MASARYKOVA UNIVERZITA V BRNĚ
FAKULTA INFORMATIKY

Realizace formulářových operací a komunikace mobilních zařízení s CGI skripty

BAKALÁŘSKÁ PRÁCE

Juraj Hudák

Brno, podzim 2004

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Ing. Michal Brandejs CSc.

Shrnutí

Informační systémy založené na bázi internetových portálů obsahují propracovaná rozhraní založená na základě známého značkovacího jazyka HTML. Ten však svou volnou strukturou nevyhovuje možnostem mobilních zařízení (zejména mobilních telefonů), která jsou omezena kapacitou dostupné paměti, výkonem mikroprocesorů a přenosovou rychlostí přístupu ke službám sítě Internet.

Cílem této práce je poskytnout ucelený pohled na problematiku komunikace mobilních zařízení a skriptů umístěných na webovém serveru. Popsal jsem zde základní principy komunikace s CGI skripty a omezení mobilních zařízení. Dále uvádím množství příkladů v programovacím jazyce Perl a ve značkovacím jazyce WML. Ty tvoří základ řešení vzájemné komunikace a je pomocí nich vytvořena jednoduchá ukázková aplikace. Práce poskytuje návod, jak vytvářet stránky a formuláře ekvivalentní se značkovacím jazykem HTML.

Klíčová slova

CGI, HTML, HTTP, Perl, WAP, WML

Obsah

Kapitola 1

Úvod

Již velmi dlouhou dobu si lidé vyměňují nejrůznější informace. Nejdříve se informace rozšiřovaly pouze velmi pomalu, cestovatelé a obchodníci se vzdálenými kraji byli jediným zdrojem informací o dění na různých místech. Objevem papíru bylo umožněno předávat kvalitnější informace — nezkreslené pamětí osob — a také uchovávat nejrůznější informace pro pozdější dobu. Jedním z podstatných objevů je však možnost přenášet informace pomocí elektrických impulsů — pomocí telegrafu. Pomocí telegrafu bylo možné přenášet informace na relativně velmi vzdálená místa s malým časovým zpožděním, např. informace o burzovních operacích, dění na druhé straně kontinentu, oceánu ...

Zlomovým se jeví vynález přenosu lidského hlasu pomocí signálu šířeného pomocí metalického vedení, tedy vynález telefonního přístroje. Od doby vynálezu telefonu prošlo celé odvětví bouřlivým rozvojem. Informace jsou nyní předávány velkou rychlostí a ve velkých objemech na prakticky neomezenou vzdálenost. Díky satelitům je v současnosti pokryt celý povrch naší planety signálem pro satelitní telefonní přístroje (Inmarsat, Iridium, ...) nebo pomocí pozemních vysílačů signálem GSM. S příchodem tzv. *informačního věku*, jak je tato doba nazývána, se výměna informací někdy až neuvěřitelným způsobem urychlila. Tento vývoj je umožněn právě pokrokem v oblasti komunikací a informačních technologií.

Právě mobilní zařízení se v posledních letech stávají velmi důležitým informačním médiem. Uživatelé mohou být osloveni kdykoliv se nacházejí v dosahu některého vysílače. Velmi lákavá je možnost dopravit různé aplikace přímo k uživateli bez různých omezení, např. lokalitu nebo nutnost mít k dispozici osobní počítač. Uživatelé tak již dnes mohou prohlížet obsah internetových stránek ve svých mobilních telefonech nebo osobních organizerech na nejrůznějších místech — na vycházce v přírodě nebo ve své oblíbené restauraci. Možnostmi a problémy se zabývá tato práce. Na příkladech budou jednotlivé případy popsány tak, aby přechod ke tvorbě stránek pro mobilní telefony nebyl obtížný.

Kapitola 2

CGI

CGI, neboli Common Gateway Interface, je rozhraní umožňující spouštění aplikací na vzdáleném serveru, předávání uživatelských dat těmto programům a zobrazení jejich výstupu. Programy mají tedy možnost reagovat na uživatelské požadavky a zobrazovat výstupy různých výpočtů či výsledků hledání. K zasílání požadavků je používán protokol HTTP (Hypertext Transfer Protocol). Nejznámější aplikací jsou zřejmě nejrůznější webové stránky v jazyce HTML (Hypertext Markup Language) a informační systémy, např. Informační Systém Masarykovy univerzity. Seznámíme se zde také s programovacím jazykem Perl, který je vhodný pro tvorbu CGI aplikací a rozsáhlých projektů.

Popis CGI rozhraní je možné najít na mnoha stránkách v síti Internet, některé jsou uvedeny v [CGI]. Zde je také možné zjistit v jaké fázi se nachází novější verze CGI. V současné době se používá rozhraní verze 1.1, další stupeň vývoje označuje verze 1.2.

2.1 HTTP

Základním protokolem umožňující komunikaci webových serverů a jejich klientů je protokol HTTP. Pokud existuje možnost, aby (mobilní) zařízení komunikovalo pomocí tohoto protokolu, potom s velkou pravděpodobností bude možné přistupovat do sítě Internet. Příkladem je protokol WAP použitý u mobilních telefonů.

Protokol HTTP je aplikační protokol určený k přístupu a výměně informací v prostředí distribuovaných informačních systémů. Dnes tento protokol tvoří základ sítě Internet. Původní verze protokolu byla (zpětně) označena jako *HTTP/0.9*, v současnosti se používají verze *HTTP/1.0* a *HTTP/1.1*.

Všechny verze protokolu je možné použít i pro nepřímou komunikaci s webovými servery — přes brány nebo proxy servery. Potom platí, že klient musí obdržet odpověď ve stejné verzi protokolu, kterou byl odeslán požadavek. Stejně tak nesmí server nic předpokládat o možnostech klienta a musí odpovídat stejnou verzí.

2.1.1 HTTP verze 0.9

Nyní si představíme základní a společné vlastnosti protokolu HTTP. Jedná se o bezstavový protokol pracující v režimu *dotaz/odpověď*. Pro správnou funkci je požadováno použití spolehlivé transportní služby — libovolného transportního protokolu, zde TCP. K adresování informačních zdrojů je použito schéma označované jako *URL* (Uniform Resource Locator). Pomocí URL jsme schopni adresovat nejen server ale i dokument, přístupový protokol, atd. Formát URL pro HTTP je následovný: `http://host[:port]/path`. `host` určuje adresu vzdáleného stroje, volitelné `port` port, na kterém naslouchá webový server (standardně 80) a `path` je absolutní cesta k požadovanému dokumentu. Odesílání požadavků se děje vytvořením TCP spojení a posláním řádku s požadavkem: `GET URL<CR><LF>`. Odpovědí je požadovaný soubor.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /
<html>
...
</html>
```

Server po obdržení HTTP požadavku verze 0.9 nečeká na žádné další informace od klienta a ihned zasílá odpověď, bez identifikace obsahu.

Příklad 2.1.1: HTTP 0.9

2.1.2 HTTP verze 1.0

Protokol HTTP/0.9 byl nahrazen komplexnější verzí — HTTP/1.0. Ta je již univerzálnější, umožňuje přenášet různá data a identifikovat jejich obsah. Byl také pozměněn formát adresy. URL bylo nahrazeno URI, Uniform Resource Identificator, RFC 1738, 1808, 2396. Změnil se také tvar požadavku, protože přibyla nutnost rozeznávat různé verze protokolu a také typ požadavku: `METHOD URL HTTP/1.0<CR><LF>`. `METHOD` zde zastupuje např. *GET*, *POST* nebo *HEAD*. Na dalších řádcích je potom možné připojit k požadavku doplňující hlavičky. Hlavičky jsou velmi podobné hlavičkám používaných v MIME (RFC 1521). Můžeme tak např. požadovat stránku v kódování ISO-8859-2, pokud je k dispozici dostane přednost: `Accept-Charset: iso-8859-2<CR><LF>`.

Naopak odpověď serveru *musí* jako první řádek poslat verzi protokolu doplněnou o kód odpovědi a dále obsahovat hlavičku identifikující obsah. Podrobný a přesný popis lze nalézt přímo v dokumentu RFC 1945.

Z řádku č. 7 ukázky 2.1.2 je vidět, že verze HTTP odpovědi je vyšší než požadavku. To však není velký problém, protože neznámé položkou jsou klientem ignorovány.

2.1.3 HTTP verze 1.1

Tato verze rozšiřuje předchozí o některé zajímavé prvky. Mezi hlavní patří vylepšené použití dočasných pamětí, perzistentní spojení nebo identifikace cíle. Lepší řízení dočasných pamětí má dovolit tvorbu spolehlivějších aplikací, paměti se řídí pomocí hlavičky `Cache-Control`. Vylepšení spočívá také v použití `Age`, zde se počítá celková doba, kterou objekt přebýval v dočasné paměti a kterou cestoval po linkách mezi uzly. Zvýšení výkonu a snížení zátěže HTTP serverů by mělo být dosaženo použitím trvalých spojení, jedná se o možnost zaslat serveru více požadavků během jediného spojení. Pomocí `Connection: close` docílíme původního chování — jedno spojení, jeden požadavek. Dotaz prohlížeče Mozilla v HTTP 1.1 je možné vidět na příkladu 2.1.3.

RFC dokument pojednávající o tomto protokolu nese číslo 2616 a je opět dostupný z ftp serveru Fakulty Informatiky, [RFCs].

2.2 Common Gateway Interface

Jak již bylo řečeno, jedná se o obecné rozhraní mezi serverem a klientskou aplikací. Zde je definováno, jakým způsobem mají být posílány požadavky serveru, jak se předávají proměnné a jejich hodnoty, jaký jazyk preferuje prohlížeč, jak předávat dodatečné informace (*hlavičky*, *cookies*), atd. Také předepisuje mechanismus předávání uživatelských dat jiným programům, které běží na serveru. Naopak je zde zakotveno, jak musí vypadat odpověď vytvořená spouš-

```
1  $ telnet localhost 80
2  Trying 127.0.0.1...
3  Connected to localhost.
4  Escape character is '^]'.
5  GET / HTTP/1.0
6
7  HTTP/1.1 200 OK
8  Date: Sat, 15 May 2004 18:42:04 GMT
9  Server: Apache/1.3.27 (Unix)
10 Last-Modified: Thu, 11 Mar 2004 12:36:02 GMT
11 ETag: "1039a-217-40505d32"
12 Accept-Ranges: bytes
13 Content-Length: 535
14 Connection: close
15 Content-Type: text/html
16
17 <html>
18 ...
19 </html>
20 Connection closed by foreign host.
```

Ukázka požadavku GET na kořenový dokument webového serveru. Řádky 5, 7 a 15 odpovídají požadavku, verzi HTTP protokolu a kód odpovědi a identifikaci obsahu zasílaného dokumentu.

Příklad 2.1.2: HTTP 1.0

těnými programy (binární programy, skripty), odpověď aplikace předá webovému serveru a ten ji pošle klientské straně. Dodatečné informace se předávají jako hlavičky ve tvaru `Název-hlavičky: hodnoty`.

Základním požadavkem na klientskou stranu je, aby první řádek určoval použitou metodu (GET, POST, HEAD), URL (adresu dokumentu) a případně identifikaci použité verze protokolu HTTP — 1.0, 1.1 nebo nic v případě verze 0.9. Posílají-li se hlavičky, musí být dotaz ukončen prázdným řádkem. Naopak odpověď serveru musí začínat stavovým řádkem s označením protokolu HTTP a stavu, zda byl požadavek přijat a správně zpracován. Zejména musí cgi-aplikace poslat hlavičku s identifikací obsahu výsledku — např. `Content-type: text/html` — tato hlavička je povinná.

Na tvorbu aplikací na straně serveru si můžeme zvolit nejrůznější programovací jazyky. Musíme však zvážit některé aspekty spojené s programováním takových to aplikací. V [Gund98] jsou na str. 10 zmíněny následující tři, k nimž je však vhodné doplnit ještě jeden.

- Snadnost manipulace s textem.
- Možnost spolupráce s dalšími softwarovými knihovnami a utilitami.
- Možnost přístupu k systémovým proměnným.
- Bezpečnost aplikací.

Snadno se nahlédne praktičnost všech uvedených bodů. Všechny předávané parametry jsou ve tvaru řetězců nějaké abecedy a kódování (ASCII, ISO Latin 2, UTF-8, ...), proto je vhodné mít efektivní nástroje umožňující práci s řetězci. Často budeme požadovat přístup k databázovým serverům nebo tvorbu obrázků, např. tvorba grafů zátěže apod. K tomu je možné použít již existující program nebo knihovny. Pomocí systémových proměnných jsou aplikacím předávány důležité informace o požadavcích (požadovaný jazyk, rozpoznatelné dokumenty). A

```
$ telnet www.google.com 80
Trying 216.239.59.99...
Connected to www.google.com.
Escape character is '^]'.
GET http://www.google.com/ HTTP/1.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;
Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7
Accept-Encoding: gzip,deflate
Accept-Language: cs,en-us;q=0.7,en;q=0.3
Host: www.google.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; rv:1.7.3) Gecko/20040914
Keep-Alive: 300
Proxy-Connection: keep-alive

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html
Content-Encoding: gzip
Server: GWS/2.1
Content-Length: 1163
Date: Sun, 07 Nov 2004 17:57:10 GMT
```

Hlavičky `Accept` a `User-Agent` jsou zkráceny na šířku stránky.

Příklad 2.1.3: HTTP 1.1

zcela určitě chceme, aby naše aplikace běžela v bezpečném režimu — nedůvěřujeme hodnotám, které k nám přicházejí z formulářů, ani v případě jejich kontroly na straně klienta, např. pomocí JavaScriptu.

Na tomto místě uveďme některé oblíbené a používané programovací jazyky. Své jisté místo zde má jazyk C a jeho příbuzný C++, dále je možné použít různé interprety příkazů (sh, bash, csh) a zcela jistě je možné použít tzv. interpretované jazyky — Perl, Python nebo PHP. Tento výčet rozhodně není a ani nemůže být úplný, ale v Internetu se pravděpodobně setkáme ve velké míře právě s těmito jazyky. My se zde však budeme zbývat pouze jazykem Perl.

2.2.1 GET

Pomocí metody *GET* předáváme parametry nejjednodušším způsobem — jako součást URL. Oddělovačem dvojice název proměnné a její hodnota je znak rovnítko ('='), jednotlivé dvojice se potom oddělují pomocí znaku ampersand ('&'), resp. pomocí entity &. Ke jménu skriptu se parametry připojují otazníkem ('?'). Výsledné URL má potom tvar `http://host/path/script.cgi?var1=value1&var2=value2`. Nevýhodou tohoto formátu je skutečnost, že se toto URL objeví v záznamu o činnosti webového a případně i proxy serveru. Ukázku požadavky *GET* je možné vidět v příkladu 2.1.3.

2.2.2 POST

Metoda *POST* je ve srovnání s *GET* bezpečnější, parametry a hodnoty předávané se neobjeví v žádném logovacím souboru na webovém ani na proxy serveru. Data jsou odesílána jako proud dat. Nyní je ovšem nutné uvést délku odesílaných dat, aby server věděl, kolik dat musí načíst ze vstupního proudu.

Nutno podotknout, že stejného výsledku (jako v 2.2.1) dosáhneme i použitím metody *GET*.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to arthur.
Escape character is '^]'.
POST /~jura/bc_example/expost.cgi HTTP/1.0
Content-length: 16

parameter=value
HTTP/1.1 200 OK
Date: Tue, 09 Nov 2004 10:55:21 GMT
Server: Apache/1.3.31 (Unix) PHP/4.3.9
Connection: close
Content-Type: text/html

<html>
 <head><title>POST example</title></head>
 <body>
  parameter p=value
 </body>
</html>
```

Příklad 2.2.1: Ukázka metody POST

Postará se o to již zmiňovaná funkce `param()`. Požadavek by měl tvar `GET /~jura/bc_example/expost.cgi?parameter=value HTTP/1.0`.

2.2.3 HEAD

Metodu *HEAD* zde uvádím pro doplnění. Její význam můžeme ocenit, pokud ji použijeme pro ladící účely. Touto metodou získáme všechny hlavičky, které jsou vygenerovány při volání skriptu, tělo dokumentu se však neodesílá.

2.3 Perl

Programovací jazyk Perl je velmi vhodným kandidátem pro tvorbu CGI aplikací. Jak je známo, je také jedním z nejoblíbenějších. Tento jazyk byl vytvořen za účelem zpracovávání textových informací, jeho autor — Larry Wall — nebyl spokojen s nástroji pro práci s texty. A tak stvořil Perl. Vynikající implementace regulárních výrazů ho činí ideálním jazykem pro CGI skripty.

Pro webové servery existují i moduly, které zrychlují běh perlových skriptů v porovnání s během jako CGI skript (mod_perl pro Apache). Pro zájemce Perlu neznalé doporučuji knihu [Perlgreen].

2.3.1 Moduly

Tento jazyk je velmi dobře rozšiřitelný pomocí modulů. Moduly umožňují perlovému skriptu používat různé instalované programy, pracovat s databázemi (`DBI`, `DBD`), vytvářet obrázky (`GD`), tvořit grafické aplikace (`Tk`), atd. Z našeho pohledu je však nejdůležitější modul *CGI*, který zjednodušuje práci se formuláři a tvorbu webových stránek.

Moduly jsou dostupné z archivu CPAN, [Perl]. Za použití webového rozhraní je možné vyhledávat moduly a také pročítat jejich dokumentaci. Dokumentaci lze procházet také programem **perldoc**. Při hledání řešení je vhodné hledat nejdříve zde, je velmi pravděpodobné, že

```
#!/usr/bin/perl

use CGI;
my $cgi = new CGI;
my $p = $cgi->param("parameter");

print<<HTML;
Content-type: text/html

<html>
 <head><title>POST example</title></head>
 <body>
  parameter p=$p
 </body>
</html>
HTML
```

Příklad 2.2.2: Skript expost.cgi

podobný problém již někdo (vy)řešil.

2.3.2 Modul CGI

Práci s formulářovými daty tedy usnadňuje modul `CGI`. Můžeme jednoduše získat hodnoty z textových polí, vybrané položky ze seznamu nebo si zpřístupnit obsah přijímaného souboru. Tento modul je objektově orientovaný, lze ho však používat také „neobjektově". Rozdíl spočívá v inicializaci modulu a následně v používání jeho funkcí (metod).

```
#!/usr/bin/perl
#objektová varianta
use CGI;
my $cgi = new CGI;
my $name = $cgi->param("name");

# neobjektová varianta
use CGI qw/:standard/;
my $jmeno = param("jmeno");
```

Příklad získání hodnot zaslaných skriptu — proměnné `name` a `jmeno`.

Příklad 2.3.1: Modul CGI

Použitím tohoto modulu nemusíme zjišťovat, jaká metoda byla použita pro zaslání požadavku, a tedy jak máme získat hodnoty proměnných. Samozřejmě lze získat i tuto informaci — z proměnné prostředí `REQUEST_METHOD`, resp. `$ENV{"REQUEST_METHOD"}`. Funkce `param()` nám vrátí přijatou hodnotu, případně seznam hodnot daného parametru. Podobně metoda `upload` vrátí `filehandle` pro přístup k obsahu datového toku.

Kapitola 3

Mobilní zařízení

Pod označení mobilní zařízení lze zařadit mobilní telefony, PDA a jiná *malá* zařízení. Tato zařízení nedisponují hardwarovými kapacitami srovnatelnými s osobními počítači a je tedy nutné vytvářet stránky s jistou obezřetností. Mezi nejvýraznější rozdíly patří kapacita pamětí, výkon procesorů a rychlost přenosu dat. Současné mobilní telefony jsou již vybaveny dostatečně výkonnými mikroprocesory a dostačující kapacitou paměti. Tento pokrok umožňuje přesunout internetové stránky i na obrazovky těchto zařízení.

3.1 Mobilní telefony

Mobilní telefony však musí splňovat jisté požadavky — musí podporovat protokol WAP. Splněním tohoto požadavku je mobilní telefon schopen připojit se nějakým způsobem k síti Internet. Může se jednat o připojení pomocí vytáčeného spojení (CSD, HSCSD) nebo pomocí paketového spojení (GPRS, EDGE). V prvním případě uživatel platí dobu spojení, ve druhém by měl platit za přenesená data. Bohužel skutečnost se může lišit a uživatel bude platit za wapové spojení platit v obou případech stejně.

Majitelé mobilních telefonů se také musí smířit se skutečností, že displej jejich zařízení se ani v nejmenším nepřibližuje monitoru připojeného k počítači.

3.2 Omezení

Existuje několik základních omezení mobilních zařízení. Nejvíce nás omezují velikost paměti, výkon mikroprocesoru a šířka dostupného pásma. Za patřičnou šířku pásma musíme samozřejmě zaplatit. Výkon procesorů se však nezadržitelně zvyšuje i v mobilních zařízeních. Navíc WML stránky do zařízení putují již ve zpracované formě — kompilované wapovou bránou.

Hlavní omezení je však paměť. Prvním mobilním telefonem s podporou WAPu a stránek pro mobilní telefony byl model 7110 od společnosti Nokia. Možnostmi tohoto telefonu jsou také určena některá omezení pro tvorbu stránek. Maximální velikost zobrazitelné stránky činí u „sedmdesátjednadesítky" 1397 bajtů. Hodnota je uvedena po zpracování na wapové bráně, délka zdrojového textu tedy může tuto hodnotu o něco málo překročit. Pokud nepřekročí, měli bychom mít jistotu, že se nám stránka zobrazí na všech novějších telefonech.

Vyskytuje se však ještě jedno omezení, které by mohlo znepříjemnit život tvůrci stránek. Tím je délka URL a tedy délka parametrů. Zde je magickým číslem 500, opět vychází z možností MT Nokia 7110. Tato nepříjemnost se tedy týká použití metody GET. Omezení se však liší nejen mezi telefony různých výrobců, ale také mezi jednotlivými modely stejného výrobce.

3.3 WAP

WAP, neboli *Wireless Application Protokol*, je sada protokolů umožňující přenos stránek z Internetu do mobilního telefonu. Nejen mobilních telefonů, ale také dalších mobilních zařízení jako jsou PDA. K přenosu se používá vlastní vrstva nad protokolem TCP/IP. Nicméně je možné k přenosu dat využít také paketového spojení přes GPRS nebo EDGE. (V době psaní této práce však technologii EDGE provozovala pouze společnost T-Mobile.)

3.3.1 WAP 1.0

Verze 1.0 staví na jazyce WML, ten se řadí mezi jazyky rodiny XML a je inspirován jazykem HTML. Mnoho konstrukcí je podobných, některé jsou poněkud odlišné.

3.3.2 WAP 2.0

Tato nová verze protokolu WAP rozšiřuje možnosti předchozích verzí. Zejména se rozšiřuje jazyk WML, ale také je přidána podpora pro značkovací jazyk *XHTML MP*, tedy eXtensible HyperText Markup Language Mobile Profile. Jedná se o profil jazyka XHTML určený pro mobilní zařízení. Nové mobilní telefony podporující WAP verze 2.0 budou schopné zobrazovat stránky napsané jak v jazyce WML tak v jazyce XHTML MP.

Pro WAP verze 2.0 byl také navržen protokol *WP-HTTP*, Wireless Profiled-HTTP. Jedná se o upravený protokol založený na HTTP 1.0. Mimo jiné tento protokol umožňuje přenášet komprimovaná data (deflate, gzip). Je vidět, že vývoj komunikačních protokolů směřuje stále více ke známým a osvědčeným standardům, proto se nesmíme divit, že WP-HTTP umožní skriptům uložit v prohlížeči dokonce sušenky (Cookies).

Kapitola 4

HTML, WML

V této kapitole si představíme dva jazyky určené ke tvorbě dokumentů. Všeobecně známý jazyk HTML a jazyk WML určený pro mobilní zařízení. Jazyk HTML zde bude představovat referenční jazyk pro WML. Ukážeme, že pomocí jazyka WML jsme schopni vytvořit ekvivalentní stránky a formuláře.

4.1 HTML

Jazyk HTML je nejrozšířenějším jazykem pro tvorbu stránek v síti Internet. Jedná se o tak známý jazyk, že snad není potřeba ho ani hlouběji představovat. Bylo o něm napsáno nepřeberné množství kniha a článků. Neznalého čtenáře si tímto dovolím odkázat na [Kosek] a dále budu předpokládat znalost tohoto jazyka, resp. některé z jeho novějších specifikací XHTML.

Nejpodstatnější prvky jazyky pro nás tvoří formulářové prvky, nebot'ty uživateli umožňují interaktivně komunikovat se skripty tvořící nějaký informační systém.

4.1.1 Formuláře

Vytvoření formuláře pro získání dat je velice jednoduché. Podle typu informace si zvolíme některý z prvků. Příklad 4.1.1 poskytuje stručný přehled o prvcích v HTML.

form definuje jeden formulář s prvky určenými k odeslání

input textové vstupní pole, možnost výběru (radio), zaškrtávací pole (checkbox)

textarea textové pole pro delší a přehlednější vstup

select možnost výběr (z) více hodnot

Příklad 4.1.1: Formulářové prvky

Použitím těchto prvků můžeme vytvořit formulář, kterým získáme od návštěvníka nějaké informace nebo jehož pomocí návštěvník zadá nějaký požadavek. Příkladem jednoduchého formuláře je 4.1.2. Odeslané hodnoty je možné zpracovat a zobrazit např. skriptem 4.1.3.

4.2 WML

Jazyk WML (Wireless Markup Language) byl navržen pro značkování textů a stránek určených pro zobrazování na obrazovkách mobilních zařízení. V současné době jsou dostupné verze 1.0, 1.1, 1.2 a 1.3, další chystanou verzí je verze 2.0. V ukázkách bude použita verze 1.1.

Jazyk WML patří do rodiny jazyků XML. Každá stránka tedy musí splňovat požadavky kladené na validní XML dokument. Na začátku stránky je nutné uvést verzi XML a definici

```
<html>
 <head><title>Jednoduchý formulář</title></head>
 <body>
  <form action="./htmlex.cgi" method="get">
   Jméno: <input name="jmeno"><br>
   Pohlaví: <input name="pohlavi" type="radio" value="muž" > muž
          <input name="pohlavi" type="radio" value="žena"> žena<br>
   Typ průkazu:
     <select name="prukaz" multiple="multiple">
      <option value="občanský průkaz">občanský</option>
      <option value="pas">cestovní</option>
      <option value="zbrojní průkaz">zbrojní</option>
     </select><br>
   Váš vzkaz: <textarea name="vzkaz"></textarea><br>
   <input type="submit">
  </form>
 </body>
</html>
```

Příklad jednoduchého formuláře vyzvídajícího základní informace o návštěvníkovi stránky, *htmlex.html*.

Příklad 4.1.2: Jednoduchý HTML formulář

typu dokumentu (DTD). DTD pro WML 1.1 je dostupné z adresy `http://www.wapforum.org/DTD/wml_1.1.xml` (třetí řádek 4.2.2).

Webový server musí posílat správné identifikace souborů (4.2.1). Komunikace neprobíhá přímo mezi mobilním telefonem a webovým serverem nýbrž prostřednictvím wapové brány. Wapová brána převádí obsah požadovaných stránek do binární podoby, kterou odesílá mobilnímu telefonu. Bohužel mobilní operátoři blokují porty použité ke komunikaci a není tedy možné využít vlastní wapovou bránu.

4.2.1 Struktura dokumentu

Dokument je tvořen tzv. *deck*em. Ten se dále dělí na jednotlivé karty, *card*. Karty odpovídají jednotlivým obrazovkám, které se mohou zobrazit na obrazovce zařízení. Na prvním řádku musí být uvedena specifikace verze XML a kódování znaků pokud se liší od UTF-8 nebo UTF-16, v našem případě (4.2.2) je verze 1.0 a kódování iso-8859-2.

Jednoduchá stránka je tedy tvořena kartami, mezi nimiž je možné přecházet pomocí odkazů. Obsah stránky musí být obklopen značkami pro odstavec, tedy `p`. Pomocí značky `a` a identifikátoru karty se můžeme pohybovat mezi kartami uvnitř jednoho `deck`u. Nedochází tak k přenosu dat, protože ta již jsou přítomna. Naopak použitím adresy jiné stránky lze přejít na jinou stránku, včetně možnosti vybrat přímo některou kartu. Jedná se tedy o obdobu adresování návěští v jazyce HTML, v obou jazycích se přenáší celá stránka a následně jsme nasměrováni na správnou část dokumentu. Výhodou klasických HTML stránek a prohlížečů je, že pro nás velikost stránky netvoří vážný problém. Prohlížeč zobrazí stejně dobře stránku velkou jeden kilobajt jako stránku velkou půl megabajtu. V případě WML stránek ovšem musíme dávat pozor, protože mobilní telefony nedisponují srovnatelnými kapacitami, tento problém bude diskutován později, 3.2.

```
#!/usr/bin/perl
use CGI;
my $cgi = new CGI;
my %v = $cgi->Vars;
my (@p) = $cgi->param("prukaz");
my $prukazy = join(", ", @p);

print<<HTML;
Content-type: text/html

<html>
 <head><title></title></head>
 <body>
  Jmenujete se $v{"jmeno"}, jste $v{"pohlavi"} a vlastníte $prukazy.<br>
  A Váš vzkaz zní: '$v{"vzkaz"}'.
 </body>
</html>
HTML
```

Jednoduchý skript, který zobrazuje uživatelem zadané informace, *htmlex.cgi*.

Příklad 4.1.3: Zpracování formuláře

```
application/vnd.wap.wmlc        wmlc
application/vnd.wap.wmlscriptc  wmlsc
text/vnd.wap.wml                wml
text/vnd.wap.wmlscript          wmls
image/vnd.wap.wbmp              wbmp
```

Příklad 4.2.1: MIME typy

4.2.2 Odkazy

K vytvoření odkazů mezi stránkami hypertextového dokumentu používáme stejnou značku jako v případě jazyka HTML, tedy `a`. Její parametr `href` určuje, kam se přesuneme. Můžeme procházet jednotlivé karty aktuální stránky nebo přejít na úplně jinou stránku. První varianta odpovídá přechodu na nějaké návěští uvnitř HTML stránky, druhá možnost donutí prohlížeč nahrát odkazovanou stránku a zobrazit ji.

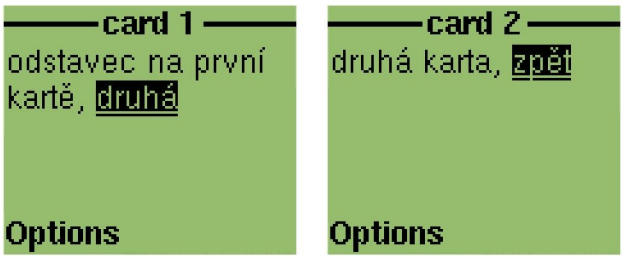
Odkazy lze využít pro předání hodnot parametrů, v příkladu 4.2.3 se jedná o řádky 3 a 4. Na řádku č. 4 je použita poněkud zvláštní konstrukce. Ta nám umožní odeslat hodnotu nějakého formulářového prvku, tato funkcionalita bude diskutována v 4.2.6.

4.2.3 Vstupní prvky

Množina vstupních prvků je na první pohled chudší než tomu je u jazyka HTML. V jazyce WML tedy máme k dispozici pouze značku pro vstupní pole a pro výběrový seznam několika možností, `input` a `select`. Ani takovéto omezení nám ovšem nebrání v tvorbě srovnatelných formulářů (jako u HTML).

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
 <card id="c1" title="card 1">
  <p>odstavec na první kartě, <a href="#c2">druhá</a></p>
 </card>

 <card id="c2" title="card 2">
  <p>druhá karta, <a href="#c1">zpět</a></p>
 </card>
</wml>
```



Ukázka jednoduché WML stránky se dvěma kartami propojenými odkazy, *basic.wml*.

Příklad 4.2.2: Jednoduchá WML stránka

```
<a href="#c2"> karta č. 2</a>
<a href="http://www.fi.muni.cz/~xhudak/bc/index.wml">FI</a>
<a href="./param.cgi?parameter=hodnota"> URL s parametrem</a>
<a href="./param.cgi?parameter=$(par)"> URL s parametrem jinak</a>
```

Odkaz na prvním řádku nás zavede na kartu označenou identifikátorem `c2`, druhý potom na stránku v Internetu.

Příklad 4.2.3: Cíle odkazů

4.2.4 Značka input

Značka `input` představuje základní vstupní prvek každého formuláře, jinak tomu není ani ve WML. Abychom získaly vloženou hodnotu, musíme uvést atribut `name`. Pomocí tohoto jména se nyní můžeme na stránce odvolávat na obsah vstupního pole. To provedeme zápisem `$jmeno` nebo `$(jmeno)`, kde `jmeno` je hodnota atributu `name`. První případ použiji při zobrazování hodnoty v kartě, druhý pro předání hodnoty k odeslání. Ještě je možné použít třetí formát: `$(jmeno:konverze)`, *konverze* zde značí `noesc`, `escape` a `unesc`. Tyto hodnoty postupně značí „žádná změna hodnoty“, „zakódování pro URL“ a „dekódování“. <input name="jmeno" /> a `$(jmeno)` tedy představují platné hodnoty. Odkaz na obsah pole můžeme použít jako v příkladě 4.2.3 nebo 4.2.9.

Atributem `type` nastavíme typ vstupního pole. Pro typ jsou povolené pouze dvě hodnoty — `text` a `password`. Hodnota text je výchozí. Zde vidíme rozdíl proti HTML, chybí možnost vytvořit zaškrtávací políčko a přepínač. S úspěchem však použijeme `select`.

Z jazyka HTML známe atributy `size` a `maxlength`. Ty jsou dostupné i pro vstupní pole se stejným významem. Novým atributem je však `emptyok` s hodnotami `true` a `false`. Tento atribut řídí povolení prázdného vstupu u formátovaného vstupního pole.

Úplnou novinku představuje možnost formátování vstupu. Můžeme tedy vynutit vstup složený pouze z číslic nebo malých znaků ... Výhoda spočívá také v automatické změně na požadovaný vstup na straně mobilního zařízení — klávesnice se přepne např. do režimu vklá-

dání číslic. Formátování dosáhneme použitím atributu `format` a uvedením příslušné hodnoty. Hodnoty představují obdobu regulárních výrazů nad texty, nicméně kvantifikátory se uvádějí před požadovaným formátovacím znakem.

A,a velké (malé) písmeno nebo interpunkční znaménko

X,x velké (malé) písmeno

M,m libovolný znak, resp. velké (malé) písmeno, které je možné přepnout na malé (velké)

N číslice

\c znak (c), který se má zobrazit v poli

***f** libovolný počet opakování, f je jeden z uvedených formatovacích znaků, může se vyskytnout maximálně jednou a to na konci

nf n výskytů formátovacího prvku f, n může nabývat hodnot 1 až 9, číslo n však představuje maximální počet daných znaků

```
<input type="text" name="jmeno" format="A*a" />
<input type="text" name="narozen" format="1NN\.1NN\.NNNN" />
<input type="text" name="vek" format="3N" />
<input type="text" name="login" format="XXX5X" />
```

(zkusit!!)
Zde máme různé formáty pro některé vstupní hodnoty.

Příklad 4.2.4: Formátování vstupu

4.2.5 Značka select

Chceme-li návštěvníkovi poskytnout možnost výběru z daných možností, použijeme `select`. Opět pomocí atributu `name` určíme jméno. Dále pak pomocí `multiple` můžeme povolit výběr více hodnot, hodnota nabývá pravdivostních hodnot `true` nebo `false`.

Pokud potřebujeme nechat návštěvníka vybrat jednu z hodnot, uvedeme atribut `multiple` s hodnotou `false` nebo vůbec — <select name="vyber" multiple="false">...</select>. Tím získáváme klasický výběrový list a zároveň přepínač z jazyka HTML. Naopak, použijeme-li <select name="vyber" multiple="true">...</select>, návštěvním může vybrat více položek. Tedy obdobně jako použitím zaškrtávacích polí nebo stejného seznamu v HTML.

Jednotlivé položky definujeme stejně jako v HTML pomocí značky `option` a jejího atributu `value`. Jednoduchý seznam je na uveden v příkladu 4.2.5, komplexnější potom v 4.2.6.

Zde již narazíme na drobný problém s dekódováním hodnot. Klasické webové prohlížeče totiž vytvoří požadavek s parametry ve tvaru: skript.cgi?k=phone&k=fax. WML prohlížeč ovšem vytvoří skript.cgi?k=phone%3Bfax. Zpracujeme odeslaná data skriptem 4.2.7 (select-tex.cgi).

```
<select name="kontakt" multiple="true">
 <option value="phone">telefon</option>
 <option value="fax">fax</option>
 <option value="email">e-mail</option>
</select>
```

Jednoduchý výběrový seznam, je možné vybrat libovolnou podmnožinu možností.

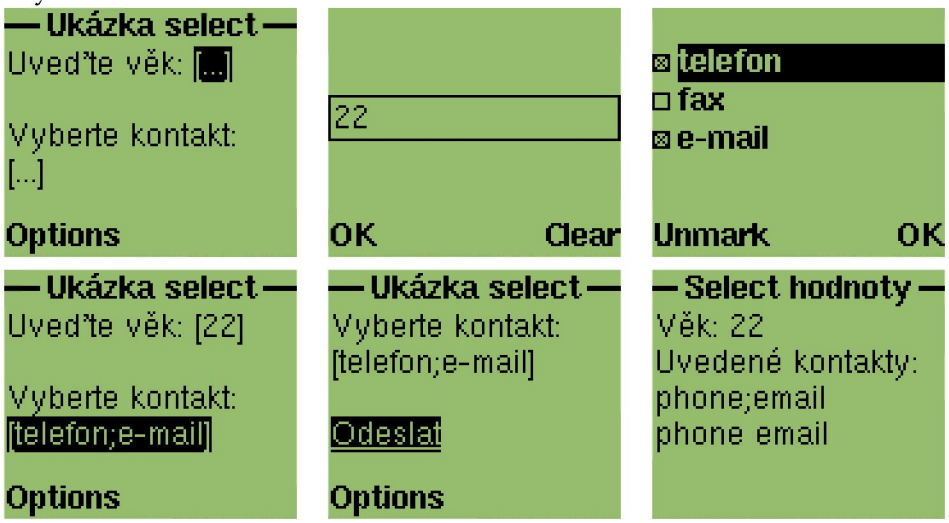
Příklad 4.2.5: Výběrový seznam

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
 <wml>
  <card id="c" title="Ukázka select">
   <p>
    Uveďte věk: <input type="text" name="v" format="N2N" /><br/>
    Vyberte kontakt:<br/>
    <select name="k" multiple="true">
     <option value="phone">telefon</option>
     <option value="fax">fax</option>
     <option value="email">e-mail</option>
    </select>
    <br/>
    <a href="./selectex.cgi?kontakt=$(k)&amp;vek=$(v)">Odeslat</a>
   </p>
  </card>
 </wml>
```

Výběrový seznam kontaktů, *selectex.wml*.

Příklad 4.2.6: Ukázka výběrového seznamu

Navštívíme-li nyní pomocí emulátoru Deck-It (5.1.1) tuto stránku, získáme následující obrazovky.



4.2.6 Nabídka a odeslání dat

Podobně jako v HTML musíme vytvořit tlačítko, po jehož stisknutí prohlížeč odešle hodnoty polí daného formuláře. V jazyce WML toho docílíme pomocí značky `do` a jejích parametrů,

```
#!/usr/bin/perl
use CGI;
my $cgi = new CGI;
my $jmeno = $cgi->param("jmeno");
my (@k) = $cgi->param("kontakt");
my (@kontakty) = split(/;/, $k[0]);

print<<WML;
Content-type: text/vnd.wap.wml

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
 <card id="c" title="Select hodnoty">
  <p>
   Jméno: $jmeno<br/>
   Uvedené kontakty: @k<br/>
   @kontakty
  </p>
 </card>
</wml>
WML
```

Skript zobrazující odeslané hodnoty, *selectex.cgi*.

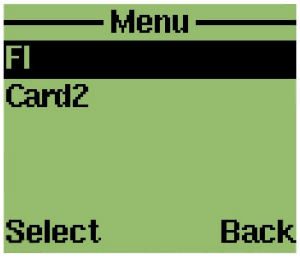
Příklad 4.2.7: Skript selectex.cgi

resp. dalších značek. Přidáme tak jednu položku do nabídky prohlížeče, tu je možné využít nejen k odeslání formulářových dat, ale také k zjednodušení navigace. Ukázky zápisu odkazu je možné vidět v příkladu 4.2.8.

Abychom nyní odeslali požadavek s parametry, musíme určit metodu přenosu dat a jména parametrů, které chceme odeslat. K tomu slouží značky `go` a `postfield`. Pomocí `go` určíme cílový skript a značkami `postfield` vyjmenujeme jména parametrů a přiřadíme jim hodnoty z formulářových prvků. Značku `go` použijeme uvnitř definice akce (`do`), podobně `postfield` uvádíme uvnitř `go`.

Z ukázky 4.2.9 je vidět, že nezáleží na pojmenování vstupních prvků. Jejich obsahy mohou být přiřazeny jiným parametrům. Nahrazením hodnoty `post` za `get` v určení metody získáváme formulář odesílaný metodou GET. Větší kouzlo však skrývá odkaz pod vstupním polem. Jeho pomocí lze také odeslat hodnotu uvedenou v poli `p`. V případě zvolení takového odkazu se do připraveného URL dosadí hodnoty a celá adresa se použije jako odkaz.

```
<card id="c1" title="Tag do">
 <do type="prev" label="Zpet">
  <prev/>
 </do>
 <do type="options" label="FI">
  <go href="http://www.fi.muni.cz/~xhudak/index.wml"/>
 </do>
 <do type="options" label="Card2">
  <go href="#c2"/>
 </do>
</card>
```



Ukázka možností značky `do`. Do nabídky prohlížeče tento kód přidá alespoň položky „FI“ a „Card2“, položka „Zpet“ může být překryta vlastním tlačítkem pro přesun na předchozí stránku. *doexample.wml*

Příklad 4.2.8: Značka <do>

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
 <card id="c1" title="Odeslání dat">
  <do type="accept" label="Odeslat POST">
   <go href="./param.cgi" method="post">
    <postfield name="parameter" value="$(p)"/>
   </go>
  </do>
  <p>Váš text: <input type="text" name="p" /></p>
  <p><a href="./param.cgi?parameter=$(p)">Odeslat odkazem</a></p>
 </card>
</wml>
```

Tímto způsobem říkáme, že obsah vstupního prvku nazvaného `p` bude skriptu `param.cgi` předán jako hodnota parametru `parameter`. K odeslání prohlížeč použije metodu POST. Pro doplnění je zde uvedena možnost odeslat data odkazem. *goexample.wml*, *param.cgi*

Příklad 4.2.9: Odeslání dat

Kapitola 5

Vývoj

V této kapitole uvedu některé nástroje usnadňující vývoj WML stránek. Tyto nástroje však neodstraní nutnost testování aplikací na mobilních telefonech a zařízeních. V různých prohlížečích se stejná stránka může zobrazit různě, opět to záleží na emulovaném telefonu nebo implementaci prohlížeče. Jejich přínos však spočívá ve snížení nákladů na testování.

5.1 Emulátory

Mezi základní nástroje patří emulátory zobrazující WML stránky podobně jako webové prohlížeče stránek HTML. Setkal jsem se s několika různými aplikacemi a vybral jsem dle mého názoru ty nejslibnější. Pravdou ovšem zůstává, že jsem nevyzkoušel mnoho integrovaných vývojových prostředí a to z prostého důvodu — byla vytvořena pro jediný operační systém.

5.1.1 Deck-It

Společnost PyWeb.com vytvořila prohlížeč WML stránek nazvaný *Deck-It*. Jedná se o emulátor mobilního telefonu Nokia 7110. Existují verze pro operační systémy Linux a MS Windows, aplikace samotná je naprogramovaná v jazyce TCL/Tk. Poslední verze byla 1.2.3. Obrázky v ukázkách jsou právě z tohoto emulátoru.

Bohužel se zdá, že společnost PyWeb.com ukončila svou činnost, resp. její webová stránka není dostupná.

5.1.2 Wapaka

Jiným prohlížečem je *Wapaka*. Výhoda tohoto prohlížeče spočívá v jeho přenositelnosti, je naprogramován v jazyce Java. Běží tedy na systémech, pro které byl implementován interpret tohoto jazyka. Používaná verze byla 3.18 (build 2090).

5.1.3 Opera

Hrubou chybou by bylo zapomenout na kvalitní webový prohlížeč *Opera*. Opera má velmi dobrou podporu XML stránek, pro nás je ovšem důležitý styl určený pro WML stránky. V Opeře tedy získáváme chytrého pomocníka pro vývoj WML stránek.

5.2 XHTML

Se zvýšením výkonu mobilních zařízení a zlepšenou podporou na straně výrobců můžeme předpokládat, že dojde k tlaku na použití varianty XHTML MP, XHTML Mobile Profile. Stránky v tomto formátu se podstatně více blíží klasickým HTML stránkám.

5.3 XSLT transformace

Dalším užitečným nástrojem jsou XSLT transformace. Pokud máme dokumenty určené pro prezentaci zájemcům uloženy v nějakém XML dokumentu, je možné napsat XSLT transformace

nejen do jazyka HTML, ale také do jazyka WML. Tím získáváme možnost oslovit více zájemců (at'jsou kdekoliv). V případě programovacího jazyka Perl můžeme využít balík `XML::XSLT`.

5.4 Java, J2ME a MIDP

Výrobci mobilních telefonů nám poskytují možnost využít rozhraní programovacího jazyka Java, resp. speciální verze Java 2 Micro Edition, ke tvorbě vlastních aplikací. Pokud chceme svým vlastním způsobem, odlišným od WML, implementovat formulářové operace, máme možnost naprogramovat si vlastní prohlížeč. Zřejmě pro něj navrhneme i vlastní jazyk, který se nám bude dobře zpracovávat. K volání a předávání parametrů můžeme plně využít CGI rozhraní.

5.4.1 J2ME

J2ME (Java 2 Micro Edition) představuje menší verzí programovacího jazyka Java 2 Standard Edition, která je určena pro mobilní zařízení. Mobilní zařízení obsahují minimální virtuální stroj pro Javu, tzv. KVM (Kilo Virtual Machine). Tato verze Javy vyžaduje poněkud odlišný přístup k překladu zdrojového kódu.

5.4.2 MIDP

Mobile Information Device Profile (MIDP) je profil J2ME používaný v bezdrátových zařízeních, který definuje prostředí pro vývoj aplikací. Minimální vlastnosti mobilního zařízení definuje následovně: displej alespoň 96 krát 54 pixelu s barevnou hloubkou alespoň 1 bit, klávesnice pro jednu nebo dvě ruce (případně dotyková obrazovka) a paměť min. 32 kB nestálé paměti pro Javu, 128 kB stálé paměti pro komponenty MIDP a 8 kB pro ukládání dat aplikací. Tyto informace a mnoho dalších nalezneme v knize [Mahmoud].

Pro tvorbu prohlížeče je podstatná existence síťové komponenty označené jako „obecný připojovací systém“ (Generic Connection Framework). Tento systém umožňuje otevírat soketová a datagramová spojení a také HTTP spojení. Parametrem pro metody vytvářející spojení je URL adresa ve známém tvaru. Volání tedy vypadá např. Connector.open("http://www.fi.muni.cz/"). Problematikou programování aplikací pro mobilní zařízení v jazyce J2ME se zabývá již zmiňovaná kniha [Mahmoud].

Kapitola 6

Závěr

V této práci jsem stručně popsal princip fungování CGI, dále některé vlastnosti mobilních zařízení a použití jazyka WML. Vytvořením jednoduchých stránek v tomto jazyce lze realizovat formulářové operace proti CGI skriptům napsaných v programovacím jazyce Perl.

Věřím, že popsané principy a příklady stránek a skriptů poslouží zájemcům jako podklad pro vlastní tvorbu stránek a aplikací. Mezi přiloženými stránkami je i reálná aplikace k odesílání a k čtení e-mailovým zpráv.

Literatura

[CGI] *Dokumenty o CGI*, CGI: Common Gateway Interface <http://hoohoo.ncsa.uiuc.edu/cgi/> <http://www.w3.org/CGI/> The Common Gateway Interface - RFC Project Page <http://cgi-spec.golux.com/>. 2

[Gund98] Gundavaram, S.: *CGI - programování webových stránek a aplikací*, Computer Press, 1998, 80-7226-088-X, <http://www.cpress.cz/>. 2.2

[Haba] Hába, M.: *WAP Tvorba stránek pro mobilní zařízení*, Computer Press, 2002, 80-7226-814-7, <http://www.cpress.cz/>.

[Kallay] Kállay, F., Peniak, P.: *Počítačové sítě a jejich aplikace*, Grada Publishing, 1999, 80-7169-407-X, <http://www.grada.cz/>.

[Kosek] Kosek, J.: *XML pro každého*, Grada Publishing, 2000, 80-7169-860-1, <http://www.grada.cz/> <http://www.kosek.cz/>. 4.1

[Mahmoud] Mahmoud, Q.: *Naučte se Java 2 Micro Edition*, Grada Publishing, 2002, 80-247-0444-7, <http://www.grada.cz/>. 5.4.2

[OMA] *Open Mobile Alliance*, <http://www.openmobilealliance.org/> WAP Forum <http://www.wapforum.org/> <http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html> WML 1.1 <www.wapforum.org/what/technical/SPEC-WML-19990616.pdf>.

[Perl] *Odkazy na Perl*, Perl <http://www.perl.com/> CPAN <http://www.cpan.org/> mod_perl pro Apache <http://perl.apache.org/>. 2.3.1

[Perlgreen] Satrapa, P.: *Perl pro zelenáče*, Neokortex, 2000, 80-86330-02-8, <http://www.neo.cz/perl_green.html>. 2.3

[RFCs] *Request for Comments*, Internet Comunity, <ftp://ftp.fi.muni.cz/pub/rfc/>. 2.1.3

[W3C] *World Wide Web Consortium*, <http://www.w3.org/>.

[w3schools] *W3Schools Online Web Tutorial*, W3Schools, <http://www.w3schools.com/>.