

Serverové skriptovací technologie

4IZ228 – tvorba webových stránek a aplikací

Jirka Kosek

Poslední modifikace: \$Date: 2011/11/10 17:41:58 \$

Copyright © 2000-2011 Jiří Kosek

Obsah

Úvod	3
Základní principy generování stránek na serveru	4
Interakce s webovou aplikací na serveru	5
Nástroje pro dynamické generování HTML stránek	6
SSI	7
Server Side Includes	8
Ukázka	9
Přehled příkazů	10
CGI skripty	11
Rozhraní CGI	12
Předávání parametrů přes rozhraní CGI	13
Další informace předávané pomocí CGI	14
Předání výstupu skriptu zpět web-serveru	16
Ukázky	17
Ukázky	18
Shrnutí	21
FastCGI	22
FastCGI	23
Použití FastCGI	24
SAPI	25
ISAPI, NSAPI, WSAPI,	26
ASP	27
Active Server Pages	28
Možnosti ASP	29
Ukázka	30
Ukázka	31
PHP	32
Hypertextový preprocesor PHP	33
Ukázka	34
Ukázka	35
Java a webové aplikace	36
Java servlety	37
Java Server Pages	38
Ukázka JSP	39
Ukázka JSP	40
ASP.NET	41
.NET	42
ASP.NET	43
Srovnání technologií	44
Přístupy k návrhu aplikací	45
Rychlost provádění aplikací	46
Rychlost vývoje aplikací	47
Další zdroje informací	48
Další zdroje informací	49

Úvod

Základní principy generování stránek na serveru	4
Interakce s webovou aplikací na serveru	5
Nástroje pro dynamické generování HTML stránek	6

Základní principy generování stránek na serveru

- na serveru je dynamicky generováno HTML na základě požadavku uživatele
- do prohlížeče je odeslán již jen čistý HTML kód
- není potřeba žádný speciální prohlížeč, lze použít libovolný se základní podporou HTML
- v případě potřeby lze na serverem generovaných stránkách použít i klientské technologie (např. JavaScript)

Interakce s webovou aplikací na serveru

- lze použít vše, co vyvolá HTTP požadavek na webový server
 - HTML formuláře
 - odkazy
 - automatické otevření nové stránky pomocí JavaScriptu
- obsah stránky se nemusí řídit jen požadavky uživatele, ale může záviset i na externích vstupech (čas) – např. graf vývoje burzovního indexu

Nástroje pro dynamické generování HTML stránek

- Server Side Includes (SSI)
- CGI skripty
- FastCGI skripty
- SAPI moduly a filtry
- Active Server Pages (ASP)
- PHP
- servlety
- Java Server Pages
- ASP.NET
- Ruby on Rails
- Django (Python)
- ... a mnoho dalších, o nichž se ani nezmíníme

SSI

Server Side Includes	8
Ukázka	9
Přehled příkazů	10

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`)
- syntaxe:

```
<!--#příkaz parametry-->
```

Ukázka

Příklad 1. Vypsání aktuálního času

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>První pokusný skript</title></head>
<body>
<h1>Aktuální čas: <!--#echo var="DATE_LOCAL"--></h1>
</body>
</html>
```

Přehled příkazů

#include

načtení externího souboru

#filesize

zjištění velikosti souboru

#lastmod

zjištění času poslední modifikace souboru

#echo

vypsání obsahu proměnné – DATE_GMT, DATE_LOCAL,
DOCUMENT_NAME, DOCUMENT_URI, LAST_MODIFIED,
QUERY_STRING_UNESCAPED

#exec

spuštění externího programu

#config

nastavení formátu výstupu ostatních příkazů

CGI skripty

Rozhraní CGI	12
Předávání parametrů přes rozhraní CGI	13
Další informace předávané pomocí CGI	14
Předání výstupu skriptu zpět web-serveru	16
Ukázky	17
Ukázky	18
Shrnutí	21

Rozhraní CGI

- CGI – Common Gateway Interface
- rozhraní definuje způsob komunikace web-serveru s aplikací
- CGI skript je program, který používá rozhraní CGI
- CGI skripty lze psát v téměř libovolném jazyce, stačí dodržet konvence rozhraní CGI
 - shell, Perl, C/C++, Pascal, Python, ...
- podpora CGI nebývá implicitní, musí se ve web-serveru zapnout (bezpečnost)

Předávání parametrů přes rozhraní CGI

- existují dvě metody – GET a POST

- způsob je určen přímo v HTML formuláři

```
<form ... method="post">
```

```
<form ... method="get">
```

- standardní je metoda GET
- před odesláním prohlížeč všechna data z formuláře zakóduje do jednoho dlouhého řetězce
 - název1=hodnota1&název1=hodnota2&...
 - hodnoty polí jsou upraveny tak, aby je šlo zapsat jako součást URL
 - mezera → +
speciální znaky, znaky s diakritikou apod. → %xx, kde xx je kód znaku v šestnáctkové soustavě
- při metodě GET jsou zakódovaná data přidána za URL požadavku (za znak ?)
 - rozhraní CGI předá skriptu data v proměnné prostředí `QUERY_STRING`
- při metodě POST jsou data předávána v těle HTTP požadavku
 - CGI skript je dostane na svůj standardní vstup

Další informace předávané pomocí CGI

- kromě samotných dat z formuláře, předá web-server i další užitečné údaje pomocí proměnných prostředí:

REQUEST_METHOD

určuje způsob předávání informací – GET nebo POST

QUERY_STRING

obsahuje data přenášená metodou GET

PATH_INFO

cesta, která má být zpracována skriptem; nejčastěji jde o část cesty v URL za jménem skriptu

PATH_TRANSLATED

cesta ke stejnému souboru jako `PATH_INFO`; v tomto případě však byla cesta přemapována podle konfigurace serveru

CONTENT_TYPE

MIME typ dat zasílaných metodou POST

CONTENT_LENGTH

délka dat zasílaných metodou POST

SCRIPT_NAME

URL právě prováděného skriptu

SERVER_NAME

jméno serveru

SERVER_PORT

číslo portu

SERVER_SOFTWARE

jméno a verze programu pracujícího jako WWW-server

SERVER_PROTOCOL

jméno a verze protokolu, kterým přišel požadavek (typicky HTTP/1.0 nebo HTTP/1.1)

GATEWAY_INTERFACE

označení a verze použitého rozhraní ke spuštění skriptu (typicky CGI/1.1)

REMOTE_HOST

doménová adresa počítače, z něž přišel požadavek

REMOTE_ADDR

IP-adresa počítače, z něž přišel požadavek

AUTH_TYPE

způsob použité autentifikace uživatele

Další informace předávané pomocí CGI (Pokračování)

REMOTE_USER

v případě, že byl uživatel autentifikován, obsahuje tato proměnná jeho jméno

Předání výstupu skriptu zpět web-serveru

- veškeré informace se předávají přes standardní výstup
 - nejdříve se posílají HTTP hlavičky
 - pak prázdný řádek
 - a nakonec samotná odpověď – typicky HTML kód
- web-server odpověď zachytí, doplní do ní chybějící hlavičky a pošle klientovi
- vždy musíme vygenerovat alespoň hlavičku `Content-Type`, která určuje druh odesílaných dat (nejčastěji `text/html`)

Ukázky

Příklad 2. Vypsání aktuálního času v C

```
#include <stdio.h>
#include <time.h>

int main()
{
    struct tm *aktualni_cas;
    time_t aktualni_sekundy;
    char s[80];

    printf("Content-type: text/html\n\n");
    printf("<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<html>
<head><title>První pokusný skript</title></head>
<body>
<h1>Aktuální čas: ");
    time(&aktualni_sekundy);
    aktualni_cas = localtime(&aktualni_sekundy);
    strftime(s, 80, "%d.%l.%Y %H:%M:%S", aktualni_cas);
    printf("%s", s);
    printf("</h1>
</body>
</html>");
    return 0;
}
```

Ukázky

Příklad 3. Jednoduchý formulář v HTML

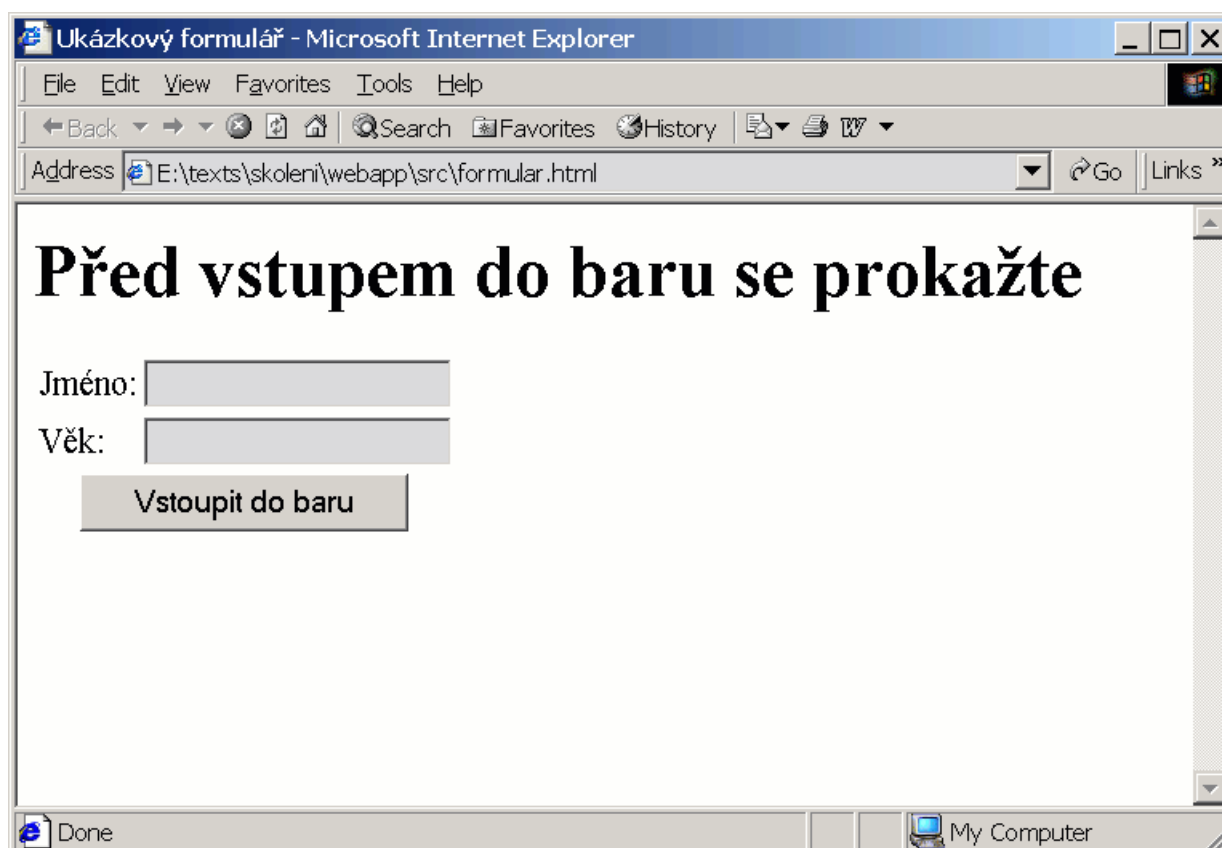
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Ukázkový formulář</title>
</head>
<body>

<h1>Před vstupem do baru se prokažte</h1>

<form action="obsluha.pl">
<table>
<tr>
<td>Jméno:</td>
<td><input name="jmeno"></td></tr>
<tr>
<td>Věk:</td>
<td><input name="vek"></td></tr>
<tr>
<td colspan="2" align="center"><input type="submit" value="Vstoupit do ►
baru"></td>
</tr>
</table>
</form>

</body>
</html>
```

Ukázky (Pokračování)



Příklad 4. Obsluha formuláře v Perlu

```
#!/usr/bin/perl
use CGI;

print "Content-type: text/html\n\n";
print <<EOF
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Obsluha formuláře</title>
</head>
<body>
EOF
;

$query = new CGI;
print "Jmenuješ se <em>", $query->param('jmeno'), "</em><br>";
if ($query->param('vek') < 18)
{
    print "a jseš moc mladý na náš bar";
}
```

Ukázky (Pokračování)

```
else
{
    print "a jsme rádi, že jsi přišel do našeho baru"
}

print <<EOF
</body>
</html>
EOF
```

Při psaní klasických CGI skriptů většinou používáme různé knihovny, které umějí dekodovat data z formuláře.

Shrnutí

- výhody:
 - pro psaní skriptů lze použít téměř libovolný jazyk
 - vývojář se nemusí učit nový jazyk
- nevýhody
 - pro obsluhu každého požadavku je spouštěn nový proces
 - pomalé a náročné na zdroje serveru
 - na více zatížených serverech nelze vůbec použít

FastCGI

FastCGI	23
Použití FastCGI	24

FastCGI

- vylepšená varianta rozhraní CGI, 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
 - primitivní řešení load-balancingu

Použití FastCGI

- na rozdíl od CGI, nepodporují FastCGI zdaleka všechny servery
- aplikace musí používat speciální knihovnu, která implementuje rozhraní FastCGI
 - C, Perl, ...
- ukázka

```
use FCGI;
```

```
while(FCGI::accept() >= 0) # čekání na požadavek
{
    # obsluha požadavku - stejná jako v případě CGI verze
}
```

- skript je v paměti vykonáván opakovaně, musíme dávat velký pozor na přetečení paměti apod.
- ve skriptu můžeme používat vlastní čítač, a po určitém počtu obslužených požadavků skript ukončit, web-server si ho při dalším požadavku sám znovu spustí

SAPI

ISAPI, NSAPI, WSAPI,	26
-----------------------------	----

ISAPI, NSAPI, WSAPI, ...

- v průběhu času začala většina serverů nabízet kromě CGI rozhraní i speciálně přizpůsobené rozhraní
- dnes nejpoužívanější je ISAPI – podporují ho servery Microsoftu a mnohé další
- aplikace napsané pro SAPI mají většinou podobu DLL knihoven
- do paměti se podobně jako FastCGI skripty načtou při prvním požadavku a pak v ní již zůstanou
- nelze rozdělit aplikaci a web-server
- SAPI moduly jsou binární nativní kód – pro tvorbu si musíme sehnat vhodný kompilátor

ASP

Active Server Pages	28
Možnosti ASP	29
Ukázka	30
Ukázka	31

Active Server Pages

- přímo do HTML kódu se zapisují jednoduché příkazy
- ASP je jen jakýsi framework
 - lze použít libovolný jazyk podporující Active Scripting
 - standardně JScript a VBScript
 - třetí firmy dodávají Perl, REXX, Python
 - ve všech 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

Možnosti ASP

- k dispozici máme všechny funkce zvoleného jazyka (bohužel VBScript a JScript jsou poměrně chudé jazyky)
- sada ASP objektů pro práci s
 - požadavkem – data z formulářů apod.
 - odpovědí – nastavování hlaviček
 - další pomocné objekty – aplikační a session proměnné, ...
- chybějící funkčnost se dodává pomocí COM objektů
 - rychlé – píše se přímo v nativním kódu
 - instalace a správa aplikace není jednoduchá, protože je roztroušená na mnoha místech

Ukázka

Příklad 5. Vypsání aktuálního času v ASP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>První pokusný skript</title></head>
<body>
<h1>Aktuální čas: <%= Now() %></h1>
</body>
</html>
```

- <% ... %> – blok příkazů
- <%= výraz %> – vypsání hodnoty výrazu přímo do stránky

Ukázka

Příklad 6. Obsluha dat z formuláře

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Ukázkový formulář</title>
</head>
<body>
Jmenuješ se <em><%= Request("jmeno")%></em><br>
<%
If Request("vek") < 18 Then
    Response.Write "a jseš moc mladý na náš bar"
Else
    Response.Write "a jsme rádi, že jsi přišel do našeho baru"
End If
%>
</body>
</html>
```

PHP

Hypertextový preprocesor PHP	33
Ukážka	34
Ukážka	35

Hypertextový preprocesor PHP

- 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
- OSS – dostupný zdarma včetně zdrojových kódů

Ukázka

Příklad 7. Vypsání aktuálního času

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>První pokusný skript</title></head>
<body>
<h1>Aktuální čas: <?php echo Date("r")?></h1>
</body>
</html>
```

- pro oddělování příkazů od HTML kódu se používají znaky <? a ?>

Ukázka

Příklad 8. Obsluha dat z formuláře

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Ukázkový formulář</title>
</head>
<body>
Jmenuješ se <em><?php echo $_REQUEST["jmeno"]?></em><br>
<?php
if ($_REQUEST["vek"] < 18)
{
    echo "a jseš moc mladý na náš bar";
}
else
{
    echo "a jsme rádi, že jsi přišel do našeho baru";
}
?>
</body>
</html>
```

Java a webové aplikace

Java servlety	37
Java Server Pages	38
Ukázka JSP	39
Ukázka JSP	40

Java servlety

- servlet je speciální třída zapsaná v jazyce Java
- web-server v sobě spustí JVM a v ní pak běží servlet
- podobně jako u ISAPI a FastCGI zůstává servlet po prvním načtení v paměti a obsluhuje další požadavky

Java Server Pages

- do HTML kódu se zapisují příkazy Javy
- k dispozici jsou podobně jako v ASP speciální objekty pro čtení dat z formulářů apod.
- pro lepší oddělení designu a logiky lze definovat „tag libraries“ – uživatelsky definované tagy, které volají předem připravené komponenty
- o spuštění JSP se stará servlet, který JSP automaticky převede do Javy, zkompiluje do byte-code a spustí

Ukázka JSP

Příklad 9. Vypsání aktuálního času

```
<%@ page language="java" import="java.text.*, java.util.*" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>První pokusný skript</title></head>
<body>
<h1>Aktuální čas: <%= new Date() %></h1>
</body>
</html>
```

Ukázka JSP

Příklad 10. Obsluha dat z formuláře

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Ukázkový formulář</title>
</head>
<body>
Jmenuješ se <em><%= request.getParameter("jmeno")%></em><br>
<% if (Integer.parseInt(request.getParameter("vek")) < 18) { %>
    a jseš moc mladý na náš bar
<% } else { %>
    a jsme rádi, že jsi přišel do našeho baru
<% } %>
</body>
</html>
```

ASP.NET

.NET	42
ASP.NET	43

.NET

- platforma Microsoftu s podobnými principy jako platforma Java
- aplikace se zdrojových kódů překládá do CIL (Common Intermediate Language) – obdoba javového bytecode
- o spouštění CIL se stará CLR (Common Language Runtime)
 - před spuštěním je vždy CIL převeden do nativního kódu (obdoba JIT kompilace v Javě)
 - Microsoft nabízí CLR pro Windows; existují i run-time pro další systémy (např. Mono)
 - existuje několik projektů, jejichž cílem je vytvoření CLR pro další platformy (např. Mono pro Linux)
- všechny jazyky, které lze kompilovat do CIL (VB.NET, Managed C++, C#, ...) používají stejné knihovny (velká změna oproti předchozím verzím jazyků)
 - výborná podpora XML
 - hlavní tři knihovny – webové služby, Web Forms (tvorba webových aplikací), Windows Forms (tvorba „klasických“ aplikací)

ASP.NET

- s klasickými ASP nemá skoro nic společného
- vyvíjí se jako klasická klientská aplikace – prvky uživatelského rozhraní a obsluha událostí
- ASP.NET si webový server přeloží do nativního kódu, který se stará o postupné zasílání HTML kódu a obsluhu formulářových dat
- vygenerovaný kód detekuje použitý prohlížeč a tomu přizpůsobí generovaný HTML a JavaScriptový kód
- VisualStudio.NET umožňuje aplikace vyvinout pouhým „naklikáním“
- později byly pro ASP.NET vytvořeny další nastavby – např. ASP.NET MVC nebo Razor

Srovnání technologií

Přístupy k návrhu aplikací	45
Rychlost provádění aplikací	46
Rychlost vývoje aplikací	47

Přístupy k návrhu aplikací

- „špagety“
 - HTML kód je promíchán s aplikační logikou (příkazy)
 - nepřehledné a neudržovatelné; zvláště pro větší projekty
 - např. PHP, ASP, JSP
- Model-View-Controller (MVC)
 - je oddělena aplikační logika (model), generování výstupů pro uživatele (view) a průběh interakce (controller)
 - velice čistý přístup, aplikace se lépe udržuje
 - oddělené M-V-C znamená více práce a kódu
 - např. J2EE, ASP.NET MVC, PHP s vhodným frameworkem
- komponentové frameworky
 - aplikace se skládá z vizuálních komponent, které na pozadí generují odpovídající HTML (+JS) kód
 - vývojář je odstíněn od webové platformy (HTML, JS, HTTP, ...)
 - např. ASP.NET, JSF
- „moderní“ frameworky
 - většinou staví na myšlence MVC, ale nenutí vývojáře psát a definovat věci, které jsou zřejmé
 - např. Ruby on Rails, Django

Rychlost provádění aplikací

- kompilované jazyky – velmi rychlé (pokud se nepoužije CGI)
 - C, C++, Pascal, Java
 - FastCGI, ISAPI, servlety
- interpretované jazyky – jsou pomalejší
 - Perl, ASP, PHP
 - většina aplikací je jednoduchá a zdržuje je práce s databází – menší výkon většinou nevádí
 - pro některé jazyky existují kompilátory (ASP.NET)
 - rychlost lze zvýšit udržováním předkompilovaných skriptů v paměti web-serveru (např. Zend Accelerator pro PHP)

Rychlost vývoje aplikací

- kompilované jazyky – pomalá
 - po provedení každé změny je potřeba program rekompilovat (pracné a pomalé)
- interpretované
 - rychlé změny – stačí opravit zdrojový kód a dát v prohlížeči reload
- rychlý běh aplikací a rychlý vývoj zároveň → JSP, ASP.NET, ...
 - programátor pracuje pouze se zdrojovým kódem skriptu
 - o kompilaci se automaticky stará webový server nebo jeho modul

Další zdroje informací

Další zdroje informací	49
------------------------------	----

Další zdroje informací

- CGI¹
- FastCGI²
- příklady servletů a JSP³
- Perl⁴
- JSP⁵
- servlety⁶
- PHP⁷
- Zend – nové jádro jazyka použitého v PHP⁸
- ASP.NET⁹
- Ruby on Rails¹⁰
- Django¹¹
- JSF¹²
- seriál o Node.js¹³
- úvod do Ruby on Rails¹⁴
- seriál o Django¹⁵

¹ <http://hoohoo.ncsa.uiuc.edu/cgi/interface.html>

² <http://www.fastcgi.com/>

³ <http://www.archive.coreservlets.com/>

⁴ <http://www.perl.com/>

⁵ <http://java.sun.com/products/jsp/>

⁶ <http://java.sun.com/products/servlet/>

⁷ <http://www.php.net/>

⁸ <http://www.zend.com/>

⁹ <http://asp.net/>

¹⁰ <http://www.rubyonrails.org/>

¹¹ <http://www.djangoproject.com/>

¹² <http://java.sun.com/javaee/javaxserverfaces/>

¹³ <http://zdrojak.root.cz/clanky/javascript-na-serveru-zaciname-s-node-js/>

¹⁴ <http://guides.rubyonrails.cz/>

¹⁵ <http://zdrojak.root.cz/clanky/django-uvod-a-instalace/>