

# Vývoj webových aplikací v C#.NET

Filip Jurnečka, Martin Osovský

PV178

20. dubna 2011

- 1 Windows Presentation Foundation
  - XAML
- 2 Silverlight
- 3 ASP.NET
  - ASP.NET soubory
  - Web sites a Web applications
  - Stav ASP.NET aplikace
- 4 ASP.NET MVC
- 5 Routování
- 6 Controller
- 7 View

# WPF (Windows Presentation Foundation)

- Knihovna .NET pro tvorbu a práci s uživatelskými rozhraními, multimédií a dokumenty.
- Skládá se z několika assemblies obsahující tisíce tříd.
- Je obsažen v namespace `System.Windows` a všech subnamespacech kromě `System.Windows.Forms`
- Striktně rozlišuje design a vývoj aplikace využitím XAML.

# XAML (eXtensible Application Markup Language)

- Jazyk pro zjednodušení tvorby UI.
- XAML je case sensitive.
- Přímo provázaný s třídami .NET pomocí “partial class definitions”.
- Rozlišuje primárně “object elementy”, tedy typy, např.  
`<LinearGradientBrush>` a jejich atributy, tedy properties, např.  
`<Button Background="Blue" Foreground="Red"  
Content="This is a button" />`
- Pracuje typicky ve spojení s tzv. “code-behind”.

# Silverlight

- Technologie pro tvorbu webových aplikací. Od verze 3.0 lze tvořit i aplikace nezávislé na webovém prohlížeči.
- Funguje i na newindowsových platformách, jelikož je primárně plug-inem webového prohlížeče.
- Skládá se z core presentation frameworku (podmnožiny WPF), .NET Framework for Silverlight (podmnožiny .NET), instalačních a aktualizčních programů.
- Kompilátor vytvoří XAP soubor, což je ve skutečnosti ZIP obsahující assembly a konfigurační soubory.
- Na rozdíl od WPF Silverlight 3.0 nepodporuje
  - tvorbu dokumentů,
  - 3-D grafiky,
  - vzájemné propojení s Windows Forms.

# Omezení Silverlightu

**Pravé tlačítko myši** : je vždy odchyceno prohlížečem.

**Double-click, kolečko** : jsou součástí Silverlight 4.

**Fonty** : jsou omezeny na relativně malou množinu kvůli rozdílnosti cílových platform.

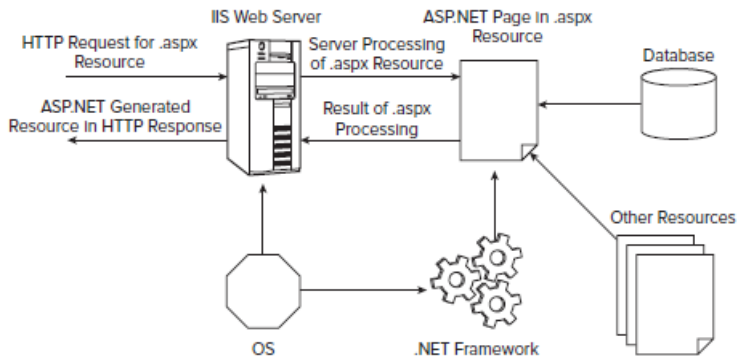
**Síťování** : jsou povoleny pouze asynchronní volání kvůli možné blokaci UI.

**Přístup na file system** : je omezen pouze na “isolated storage”. V Silverlight 4 je umožněn přístup na file system.

# ASP.NET

- Součást .NET Frameworku.
- Dovoluje dynamicky vytvářet dokumenty jako odpověď na HTTP požadavky.
- Tyto odpovědi můžou být jakéhokoliv MIME typu (tedy ne pouze HTML).
- Stylem a funkcionalitou velice podobné Windows Forms  $\Rightarrow$  *Web Forms*.
- ASP.NET stránky jsou uloženy v .aspx souborech.
- Kód obsažený v souboru třídy samostatné stránky ASP.NET odkazuje kódem na pozadí (*code-behind*). To umožňuje oddělování HTML prezentace a logiky aplikace.

# Architektura ASP.NET



Obrázek: Základní architektura ASP.NET



# ASP.NET soubory

ASP.NET soubor může obsahovat:

- Instrukce pro zpracování na serveru.
- Kód v libovolném .NET jazyku.
- Client-side script kód, jako např. JavaScript.
- Zabudované ASP.NET server controls komponenty.

# Web sites a Web applications

ASP.NET dovoluje tvorbu obojího. Rozdíl je ve způsobu zpracování.

**Web site** : kód je kompilován až při prvním požadavku na aplikaci.

**Web applications** : je předkompilována a na serveru nejsou .cs soubory, ale již celé assembly.

Obecně jsou preferovány aplikace z důvodu rychlosti. Web sites jsou používány hlavně v období vývoje.

# Stav ASP.NET aplikace

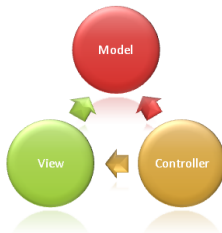
- ASP.NET neuchovává na serveru žádné stavové informace  $\Rightarrow$  je efektivně bezstavové.
- Stavové informace, jako aktuálně vybraný prvek v select listu, ve skrytém poli `viewstate` v odpovědi.
- Následné akce, jako submit formuláře opět posílají tyto informace serveru. Toto je známo jako `postback`.

# ASP.NET MVC

- MVC (Model - View - Controller) je framework pro tvorbu webových aplikací.
- Sídlí v namespace `System.Web.Mvc`.
- Integrovaný teprve do .NET 4 ve verzi MVC 2. Aktuálně je vydaná plná verze ASP.NET MVC 3.
- Snaží se čistěji oddělit uživatelské rozhraní a aplikační logiku.
- Na rozdíl od Web Forms umožňuje snadné testování.
- Využívá routovací metodologii zavedenou v .NET 3.5.

# Model - View - Controller

- Model** Odpovídá kódu, který reprezentuje data nebo stav aplikace. To nezahrnuje databázi, ale zahrnuje např. třídy manipulující s databází.
- View** Je uživatelské rozhraní zobrazující model a dovolující uživateli vykonávat akce na modelu.
- Controller** je zodpovědný za zprostředkování dat mezi modely a views. Když uživatel provede akci na view, controller na ni odpoví. Pokud je to nutné, interaguje s modelem.



# Routování v ASP.NET MVC

- Query String

`http://www.mysite.com/products.aspx?id=5&lang=cz`

- Po adrese cílové stránky může následovat query string indikovaný symbolem `?`.
- Zbytek URL sestává z jméno/hodnota párů se symbolem `=` oddělujícím jméno parametru (`id`, `lang`) od jejich hodnoty (`4`, `cz`).

- Definování cest

- Zavádí jednoduché, uživatelsky čitelné URL.
- Vyhledávací enginy jsou optimalizovány pro tyto deskriptivní URL, takže mají vyšší ranking.
- Vyžaduje manuální konfiguraci cest. Typicky aplikováno v `HttpApplication.Application_Start` event handleru.
- Kolekce definovaných cest je uchovávána v statické property `RouteTable.Routes`.

# Controller

- Controllery v MVC jsou zodpovědné za odpověď na uživatelský vstup, často tak provádějící změny v Modelu.
- Aby se třída stala controllerem, musí implementovat interface `Controller`. Aby jej pak našel routovací systém, musí navíc název třídy končit suffixem "Controller".
- Abstraktní třída `ControllerBase` přidává trochu API nad rozhraní `Controller`. Primárně zavádí properties `TempData` a `ViewData` - kontejnery pro předávání dat mezi dvěma následujícími požadavky a mezi controllerem a view.
- Standardní způsob psaní controllerů je podědit abstraktní třídu `System.Web.Mvc.Controller` implementující `ControllerBase`. Tato je zamýšlena jako základní třída pro všechny controllery.

# Akce

- Všechny veřejné metody controlleru se nazývají “akce”. Tyto jsou pak zavolatelné pomocí HTTP požadavků.
- Typicky by pak všechny akce měly mít návratovou hodnotu typu poděděného z abstraktní třídy `ActionResult`.
- Nejčastější návratový typ je `ViewResult`, který vygeneruje příslušný view na základě předaných dat.



# View

- View je zodpovědný za poskytování uživatelského rozhraní.
- Předaná data nazývána “Model” jsou transformována do formátu pro prezentaci uživateli.
- V ASP.NET MVC to znamená prohledání `ViewDataDictionary` předané controllerem pomocí property `ViewData` a transformací do HTML.
- ASP.NET MVC 3 přichází se dvěma view enginey, které jsou zodpovědné za tuto transformaci.
  - `WebFormsViewEngine` je implicitní a views jsou velice podobná těm ve Web Forms.
  - `RazorViewEngine` kladе důraz na kompaktnost a jednoduchost.
  - Dále existují např. “Spark” nebo “NHaml”.

# ViewEngine

- WebFormsViewEngine

- Na rozdíl od Web Forms neobsahuje tag `<form runat="server">`.
- Views jsou odvozené od základní třídy `System.Web.Mvc.ViewPage`, v kódu specifikováno přibližně takto

```
<%@ Page Language="C#"
MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage" %>
```
- Pro vložení kódu pro generování výstupu se používají "code nuggets"  
`<% %>`.

- RazorViewEngine

- Nahrazuje code nuggets jednodušším `@`.
- Nevyžaduje explicitní uzavírání bloků kódu - má sémantickou znalost C#/VB kódu.

Otázky?  
Na shledanou!

Otázky?  
Na shledanou!