Webový server Apache a moduly

Pavel Janík ml.
http://www.janik.cz

Webový server Apache je nejpoužívanějším webovým serverem v Internetu. Je součástí snad každé distribuce operačního systému Linux a jeho zdrojové texty jsou volně dostupné za podmínek licence, která má velmi blízko k BSD licenci. Jeho konfigurační model je velmi rozmanitý a silný nástroj, který dovoluje prakticky cokoli od přepisování jednotlivých částí URL, až po virtuální webové servery. Tyto vlastnosti jej přímo předurčují k hromadnému nasazení.

Ale v tomto článku bych nechtěl Apache jen vychvalovat. Apache má i své mouchy. Protože poskytuje velkou spoustu možností, může se stát, že na jeden konkrétní účel není tím nejvhodnějším serverem. Pokud bych měl vybrat webový server, který bude pouze poskytovat statická data (obrázky), pravděpodobně bych zvolil jiný webový server (např. thttpd). Apache tento úkol samozřejmě zvládne také, ale jeho výkon v podobných situacích může být až dvakrát menší než výkon specializovaných webových serverů. Pro náš účel poskytování pouze statických dat je velmi vhodným serverem například i kernel-httpd, který je v nových verzích linuxového jádra.

Princip sdílených modulů

Server Apache má však i další výhody, z nichž bych vám v tomto článku chtěl představit podporu sdílených modulů (Dynamic Shared Object — DSO). Do verze 1.3 byl webový server Apache jedním statickým celkem, programem o velikosti něco přes 600 kB podle toho, jaké jsme zvolili možnosti při překladu. Pokud jsme například na nějakou vlastnost zapomněli, museli jsme Apache znovu přeložit a nainstalovat. To si samozřejmě vyžádá nějaký čas, kdy naše služby nebudou fungovat perfektně. Vývojáři serveru tedy vyšli svým uživatelům vstříc a od verze 1.3 Apache podporuje sdílené moduly.

Jak tento mechanismus funguje? Webový server Apache je kompletně modularizovaný, jeho hlavní funkcionalita se nachází v modulu `mod_core.c` a vlastní binární program `httpd` obsahuje ještě podporu pro sdílené moduly. Ta je uložena v modulu `mod_so.c`, jenž je společně s modulem `mod_core.c` jediný, který nemůže být dynamicky sdílen a musí být přeložen přímo v programu `httpd`. Správce webového serveru si při instalaci může zvolit, které vlastnosti bude server obsahovat přímo v sobě a pro které si bude muset zavést sdílený modul. Tato flexibilita je jednou z hlavních předností Apache.

Překlad Apache s podporou sdílených modulů

Podívejme se tedy, jak přeložíme webový server Apache s podporou sdílených modulů. Stejně jako vše kolem Linuxu je i toto jednoduché:

```
tar xfz apache_1.3.12.tar.gz
cd apache_1.3.12
./configure --prefix=~/Apache --enable-module=so
make
make install
```

Tato sekvence příkazů nám nainstaluje webový server do adresáře `Apache` v našem domovském adresáři. Nyní si ověříme, zda je binární program `httpd` opravdu připraven pro použití modulů:

```
[pavel@SnowWhite pavel]$ ~/Apache/bin/httpd -l
Compiled-in modules:
  http_core.c
  mod_env.c
  mod_log_config.c
  mod_mime.c
  mod_negotiation.c
  mod_status.c
  mod_include.c
  mod_autoindex.c
  mod_dir.c
  mod_cgi.c
  mod_asis.c
  mod_imap.c
  mod_actions.c
  mod_userdir.c
  mod_alias.c
```

```
  mod_access.c
  mod_auth.c
  mod_so.c
  mod_setenvif.c
[pavel@SnowWhite pavel]$
```

Pomocí příkazu `httpd -l` jsme si nechali zobrazit seznam modulů, které program `httpd` má v sobě přeloženy. Mezi nimi je i modul `mod_so.c` realizující vlastní podporu sdílených modulů.

Instalujeme modul jako sdílený

V dalším textu si ukážeme, jak nainstalovat zvolený modul (v našem případě `mod_info`) jako sdílený. Pokud bychom instalovali Apache standardním postupem popsaným v dokumentaci, neměl by náš webový server vestavěnu podporu některých zajímavých modulů. Standardně jsou totiž přeloženy pouze některé moduly — viz příkaz `./configure --help`:

```
[access=yes      actions=yes     alias=yes      ]
[asis=yes        auth=yes        auth_anon=no   ]
[auth_db=no      auth_dbm=no     auth_digest=no ]
[autoindex=yes   cern_meta=no    cgi=yes        ]
[digest=no       dir=yes         env=yes        ]
[example=no      expires=no      headers=no     ]
[imap=yes        include=yes     info=no        ]
[log_agent=no    log_config=yes  log_referer=no ]
[mime=yes        mime_magic=no   mmap_static=no ]
[negotiation=yes proxy=no        rewrite=no     ]
[setenvif=yes    so=no           speling=no     ]
[status=yes      unique_id=no    userdir=yes    ]
[usertrack=no    vhost_alias=no                 ]
```

Moduly `mod_rewrite` nebo `mod_info`, které nejsou standardně přeloženy, jsou přitom velice zajímavé a jsou použity na mnoha serverech. Jak tedy donutit Apache, aby nám přeložil modul `mod_info` jako sdílený? Nebudeme jej potřebovat vždy a při reálném nasazení by nám mohl server zpomalovat či působit jiné obtíže, a proto jej přeložíme pouze jako sdílený:

```
./configure --prefix=~/Apache --enable-module=info --enable-shared=info
```

Vlastní webový server potom nebude obsahovat podporu modulu `mod_info` přímo v sobě, ale tato podpora bude připravena jako sdílený modul v souboru `~/Apache/libexec/mod_info.so`. Webový server o připraveném modulu sám o sobě neví, musíme mu říci, že jej má zavést. To provedeme přidáním následujícího řádku do konfiguračního souboru `~/Apache/etc/httpd.conf`:

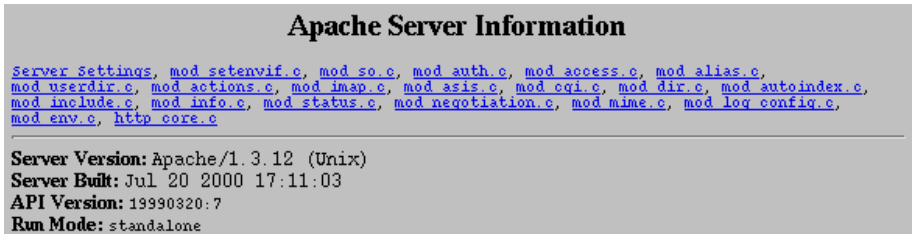
```
LoadModule info_module        libexec/mod_info.so
```

Pokud server instalujeme, je tento řádek doplněn automaticky, ale pokud budeme přidávat modul až po instalaci, bude se nám tento postup hodit.

Výborně — náš server nyní umí zavést modul `mod_info` a můžeme jej tedy nakonfigurovat. Tento modul pracuje tak, že je-li nastaveno v konfiguračním souboru `httpd.conf`, aby obsluhoval nějaké URL, například `/info`:

```
<Location /info>
  SetHandler server-info
</Location>
```

je v případě požadavku na stránku `/info` zavolán a pošle webovému prohlížeči informace o serveru samotném, o jeho modulech a o jejich konfiguraci (viz obrázek).



Modul `mod_info` je tedy výhodný při přípravě a optimalizaci samotného serveru, ale ve finální instalaci jej pravděpodobně opět zrušíme například zakomentováním řádku `LoadModule` v konfiguračním souboru webového serveru.

Jaké jsou tedy výhody a nevýhody modulů?

K čemu tedy může být podpora sdílených modulů vhodná? Jak jsme si již výše napsali, můžeme modulů s výhodou využít při optimalizaci konfigurace našeho serveru — nejprve si přeložíme Apache tak, že všechny moduly budou sdílené, a tudíž nebudou přeloženy přímo v samotném webovém serveru. Při testování či vývoji aplikace zjistíme, které moduly budeme nutně potřebovat. To vše stále bez překladu a instalace nové verze webového serveru (pouze pomocí úprav konfiguračního souboru `httpd.conf`).

Až budeme mít jistotu, že více modulů nepotřebujeme, přeložíme Apache znovu, ale tak, aby měl všechny potřebné moduly vestavěny v sobě. Pro finální nasazení je již podpora modulů zbytečná. Mohou ale existovat situace, kdy je vhodné, aby i finální webový server obsahoval podporu modulů. Například potřebujeme-li provozovat více instancí jednoho serveru, můžeme každou provozovat s jiným konfiguračním souborem, a tudíž s jinými vlastnostmi.

Pokud používáme další rozšiřující moduly, které nejsou součástí zdrojových textů Apache, je podpora modulů téměř nutností. Pokud bychom moduly nepoužili, museli bychom například při vydání každé nové verze PHP přeložit kompletně celý webový server včetně dalších vestavěných částí. S moduly pouze vytvoříme nový modul, vyzkoušíme jej na druhé instanci serveru a pokud je vše v pořádku, můžeme jej nasadit na hlavní server.

Sdílené moduly samozřejmě nemají pouze výhody — na některých platformách (Linux však mezi ně nepatří) bohužel není jejich podpora dokončena a na jiných není vzhledem k nedostatkům operačního systému vůbec možná. Další mírnou nevýhodou je, že zavádění sdíleného modulu do adresového prostoru vlastního serveru je dosti časově náročná operace, ale na výkonných serverech je tento čas minimální, avšak je třeba s ním počítat. Monolitický server tedy startuje rychleji.

Na některých platformách (i na Linuxu) může být sdílený modul díky „inteligenci“ překladače nebo „schopnostem“ dané platformy přeložen tak, že výsledná rychlost provedení některých částí kódu je nižší, než kdyby byl modul vestavěn přímo v serveru.

Píšeme vlastní modul do Apache

V předchozí části jsme si ukázali, jak jednoduchá je práce se sdílenými moduly ve webovém serveru Apache. Nyní si ukážeme, že stejně tak jednoduché je i vytvoření vlastního modulu. Ale nebudeme předbíhat.

Pokud přeložíme webový server Apache s podporou sdílených modulů, je vytvořen i perlový skript `~/Apache/bin/apxs`, který slouží ke zjednodušení práce s moduly. Kompletní popis jeho funkcí je v manuálové stránce `apxs(8)`. Tento skript můžeme použít i pro vytvoření šablony pro nový modul.

Nyní již víme, jaké funkce nám může poskytnout skript `apxs` (protože jsme si přečetli jeho manuálovou stránku), a můžeme začít psát vlastní modul. Jakou bude mít funkci? Pro začátek zkusíme něco jednoduchého — napíšeme modul, který zobrazí aktuální čas na serveru. Samozřejmě je možné tento problém vyřešit CGI skriptem, nějakým skriptovacím jazykem (např. PHP) nebo i jinak, jednodušeji, rychleji a efektivněji, ale my využijeme tento problém k vysvětlení principu tvorby modulů.

Pozor, začínáme! Pomocí skriptu `apxs` si vytvoříme šablonu pro náš modul:

```
[pavel@SnowWhite Tmp]$ apxs -g -n datum
Creating [DIR]  datum
Creating [FILE] datum/Makefile
Creating [FILE] datum/mod_datum.c
[pavel@SnowWhite Tmp]$
```

Jak vidíme, byl vytvořen adresář `datum`, který obsahuje dva soubory — `Makefile` a `mod_datum.c`. Druhý soubor obsahuje vlastní zdrojový text nového modulu. Modul tak, jak je vygenerován v šabloně, při požadavku na URL, které obsluhuje, vrátí prohlížeči uživatele pouze text

```
The sample page from mod_datum.c
```

a je tedy velmi jednoduchý. Zdrojový text je velmi dobře komentován, ale i přesto si jeho strukturu podrobně vysvětlíme, abychom předešli nejasnostem.

```
#include "httpd.h"
```

```
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"
```

Sekce hlavičkových souborů obsahuje čtyři řádky. Každý z těchto souborů obsahuje definice specifických konstant a funkcí webového serveru Apache. Možná si všimnete, že názvy všech souborů jsou uzavřeny v uvozovkách, a tudíž se při vlastním překladu musí nacházet v aktuálním adresáři nebo v adresáři, který musíme překladači oznámit. Tyto problémy za nás řeší skript `apxs`, který předá umístění adresáře s hlavičkovými soubory serveru Apache překladači. Jaký adresář to je? Opět využijeme znalostí načerpaných z manuálové stránky a použijeme následující příkaz:

```
apxs -q INCLUDEDIR
```

Skript `apxs` tedy zná jméno adresáře s hlavičkovými soubory a programátor může tyto detaily vynechat.

```
/* Dispatch list for API hooks */
module MODULE_VAR_EXPORT datum_module = {
    STANDARD_MODULE_STUFF,
    NULL,                  /* module initializer                  */
    NULL,                  /* create per-dir    config structures */
    NULL,                  /* merge  per-dir    config structures */
    NULL,                  /* create per-server config structures */
    NULL,                  /* merge  per-server config structures */
    NULL,                  /* table of config file commands       */
    datum_handlers,        /* [#8] MIME-typed-dispatched handlers */
    NULL,                  /* [#1] URI to filename translation    */
    NULL,                  /* [#4] validate user id from request  */
    NULL,                  /* [#5] check if the user is ok _here_ */
    NULL,                  /* [#3] check access by host address   */
    NULL,                  /* [#6] determine MIME type            */
    NULL,                  /* [#7] pre-run fixups                 */
    NULL,                  /* [#9] log a transaction              */
    NULL,                  /* [#2] header parser                  */
    NULL,                  /* child_init                          */
    NULL,                  /* child_exit                          */
    NULL                   /* [#0] post read-request              */
};
```

Nyní přeskočíme až na konec souboru `mod_datum.c`. Je zde totiž hlavní struktura, která musí být přítomna v každém modulu. Obsahuje odkazy na další struktury a funkce v modulu. V našem jednoduchém modulu je vyplněna pouze jedna položka a to hodnotou `datum_handlers`, což je odkaz na další strukturu definovanou v modulu:

```
/* Dispatch list of content handlers */
static const handler_rec datum_handlers[] = {
    { "datum", datum_handler },
    { NULL, NULL }
};
```

Ta obsahuje dvojici řetězec (většinou MIME typ nebo libovolný jiný identifikátor) a funkce. Funkce `datum_handler()` je tedy „srdcem“ našeho modulu:

```
/* The sample content handler */
static int datum_handler(request_rec *r)
{
    r->content_type = "text/html";
    ap_send_http_header(r);
    if (!r->header_only)
        ap_rputs("The sample page from mod_datum.c\n", r);
    return OK;
}
```

Funkci `datum_handler()` je možno rozdělit do několika důležitých částí. První příkaz nastaví MIME typ odpovědi na `text/html`. MIME typ samozřejmě záleží na vlastní funkci modulu; pokud bychom měli náhodně generovat obrázek, asi bychom zde uvedli nějaký jiný MIME typ, například `image/gif`.

Druhý příkaz vytvoří hlavičku HTTP odpovědi a zašle ji klientovi. V poslední části si funkce ověří, zda uživatel použil HTTP metodu `HEAD`, a tudíž nemá zájem o vlastní obsah, ale pouze o hlavičku odpovědi. Pokud jej zajímá i obsah (tj. použil-li HTTP metodu `GET`), pošle mu modul jednoduchý text pomocí funkce `ap_rputs()`. Struktura modulu je tedy velmi jednoduchá.

Nyní se pokusíme modul připojit k samotnému serveru a ověřit jeho funkčnost. To jej ale budeme muset přeložit ze zdrojového textu do podoby sdíleného modulu. Tuto práci nám usnadní vytvořený soubor `Makefile`, který obsahuje všechna potřebná pravidla, stačí nám pouze spustit příkaz `make`:

```
[pavel@SnowWhite datum]$ make
apxs -c    mod_datum.c
gcc -DLINUX=2 -DUSE_HSREGEX -DUSE_EXPAT -I../lib/expat-lite -fpic \
    -DSHARED_MODULE -I/home/pavel/Apache/include  -c mod_datum.c
gcc -shared -o mod_datum.so mod_datum.o
[pavel@SnowWhite datum]$
```

Modul by nyní měl být přeložen v souboru `mod_datum.so`. Instalace modulu do adresářové struktury serveru Apache je také velmi jednoduchá a opět nám ji usnadní soubor `Makefile` společně se skriptem `apxs`:

```
[pavel@SnowWhite datum]$ make install
apxs -i -a -n 'datum' mod_datum.so
cp mod_datum.so /home/pavel/Apache/libexec/mod_datum.so
chmod 755 /home/pavel/Apache/libexec/mod_datum.so
[activating module 'datum' in /home/pavel/Apache/conf/httpd.conf]
[pavel@SnowWhite datum]$
```

V tomto kroku byl vytvořený sdílený modul zkopírován do adresáře `libexec` v adresářové struktuře serveru Apache a poté byl do konfiguračního souboru serveru přidán řádek aktivující daný modul:

```
LoadModule datum_module       libexec/mod_datum.so
```

Abychom mohli modul vyzkoušet, musíme ještě v konfiguračním souboru nastavit, které URL bude sloužit k vyvolání modulu. Zvolíme si pro jednoduchost URL `/datum` a přidáme tedy do konfiguračního souboru `httpd.conf` následující řádky:

```
<Location /datum>
    SetHandler datum
</Location>
```

Soubor uložíme a nyní již jenom restartujeme vlastní server. Opět nám práci velmi ulehčí předpřipravené pravidlo `restart` v souboru `Makefile`:

```
[pavel@SnowWhite datum]$ make restart
apachectl restart
/home/pavel/Apache/bin/apachectl restart: httpd restarted
[pavel@SnowWhite datum]$
```

Vlastní webový server běží standardně na portu 8080, a proto náš prohlížeč nasměrujeme na adresu `http://localhost:8080/datum`:

```
[pavel@SnowWhite datum]$ lynx -mime_header http://localhost:8080/datum
HTTP/1.1 200 OK
Date: Sat, 22 Jul 2000 14:19:14 GMT
Server: Apache/1.3.12 (Unix)
Connection: close
Content-Type: text/html

The sample page from mod_datum.c
[pavel@SnowWhite datum]$
```

Na ukázce vidíme jak MIME typ nastavený ve funkci `datum_handler()`, tak i text, který je argumentem funkce `ap_rputs()`.

Konečně se dostáváme k tomu, abychom místo jednoduché anglicky psané věty poslali uživateli aktuální datum. Z výše uvedeného plyne, že bude stačit změnit pouze funkci `datum_handler()`:

```
/* Content handler modulu mod_datum. */
static int datum_handler(request_rec *r)
{
    struct tm *t;
    char buf[100];
    time_t now;

    /* Zjistíme aktuální datum. */
    time(&now);

    /* Převedeme do podoby použitelné ve funkci strftime. */
    t = localtime(&now);
    strftime(buf, 100, "%B %d, %Y", t);

    /* A opět pošleme uživateli zpět. */
    r->content_type = "text/html";
    ap_send_http_header(r);
    if (!r->header_only)
        ap_rprintf(r, "Dnes je %s\n", buf);
    return OK;
}
```

Nyní je nutno modul znovu přeložit a restartovat webový server. K tomu slouží pravidlo `reload` definované v souboru `Makefile`:

```
[pavel@SnowWhite datum]$ make reload
apxs -c    mod_datum.c
gcc -DLINUX=2 -DUSE_HSREGEX -DUSE_EXPAT -I../lib/expat-lite -fpic \
    -DSHARED_MODULE -I/home/pavel/Apache/include  -c mod_datum.c
gcc -shared -o mod_datum.so mod_datum.o
apxs -i -a -n 'datum' mod_datum.so
cp mod_datum.so /home/pavel/Apache/libexec/mod_datum.so
chmod 755 /home/pavel/Apache/libexec/mod_datum.so
[activating module 'datum' in /home/pavel/Apache/conf/httpd.conf]
apachectl restart
/home/pavel/Apache/bin/apachectl restart: httpd restarted
[pavel@SnowWhite datum]$
```

Opět si můžeme ověřit funkčnost našeho modulu:

```
[pavel@SnowWhite datum]$ lynx -mime_header http://localhost:8080/datum
HTTP/1.1 200 OK
Date: Sat, 22 Jul 2000 14:32:39 GMT
Server: Apache/1.3.12 (Unix)
Connection: close
Content-Type: text/html

Dnes je July 22, 2000
[pavel@SnowWhite datum]$
```

Funguje. Téměř skvěle, zapomněli jsme však na to, že náš Linux (resp. knihovna glibc) pracuje standardně v angličtině, a tak je datum napůl česky a napůl anglicky. Využijeme naší chyby k demonstraci toho, jak moduly mohou přebírat z konfiguračních souborů parametry. Vytvoříme dva nové konfigurační příkazy. První z nich (`DatumPozdrav`) bude mít pouze jeden parametr. Bude to řetězec, kterým náš modul uživatele pozdraví. Druhý příkaz (`DatumLocale`) bude sloužit k nastavení proměnné `LC_TIME`,

která ovlivní výsledek funkce `strftime()`. Jak jej ovlivní? Ukážeme si na jednoduchém příkladě, jak je program `date`, který zobrazuje aktuální čas a datum pomocí funkce `strftime()`, ovlivněn nastavením této proměnné:

```
[pavel@SnowWhite datum]$ LC_TIME=cs_CZ date
So červenec 22 22:24:26 CEST 2000
[pavel@SnowWhite datum]$ LC_TIME=de_DE date
Sam Jul 22 22:24:30 CEST 2000
[pavel@SnowWhite datum]$ LC_TIME=sk_SK date
So júl 22 22:24:36 CEST 2000
[pavel@SnowWhite datum]$
```

Program `date` používá stejně jako náš modul pro formátování data a času funkci `strftime()`, a proto se výsledek tohoto příkazu liší podle hodnoty proměnné `LC_TIME`.

Kompletní konfigurace našeho modulu bude poté vypadat například takto:

```
<Location /datum>
    SetHandler datum
    DatumPozdrav "Dnešní datum je:"
    DatumLocale cs_CZ
</Location>
```

Takto nakonfigurovaný modul nám vrátí při požadavku na URL `/datum` následující text:

```
HTTP/1.1 200 OK
Date: Sat, 22 Jul 2000 20:29:56 GMT
Server: Apache/1.3.12 (Unix)
Connection: close
Content-Type: text/html

Dnešní datum je: červenec 22, 2000
```

Pokud změníme v konfiguraci serveru jazyk například na italštinu, můžeme se dozvědět, jak se italsky řekne červenec:

```
<Location /datum>
    SetHandler datum
    DatumPozdrav "Jakpak se asi italsky řekne červenec:"
    DatumLocale it_IT
</Location>
```

Server nám odpoví:

```
HTTP/1.1 200 OK
Date: Sat, 22 Jul 2000 20:36:46 GMT
Server: Apache/1.3.12 (Unix)
Connection: close
Content-Type: text/html

Jakpak se asi italsky řekne červenec: luglio 22, 2000
```

Seznam podporovaných jazyků nám zobrazí program `locale`, pokud je spuštěn s parametrem `-a`.

Zdrojový text našeho modulu může poté vypadat například takto:

```
/*
 * Modul datum - mod_datum.c, verze 0.0.1
 *
 * (c) 2000 Pavel Janík ml. <Pavel@Janik.cz>, http://www.janik.cz/
 *
 * Příklad k článku "Webový server Apache a moduly"
 *
 */
```

```
#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"
#include <locale.h>

/* Konfigurační struktura. */
typedef struct
{
    char *Pozdrav;
    char *Locale;
}
datum_configuration;

module MODULE_VAR_EXPORT datum_module;

/* Obslužná funkce modulu mod_datum. */
static int datum_handler(request_rec *r)
{
    char buf[100];
    struct tm *t;
    time_t now;
    datum_configuration *configuration =
                        ap_get_module_config(r->per_dir_config,
                                             &datum_module);

    /* Zjistíme aktuální datum. */
    time(&now);

    /* Převedeme do podoby použitelné ve funkci strftime. */
    t = localtime(&now);

    /* Podle nastavených locale vytvoř řetězec. */
    setlocale(LC_TIME, configuration->Locale);
    strftime(buf, 100, "%B %d, %Y", t);

    /* A opět pošleme uživateli zpět. */
    r->content_type = "text/html";
    ap_send_http_header(r);
    if (!r->header_only)
        ap_rprintf(r, "%s %s\n", configuration->Pozdrav, buf);

    return OK;
}

static void *
datum_config_for_dir (pool *p, char *dir)
{
    datum_configuration *configuration =
                        ap_pcalloc (p, sizeof(datum_configuration));

    /* Pokud není v konfiguračním souboru specifikováno jinak, zobrazíme datum
       i pozdrav česky. */
    configuration->Pozdrav = ap_pstrdup(p, "Dnes je");
    configuration->Locale  = ap_pstrdup(p, "cs_CZ");
```

```
    return (void *) configuration;
}

/* Funkce pro zpracování příkazu DatumPozdrav. */
static const char *
datum_command_pozdrav (cmd_parms *cmd, void *config, char *arg)
{
    datum_configuration *configuration = (datum_configuration *) config;

    configuration->Pozdrav = ap_pstrdup(cmd->pool, arg);

    return NULL;
}

/* Funkce pro zpracování příkazu DatumLocale. */
static const char *
datum_command_locale (cmd_parms *cmd, void *mconfig, char *arg)
{
    datum_configuration *configuration = (datum_configuration *) mconfig;

    configuration->Locale = ap_pstrdup(cmd->pool, arg);

    return NULL;
}

/* Seznam obslužných funkcí. */
static const handler_rec datum_handlers[] = {
    { "datum", datum_handler },
    { NULL, NULL }
};

/* Seznam konfiguračních příkazů. */
static const command_rec datum_commands[] = {
    { "DatumPozdrav", datum_command_pozdrav, NULL, OR_ALL, TAKE1,
      "Pozdrav uvedený před aktuálním datem." },
    { "DatumLocale",  datum_command_locale,  NULL, OR_ALL, TAKE1,
      "Locale pro datum." },
    { NULL },
};

/* Hlavní struktura modulu. */
module MODULE_VAR_EXPORT datum_module = {
    STANDARD_MODULE_STUFF,
    NULL,                   /* module initializer                  */
    datum_config_for_dir,   /* create per-dir    config structures */
    NULL,                   /* merge  per-dir    config structures */
    NULL,                   /* create per-server config structures */
    NULL,                   /* merge  per-server config structures */
    datum_commands,         /* table of config file commands       */
    datum_handlers,         /* [#8] MIME-typed-dispatched handlers */
    NULL,                   /* [#1] URI to filename translation    */
    NULL,                   /* [#4] validate user id from request  */
    NULL,                   /* [#5] check if the user is ok _here_ */
    NULL,                   /* [#3] check access by host address   */
    NULL,                   /* [#6] determine MIME type            */
```

```
    NULL,                   /* [#7] pre-run fixups                  */
    NULL,                   /* [#9] log a transaction               */
    NULL,                   /* [#2] header parser                   */
    NULL,                   /* child_init                           */
    NULL,                   /* child_exit                           */
    NULL                    /* [#0] post read-request               */
};
```

Modul by se samozřejmě dal ještě vylepšit (např. možnost konfigurace formátovacího řetězce pro funkci `strftime()`, možnost definice znakové sady na výstupu, kontrola zadaných hodnot v konfiguračním souboru apod.), ale pro ilustraci práce s příkazy konfiguračního souboru je dostačující.

Pokud se vám článek líbil, zkuste si napsat vlastní modul a napište nám, nejzajímavější moduly zveřejníme na serveru LinuxWorld.

Shrnutí

V článku je shrnuta problematika modulů nejrozšířenějšího webové serveru na Internetu — serveru Apache. Je nastíněna práce s moduly, vytvoření jednoduchého modulu a zpracování parametrů z konfiguračního souboru serveru. V rámci článku byl vyvinut vzorový modul `mod_datum`, který slouží k zobrazení aktuálního data na serveru v zadaném jazyce.

Pro bližší seznámení s tvorbou modulů doporučuji přečíst zejména zdrojové texty modulu `example`, které jsou součástí serveru Apache, dokumentaci ke sdíleným modulům a API. Vzorové moduly je možno najít na serveru modules.apache.org.

Příloha

Zdrojové kódy modulu mod_datum jsou přílohou tohoto článku.

O autorovi

Autor je nezávislým konzultantem v oboru informačních technologií, specializuje se na Linux, unixové operační systémy a programování Open Source projektů.