat, BookmarkUrl,
BookmarkTitle)%>">
      <img src="<%=ResolveUrl("~/Content/Img/" + bookmark.ImageName) %>"
alt="<%=bookmark.Text%>" />
      <%=Model.CurrentItem.ShowText ? bookmark.Text : null%>
    </a>
  <%}%>
</div>
```

Kód 3 - View engine mix kódu a HTML

Kód 3 ukazuje jak se dají uložit řetězce do deklarovaných proměnných, které jsou definované, nebo například jak vytvořit cyklus, kterým se vypíše odkaz a obrázek s textem pro každou záložku (BookMark), která je součásti modelu. HTML kód se míchá s tagy `<%` … `%>`, které deklarují kódovou část, tedy vypisování proměnných a jednoduchých funkcí.

MVC 3 tak umožňuje využít, jak tento klasický zápis pocházející z WebForm syntaxe, tak i zápisy jiné, důležité je to pro syntaxi Razor, která je jednou z nejdůležitějších inovaci ASP.NET MVC 3 a dozvíme se o ní později v této kapitole.

4.3.4 Vylepšení Řadiče (Controller)

Logika řadiče byla vylepšena zejména pro větší komfort uživatelů a pro zjednodušení práce s následujícími prvky (ASP.NET, 2011):

4.3.4.1 Globální "action filters"

V některých případech je zapotřebí spustit logický blok funkcí před tím nebo po tom co se spustí metoda obsluhující akci uživatele. V MVC 2 vznikly za účelem obsluhy těchto funkčních bloků tzv. "action filters". Jednalo se o atributy, které deklarovali chování před a po akční metodě. Existují však případy, kdy programátor potřebuje, aby se určité chování před a po metodě aplikovalo na všechny druhy akcí řadiče. Za tímto účelem vznikla možnost deklarovat filtry globální, uložené v "Global Filters" kolekci. (ASP.NET, 2011)

4.3.4.2 Nový parametr "ViewBag"

Zlepšení, které nemění vlastní logiku programování v MVC, ale pouze zjednodušuje dříve používané ukládání proměnných do proměnné "ViewData", za účelem jednoduší deklarace. Přiřazování proměnných v předchozích verzích MVC probíhalo pomocí jednoznačně definované třídy ("ViewData"), bylo zapotřebí stanovit, jaké proměnné se budou odesílat a pro jaké pohledy `ViewData["Text"]="text"`. (ASP.NET, 2011)

Proměnná "ViewBag" je dynamicky generovaná a pro správné dosazení proměnné do pohledu, stačí využít pouze jednoduchou syntaxi `ViewBag.Text="text"`. Protože se jedná o dynamickou proměnnou, je možné přiřazovat vlastnosti objektu a ten je dynamicky obslouží za běhu. Proměnná "ViewBag" je ve své podstatě párová hodnota proměnná/hodnota, uložená přes vnitřní kód do "ViewData", představuje tak pouze komfortnější užití této proměnné.

4.3.4.3 Nové typy "ActionResults"

"Action Results" je označení metod, které ošetřují akci uživatele, mají na starost zjištění, zda uživatel požádal o další pohled nebo zda volá metodu, která upravuje data v databázi, odesílá formulář a podobně. Základní chování, tedy načtení nového pohledu, se provádí přes "HTTP GET", pro ostatní chování je nezbytné v kódu pohledu deklarovat definici o jakou formu požadavku se jedná. Pro ukázku Kód 4 deklaruje metodu pro ošetření "HTTP POST". (ASP.NET, 2011)

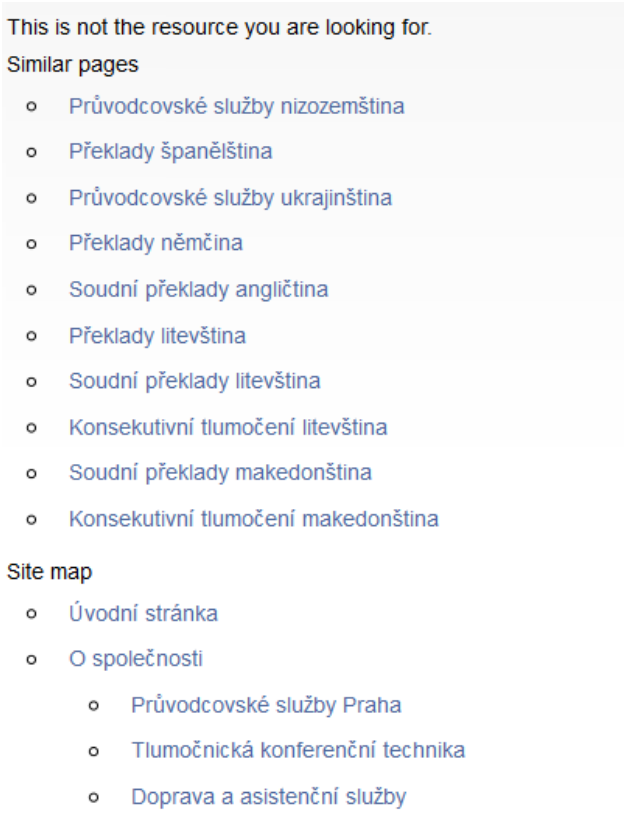
```
[HttpPost]
public ActionResult Index(Models.ViewModels.RealEstateViewModel model)
{
   // Kód metody
}
```

Kód 4 - Ošetření akce POST

V MVC 3 došlo k rozšíření počtu druhů "ActionResult" o následující (ASP.NET, 2011):

- **HttpNotFoundResult**

Umožňuje ošetřit případ, kdy aplikace vrátí uživateli chybu 404, tedy stránka nenalezena a umožní programátorovi deklarovat, jaký pohled se uživateli odešle a tím nedeklaruje pouze hloupou prázdnou stránku 404, která je v dnešní době většinou deklarována jako čisté HTML, umožňující pouze návrat na stránku předchozí, ale programátor nabídne například možnost napovědět uživateli, jakou stránku se pokusil najít pomocí vyhledávání, či například jen mapy stránek, jak je ilustrováno chováním aplikace na nesmyslnou URL na Obr 6. (ASP.NET, 2011)



Obr 6 - chování aplikace ILTS.cz (vytvořené autorem práce) při nesmyslném požadavku na stránku (404)

RedirectResult

Pro potřeby přechodného (HTTP 302) či permanentního (HTTP 301) přesměrování, vznikl další "ActionResult", v návaznosti na to vznikla série nových funkcí pro přesměrování, k předchozímu "HttpResponse.Redirect('TargetURL')" přibyly "HttpResponse.RedirectPermanent", "HttpResponse.RedirectToRoutePermanent" a "HttpResponse.RedirectToActionPermanent". (ASP.NET, 2011)

- *RedirectPermanent*

Přesměrování poskytuje HTTP status 301 a prohlížeči předává údaje, pro přesměrování na novou URL. Volání této metody ukončuje obsluhu stávajícího požadavku a přechází přímo na novou URL. (ASP.NET, 2011)

- *RedirectToRoutePermanent*

Umožňuje navíc k přesměrování relativní cesty (tedy nikoliv kompletní změny URL) i parametry URL, tedy proměnné obsažené v sekci PARAMS požadované URL, pro ukázku dotaz na tuto stránku *www.url.cz/Product?productId=1&category=widgets*, může v řadiči být řešen voláním metody Kód 5. (ASP.NET, 2011)

```
Response.RedirectToRoutePermanent("Product", new RouteValueDictionary
{productId="1", category="widgets"});
```

Kód 5 - Volání metody Response.RedirectToRoutePermanent

- *RedirectToActionPermanent*

Poslední nové přesměrování, nepoužívá klasické přesměrování na jinou URL, ale umožňuje programátorovi, při setrvání na stránce, zavolat jinou akci řadiče, což lze prakticky využít pro případné odeslání jiné formy dat do stejného pohledu, či případně, v závislosti na akci, volat jiný pohled se stejnými daty. I toto přesměrování obsahuje přetížení pro parametry URL, stejně jako v RedirectToRoutePermanent. (ASP.NET, 2011)

- **HttpStatusCodeResult**

Předchozí "ActionResults" poskytovali možnost deklarování, či ošetřování nejběžnějších HTTP statusů, pro ostatní možné lze deklarovat návratový status, který uživateli specifikuje, proč stránku nelze zobrazit. (ASP.NET, 2011) Pro příklady nejvyužívanějších statusů viz Obr 7.

Code	Meaning	Code	Meaning
200	OK	401	Unauthorised
201	Created	403	Forbidden
204	No Content	404	Not Found
206	Partial Content	410	Gone
301	Moved Permanently	411	Length Required
302	Moved Temporarily	500	Internal Server Error
303	See Other	501	Not Implemented
304	Not Modified	502	Bad Gateway
400	Bad Request	503	Service Unavailable

Obr 7 - Nejpoužívanější HTTP status kódy (Virgin_media, 2009)

4.3.5 Vylepšení užití jazyka Javascript a asynchronního volání AJAX

Pro výrazné zjednodušení programování, přináší MVC 3 zlepšení v oblasti přístupu k validaci a asynchronnímu volání obsahu stránek AJAX. Nový přístup zavádí velmi nevtíravý způsob deklarace, snaha je se vyvarovat vkládání javascript kódu přímo do HTML, čímž se stává čistším a i přenosově menším. MVC 3 také v základu poskytuje validaci prostřednictvím knihovny *jQueryValidate*, která zjednodušuje validaci, jak na klientově straně, tak i na straně serveru. (ASP.NET, 2011)

4.3.5.1 Klientská validace

V základním nastavení je v MVC 3 klientská validace zapnutá pro každé deklarované pole formuláře, vše lze změnit nastavením aplikace v souboru *web.config*. Je ale nezbytné mít v hlavičce aplikace deklarované knihovny pro validaci jQuery, aby validace fungovala. (ASP.NET, 2011)

4.3.5.2 Vzdálená serverová validace

Velmi přínosné v této oblasti je využití podpory jQuery knihovny *jQueryValidate*, pro vzdálenou validaci na serveru, funguje asynchronním způsobem. Validátor se na základě vloženého řetězce dotáže serveru na určitou funkci a ten ověří validitu vstupu. Definice názvu volané funkce se nachází přímo v modelu a to použitím prefixu [Remote("CheckAvilability", "Emails")], kde řetězec "CheckAvilability" představuje název funkce v řadiči (controller) a řetězec "Emails" specifikuje název řadiče. Funkce vzdáleně volaná, pak na základě vstupního řetězce, ověří validitu a vrací boolean hodnotu true nebo false, která vypovídá o validitě hodnoty. (ASP.NET, 2011)

4.3.5.3 Podpora JSON

MVC 3 napomáhá jednoduchému dekódování objektu JSON, odesílanému jako požadavek na stranu serveru. V předchozích verzích bylo zapotřebí objekt serializovat, aby mu logika řadiče porozuměla, nyní stačí dodržení názvů proměnných, jak na straně javascritpu, tak definovaného objektu. O logiku serializování a dekódování JSON objektu se stará logika na pozadí MVC frameworku. (ASP.NET, 2011)

4.3.6 Vylepšení serverové validace na základě modelu

MVC ve verzi 3.0 umožňuje modelu deklarovat přímo prefixem, jakým způsobem hodnoty validovat. Slouží k tomu kupříkladu prefix [Required], ukládající parametru povinnost zadání při vytváření instance modelu, dále pomocí parametrů umožňuje i přímé porovnání dvou hodnot, kupříkladu je to vhodné jak ukazuje Kód 6, pro validaci hesla a jeho opakování pro potvrzení. (ASP.NET, 2011)

```
public class User
{
    [Required]
    public string Password { get; set; }
    [Required, Compare("Password")]
    public string ComparePassword { get; set; }
}
```

Kód 6 - využití deklarace prefixu modelu pro validaci hesla s opakováním (ASP.NET, 2011)

4.3.7 Razor View Engine

ASP.NET MVC 3 přichází s novým nástrojem, který výrazně zjednodušuje programátorům psaní pohledů pro jednotlivé aplikační modely. Je jím Razor View Engine, v kapitole 4.3.3 jsme rozebírali jeho podporu pro různé projekty spolu s dalšími View Engine syntaxemi. Razor je nativně zařazen vedle klasické Webform syntaxe (koncovka pohledu .aspx) a vytváří novou syntaxi, psanou v souborech s novou koncovkou (.cshmtl a .vbhtml), která je spojením aplikačního kódu (C#, Visual Basic) a HTML. Razor svou syntaxí výrazně zjednodušuje přehlednost zápisu jednotlivých webových komponent. (ASP.NET, 2011)

```
<div class="editor-label">
      <label for="userName"><%=GetLocalResourceObject("UserName") %>
</label>
</div>
<div class="editor-field">
      <input id="userName" name="userName" class="tb" />
</div>
```

Kód 7 - Div s popiskem a textovým polem pro Login v klasické Webform syntaxi

Jak vidíme v Kód 7, v klasické syntaxi se pro zapsání využívají standardní HTML značky label a input. Po odeslání formuláře jsou identifikovány v řadiči podle id vlastnosti elementu.

```
<div class="editor-label">
       @Html.LabelFor(m => m.UserName)
</div>
<div class="editor-field">
       @Html.TextBoxFor(m => m.UserName)
       @Html.ValidationMessageFor(m => m.UserName)
</div>
```

Kód 8 - Stejný kód jako v Kód 7 zapsaný v Razor syntaxi

Kód 8 ukazuje rozdílný přístup v Razor syntaxi. Je zde využíván jmenný prostor Html, který obsahuje jednotlivé HTML prvky, jakými jsou právě textová pole, rozbalovací seznamy a mnoho dalších. Uvedený Kód 8 navíc přidává i aktivní klientskou validaci, která využívá vylepšení zmíněné v kapitolách 4.3.5 a 4.3.6.

Toto srovnání ukazuje, že Razor syntaxe je čistší a přehlednější a programátorovi ušetří svou jednoduchostí čas. Pro ještě větší pohodlí je ve Visual

Studiu podporován tzv. IntelliSense, který napovídá jednotlivé části kódu, přímo při jejich psaní a zkracuje tak čas potřebný k psaní kódu.

4.3.7.1 Některé zlepšení funkcionality, které přináší Razor

- @model - Specifikuje typ datového objektu, který je předáván pohledu.
- @* *@ - Zjednodušená tvorba komentářů
- Specifikace základního rozložení (dříve Templates) tzv. Layout, tedy rozložení základního rámce stránek (např. nahoře menu, napravo panel novinek a hledání), zbytek pak již deklarují jednotlivé pohledy.
- Html.Raw - Funkce, která vypisuje text zapsaný v HTML. Vhodné pro ukládání některých naformátovaných odstavců a delších textů, které obsahují již v databázi HTML značky. Html.Raw značky rozkóduje a do stránky vloží text tak, jak má být zobrazený.
- Podpora pro sdílení kódu mezi více pohledy (soubory _ViewStart.cshtml nebo _ViewStart.vbhtml) (ASP.NET, 2011)

4.3.7.2 Nová funkcionalita přinesena Razor View Engine

Razor, kromě již zmíněné funkcionality, zjednodušuje stávající zápis pohledů a přináší i nové objekty z třídy HtmlHelper. Jedná se o syntaktické elementy jakými jsou v Kód 8 textboxy a validatory. Jedná se prakticky o zástupné znaky pro zápisy jednotlivých tag elementů. Razor přináší tyto kompletně nové (ASP.NET, 2011):

- Chart - Vytváří grafy podobně jako v ASP.NET 4.0 prvek Chart.
- WebGrid - Generuje datovou tabulku se stránkováním a řazením datových záznamů.
- Crypto - Využívá tzv. Hashování k uschování hesla či podobných řetězců, které je třeba zabezpečit.
- WebImage - Generuje přiřazený obrázek.
- WebMail - Odesílá e-mailovou zprávu.

4.3.7.3 Dynamický obsah v pohledu s Razor syntaxí

K tvorbě úspěšných webových aplikací je nezbytné, využít kromě statického HTML kódu a statických proměnných, také dynamický obsah, který je generovaný za běhu a může být různý pro každý jednotlivý požadavek.

Do stránek je možné přidat dynamický obsah následujícími způsoby (Freeman, a další, 2011):

Tabulka 1 - Přidání dynamického obsahu do pohledu

Technika	Kdy ji použít
Součást kódu pohledu	Používá se zejména pro malé samostatné bloky kódu vepsané mezi HTML pohledu. Nejlepším příkladem je podmínka "if" nebo cyklus "foreach". Toto jsou základní bloky aplikačního kódu v pohledu. Ostatní přístupy níže částečně staví na těchto základech.
Metody HTML helperu	Používá se pro jednotlivé elementy nebo jejich malé kolekce, typicky založené na modelu pohledu nebo datových podkladů pohledu. ASP.NET MVC 3 framework obsahuje řadu těchto HTML metod, a je relativně snadné si vytvořit svůj vlastní. Každá metoda, která navrací hodnotu ve formátu MvcHtmlString, může být metodou HTML helperu.
Částečné pohledy	Použití je vhodné zejména pro opakující se bloky pohledů mezi jednotlivými stránkami. Může obsahovat také aplikační kód z bodu jedna, využívat jednotlivé HTML helpery a také může odkazovat na další částečné pohledy. Částečné pohledy implicitně nespouští akci řadiče, takže nespouští aplikační logiku, tuto logiku je však možné volat využitím volání URL akce.
Následnící akcí	Užití je vhodné v případě, že vytváříme ovládací prvek nebo komponentu, kterou zamýšlíme využívat opakovaně na různých místech a potřebujeme aby spustila aplikační logiku. Pokud použijeme následníka akce (child action), spouští se metoda akce řadiče, vykreslí se pohled a tento výsledek se přidá k odpovědi ze serveru.

4.3.8 Použití aplikační logiky jako programový kód v pohledu (Inline code)

Součástí HTML syntaxe může být i programový kód, například pro kód 9 bychom v ASP.NET využili komponentu asp:Repeater, kterou bychom museli napojit na datový zdroj v code behind nebo případně použitím další komponenty z palety Data Source. (ASP.NET, 2011)

V případě Razor syntaxe je možné využít volání cyklu foreach, jak je ukázáno v Kód 9.

```
<div class="realEstateGallery topBorder botBorder left topPadding
botPadding">
@foreach (Dinamico.Models.Pages.GalImage gi in Model.RealEstate.GalImages)
{
   <a class="grouped_elements" rel="gallery" href='@gi.Image'
target="_blank">
       <img src='@gi.PreviewImage' style="height:86px" alt="@gi.Text"
class="GalImg" />
   </a>
}
</div>
```

Kód 9 - Použití foreach pro tvorbu galerie obrázků

Foreach v tomto případě prochází jednotlivé elementy druhu GalImage uvnitř pole GalImages, která je součástí datového modelu pohledu. Tedy přímo součástí dat o realitě. Poté je vygeneruje i s odkazy (vhodné pro další využití kupříkladu s komponentou jQuery Fancybox).

4.3.9 Metody HTML helperu (HTML helper methods)

Přístup k základním elementům jakými jsou tagy , <a>, <p>, a podobné je řešen klasickými HTML pravidly. Rozdíl však nastává při užití aktivních prvků stránky, formulářů, polí atp. Na příkladu Kód 10 je ukázáno, jakým způsobem lze využít syntaxi Razor a jmenný prostor "Html" pro vytvoření rozbalovacího seznamu (DropDownListFor).

```
<div class="localitySelector">
<div>
@{
     Dinamico.Models.Pages.RealEstateHolder reh =
(Dinamico.Models.Pages.RealEstateHolder)Model.RealEstate.Parent.Parent;
}
@Html.DropDownListFor(x => x.Localities, new
SelectList(Model.Localities.OrderBy(x => x.ID) as
System.Collections.IEnumerable, "Id", "Title"), reh.chooselabel, new {
@class = "localitySelectDetail", onchange = "window.location.href='" +
Model.RealEstate.Parent.Parent.Url + "?locationID=' + $(this).val()" })
</div>
<div class="localSelected">@Model.RealEstate.Parent.Title</div>
</div>
```

Kód 10 - ukázka využití Razor DropDownListFor

Využitím aplikace založené na webform principech bez MVC by pro deklarování takovéhoto seznamu bylo zapotřebí vložit komponentu <asp:DropDownList> s parametry a z tzv. CodeBehind, tedy aplikačního kódu, jej naplnit daty z databáze. V případě využití ASP.NET MVC 3 a Razor View Engine si postačí pouze s modelem a jeho parametry. Na začátku příkladu si do proměnné reh ukládá model RealEstateHolder, který slouží jako složka lokalit a v každé lokalitě se nacházejí nějaké reality.

Pro datové vrstvení proměnnou RealEstateHolder nazýváme holder nebo lze použít název container. To však záleží na datovém konceptu aplikace. V další části již potom deklarujeme samotný rozbalovací seznam. První parametr definuje formulářový výraz. Druhý parametr `x => x.Localities` deklaruje, jaký datový seznam z modelu pro zobrazení využít, tedy lokality uložené v modelu ve formě pole párových hodnot ID a Title. Třetí proměnná je brána také z databáze a jedná se o první vybranou hodnotu (kupříkladu se využívá "--- prosím vyberte ---" apod.) a na závěr se zde deklaruje objekt HTML attributů, které bude výsledný "input" typu

"select" mít deklarované. Tedy, v tomto případě, třídu kaskádových stylů CSS a funkci, která se provede při výběru hodnoty (javascript).

Tento příklad efektivně ukazuje, jak jednoduše na jednom místě, se dá deklarovat celý funkční celek, bez nutnosti provádět kód na straně serveru, který by se staral o naplnění seznamu daty, ošetřoval funkci, která se volá při volbě hodnoty políčka. Vše je jednoduše přístupné na jediném místě a odpadá tak nutnost pátrat po případných chybách.

4.3.10 Částečné pohledy (Partial views)

Ukázkový Kód 11 je sdílený oddíl patičky, je to částečný pohled, který v sobě skrývá další částečný pohled (TopNavigation). V první sekci si vytahuje z databáze úvodní stránku, v podstatě container celého webu v N2 Dinamico CMS systému. Z něho pak získává odkazy na FB a Twitter.

```
<div class="inner">
@{
    Dinamico.Models.StartPage sp =
(Dinamico.Models.StartPage)N2.Find.StartPage;
}
<div class="iconFB">
  <a href="@sp.FBLink" title="Facebook">
     <img src="/Upload/facebook_32.png" alt="Facebook" />
  </a>
</div>
<div class="iconTwitter">
  <a href="@sp.TwitterLink" title="Twitter">
     <img src="/Upload/twitter_32.png" alt="Twitter" />
  </a>
</div>
<div class="created">
  made by
  <a href="http://develion.cz" title="Develion">
     develion<img src="/Upload/develion-footer-logo.jpg" alt="Develion">
  </a>
</div>
@{ Html.RenderPartial("LayoutPartials/TopNavigation"); }
<div class="cleaner"></div>
</div><!-- /.inner -->
```

Kód 11 - Použití vnořeného částečného pohledu pro patičku (footer)

4.3.11 Vytváření datového modelu

Model v MVC architektuře je nejdůležitější částí aplikace a reprezentuje objekty z reálného světa, procesy a pravidla, která definují podstatu toho, co model zastupuje. Model obsahuje C# třídy, které vytvářejí základní definují datový základ naší aplikace a metody, které umožní aplikaci s nimi manipulovat. Pohledy (Views) a Řadiče (Controllers) poté zobrazí data modelu klientovi jako celek. Základem dobře vytvořené aplikace je správné definování modelu, který je pak počátečním bodem k tvorbě pohledů a řadičů. (Freeman, a další, 2011)

4.3.11.1 Přidání třídy modelu (Model)

Konvence MVC určuje přidávání modelových tříd do složky Models v rootu aplikace, dodržením této konvence je i vytvoření složky Models do aplikační podložky naší části aplikace, to se využívá zejména, pokud tvoříme aplikaci na základě nějakého druhu komplexnějšího CMS systému, kupříkladu N2CMS. Ve Visual Studiu pravým kliknutím na složku Models vybereme přidání třídy, zvolíme název modelu a potvrdíme přidání. Kód 10 ukazuje jednoduchý příklad tvorby modelu pro logovací stránku aplikace, definuje polička uživatelského jména a hesla.

```
namespace Dinamico.Models
{
      [PageDefinition]
      public class LoginForm : PageModelBase
      {
         public virtual string FieldUserName { get; set; }
         public virtual string FieldPassword { get; set; }
      }
}
```

Kód 12 - Definice modelu pro logovací stránku

Při deklarování modelu je možné využít i validačních omezení pro jednotlivé proměnné, jedná se o omezení stran validity vstupních řetězců viz podkapitola 4.3.6.

4.3.11.2 Vytvoření řadiče (Controller)

Řadičem je další třída, která má na starost formátování modelu před zobrazením v pohledu, ale také odchytávání událostí, které uživatel na své straně spustí a následně pak volání metod pro manipulaci s modelem.

```
namespace Dinamico.Controllers
{
    [Controls(typeof(Models.LoginForm))]
    public class LoginFormController
    {

      public IFormsAuthenticationService FormsService { get; set; }
      public IMembershipService MembershipService { get; set; }
      public ActionResult Index()
      {
            return View(Content.Current.Item);
      }

      [HttpPost]
      public ActionResult SubmitLoginForm(Models.LoginFormViewModel model)
      {
            // Login user
            if (ModelState.IsValid)
            {
              if (MembershipService.ValidateUser(model.FieldUserName,
model.FieldPassword))
              {
                    FormsService.SignIn(model.FieldUserName, false);
              }
            }
            return View(model);
      }
    }
}
```

Kód 13 - Řadič (Controller) pro logovací stránku

Řadič v kódu 13 nejdříve deklaruje s jakým typem modelu pracuje, v tomto případě je to model z podkapitoly 4.3.11.1, tedy model pro logování do systému. Třída řadiče poté obsahuje dvě funkce, obě mají v návratové hodnotě objekt pohledu na model. První funkce nazvaná Index, vrací defaultní pohled bez vyplněných hodnot, po nadefinování pohledu vrátí políčka s tlačítkem pro logování do aplikace. Druhá funkce s prefixem `HttpPost` deklaruje odchycení funkce formuláře pohledu, tedy stisknutí tlačítka pro přihlášení do systému. Od pohledu dostane v parametru upravený model formulářem, tedy model z kapitoly 6.9.1 s vyplněnými hodnotami pro uživatelské jméno a heslo. V zápětí ověří validitu vstupu, tedy zda vyplněné hodnoty splňují klientskou validaci. V ukázce modelu jsme pro jednoduchost

nezadávali validační omezení, takže hodnoty formuláře jsou vždy validní a funkce postoupí k volání `IMembershipService`, která je součástí knihoven .NET, ověří uživatele a vzápětí jej zaloguje zavoláním metody SignIn služby `IFormsAuthenticationService`. Funkce na závěr vrátí uživatele zpět na pohled s modelem s vyplněnými hodnotami, vhodné by bylo využít například ViewBag pro uložení výsledku logování a následné zobrazení uživateli.

4.3.11.3 Tvorba pohledu

Pro kontinuitu aplikace bychom nyní vytvořili pohled pro přihlašovací stránku, nejvhodnějším způsobem pro tvorbu je využití Razor syntaxe, kterou popisuje kapitola 6.2, ve které je také ukázkový kód přihlašovacího pohledu.

4.3.11.4 Přidání validace modelu

Rozšíření modelu o validace je důležitým prvkem aplikace, pomůže nám zabránit uživateli zadávat nesmyslná data do systému.

```
namespace Dinamico.Models
{
      [PageDefinition]
      public class LoginForm : PageModelBase
      {
         [Required(ErrorMessage = "Enter your username")]
         [RegularExpression("[0-9a-zA-Z]", ErrorMessage = "Invalid
username")]
         public virtual string FieldUserName { get; set; }
         [Required(ErrorMessage = "Enter your password")]
         public virtual string FieldPassword { get; set; }
      }
}
```

Kód 14 - Validace modelu

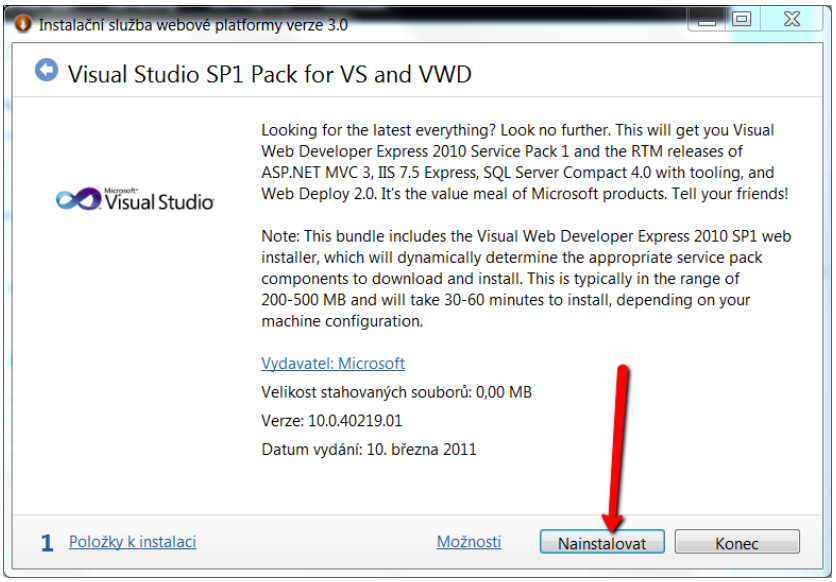
Prakticky tak v Kód 14 přidáváme validaci uživatelského jména, které je povinné pole a je omezeno regulárním výrazem pouze na velká a malá písmena a čísla. Heslo je na vstupu také povinným prvkem. Nyní už řadič v první podmínce odchycení události klienta zjistí, zda je model validní a případně vrátí pohled uživateli s validačním ověřením a chybovými hláškami pro zobrazené v nevalidních polích.

4.3.12 Instalace ASP.NET MVC 3 do Visual studia 2010

O Instalaci ASP.NET MVC 3 se přímo stará webový instalátor na adrese http://www.asp.net/mvc, který spustíme kliknutím na označené tlačítko v Obr 8.

Obr 8 - Spouštění instalace ASP.NET MVC 3 (Microsoft, 2011)

Po stisknutí tlačítka budete přesměrování do instalační části, která se rovnou snaží o spuštění Web Platform Installer 3.0, což je nezbytné potvrdit v nabídce, která po chvilce vyskočí. Pro zabezpečení Windows 7 nebo Windows Vista budete dotázání zda chcete Installer opravdu spustit. Po jeho spuštění se vám po chvilce nabídne okno Obr 9, které nabízí instalaci potřebného servisního balíčku pro Visual Studio obsahující ASP.NET MVC 3.



Obr 9 - Instalace Service pack, jehož součásti je i podpora ASP.NET MVC 3 (Microsoft, 2011)

5. Pravidla tvorby ASP.NET MVC aplikací

Při tvorbě aplikací s využitím architektury MVC, existuje riziko nesprávného použití některých jejich částí. Je nezbytné řídit se několika pravidly, pro její co nejefektivnější využití, aby se zajistilo, že opravdu bude plnit roli směřující k zvýšení efektivity, zpřehlednění kódu a zjednodušení práce programátorů.

Nejdříve je potřeba si uvědomit prvotní účel webových aplikací, tím je zobrazení dat do HTML struktury, aby je bylo možné uživateli zobrazit. Tento samotný proces není jednoduchý a proto je důležité zvolit správnou HTML strukturu, která nebude snadná na prohledávání a lehce se v ní budou hledat případné nesrovnalosti.

5.1 Eliminování nepotřebných částí kódů v pohledu

Na začátek je nezbytné nevytvářet přehnané HTML struktury, které by zatěžovali server při přístupech. Je zapotřebí říct, že některý kód do pohledu patří, kupříkladu pro cyklus, který vytváří tabulku, případně využití podmínek pro zobrazování některých elementů jen v případě, že je daná podmínka splněná (přihlášení, role atp.). Ale co by se v pohledech objevit nemělo jsou například úpravy formátů řetězců, nebo parsování proměnných, toto jsou aspekty, které by měly být ošetřeny v modelu.

5.2 Využívání Master Page

Master page umožňuje uložení obalového kódu do jednoho souboru a není tak nutné pro každý specifický pohled vytvářet obrovský soubor s HTML tagem title,head, body atp. Template page slouží k umístění neměnného kódu stránky. Kupříkladu hlavička se scripty nebo základní rozložení stránky.

5.3 Využívání HTML Helpers

HTML Helpers jsou nejjednodušší cestou, jak zobrazit a umožnit editaci jakýchkoliv dat uložených v databázi. ASP.NET MVC umožňuje programátorovi využít, ke všem potřebným úkonům, klasické HTML, ale využití HTML Helperu výrazně zkrátí a zjednoduší práci.

HTML Helpers také výrazně omezí potencionální zmatek v syntaxi pohledů (Views). ASP.NET MVC Framework obsahuje několik exaktních helperů, které

umožní zobrazit různé datové typy jednotlivých databázových proměnných, udrží se tím jednoduchost a přehlednost pohledů a případné hledání chyby zabere menší objem práce, což má za následek zvýšení efektivity práce.

5.4 Používání integrované validace

Validace pro jednotlivé formuláře v projektu je možné validovat na straně klienta více způsoby, kupříkladu prostřednictvím jQuery knihovny *jQueryValidate.js*, ale správnou deklarací modelu a přesným typovým rozdělením jednotlivých proměnných, lze dosáhnout výrazné úspory času. Vestavěná validace ve frameworku ASP.NET MVC nedovolí uživateli odeslat formulář bez správně a validně vyplněných povinných políček, čímž se eliminuje potřeba jejich ověřování na straně serveru, což vede k úspoře času programátora a také k úspoře výpočetního výkonu serveru.

5.5 Zabezpečení pohledů

Z hlediska bezpečnosti se o velkou část ochrany stará samotný framework, bez výrazné potřeby zasahovat do jeho běhu, avšak je potřeba zamezit případnému ohrožení ze strany skriptovacích útoků.

Skriptovacím útokem je myšleno využití formulářového k políčka k odeslání potencionálně nebezpečného skriptu, který kdyby se na serveru provedl, mohl by odeslat utajené informace, či dokonce narušit chod aplikace.

Z technického hlediska není potřeba ověřovat veškeré vstupy, především ASP.NET prověřuje v základu všechny vstupy přicházející na server, avšak nebezpečí mohou skrývat přenosy nezakódovaného HTML, kupříkladu z velkých editovacích políček, které umožňuje strukturaci obsahu pomocí HTML. ASP.NET MVC pro takové políčka obsahuje HTML Helper s názvem Html.Encode, příklad použití na Kód 15.

```
<%= Html.Encode(Model.Message) %>
```

Kód 15 - Využití helperu Html.Encode

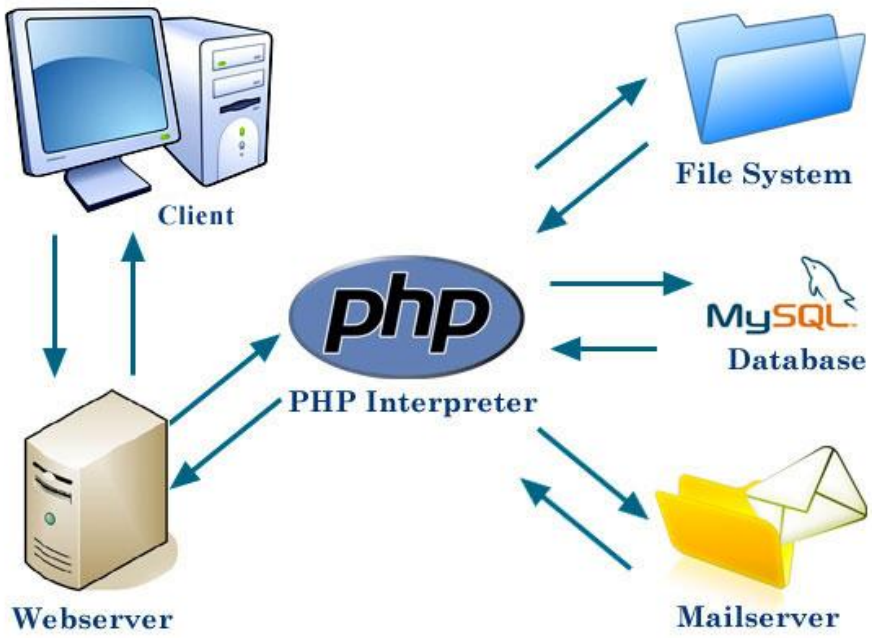
6. Porovnání interpretovaného jazyka Razor s PHP

Tato kapitola porovnává syntaxi PHP se syntaxi jazyka Razor, v první podkapitole je popsáno PHP. Jelikož je tato práce cílená hlavně na programátory se zkušenostmi s ASP.NET, jsou uvedeny základní vlastnosti PHP a jeho jednoduchá syntaxe a následuje porovnání s jazykem Razor.

6.1 Základní vlastnosti PHP

PHP je programovací jazyk, pracující na straně serveru, jeho prostřednictvím lze vytvářet a dynamicky měnit obsah webových stránek. V době svého vzniku, rok 1996, byl původně pojmenován Personal Home Page (PHP), ale postupem vývoje se přešlo k obecnějšímu názvu Hypertext Preprocessor, došlo také k významným změnám a vývoji v tomto jazyku. (JakPsátWeb, 2009)

PHP má podobně jako platforma .NET široké spektrum možností, jak názorně zobrazuje obrázek funkcionality Obr 10.



Obr 10 - Rozsah fukcionality PHP (PHP & MySQL Tutorial)

6.1.1 Využitelnost jazyka PHP

S PHP je relativně s nízkými náklady, možné vytvořit jednoduché aplikace typu diskusních fór, anket atp. PHP však lze využít i k tvorbě celých komplexních portálů, obsahujících mnoho komponent a funkcí .

PHP je relativně jednoduchý jazyk a v dnešní době existuje jako nadstavba nad tento základ i mnoho a mnoho placených, ale i openSource Frameworků a prostředí, která vám pomohou vytvořit rychle a jednoduše portál dle představ zákazníka. Vhodným příkladem je pro výše zmíněná fóra projekt PHP BB, jehož instalací na server a nastavením prostřednictvím administrace získáte diskusní fórum s možností tvorby místností, registrací uživatelů, jejich hodnocení a podobné funkcionality s jejichž tvorbou by i zkušený programátor strávil velké množství času.

6.1.2 K čemu PHP?

PHP projekty jsou ve velkém množství případů šířené formou Open Source, tedy bezplatně, a jednoduše lze získat Best Practice řešení pro webové portály. Pro začínající programátory je tento přístup velmi přínosný a procházením a učením se přímo z kódu získají potřebné znalosti pro práci s tímto jazykem.

Při současném vývoji aplikační logiky, a zejména pak při pohledu na vývojové trendy webových aplikací, tak vzniká situace, kde i zkušení programátoři se ve své podstatě pouze upravují kód jednotlivých Open Source řešení a uzpůsobují si je pro své potřeby a pro potřeby jejich zákazníků. Tento trend, ačkoliv degraduje uměleckou práci programátorů, je však velmi přínosný pro optimalizaci systémů a také pro zrychlení vývoje aplikační logiky. Pravdou v této době je, proč něco vymýšlet znovu, když už to někdo vymyslel a vytvořil a s velkou pravděpodobností i lépe než by se to povedlo vám.

6.1.3 Nároky PHP

Systémové a Hardwarové nároky pro vývoj a provoz PHP aplikací jsou při dnešním výkonu Hardware naprosto minimální. Provoz aplikace je zajištěn v základu webovým serverem a knihovnami PHP. Jako serveru lze využít volně stažitelný server Apache, nainstalovat a nakonfigurovat na něj knihovny a je možné začít s vývojem aplikace. Opět zde však platí pravidlo, proč dělat věci složitě když to jde

jednoduše, například pomocí programu PHPTriad, který je volně ke stažení a postará se o kompletní instalaci knihoven i serveru. (JakPsátWeb, 2009)

6.1.4 Webhosting

Pro hostování aplikace, je zapotřebí, mít možnost provozovat server apache, bez většího kolísání dostupnosti, je také zapotřebí zajistit serveru veřejnou IP adresu, zakoupit doménu a na zmíněnou IP adresu přesměrovat její DNS záznam. Pro počáteční pokusy s prvními fóry a osobními stránkami, postačuje pro takovéto hostování aplikace starší počítač umístěný tak, aby jeho nonstop provoz neobtěžoval a je možné první PHP aplikace zveřejnit světu. (JakPsátWeb, 2009)

Kromě možnosti vytvoření vlastního serveru a vlastní provozování hostingu, existuje i mnoho poskytovatelů této služby, kteří v některých případech poskytují základní PHP hosting zadarmo s podmínkou umístění reklamního banneru na hostované stránky. Omezení těchto řešení zdarma bývá často v malém prostoru v databázi a také na diskovém úložišti serveru. Pro osobní portál nemusí být tento problém znatelný, avšak pro diskusní fóra už může být potřeba velikosti Databáze mnohem vyšší, než kolik je poskytovatel ochoten uvolnit zdarma, navíc také reklamní banner by mohl vadit zákazníkovi a není vhodný k firemním webovým aplikacím. Poplatky za poskytnutí rozšířeného hostingu se pohybují v rozmezí 25 až 250 Kč měsíčně (JakPsátWeb, 2012) v závislosti na poskytované podpoře, diskovém a databázovém prostoru, garantované dostupnosti atp.

6.1.5 Syntaxe PHP a porovnání se syntaxí C#

Interpretování kódu PHP, se provádí uvnitř takzvaných oddělovačů, tedy znaků, které oddělují kód PHP od zbytku, tedy HTML, javascript atp. Nejpoužívanějším oddělovačem je "<?php" jako počáteční oddělovač a "?>" pro jeho ukončení. Dalším často užívaným typem je HTML tag " <script language="php"> /*KÓD PHP*/ </script>", využívají se i zkrácené formy "<?" a "<?=" ukončené "?>", stejně jako je možné použít i klasické ASP oddělovače "<%" a "<%=" s koncovým "%>".

Proměnné v PHP jsou označený znakem dolaru $ a jejich názvy jsou "case sensitive", tedy citlivé na velká a malá písmena, jejich zápis musí být vždy stejný jako při deklaraci.

Vlastní deklarace funkcí je velmi podobná syntaxi užívané v ASP.NET C#, pro ilustraci Kód 16.

```
PHP:    public static function welcom($name, $surname)
      {
            return "Hello " . $name . " " . $surname . "!";
          }
C#      public static string welcome(string name, string surname)
        {
                 return "Hello " + name + " " + surname + "!";
        }
```

Kód 16 - Porovnání deklarace funkce PHP a C#

Ukázka zobrazuje velmi podobnou syntaxi pro zápis funkce, viditelné jsou drobné rozdíly. Vlastni deklarace PHP určuje, že daný kód je funkcí (klíčové slovo function), namísto toho mají funkce C# deklarovaný typ návratové hodnoty (pro řetězec znaků string), dále proměnné v parametrech funkce není v případě PHP nutné označit typem, mají však vložený prefix $ pro identifikaci že se jedná o proměnné, ve funkcích C# je nezbytné označit jejich datovým typem. Zbývající části kódu jsou prakticky totožné, s výjimkou navazování řetězců, kde u PHP se používá "." a u C# "+".

6.2 Využití interpretovaného jazyka v syntaxi Razor

Podobně, jako u PHP, umožňuje Razor využívat interpretaci částí programového kódu, vnořeného do HTML struktury. Používá oddělovače "@", který může být využit pro vypsání proměnné nebo ve spojení "@{" a ukončení "}" pro větší blok kódu, případně deklaraci funkce. Oproti syntaxi PHP je však použita kombinace programového kódu a syntaxe HTML bez závislosti na umístění oddělovačů.

```
@{ using (Html.BeginForm("SubmitLoginForm", null, new { redirectPath =
Content.Current.Page.Url }, FormMethod.Post, new { id = "mypage_form" }))
        {
         <fieldset class="divided">
             <table cellpadding="0" cellspacing="0">
                 <tbody>
                     <tr>
                       <td>
                           @Html.LabelFor(x => x.MyPageForm.FieldUserName):
                       </td>
                       <td style="width: 250px;" colspan="2">
                           @Html.TextBoxFor(x => x.MyPageForm.FieldUserName)
                           @Html.ValidationMessageFor(x =>
x.MyPageForm.FieldUserName)
                       </td>
                     </tr>
                     <tr>
                       <td>Html.LabelFor(x => x.MyPageForm.FieldPassword):
                       </td>
                       <td>
                           @Html.TextBoxFor(x => x.MyPageForm.FieldPassword)
                           @Html.ValidationMessageFor(x =>
x.MyPageForm.FieldPassword)
                       </td>
                     </tr>
                 </tbody>
             </table>
           <button type="buton" id="send-mypage" name="send">Login</button>
         </fieldset>
        }
```

Kód 17 - Využití Razor syntaxe pro logovací stránku

Kód 17 v praxi ukazuje, jakým způsobem je možné kombinovat sekce kódu a sekce HTML, popřípadě komponent jmenného prostoru Html, obsahující jednotlivé prvky HTML struktury, jako jsou například textové boxy, tlačítka či popisky. Nejdříve v kódu deklarujeme využití formuláře "Html.BeginForm", jehož parametry jsou jméno akce, což je název funkce controlleru, který bude odchytávat akci tohoto pohledu, tedy pokus o přihlášení. Dále název controlleru, v tomto případě je hodnota nastavena na "null", jelikož chceme, aby formulář volal controller dané zobrazené stránky, takže jej není třeba specifikovat. Další parametr deklaruje kam

přesměrovává formulář svůj požadavek, právě zde je deklarováno, že volá sám sebe, proto není potřeba controller specifikovat v dřívějším parametru, hned další parametr potom specifikuje, jakou metodou volání používá (POST/GET). Posledním je pak objekt atributů HTML, kde se dají deklarovat jednotlivé třídy, id a další parametry HTML.

Kombinací HTML a komponent Html jmenného prostoru je pak dotvořen vzhled celého formuláře, a jak je v ukázce vidět, je uvnitř oddělovačů, není tedy předem určeno jestli se jedná o programový kód nebo html, to rozpoznává až knihovna syntaxe Razor, při interpretaci kódu, na základě dotazu uživatele.

Vlastním rozdílem oproti klasickému PHP je práce s daty, zatímco v PHP se dotazy do databáze volají přímo ze souboru, který je vlastně pohledem, v MVC s Razor syntaxí je toto zakázáno ve specifikaci. Samotný kód by mohl dotazovat proměnné z databáze, ale aplikace by pak obcházela specifikace ASP.NET MVC a právě důležité oddělení vrstev pro zobrazení, zpracování a definici dat.

Syntaxe Razor získala, možností interpretování kódu, velmi silnou zbraň pro boj s konkurenčním PHP, společnost Microsoft se neustále snaží potlačit Open Source platformu PHP a prosadit na celém trhu svá řešení postavená na ASP.NET. Její snaha zatím naráží na problematiku rozdílu Open Source a placeného řešení, která spočívá ve finančních nárocích. Mnoho uživatelů a zejména začínajících programátorů zvolí raději platformu s plnou podporou zdarma, než-li placené služby od společnosti Microsoft.

7. Závěr

Architektura MVC spojená s platformou ASP.NET je současným vývojovým trendem a přináší mnoho výhod a nástrojů pro zvýšení produktivity práce programátorů. Zjednodušuje práci a ulehčuje nebo úplně eliminuje rutinní úkony, které jsou potřeba prakticky v každém projektu.

Model View Controller je architektonickým vzorem používaným při vývoji softwaru, kapitola číslo 3 popisuje jak vznikla architektura MVC a pokládá teoretické předpoklady, které vedly ke vzniku frameworku ASP.NET MVC, což je spojení výhod rozdělení architektury MVC do tří aplikačních vrstev s frameworkem ASP.NET. Kapitola číslo 4 ukazuje vývoj frameworku ASP.NET MVC v návaznosti na MVC architektuře a na příkladech ukazuje jakým způsobem je lze v dnešní době využít. Použitím zvolených příkladů je možné aby čtenář této práce využil své poznatky z vývoje webových aplikací na platformě .NET a zkombinoval je s poznatky načerpanými z této diplomové práce k tvorbě základních webových aplikací s použitím frameworku ASP.NET MVC. Tímto způsobem práce splňuje hlavní stanovený cíl.

Při tvorbě aplikací s využitím architektury MVC existuje riziko nesprávného použití některých jejich částí. Je nezbytné řídit se pravidly pro její co nejefektivnější využití aby se zajistilo, že opravdu bude plnit roli směřující k zvýšení efektivity, zpřehlednění kódu a zjednodušení práce programátorů. Poznatky v kapitole číslo 5 čtenáři práce pomohou vyhnout se zásadním chybám a nedostatkům při tvorbě webových aplikaci s použitím frameworku ASP.NET MVC, čímž napomůže k zvýšení efektivity jeho práce. Tato kapitola tak splňuje první stanovený dílčí cíl.

PHP jako relativně jednoduchý jazyk nabízí relativně jednoduchý způsob jak programovat webové aplikace. PHP projekty jsou ve velkém množství případů šířené formou Open Source, tedy bezplatně a jednoduše lze získat Best Practice řešení pro webové portály. Pro začínající programátory je tento přístup velmi přínosný a procházením a učením se přímo z kódu získají potřebné znalosti pro práci s tímto jazykem. Podobně jako u PHP umožňuje Razor využívat interpretaci částí programového kódu vnořeného do HTML struktury. Kapitola 6 porovnává PHP se syntaxi Razor a poskytuje tak perspektivu pro programátory vyvíjející aplikace v jazyce PHP a umožňuje jim identifikovat určitou podobnost a jednoduchost v rozdělení aplikace dle architektury MVC. PHP programátorům je tak prostřednictvím interpretovaného jazyka Razor poskytnuta alternativa využívající a poskytující výhody frameworku ASP.NET MVC. Toto porovnání tak splňuje druhý dílčí cíl této práce. Ačkoliv jak je uvedeno v závěru kapitoly 6, problematika spočívá ve finanční náročnosti spojené s využitím platformy .NET pro webový vývoj.

8. Terminologický slovník

Termín	Zkratka	Význam [zdroj]
Intermidiate Language	IL	Společný jazyk vzniklý kompilací různých programovacích jazyků v ASP.NET [Spuzta, MacDonald ASP.NET 3.5]
Aplikační software		Aplikační software (zkráceně aplikace) je v informatice veškeré programové vybavení počítače (tj. software), které umožňuje provádět nějakou užitečnou činnost (řešení konkrétního problému, interaktivní tvorbu uživatele – např. textový procesor apod.) [http://cs.wikipedia.org/wiki/Aplikační_software]
Model-view-controller	MVC	Model-view-controller (MVC) (někdy také nesprávně označovaná jako Model-2) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. [http://cs.wikipedia.org/wiki/MVC]
dynamické HTML	DHTML	je kombinace technologií používaných ke tvorbě dynamických a interaktivních webových stránek. Těmito technologiemi se většinou myslí HTML, klientský JavaScript, kaskádové styly (CSS) a někdy DOM [http://cs.wikipedia.org/wiki/DHTML]
Internetová informační služba	IIS	Sada internetových služeb vytvořená společností Microsoft za účelem využití ve spojení s Microsoft Windows. [http://en.wikipedia.org/wiki/Internet_Information_Services]
Hypertext Preprocessor	PHP	skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek [http://cs.wikipedia.org/wiki/PHP]
Hypertext Transfer Protocol	HTTP	internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML [http://cs.wikipedia.org/wiki/HTTP]
Uniform Resource Locator	URL	řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu [http://cs.wikipedia.org/wiki/URL]
HyperText Markup Language	HTML	značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. [http://cs.wikipedia.org/wiki/HTML]
Extensible Markup Language	XML	značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat [http://cs.wikipedia.org/wiki/XML]

Asynchronous JavaScript and XML	AJAX	AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. [http://cs.wikipedia.org/wiki/AJAX]
	.NET	.NET („dotnet“ podle anglického dot NET = tečka NET, NET pochází z network, síť) je zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu, která je dostupná nejen pro Web, Windows i Pocket PC. Common Language Infrastructure je standardizovaná specifikace jádra .NET. [http://cs.wikipedia.org/wiki/.NET]
Active Server Pages	ASP	skriptovací platforma společnosti Microsoft, primárně určená pro dynamické zpracování webových stránek na straně serveru. [http://cs.wikipedia.org/wiki/Active_Server_Pages]
	ASP.NET	ASP.NET je součást .NET Frameworku pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages). [http://cs.wikipedia.org/wiki/ASP.NET]
Java server pages	JSP	JavaServer Pages (JSP) je technologie která pomáhá softwarovým vývojářům vytvářet dynamicky generované internetové stránky založené na HTML, XML nebo dalších dokumentových typech. Vypuštěna byla v roce 1999 společností Sun Microsystems, JSP je podobné PHP, ale používá programovací jazyk Java. [http://en.wikipedia.org/wiki/JavaServer_Pages]
JavaScript Object Notation	JSON	JavaScript Object Notation (JavaScriptový objektový zápis, JSON) je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat, která mohou být organizována v polích nebo agregována objektech. Vstupem je libovolná datová struktura (číslo, řetězec, boolean, objekt nebo z nich složené pole), výstupem je vždy řetězec. Složitost hierarchie vstupní proměnné není teoreticky nijak neomezena. [http://cs.wikipedia.org/wiki/JSON]

9. Citovaná literatura

ASP.NET, Microsoft. 2011. ASP.NET MVC 3. *Microsoft ASP.NET*. [Online] Duben 2011. [Citace: 19. Leden 2012.] http://www.asp.net/mvc/mvc3.

—. 2010. Creating a MVC 3 Application with Razor and Unobtrusive JavaScript. *Microsoft ASP.NET*. [Online] 1. Listopad 2010. [Citace: 19. Leden 2012.] http://www.asp.net/mvc/tutorials/overview/creating-a-mvc-3-application-with-razor-and-unobtrusive-javascript.

Běhálek, Marek. 2007. Architektura .NET Framework. *Katedra informatiky.* [Online] Vysoká škola báňská Fakulta elektrotechniky, 2007. [Citace: 25. 6 2012.] http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch01s01.html.

Bernard, Borek. 2009. Úvod do architektury MVC. *Zdroják.cz.* [Online] 7. Květen 2009. [Citace: 19. Leden 2012.] http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc/.

Dvořák, Miloš. 2003. Návrhové vzory - úvod. *Objektová analýza, návrh a programování.* [Online] VŠE, 2003. http://objekty.vse.cz/Objekty/Vzory-uvod#CoPred.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlisside. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* místo neznámé : Addison Wesley, 1995. ISBN 0201633612.

Frank Buschmann, Hans Rohnert, Michael Stal, Peter Sommerlad, Regine Meunier. 1996. *Pattern-Oriented Software Architecture.* místo neznámé : Wiley, John & Sons, Incorporated, 1996. ISBN-13: 9780471958697.

Freeman, Adam a Sanderson, Steve. 2011. *Pro ASP.NET MVC 3 Framework.* New York : Apress, 2011.

Hradecký, Radim. 2009. Srovnání softwarových frameworků. *Archív diplomových prací ČVUT.* [Online] 18. 5 2009. [Citace: 12. 6 2012.] https://dip.felk.cvut.cz/browse/pdfcache/hrader1_2009dipl.pdf.

2010. Introducing “Razor” – a new view engine for ASP.NET . *asp.net.* [Online] 2. Červenec 2010. [Citace: 19. Leden 2012.] http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx.

JakPsátWeb. 2012. Hosting s PHP. *Jak psát web.* [Online] 13. 5 2012. [Citace: 26. 6 2012.] http://www.jakpsatweb.cz/katalog/hosting-php.html.

—. 2009. PHP - Jak začít. *Jak Psát Web.* [Online] 2009. [Citace: 26. 6 2012.] http://www.jakpsatweb.cz/php/jak-zacit.html.

Kuznetsov, Stanislav. 2009. Bakalářská práce Použití technologie Model View Controller (MVC). *Archív diplomových prací ČVUT.* [Online] 6 2009. [Citace: 25. 6 2012.] https://dip.felk.cvut.cz/browse/pdfcache/kuznes1_2009bach.pdf.

MacDonald, Matthew a Szpuszta, Mario. 2008. *ASP.NET 3.5 a C# 2008.* Brno : Zoner Press, 2008.

Microsoft. 2011. Getting Started with ASP.NET MVC 3. *ASP.NET.* [Online] Microsoft, 2011. http://www.asp.net/mvc.

—. 2010. What’s New in ASP.NET MVC 2. *Microsoft ASP.NET.* [Online] 2010. [Citace: 4. Červen 2012.] http://www.asp.net/whitepapers/what-is-new-in-aspnet-mvc.

Riehle, Dirk. 2000. *Framework Design: A Role Modeling Approach.* [Ph.D. Thesis] Zürich, Switzerland : ETH Zürich, 2000. No. 13509.

Tutorial, PHP & MySQL. PHP & MySQL Tutorial. *PHP & MySQL Tutorial.* [Online] [Citace: 25. 6 2012.] http://www.bogotobogo.com/php/images/php1/php_interpreter.jpg.

Virgin_media. 2009. HTTP status codes. *Virgin media.* [Online] 21. Duben 2009. [Citace: 24. Leden 2012.] http://help.virginmedia.com/system/selfservice.controller?CONFIGURATION=1002&PARTITION_ID=1&TIMEZONE_OFFSET=&USERTYPE=&VM_CUSTOMER_TYPE=Cable&CMD=PRINT_ARTICLE&ARTICLE_ID=2537.

Vochozka, Josef. 2003. .NET, stručná informace. *Zpravodaj ÚVT MU.* [Online] 5. 6 2003. [Citace: 16. 6 2012.] http://www.ics.muni.cz/bulletin/articles/281.html. ISSN 1212-0901.

Vondrouš, Jan. 2008. Diplomová práce Renovace MVC frameworků. *Písař.* [Online] 2008. [Citace: 25. 6 2012.] http://pisar.wz.cz/Struts1toStruts2/pdf/dp-print.pdf.

Wikipedia. 2009. Architectural pattern. *Wikipedia.* [Online] 2009. [Citace: 17. 6 2012.] http://en.wikipedia.org/wiki/Architectural_pattern.

—. 2012. Design pattern. *Wikipedia.* [Online] 23. 5 2012. [Citace: 17. 6 2012.] http://en.wikipedia.org/wiki/Design_Pattern.

—. 2012. Framework. *Wikipedia.* [Online] 6. 5 2012. [Citace: 16. 6 2012.] http://cs.wikipedia.org/wiki/Framework.

—. 2011. Hypertext Markup Language. *Wikipedia.* [Online] 28. Prosinec 2011. [Citace: 27. Leden 2012.] http://cs.wikipedia.org/wiki/HyperText_Markup_Language.

—. 2012. Návrhový vzor. *Wikipedia.* [Online] 10. 6 2012. [Citace: 17. 6 2012.] http://cs.wikipedia.org/wiki/N%C3%A1vrhov%C3%BD_vzor.

—. 2012. Software framework. *Wikipedia EN.* [Online] 12. 6 2012. [Citace: 14. 6 2012.] http://en.wikipedia.org/wiki/Software_framework.

—. 2011. Wikipedia ASP.NET MVC Framework. *Wikipedia.* [Online] 13. Leden 2011. [Citace: 4. Červen 2012.] http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework.

—. 2011. Wikipedia Smalltalk. *Wikipedia.* [Online] 7. Červen 2011. [Citace: 19. Leden 2012.] http://cs.wikipedia.org/wiki/Smalltalk.

10. Seznam obrázků