Webové programování a Open Source

Lukáš Zapletal

Liberix o.p.s.

23. 11. 2006

Obsah prezentace

Základy

- protokol HTTP (klient-server), jazyk HTML/XHTML, CSS
- funguje systémem dotaz-odpověď (neuchovává stav)
- HTML/XHTML – nutno dodržovat standardy, validovat
- dnešní směr - oddělení dat a grafiky na úrovni XHTML (grafika v CSS, vlastní obsah v XHTML)

WWW

- webový server – program pro ”podávání” dokumentů
- klient (prohlížeč) protokolem HTTP požádá o dokument (obvykle HTML – obyčejný text), ten poté zobrazí v okně aplikace, přičemž po serveru požaduje další dokumenty, které najde na stránce (obrázky, flash soubory atd)
- server tedy jen podává dokumenty, obrázky, soubory (často se HTTP používá i pro stahování binárních souborů)
- protokol HTTP textový (ale data mohou chodit i binární), bezstavový
- v požadavcích i odpovědích mohou být tzv HTTP hlavičky, které vše doplňují
- požadavek: specifikace dotazu (GET, POST a další), virtuální host, Referrer, Connection, User-Agent, . . .
- odpověď: typ vráceného dokumentu, jeho délka, použitá komprese, Server, Expires, . . .

CGI

- www server se dá nastavit tak, aby místo načtení statického souboru ze souborového systému spustil program a jeho výstup vrátil klientovi
- CGI – Common Gateway Interface: ”protokol”, pomocí kterého se programu předají potřebné informace a program předá www serveru zpět odpověď (jedná se o domluvené názvy proměnných prostředí)
- CGI programy mohou být napsány v jakémkoliv jazyce (Perl, Python, C, Bash . . .)
- server program při každém požadavku spustí, nastaví proměnné prostředí a klientovi odešle výsledek volání skriptu (výstup ”do konzole”)
- ukázka CGI skriptu

CGI

- CGI skripty jsou pomalé
- řešení – FastCGI: program je spuštěn jako daemon a požadavky mu webový server přesměrovává
- nabízí se otázka: proč by nemohl odpověď generovat přímo webový server
- to je možné – používají se tzv moduly: ISAPI (MS ISS), modules (Apache)
- FastCGI se dnes někdy používá v kombinaci s jazyky Perl, Python, Ruby a některými webovými servery nekompatibilními s ISAPI/Apache moduly

Moduly

- modul je komponenta, která rozšiřuje webový server o nějakou funkčnost
- pro naše účely uvažujeme moduly s podporou dynamického generování obsahu
- modul si můžete napsat (Linux = Apache moduly, Windows = ISAPI moduly)
- spíše ale jen použijeme modul, který přidá podporu pro nějaký již existující programovací jazyk
- PHP – nejrozšířenější (php5isapi.dll pro Windows, mod-php5 pro Apache)
- podobně také jiné skriptovací jazyky: Perl, Python, Ruby,
- modul je třeba asociovat s určitou množinou URL adres (například /mojeaplikace), či skupiny souborů (*.php – modul je naprogramován tak, že soubory čte, interpretuje, vrací výsledky)

PHP

- veleúspěšný skriptovací jazyk pro tvorbu dynamického obsahu
- je možné ho vkládat přímo do HTML stránek, modul PHP soubory načítá, spouští a vrací výsledky
- za úspěchem stojí jednoduchá C-Perl syntaxe a velmi snadné nasazení (zmíněné moduly stačí stáhnout, instalace triviální)
- PHP podporuje velké množství providerů, je snadné se naučit základy
- PHP se ale nehodí na větší aplikace, jazyk prožívá stále bouřlivý vývoj, OOP přidáno násilně, nepodporuje jmenné prostory, nastávají problémy ve vícevláknových prostředích, nekonzistence mezi verzemi (magic quotes, odkazy, třídy)
- to ale neznamená, že se v PHP nedají psát velké aplikace s dobrým návrhem – jen člověk musí do vývoje věnovat více úsilí

Sezení (sessions)

- protokol HTTP je bezstavový, teoreticky po každém požadavku (obrázek, soubor) se spojení ukončí a server ”zapomene”, kdo se ho na to ptal
- pro představu: přestavme si aplikaci, která každému prohlížeči (každému klientovi) vrátí celé číslo z posloupnosti 1, 2, . . .
- k serveru se připojí Jirka a aplikace by měla vypsat 1
- k serveru se připojí milan a aplikace by měla vypsat 2
- k serveru se připojí znovu Jirka a aplikace by měla vypsat jeho 1
- web server ale mezitím dávno zapoměl, že byl k němu nějaký Jirka připojen
- řešení – vytvoření sezení (session)

Sezení (sessions)

- sezení je realizováno obvykle pomocí cookies
- cookie je technologie, která umožňuje webovým serverům (potažmo aplikacím) ukládat na klientovi nějaká data (obvykle malého rozsahu - pár čísel, datum, krátký text)
- webový prohlížeč pak při každém požadavku tyto cookies posílá zpět na server
- z důvodů bezpečnosti má každý server právo zapisovat a číst jen svoje hodnoty (nevidí hodnoty jiných serverů)

Sezení (sessions)

- cookies mohou být také časově omezeny (implicitně do ukončení prohlížeče)
- pro účely realizace sessions se vytvoří cookie (obvykle nazvaná SessionID, SID nebo podobně) – jedinečný identifikátor v rámci celého serveru (velké číslo, nesmí být snadno predikovatelné, aby se nedalo sezení ”ukrást” a pracovat například pod jiným uživatelem/sezením)
- prohlížeč při dalším pohybu po webové aplikaci SID serveru vrací a ten je schopen přesně identifikovat, o jakého klienta jde

Sezení (sessions)

- aplikace si obvykle do sezení musí uložit přihlašovací jméno, pod kterým se uživatel přihlásil, jeho nastavení a podobně
- aby tyto informace neproudily po internetu tam a zpět (protokol HTTP není navíc zabezpečený), tak se tyto informace ukládají na serveru
- obvykle se použije databáze, sdílená paměť nebo soubory
- po síti jako cookie putuje pouze jedno velké číslo (SID)
- jazyk PHP obsahuje funkce pro práci se sezením přímo v sobě, jiné jazyky (Perl, Python) musejí využít nějakou knihovnu (nebo si můžete podporu pro sezení napsat sami)

Jak psát webové aplikace

- u rozsáhlejších aplikací je nutno postupovat rozumně – jinak se můžeme dočkat špatně spravovatelného kódu
- prvním krokem je obvykle volba vhodného jazyka (C/C++, jazyk skriptovací, Java, CSharp)
- dalším krokem výběr platformy (Zend, Turbogears, Django, Ruby On Rails, CakePHP, Symphony PHP, ASP.NET, JSP, JSF, Struts, Tapestry, Turbine-Velocity/WebMacro, Zope)
- výběr databáze je také důležitý (konkurenční přístup, replikace, ovladače pro daný jazyk...)
- velký boom zažívá AJAX (XmlProcedureCall) – pokud jej chceme využít, tak je nutno promyslet, jak bude do celé aplikace zapadat, jakou použijeme knihovnu

Jak psát webové aplikace

- pokud použijete nějaký vkládaný jazyk (PHP, JSP, ASP), není dobré větší aplikaci skládat z vícero samostatných stránek (index.php, login.php, saveEmployee.php)
- aplikace by měla mít jednu vstupní bránu – tzv dispečer (dispatch controller, front controller)
- může to být skript (index.php), nebo nějaká komponenta (Servlet)
- všechny požadavky pak chodí přes jedno místo – je snadné pak přidat například přihlašování, caching, statistiky a podobně

Jak psát webové aplikace

- data (datovou vrstvu) je vhodné oddělit od logiky
- v ideálním případě (pokud je to možné) použít datový návrhový vzor DAO (Data Access Object – objekt, který zapouzdřuje přístup do databáze)
- můžete buď napsat vlastní vrstvu s použitím nízkoúrovňové API daného jazyka (JDBC, Python DB-API . . .), nebo využít O-R mapovací knihovnu (SQLObject pro Python, JDO, EJB, TopLink, Hibernate, iBatis pro Javu
- dnes je čas zvážit i (čistě) objektové databáze (ZopeDB, db4o, Caché)
- logika by měla stát mimo databázovou vrstvu

Jak psát webové aplikace

- prezentační část by měla být také oddělena – šablony (HTML, XHTML, CSS) obvykle píší designéři, kteří neprogramují (a opačně – programátoři zase nedělají design)
- designéři navíc používají vlastní nástroje pro tvorbu grafiky a designu stránek (Macromedia, Adobe atd)
- prezentační část musí jít snadno vyměnit
- proto je vhodné nemíchat tyto tři věci: data (např. nákupní košík, položka), logiku (vložit do košíku, zaplatit) a prezentaci (obrazovka s nákupním košíkem, uvítací obrazovka)
- dostáváme se na návrhový vzor MODEL-VIEW-CONTROLLER

MVC

- Model-View-Controller návrh bude pravděpodobně vyhovovat většině aplikací
- MODEL (data), VIEW (šablona), CONTROLLER (rozdělovač, dispečer, ”vstupní brána” aplikace) + front controller
- přijde požadavek, předsunutý controller rozhodne na základě adresy URL, jakou zavolá akci/objekt/metodu (podle frameworku)
- běh je předán akci, ta provede danou logiku s modelem, obvykle načte nějaká výsledná data a ta jsou předána šabloně
- dělení podle toho, zda se data do šablon ”vtlačují” (push) nebo ”vytahují” (pull) – Pull MVC, Push MVC

Typy webových frameworků

- leckotonážní – Ruby On Rails, CakePHP, Symphony, Django, Turbogears, Zope, Snails
- střednětonážní – Webware, Struts, JSP/JSF, Tapestry, Turbine/Velocity, Mono ASP.NET (ASP.NET na Linuxu – zatím v plenkách)
- obchodní (enterprise) webové aplikace jsou v drtivé většině psány v technologiích J2EE (vícevrstvé aplikace), eventuelně různých napodobeninách (Spring Framework) či pomocí technologie ASP.NET 2.0
- netradiční přístup – Zope, Apache Cocoon
- dneska je hitem hlavně Java, Python/Ruby, PHP a ASP.NET

Webové programování v Javě

- co budeme potřebovat? JDK (Java Development Kit) 5.0, vývojové prostředí (dnes populární Netbeans IDE, Eclipse IDE nebo komerční IntelliJ IDEA)
- java.sun.com
- webový server (kontejner) – Apache Tomcat, Jetty, Resin
- jakarta.apache.org
- pokud budeme chtít psát aplikaci v nestandardním frameworku, tak knihovny

Webové programování v Javě

- proč právě Java pro webové programování?
- vysoký výkon
- přenositelnost
- snadné nasazení
- stabilita a robustnost
- komerční podpora mnoha firem (né jedné)

Webové programování v Javě

- pojem Servlet, JSP/JSF
- možno psát jednoduché JSP skripty i robustní MVC aplikace
- ukázka: Netbeans IDE a jednoduchý JSP projekt
- ukázka: Netbeans IDE s VWP a JSF projekt

FOSS DB systémy

Relační databázové systémy:

- PostgreSQL
- MySQL
- Firebird
- SQLite
- Apache Derby

FOSS ODB systémy

Objektové a ostatní db systémy:

- db4o (Java, .NET)
- Oracle BerekeleyDB (klíč-hodnota databáze, nativní a Java verze)
- ZopeDB (Python)

Shrnutí

- tvorba webových aplikací dnes: PHP, Python, Java, ASP.NET
- pokud zbude čas – prezentace Java pro vývojáře .NETu
- diskuse