PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Vývoj webových aplikací
v Microsoft ASP.NET MVC a WebForms

2011 Tomáš Podešva

Anotace

Tato bakalářská práce předkládá srovnání vývoje webové aplikace v Microsoft ASP.NET MVC a ASP.NET WebForms spolu s vyhodnocením výhod a nevýhod těchto frameworků. Základními srovnávanými parametry jsou zpracování HTML requestů, tvorba složitějších znovupoužitelných komponent, efektivita vývoje a učící křivka. Srovnání bylo provedeno na vývoji aplikace – jednoduchého redakčního systému komunitního webu, přičemž byla aplikace vytvářena tak, aby využívala v maximální míře typických rysů srovnávaných frameworků.

Děkuji vedoucímu práce Ing. Jiřímu Hronkovi a Bc. Radku Matějovi za praktické připomínky. Taktéž rodině za trpělivost.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

1.1. Téma bakalářské práce

Přibližně deset let se zabývám tvorbou webových prezentací a aplikací, a proto jsem pro bakalářskou práci, jíž uzavírám své studium aplikované informatiky, hledal námět, který by se webu úzce dotýkal. Tématem, které mě oslovilo a které jsem se rozhodl zpracovat, se pak stalo porovnání frameworků ASP.NET Web-Forms a ASP.NET MVC.

Toto téma navrhla studentům Katedry informatiky v Olomouci firma GMC Software Technology, jež se zabývá implementací programů pro efektivní personalizovanou komunikaci a v době, kdy téma předkládala, hledala technologii, kterou zvolí jako prostředek pro další projekty. Se zástupcem firmy jsem spolupracoval po celý čas vytváření této práce, cenné pro mě byly především konzultace k aplikacím, které jsem pro srovnání frameworků vyvíjel.

1.2. Základní užívané pojmy

Pro porozumění této práci je podstatná shoda v interpretaci základních pojmů u autora i čtenáře. Na tomto místě proto uvádím definice některých názvů a slovních spojení, které jsou v textu použity.

- framework – knihovna funkcí
- vývoj webové aplikace – proces vytváření webové aplikace a to od fáze nastudování problematiky, přes sběr požadavků, analýzu a návrh, po implementaci, testování a nasazení
- programovací jazyk – dohodnutá konvence syntaxe a sémantiky, umožňující implementaci programu
- technologie – souhrn použitých frameworků a programovacího jazyka
- html tagy – statické vyhrazené značky strukturující text internetové stránky, které určují vzhled a chování internetové stránky
- asp tagy – od Microsoftu, dynamické vyhrazené značky, strukturující text internetové stránky, určující vzhled a chování internetové stránky
- webová aplikace – jakákoliv aplikace umístěná na webovém serveru, přístupná přes internetový protokol http
- životní cyklus stránky – pořadí akcí vykonávaných při zpracování požadavku klienta na serveru

- server IIS – webový server od Micosoftu
- databázový server – systém řízení báze dat, ukládající data do relačních tabulek a vykonávající nad nimi operace (uložené procedury)
- databázové tabulky – relační tabulky
- uložené procedury v databázovém serveru – naprogramované procedury uložené v databázi
- CLR – (common language runtime) – běhové prostředí jazyků .NET, v němž se vykonává kód, který může být napsán v jakémkoli jazyku, pro který funguje kompilátor v souladu s požadavky CLR
- C# – jazyk ze skupiny .NET, který může být spuštěn v CLR
- Visual Basic – jazyk ze skupiny .NET., který může být spuštěn v CLR
- DLL – (dynamic link library) – knihovna funkcí, která je spustitelná na operačním systému firmy Microsoft
- beta verze – zveřejněná testovací verze
- ASP.NET WebForms – technologie k programování internetových stránek z dílny Microsoft, hlavním rysem je stavové prostředí nad bezstavovým protokolem http
- ASP.NET MVC – technologie k programování internetových stránek z dílny Microsoft, hlavním rysem je deklarovaná udržovatelnost, testovatelnost a oddělení prezenční, řídící a datové logiky
- http protokol – daná sada pravidel, kterou dodržují počítače komunikující přes internet
- WebSite – webová stránka, konkrétně je míněna nezkompilovaná ASP.NET aplikace
- WebApplication – webová aplikace, zde je míněna zkompilovaná ASP.NET aplikace
- viewstate – skryté pole internetové stránky, které uchovává stav prvků na stránce, jedná se o koncept firmy Microsoft a překonává bezstavovost stránek
- ASPX stránka – webová stránka v technologii ASP.NET od verze 2.0
- postback – vlastnost ASPX stránky, opakované načtení stránky

- request – požadavek uživatele webových stránek (klienta), který se odešle na server (například po stisku tlačítka)
- TempData – kolekce ASP.NET MVC technologie, kam programátor může uložit data na dobu jednoho požadavku
- Session – kolekce http protokolu, kam může programátor uložit data, dokud nejsou vynulována nebo přepsána

1.3. Poznámka k webové aplikaci

Frameworky ASP.NET MVC a ASP.NET WebForms jsou zde srovnávány prostřednictvím webové aplikace s jasně definovanou funkcionalitou a vzhledem, již jsem vytvořil zvlášť v každé z technologií. O samotném vývoji aplikace pojednávám níže v kapitole 3., zde bych však rád uvedl několik slov k jejímu praktickému rozměru.

Webová aplikace vznikla ve spolupráci se svépomocným sdružením lidí s duševním onemocněním Meziříčské makovice. Sdružení hledalo způsob, jak posílit prostřednictvím webu komunikaci – a to jak uvnitř komunity lidí s duševní nemocí, tak také směrem ven k široké veřejnosti.

Zakázka tedy, zjednodušeně řečeno, byla na vytvoření webové aplikace se snadnou správou obsahu, jejímž prostřednictvím bude možné aktivní sdílení a předávání informací, názorů a zkušeností týkajících se duševních nemocí a života s nimi.

2. Microsoft ASP.NET framework

ASP.NET framework je produktem společnosti Microsoft a „... je součást .NET Frameworku pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages)“ [18].

ASP.NET je založen na CLR (Common Language Runtime), který sdílí veškeré aplikace postavené na .NET frameworku. Programy tak lze vyvíjet v jakémkoli jazyce podporujícím CLR, např. VisualBasic.NET, C#, Jscript, Managed C++, stejně tak např. mutace Perl, Python apod.

Aplikace založené na ASP.NET se neinterpretují, ale kompilují do jednoho či několika málo DLL (Dynamic-link library) souborů. Kód pro webovou stránku je možné napsat v jakémkoli textovém editoru, zkopírovat jej do virtuálního adresáře na webovém serveru a aplikace se pak dynamicky zkompiluje ve chvíli, kdy k ní přistoupí klient. Kopie aplikace se uloží do cache pro potřeby budoucích požadavků. Pokud se některý ze souborů změní, aplikace se automaticky překompiluje, jakmile ji bude vyžadovat klient. To je zásadní rozdíl oproti skriptovacím jazykům, u kterých jsou stránky při každém přístupu klienta znovu a znovu parsovány. Aplikace v ASP.NET je proto znatelně rychlejší.

Kompilace probíhá u aplikací v ASP.NET ve dvou krocích. V první fázi je kód v použitém programovacím jazyce, např. VisualBasic.NET nebo C#, zkompilován do přechodného jazyka MSIL, tj. Microsoft Intermediate Language, do souboru zvaného assembly. Kompilace do MSIL může proběhnout automaticky při prvním vyžádání stránky nebo jako tzv. předběžná kompilace (precompiling). Druhá fáze kompilace proběhne těsně před skutečným vykonáním stránky, kdy se kód MSIL zkomprimuje do nativního nízkoúrovňového strojového kódu. Jedná se o tzv. kompilaci JIT, tj. just-in-time.

Pro programátory ASP.NET znamená usnadnění přechodu od klasických aplikací pro Windows do prostředí webu, de facto odstraňuje rozdíl mezi vývojem aplikací a webovým vývojem, protože nástroje a technologie, které dříve používali výlučně vývojáři desktopových aplikací, rozšiřuje do oblasti vývoje webu. ASP.NET stránky jsou sestaveny z objektů, ovládacích prvků, kterým je možné přiřazovat konkrétní vlastnosti, zachytávat na nich události apod. Tyto prvky lze považovat za jakýsi ekvivalent ovládacích prvků ve Windows a při vývoji webových stránek je pak možné používat ovládací prvky jako nápis (`Label`), tlačítko (`Button`) atd. Obdobně, jako se ovládací prvky pro Windows kreslí samy do formulářů na obrazovku, generují webové ovládací prvky HTML kód, který pak tvoří část výsledné stránky poslané klientovi.

ASP.NET Framework je v současné době používán ve dvou typech a to ASP.NET WebForms a ASP.NET MVC.

2.1. ASP.NET WebForms

ASP.NET WebForms je starší ze srovnávaných technologií, poprvé byl tento framework vydán v roce 2002. Cílem vývoje ASP.NET WebForms bylo odstranit problémy, které vývojářům působilo bezstavové webové prostředí; zachování stavu, tj. souvislostí mezi jednotlivými požadavky, je totiž nezbytné pro událostmi řízené programování. Překonat problémy s bezstavovým prostředím se podařilo zkombinováním HTML a JavaScriptu, s pomocí technik `ViewState`, kdy jsou informace uchovávány mezi postbacky ve skrytých formulářových polích, a `Session`, kdy jsou informace ukládány na serveru a předávány dál jako jednoznačný identifikátor (cookie nebo součást URL).

ASP.NET WebForms odděluje prezentační a aplikační vrstvu a ASP.NET stránka zde má dvě hlavní části:

- soubor `*.aspx`, s ASP a HTML kódem,
- na serveru uložený soubor na pozadí (CodeBehind), se zdrojovým kódem, který obsluhuje události vyvolané ve stránce. Kód je zapisován v některém z programovacích jazyků ze skupiny .NET, nejčastěji ve VisualBasic či C#.

Pro všechny asp tagy v souboru `*.aspx` je charakteristický atribut `runat` s přiřazenou hodnotou `server`, čímž je zajištěno zpracování komponenty na serveru. Nositelem atributu `runat="server"` musí být rovněž jeden html tag form. Právě díky možnosti vázat danou stránku s asp tagy na soubor na pozadí má vývojář příležitost obcházet bezstavovost http protokolu (např. pomocí `viewstate`), ukládat stav a později s ním podle potřeby pracovat.

Pro ASP.NET WebForms je (resp. byla) také typická souvislost mezi údaji v adresovém řádku internetového prohlížeče a serverem, přičemž adresa zapsaná do adresového řádku přesně odpovídá umístění stránky na webovém serveru. Uživatel tedy volá vždy jednotlivé stránky, které dále zpracovává příslušný kód na pozadí. Verze ASP.NET WebForms 4 již ovšem nabízí programátorovi i možnost volat metodu v kódu pomocí routovacích (směrovacích) pravidel, pak údaje v adresovém řádku odpovídat skutečnému umístění stránky na serveru nemusí.

Koncepci ASP.NET WebForms lze využít jako tzv. Website (jednoduché stránky) či WebApplication (webové aplikace), WebApplication vytváří i `.dll` soubor, jímž je podmíněna testovatelnost. Pro ASP.NET WebForms webové aplikace je charakteristický životní cyklus v závislosti na stránce; životní cyklus stránky je podrobněji rozebrán v 5. kapitole této práce, konkrétně v podkapitole 5.2.1.

K vývoji webové aplikace pro srovnání frameworků ASP.NET WebForms a ASP.NET MVC jsem použil verzi 2.0 ASP.NET stránek.

2.2. ASP.NET MVC

ASP.NET MVC je oproti ASP.NET WebForms novější framework, firma Microsoft se jeho vývojem zabývá od roku 2007. MVC ovšem nemá WebForms nahradit, je alternativní možností využití ASP.NET. ASP.NET MVC využívá stejně jako ASP.NET WebForms běžných vlastností ASP.NET, např. uživatelské role, sessions a další. Na rozdíl od WebForms ovšem používá softwarovou architekturu Model – View – Controller (zkráceně MVC).

MVC jako první popsal Trygve Reenskaug již v roce 1979, poprvé pak byla architektura použita v jazyce Smalltalk, vyvíjeném v Xerox research labs. V současné době se koncept MVC užívá především jako architektura webových aplikací, implementace MVC u složitějších aplikací zajišťuje přizpůsobivost a spolehlivost i při častých změnách a rychlém vývoji [12]. Pro označení MVC se používá také český ekvivalent MPO, který znamená Model – Pohled – Ovládání. Pecinovský uvádí: „Stručná charakteristika vzoru: MPO (Model/ Pohled/ Ovládání) sestává ze tří druhů objektů. Model představuje objekt aplikace, Pohled prezentaci na obrazovce a Ovládání definuje způsob, jak uživatelské rozhraní reaguje na vstup od uživatele. MPO odděluje tyto objekty a tím zvyšuje flexibilitu a znovupoužitelnost celého řešení“ [4].

ASP.NET MVC má tedy tři komponenty: Model, který ošetřuje veškerá data aplikace a zajišťuje všechny operace, které s daty souvisejí, View (Pohled), který zobrazuje uživatelské rozhraní, tj. webovou stránku, a Controller (Ovládání, nebo také Řadič), který zprostředkovává komunikaci mezi Modelem a Pohledem. Řadič má většinou na starosti také validace dat předaných z Pohledu do Modelu. V aplikaci pak např. databázový dotaz pak probíhá tak, že tento dotaz převezme od uživatele Controller, předá jej Modelu a následně Model předá dotaz databázi. Životní cyklus ASP.NET MVC 1 stránky je podrobněji popsán v 5. kapitole této práce, resp. v podkapitole 5.2.1.

S implementací architektury MVC úzce souvisí systém routování (směrování). Protože Views nejsou volány přímo, lze určit prostřednictvím routovacích tabulek, který Controller, akce a následně View bude zavolán. Obsah adresního řádku prohlížeče ve View tak nemusí přesně odpovídat umístění souboru v adresářové struktuře na serveru. To umožňuje použití „SEO Friendly“ URL, tj. lze tak text v adresním řádku prohlížeče zkrátit a eliminovat v řetězci ty znaky, které nejsou na české klávesnici, jsou pro uživatele nesrozumitelné, či na něj působí rušivě atd.

Proti ASP.NET WebForms má ASP.NET MVC ještě další zásadní odlišnost. Protože jsou logika a prezentace webové aplikace díky architektuře MVC odděleny a jednotlivé komponenty jsou na sobě de facto nezávislé, umožňuje ASP.NET MVC bez problémů paralelní vývoj aplikace, kdy se jeden programátor může zabývat Modelem, jiný Views a další Controllerem.

Při vývoji aplikace užité ke srovnání frameworků ASP.NET WebForms a ASP.NET MVC jsem použil ASP.NET MVC 1. V současné době je k dispozici i novější verze ASP.NET MVC 2, která se od 1. verze mírně liší (například se

snadněji ošetřuje chyba, kdy uživatel může podvrhnout nebezpečný skript místo textu), a na začátku roku 2011 byla vydána verze ASP.NET MVC 3.

3. Vývoj webové aplikace v Microsoft ASP.NET frameworku

Předmětem této práce není vývoj webové aplikace jako takový, ale porovnání vývoje aplikace ve frameworcích ASP.NET MVC a ASP.NET WebForms. Proto je kapitola o vývoji aplikace poměrně stručná a cíleně uvádím téměř výhradně ty informace, které přímo souvisejí s procesem srovnávání technologií. Z tohoto důvodu zde nejsou uvedeny např. podrobnější informace o použitelnosti aplikace pro uživatele, nezabývám se blíže popisem získávání informací pro vývoj aplikace, analýzou systému a specifikací požadavků na hardware a software, stejně tak přímo v textu práce neuvádím příklady a odkazy na zdrojový kód aplikace. Nicméně zdrojový kód obou verzí aplikace a skript generující databázovou složku aplikací jsou k dispozici na přiloženém CD.

Grafické zpracování aplikace odpovídá přáním odpovědného zástupce sdružení Meziříčské makovice, které bude vyvinutou webovou aplikaci užívat v praxi.

3.1. Charakteristika webové aplikace

Vytvořená webová aplikace je redakční systém s administrační částí s jednoduchým a přehledným ovládáním, zohledněny jsou zvyklosti návštěvníků stránek.

Uživateli aplikace jsou dvě skupiny osob, jednak členové sdružení Meziříčské makovice, jednak další osoby, které web navštíví. Z hlediska užívání aplikace jsou uživatelé rozděleni dvojím způsobem

- zaregistrovaný uživatel, nebo nezaregistrovaný uživatel,
- zaregistrovaní uživatelé mají dále buď roli „admin“, nebo „redaktor“.

Aby bylo možné provést plnohodnotné porovnání vývoje aplikace v ASP.NET MVC a ASP.NET WebForms

- implementoval jsem systém členství (pro autentizaci),
- implementoval jsem systém rolí (pro autorizaci),
- implementoval jsem jednoduché vyhledávání,
- aplikaci obsahuje kategorie, kam lze podle autorizačních pravidel přidávat obsah,
- k článku lze přidávat komentáře,
- aplikace obsahuje diskusní fórum,
- aplikace obsahuje administrátorský panel, kde může administrátor (uživatel v roli „Admin“) provádět nastavení aplikace a spravovat obsah.

Dále jsem vytvořil následující moduly

- modul pro práci s daty (`DataAndLogic`),
- modul, který má na starosti menu (`CustomMenu`),
- modul administrátorského editoru (`adminEditor`) – ten je odvozený od ajaxcontrollToolkit.htmlEditor – respektive od jeho překladu do češtiny, který jsem provedl,
- modul redaktorského editoru (`redaktorEditor`) – taktéž odvozený od ajaxcontrollToolkit.htmlEditor – resp. jeho překladu do češtiny.

3.2. Použitý hardware a software

Aplikaci jsem vyvíjel v domácích podmínkách s použitím notebooku FSC Amilo PRO V3505, Genuine Intel CPU T2250 @1,73 GHz, 2GB RAM a operačního systému MS Windows XP, software IIS 5, IIS 7 (při nasazení), VisualStudio 2008 Express, MSSQL 2008 Express, MSSQL 2005 (nasazení), NUnit 2.5.7, NunitAsp, framework ASP.NET MVC 1, .NET 3.5, C#.

Server ISS 5.1 jsem nainstaloval kvůli testovatelnosti frameworku ASP.NET WebForms, na server MSSQL 2008 Express jsem nainstaloval databázi s tabulkami a uloženými procedurami aplikace.

3.3. Posloupnost vývoje aplikace v ASP.NET WebForms a ASP.NET MVC

Úkolem této podkapitoly je seznámit čtenáře se základními fakty o tom, jak jsem postupoval při studování problematiky a vývoji aplikace. Na těchto skutečnostech je postaveno hodnocení učící křivky a na základě zkušeností nabytých při vývoji aplikace, jak jej popisuji, jsem pak také volil kvantitativní ukazatele pro další kritéria, prostřednictvím kterých jsem srovnával vývoj aplikace ve frameworcích ASP.NET WebForms a ASP.NET MVC.

3.3.1. Využívání literatury a dalších zdrojů

Když jsem začínal s vývojem webové aplikace pro tuto práci, bylo pro mne vyhledávání informací k ASP.NET frameworkům poměrně obtížné. Literárních zdrojů, o které bych se mohl opřít, bylo velmi málo, především dokumentace k ASP.NET MVC byla prakticky nedostupná. Podstatnou roli zde ovšem sehrál fakt, že v době, kdy jsem svou práci zahajoval, tedy v roce 2009, nebyla Microsoftem uvolněna ještě ani první oficiální verze ASP.NET MVC.

Dostupnější mi byly z počátku informace o ASP.NET WebForms a s tímto frameworkem jsem proto začal pracovat jako s prvním. K úvodu do problematiky

jsem použil publikací MacDonalda a Szpuszty [2], tato kniha mi ovšem velmi dobře posloužila i později k pokročilejší práci ve WebForms. Následně jsem čerpal z knihy Bellinasa [1] a z webových stránek Microsoft ASP.NET [9].

Při vývoj aplikace v MVC jsem narazil na naprostý nedostatek informačních zdrojů v českém jazyce. Pro prvotní seznámení s frameworkem jsem tedy využil webové stránky Microsoft ASP.NET [10] a wikipedii [11], později mě inspirovaly i další internetové zdroje [13], [14], [15], [16]. Ke studiu problematiky jsem použil také publikaci Sandersona [3] a její druhou edici zabývající se ASP.NET MVC 2. Tato publikace dává velmi srozumitelně a obsáhle návod, jak pracovat s frameworkem, zmiňuje možnost, jak mohou ASP.NET WebForms a ASP.NET MVC pracovat společně v jedné aplikaci, postrádal jsem však další doporučení pro práci se `ScriptManagerem`, potažmo s prvky `ajaxcontrolltoolkitu`. Výhodou pří práci s knihou pro mne bylo, že jsem ji zakoupil jako dokument ve formátu pdf, což mi velmi usnadňovalo práci s textem, především jsem využil rychlého vyhledávání.

Vedle zdrojů k jednotlivým frameworkům byla pro mne velmi cenná Pricova publikace [6], kterou jsem používal jako manuál k práci s databází MSSQL. Publikaci jsem využil jak pro práci s WebForms, tak k práci s MVC, jelikož jsou oba framworky na MSSQL založeny. Jako základ pro práci s JavaScriptem jsem využil Thauovu publikaci [7] a pro práci s CSS styly jsem čerpal z McFarlandova manuálu [8].

3.3.2. Praktická část vývoje

Vývoj aplikace jsem zahájil návrhem jejích obecných rysů, tj. návrhem vzhledu a funkcionality. Tento návrh jsem konzultoval se zástupcem sdružení Meziříčské makovice, zapracoval jsem zjištěné připomínky a výstup jsem použil jako základ pro další práci. První zvolená technologie, ve které jsem vyvíjel aplikaci, byla ASP.NET WebForms, jelikož jsem k ní měl více studijních materiálů. Později jsem souběžně s prací ve WebForms vyvíjel aplikaci v MVC.

Začal jsem návrhem společné datové vrstvy, jak uvádím níže, a následně jsem se seznamoval s ovládacími prvky technologie ASP.NET WebForms, které slouží ke svázání dat a webové stránky. První verzi webové aplikace vyvíjené v ASP.NET WebForms jsem předložil zástupci sdružení Meziříčské makovice k připomínkování. Zprvu byla tato verze přijata bez výhrad, později jsem ale obdržel, a následně také zapracoval, zpřesňující připomínky ke vzhledu a funkcionalitě aplikace. Novým požadavkem byla např. implementace mapy webu, vyhledávání a autorizační principy, se kterými nebylo v základním návrhu počítáno. Na základě připomínek, a především požadavku na respektování soukromí uživatelů a omezení počtu povinných údajů při registraci uživatele, jsem odstranil funkci „profil uživatele“, kterou jsem do aplikace původně zařadil jako standardně užívaný prvek redakčního systému. Podle připomínek jsem rovněž rozdělil modul editoru pro vkládaný text na jednodušší a širší verzi. Jednodušší verze je pro registrované uži-

vatele v roli „Redaktor“, kde jsem odebral tlačítka pro formátování textu. Tím jsem splnil požadavek na zachování maximální možné stálosti vzhledu aplikace, kterou by pravděpodobně různý přístup k formátování u jednotlivých redaktorů narušoval. Složitější verze editoru je pro registrované uživatele v roli „Admin“, kde je umožněno formátování textu v plném rozsahu. V roli „Admin“ vystupuje předseda sdružení Meziříčské makovice a jím pověření členové sdružení, kteří přístup k formátování vzájemně konzultují a jednají na základě konsensu o vzhledu webu.

V práci ve WebForms pro mne byla dosti obtěžující dvojznačnost, se kterou lze v této technologii nahlížet na řešení aplikace a která nechává stále otevřenu otázku, zda přistupovat k řešení deklarativně atributem ovládacího prvku, či imperativně z kódu. Tento problém jsem řešil zvláště u validací a samotných validátorů, nicméně později v MVC se projevila tato potíž v ještě větší míře.

Při vývoji aplikace v MVC jsem do ní začlenil společné prvky pro obě technologie, jako jsou `DataAndLogic`, `CustomMenu`, `adminEditor`, `redaktorEditor`. Se zapojením panelu `adminEditor` a `redaktorEditor` jsem měl ale z počátku problém. Panely jsou postaveny na prvku htmleditor, který je dán k dispozici společností Microsoft jako jeden z prvků ve volně dostupném `AjaxcontrollToolkitu`. `Ajaxcontrolltoolkit` pro svůj běh ale vyžaduje `ScriptManager`, což je ovládací prvek svázaný s technologií ASP.NET WebForms. Nicméně nakonec se mi vzhled a funkcionalita podařila mezi oběma frameworky sladit.

Po dokončení aplikace ve WebForms i MVC jsem s odstupem usoudil, že administrativu stránek by usnadnilo vytvoření vlastní administrátorské sekce. Dodatečně jsem tedy obě verze aplikace takto upravil.

Pro odzkoušení a předání aplikace jsem si zaregistroval domény www.bc-wf.cz, kam jsem nainstaloval aplikaci v ASP.NET WebForms, a www.bc-mvc.cz, kam jsem umístil aplikaci opírající se o ASP.NET MVC. Rovněž jsem instaloval databázi na příslušné databázové servery svázané s příslušnou hostující aplikací. Postup publikace byl totožný pro každou verzi aplikace. Instalace proběhla prostřednictvím funkce VisualStudia pro publikování webu a pro publikování databáze. Obě verze aplikace jsou hostovány společností Asp One, která má na svých stránkách velmi dobrou podporu ve formě školicích animací, snímků obrazovek, diskusí a odkazu na často kladené otázky k publikování obsahu.

Po vyhotovení obou verzí aplikace, jsem dokončil testy a srovnal jsem snadnost (jednoduchost) učení, vývoje, údržby, modularity a nasazení aplikace v každém z těchto dvou frameworků, stejně tak jsem pak srovnal silné a slabé stránky frameworků ASP.NET WebForms a ASP.NET MVC.

3.3.3. Databáze

Pro webovou aplikaci bylo nutné vyvinout společnou datovou vrstvu pro obě verze. Vytvořil jsem tedy vrstvu využívající Microsoft SQL 2008 Express server. Jako elementární součást datové vrstvy jsem vytvořil i databázové tabulky

se zkušebními daty, tj. vlastním obsahem redakčního systému. Vedle toho jsem naprogramoval uložené procedury databáze MSSQL. K rozhodnutí použít uložené procedury mě vedla přirozená snaha o modularitu a především bezpečnost, kterou lze zajistit řízením oprávnění na úrovní jednotlivých procedur. Dále jsem eliminoval pokus o vložení skriptu do databázového dotazu, čehož jsem dosáhl parametrizací příkazu. Výsledkem této eliminace je navázání všech předávaných vstupujících dat do databáze na příslušné parametry. Využil jsem zde doporučený postup podle Bellinasa [1] strana 80.

Data získaná z databáze jsem navázal na objekty jazyka C#, které jsou instancemi příslušných tříd z vlastního jmenného prostoru `MM.DL` (zkratka „Meziříčské makovice – datová vrstva“). Uživateli jsou předána jako instance `MM.AL` (zkratka „Meziříčské makovice – aplikační vrstva“). Tyto třídy jsou zatím téměř identické. Nicméně tímto rozdělením sleduji lepší schopnost přidat v budoucnu aplikační logiku, například validaci vstupu, do `MM.AL` tříd. Předání z dat `MM.DL` do `MM.AL` probíhá přes třídu `MM.AL.Client.ExecutiveLogic`. To vše je zapouzdřeno v modulu s názvem `DataAndLogic.dll`, který je společný pro řešení v obou srovnávaných frameworcích. Každá verze aplikace využívá v produkčním prostředí vlastní databázi.

3.3.4. Testování

Během vývoje jsem obě verze aplikace průběžně testoval, WebForms pomocí knihovny `NUnitAsp.dll` a MVC pomocí `NUnit.dll`. V externím programu NUnit jsem průběžně spouštěl velké množství automatizovaných testů. Změřil jsem se především na automatizované white box unit testování.

Podařilo se mi otestovat stav a chování aplikace ve WebForms, což rozhodně nebylo triviální záležitostí. Testování proběhlo výše zmíněným NUnitAsp doplňkem pro NUnit. Pro testování WebForms bylo nutno nainstalovat IIS 5.1 server, samotné testování pak probíhalo na simulaci prohlížeče, čímž bylo možno otestovat také chování.

Testování MVC bylo oproti testování WebForms snadnější a přímočařejší, nebylo třeba mít nasazenou aplikaci na webový server. Ovšem pro plnohodnotné automatizované testování chování by byl server nutný, a to ve verzi vyšší než IIS 5.1. Ten jsem bohužel neměl k dispozici a toto testování jsem tedy nemohl provést.

3.3.5. Slepé uličky

Z výše uvedeného je patrné, že vývoj aplikace neprobíhal bez potíží ani v jednom ze srovnávaných frameworků. Považuji za užitečné zde zmínit několik situací, které lze nazvat „slepými uličkami“.

V ASP.NET MVC:

1. chyba: používání editoru bez `ScriptManager`
    - chybou byla snaha oddělit co nejvíc řešení MVC od WebForms (snaha o „ryzí aplikace“)
    - řešení: použití `ScriptManageru`, čili, jak uvádějí tvůrci ASP.NET MVC, ASP.NET MVC neaspiruje na snahu kompletně nahradit ASP.NET WebForms, jedná se spíš o jiný styl práce a doplnění původního WebForms než o „WebForms 4“ (Sanderson v [3] str. 11 a prakticky celou poslední 16. kapitolu věnuje problematice, jak mohou oba frameworky existovat souběžně v jednom projektu.)
2. chyba: použití komponent vyžadující `ViewState`
    - řešení: použití `TempData`
3. chyba: problematickým se jeví použití `Session`
    - je nutno nulovat obsah, je velká paměťová náročnost
    - řešení: použití `TempData`, konkr. kolekce, která má životnost rovnou délce jednoho požadavku (requestu)

V ASP.NET WebForms:

1. chyba: použití stránky se dvěma odlišnými políčky
    - stránka musí použít právě jeden formulář, obtížně se pak používá stránka, kde jsou dvě odlišná políčka. Např. umístěním políčka „editor článku“ a pod něj políčka „odeslání komentáře ke článku“ dochází k tomu, že stisknutím pro odeslání (submit, schválení) jednoho z textových polí je provedena validace obou políček.
    - řešení: použít explicitně, deklarativně u asp tagu `CausesVaildation="false"` (standardně je nastaveno na `true` u všech políček)
2. chyba/ resp. obtíž: průběžné psaní testů
    - stránky jsou svázány s kódem na pozadí, nemohou být tedy testovány jednotlivé moduly, pouze stránky jako celek. Navíc je nutno znát klientské id jednotlivých tagů, na kterých chce programátor testovat události, či jejich viditelnost.
    - testování ASP.NET WebForms patří mezi slabší stránky frameworku, i když je možné např. pomocí NunitAsp – problém je v tom, že se musí simulovat běh webového serveru, neboť ve WebForms se netestuje dll modul jako v MVC, ale simuluje se chování uživatele při práci s prohlížečem

- řešení: psát testy s pomocnou metodou zjišťující klientské id. Nejedná se bohužel o řešení v pravém slova smyslu, neboť dojde-li k jakékoliv změně v aspx stránce, například je jiné id či jiné autorizační pravidlo, pak se musí testy kompletně přepsat.

3. další obtíže u ASP.NET WebForms

   - testování chování, zvláště komplexnějšího, je velmi časově náročné
   - doba vykonávání testů je velmi dlouhá

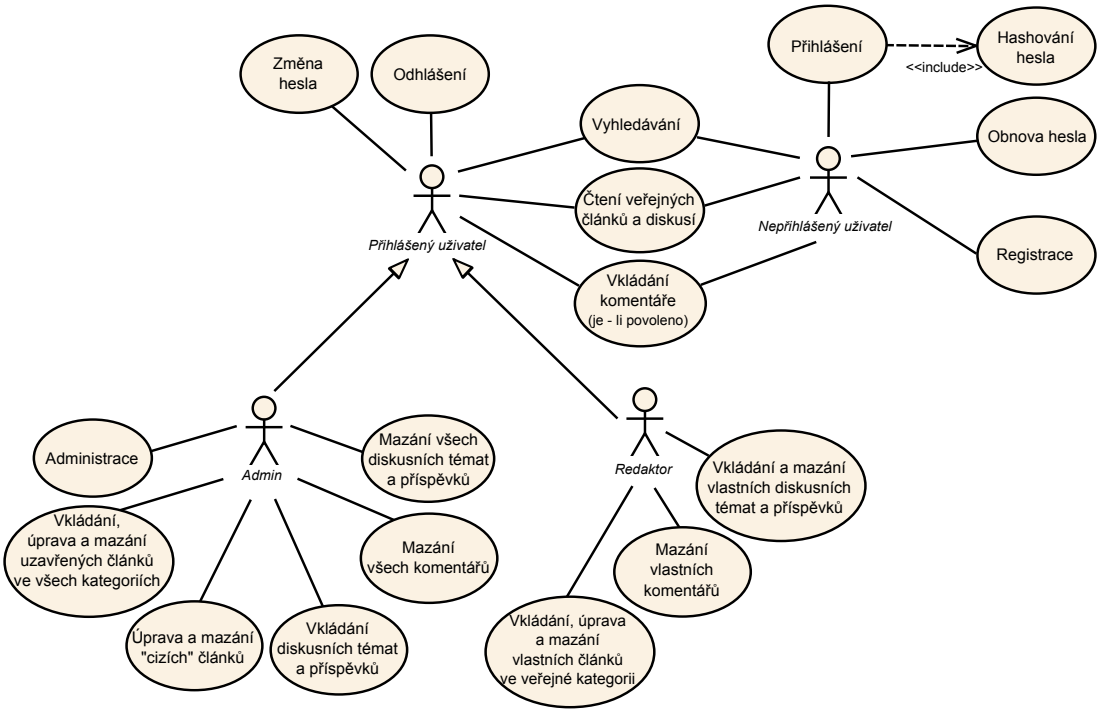
Společný problém při nasazení na server s ASP.NET – chyba při pokusu o předání nebezpečného kódu přes textové políčko webové aplikace

- řešení viz Pokyny pro instalaci v příloze.

4. Funkcionalita webové aplikace

Funkcionalita webové aplikace byla určena jednak na podkladě zakázky sdružení Meziříčské makovice a jednak na základě kritérií, podle kterých měl být srovnáván vývoj aplikace v ASP.NET MVC a ASP.NET WebForms, jak je popsáno v kapitole 3. této práce.

Základní náhled do funkcionality webové aplikace nabízí obrázek, podrobnější popis je pak obsažen pod obrázkem, kde jsou k funkcím přiřazeny odpovídající metody v jednotlivých frameworcích.



Obrázek 1. Funkcionalita aplikace

Přehled funkcionality

1. Registrace uživatele.
   - **ASP.NET WebForms.** Stránka `Register.aspx` s `Register.CreatingUser`. Poté je zavolána `CreatedUser` + `Created` nebo `CreateError`.
     - Validace probíhá v kódu.
     - Nastavení odesílání mailu probíhá ve stránce.
     - V kódu je z důvodu možnosti script injection použitý `Encode`.

- **ASP.NET MVC.** `Controllers/AccountController.Register [Get]` volá pohled `Views/Account/Register.aspx`. Po odeslání se aktivuje `Controllers/AccountController.Register[Post]`.
  - Validace probíhá v `[Post]` metodě.
  - Odesílání probíhá v `[Post]` metodě (volá `SendingMail.cs`).

2. Přihlášení uživatele.

   - **ASP.NET WebForms.** Stránka `LogIn.aspx` s `LogIn.LoggingIn` a `LogIn.loginCtrl_Auth`, v kódu je z důvodu možnosti script injection použitý `Encode`.
   - **ASP.NET MVC.** `Controller/AccountController.LogOn [Get]` volá pohled `Views/Account/LogOn.aspx` Po odeslání se aktivuje `Controller/AccountController.LogOn [Post]`.

3. Změna hesla.

   - **ASP.NET WebForms.** Stránka `ChangePassword.aspx` – validace ve stránce v prvku `<asp:ChangePassword>`.
   - **ASP.NET MVC.** `Controller/AccountController.ChangePassword [Get]` volá pohled `Views/Account/ChangePassword.aspx`. Po odeslání se aktivuje `Controller/AccountController.ChangePassword [Post]` – validace v kódu. Při úspěchu zobrazí pohled `Views/Account/ChangePasswordSuccess.aspx`.

4. Obnova hesla.

   - **ASP.NET WebForms.** Stránka `PRecovery.aspx` spustí `PRecovery.PreRenderRecovery` – zajistí správné zobrazení enkódované hodnoty Stránka při verifikaci použije metodu `PRecovery.VerifyingUser`, která hodnotu enkóduje, dál verifikace pokračuje ve stránce v `<asp:PasswordRecovery>`.
     - Případně se spustí `PRecovery.ErrorUser`, která opět zajistí správné zobrazení enkódované hodnoty.
     - Nastavení odesílání mailu probíhá ve stránce.
   - **ASP.NET MVC.** `Controller/AccountController.RecoveryStep1 [Get]` volá pohled `Views/Account/RecoveryStep1.aspx` po vyplnění je aktivována metoda `Controller/AccountController.RecoveryStep1 [Post]`, tam probíhá získání bezpečnostní uživatelské otázky pro obnovu hesla, zobrazí se v pohledu `Views/Account/RecoveryStep2.aspx`. Při vyplnění odpovědi je volán `Controller/AccountController.RecoveryStep1`

[Post], kde probíhá validace. Při chybě je zavolán pohled zpět, při úspěchu je volán pohled Views/Account/ChangePasswordSuccess.aspx.

5. Zobrazení seznamu článků v kategorii.

   - **ASP.NET WebForms.** Stránka Clanky.aspx ve spolupráci s Clanky.aspx.cs – nastaví viditelnost panelů. O zobrazení seznamu článků se stará ListArticles.ascx – tam <asp:ObjectDataSource> s příslušnou metodou (v instanci MM.AL.Client.ExecutiveLogic GetPublishedByCategory) a <asp:GridView>.
   - **ASP.NET MVC.** Controller/HomeController.Index [Get] nastavaví IndexModel a pošle pohledu Views/Home/Index.aspx. O viditelnost se starají javascriptové funkce v tomto pohledu a inline kód.

6. Vložení nového článku do kategorie.

   - **ASP.NET WebForms.** Stránka Clanky.aspx ve spolupráci s Clanky.aspx.cs – nastaví viditelnost prvků. O vložení článku se stará AddNewArticle.ascx a její metoda v kódu na pozadí AddNewArticle.ascx.cs – Page_Load – nastaví viditelnost panelu. O vložení se stará metoda InsertArticle v AddNewArticle.ascx.cs – provede i validaci.
   - **ASP.NET MVC.** Pohled Views/Home/Index.aspx vykresluje část AddNewArticle a předává ji model. (<% Html.RenderPartial("AddNewArticle", Model); %>) To vykreslí Views/Home/AddNewArticle.ascx s panelem pro vložení článku. Po jejím vyplnění se aktivuje Controller/HomeController.Index [Post], která provede příslušnou validaci. Také mapuje obsah těla na TempData, neboť jako editor těla je použitý AjaxControllToolkit vyžadující ScriptManager, tento editor není z důvodu jiného životního cyklu MVC (než WF) viditelný v Controlleru.

7. Zobrazení celého článku.

   - **ASP.NET WebForms.** Stránka DetailClaku.aspx, nastavení viditelnosti prvků v DetailClanku.aspx.cs v metodě Page_Load. Zobrazení článku má na starosti DetailArticle.ascx a její <asp:DetailsView>, jako zdroj využívá <asp:ObjectDataSource> v DetailClanku.aspx. (Instance MM.AL.Client.ExecutiveLogic, metoda GetArticleByID). DetailArticle.ascx.cs v Page_Load řídí viditelnost prvků.

- **ASP.NET MVC.** Přes `Controller/ArticleController.Index [Get]` je volán pohled `Views/Article/Index`. Ten pro zobrazení detailu vykresluje část detailu `Views/Article/EditArticle.ascx`, kam je předaný model. O viditelnost prvků se starají javascriptové metody v `Views/Article/EditArticle.ascx`.

8. Editace už vytvořeného článku.

   - **ASP.NET WebForms.** Stránka `DetailClaku.aspx`, nastavení viditelnosti prvků v `DetailClanku.aspx.cs` v metodě `Page_Load`. Editaci článku má na starosti `EditArticle.ascx` – `<asp:DetailsView>`. V `EditArticle.aspx.cs` se řídí viditelnost (v `Page_Load`) a v `OnItemCommand` prvku `<asp:DetailsView>` (`e.CommandName = "Cancel"` nebo `"Edit"`). V `OnItemCommand` prvku `<asp:DetailsView>` (`e.CommandName = "Update"`) se zavolá `Update` metoda. Kód na serveru provede validaci a uložení do databáze.
   - **ASP.NET MVC.** Přes `Controller/ArticleController.Index [Get]` je volán pohled `Views/Article/Index`. Ten pro zobrazení editačního panelu vykresluje část detailu `Views/Article/EditArticle.ascx`, kam je předaný model. O viditelnost prvků se starají javascriptové metody v `Views/Article/EditArticle.ascx`. A tamtéž i inline kód závislý na modelu. Po vyplnění editačního panelu tamtéž se aktivuje `Controller/ArticleController.Index [Post]` – ten na základě klíčů `FormCollection` zjistí hodnoty polí v editačním panelu, validuje je a uloží je.

9. Zobazení komentářů k článku.

   - **ASP.NET WebForms.** Stránka `DetailClaku.aspx`, nastavení viditelnosti prvků v `DetailClanku.aspx.cs` v metodě `Page_Load`. Zobrazení komentářů k článku má na starosti `DetailComment.ascx`. Viditelnost prvků na stránce pomáhá řídit `DetailComment.ascx.cs` – `Page_Load`.
   - **ASP.NET MVC.** Přes `Controller/ArticleController.Index [Get]` je volán pohled `Views/Article/Index`. Ten pro zobrazení komentářů vykresluje část detailu `View/Article/ListComments.ascx`. O viditelnost prvků se starají javascriptové metody v `Views/Article/ListComments.ascx`. A tamtéž i inline kód závislý na modelu.

10. Vložení nového komentáře.

- **ASP.NET WebForms.** Stránka `DetailClaku.aspx`, nastavení viditelnosti prvků v `DetailClanku.aspx.cs` v metodě `Page_Load`. Vložení komentáře k článku má na starosti `AddNewComment.ascx`. Viditelnost prvků na stránce pomáhá řídit `AddNewComment.ascx.cs` – `Page_Load`. Po vyplnění probíhá validace na serveru v `AddNewComment.ascx.cs` v metodě `btnSubmitComment_Clicked`.
- **ASP.NET MVC.** Přes `Controller/ArticleController.Index [Get]` je volán pohled `Views/Article/Index`. Ten pro vložení komentáře vykresluje část detailu `Views/Article/AddNewComment.ascx`, kam je předaný model. O viditelnost prvků se starají javascriptové metody v `Views/Article/AddNewComment.ascx`. Po vyplnění se aktivuje `Controller/ArticleController.Index [Post]` – ten na základě klíčů `FormCollection` zjistí hodnoty polí nového komentáře, validuje je a uloží je. Čili metoda `Controller/ArticleController.Index [Post]` je stejná, která má na starosti i editaci článku. Rozlišuje se pomocí klíčů právě `FormCollection`. Což je trochu nešikovné a je to „zbytek myšlení ve WebForms“, kde musí být právě 1 formulář. Narozdíl od MVC, kdy v jedné stránce může být zvláštní formulář a na něj navázaná akce pro víc prvků na jedné stránce.

11. Zobrazení seznamu diskusních fór.

    - **ASP.NET WebForms.** Stránka `Forum/Diskusni-forum.aspx` ve spolupráci s `Forum/Diskusni-forum.aspx.cs` `Page_Load` nastaví viditelnost. O zobrazení senamu diskusí se stará `Forum/ListForums.ascx`.
    - **ASP.NET MVC.** `Controller/ForumController.Index [Get]` předá model pohledu `Views/Forum/Index.aspx`. Tento pohled pak ve foreach smyčce zobrazí všechny diskuse.

12. Vložení nové diskuse.

    - **ASP.NET WebForms.** Stránka `Forum/Diskusni-forum.aspx` ve spolupráci s `Forum/Diskusni-forum.aspx.cs` `Page_Load` nastaví viditelnost. O vložení nové diskuse se stará vkládací panel `Forum/AddNewForum.ascx`. Její `Forum/AddNewDiskuse.ascx.cs` `Page_Load` nastaví viditelnost. (Tu lze dále upravovat klikáním na ikony otevření/zavření). Validace se provádí v `Forum/AddNewDiskuse.ascx.cs` `btnSubmitForum_Clicked`.
    - **ASP.NET MVC.** `Controller/ForumController.Index [Get]` předá řízení pohledu `Views/Forum/Index.aspx`. Tento pohled obsahuje mimo jiné `using (Html.BeginForm("Open", "Forum", FormMethod.Post))` – způsobí vyvolání

`Controller/ForumController.Open [Post]`, která otevře panel vkládání. Podobně `(Html.BeginForm("Close", "Forum", FormMethod.Post))` – způsobí vyvolání `Controller/ForumController.Close [Post]`, která zavře panel vkládání. Samotný panel vložení nové diskuse je relaizován v `using (Html.BeginForm("New", "Forum", FormMethod.Post))` – odeslání po vyplnění způsobí vyvolání kódu v `Controller/ForumController.New [Post]` – ta má na starosti validaci, změnu modelu a předání řízení příslušnému pohledu (v závislosti na výsledku validace).

13. Zobrazit detaily fóra.

    - **ASP.NET WebForms.** Stránka `Forum/Show-forum.aspx`, viditelnost prvků se nastaví v `Forum/Show-forum.aspx.cs Page_Load`. Detaily fóra tamtéž v prvku `Forum/HeaderForum.ascx`.
    - **ASP.NET MVC.** `Controller/ForumController.Detail` předá model pohledu `Views/Forum/Detail.aspx`. Ten jej zobrazí. Viditelnost jednotlivých prvků na stránce je řízená javascriptem a inline kódem.

14. Zobrazit seznam příspěvků.

    - **ASP.NET WebForms.** Stránka `Forum/Show-forum.aspx`, viditelnost prvků se nastaví v `Forum/Show-forum.aspx.cs Page_Load`. Seznam příspěvků tamtéž v prvku `Forum/ListPosts.ascx`, viditelnost jednotlivých tlačítek je řízená `Forum/ListPosts.ascx.cs` v `Page_Load`. Dále jsou v Forum/ListPosts.ascx.cs obsluhy jednotlivých tlačítek.
    - **ASP.NET MVC.** `Controller/ForumController.Detail` předá model pohledu `Views/Forum/Detail.aspx`. Ten jej zobrazí. Ve foreach smyčce projde a zobrazí příspěvky diskuse. Viditelnost jednotlivých prvků na stránce je řízená javascriptem a inline kódem.

15. Nový příspěvek do diskuse.

    - **ASP.NET WebForms.** Stránka `Forum/Show-forum.aspx`, viditelnost prvků se nastaví v `Forum/Show-forum.aspx.cs Page_Load`. Vložení příspěvu tamtéž v prvku `Forum/AddNewPost.ascx`. Viditelnost jejích prvků se nastaví v `Forum/AddNewPost.ascx.cs Page_Load`, schválení po odeslání vyplněných polí se provede (včetně validace) v `Forum/AddNewPost.ascx.cs btnSubmitPost_Clicked`.

- **ASP.NET MVC.** `Controller/ForumController.Detail` předá model pohledu `Views/Forum/Detail.aspx`. Ten jej zobrazí. Viditelnost jednotlivých prvků na stránce je řízená javascriptem a inline kódem. Příslušné vložené `<% using (Html.BeginForm()) {` zobrazí formulář pro vkládání či odpověď. Po vyplnění se odešle a zpracuje v `Controller/ForumController.Detail`. Tam se na základě existence klíčů v kolekci `Request.Param` zjistí, zda-li jde o odeslání formuláře a případně jakého (vložení nebo odpověď).

16. Vymazání dílčího obsahu.

    - **ASP.NET WebForms.** Triviálně: Kliknutí na příslušná tlačítka v seznamech obsahu (seznam článků, komentářů, diskusí, příspěvků).
    - **ASP.NET MVC.** Triviálně: Kliknutí na příslušná tlačítka v seznamech obsahu (seznam článků, komentářů, diskusí, příspěvků).

17. Administrátorské stránky.

    - **ASP.NET WebForms.** Triviálně: Stránky v `Administrator/`, které s příslušným kódem na pozadí plní vždy jeden účel. Podle toho jsou pojmenovány (anglicky).
    - **ASP.NET MVC.** Triviálně: Pohledy ve `Views/Administrator` jsou svázany s příslušnými metodami v `Controller/AdministratorController` – plní vždy tu funkci, podle které jsou pojmenovány (anglicky).

5. Srovnání frameworků Microsoft ASP.NET WebForms a ASP.NET MVC z hlediska vývoje webových aplikací

Srovnání frameworků ASP.NET WebForms ASP.NET MVC v této práci zahrnuje srovnání

- jednotlivých definovaných kritérií,
- výhod a
- nevýhod.

Náčrty porovnávající framewoky ASP.NET WebForms a ASP.NET MVC obsahují některá internetová fóra, příkladem může být diskuse na webu programujte.cz [19]. Srovnání lze případně nalézt také na internetové stránce msdn [20], kde je brána v potaz verze .NET 4, či na webové stránce stackoverflow [21]. Tyto náčrty ovšem postrádají komplexnost a nejsou doloženy konkrétními daty.

Bylo by jistě přínosné porovnat výsledky mého srovnání vývoje aplikace v ASP.NET WebForms a ASP.NET MVC se zjištěními dalších programátorů, k relevantnímu porovnání bohužel není dostupný dostatek podkladů. Mohu jen konstatovat, že vyhodnocení výhod a nevýhod jednotlivých frameworků, k němuž jsem dospěl, se prakticky shoduje se zkušenostmi, o kterých je referováno v diskusních fórech.

5.1. Kritéria srovnání a určení skóre

Kritéria, která slouží ke srovnání frameworků ASP.NET WebForms a ASP.NET MVC, byla především určena zadáním a částečně jsem je doplnil na základě vlastní zkušenosti s tím, co ovlivňuje komfort vývoje webové aplikace. Firma GMC Software Technology žádala srovnání zpracování HTML requestů, tvorby složitějších znovupoužitelných komponent, efektivity vývoje a učící křivky. K těmto kritériím jsem připojil ještě snadnost vývoje, snadnost údržby, testovatelnost, modularitu a nasazení aplikace do produkčního prostředí. Srovnání je cíleno čistě na vývoj aplikace ve frameworcích ASP.NET WebForms a ASP.NET MVC, proto nejsou porovnávány parametry související s během aplikace a výkonností frameworků, podstatný je v této práci pohled vývojáře, nikoli uživatele.

Každé ze stanovených kritérií má přidělenu celkovou hodnotu jednoho bodu, kritéria jsou tedy ve výsledném srovnání rovnocenná. Bod příslušného kritéria je rozdělen podle míry splnění požadavků na kritérium kladených mezi frameworky MVC a WebForms. Většina kritérií je pro přesnější vyhodnocení míry splnění požadavků rozdělena na podkritéria, mezi která je bod takového kritéria rozprostřen. Skóre v tomto kritériu pak značí poměr souhrnného zhodnocení podkritérií. Výsledné skóre ASP.NET WebForms a ASP.NET MVC je následně celkovým

poměrem bodů rozdělených mezi frameworky podle jednotlivých srovnávaných kritérií.

5.2. Zpracování srovnání frameworků ASP.NET WebForms a ASP.NET MVC

Vyhodnocoval jsem celkem devět kritérií, prostřednictvím kterých jsem srovnával vývoj webové aplikace v ASP.NET WebForms a ASP.NET MVC. Těchto devět kritérií jsem rozdělil do osmi bodů, přičemž do prvního bodu jsem sloučil srovnání zpracování HTML requestů a tvorbu složitějších znovupoužitelných komponent. Ke spojení těchto dvou kritérií jsem přistoupil proto, že spolu úzce souvisejí a hodnotil jsem je na shodném podkladě.

5.2.1. Zpracování HTML requestů, tvorba složitějších znovupoužitelných komponent

V rámci tohoto kritéria jsou srovnávány fáze životního cyklu webových aplikací v jednotlivých frameworcích a snadnost zapojení vlastních modulů k aplikaci.

Životní cyklus ASP.NET WebForms stránky.
Životní cyklus ASP.NET WebForms stránky probíhá ve dvou krocích [17].

Krok 1.:
V tomto kroku

- Uživatel odešle požadavek na IIS.
- ASP.NET vytvoří prostředí pro zpracování požadavku.
- Jakmile je prostředí vytvořeno, zpracuje se série událostí (za použití modulů, ovladačů a objektů stránky).

Krok 1. je vykonán v šesti dílčích částech:

1.1.	Uživatel zašle požadavek na IIS. IIS zkontroluje, které ISAPI rozšíření může zpracovat tento požadavek, záleží na příponě požadavku. Jestliže stránka je ASPX stránkou, pak se požadavek zpracuje modulem `aspnet_isapi.dll`.
1.2.	Jestliže se jedná o první požadavek na stránku, pak třída s názvem `ApplicationManager` vytvoří aplikační doménu, na které může stránka běžet. Tím dojde k odizolování od ostatních aplikací hostovaných na stejném IIS.

1.3.	Nově vytvořená aplikační doména vytvoří hostující prostředí (např.: objekt `HttpRuntime`). Jakmile je toto hostující prostředí vytvořeno, pak se vytvoří jádrové ASP.NET objekty, např. `HttpContext`, `HttpRequest`, `HttpResponse`.
1.4.	Jakmile jsou vytvořeny všechny jádrové objekty, je dále vytvořen `HttpApplication` objekt k obsloužení požadavku. V případě, že je v systému soubor `global.asax`, pak se vytvoří i on. (Pozn.: `global.asax` dědí `HttpApplication` třídy).
1.5.	`HttpApplication` objekt je přiřazen k jádru ASP.NET objektu, aby obsloužil stránku.
1.6.	`HttpApplication` pak začíná vykonávat požadavek pomocí http událostí, modulů, ovladačů a událostí stránky. Spouští krok 2.

Tabulka 1. Vytvoření objektů v životním cyklu stránky ASP.NET WebForms.

Krok 2.:
V tomto kroku

- Jakmile je objekt `HttpApplication` vytvořen nastartuje se vykonávaní požadavku skrz tři odlišné sekce a to `HttpModule`, `Page`, `HttpHandler`.
- V jedotlivých sekcích se vyvolávají odlišné události, které může vývojář rozšiřovat a kam může přidávat vlastní logiku.
- `HttpModule` a `HttpHandler` pomáhají vývojářovi vložit vlastní logiku před a za vykonání procesu ASP.NET stránky, přičemž `HttpHandler` je procesor založený na příponě a `HttpModule` je procesor založený na událostech.

Podrobněji lze krok 2. popsat takto:

Sekce	**Událost**	**Popis**
`HttpModule`	`BeginRequest`	Signalizuje nový požadavek; je garantováno, že bude událost vyvolána při každém požadavku.
`HttpModule`	`AuthenticateRequest`	Signalizuje, že běhové prostředí ASP.NET je připraveno k autentizaci uživatele. Vhodné na autentizační kód.
`HttpModule`	`AuthorizeRequest`	Signalizuje, že běhové prostředí ASP.NET je připraveno k autorizaci uživatele. Vhodné na autorizační kód.

`HttpModule`	`ResolveRequestCache`	V této události běhové prostředí ASP.NET určuje, zda stránka má být obsloužena z cache raději než nahráním. Zde může být vložena specifická cachovací aktivita.
`HttpModule`	`AcquireRequestState`	Tato událost signalizuje, že běhové prostředí je připraveno získat session proměnné (proměnné prostředí). Vhodné na jakékoli vykonávaní s proměnnými session.
`HttpModule`	`PreRequestHandlerExecute`	Tato událost je vyvolána dříve, než je obsloužen prvek v `HttpHandleru`. Vhodné na kód, který chceme vykonat dřív, než bude kontrola předána handleru.
`HttpHandler`	`ProcessRequest`	Je vykonána logika HttpHandleru. Do této sekce můžeme napsat kód, který potřebujeme vykonat jako rozšíření stránky.
`Page`	`Init`	Tato událost nastává v ASP.NET stránce a může být užita pro • vytváření dynamických ovládacích prvků, • inicializaci nastavení, • `MasterPages` a jejich nastavení. V této sekci není přístup do `ViewState`, odeslaných hodnot ani inicializovaných ovládacích prvků.

`Page`	`Load`	V této sekci jsou ASP.NET ovládací prvky plně nahrány a můžeme tedy psát logiku UI nebo jakoukoli další nad tímto.
`Page`	`Validate`	Máme-li na stránce validátory, můžeme je zkontrolovat tady.
`Page`	`Render`	Teď je čas odeslat výstup prohlížeči. Jestliže chceme udělat změny na finálním HTML, který odešleme prohlížeči, pak může být zde zadána naše HTML logika.
`Page`	`Unload`	Page objekt je uvolněn z paměti.
`HttpModule`	`PostRequestHandlerExecute`	Prostor pro jakoukoli logiku, kterou chceme vložit poté, co jsou ovladače vykonány.
`HttpModule`	`ReleaseRequestState`	Prostor pro uložení změny stavu nějaké proměnné (např.: Session proměnná)
`HttpModule`	`UpdateRequestCache`	Vhodné pro zaktualizování cache, dřív než skončíme.
`HttpModule`	`EndRequest`	Toto je poslední událost, dřív, než je výsledek odeslán na klientský prohlížeč.

Tabulka 2. Fáze zpracování požadavků po vytvoření objektů

Velmi důležitou sekcí v popisovaném druhém kroku je ASP.NET stránka (`Page`). Tato ASP.NET stránka má dvě části:
1. část: Je zobrazena v prohlížeči (HTML tagy, skrytá pole ve formulářích, `viewState` a data v HTML vstupech). Když je stránka odeslána, tyto HTML tagy jsou vytvořeny v ASP.NET ovládacích prvcích s `viewState` a formulářovými daty svázanými dohromady a předány na server.
2. část: Kód na pozadí (tehdy když dostaneme tyto serverové ovladací prvky), který můžeme vykonávat a napsat do něj vlastní logiku, poté se zpětně renderuje stránka zpátky na prohlížeč. Dřív než tento HTML ovládací prvek přijde na server jako ASP.NET ovládací prvek, ASP.NET stránka vystaví mnoho událostí, které mohou být konzumovány vloženou logikou.

Seq	Událost	Vlastnosti události
1	`Init`	• Ovládací prvky inicializovány? **Ne**. • `ViewState` dostupné? **Ne**. • `Form` data dostupné? **Ne**. Můžeme přistupovat k formulářovým datům užitím ASP.NET request objektu, nikoli však pomocí serverových ovládacích prvků. Vytváření prvků dynamicky, v případě, že potřebujeme ovládací prvky vytvářet v běhovém prostředí. Inicializace nastavení. `MasterPages` a jejich nastavení. Nemáme zde přístup k `viewState`, odeslaným hodnotám ani inicializovaným ovládacím prvkům.
2	`Load viewstate`	• Ovládací prvky inicializovány? **Nezaručeno**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Nezaručeno**. Můžeme přistupovat k `viewStates` a jakékoli logice, kde chceme viewState předat proměnným v kódu na pozadí.
3	`Postback` data	• Ovládací prvky inicializovány? **Nezaručeno**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Můžeme přistupovat k formulářovým datům. Vhodné k jakékoli logice, kdy chceme formulářová data předat proměnným v kódu na pozadí.

4	`Load`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Toto je místo, kde položíme jakoukoli logiku na operování s ovládacími prvky. V této události jsme získali přístup ke všem ovládacím prvkům, viewstates a odeslaným hodnotám.
5	`Validate`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Jestliže má stránka validátory nebo chceme vyvolávat validaci našich stránek, pak toto je správné místo.
6	`Event`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Jestliže toto je postback z kliknutí tlačítka nebo změny dropdown, pak se vztahující událost vyvolá. Jakýkoli kód k obsloužení této události může být zde.

7	`Prerender`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Prostor pro závěrečné změny na UI objektech, jako je změna stromové struktury nebo hodnoty nějaké vlastnosti, dřív, než se tyto ovládací prvky uloží do `viewState`.
8	`Save viewstate`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Jakmile jsou všechny změny serverových ovládacích prvků vykonány, pak tato událost může být příležitost na uložení dat ovládacích prvků do `viewState`.
9	`Render`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Jestliže chceme přidat nějaký vlastní HTML kód na výstup, tady je na to správná chvíle.
10	`Unload`	• Ovládací prvky inicializovány? **Ano**. • `ViewState` dostupné? **Ano**. • `Form` data dostupné? **Ano**. Jestliže chceme vložit kód na úklid po předešlých událostech.

Tabulka 3. Fáze zpracování objektu stránky

Životní cyklus ASP.NET MVC 1 stránky
Podle Sandersona [3] str.213 – str.219.

Vždy, když http požadavek dorazí na IIS, ovladač jádra Windows `HTTP.SYS` rozhodne podle kombinace URL/ číslo portu/ IP adresa a předá požadavek příslušné aplikaci. ASP.NET může běžet v jednom ze dvou řízených módů vykonávání. První možností je ISAPI mód (taky zvaný klasický mód), přes rozšíření `aspnet_isapi.dll` podle přípony (např.: `.aspx`, `.ashx`, `.mvc`), druhou možností je integrovaný mód (podporovaný od IIS 7, .NET přirozená část IIS požadavků na vykonávací cyklus, není tudíž potřeba ISAPI rozšíření a její asociace s jednotlivými příponami. Z toho následně vyplývá čisté URL.)

Příchozí Http požadavek do IIS/ASP.NET je obsloužen ve třech postupných oblastech

- jádro směrovacího mechanismu (Core routing),
- řadiče (controllers) a akce a
- pohledy.

a to následujícím způsobem:

1. krok: Požadavek se předá `UrlRoutingModule`. To způsobí, že se rozběhne směrovací systém System.Web.Routing. Modul vytvoří request context. Dále modul Zkontroluje, zda příchozí url koresponduje se souborem na disku (to znamená pro statické soubory – např.: `.gif`, `.jpeg`, `.png`, `.css`, `.js` je obslouží přirozeně přímo). Obdobně jsou vykonány klasické ASP.NET WebForms `.aspx` stránky. U stránek rozhodne podle routovací konfigurace (kolekce System.Web.Routing.RouteTable.Routes). Tuto kolekci naplní vývojář v metodě `RegisterRoute()` v `Global.asax.cs` souboru.

2. krok: Prochází se skrz tyto směrovací položky a rozhodne se podle první vyhovující. V závislosti na tom se rozhodne o řadiči, kterému se předá řízení. Přesněji je použitý controller factory objekt. Výchozí je `DefaultControllerFactory`, který následuje dílčí jmennou konvenci a konvenci jmenných prostorů. Tak se předá řízení konkrétnímu řadiči. Tato logika může být námi změněná. Stačí vytvořit vlastní objekt implementující `IControllerFactory`, či vytvořit podtřídu od

`DefaultControlllerFactory`. Po zavolání controlleru se vykoná veřejná akce (action method). Tato metoda může přímo poslat výstup do http odpovědi, avšak není to obvyklé. Obvyklé je vracet `ActionResult`, který popisuje zamýšlený výstup. Tyto akce mají flexibilní systém filtrů, které se připojují před hlavičku akce, jako atribut. Tak můžeme vložit vlastní logiku, která se vykoná před či po akci.

3. krok: Návratový typ akce mohou být různé podtřídy `ActionResult` (např.: `ViewResult`) vykreslí pohled, v závilosti na použitém „view engine“ implementující `IViewEngine`. Výchozí „view engine“ je `WebFormsViewEngine`, kdy pohledy jsou WebForms ASPX stránky. V MVC (kvůli oddělení působnosti) by na rozdíl od WebForms neměla mít jinou funkci než vygenerování HTML. Proto není potřeba znalost životního cyklu stránky ASP.NET WebForms. Oddělení působnosti přináší jednoduchost a udržovatelnost. Navíc i WebForms view engine použitý jako výchozí v MVC můžeme nahradit vlastním, který implementuje `IViewEngine`.

Zhodnocení: Z popisu životního cyklu stránky v ASP.NET WebForms a ASP.NET MVC 1 je zcela zřejmé, že životní cyklus stránky ve druhém jmenovaném frameworku je podstatně flexibilnější a jednodušší. Stejně tak je rozšiřitelnost přirozeným rysem tohoto frameworku, rozšiřování ve WebFoms je naopak obtížné.

Skóre kritéria „Zpracování HTML requestů a tvorba komponent“	
ASP.NET WebForms	**ASP.NET MVC 1**
0	1

Tabulka 4. Zpracování HTML requestů a tvorba složitějších znovupoužitelných komponent

5.2.2. Efektivita vývoje

V rámci tohoto kritéria jsou srovnávány kvantitativní ukazatele, tj. počty

- příkazů v kódu (bez testů),
- příkazů ve stránkách,
- tagů ve stránkách a
- atributů ve stránkách.

Efektivnějším je ten framework, který má tyto ukazatele nižší.

Efektivta vývoje			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Příkazy v kódu (bez testů)	1511	1144	0: 1
Příkazy ve stránkách	311	663	1: 0
Tagy ve stránkách	1769	1815	1: 0
Atributy ve stránkách	3365	1407	0: 1
Výsledek srovnání			**2: 2**

Tabulka 5. Efektivita vývoje.

Zhodnocení: Efektivita vývoje je v obou srovnávaných frameworcích, jak je patrné z měření, v zásadě srovnatelná. Za zmínku nicméně stojí nesrovnatelně větší nutnost používání atributů v ASP.NET WebForms. Z toho je zřejmé, že metoda práce v WebForms je deklarativní, kdežto metoda práce v MVC je spíš imperativní.

Skóre kritéria „Efektivita vývoje“	
ASP.NET WebForms	**ASP.NET MVC 1**
0,5	0,5

Tabulka 6. Skóre efektivity vývoje.

5.2.3. Učící křivka

V rámci tohoto kritéria je srovnávána náročnost získávání a vštěpování informací o problematice, přičemž tam, kde je náročnost získávání a vštěpování informací vyšší, tam je učící křivka mírnější a vývoj aplikace z tohoto hlediska méně výhodný.

Při interpretaci výsledků srovnání frameworků ASP.NET WebForms a ASP.NET MVC 1 z hlediska tohoto parametru je třeba zohlednit, že hodnocení je subjektivní. Učící křivku jednoznačně ovlivňuje aktuální profesní vybavenost vývojáře, jeho návyky a předchozí zkušenosti s programováním, nejen webových aplikací. Já sám jsem před prací ve srovnávaných frameworcích programoval v Basicu, Pascalu, Javě, HTML, PHP a C#.

Učící křivka	
ASP.NET WebForms	**ASP.NET MVC 1**
Učící křivka je strmá, začátečník se tento framework rychle naučí používat. Křivka mírně kolísá, nevýhodou frameworku je „probublávání“ událostí a omezení jedním formulářem na stránku.	Učící křivka je mírná, model zpracování stránky je zcela odlišný. Je jen málo dostupné literatury, v začátku mé práce jsem neměl k dispozici žádné zdroje v češtině.

Tabulka 7. Učící křivka.

Zhodnocení: Celkově je podle mého soudu programování s využitím ASP.NET WebForms dokumentováno lépe než v ASP.NET MVC, učící křivka je u ASP.NET WebForms strmější.

Skóre kritéria „Učící křivka“	
ASP.NET WebForms	**ASP.NET MVC 1**
1	0

Tabulka 8. Skóre učící křivky

5.2.4. Jednoduchost vývoje

V rámci tohoto kritéria jsou srovnávány kvantitativní ukazatele, tj.

- počet řádků zdrojového kódu (bez testů) a celkový počet řádků zdrojového kódu (s testy),
- počet tříd (bez testů) a celkový počet tříd (s testy),
- počet rutin (bez testů) a celkový počet rutin (s testy),
- počet stránek, respektive pohledů, a počet uživatelských ovládacích prvků, respektive dílčích pohledů,
- počet řádků ve stránkách a uživatelských ovládacích prvcích; respektive pohledech a dílčích pohledech, včetně vlastních javascriptových funkcí.

Za framework, ve kterém je vývoj aplikace jednodušší, je pokládán ten s nižšími ukazateli.

Jednoduchost vývoje 1			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Počet řádků (bez testů)	1082	1420	1: 0
Celkový počet řádků	40072	7838	0: 1
Počet tříd (bez testů)	57	19	0: 1
Celkový počet tříd	179	36	0: 1
Počet rutin (bez testů)	216	325	1: 0
Celkový počet rutin	3542	1239	0: 1
Výsledek dílčího srovnání			**2: 4**

Tabulka 9. Jednoduchost vývoje – kód na pozadí, resp. v kontrolerech a modelech.

Jednoduchost vývoje 2			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Počet stránek resp. pohledů	44	48	1: 0
Počet uživatelských ovládacích prvků, resp. dílčích pohledů	11	5	0: 1
Počet řádků ve stránkách a uživatelských ovládacích prvcích, resp. pohledech a dílčích pohledech + vlastních javascriptových funkcí	4344 + 72	3294 + 85	0: 1
Výsledek dílčího srovnání			**1: 2**

Tabulka 10. Jednoduchost vývoje – kód ve stránce, resp. pohledu.

Zhodnocení: ASP.NET WebForms vychází lépe ve srovnání počtu řádků kódu (bez testů), počtu rutin (bez testů) a v počtu stránek. Všude jinde dominuje ASP.NET MVC 1. Jeví se být lepší především ve snadnosti testovatelnosti. Podotýkám nicméně, že jsem nebral v potaz počet dílčích pohledů oproti uživatelským ovládacím prvkům, nýbrž celkový součet stránek a uživatelských ovládacích prvků na jedné straně a na druhé straně součet pohledů a dílčích pohledů.

Pokud bych odhlédl od faktorů testovatelnosti, vyšlo by pomocné skóre ASP.NET WebForms : ASP.NET MVC 1 jako 3:3. Ovšem pokud bych zároveň zohlednil ve srovnání celkový počet řádků v kódu a celkový počet řádků ve

stránce (+ uživatelských ovládacích prvcích), respektive v pohledu (+ dílčích pohledů), vyšlo by srovnání lépe opět pro ASP.NET MVC 1, ať už bychom počítali i počet řádků testů, či nikoliv.

Sám ještě považuji z hlediska jednoduchosti vývoje za přínosnou integraci MVC do IntelliSence ve VisualStudiu ve vázání modelu (dat) do pohledu (stránky). To při konstrukci Eval ve WebForms nebylo přítomno.

Skóre kritéria „Jednoduchost vývoje“	
ASP.NET WebForms	**ASP.NET MVC 1**
0,33	0,67

Tabulka 11. Skóre jednoduchosti vývoje

5.2.5. Jednoduchost údržby

V rámci tohoto kritéria je srovnáván celkový a průměrný index udržovatelnosti a cyklomatická složitost. Index udržovatelnosti je vlastní metrika Visual Studia 2010 Premium Edition a výše, cyklomatická složitost je metrika převzatá z Visual Studia 2010 Premium Edition a výše, přičemž čím víc je větvení, tím větší je cyklomatická složitost. Dále jsou srovnávány maximální hloubka dědění a celková a průměrná provázanost tříd (bez testů a celkem), tj. metriky popisující ze své definice udržovatelnost.

Jednodušší údržba je ve frameworku, který dosahuje vyšších hodnot u indexu udržovatelnosti a naopak nižších hodnot u cyklomatické složitosti, maximální hloubky dědění a provázanosti tříd.

Jednoduchost údržby – index udržovatelnosti			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Originální	68	74	0: 1
Průměrný bez testů	78,3	82,4	0: 1
Průměrný celkem	67,9	72,9	0: 1
Výsledný poměr dílčího srovnání			**0: 1**

Tabulka 12. Jednoduchost údržby – index udržovatelnosti

Jednoduchost údržby 2 – cyklomatická složitost			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Bez testů	536	677	1: 0
Celkem	4144	1950	0: 1
Výsledný poměr dílčího srovnání			**0,5: 0,5**

Tabulka 13. Jednoduchost údržby – cyklomatická složitost

Jednoduchost údržby 3 – maximální hloubka dědění			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Bez testů	5	4	0: 1
Celkem	5	4	0: 1
Výsledný poměr dílčího srovnání			**0: 1**

Tabulka 14. Jednoduchost údržby – maximální hloubka dědění

Jednoduchost údržby 3 – maximální hloubka dědění			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Originální	169	125	0: 1
Průměrná, jednotlivě, bez testů	6,3	3,7	0: 1
Průměrná, jednotlivě, celkem	6,8	7,4	1: 0
Celkem	5	4	0: 1
Výsledný poměr dílčího srovnání			**0,33: 0,67**

Tabulka 15. Jednoduchost údržby – provázanost tříd

„Jednoduchost údržby“		
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**
Index udržovatelnosti	0	1
Cyklomatická složitost	0,5	0,5
Maximální hloubka dědění	0	1
Provázanost tříd	0,33	0,67
Celkem	0,83	3,17

Tabulka 16. Souhrn dílčích srovnání jednoduchosti údržby

Zhodnocení: Z hlediska udržovatelnosti vychází ze srovnání podstatně lépe technologie ASP.NET MVC 1.

Skóre kritéria „Jednoduchost údržby“	
ASP.NET WebForms	**ASP.NET MVC 1**
0,21	0,79

Tabulka 17. Skóre jednoduchosti údržby

5.2.6. Modularita

V rámci tohoto kritéria jsou srovnávány nahraditelnost a plná zastupitelnost jednotlivých komponent srovnávaných frameworků s jinými dalšími komponentami, přičemž čím snadnější je nahraditelnost a zastupitelnost, tím vyšší je modularita.

Modularita		
ASP.NET WebForms	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Není modulární. Jako základní modul může sloužit dvojice stránka a kód na pozadí. Avšak ani tak není řešeno oddělení aplikační a prezentační logiky. Často je také jak ve stránce, tak v kódu na pozadí přítomná složka pracující s daty.	Je modulární. Každá část aplikace má na starosti jen specifickou část – modely, pohledy, ovládání.	0: 1

Není možná nahraditelnost.	Je možná nahraditelnost.	0: 1
Výsledek srovnání		**0:2**

Tabulka 18. Modularita.

Zhodnocení: Modularita ASP.NET MVC 1 je jednoznačně vyšší oproti ASP.NET WebForms.

Skóre kritéria „Modularita“	
ASP.NET WebForms	**ASP.NET MVC 1**
0	1

Tabulka 19. Skóre modularity

5.2.7. Testovatelnost

V rámci tohoto kritéria je srovnávána jednoduchost vytváření unitních testů. Jednoduchost je určována prostřednictvím kvantitativních metrik, tj. počtem řádků testů, počtem tříd testů, počtem rutin testů, průměrným počtem řádků na třídu a průměrným počtem řádků na rutinu. Jednodušší je testovatelnost u frameworku s nižšími hodnotami.

Testovatelnost			
Ukazatel	**ASP.NET WebForms**	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Počet řádků testů	38990	6418	0: 1
Počet tříd testů	122	17	0: 1
Počet rutin testů	3326	914	0: 1
Průměrný počet řádků na třídu	319,6	377,5	1: 0
Průměrný počet řádků na rutinu	11,7	7	0: 1
Výsledek srovnání			**1: 4**

Tabulka 20. Testovatelnost.

V testování ASP.NET WebForms jsou kromě testů na viditelnost prvků začleněny i testy na chování. ASP.NET MVC 1 oproti tomu má unit testy, pomocí nichž jsem se při testování v maximální možné míře přiblížil testům chování. Měl-li bych k dispozici IIS od verze 6 výše, postupoval bych při plnohodnotném integritním testování, včetně komplexnějších testů chování, stejně jako v případě testu chování v ASP.NET WebForms (NUnitAsp a webový server). V případě ASP.NET WebForms postačuje k testování nainstalovaný

IIS 5.1. U ASP.NET MVC 1 už však je můj testovací počítač s Windows XP a IIS 5.1 nevyhovující a to kvůli absenci podpory MVC technologie. Na druhou stranu je výhodou technologie ASP.NET MVC 1, i jejích pozdějších verzí, možnost vytvářet „mockable“ objekty, kdy se funkcionalita otestuje na jiných než produkčních datech.

Zhodnocení: Ze srovnání testovatelnosti vychází jednoznačně lépe technologie ASP.NET MVC. Zvláště ukazatel průměrný počet řádků na rutinu je víc než o třetinu nižší oproti počtu v ASP.NET WebForms. Navíc ASP.NET WebForms nelze testovat bez webového serveru.

Skóre kritéria „Testovatelnost“	
ASP.NET WebForms	**ASP.NET MVC 1**
0,2	0,8

Tabulka 21. Skóre testovatelnosti

5.2.8. Snadnost nasazení aplikace do produkčního prostředí

V rámci tohoto kritéria je srovnávána snadnost publikace webové aplikace z lokálního počítače, kde je aplikace vyvíjena, na webový server.

Snadnost nasazení aplikace do produkčního prostředí		
ASP.NET WebForms	**ASP.NET MVC 1**	**Pomocné skóre WF: MVC**
Publikace je snadná.	Publikace je snadná.	0,5: 0,5
Visual Studio má zabudovaný publikační nástroj.	Visual Studio má zabudovaný publikační nástroj, problémové je ale nasazení na starší IIS, ideální je IIS 7 a novější.	1: 0
Výsledek srovnání		**1,5: 0,5**

Tabulka 22. Snadnost nasazení aplikace do produkčního prostředí .

Zhodnocení: Ze srovnání snadnosti nasazení aplikace do produkčního prostředí vychází lépe technologie ASP.NET WebForms.

Skóre kritéria „Snadnost nasazení aplikace“	
ASP.NET WebForms	**ASP.NET MVC 1**
0,75	0,25

Tabulka 23. Skóre snadnosti nasazení aplikace do produkčního prostředí

5.3. Shrnutí všech kritérií

	Kritérium	Skóre WF: MVC
1	Zpracování HTML requestů, tvorba složitějších znovupoužitelných komponent	0: 1
2	Efektivita vývoje	0,5: 0,5
3	Učící křivka	1: 0
4	Jednoduchost vývoje	0,33: 0,67
5	Jednoduchost údržby	0,21: 0,79
6	Modularita	0: 1
7	Testovatelnost	0,2: 0,8
8	Snadnost nasazení aplikace do produkčního prostředí	0,75: 0,25
Výsledek srovnání		**2,99: 5,01**

Tabulka 24. Shrnutí všech kritérií

Zhodnocení: Z výsledku srovnání frameworků ASP.NET WebForms a ASP.NET MVC 1 vychází pro vývoj webových aplikací jako výhodnější technologie ASP.NET MVC 1. resp. ASP.NET MVC 1 se jeví být výhodnější pro vývoj takového typu aplikací, jaký jsem použil ke srovnání já. Pokud by byl srovnáván vývoj jiného typu aplikace, zejména, jednalo-li by se o jednodušší aplikace, je pravděpodobné, že by se výsledek lišil.

Velkou výhodou ASP.NET WebForms je snadnost učení, k dispozici je dostatek studijního materiálu jak v angličtině, tak v češtině. ASP.NET WebForms má rovněž převahu z hlediska snadnosti nasazení aplikací do produkčního prostředí, především při nasazení na IIS servery starší než 7.0.

Naproti tomu ASP.NET MVC 1 má snáz pochopitelný model zpracování HTML požadavků. Jednoduchost vývoje je u ASP.NET MVC 1 o málo lepší než v ASP.NET WebForms, suverénně pak ovšem dominuje v oblasti jednoduchosti údržby, modularity a testovatelnosti.

Silné a slabé stránky jednotlivých frameworků shrnuje následující tabulka.

Silné a slabé stránky frameworku		
	ASP.NET WebForms	**ASP.NET MVC**
Výhody	• Strmá učící křivka. • Dostupnost komponent. • Snaží se o zavedení stavovosti (podobně jako u WinForms).	• Třívrstvá architektura. • Snazší testovatelnost. • URL je přátelské k vyhledávačům. • Kontrola nad kódem (i v pohledech). • Snazší modifikace.
Nevýhody	• Obtížná testovatelnost. • Těžší modifikace.	• Mírná učící křivka. • Není `ViewState`. • Není přístup k `PostBack`.

Tabulka 25. Silné a slabé stránky frameworků

Závěr

V této práci jsem se věnoval srovnání frameworků ASP.NET WebForms a ASP.NET MVC z hlediska vývoje aplikací v nich.

V současnosti dostupná srovnání se mi jeví jako příliš triviální, neboť jim chybí opora v konkrétní aplikaci, na které by byly frameworky porovnány. Pro vlastní srovnání technologií jsem vytvořil středně složitou webovou aplikaci, která využívá v maximální míře rysy jednotlivých frameworků.

Oba porovnávané frameworky se používají na programování dynamických internetových stránek. Jejich nosnou myšlenkou je oddělení vykonávající logiky, každá technologie toho však dociluje jiným způsobem. ASP.NET WebForms se zaměřuje na oddělení deklarativní části a kódu na pozadí, ASP.NET MVC pak rozděluje vykonávání programu do tří částí, jimiž jsou model dat, uživatelské rozhraní a řídící třídy.

Vyvinul jsem malý komunitní systém, včetně jednoduchých modulů. K dispozici jsem měl nepoměrně větší množství literatury o ASP.NET WebForms oproti ASP.NET MVC, některé problémy při vývoji aplikace jsem v důsledku toho řešil s komplikacemi a zbytečně složitě. Takovým řešeným problémem bylo např. začlenění html editoru, který je jedním z ajaxových ovládacích prvků a vyžaduje ScriptManager.

Později jsem si doplnil znalosti i k technologii ASP.NET MVC. Tato se mi jeví lepší po stránce udržovatelnosti, modularity, testovatelnosti a , pokud má vývojář potřebné znalosti, je lepší i v jednoduchosti vývoje a zpracování html požadavků. ASP.NET WebForms má proti ASP.NET MVC výhodu strmé učící křivky a nižších nároků na nasazení. Z hlediska efektivity, pokud nezohledňuji testovatelnost, vychází srovnání frameworků na užité aplikaci podobně.

V souhrnu jako jednoznačně lepší technologií pro vývoj webových aplikací hodnotím framework ASP.NET MVC 1.

Conclusions

I have followed a confrontation framework ASP.NET WebForms and ASP.NET MVC in light of developing applications with them.

Existing comparisions seemed too trivial to me. The reason was missing application to compare frameworks to it. Therefore, I has made own moderately difficult application, which use in full rate features of frameworks.

Both frameworks are used to programming dynamic internet pages. They are both separating executing logic, but each one by different way. ASP.NET WebForms is focuses on separating declarative part of code and code behind. ASP.NET MVC divided program executing into three parts – data model, user interface and controll classes.

I developed the small community system first, including simple modules. I haved much more literature available about ASP.NET WebForms than about ASP.NET MVC. So I solved some problems more difficult than was necessary. The example of this is including html editor, which is one of the ajax control toolkit requiring a ScriptManager.

However, after updating my knowledge of ASP.NET MVC technology, it seems to me better in terms of maintainability, modularity, testability and after this knowledge is better in ease of development and processing html requests. On the contrary the biggest advantage of ASP.NET WebForms is a steep learning curve and low profile requirements for deployment. In terms of efficiency are both frameworks on examined application equally, but only, when we do not take into account the testability.

As clearly better technology for developing web applications I evaluate the above reasons ASP.NET MVC 1 framework.

Reference

[1] Bellinaso, Marco. *Webové programování v ASP.NET 2.0 problém, návrh, řešení.* Computer Press, Brno, 2007. ISBN 978-80-251-1893-1.

[2] MacDonald, Matthew. Szpuszta, Mario *ASP.NET 3.5 a C#2008, tvorba dynamických stránek PROFESIONÁLNĚ.* Zoner Press, Brno, 2008. ISBN 978-80-7413-008-3.

[3] Sanderson, Steven *Pro ASP.NET MVC Framework.* Apress, New York, 2009. ISBN-13 (electronic): 978-1-4302-1008-5.

[4] Pecinovský, Rudolf *Návrhové vzory.* Computer Press, Brno, 2007. ISBN 978-80-251-1582-4.

[5] McConnell, Steve *Dokonalý kód, umění programování a techniky tvorby software.* Computer Press, Brno, 2005. ISBN 80-251-0849-X.

[6] Price, Jason *C# programování databází.* Grada Publishing, Praha, 2005. ISBN 80-247-0982-1.

[7] Thau, Dave *Velký průvodce JavaScriptem – tvorba interaktivních webových stránek v praxi.* Grada, Praha, 2009. ISBN 978-80-247-2211-5.

[8] McFarland, David Sawyer *CSS – chybějící manuál.* Grada Publishing, Praha, 2007. ISBN 978-80-247-2122-4.

[9] *http://asp.net/web-forms/.* Elektronická publikace, 21.4.2010.

[10] *http://asp.net/mvc/.* Elektronická publikace, 21.4.2010.

[11] *http://en.wikipedia.org/wiki/Model-view-controller.* Elektronická publikace, 21.4.2010.

[12] *http://cs.wikipedia.org/wiki/MVC.* Elektronická publikace, 16.2.2011.

[13] *http://rarous.net/weblog/tag/mvc.aspx.* Elektronická publikace, 14.3.2011.

[14] *http://haacked.com/tags/aspnetmvc/default.aspx.* Elektronická publikace, 14.3.2011.

[15] *http://www.hanselman.com/blog/CategoryView.aspx?category=ASP.NET+MVC.* Elektronická publikace, 14.3.2011.

[16] *http://weblogs.asp.net/scottgu/.* Elektronická publikace, 14.3.2011.

[17] *http://www.codeproject.com/KB/aspnet/ASPDOTNETPageLifecycle.aspx.* Elektronická publikace, 19.7.2010.

[18] *http://cs.wikipedia.org/wiki/ASP.NET.* Elektronická publikace, 1.6.2011.

[19] *http://programujte.com/?akce=diskuze&kam=vlakno&tema=15939-rozhodovani-mezi-asp-net-webforms-nebo-mvc.* Elektronická publikace, 30.5.2010.

[20] *http://msdn.microsoft.com/en-us/library/dd381619.aspx.* Elektronická publikace, 7.5.2011.

[21] *http://stackoverflow.com/questions/102558/biggest-advantage-to-using-asp-net-mvc-vs-web-forms.* Elektronická publikace, 22.9.2008.

A. Pokyny pro instalaci

1. Předpoklad pro lokální spouštění.
   - Nainstalujte IIS 5.1 nebo vyšší – nutné pro potřeby testování ASP.NET WebForms.
   - Nainstalujte MSSQL Express 2008 (stačí však i MSSQL 2005 Express), pokud už není (nemusí být Express verze).
2. Spuštěním skriptu `scriptMM.sql` v SQL Server Management Studiu nainstalujete potřebné databázové tabulky a uložené procedury – je nutné pojmenovat nový InitialCatalog jako `MM`.
3. Nastavte plná práva účtu „localhost/ASPNET“ ke katalogu `MM` v databázi.
4. Upravte connectionString na aktuální a to v těchto umístěních:
   - u WebForms
     (a) v `web.config` – connectionString se jménem `MM_aspnetdb`,
     (b) ve složce `Tests` soubor `NunitAdapter.cs` metoda `Init()` – její proměnnou `connectionString` (řádek 66),
     (c) modul `DataAndLogic` – složka `DL` soubor `DataAccessProvider.cs`, konstruktor `DataAccessProvider()` a jeho catch klauzule proměnná `connectString`, (řádek 21);
   - u MVC
     (a) v `web.config` v adresáři MakoviceMVC stejně tak `web.config` v adresáři `MakoviceMVC/Tests` a sice `connectionString` se jmény `MM_aspnetdb`, `MMEntities` a `MMEntities1`,
     (b) modul `DataAndLogic` (adresář `DataAndLogicMVC`) složka `DL` soubor `TestDatabaseAccessProvider.cs` proměnná `connectString` (řádek 16),
     (c) modul `DataAndLogic` (adresář `DataAndLogicMVC`) složka `DL` soubor `DataAccessProvider.cs`, konstruktor `DataAccessProvider`, klauzule catch, proměnná `connectString` (řádek 21).
5. Pro potřeby MVC, pokud není, nainstalujte MVC 1 pomocí AspNetMVC1.msi.
6. Nahrajte webovou aplikaci WebForms na lokální webový server. – pro potřeby testů v WebForms a z důvodu dostupnosti `admin` role nazvěte aplikaci `MMWebForms`.

7. V případě, že webový server běží na .NET 4 – upravte `web.config` v kořenu webového serveru o:

```
<system.web>
<httpRuntime requestValidationMode="2.0" />
<pages validateRequest="false"/>
</system.web>
```

B. Pokyny pro převod do Visual Studia 2010

1. Při převodu mých webových aplikací do Visual Studia 2010 – nechejte aktuální verzi .NET (nepovyšovat).
2. při převodu mých MVC webových aplikací – v souboru `ItemHelper.cs`:
   (a) řádek 39 – změnit `string` na `MvcHtmlString`,
   (b) řádek 40 – změnit `aTag` na `aTag.ToString()`.

C. Postup testování

1. Předpoklad pro testování
   (a) Je nainstalovaný program NUnit – volně stažitelný.
   (b) Předpokladem pro testování WebForms je správně nainstalovaná aplikace ASP.NET WebForms na lokálním počítači v MMWebForms (viz pokyny pro instalaci) včetně podpůrné databáze MM.
2. Testování
   (a) Pro samotné testování je potřeba, aby databáze příslušná connnectionStringům, byla nainstalována (viz pokyny pro instalaci) a běžící.
   (b) Pro samotné testování je dále v případě WebForms potřeba, aby byl spuštěný lokální server, kde je aplikace nahrána.
   (c) Spuštění NUnit.
   (d) Nahrání testů do NUnit
      - u WebForms – kliknutí na složku s webovou aplikaci ASP.NET WebForms a její projektový soubor, který je umístěn v lokálním serveru (u mě `C:\Inetpub\wwwroot\MMWebForms\MakoviceWebForms.csproj`),
      - u MVC – kliknutí na binární dll soubor `MakoviceWebForms.dll` (u mě `C:Documents and Settings\Tomáš Podešva\Dokumenty\Visual Studio 2008\Projects\MakoviceWebForms\MakoviceWebForms\bin\MakoviceWebForms.dll`).
   (e) Kliknutí na tlačítko Run.

D. Obsah přiloženého CD

`bin/`
: Kompletní adresářová struktura obou webových aplikací pro zkopírování na webový server (v ZIP archivu). Adresář obsahuje i všechny potřebné knihovny a další soubory pro bezproblémový provoz na webovém serveru.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu).

`src/`
: Kompletní zdrojové texty webových aplikací se všemi potřebnými (i převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření adresářové struktury pro zkopírování na webový server (v ZIP archivu).

`readme.txt`
: Instrukce pro nasazení webových aplikací na webový server, včetně požadavků pro jejich provoz, a webová adresa, na které jsou aplikace nasazeny pro testovací účely a pro účel obhajoby práce.

Navíc CD obsahuje:

`tools/`
: Nástroje pro měření metrik kódu a nástroje pro testování.

U veškerých odjinud převzatých materiálů obsažených na CD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru `readme.txt`.