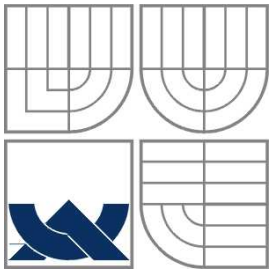
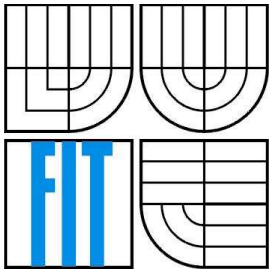


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NOVÉ MOŽNOSTI RÁMCE .NET 2.0 PRO TVORBU INFORMAČNÍHO SYSTÉMU

NEW POSSIBILITIES OF .NET FRAMEWORK 2.0 FOR INFORMATION SYSTEM IMPLEMENTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — TOMÁŠ MARŠÁK
AUTHOR

VEDOUCÍ PRÁCE — ING. ZBYNĚK KŘIVKA
SUPERVISOR

BRNO 2007

Abstrakt

V této práci jsou popsány nové možnosti rámce .NET 2.0 a nové verze ASP.NET 2.0. Dále zde jsou v krátkosti představeny nové vývojové nástroje, které jsou potřeba pro vývoj webových aplikací.

Hlavním obsahem této práce je předvedení nových komponent obsažených v ASP.NET 2.0. Převážně se jedná o komponenty pro správu uživatelů, Master Pages, komponenty pro zobrazení dat a nových rozšiřujících možností skinování webových aplikací.

Klíčová slova

ASP.NET 2.0, .NET Framework, Membership, Login, LoginStatus, LoginView, ChangePassword, CreateUserWizard, Themes, Master Pages, GridView, DetailsView, FormView

Abstract

In this project are described new possibilities of .NET 2.0 Framework and new version ASP.NET 2.0. There are presentation of new development tools, which we need to development of web applications here.

Main subject of this project is demonstrating of new controls, which are included in ASP.NET 2.0. Mainly, they are user management control, Master Pages, monitoring of data display and new extension possibilities of skinning of web applications.

Keywords

ASP.NET 2.0, .NET Framework, Membership, Login, LoginStatus, LoginView, ChangePassword, CreateUserWizard, Themes, Master Pages, GridView, DetailsView, FormView

Citace

Tomáš Maršák: Nové možnosti rámce .NET 2.0 pro tvorbu informačního systému, Brno, FIT VUT v Brně, 2007

Nové možnosti rámce .NET 2.0 pro tvorbu informačního systému

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zbyňka Křivky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…….....
Tomáš Maršák

© Tomáš Maršák, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Úvod

ASP.NET 2.0 přináší celou řadu nových technologií, které společně s prostředím Visual Studio 2005 vedou k významnému zrychlení a ulehčení práce na webových aplikacích. Jedná se zejména o novinky, jako jsou Master Pages, témata, správa uživatelů a v neposlední řadě datové komponenty pro jednoduché zobrazování dat z libovolného datového zdroje, v našem případě se bude jednat konkrétně o data získaná z databázových tabulek umístěných na SQL serveru.

Každý vývojář webových aplikací se stále opakovaně setkává s problémem implementace správy uživatelů. ASP.NET 2.0 tuto problematiku řeší pomocí Membership API. Proto dalším cílem této práce je představit nové webové kontroly, které ulehčují vývojáři navrhnout webovou aplikaci, která by umožňovala jednoduché, efektivní a v neposlední řadě bezpečné řešení autorizace uživatelů a jejich správy. Jsou zde představeny nástroje a postupy pro vytvoření jednotlivých částí webové aplikace, které jsou zaměřeny na přihlášení, registraci a správu uživatelů.

Dále jsou zde popsány metody pro jednoduché rozvržení webových aplikací. Každá stránka se většinou skládá z hlavičky, navigační části, hlavního těla stránky, kde jsou zobrazovány zvolené informace, a ze zápatí stránky. Takovéto rozvržení obsahuje statické a většinou neměnné části webové aplikace. ASP.NET 2.0 nám dává možnost pomocí Master Pages tuto situaci jednoduše a efektivně řešit. Master Pages nám umožňují definovat statické prvky stránky, které jsou stejné pro všechny části webové aplikace, tak, aniž bychom museli v každé stránce dokola opisovat tento stejný kód. Dále nám vymezí oblast, kterou používáme pro zobrazení jednotlivých obsahových částí stránek.

ASP.NET 2.0 přináší ještě nové možnosti, co se týče návrhu vzhledu webových aplikací. Jedná se konkrétně to tzv. „Themes“. Themes neboli motivy nám umožňují měnit vzhled vkládaných ASP kontrol a popřípadě definovat jejich vlastnosti.

Posledním bodem je ukázka a představení nových kontrol pro jednoduché zobrazování dat. Tyto kontroly, mezi které patří např. GridView, DetailsView a FormView vývojáři umožňují jednoduchou a rychlou implementaci formulářů svázaných z daným datovým zdrojem webové aplikace. Nabízejí mimo jiné možnosti editace, vkládání a úpravy dat.

Všechny tyto kontroly jsou implementovány v jednoduchém ukázkovém informačním systému. Ten je umístěn na freehostingovém serveru pro jednoduchý přístup.

Adresa ukázkového IS:
http://marsos.qsh.cz
Přihlašovací údaje administrátora IS:
Uživatelské jméno: Maršos
Heslo: b8jugoca

Použití a funkcionalita tohoto IS je popsána v článku vloženém do tohoto ukázkového IS. Jedná se o jednoduchý redakční systém, který umožňuje:

- vkládání článků
- vkládání upoutávkových obrázků k článkům
- dělení článků do kategorií
- možnost komentování vložených článků
- vytváření diskusních fór
- vkládání příspěvků do těchto fór registrovaným uživatelům
- správu uživatelů
- správu uživatelských rolí

Potřebné softwarové vybavení

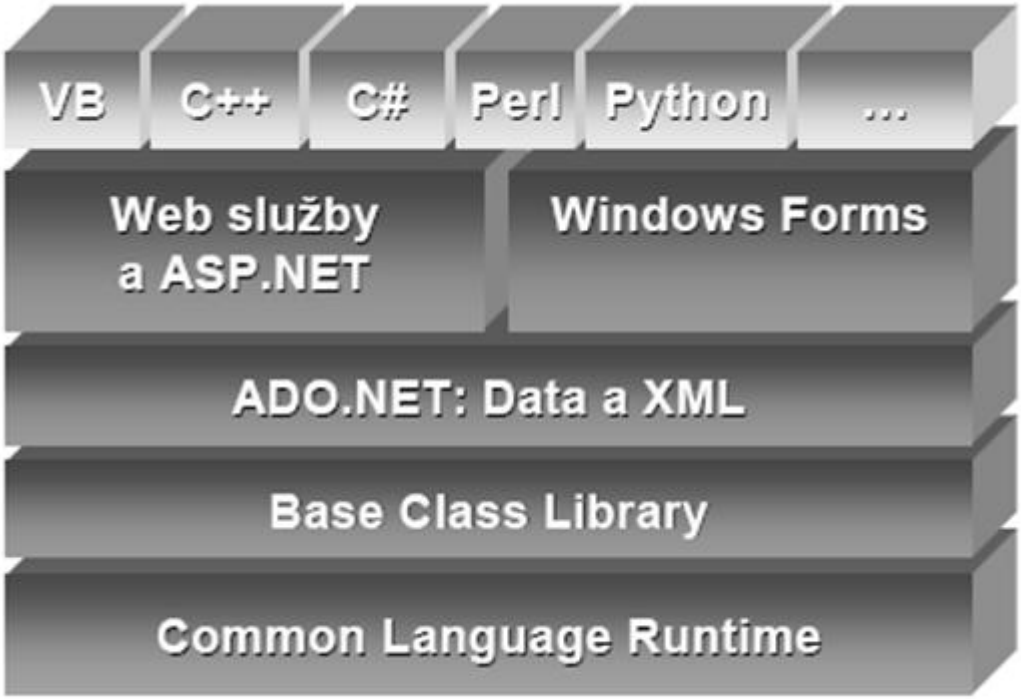
1.1 .NET framework

Pro činnost webových stránek v ASP.NET 2.0 je třeba komponenta zvaná „Microsoft . NET Framework 2.0“ , můžeme se setkat i se staršími verzemi 1.0 a 1.1, tyto verze nejsou dostatečné pro vývoj našich aplikací. Nově v poslední době se již zavádí „Microsoft .NET Framework 3.0“. Tento rámec je součástí nového operačního systému Windows Vista a zahrnuje v sobě další rozšiřující vlastnosti.

Rámec se nám stará o věci, které dříve museli vývojáři často řešit. Dnes tyto záležitosti považují za samozřejmé. .NET framework se postará o řadu nízkoúrovňových a nezáživných povinností, jakými jsou:

- správa paměti, vytváření a rušení objektů
- spouštění a zastavování vláken kódu
- bezpečnost kódu a kontrola oprávnění k prováděným operacím
- zavádění potřebných knihoven a komponent do paměti apod.

O tyto téměř neviditelné, ale velmi důležité operace se stará část zvaná „Common Language Runtime“ (CLR).



Obrázek 1: Obrázek celého .NET frameworku

Mezi další komponenty .NET frameworku patří „Base Class Library“ (BLC) je to knihovna obsahující nejčastější pomocné funkce pro práci se soubory, třídění, diagnostiku síťovou komunikaci apod. ADO.NET je knihovna pro práci s daty s možností jejich XML reprezentace. Dále jsou na obrázku dvě knihovny pro vývoj uživatelského rozhraní „Windows Forms“ pro desktopové aplikace a „ASP.NET“ pro webové uživatelské rozhraní. Pro nás bude právě tato naposled jmenovaná nejdůležitější.

Velmi důležitou částí .NET frameworku jsou podporované jazyky. .NET framework je jazykově nezávislý, pro libovolnou úlohu lze v principu použít jakýkoliv z podporovaných jazyků. Pro svůj ukázkový Informační systém používám programovací jazyk C#.

1.2 SQL Server 2005 Express Edition

V typických aplikacích zpravidla potřebujeme zobrazit data z relační databáze, ať už je to katalog zboží, seznam oblíbených písniček nebo v mém případě obsah jednotlivých článků a diskuzí. Webové stránky v ASP.NET mohou zobrazit data z libovolné databáze, ke které je k dispozici ODBC nebo OLEDB ovladač.

V mém případě jsem využíval relační databázi SQL Server 2005 Express Edition. Za pomocí tohoto SQL serveru jsem vytvářel informační systém, na prezentaci a kategorizaci článků a jednotlivých diskuzí. SQL server nám mimo jiné nabízí referenční integritu mezi tabulkami, uložené procedury, triggery, uživatelsky definované funkce, přirozenou podporu XML typu a možnost rozšiřování funkcí v .NET jazycích díky integraci .NET frameworku do databázového stroje.

1.3 Visual Studio 2005 Professional

K vlastnímu vývoji můžeme použít prostředí „Visual Studio 2005 Professional“ nebo volně šiřitelný „Visual Web Developer 2005 Express Edition“ . Oba tyto nástroje jsou plnohodnotná prostředí pro vývoj aplikací v ASP.NET. Umožňují nám mimo jiné jak grafický tak i kódový návrh webové aplikace. Obsahují všechny ASP.NET 2.0 webové kontroly, které můžeme pomocí pouhého přetažení myší umístit na návrh aplikace a dále s nimi pracovat. Měnit jejich vlastnosti upravovat vzhled a funkcionalitu.

1.4 Webový server

K provozování webových aplikací v ASP.NET potřebujeme samozřejmě webový server. Tento problém nemusíme při vývoji řešit, neboť vývojové prostředí, které můžeme používat, obsahuje vestavěný webový server „ASP.NET Development Server“. Pro vývoj je naprosto dostatečný, má určitá omezení, která znemožňují jeho použití pro provoz hotového webu. Podporuje totiž pouze lokální připojení (prohlížeč a webový server musí být na stejném počítači), není automaticky spouštěn při startu apod.

Pro lepší provoz hotového webu a ladění jsem ho umístil na „Internet Information Server“ (IIS) a posléze z důvodu prezentace a dostupnosti na freehostingový server volně dostupný z Internetu.

1.5 Shrnutí

Po obstarání tohoto SW vybavení a jeho správném nainstalování a nastavení, máme plně připravené prostředí pro vývoj webových aplikací v ASP.NET 2.0. Volba programovacího jazyka je pouze na vývojáři, stejně tak i v jakém vývojovém prostředí bude aplikace vytvářet.

Mnou demonstrovaný informační systém je vyvíjen v prostředí „Visual Studio 2005 Professional“. Tento nástroj jsem si zvolil už jen z důvodu, že je pro studenty volně dostupný a využíval jsem ho i pro práci na jiných projektech. Všechny ukázkové aplikace budou psány v jazyce C#. Funkčnost je ovšem ale stejná při použití jakéhokoliv jiného jazyka.

Základní principy ASP.NET

1.6 Objektový model webových stránek

Do příchodu ASP.NET byl velký rozdíl mezi vývojem aplikací pro desktop a pro web. ASP.NET přineslo v tomto směru velkou změnu – vývoj webových a desktopových aplikací je založen na podobných principech.

1.6.1 Rozdíl mezi tradičním vývojem pro desktop a pro web

Desktopové aplikace se poslední dobou vyvíjejí velmi snadno. K dispozici jsou moderní prostředí, jako je Visual Basic anebo Delphi. Aplikace se skládají z navazujících formulářů. Každý formulář je složen z ovládacích prvků – tlačítek, rozbalovacích seznamů, stromů, kalendářů apod. Vývojáři vytvářejí formuláře „přetahováním“ jednotlivých prvků z palety dostupných prvků. Každý nástroj jich obsahuje omezené množství – pokud nám nabídka ovládacích prvků nestačí, můžeme si sehnat další ovládací prvky a komponenty, anebo si je sami napsat.

V pozadí ovládacích prvků je vyspělý objektový model. Každý ovládací prvek je po softwarové stránce objektem. Jeho vzhled a chování je určováno vlastnostmi (výška, šířka, barva pozadí, zobrazený text apod.). Prvky též na základě akcí uživatele vyvolávají události (stisknutí tlačítka, výběr prvku ze seznamu, změna textu v editačním poli apod.). Vlastní vývoj uživatelského rozhraní pak spočívá z velké části ve vytváření obslužného kódu událostí, např. při výběru prvku z rozbalovacího seznamu se aktualizují některá editační pole, při stisknutí tlačítka se zavolá zpracování právě prováděné úlohy apod.

Jednoduchost vývoje desktopových aplikací vedla k jejich velké popularitě, neboť vývoj uživatelského rozhraní byl pohodlný a rychlý. Také z hlediska koncových uživatelů jsou desktopové aplikace velmi příjemné – nabízejí velký komfort ovládání (klávesové zkratky, drag&drop myší apod.), skvěle využívají dostupný hardware a periférie, umožňují uložení dat na lokálním počítači apod. Naopak správci sítí desktopové aplikace nemilovali – instalace, podpora a aktualizace desktopových aplikací byla jejich noční můrou. A přestože Java i .NET dnes nabízejí technologie pro bezpracnou aktualizaci a údržbu desktopových aplikací, určitá averze vůči nim stále přetrvává.

Pracnost správy desktopových aplikací vedla v posledních letech k velkému rozmachu webových aplikací, vyžadujících na straně klienta pouze jakýkoliv prohlížeč s podporou HTML. Množství webových intranetových i Internetových aplikací tehdy rostlo geometrickou řadou. Ne každý si však plně uvědomoval negativa tohoto přístupu. Webové aplikace mají svoje limity, pokud jde o komfort uživatelského rozhraní a využívání možností lokálního počítače. Například pokud chcete z webové aplikace používat nějakou periférii, musíte stejně na klienta nainstalovat nějakou komponentu (Java applet nebo ActiveX prvek), čímž do ní vnesete závislost na prostředí koncového uživatele a noční můra administrátorů je zpět

Ještě větším problémem tradičně vyvíjených webových aplikací je neefektivita a nákladnost jejich vývoje. V prohlížeči jsou jednotlivé stránky zobrazovány na základě HTML značek v textovém souboru. Pomocí HTML lze snadno zobrazit pouze jednoduché vizuální prvky, k vytvoření složitějších prvků jako jsou kalendáře, menu nebo rozbalovací stromy, je již třeba velmi pokročilých znalostí. Vlastní vývoj webových aplikací v tradičních technologiích (PHP, ASP, Perl, JSP apod.) spočívá v míchání HTML značek s kódem v programovacím jazyce či skriptu. Výsledkem je velmi špatně čitelný kód, který se velmi pracně udržuje a mění, je velmi těžké sdílet ho mezi více projekty, vytvářet v týmu apod. Chybějící objektový model je důvodem pro chybějící kvalitní vývojové prostředí. Většina „vývojových“ prostředí jsou ve skutečnosti pouze lepšími editory a ke komfortu vývojových prostředí pro desktopové aplikace mají stejně daleko jako Velorex k Mercedesu.

1.6.2 Odlišnosti ASP.NET

ASP.NET není nic jiného, než přenesení osvědčených principů desktopového vývoje do prostředí vývoje webového. Podobně jako v desktopovém vývoji, ASP.NET stránka se skládá z ovládacích prvků – tlačítek, kalendářů, stromů apod., což jsou čistokrevné objekty na straně serveru. Jejich vzhled a chování je rovněž určováno vlastnostmi (výška, šířka, barva pozadí, zobrazený text apod.). Prvky stejně tak na základě akcí uživatele vyvolávají události (stisknutí tlačítka, výběr prvku ze seznamu, změna textu v editačním poli apod.). Vývojáři rovněž vytvářejí stránky „přetahováním" jednotlivých prvků z palety dostupných prvků v komfortním vývojovém prostředí. Pokud vám nabídka dostupných ovládacích prvků nestačí, můžete si rovněž za peníze či zadarmo sehnat další ovládací prvky a komponenty, anebo si je dokonce můžete napsat sami.

Na druhou stranu na klientovi je stejně jako u webových aplikací pouze čisté HTML verze 3.2, případně dokonce čisté XHTML. Na klienta se neinstalují žádné objekty – ani Java applety ani ActiveX ani .NET framework ani nic jiného. Jedinou interaktivitou na klientovi jsou jednoduché Java skripty, bez kterých se koneckonců dnes již téměř žádný web neobejde. Webové aplikace napsané v ASP.NET tak fungují v libovolném běžně dostupném prohlížeči s podporou HTML nebo XHTML a Java skriptu.

ASP.NET tedy ideálním způsobem kombinují rychlost a jednoduchost vývoje desktopových aplikací s minimálními požadavky na klientský počítač webových aplikací.

1.7 Vnitřní objektový model ASP.NET

Objekty v ASP.NET „žijí" na serveru, nikoliv na klientovi. Objekty jsou definovány v souboru
aspx pomocí speciálních značek. Objekty, jako jsou tlačítka či kalendáře, existují v paměti serveru pouze po dobu vykonávání žádosti klienta. Po vykonání celé stránky se na klienta pošle „obraz" jednotlivých objektů v podobě HTML značek.

1.7.1 Pojem PostBack

Pokud dojde k událostem (např. stisknutí tlačítka), je prostřednictvím Java skriptu znovu zavolán server a jsou mu předány informace o proběhlé události. Na straně serveru se znovu vytvoří objektový model stránky (tlačítka, kalendáře apod.) a je zde vyvolána událost příslušného prvku. Pokud jsme napsali kód obsluhující tuto událost, je tento kód vykonán. Na závěr je obraz celé (někdy aktualizované) stránky odeslán do klientského prohlížeče.

Zároveň po odeslání stránky na klienta si můžeme všimnout, že dojde k obnově stránky na klientovi, což se projeví „bliknutím" stránky v prohlížeči. Toto je úskalí tohoto postupu – každou takovou akcí dochází k zatížení serveru a k malému zdržení na klientovi. Máme-li na serveru dostatečnou výkonovou rezervu a síťové spojení mezi serverem a prohlížečem je dostatečně rychlé, nezaznamenáme žádný pozorovatelný problém. V opačném případě se může aplikace jevit jako pomalu reagující.

1.7.2 Pojem ViewState

Pokud objekty „žijí" na serveru pouze krátkodobě, server musí při příštím požadavku zjistit, zda je statický text právě zobrazen či nikoliv. Přesněji kde se mezi jednotlivými PostBack událostmi uchovává stav stránky.

Řešení je jednoduché, server na konci zpracování získá od jednotlivých ovládacích prvků jejich stav a pošle ho zakódovaný a digitálně podepsaný ve skrytém vstupním poli prohlížeči. Tento řetězec je pak při PostBack události předán zpět na server, kde slouží k rekonstrukci stavu ovládacích prvků.

1.7.3 Ovládací prvky

Součástí ASP.NET 2.0 je několik desítek dostupných ovládacích prvků. Pro pohodlí vývojářů jsou ve Visual Studiu seskupeny do několika kategorií.

Popis prvků v jednotlivých kategoriích:

- Standard – zde najdeme nejčastěji používané ovládací prvky (tlačítka, textové vstupy, rozbalovací seznamy apod.)
- Data – druhá nejčastěji používaná skupina prvků, obsahuje prvky pro práci s daty.
- Validators – obsahuje validátory, které slouží ke kontrole zadaných hodnot na stránce.
- Navigation – zde se nachází prvky pro zobrazování hierarchických dat (mapy webu, struktury katalogu zboží, obsahu knihy apod.
- Login – setkáme se zde s prvky pracujícími s přihlašovacími účty uživatelů, přihlášení a odhlášení, založení nového účtu, změnu hesla apod.
- WebParts – ovládací prvky pro vytváření personalizovaných stránek. Tyto ovládací prvky se nejčastěji používají pro vytváření portálů, u kterých si uživatelé mohou sami do určitě míry měnit rozmístění a nastavení jednotlivých webových dílců (web parts).

V případě že vývojář potřebuje nějaký ovládací prvek, který není součástí ani jedné jmenované kategorie má dvě možnosti: napsat si ho sám anebo se podívat, zda to již někdo jiný tento prvek neudělal.

1.8 Shrnutí

V této kapitole jsem popisoval některé technologické základy, ze kterých ASP.NET vychází. Jednotlivým kontrolám se budu věnovat později, kdy budu i popisovat jak se s nimi pracuje, jak se používají a co vše je možné pomocí nich realizovat.

Membership API

1.9 Úvod do Membership API

Pracovní rámec Membership API poskytuje kompletní sadu funkcí pro správu uživatelů, kterou lze snadno a rychle integrovat:

- Lze vytvářet a odstraňovat uživatele buď programátorsky, nebo prostřednictvím webové konfigurační utility ASP.NET.
- Lze obnovovat hesla, a také automaticky odesílat e-mail sloužící k obnově hesla, pokud je k danému uživateli uložena také jeho e-mailová adresa.
- Hesla lze generovat uživatelům automaticky, pokud se uživatelé vytvářejí programátorsky na pozadí. Hesla se mohou také odesílat uživatelům automaticky (je-li k dispozici jejich e-mailová adresa).
- Lze vyhledávat uživatele v úložišti dat a získat tak nejenom seznam uživatelů, ale i podrobnosti o jednotlivých uživatelích.
- K dispozici je sada zabudovaných ovládacích prvků, připravených pro vytváření přihlašovacích stránek a registračních stránek. Dají se s nimi zobrazovat různé stavy přihlášení i vytvářet různá zobrazení pro autentizované uživatele, a pro uživatele, kteří svou totožnost ještě neprokázali.
- Je zde také vrstva abstrakce pro aplikaci, aby aplikace nebyla závislá na úložišti dat, a to prostřednictvím Membership Provider Classes. Veškerá funkcionalita pracuje zcela nezávisle na úložišti dat, přičemž používané úložiště dat lze nahradit jiným, aniž by se musela aplikace nějak modifikovat. Stačí pouze modifikovat soubor web.config, kde se nadefinuje nové úložiště dat.

1.10 Práce s Membership API

Než začneme pracovat s Membership API a s ovládacími prvky pro bezpečnost ASP.NET, musíme provést několik předběžných nastavení:

1. Nakonfigurovat v souboru web.config formulářovou autentizaci a odepřít přístup anonymním uživatelům.
2. Připravit úložiště dat pro Membership API. Například vytvořit ve SQL databázi, kterou si zvolíme, pár tabulek a uložených procedur, pokud pracujeme s SQL serverem.
3. Nakonfigurovat v souboru web.config naší aplikace atributy connection string k databázi a membership provider, se kterým chceme pracovat.
4. Vytvořit ve svém úložišti dat pro Membership API uživatele, a to buď pomocí k tomu určené konfigurační utility ASP.NET, nebo pomocí vlastní administrační stránky.
5. Vytvořit přihlašovací stránku, která bude používat zabudovaný ovládací prvek Login, nebo vytvořit přihlašovací stránku, která bude ověřovat platnost předložených dokladů a autentizovat uživatele pomocí třídy Membership.

1.10.1 Konfigurace formulářové autentizace

Membership API poskytuje infrastrukturu pro správu a autentizace uživatelů. Je usazena na vrcholu formulářové autentizace. Jako první krok musíme nakonfigurovat svou aplikaci tak, aby využívala formulářovou autentizaci. Kořenový adresář webové aplikace často uděluje přístup anonymním uživatelům, zatímco zdroje s restriktivním přístupem jsou uloženy v podadresářích s restriktivním přístupem. Tyto podadresáře mají své vlastní soubory web.config, ve kterých se odepře přístup anonymním uživatelům. Jakmile se někdo pokusí přistoupit ke zdroji, který je uložený v takto zabezpečeném adresáři, runtime ASP.NET automaticky přesměruje uživatele na přihlašovací stránku. Kořenový adresář, obsahuje schopnosti, jako jsou přihlašovací a registrační stránka.

V kořenovém adresáři webové aplikace nakonfigurujeme formulářovou autentizace tak, že do souboru web.config přidáme následující:

```
<system.web>
      <authentication mode="Forms" />
</system.web>
```

Tato konfigurace specifikuje formulářovou autentizaci a povoluje anonymní přístup ke stránkám. V zabezpečeném podadresáři pak přidáme jeden soubor web.config navíc s následujícím obsahem:

```
<configuration xmlns=http://schemas.microsoft.com/.NetConfiguration/v2.0>
      <system.web>
            <authorization>
                  <deny users="?" />
            </authorization>
      </system.web>
</configuration>
```

Tato konfigurace odepírá přístup anonymním uživatelům k zabezpečené podsložce webu. Jestliže se pokusí nepřihlášený uživatel o přístup k prostředkům umístěným v tomto adresáři, runtime ASP.NET automaticky uživatele přesměruje na veřejně dostupnou přihlašovací stránku.

1.10.2 Vytvoření úložiště dat

Když používáme Membership API, musíme si připravit nějaké úložiště dat, které bude používat náš membership provider. Jedná-li se o SQL Server, stačí vytvořit pár tabulek buď v existující, nebo nové databázi SQL Serveru. .NET Framework se dodává s nástrojem aspnet_regsql.exe, který nám automaticky vytvoří tabulky. Jestliže se rozhodneme používat vlastního poskytovatele, budeme si muset připravit a nakonfigurovat vlastní úložiště dat, které bude náš membership provider používat.

1.10.2.1 Databázové skripty pro služby ASP.NET

Nástroj aspnet_regsql.exe vykoná několik skriptů, které vytvářejí (nebo odstraňují) databáze a databázové tabulky související s Membership API. Skripty jsou opět součástí dodávky .NET Frameworku.

Přehled instalačních skriptů:

- InstallCommon.sql – vytvoří některé společné tabulky a uložené procedury, které jsou nezbytné jak pro membership API, tak pro API rolí.
- InstallMembership.sql – vytvoří tabulky, uložené procedury a triggery, které používá membership API.
- InstallRoles.sql – vytvoří tabulky a uložené procedury, které se požadují, když se s uživateli sdružují role aplikace.
- InstallPersonalization.sql – obsahuje DDL pro vytvoření kterékoliv tabulky či uložené procedury, které se požadují, když se s tzv. Web Parts vytvářejí aplikace personalizovaných portálů.
- InstallProfile.sql – vytvoří všechny nezbytné tabulky a uložené procedury pro podporu uživatelských profilů ASP.NET 2.0.
- InstallSqlState.sql – vytvoří tabulky pro trvalý stav relace do databáze TEMP SQL Serveru. To znamená, že pokaždé, když se shodí služba SQL Serveru, stav relace se ztratí.
- InstallPersistSqlState.sql – vytvoří tabulky pro trvalý stav relace do samostatné databáze ASPState. Stav relace se zachová, i když se služba SQL Serveru restartuje.

1.10.2.2 Úložiště SQL Serveru založené na souboru

SQL Server 2005 podporuje mód, který umožňuje přímo přistupovat k souboru databáze, takže se dá k databázím přistupovat přímo, prostřednictvím jejich souborů MDF, aniž bychom je museli vytvářet nebo připojovat v nějaké instanci SQL Serveru.

S touto možností můžeme zkopírovat na cílový server soubor databáze, který aplikace používá. Poskytovatel SQL Serveru pak použije připojovací řetězec, který přistupuje k databázím přímo. SQL Server připojí automaticky databázi a umožní nám s ní přímo pracovat, aniž bychom museli provádět jakékoliv dodatečné konfigurační kroky.

Databázové soubory jsou umístěny ve speciálním podadresáři App_Data naší aplikace. V případě používání ASP.NET s výchozí konfigurací se vytvoří databázový soubor automaticky. Podnět pro vytvoření souboru dá první použití jakékoliv schopnosti, kterou vyžaduje konkrétní typ funkcionality. Poskytovatel poté automaticky vytvoří databázový soubor s nezbytným obsahem. Tuto funkcionalitu poskytuje třída SqlMembershipProvider.

1.10.3 Konfigurace Connection String a Membership Provider

Při výchozí konfiguraci, a s nainstalovaným SQL Serverem 2005, nemusíme připravovat ani úložiště dat, ani konfigurovat poskytovatele Membership API, protože runtime ASP.NET použije poskytovatele SQL Serveru 2005 založeného na souboru a automaticky pro nás vytvoří soubor databáze.

Chceme-li však používat vlastní databázi SQL Serveru, nebo vlastního poskytovatele Membership API a vlastní úložiště dat, budeme muset sami nakonfigurovat poskytovatele i připojovací řetězec k databázi sloužící jako úložiště dat Membership API. Při tomto postupu budeme muset ručně modifikovat soubor web.config naší aplikace.

V případě, že využíváme úložiště SQL Serveru (nebo jiné databázové úložiště), musíme prvně nakonfigurovat připojovací řetězec. To se dělá prostřednictvím sekce <connectionStrings /> v souboru web.config naší aplikace. Pokud například chceme použít lokální databázi s názvem MyDatabase, kam jsme nainstalovali potřebné databázové tabulky, musíme připojovací řetězec nakonfigurovat následovně:

```
<connectionStrings>
      <add name="MyMembershipConnString"
            ConnectionString="data source=(local); Integrated
            Security=SSPI; Initial catalog=MyDatabase" />
</connectionStrings>
```

Poté, co máme nakonfigurovaný připojovací řetězec pro vlastní úložiště Membership API, musíme nakonfigurovat poskytovatele Membership API pro aplikaci. Zde musíme opět přidat do souboru web.config sekci <membership />, opět příklad použití:

```
<system.web>
      <authentication mode="Forms" />
      <authorization>
            <deny users="?" />
      </authorization>
      <membership defaultProvider="MyMembershipProvider">
            <providers>
                  <add name="MyMembershipProvider"
                        connectionStringName="MyMembershipConnString"
                        applicationName="MyMembership"
                        enablePasswordRetrieval="false"
                        enablePasswordReset="true"
                        requiresQuestionAndAnswer="true"
                        requiresUniqueEmial="true"
                        passwordFromat="Hashed"
                        type="System.Web.Security.SqlMembershipProvider" />
            </providers>
      </membership>
</system.web>
```

Uvnitř sekce <membership /> můžeme přidat více poskytovatelů jako dceřiné prvky sekce <providers />. Je důležité, abychom nezapomněli na atribut defaultProvider. Ten vyjadřuje, který poskytovatel Membership API se použije, pokud použitého poskytovatele nepřekryjeme ve svém kódu.

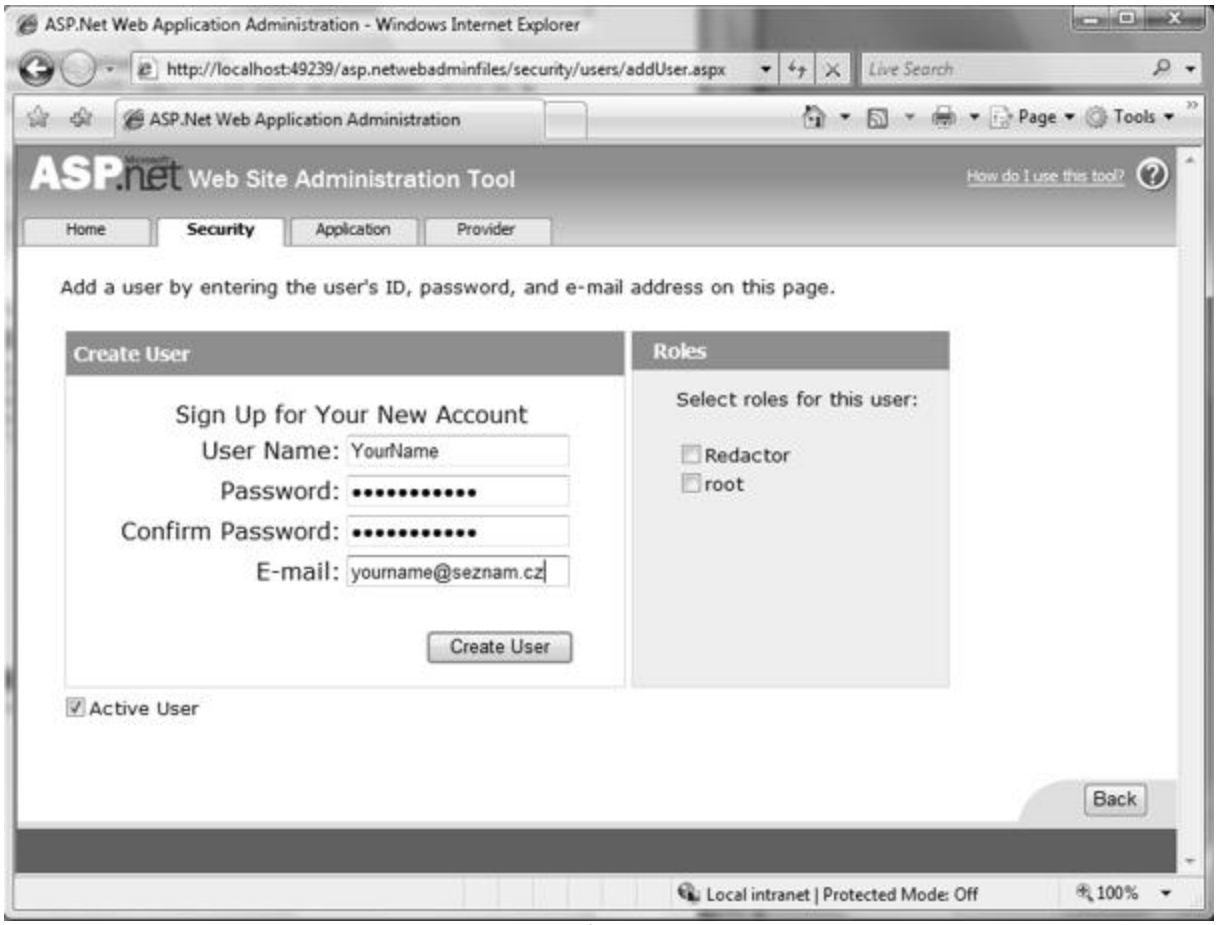
Vlastnost	**Popis**
Name	Název poskytovatele Membership API. Můžeme zvolit libovolný název. Tento název se používá při programátorském přístupu k seznamu nakonfigurovaných poskytovatelů Membership API.
ApplicationName	Název aplikace, pro kterou poskytovatel Membership API spravuje uživatele a jejich nastavení.
Description	Volitelný popis poskytovatele Membership API.
PasswordFormat	Získá nebo nastaví formát, ve kterém se budou ukládat hesla v úložišti dokladů. Platné volby jsou Clear pro ukládání hesel jako čistý text, Encrypted, když se hesla při ukládání do úložiště šifrují, a Hash, když se hesla ukládaná do úložiště Membership API hašují.
MinRequiredNonAlphanumericCharacters	Počet jiných znaků než písmeny a číslice, které musí heslo obsahovat. Je to důležitá část ověřování platnosti hesla. Umožňuje posílit požadavky na hesla, která budou muset uživatelé používat.
MinRequiredPasswordLength	Umožňuje specifikovat minimální délku hesel používaných uživateli aplikace. Další důležitá vlastnost pro posílení bezpečnosti hesla.
PasswordStrengthRegularExpression	Můžeme využít regulérních výrazů pro definování, kdy je heslo platné. Tato volba umožňuje naprosto flexibilně stanovit kritéria pro platnost hesel.
EnablePasswordReset	Membership API obsahuje funkcionalitu umožňující obnovit uživateli heslo a volitelně mu jej zaslat na zadaný email.
EnablePasswordRetrievel	V případě povolení můžeme získat heslo objektu MembershipUser tak, že zavoláme jeho metodu GetPassword. Funguje jen v případě, jestliže není heslo hašované.
MaxInvalidPasswordAttempts	Počet neplatných pokusů o ověření platnosti, než se před uživatelem aplikace uzamkne.
PasswordAttemptWindow	Zde můžeme nastavit dobu v minutách, ve které může uživatel odpovědět neplatným heslem nebo nesprávně na otázku sdruženou s heslem nejvýše tolikrát, kolik určuje stanovený maximální povolený počet neplatným pokusů. Pak se před uživatelem aplikace plně uzamkne. Jeho účet musí poté znovu administrátor aktivovat.
RequiresQuestionAndAnswer	Specifikuje, zdali se pro aplikaci požaduje otázka sdružená s heslem a odpověď na ni. Otázka se obvykle klade, když uživatel zapomněl své heslo. Správná odpověď na otázku mu dává možnost, aby získal přes email nové, automaticky vygenerované heslo.

RequiresUniqueEmail	Specifikuje, zdali musí být emailové adresy v úložišti Membership API pro jednotlivé uživatele jedinečné.

Tabulka 1: Popis některých dalších vlastností SqlMembershipProvider

1.10.4 Vytváření a autentizace uživatelů

Nové uživatele lze vytvořit v právě zřízeném úložišti dat poskytovatele Membership API tak, že z nabídek Visual Studia spustíme Web Site Administration Tool. Přejdeme na záložku Security a vybereme Create User, viz obrázek níže.



Obrázek 2: Vytváření uživatelů s Web Site Administration Tool.

Pomocí tohoto průvodce můžeme vytvořit uživatele, kteří budou mít přístup do naší webové aplikace. Při vytváření jednotlivých uživatelů, jim můžeme rovnou přidělit uživatelskou roli z nabídky. Zadávaná pole jsou volitelná, jejich nastavení se provádí v konfiguraci membership providera. Na obrázku je vidět že náš membership provider je nakonfigurován tak, že stačí zadat uživatelské jméno, heslo, potvrzení hesla a email uživatele. Kontrolní otázka sdružená s heslem je v naší konfiguraci membership provider vypnutá. Není tedy možné zasílat nově vygenerované heslo zpět na uživatelem zadaný email.

Poté co jsme přidali uživatele do úložiště Membership API, můžeme provést jejich autentizaci (ověřit jejich totožnost) pomocí membership API. Pro tento účel si můžeme vytvořit nějakou přihlašovací stránku, která se uživatele zeptá na uživatelské jméno a heslo, a pak porovná doklady, jimiž se prokázal, s informacemi z úložiště dokladů:

Příklad ověření totožnosti pomocí Membership API:

```
Protected void LoginAction_click(object sender, EventArgs e)
{
```

```
        If (Membership.ValidateUser(UsernameText.Text, PasswordText.Text))
        {
              FormsAuthenticatio.RedirectFromLoginPage(UsernameText.Text,
                    false);
        }
        else
        {
              LegendStatus.Text = "Invalid user name or password!";
        }
}
```

Nepotřebujeme přitom vědět, jakého poskytovatele aplikace právě používáme. Chceme-li použít jiného poskytovatele Membership API, stačí jednoduše změnit konfiguraci tak, aby membership API používalo jiného poskytovatele. V dalších kapitolách jsem se zaměřil na nové ovládací prvky pro bezpečnost, kde i bude vysvětlena jejich funkčnost a použití.

Ovládací prvky pro bezpečnost

1.11 Přehled nových prvků v ASP.NET 2.0 pro bezpečnost

Přihlašovací stránky se musejí vytvářet znovu a znovu. ASP.NET 2.0 se dodává s novými ovládacími prvky, které zjednodušují proces vytváření přihlašovací stránky, i s veškerou k ní se vztahující funkcionalitou. Ovládací prvky pro bezpečnost, které jsou dodávány, se spoléhají na formulářovou autentizaci a na infrastrukturu membership API.

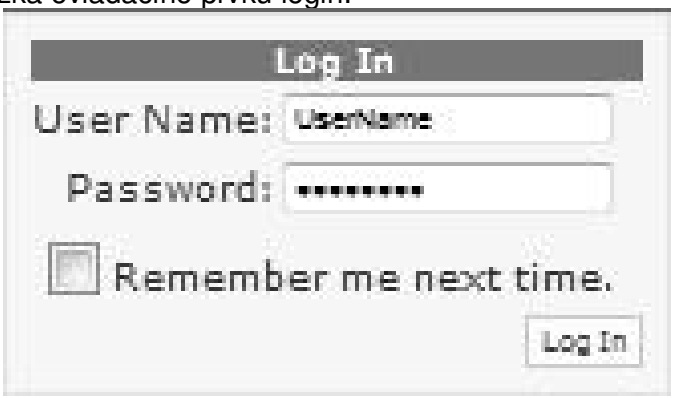
Ovládací prvek	Primární účel
Login	Ovládací prvek Login je složený ovládací prvek, který řeší nejběžnější úkol v aplikacích, které jsou založeny na formulářové autentizaci – zobrazit textová pole pro uživatelské jméno a heslo, a tlačítko, kterým se pokusí přihlásit. Kromě toho, pokud se zachycují události prostřednictvím vlastních událostních procedur, automaticky ověřují platnost uživatele vzhledem k výchozímu poskytovateli Membership API.
LoginStatus	Stav přihlášení je prostý ovládací prvek, který ověří stav autentizace aktuální relace. Jestliže uživatel ještě neprokázal svou totožnost (není autentizovaný), nabídne se mu tlačítko, které ho přesměruje na nadefinovanou přihlašovací stránku. Jinak se jednoduše zobrazí tlačítko, s jehož pomocí se bude uživatel moci případně odhlásit.
LoginView	Ovládací prvek, který umožňuje zobrazit jednu sadu ovládacích prvků pro ty uživatele, kteří už prokázali svou totožnost, a jinou sadu ovládacích prvků pro uživatele, kteří ještě svou totožnost neprokázali. Kromě toho umožňuje zobrazovat různé ovládací prvky pro uživatele, kteří mají přiděleny různé role.
PasswordRecovery	Umožňuje uživateli znovu získat heslo, pokud během registrace poskytl nějakou emailovou adresu. Prvek pořádá uživatele o jeho uživatelské jméno, a pak automaticky zobrazí uživatelské rozhraní, ve kterém uvidí otázku spojenou s heslem a pole pro vložení požadované odpovědi. Odpoví-li uživatel správně, bude uživateli pomocí membership API zasláno nové heslo.
ChangePassword	Složený ovládací prvek, který uživatele požádá o staré heslo, a který mu umožní zadat nové heslo a potvrdit jej.
CreateUserWizard	Obsahuje kompletního průvodce, který krok za krokem vede uživatele (nebo administrátora) procesem vytvoření nového uživatele.

Tabulka 2: Přehled ovládacích prvků ASP.NET 2.0 pro bezpečnost

Tyto ovládací prvky můžeme používat společně s libovolnými jinými ovládacími prvky. Například ovládací prvek Login můžeme použít buď na své hlavní stránce, nebo na nějaké samostatné přihlašovací stránce. Všechny ovládací prvky fungují obdobně – nezachytáváme-li žádné vlastní události, pracují všechny ovládací prvky standardně s membership API. Jakmile začneme zachytávat události, jimiž jsou tyto ovládací prvky vybaveny, převezmeme tím zodpovědnost za řádné dokončení celého úkolu. Například ovládací prvek Login podporuje událost Authenticate. Jestliže ji nezachytáváme, prvek automaticky použije membership API. V opačném případě máme ověření platnosti dokladů předložených uživatelem na starosti my.

1.12 Ovládací prvek Login

Ovládací prvek Login poskytuje hotové uživatelské rozhraní, které se uživatele ptá na jeho uživatelské jméno a heslo, a dále nabízí tlačítko, pomocí kterého se uživatel fakticky pokusí přihlásit. Ukázka ovládacího prvku login:



Obrázek 3: Ovládací prvek Login

Ovládací prvek Login je jednoduše složený ovládací prvek ASP.NET. Je plně rozšiřitelný, takže umožňuje překrýt jakékoliv styly rozvržení a vlastnosti. Také zachytává události, které ovládací prvek generuje, aby jsme překryli jeho výchozí chování. Ponecháme-li ovládací prvek Login takový, jaký je, a nebudeme zachytávat žádné jeho události, automaticky použije toho poskytovatele Membership API, který byl nakonfigurován pro danou aplikaci. Ukázka nejjednodušší formy ovládacího prvku Login na stránce:

```
<form id="form1" runat="server">
      <asp:Login ID="Login1" runat="server">
      </asp:Login>
</form>
```

Vzhled ovládacího prvku lze přizpůsobit několika vlastnostmi. Můžeme například používat různá nastavení pro styly, které ovládací prvek Login podporuje, např. následující ukázka:

```
<form id="form1" runat="server">
      <asp:Login ID="Login1" runat="server">
            BackColor="aliceblue" BorderColor="Black" BorderStyle="double">
            <LoginButtonStyle BackColor="darkblue" ForeColor="White" />
            <TextBoxStyle BackColor="LightCyan"  ForeColor="Black"
                  Font-Bold="true" />
            <TitleTextStyle Font-Italic="true" Font-Bold="true"
                  Font-Name="Verdana" />
      </asp:Login>
</form>
```

Vzhled ovládacího prvku Login můžeme také přizpůsobovat pomocí tříd CSS. Každá vlastnost stylu, kterou podporuje ovládací prvek Login, obsahuje vlastnost CssClass. Podobně jako je tomu u všech ostatních ovládacích prvků ASP.NET, umožňuje i tato vlastnost nastavit pro ovládací prvek Login název nějaké třídy CSS, kterou jsme přidali do svého webu předtím.

Styl	**Popis**
CheckBoxStyle	Definuje vlastnosti stylu pro zaškrtávací políčko Remember Me.
FailureStyle	Definuje styl textu, který se zobrazí, jestliže přihlášení nebylo úspěšné.
HyperlinkStyle	Ovládací prvek Login umožňuje definovat několik typů hypertextových odkazů, například odkaz na nějakou registrační stránku. Styl definuje vzhled hypertextových odkazů.

InstructionTextStyle	Ovládací prvek Login umožňuje specifikovat text nápovědy, který se zobrazí přímo uvnitř ovládacího prvku Login. Styl definuje vzhled textu.
LabelStyle	Definuje styl popisků User Name a Password
LoginButtonStyle	Definuje styl pro přihlašovací tlačítko.
TextBox Style	Definuje styl textových polí pro User Name a Password.
TitleTextStyle	Definuje styl pro text titulku ovládacího prvku Login.
ValidatorTextStyle	Definuje styl validačních ovládacích prvků, které ověřují platnost User Name a Password.

Tabulka 3: Styly, které podporuje ovládací prvek Login

Výše uvedené styly nejsou jedinými možnostmi, jak se dá přizpůsobovat ovládací prvek Login. Prostřednictvím několika vlastností je přizpůsobitelný jakýkoliv obsah, který se zobrazuje uvnitř ovládacího prvku. Můžeme tak například specifikovat text, který se má zobrazit na přihlašovacím tlačítku, nebo můžeme místo přihlašovacího tlačítka (což je výchozí volba) zobrazit přihlašovací odkaz. Dále můžeme do ovládacího prvku Login přidat několik hypertextových odkazů, jako je třeba hypertextový odkaz na stránku s textem nápovědy, nebo hypertextový odkaz na registrační stránku. Samozřejmě obě stránky musejí být přístupné pro anonymní uživatele, protože nápovědu je potřeba poskytnout anonymním uživatelům. Ukázka upraveného ovládacího prvku, zobrazeného dříve, s přidanými dodatečnými odkazy:

```
<form id="form1" runat="server">
      <asp:Login ID="Login1" runat="server">
            BackColor="aliceblue" BorderColor="Black" BorderStyle="double"
            CreateUserText="Register"
            CreateUserUrl="Register.aspx" HelpPateText="Additional Help"
            HelpPageUrl="HelpMe.aspx"
            InstructionText="Please enter your user name and password for
            loggining into the system.">
            <LoginButtonStyle BackColor="darkblue" ForeColor="White" />
            <TextBoxStyle BackColor="LightCyan"  ForeColor="Black"
                  Font-Bold="true" />
            <TitleTextStyle Font-Italic="true" Font-Bold="true"
                  Font-Name="Verdana" />
      </asp:Login>
</form>
```

Tento kód zobrazí dva dodatečné odkazy – jeden na stránku s nápovědou, druhý na registrační stránku. Pod záhlaví ovládacího prvku Login se dále přidá krátký instruktivní text. Na tyto vlastnosti se aplikují styly, které jsem popsal výše.

Vlastnost	Popis
TitleText	Text, který se zobrazí jako záhlaví ovládacího prvku.
InstructionText	Tato vlastnost obsahuje text, který se zobrazí pod záhlavím ovládacího prvku.
FailureText	Text, který zobrazí ovládací prvek Login, když přihlášení nebylo úspěšné.
UsernameLabelText	Text, který se zobrazí jako popisek před textovým polem pro user name.
PasswordlabelText	Text, který se zobrazí jako popisek před textovým polem pro password.
UserName	Počáteční hodnota, která zaplní textové pole pro user name.
UsernameRequiredErrorMessage	Chybová zpráva, která se zobrazí, když uživatel nezadá user name.

PasswordRequiredErrorMessage	Chybová zpráva, která se zobrazí, když uživatel nezadá password.
LoginButtonText	Text na přihlašovacím tlačítku.
LoginButtonType	Přihlašovací tlačítko se dá zobrazit jako odkaz, tlačítko, nebo obrázek, vzhled určuje hodnota této vlastnosti. Podporované hodnoty jsou Link, Button a Image.
LoginButtonImageUrl	Zobrazujeme-li přihlašovací tlačítko jako obrázek, musíme pro tlačítko dodat URL obrázku.
DestinationPageUrl	Jestliže bylo přihlášení úspěšné, ovládací prvek Login přesměruje uživatele na tuto stránku. Standardně je tato vlastnost prázdná, a v takovém případě se pro přesměrování automaticky použije infrastruktura formulářové autentizace – buď na původně požadovanou stránku, nebo na defaultUrl, která je pro formulářovou autentizaci nastavena v souboru web.config.
DisplayRememberMe	Umožňuje zobrazit nebo skrýt zaškrtávací políčko Remember Me. Výchozí hodnota vlastnosti je true.
FailureAction	Definuje akci, kterou provede ovládací prvek po neúspěšném pokusu o přihlášení. Dvě platné volby jsou Refresh a RedirectToLoginPage. První volba jenom aktualizuje aktuální stránku, druhá provede přesměrování na nakonfigurovanou přihlašovací stránku. Druhá volba je samozřejmě prospěšná tehdy, používáme-li ovládací prvek kdekoliv jinde než na přihlašovací stránce.
RememberMeSet	Definuje výchozí hodnotu zaškrtávacího políčka Remember Me. Výchozí hodnota je false, což znamená, že políčko standardně není zaškrtnuté.
VisibleWhenLoggedIn	Je-li nastavena na false, ovládací prvek se automaticky skryje, je-li uživatel už přihlášen. Je-li nastavena na true (výchozí), ovládací prvek Login se zobrazí i tehdy, když už je uživatel přihlášený.
CreateUserUrl	Definuje hypertextový odkaz na nějakou stránku webu, která umožní vytvořit (zaregistrovat) nového uživatele. Vlastnost se typicky využívá tehdy, když se uživatelům umožňuje přístup k nějaké registrační stránce. Stránka obvykle zobrazí ovládací prvek CreateUserWizard.
CreateUserText	Definuje text, která se zobrazí v hypertextovém odkazu CreateUserUrl.
CreateUserIconUrl	Definuje URL obrázku, který se zobrazí společně s textem hypertextového odkazu CreateUserUrl.
HelpPageUrl	URL pro přesměrování uživatele na stránku s nápovědou.
HelpPageText	Text, který se zobrazí v hypertextovém odkazu nakonfigurovaném vlastností HelpPageUrl.
HelpPageIconUrl	URL ikony, která se zobrazí společně s textem hypertextového odkazu HelpPageUrl.
PasswordRecoveruUrl	URL pro přesměrování uživatele na stránku pro obnovu hesla. Tato stránka se používá tehdy, když uživatel zapomněl své heslo. Na stránce se typicky zobrazí ovládací prvek PasswordRecovery.
PasswordRecoveryText	Text, který se zobrazí v hypertextovém odkazu nakonfigurovaném v PasswordRecoveryUrl.
PasswordRecoveryIconUrl	Ikona, která se zobrazí společně s textem pro PasswordRecoveryUrl.

Tabulka 4: Relevantní vlastnosti pro přizpůsobování ovládacího prvku Login

1.12.1 Šablony a ovládací prvek Login

Prostřednictvím vlastností uvedených výše se dá nakonfigurovat v ovládacím prvku téměř vše. Nedají se však nadefinovat žádné validační výrazy pro ověřování platnosti vstupních dat. Ověřování můžeme provádět na straně serveru uvnitř událostních procedur, které nabízí ovládací prvek Login. Obvykle (pokud chceme přidat k ovládacímu prvku Login jakékoliv další ovládací prvky) se to však nedá dělat prostřednictvím vlastností.

Ovládací prvek proto podporuje šablony právě tak, jako jiné ovládací prvky, například GridView. S šablonami můžeme obsah ovládacího prvku Login přizpůsobovat bez jakýchkoliv omezení. Do ovládacího prvku Login můžeme přidat jakékoliv ovládací prvky. Vlastní šablonu pro ovládací prvek Login vytvoříme prostřednictvím značky LayoutTemplate:

```
<asp:Login ID="Login1" runat="server" BackColor="aliceblue"
      BorderColor="Black" BorderStyle="double">
      <LayoutTemplate>
            <h4>Log-In to the System</h4>
            <table>
                  <tr>
                        <td>User Name:</td>
                        <td>
                              <asp:TextBox ID="UserName" runat="server" />
                              <asp:RequiredFieldValidator
                                    ID="UserNameRequired"
                                    runat="server"
                                    ControlToValidate="UserName"
                                    ErrorMessage="*" />
                              <asp:RegularExpressionValidator
                                    ID="UserNameValidator"
                                    runat="server"
                                    ControlToValidate="UserName"
                                    ValidationExpression=" [\w| ]* "
                                    ErrorMessage="Invalid User Name" />
                        </td>
                  </tr>
                  <tr>
                        <td>Password:</td>
                        <td>
                              <asp:TextBox ID="Password" runat="server"
                                    TextMode="Password" />
                              <asp:RequiredFieldValidator
                                    ID="PasswordRequired"
                                    runat="server"
                                    ControlToValidate="Password"
                                    ErrorMessage="*" />
                              <asp:RegularExpressionValidator
                                    ID="PasswordValidator"
                                    runat="server"
                                    ControlToValidate="Password"
                                    ValidationExpression="[\w| ]*"
                                    ErrorMessage="Invalid Password" />
                        </td>
                  </tr>
            </talbe>
            <asp:CheckBox ID="RememberMe" runat="server"
                  Text="Remember Me" />
            <asp:Literal ID="FailureText" runat="server" /><br />
            <asp:Button ID="Login" runat="server" CommandName="Login"
                  Text="Login" />
```

```
      </LayoutTemplate>
</asp:Login>
```

Použijeme-li správné ovládací prvky a patřičné hodnoty ID pro ovládací prvky, které jsme zařadili, nemusíme psát žádný kód pro zpracování událostí. Kód pracuje jako obvykle, s jedinou výjimkou je to, že definuje sadu ovládacích prvků a jejich rozvržení. Ovládací prvek Login ve skutečnosti požaduje přinejmenším dvě textová pole s ID UserName, resp. Password. Jestliže chybějí (nebo nemají patřičné hodnoty ID), vygeneruje ovládací prvek výjimku. Všechny ostatní ovládací prvky jsou volitelné, ale pokud specifikujeme odpovídající hodnoty ID (například Login pro přihlašovací tlačítko), bude ovládací prvek Login automaticky zpracovávat jejich události a chovat se tak, jako kdybychom použili předdefinované rozvržení ovládacího prvku.

ID ovládacího prvku	Typ ovládacího prvku	Povinný
UserName	System.Web.UI.WebControls.TextBox	Ano
Password	System.Web.UI.WebControls.TextBox	Ano
RememberMe	System.Web.UI.WebControls.CheckBox	Ne
FailureText	System.Web.UI.WebControls.Literal	Ne
Login	Jakýkoliv ovládací prvek, který podporuje probublávání událostí a CommandName	Ne

Tabulka 5: Speciální ovládací prvky pro šablonu ovládacího prvku Login

1.12.2 Programování ovládacího prvku Login

Ovládací prvek Login podporuje několik událostí a vlastností, jimiž můžeme přizpůsobovat jeho chování. To nám dává nad přizpůsobováním ovládacího prvku Login úplnou kontrolu (spolu s ostatními přizpůsobovacími možnostmi, jako jsou šablony a vlastnosti pro vlastní styly). Ovládací prvek Login podporuje uvedené události:

Událost	Popis
LoggingIn	Vyvolá se před tím, než ovládací prvek provede autentizaci uživatele.
LoggedIn	Vyvolá se poté, co ovládací prvek provedl autentizaci uživatele.
LoginError	Vyvolá se, když se z nějakého důvodu nepodařilo přihlášení uživatele (například špatně zadané heslo nebo uživatelské jméno).
Authenticate	Vyvolá se, aby se provedla autentizace uživatele. Zachytáváme-li tuto událost, musíme autentizaci uživatele provést sami – ovládací prvek Login se bude plně spoléhat na náš autentizační kód.

Tabulka 6: Události ovládacího prvku Login

První tři události můžeme zachytit, chceme-li vykonávat nějaké akce předtím, než se provede autentizace uživatele, nebo poté, co se autentizace provede, resp. tehdy, když během procesu autentizace dojde k nějaké chybě. Například pomocí události LoginError můžeme uživatele po stanoveném počtu neúspěšných pokusů o přihlášení automaticky přesměrovat na stránku pro obnovu hesla.

```
protected void Page_Load(object sender, EventArgs e)
{
      If (!IsPostBack)
            ViewState[„LoginErrors“] =  0;
}
protected void LoginCtrl_LoginError(object sender, EventArgs e)
{
      //zvýší čítač neplatných přihlášení
      Int ErrorCount = (int) ViewState["LoginErrors"] +1;
      ViewState["LoginErrors"] = ErrorCount;
```

```
      //nyní se ověří počet chyb
      If ((ErrorCount > 3) &&
            (LoginCtrl.PasswordRecoveryUrl != string.Empty))
            Response.Redirect(LoginCtrl.PasswordRecoveryUrl);
}
```

1.13 Ovládací prvek LoginStatus

Ovládací prvek LoginStatus je jednoduchý ovládací prvek, který zobrazí buď odkaz k přihlášení, pokud ještě nebyla provedena autentizace daného uživatele, nebo odkaz k odhlášení, jedná-li se o uživatele, jehož autentizace už proběhla. Odkaz k přihlášení automaticky uživatele přesměruje na nakonfigurovanou přihlašovací stránku, odkaz k odhlášení automaticky zavolá metodu FormsAuthentication.SignOut, která uživatele odhlásí.

```
<asp:LoginStatus ID="LoginStatus1" runat="server" LoginText="Sign In"
      LogoutText="Sign Out"
      LogoutPageUrl="~/Default.aspx"
      LogoutAction="Redirect" />
```

Ovládací prvek LoginStatus nabízí několik vlastností pro přizpůsobení textu odkazů a pro nastavení URL, kam se má uživatel přesměrovat, když klikne na příslušném odkazu.

Vlastnost	Popis
LoginText	Text, který se zobrazí , není-li uživatel přihlášený.
LoginImageUrl	URL obrázku, který se zobrazí jako ikona odkazu k přihlášení.
LogoutText	Text, který se zobrazí, jestliže už byla autentizace uživatele provedena.
LogoutImageUrl	URL obrázku, který se zobrazí jako ikona odkazu k odhlášení.
LogoutAction	Specifikuje akci, kterou ovládací prvek vykoná, když uživatel klikne na odkazu k odhlášení. Platné volby jsou Refresh, Redirect a RedirectToLoginPage. První volba jednoduše aktualizuje aktuální stránku, druhá provede přesměrování na stránku nakonfigurovanou v LogoutPageUrl, poslední volba provede přesměrování na přihlašovací stránku.
LogoutPageUrl	Stránka, na kterou bude uživatel přesměrován, pokud klikne na odkaz k odhlášení, a pokud je vlastnost LogoutAction nastavena na hodnotu Redirect.

Tabulka 7: Vlastnosti pro přizpůsobování ovládacího prvku LoginStatus

1.14 Ovládací prvek LoginView

Tento jednoduchý, ale neobyčejně mocný, ovládací prvek umožňuje zobrazit nějakou sadu ovládacích prvků pro anonymní uživatele, a odlišnou sadu prvků pro uživatele, kteří už úspěšně prošli procesem autentizace. Kromě toho umožňuje zobrazovat různý obsah na základě toho, jakou roli má aktuálně přihlášený uživatel přiřazenou.

Ovládací prvek LoginView je svým charakterem šablona. Poskytuje různé druhy šablon, jednu pro anonymní uživatele, druhou pro autentizované uživatele, a další pro podporu šablon založených na rolích. Uvnitř těchto šablon jednoduše přidáváme ovládací prvky, které se mají v odpovídajících situacích zobrazit.

```
<asp:LoginView ID="LoginViewCtrl" runat="server">
      <AnonymousTemplate>
            <h2> You are anonymous </h2>
      </AnonymousTemplate>
      <LoggedInTemplate>
            <h2> You are logged in </h2>
```

```
      </LoggedInTemplate>
      <RoleGroups>
            <asp:RoleGroup Roles="Administrator">
                  <ContentTemplate>
                        <h2> You are logged in </h2>
                        User is in role „Administrator“
                  </ContentTemplate>
            </asp:RoleGroup>
      </RoleGroups>
</asp:LoginView>
```

Ovládací prvek zobrazí pro anonymního uživatele, registrovaného uživatele a uživatele s přiřazenou administrátorskou rolí pokaždé jiný informativní text. Prvek LoginView kromě toho podporuje dvě události, které můžeme zachytávat, chceme-li patřičně inicializovat obsah ovládacích prvků různých šablon ještě předtím, než se zobrazí:

- ViewChanging – vyvolá se předtím, než ovládací prvek zobrazí obsah definovaný jiné šabloně.
- ViewChanged – vyvolá se poté, co ovládací prvek změnil zobrazený obsah z jedné šablony na jiný.

1.15 Ovládací prvek ChangePassword

Ovládací prvek se dá využít jako standardní ovládací prvek pro řešení úkolu, jak umožnit uživateli, aby si mohl změnit své heslo. Ovládací prvek jednoduše požádá uživatele o jeho uživatelské jméno a staré heslo. Pak požádá, aby uživatel zadal nové heslo a potvrdil ho. Je-li uživatel už přihlášen, ovládací prvek automaticky skryje textové pole pro uživatelské jméno a použije jméno autentizovaného uživatele. Na zabezpečené stránce můžeme tento ovládací prvek použít takto:

```
<asp:ChangePassword ID="ChangePwdCtrl" runat="server" BorderStyle="groove"
BackColor="aliceblue">
      <TitleTextStyle Font-Bold="true" Font-Underline="true"
      Font-Names="Verdana" ForeColor="blue" />
</asp:ChangePassword>
```

Jako u všech ovládacích prvků, i tento lze přizpůsobovat prostřednictvím vlastností a stylů, a také přes šablonu. Tentokrát se ale při přizpůsobování ovládacího prvku požadují dvě šablony:

- ChangePasswordTemplate – zobrazí pole pro zadání starého uživatelského jména a hesla, pole pro zadání nového hesla a pole pro potvrzení nového hesla.
- SuccessTemplate – zobrazí zprávu o úspěšné změně hesla.

ChangePasswordTemplate požaduje, abychom přidali určité speciální ovládací prvky se speciálními ID a hodnotami vlastnosti CommandName. Hodnoty ID a CommandName ovládacích prvků jsou uvedeny v následujícím příkladu:

```
<asp:ChangePassword ID="ChangePwdCtrl" runat="server">
      <ChangePasswordTemplate>
            Old Password:
            <asp:TextBox ID="CurrentPassword" runat="server"
            TextMode="Password" /><br />
            New Password:
            <asp:TextBox ID="NewPassword" runat="server"
            TextMode="Password" /><br />
            Confirmation:
            <asp:TextBox ID="ConfirmNewPassword" runat="server"
            TextMode="Password" /><br />
            <asp:Button ID="ChangePasswordPushButton"
```

```
                  CommandName="ChangePassword" runat="server"
                  Text="Change Password" />
            <asp:Button ID="CancelPushButton" CommandName="Cancel"
                  runat="server" Text="Cancel" />
            <asp:Literal ID="FailureText" runat="server"
                  EnableViewState="false" />
      </ChangePasswordTemplate>
      <SuccessTemplate>
            Your password has been changed!
            <asp:Button ID="ContinuePushButton"
                  CommandName="Continue" runat="server" Text="Continue" />
      </SuccessTemplate>
</asp:ChangePassword>
```

Ovládací prvky pro všechna textová pole jsou v ChangePasswordTemplate povinná, ostatní ovládací prvky jsou volitelné. Vybereme-li patřičné vlastnosti ID, a také vlastnosti CommandName tlačítek, nebudeme muset psát žádný dodatečný kód.

1.16 Ovládací prvek CreateUserWizard

Ovládací prvek umožňuje za pár minut vytvořit registrační stránky. Jedná se o ovládací prvek charakteru průvodce, který má standardně dva kroky. V prvním kroku se ptá uživatele na všeobecné informace, ve druhém zobrazuje potvrzující zprávu. Samozřejmě – protože CreateUserWizard dědí z bázového ovládacího prvku Wizard, můžeme přidat do průvodce tolik kroků, kolik potřebujeme. Ale i když jednoduše přidáme na naši stránku ovládací prvek CreateUserWizard tak, jak je připravený, máme vytvořenou plně funkční registrační stránku.

```
<asp:CreateUserWizard ID="RegisterUser" runat="server" BorderStyle="ridge"
      Backcolor="aquamarine">
      <TitleTextStyle Font-Bold="true" Font-Names="Verdana" />
      <WizardSteps>
            <asp:CreateUserWizardStep runat="server">
            </ asp:CreateUserWizardStep>
            <asp:CompleteWizardStep runat="server">
            </ asp:CompleteWizardStep>
      </WizardSteps>
</asp:CreateUserWizard>
```

Výchozí vzhled ovládacího prvku lze opět přizpůsobit prostřednictvím vlastností a stylů. Ovládací prvek sice nabízí spoustu stylů, ale v zásadě je jejich význam obdobný významu stylů, které jsou použity u předchozích ovládacích prvků.

Použijemeli ovládací prvek CreateUserWizard tak, jak jsem uvedl výše, nemusíme nastavovat nic speciálního. Ovládací prvek automaticky použije pro vytvoření uživatele nakonfigurovaného membership provider a bude obsahovat dva kroky: výchozí CreateUserWizardStep, který vytvoří ovládací prvky potřebné k tomu, aby se shromáždily všechny nezbytné informace, a CompleteWizardStep, který zobrazí potvrzující zprávu. Oba kroky lze přizpůsobovat prostřednictvím stylů nebo šablon. I když oba kroky můžeme přizpůsobovat, nemůžeme je odstranit. Použijeme-li šablony, bude na nás, abychom vytvořili nezbytné ovládací prvky.

ID	Typ	Povinný	Poznámky
UserName	System.Web.UI.WebControls.TextBox	Ano	Vždy se požaduje
Password	System.Web.UI.WebControls.TextBox	Ano	Vždy se požaduje
Confirm-Password	System.Web.UI.WebControls.TextBox	Ano	Vždy se požaduje

Email	System.Web.UI.WebControls.TextBox	Ne	Požaduje se pouze tehdy, je-li vlastnost RequireEmail ovládacího prvku nastavena na true
Question	System.Web.UI.WebControls.TextBox	Ne	Požaduje se pouze tehdy, jestliže membership provider požaduje otázku sdruženou s heslem.
Answer	System.Web.UI.WebControls.TextBox	Ne	Požaduje se pouze tehdy, jestliže membership provider požaduje otázku sdruženou s heslem.
Continue-Button	Jakýkoliv ovládací prvek, který podporuje probublávání událostí.	Ne	Nepožaduje se nikdy, ale pokud jej zařadíme, musíme nastavit CommandName na Continue

Tabulka 8: Požadované a volitelné ovládací prvky

Jakmile začneme do průvodce přidávat dodatečné kroky, budeme muset zachytávat události, a uvnitř event handler provádět určité akce. Pokud například nashromáždíme s průvodcem od uživatele nějaké dodatečné informace, budeme je muset někam uložit, a pro tento účel vykonat nad naším úložištěm dat potřebné další operace. Události specifikované pro ovládací prvek CreateUserWizard jsou vypsány v následující tabulce. Ovládací prvek také dědí všechny události, které jsou součástí ovládacího prvku Wizard.

Událost	Popis
ContinueButtonClick	Vyvolá se, když uživatel klikne v posledním kroku průvodce na tlačítko Continue.
CreatingUser	Vyvolá průvodce předtím, než vytvoří prostřednictvím membership API nového uživatele.
CreatedUser	Tuto událost ovládací prvek vyvolá poté, co byl uživatel úspěšně vytvořen.
CreateUserError	Vyvolá se, jestliže se nepodařilo úspěšně vytvořit uživatele.
SendingMail	Ovládací prvek může vytvořenému uživateli odeslat e-mail, pokud je nakonfigurovaný nějaký poštovní server. Událost se vyvolá předtím, než se odešle e-mail, takže máme možnost modifikovat obsah e-mailové zprávy.
SendMailError	Jestliže ovládací prvek nebyl z nějakého důvodu schopen e-mail odeslat, například proto, že poštovní server nebyl dostupný, vyvolá tuto událost.

Tabulka 9: Události ovládacího prvku CreateUserWizard

Samozřejmě můžeme přidat krok průvodce, který požádá uživatele o dodatečné informace, jako jsou například křestní jméno a příjmení, a automaticky uložit informace do vlastní databázové tabulky. Tyto informace by se dali uložit do profilu, ale když uživatel prochází průvodcem, není ještě autentizovaný, a proto nemůžeme tyto informace ukládat do profilu hned, protože profil je dostupný pouze pro autentizované uživatele. Proto buď musíme informace uložit od nějaké naší vlastní databázové tabulky, nebo dát uživateli možnost, aby mohl po registračním procesu editovat profil.

Dále událost CreatedUser se vyvolá okamžitě poté, co se úspěšně dokončil CreateUserWizardStep. Chceme-li tedy ukládat dodatečné informace v rámci této události,

musíme informace nashromáždit v předchozích krocích. Pro tento účel je postačující umístit další kroky průvodce před značku <asp:CreateUserWizardStep>. Ve všech ostatních případech budeme muset informace ukládat v rámci některé jiné události (například FinishButtonClick). Ale protože si nemůžeme být jisti, že uživatel opravdu projde celým průvodcem a klikne na tlačítko Finish, je rozumné nashromáždit všechny dodatečné informace před krokem CreateUserWizardStep, a uložit pak veškeré dodatečné informace prostřednictvím ovladače události CreatedUser.

1.17 Shrnutí a srovnání s ASP.NET 1.x

Ovládací prvek CreateUserWizard je silný ovládací prvek založený na membership API, který lze přizpůsobovat právě tak, jako jiné prvky dodávané s ASP.NET 2.0. Ovládací prvky, které mají charakter šablony, poskytují plnou flexibilitu a umožňují mít nad ovládacími prvky úplnou kontrolu, přesto ovládací prvky pořád dělají spoustu různé práce za nás – zejména co se týče komunikace s Membership. A pokud chceme sami provádět další akce, můžeme zachytávat několik událostí ovládacích prvků.

V nižších verzích ASP.NET 1.x nemáme k dispozici žádné takovéto kontroly, jako jsou například Login, LoginView, CreateUserWizard. Programátor musí řešit všechny tyto vstupní formuláře a funkcionalitu sám, nebo může použít ovládací prvky třetích stran. To znamená, že kdybychom chtěli třeba navrhnout stránku pro přihlášení do systému, museli bychom navrhnout jak kódovou část stránky, která by nám zajišťovala ověření uživatele, tak i formulář stránky, kam by uživatel vyplnil své přihlašovací údaje.

V nové verzi ASP.NET 2.0 postačí pouze umístit kontrolu Login na danou stránku, tím získáme jak formulář, tak i na pozadí předdefinované chování, pro přihlašování a ověřování uživatele.

Použití třídy Membership

Třída Membership je programovací rozhraní pro Membership API. Používají ji všechny ovládací prvky a celá infrastruktura Membership API. Skládá se ze dvou částí. První, která má pár vlastností a metod, se jmenuje Membership, druhá třída je MembershipUser a zapouzdřuje vlastnosti jednoho uživatele. Metody třídy Membership provádějí následující fundamentální operace:

- Vytvářejí nové uživatele.
- Odstraňují existující uživatele.
- Aktualizují existující uživatele.
- Získávají seznam uživatelů.
- Získávají podrobnosti o jednom uživateli.

Mnohé metody třídy Membership přijímají jako parametr instanci třídy MebmershipUser, nebo ji vracejí, nebo přijímají či vracejí dokonce celou kolekci instancí MembershipUser. Když například načteme nějakého uživatele metodou Membership.GetUser, nastavíme vlastnosti této instance, a pak ji předáme do metody UpdateUser třídy Membership. Tím můžeme jednoduše aktualizovat vlastnosti daného uživatele. Třída Membership i třída MembershipUser poskytují nezbytnou vrstvu abstrakce mezi skutečným poskytovatelem a naší aplikací. Vše, co děláme s třídou Membership, závisí na našem poskytovateli. To znamená, že pokud vyměníme poskytovatele Membership API, nebude to mít žádný vliv na naši aplikaci, pokud je implementace poskytovatele Membership API kompletní, a pokud podporuje všechny schopnosti propagované základní třídou MebershipProvider.

Všechny třídy, které se používají v Membership API, jsou definovány ve jmenném prostoru System.Web.Security. Třída Membership je jednoduše pouze jednou ze tříd s mnoha statickými metodami a vlastnostmi.

1.18 Získání uživatelů z úložiště

První ukázkový příklad nám ukazuje jak načíst seznam uživatelů z úložiště dat a spojit ho například s prvkem GridView. Uvedený kód je zobrazení funkční části jednoduché stránky, na které je prvek GridView pomocí kterého budeme vypisovat všechny uživatele.

```
public partial class _Default : System.Web.UI.Page
{
      //Definování proměnné pro kolekci uživatelů
      MembershipUserCollection _MyUsers;
      protected void Page_Load(object sender, EventArgs e)
      {
            //Naplnění proměnné z úložiště dat
            _MyUsers = Membership.GetAllUsers();
            //Naplnění ovládacího prvku GridView kolekcí uživatelů
            UsersGridView.DataSource = _MyUsers;
            If (!IsPostBack)
                  UsersGridView.DataBind();
      }
}
```

Třída Membership má metodu GetAllUsers, která vrací instanci typu MembershipUserCollection. S touto kolekcí se pracuje stejným způsobem, jako s jakoukoliv jinou kolekcí. Každá položka kolekce obsahuje všechny vlastnosti jednoho uživatele.

Většina metod třídy Membership má parametr UserName. Za pomocí tohoto parametru si dále ukážeme, jak získat informace o konkrétním uživateli z úložiště dat. Opět si vytvoříme nějakou jednoduchou stránku, která bude obsahovat prvky Label, TextBox a

CheckBox, které nám umožní zobrazit jednotlivé informace. Uvedený kód je opět funkční část stránky, která nám sváže jednotlivé položky informací s příslušnými prvky na naší stránce.

```
public partial class _Defauls : System.Web.UI.Page
{
      string userName = "HledanyUzivatel";
      //Definování proměnné pro vybraného uživatele
      MembershipUser _CurrentUser;
      //Naplnění proměnné z úložiště dat
      _CurrentUser = Membership.GetUser(userName);
      //Svázání dat s jednotlivými ovládacími prvky stránky
      UsernameLabel.Text = _CurrentUser.UserName;
      PwdQuestionLabel.Text = _CurrentUser.PasswordQuestion;
      LastLoginLabel.Text = _CurrentUser.LastLoginDate.ToShortDateString();
      EmailText.Text = _CurrentUser.Email;
      CommentTextBox.Text = _CurrentUser.Comment;
      IsApprovedCheck.Checked = _CurrentUser.IsApproved;
      IsLockedOutCheck.Checked = _CurrentUser.IsLockedOut;
}
```

1.19 Aktualizace uživatelů v úložišti

Aktualizace uživatelů je téměř stejně snadná, jako získávání uživatelů z úložiště. Jakmile máme po ruce instanci MembershipUser, můžeme obvyklým způsobem aktualizovat vlastnosti, jako například e-mailovou adresu nebo komentář. Pak stačí zavolat metodu UpdateUser třídy Membership.

```
public partial class _Defauls : System.Web.UI.Page
{
      string userName = "EditovanyUzivatel";
      //Definování proměnné pro vybraného uživatele
      MembershipUser _CurrentUser;
      //Naplnění proměnné z úložiště dat
      _CurrentUser = Membership.GetUser(userName);
      //Svázání dat s jednotlivými ovládacími prvky stránky
      _CurrentUser.Email = EmailText.Text;
      _CurrentUser.Comment = CommentTextBox.Text;
      _CurrentUser.IsApproved = IsApprovedCheck.Checked;
      //Aktualizace uživatelských informací
      Membership.UpdateUser(_CurrentUser);
}
```

Metoda UpadateUser jednoduše přijme upravenou instanci MembershipUser, neboli uživatele, kterého chceme aktualizovat. Než ji zavoláme, musíme aktualizovat vlastnosti instance. Je tu však několik výjimek, některé vlastnosti se nastavovat nedají. Nastavují se automaticky při vyvolání příslušné události. Mezi tyto vlastnosti patří například IsLockedOut, Password atd. Ke změně těchto vlastností obsahuje třída MmebershipUser metody jako jsou například GetPassword, ChangePassword nebo UnlockUser.

1.20 Vytváření a odstraňování uživatelů

Uživatele přidáváme do úložiště tak, že jednoduše zavoláme metodu CreateUser třídy Membership. Pokud tedy chceme do svého webu zařadit schopnost vytvářet uživatele, můžeme přidat novou stránku obsahující nebytná textová pole, do nichž se budou zadávat požadované informace, a tlačítko. Událost Click tohoto tlačítka budeme zachytávat pomocí následujícího kódu.

```
protected void ActionAddUser_Click(object sender, EentArgs e)
{
      try
      {
            Membership.CreateUser(UsernameText.Text,
                                  PasswordText.Text,
                                  UserEmailText.Text,
                                  PwdQuestionText.Text,
                                  PwdAnswerText.Text);
      }
      catch(Exception ex)
      {
            Debug.WriteLine("Exception: " + ex.Message);
      }
}
```

Odstraňování uživatelů je stejně snadné jako jejich vytváření. Třída Membership nabízí metodu Delete, která požaduje jako parametr uživatelské jméno. Metoda odstraní uživatele z úložiště dat, a pokud chceme, odstraní i všechny informace, které se k němu vztahují.

1.21 Ověřování uživatelů

Poslední metodou, o které se budu zmiňovat, je metoda pro ověření uživatele Membership API. Když nějaký uživatel zadá na nějakém přihlašovacím místě své uživatelské jméno a heslo, můžeme programátorsky ověřit informace zadané uživatelem tímto způsobem.

```
if (Membership.ValidateUser(UsernNameText.Text, PasswordText.Text))
{
      FormsAuthentication.RedirectFromLoginPage(UserNameText.Text, false);
{
else
{
      //Neplatné užvatelské jméno nebo heslo
}
```

Motivy

Stylové předpisy (tzv. pravidla CSS) se omezují na fixní sadu atributů stylu. Umožňují opětovně využívat konkrétní formátovací detaily, ale evidentně neumějí řídit jiné aspekty ovládacích prvků ASP.NET. A kromě toho bychom také chtěli třeba jednotně definovat část chování ovládacího prvku, nikoli pouze jeho vzhled. Chtěli bychom například standardizovat režim výběru v ovládacím prvku Calendar, nebo způsob zalamování v textových polích. To evidentně prostřednictvím CSS udělat nejde.

Motivy (Themes) tuto mezeru vyplňují. Podobně jako CSS, i motivy umožňují definovat sadu stylových atributů, které se pak aplikují na ovládací prvky na více stránkách. Ovšem na rozdíl od CSS nejsou motivy implementovány prohlížečem. Je to proprietní řešení ASP.NET implementované na serveru. I když motivy rozhodně nemají nahradit styly, mají některé schopnosti, které CSS poskytnout nemohou. Klíčové rozdíly jsou popsány v následujícím výčtu.

- Motivy jsou založené na ovládacích prvcích, nikoliv na HTML. Důsledkem je, že motivy umožňují definovat a opětovně využívat téměř jakoukoliv vlastnost ovládacího prvku. Stylové předpisy CSS jsou omezeny pouze na ty atributy stylu, které se aplikují přímo na HTML.
- Motivy se aplikují na serveru. Když se na stránku aplikuje motiv, uživateli se odešle finální stránka, kde u jsou promítnuty všechny změny. Když se na stránku aplikuje stylový předpis, obdrží prohlížeč jak stránku, tak informace o stylu – to znamená, že se aplikují až na stroji klienta.
- Motivy lze aplikovat prostřednictvím konfiguračních souborů. Umožňuje to aplikovat jeden motiv na celou složku nebo na celý web, a nemusíme přitom modifikovat ani jedinou webovou stránku.
- Motivy nejsou kaskádové jako CSS. Specifikujeme-li nějakou vlastnost v motivu i v ovládacím prvku, přepíše hodnota motivu hodnotu vlastnosti v ovládacím prvku. Máme však možnost změnit toto chování a dát přednost vlastnostem ve stránce, takže se bude chování motivů více podobat chování stylových předpisů.

Motivy nenahrazují CSS, spíše reprezentují model vyšší úrovně, dále je možné použít stylový předpis jako součást motivu. Nejsme tedy omezeni tím, že musíme zvolit mezi jednou, nebo druhou variantou.

1.22 Složky motivu a skinové předpisy

Motivy se vztahují k celé aplikaci. Chceme-li ve webové aplikaci použít motiv, musíme vytvořit složku, která bude motiv definovat. Tuto složku musíme umístit do složky s názvem App_Theme, která musí být v kořenové části složky aplikace.

Aplikace může obsahovat definici více motivů, pokud je každý motiv v samostatné složce. Pro danou stránku může být v daném okamžiku aktivní pouze jediný motiv.

Soubor skinových předpisů je v podstatě seznam značek ovládacích prvků. Značky nemusí ale plně definovat ovládací prvek. Nastavují se jen ty vlastnosti, které chceme standardizovat jak je uvedeno na následující ukázce.

```
<asp:ListBox runat="server" ForeColor="White" BackColor="Orange" />
```

Část runat=“server“ je vždy povinná, vše ostatní je nepovinné. Atribut ID není povolen, protože se požaduje jako jednoznačný identifikátor ovládacího prvku.

1.23 Aplikace jednoduchého motivu

Chceme-li motiv aplikovat na webové stránce, musíme nastavit atribut Theme direktivy Page na název složky našeho motivu.

```
<%@ Page Language="C#" AutoEventWireup="true" ... Theme="DefaultTheme" %>
```

Když aplikujeme na stránku nějaký motiv, ASP.NET prochází všechny ovládací prvky nacházející se na stránce a kontroluje, zdali nejsou v souborech skinových předpisů pro ně definovány nějaké vlastnosti. Najde-li shodnou značku, překryjí informace ze souboru skinových předpisů aktuální vlastnosti ovládacího prvku.

1.24 Použití CSS v motivu

ASP.NET poskytuje možnost použít jako část motivu stylový předpis. Chceme-li ho v motivu použít, musíme nejprve stylový předpis přidat do složky motivu. ASP.NET tuto složku prohledává a dynamicky váže všechny soubory .css ke všem stránkám, které používají daný motiv.

Abychom mohli svázat stránku s nějakým stylovým předpisem, musí být ASP.NET schopno přidat do sekce <head> webové stránky značku <link>. To je možné pouze tehdy, má-li značka <head> atribut runat="server". Tím se prvek změní na ovládací prvek na straně severu a ASP.NET ho bude moci modifikovat.

Dále stačí už jenom nastavit atribut Theme stránky, aby získala přístup k pravidlům stylového předpisu. Pak můžeme nastavit vlastnost CssClass těch ovládacích prvků, které chceme naformátovat.

1.25 Aplikování motivů prostřednictvím konfiguračního souboru

Pomocí direktivy Page můžeme motiv svázat s jedinou stránkou. Jestliže chceme použít motiv pro celou webovou aplikaci, nejčistším způsobem, jak se dá takto motiv aplikovat, je nakonfigurovat <pages> v souboru web.config aplikace.

```
<configuration>
      <system.web>
            <pages theme="DefaultTheme" />
      </system.web>
</configuration>
```

Specifikovaný motiv se bude aplikovat na všechny stránky webu, za předpokladu, že stránky nemají nastavený svůj vlastní motiv. V opačném případě nastavení motivu ve stránce má přednost před nastavením v souboru web.config.

1.26 Shrnutí a srovnání s ASP.NET 1.x

Ve starších verzích ASP.NET jsme neměli možnost vkládat již hotové kontroly, proto nebylo ani nutné a možné používat Motivy stránek. Dalo by se říct, že motivy stránek mají za úkol tvořit nadstavbu CSS stylům, pro úpravu vzhledu a chování kontrol, které máme v novějších verzích ASP.NET k dispozici.

Ve starších verzích jsme měli možnost pro úpravu vzhledu používat standardní prvky, jako jsou kaskádové styly a samozřejmě html znaky. Neměli jsme tedy možnost, pomocí přiřazení skinu vkládanému objektu nadefinovat jeho vzhled a nějaké standardní nastavení v chování, což nám motivy umožňují.

Master Pages

Aby byly Mater Pages praktickým a flexibilním řešením, musejí vyhovovat mnoha požadavkům:

- Část stránky se dá definovat samostatně a opětovně využívat na více stránkách.
- Dá se vytvořit uzamčené rozvržení s definovanými regiony, které bude možné upravovat. Stránky, které budou opětovně využívat šablonu tohoto druhu, pak boudou mít povoleno přidávat či modifikovat obsah jen v těchto regionech.
- Dá se povolit přizpůsobování prvků, které se na stránkách opětovně využívají.
- Stránka se dá deklarativně svázat s nějakou stránkou šablony, nebo programátorsky při běhu.
- Stránka, která používá šablonu stránky, se dá navrhnout v nějakém designérském nástroji, jako je například Visual Studio

Master Pages vyhovují všem těmto požadavkům. Poskytují nejenom systém pro opětovné využívání šablon, ale také způsob, jakým lze šablony modifikovat, a v neposlední řadě rovněž bohatou podporu pro dobu návrhu.

ASP.NET definuje dva nové typy stránek. Stránky, které slouží jako vzor, a stránky s obsahem. Stránka, která slouží jako vzor, neboli Master Page, je nějaká šablona stránky. Může obsahovat libovolnou kombinaci HTML, webových ovládacích prvků, a dokonce i kódu. Ve vzoru stránek mohou být kromě toho také zástupci obsahu (content placeholders) – definované regiony, které se mohou modifikovat. Každá stránka s obsahem (Content Page) se odkazuje na jediný vzor stránek a převezme ze vzoru rozvržení a obsah. Stránka s obsahem může kromě toho přidávat do jakýchkoliv zástupců obsahu svůj vlastní obsah, který je specifický pro danou stránku.

1.27 Jednoduchá Master Page

Master Page se podobá normálnímu webovému formuláři ASP.NET. Jedním z rozdílů je to, že zatímco webový formulář zahajuje direktiva Page, Master Page začíná direktivou Master, ve které se specifikují stejné informace:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="WebTemplate.master.cs" Inherits="WebTemplate_master" %>
```

Dalším rozdílem mezi Master Page a obyčejným webovým formulářem je to, že Master Page může obsahovat ovládací prvek ContentPlaceHolder, který není na obyčejných stránkách povolený. ContentPlaceHolder je ta část stránky, do které pak může stránka s obsahem vkládat obsah. Master Page může obsahovat libovolný počet ContentPlaceHolderů.

Ukázka použití MasterPage se statickým záhlavím a zápatím stránky:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="WebTemplate.master.cs" Inherits="WebTemplate_master" %>
<html xmlns=http://www.w3.org/1999/xhtml>
<head runat="server">
      <title>Nadpis stránky</title>
</head>
<body>
      <form id="form1" runat="server">
      <div class="header">
            Statická hlavička stránky
      </div>
      <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
      </asp:ContentPlaceHolder>
      <div class="footer">
```

```
            Statické zápatí stránky
      </div>
      </form>
</body>
</html>
```

1.28 Jednoduchá Content Page

Chceme-li aplikovat Master Page v obyčejné webové stránce, musíme do direktivy Page přidat atribut MasterPageFile. Specifikuje název souboru Master Page, který chceme použít:

```
<%@ Page Language="C#" MasterPageFile="~/WebTemplate.master"
AutoEventWireup="true" CodeFile="SimpleContentPage.aspx.cs"
Inherits="SimpleContentPage_aspx" Title="Title Page" %>
```

Jenom nastavení atributu MasterPageFile ale nestačí k tomu, aby se stránka transformovala na stránku s obsahem používající Master Page. Stránka s obsahem má na starosti jediné, a to definovat obsah, který se vloží do jednoho nebo několika ovládacích prvků ContentPlaceHolder. Stránka s obsahem nedefinuje samotnou stránku, protože si ji už převzala ze vzoru stránek. Nemůžeme tedy vkládat prvky jako <html>, <head> nebo <body> protože už jsou definovány v Master Page.

Obsah pro ContentPlaceHolder se poskytuje pomocí specializovaného ovládacího prvku Content. Pro každý ContentPlaceHolder v Master Page dodá Content Page odpovídající ovládací prvek Content (kromě situace, že bychom pro daný region nechtěli dodat vůbec žádný obsah). ASP.NET propojí odpovídající ovládací prvek Content s patřičným ContentPlaceHolder tak, že najde k ID ovládacího prvku ContentPlaceHolder shodnou hodnotu vlastnosti Content.ContentPlaceHolderID ovládacího prvku Content.

Ukázka kompletní Content Page:

```
<%@ Page Language="C#" MasterPageFile="~/WebTemplate.master"
AutoEventWireup="true" CodeFile="SimpleContentPage.aspx.cs"
Inherits="SimpleContentPage_aspx" Title="Title Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
      <div class="bodyPage">
            Obsah jedoduché content page vkládaný do master page.
      </div>
</asp:Content>
```

1.29 Dynamické nastavení Master Page

Někdy můžeme potřebovat změnit Master Page za pochodu. Programátorským způsobem provést tuto změnu je velmi jednoduché. Nemusíme dělat nic víc, než nastavit vlastnost Page.MasterPageFile. Podmínkou je ale to, že tento krok musíme udělat v etapě Page.Init. Pokusíme-li se to udělat později, způsobí to výjimku.

1.30 Shrnutí a srovnání s ASP.NET 1.x

Opět ve starších verzích ASP.NET 1.x nám chybí možnost jednoduchého, rychlého a pohodlného vytváření rozvržení klasických webů. Většinou chceme, aby stránky měli nějakou hlavičku, navigační menu, a hlavní tělo stránky, kde by se zobrazoval příslušný obsah. Ve starších verzích ASP.NET musíme toto řešit na úrovni html kódu například pomocí „frames“. Další možností je použít nějaký programový kód, který nám bude simulovat funkci Master Page a jednotlivých vkládaných obsahových částí.

Ovládací prvky pro zobrazení dat

ASP.NET 2.0 nám poskytuje řadu ovládacích prvků pro zobrazení dat. Spojením těchto prvků s příslušnými zdroji dat získáme mocné nástroje pro jednoduché zobrazení databázových i jiných dat použitých v naší webové aplikaci.

1.31 Ovládací prvek GridView

GridView je neobyčejně flexibilní ovládací prvek pro zobrazování dat. Obsahuje širokou škálu interně zabudovaných schopností, jako jsou výběr, stránkování a editování, a dá se rozšiřovat pomocí šablon. Skvělou předností GridView oproti starému ovládacímu prvku z ASP.NET 1.0 DataGridu je jeho podpora scénářů. Pomocí GridView se dá vyřešit mnoho běžně vyskytujících úloh, jimiž jsou stránkování a výběr, aniž bychom museli psát nějaký další kód.

Použijeme-li ovládací prvek GridView s nastavenou vlastností AutoGenerateColumns na true, prozkoumá GridView datový objekt pomocí reflexe a vyhledá všechny sloupce (záznamu), nebo vlastnosti (vlastního objektu). Pak vytvoří své sloupce v tom pořadí, v jakém odpovídající sloupce, respektive vlastnosti našel.

Automatické generování sloupců je šikovné, když potřebujeme rychle vytvořit nějaké testovací stránky, neposkytuje však takovou flexibilitu, jakou obvykle potřebujeme v praxi. Chceme-li získat plnou flexibilitu ovládacího prvku, jako například změnit pořadí sloupců, nebo nakonfigurovat nějaký aspekt jejich zobrazení, jako je formát nebo text v záhlaví, musíme nastavit vlastnost AutoGenerateColumns na false a definovat sloupce v sekci <Columns>.

Použití ovládacího prvku s ručně definovanými sloupci:

```
<asp:GridView ID="gridEmployees" runat="server"
DataSource="sourceEmployees" AutoGenerateColumns="false">
      <Columns>
            <asp:BoundField DataField="FirstName"
                  HeaderText="First Name" />
            <asp:BoundField DataField="LastName" HeaderText="Last Name" />
            <asp:BoundField DataField="Title" HeaderText="Title" />
            <asp:BoundField DataField="City" HeaderText="City" />
      </Columns>
</asp:GridView>
```

1.31.1 Formátování polí vázaných na data

Každý sloupec typu BoundField poskytuje vlastnost DataFormatString, s jejíž pomocí můžeme nakonfigurovat vzhled hodnot vyjadřujících čísla nebo datum pomocí formátovacího řetězce.

Použití formátovacího řetězce ve sloupci:

```
<asp:BoundField DataField="Price" HeaderText="Price" DataFormatString="{0:
C}" />
```

Formátovací řetězce se obvykle skládají ze zástupných symbolů a indikátorů, které jsou uvnitř složených závorek. Typický formátovací řetězec vypadá takto:

```
{0:C}
```

V tomto případě reprezentuje 0 hodnotu, která se bude formátovat, a písmeno vyjadřuje předdefinovaný formátovací styl. C zde znamená formát měny, a slouží k naformátování čísla jako částky se symbolem měny, podle místního nastavení.

1.31.2 Šablony

Předchozí příklad pro ovládací prvek GridView, zobrazoval data pomocí oddělených vázaných sloupců pro každý sloupec ze zdroje dat. Chceme-li umístit do jedné buňky více než jednu hodnotu, nebo mít neomezenou možnost přizpůsobovat obsah buňky tím, že budeme do ní přidávat nějaké HTML značky a serverové ovládací prvky, budeme potřebovat sloupec typu TemplateField.

Sloupec typu TemplateField umožňuje definovat kompletně přizpůsobitelnou šablonu sloupce. Do šablony můžeme přidávat značky ovládacích prvků, libovolné HTML prvky či výrazy vázání dat. Máme úplnou volnost, takže si můžeme vše uspořádat tak, jak chceme.

Ukázka použití šablony v prvku GridView:

```
<asp:GridView ID="gridEmployees" runat="server"
DataSource="sourceEmployees" AutoGenerateColumns="false">
      <columns>
            <asp:TemplateField HeaderText="Name">
                  <ItemTemplate>
                        <%# Eval("TitleOfCourtesy") %>
                        <%# Eval("FirstName") %>
                        <%# Eval("LastName") %>
                  </ItemTemplate>
            </asp:TemplateField>
      </columns>
</asp:GridView>
```

Když teď svážeme GridView s daty, vytáhne si data ze zdroje dat a projde kolekci prvků. Pro každý z prvků zpracuje ItemTemplate, vyhodnotí výrazy vázání dat a přidá vygenerovaný HTML kód do tabulky. Tato šablona je velmi jednoduchá – jednoduše definuje tři výrazy vázání dat. Když se výrazy vyhodnotí, převedou se na obyčejný text.

Ve výrazech se volá Eval(), což je statická metoda třídy System.Web.UI.DataBinder.Eval(), automaticky totiž získá prvek dat, který je vázaný k aktuálnímu řádku prvku GridView, pomocí reflexe najde odpovídající sloupec databázové tabulky (jedná-li se o řádek), nebo vlastnost (jedná-li se o vlastní objekt), a získá hodnotu. Proces reflexe vnáší trochu práce navíc, není však pravděpodobné, že by tato dodatečná zátěž nějak znatelně prodlužovala zpracování požadavků. Pokud bychom nepoužili metodu Eval(), museli bychom k datovému objektu přistoupit prostřednictvím vlastnosti Container.DataItem a přetypovat jej, podobně jako zde:

```
<%# ((EmployeeDetails)Container.DataItem) ["FirstName"] %>
```

Nevýhodou tohoto přístupu je, že musíme znát přesný typ datového objektu. Například právě uvedený výraz vázání dat předpokládá, že svazujeme objekty EmployeeDetails prostřednictvím ObjectDataSource. Přejdeme-li k SqlDataSource, nebo přejmenujeme-li třídu EmployeeDetails, svou stránku tak narušíme. Když oproti tomu použijeme metodu Eval (), bude výraz vázání dat fungovat pořád.

1.32 Ovládací prvek DetailsView

Účelem ovládacího prvku DetailsView je zobrazit v daném okamžiku jeden záznam. Každý elementární díl informace (pole nebo vlastnost) umístí do separátního řádku tabulky.

Použití tohoto ovládacího prvku svádí k tomu, aby se člověk pokusil pomocí stránkovacího ovládacího prvku DetailsView vyrobit něco jako příruční prohlížeč záznamů. Takový přístup může ale být hodně neefektivní. Například vždy, když uživatel přejde z jednoho záznamu na jiný, vyžaduje to separátní odeslání stránky zpět na server (zatímco ovládací prvek GridView umí zobrazit najednou celou sadu záznamů). Ale opravdovým nedostatkem je to, že pokaždé, když se stránka odešle zpět na server, získává se úplná sada záznamů, i když se přitom zobrazuje jen jediný z nich. Rozhodneme-li se vytvořit stránku pro prohlížení záznamů po jednom pomocí DetailsView, musíme přinejmenším

zapnout ukládání do cache, abychom alespoň trochu zredukovali práci, kterou musí vykonávat databáze.

Často je lepší, když si vytvoříme naše vlastní stránkovací ovládací prvky pomocí nějaké podmnožiny dat. Mohli bychom například vytvořit rozevírací seznam a ten svázat se zdrojem dat, takže bychom se databáze dotazovali pouze na klíčové položky záznamů. Až by pak uživatel vybral některou položku, stačilo by pak pomocí jiného zdroje dat získat kompletní informace pro právě vybraný záznam.

1.32.1 Definice polí

DetailView generuje pole, která zobrazuje, pomocí reflexe. To znamená, že prozkoumá datový objekt a vytvoří separátní pole pro každý sloupec (řádku) nebo vlastnost (vlastního objektu) právě tak, jako GridView. Automatické generování polí můžeme vypnout, když nastavíme AutoGenerateRows na false. Pak bude na nás, abychom deklarovali objekty polí.
Při budování DetailsView používáme stejné objekty, jako když jsme navrhovali GridView. Například pole z datového prvku jsou reprezentovány značkou BoundField, tlačítka se dají vytvářet pomocí ButtonField atd.

Ukázka deklarace polí DetailsView:

```
<asp:DetailsView ID="DetailsView1" runat="server" AutogenerateRows="false">
      <Fields>
            <asp:BoundField DataField="EmployeesId" HeaderText="Id" />
            <asp:BoundField DataField="FirstName"
                  HeaderText="First Name" />
            <asp:BoundField DataField="LastName" HeaderText="Last Name" />
            <asp:BoundField DataField="Title" HeaderText="Title" />
            <asp:BoundField DataField="City" HeaderText="City" />
      </Fields>
</asp:DetailsView>
```

1.33 Ovládací prvek FormView

DetailsView podporuje všechny typy sloupců GridView kromě sloupců založených na šabloně. Potřebujeme-li nenahraditelnou flexibilitu šablon, obrátíme se na FormView, který poskytuje ovládací prvek šablony pro zobrazení a editaci jediného záznamu.

Krása modelu šablony FormView tkví v tom, že se vcelku shoduje s modelem TemplateField v GridView. To znamená, že můžeme pracovat s následujícími šablonami:

- ItemTemplate
- EditItemTemplate
- InsertItemTemplate
- FooterTemplate
- HeaderTemplate
- EmptyDataTemplate
- PageTemplate

Znamená to také, že můžeme vzít obsah šablony, který jsme umístili do nějakého sloupce typu TemplateField v nějakém GridView, tak, jak je, a umístit ho do FormView.

Chceme-li podporovat editaci, budeme muset přidat ovládací prvky tlačítek, které budou spouštět proces Edit a Update.

1.34 Shrnutí a srovnání s ASP.NET 1.x

S novějšími verzemi ASP.NET 2.0 přišla i podpora hotových předdefinovaných kontrol, které nám usnadňují spoustu práce, jak při návrhu, tak i díky jejich propracovanosti při používání webových aplikací. Stejné je to i v případě přístupu a zobrazení dat na dané stránce.

Ve starších verzích ASP.NET 1.x jsme neměli k dispozici kontroly jako GridView, FormView atd. Při návrhu webové aplikace jsme museli všechny formuláře a jejich funkcionalitu doplnit sami. Od verze 2.0 třeba prvek FormView nám zaručí jednoduché vkládání, editaci záznamu například z připojené databáze. Data z dané DB můžeme opět jednoduše získat pomocí kontroly SqlDataSource, která nám pak jimi naplní příslušný připojený formulář.

Závěr

Přínosem této práce je představení čtenáři nové možnosti rámce .NET 2.0 pro tvorbu informačních systémů. Čtenář se okrajově seznámí s vývojovými nástroji použitelnými pro vývoj webových aplikací založených na architektuře ASP.NET 2.0. Jsou představeny základní principy a kontroly pro správu a registraci uživatelů ve vytvářených webových aplikacích.

Čtenář se dále seznámí s možností nastavení jednotlivých vlastností kontrol a jejich jednoduchým použitím, ať už se jedná o kontroly již zmiňované pro správu uživatelů, nebo kontroly pro jednoduché zobrazování, editaci a vkládání nových dat do datového úložiště spojeného s vyvíjenou webovou aplikací.

Dále by se čtenář měl seznámit s možnostmi návrhu webových aplikací, ve kterých by měl být schopen použít základní dělení stránek pomocí Master Pages a dozvědět se o nových možnostech úpravy vzhledu pomocí motivů.

V práci se čtenář také dozví, jak se liší nová verze ASP.NET 2.0 od verzí starších. Spíše má možnost zjistit, co nového mu novější verze ASP.NET přináší a co vše mu nové kontroly ulehčují při návrhu informačního systému.

Předpokladem pro jednodušší porozumění práce je, aby čtenář měl již minimální znalosti v oblasti architektury ASP.NET.

Téma této bakalářské práce by se dalo dále rozvíjet. Mohly by být představeny méně významné kontroly, nebo by šlo dále navázat na téma nových možností rámce .NET 3.0.

Demonstrační informační systém by mohl být rozšířen o další schopnosti, a tím by se mohl stát v případě zájmu plnohodnotným redakčním systémem nějaké organizace.

Literatura

[1] Matthew MacDonald:
Apress – Beginning ASP.NET 2.0 in C# From Novice to Professional,
ISBN 978-1-59059-572-5

[2] Matthew MacDonald and Mario Szpusta:
Apress – Pro ASP.NET 2.0 in C# 2005,
ISBN 978-1-59059-496-4

[3] Chris Hart, John Kaufman, David Sussman, Chris Ullman:
Wrox – Beginning ASP.NET 2.0,
ISBN 978-0-7645-8850-1

[4] Bill Evjen, Scott Hanselman, FarhanMuhammad, Srinivasa Sivakumar, Devin Rader:
Wrox – Professional ASP.NET 2.0,
ISBN 978-0-7645-7610-2

[5] Stefan Schackow:
Wrox – Professional ASP.NET 2.0 Security, Membership, and Role Management,
ISBN 978-0-7645-9698-8

[6] Damien Foggon:
Apress – Beginning ASP.NET 2.0 Databases From Novice to Professional,
ISBN 978-1-59059-577-0

[7] URL <http://www.zive.cz> [duben 2007]