Webová aplikace

- princip webové aplikace, důvody zavedení této platformy
- struktura webové aplikace, vrstvy
- Aplication Service Provider
- webový server a jeho funkce ve webové aplikaci
- metody přenosu dat od klienta na server
- uživatelské rozhraní webové aplikace
- technologie tvorby aplikační loginy webové aplikace
- datová vrstva

Princip webové aplikace, důvody zavedení této platformy

Webová aplikace je v softwarovém inženýrství aplikace poskytovaná uživatelům **z webového serveru** přes **počítačovou síť** Internet, nebo její vnitropodnikovou obdobu (intranet). Webové aplikace jsou populární především pro všudypřítomnost webového prohlížeče jako klienta. Prohlížeč se pak nazývá tzv. **tenkým** klientem**, neboť sám o sobě nezná logiku aplikace.**

Schopnost aktualizovat a spravovat webové aplikace bez nutnosti šířit a instalovat software na potenciálně tisíce uživatelských počítačů je hlavním důvodem jejich oblíbenosti. Webové aplikace jsou používány pro implementaci mnoha podnikových i jiných informačních systémů, ale i freemailů, internetových obchodů, online aukcí, diskusních fór, weblogů.

Struktura webové aplikace, vrstvy

Ačkoliv je mnoho možností, webové aplikace jsou obvykle strukturovány jako **třívrstvé**. V té nejběžnější formě je:

- **prezentační vrstva** (první vrstva) - webový prohlížeč
- **vrstva aplikační logiky** (střední vrstva) - nástroje pro dynamické generování stránek (např. CGI, PHP, javové servlety nebo ASP)
- **datová vrstva** (třetí vrstva) - např. databáze

Webový prohlížeč posílá požadavky střední vrstvě, která je obsluhuje prostřednictvím dotazů do databáze a generováním uživatelského rozhraní.

Aplication Service Provider (ASP)

Jednou ze strategií pro softwarové firmy je poskytnout přístup přes web k aplikacím, které byly dříve nabízeny a šířeny jako aplikace lokální. V závislosti na typu aplikace může takový přechod vyžadovat vývoj zcela odlišného uživatelského rozhraní určeného webovému prohlížeči nebo jen přizpůsobit stávající aplikaci pro použití jiné prezentační technologie. Tyto programy umožňují uživatelům platit měsíční či roční poplatek za používání aplikace, aniž by si jej museli nainstalovat na svůj pevný disk. Firmy podnikající s touto strategií jsou známé jako **poskytovatelé aplikačních služeb** (Application Service Provider - ASP).

Webový server a jeho funkce ve webové aplikaci

Samotná aplikace je umístěna na webovém serveru (řekněme centrálně), kam se k ní můžou připojit ostatní uživatelé. Webový server zpřístupňuje aplikaci uživatelům (klientům - skrz webový prohlížeč) a je na něm spouštěna samotná aplikace, která reaguje na uživatelovi podněty - vypiš XYZ, vlož ABC, smaž DGX, apod.

Nejpoužívanějším webovým serverem je **Apache**. Jedná se open-source projekt, který tak lze zdarma šířit, upravovat a měnit. Apache je velice komplexní a robustní server. Z toho důvodu je mu někdy vyčítána komplikovanost (např. komplikovaná změna nastavení - existují desítky různých konfiguračních voleb - direktiv).

Často nasazovaným je také webserver **nginx** (čti engine-x). Ten je zaměřen především na vysoký výkon. Základním cílem je rychlá distribuce statického obsahu a možnost rozložení zátěže na další servery dle nastavené priority. Často bývá nasazován tam, kde je nutné poskytnout uživateli přístup k velkému množství statických souborů (např. obrázků, statických HTML souborů). V těchto případech bývá rychlejší než např. Apache. Bývá používán i jako proxy server. Často ho nasazují velké firmy (např. Seznam.cz, Nokia, GitHub (služba pro sdílení zdrojového kódu), WordPress, Dropbox, apod.).

Na platformě Windows (především v rámci **Windows Serveru**) se používá webový server od Microsoftu - IIS. Jedná se o softwarový webový server s kolekcí rozšiřujících modulů, vytvořený společností Microsoft pro operační systém Windows. IIS 7.5 podporuje protokoly HTTP, HTTPS, FTP, FTPS, SMTP a NNTP. Je součástí produktové řady Windows Server a také některých vydání Windows XP, Windows Vista a Windows 7. IIS není ve výchozím nastavení Windows zapnut.

Pro generování dynamických webových stránek se používá řada technologií:

- **SSI (Server Side Includes)**
  - do HTML kódu se zapisují jednoduché instrukce, které zpracovává přímo webový server
  - to, že se v souboru mají hledat SSI, se pozná podle přípony souboru (obvykle .shtml)
- **CGI**
  - aplikace používající Common Gateway Interface - protokol, který definuje způsob komunikace webového serveru s aplikací
  - CGI aplikace (nejčastěji skripty) lze psát v téměř libovolném jazyce, stačí dodržet konvence rozhraní Perl, C/C++, Pascal, Python, ...
  - rozhraní CGI bylo v prostředí internetu přítomno již od počátku 90. let a ve své době představovalo jediný způsob dynamického zpracování obsahu
  - podpora CGI nebývá implicitní, musí se ve web-serveru zapnout (bezpečnost)
  - hlavní nevýhoda: pro obsluhu každého požadavku je spouštěn nový proces
- **FastCGI**
  - vylepšená varianta CGI, které významně snižuje zátěž serveru
  - každý skript se do paměti načítá jen jednou, pak postupně obsluhuje další požadavky
  - web-server s aplikací komunikuje pomocí TCP/IP
  - web-server a aplikaci je možné rozdělit na samostatné počítače
  - po určitém počtu obsloužených požadavků skript je možné kdykoliv podle potřeby ukončit, webový server si ho při dalším požadavku sám znovu spustí
- **SAPI (moduly webserverů)**
  - v průběhu času začala většina serverů nabízet kromě CGI rozhraní i speciálně přizpůsobené rozhran,

které umožní, že aplikace je integrována přímo do webového serveru (nejčastěji jako DLL knihovna)
  - dnes nejpoužívanější je ISAPI – podporují ho servery Microsoftu a mnohé další
  - do paměti se podobně jako FastCGI skripty načtou při prvním požadavku a pak v ní již zůstanou
  - SAPI moduly jsou binární nativní kód – pro tvorbu si musíme sehnat vhodný kompilátor
- **ASP (Active Server Pages)**
  - přímo do HTML kódu se zapisují jednoduché příkazy
  - ASP je tzv. framework
  - standardně je používán programovací jazyk JScript nebo VBScript
  - ve jazycích jsou dostupné základní objekty s důležitými informacemi (data z formulářů apod.)
  - standardní součást webových serverů MS
  - podpora jiných serverů a platforem je velice slabá
- **ASP.NET**
  - ASP.NET je součást .NET Frameworku pro tvorbu webových aplikací a služeb
  - ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku
  - programátoři tak mohou realizovat své projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic.NET, JScript.NET, C#, Managed C++, ale i mutace Perlu, Pythonu a další
  - aplikace založené na ASP.NET jsou rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu interpretovány
- **PHP (PHP Hypertext Preprocessor)**
  - přímo do HTML kódu se zapisují jednoduché příkazy
  - jednoduchá syntaxe založená na C, Perlu a Javě
  - speciálně navržený jazyk pro tvorbu webových aplikací
  - velmi rozsáhlá knihovna funkcí
  - nezávislost na platformě – může spolupracovat s v podstatě libovolným serverem na libovolné platformě
  - dostupný zdarma včetně zdrojových kódů
- **Java servlet**
  - servlet je speciální třída zapsaná v jazyce Java
  - podobně jako u ISAPI a FastCGI zůstává servlet po prvním načtení v paměti a obsluhuje další požadavky
  - vyžaduje webserverovou podporu Javy - např. Apache Tomcat pro server Apache nebo Adobe JRun pro server IIS
- **JSP (Java Server Pages)**
  - do HTML kódu se zapisují příkazy Javy
  - k dispozici jsou (podobně jako v ASP) speciální objekty pro čtení dat předávaných pomocí HTTP
  - zdrojový kód stránky JSP je vyparsovaný a následně kompilovaný na servlet (bytecode), který generuje HTML kód
  - vyžaduje webserverovou podporu Javy - např. Apache Tomcat pro server Apache nebo Adobe JRun pro server IIS
- a další programovací jazyky nebo frameforky, jako jsou např. Ruby on Rails, Django - Python, ...

Metody přenosu dat od klienta na server

Pro přenos dat od uživatele na server (např. data z formuláře) se používají dvě metody - metoda **POST** a

metoda **GET**.

Data odeslaná metodou **GET** se stanou **součástí URL**, doplní se za otazník a jsou tedy vidět v adresním řádku prohlížeče. Naopak metoda **POST** data odesílá **odděleně od URL**.

Pokud se předaná data dají chápat jako parametry stránky (tedy např. předání ID článku nebo vyhledávaného řetězce), je vhodné je poslat metodou GET, v ostatních případech je lepší použít metodu POST (obzvláště pokud je dat hodně – např. Internet Explorer dokáže zpracovat URL jen do délky 2083 znaků).

Jakékoliv stažení objektu ze serveru je provedením příkazu GET. Vyvolá jej každé kliknutí na odkaz `<a href=...>`, transparentně jej provede např. stažení obrázku (`<img src=...>`).

POST je ryze formulářová záležitost. Data se odesílají mimo URL – to se ovšem týká jen položek formuláře. Zároveň je totiž možné odesílat ještě další data v URL, které je uvedeno v action formuláře. Zjednodušeně řečeno, POST umí totéž co GET a ještě o dost víc.

Použití metody POST by také mělo být samozřejmé v případě, kdy předaná data obsahují citlivé údaje (jako např. heslo), protože jinak se tato data dají získat řadou způsobů (např. z historie prohlížeče, na proxy serveru, z logů webového serveru nebo z HTTP hlavičky Referer).

Metoda POST se používá hlavně v případě, když:

- odeslání formuláře způsobí vnitřní změnu serveru (zápis do databáze, odeslání e-mailu, ...)
- odesílaná data nejsou zcela veřejná
- velikost dat může překročit 1000 bajtů (např. odeslání souboru na server - v tom případě je potřeba ve formuláři uvést atribut `enctype="multipart/form-data"`)

Uživatelské rozhraní webové aplikace

Uživatelské rozhraní webové aplikace je podobně jako obyčejná webová stránky (ona to je totiž ve skutečnosti webová stránka) napsána pomocí HTML. Vizuální podoba rozhraní a rozložení jednotlivých prvků na stránce je v režii kaskádových stylů (CSS). Pro získání většího dojmu interaktivity se používá klientské skriptování - pomocí skriptovacího jazyka JavaScript.

JavaScript manipuluje s prvky na stránce pomocí objektového rozhraní DOM (Document Object Model - objektový model dokumentu). Např. můžeme prvky na stránce dynamicky měnit na základě činnosti uživatele. S JavaScriptem a moderními webovými aplikacemi souvisí i technologie AJAX.

AJAX se používá pro přenos dat ze serveru ke klientovi bez nutnosti načítat znovu celou stránku. Pouze se stáhnou potřebná data nebo úryvek HTML kódu. Tento kód nebo data následně nahradí nebo doplní určitou oblast stránky (stáhnou se např. nejnovější příspěvky v chatu, stáhne a zobrazí se náhled výsledné podoby blogového příspěvku, apod.).

Součástí webového uživatelského rozhraní jsou **odkazy**, **tabulky**, **tlačítka**, **formuláře** a případně další prvky odpovídající konkrétnímu účelu aplikace.

Technologie tvorby aplikační loginy webové aplikace

Ačkoli je mnoho webových aplikací psáno přímo v čistém programovacím jazyce jako je PHP či Perl, existuje pro jejich tvorbu řada systémů, tzv. **frameworků**, které díky automatizaci tohoto procesu nabízejí programátorům možnost popsat program na vyšších úrovních. Užití takových systémů může často snížit počet chyb v aplikaci, především díky větší jednoduchosti a přehlednosti kódu a také možností koncentrovat se na důležitější části kódu.

Datová vrstva

- zajišťuje práci s vlastními daty aplikace
- uchovává tato data a skrz dané rozhraní je zpřístupňuje vlastní aplikační logice (samotné aplikaci)
- často se jedná o databázi, kterou pohání některý z databázových serverů (MySQL, MSSQL, Oracle, ...)
- v databázi jsou data uchovávána v databázových **tabulkách**
- nic však nebrání tomu napsat si datovou vrstvu založenou na ukládání a čtení dat do/ze souborů

Použití databáze má tu výhodu, že poskytuje dostatečnou abstrakci pro operace s daty. Tyto operace bývají často dobře navržené a optimalizované pro maximální možný výkon při práci s daty. Umožňují tak velice rychle vybírat data z tabulek, provádět s daty výpočty, spojovat data z více tabulek do jedné datové sady, rychle nalezená data řadit (vzestupně/sestupně) apod. V případě databází se pro manipulaci s daty nejčastěji používá jazyk **SQL**.