LV 4752 / 7363
Web-Engineering/WBA
Übung 02

Installation und Konfiguration eines Web-Servers

Apache 2.x

Lighttpd 1.4.x

Ziele

- Fähigkeit, einen eigenen Web-Server für die Projektarbeiten im Rahmen dieses Kurses einzurichten
  - Keine spezielle Sicherheitsschulung, da nur lokaler Betrieb
  - Keine Hochleistungskonfiguration (hier nicht benötigt)
  - Keine Berücksichtigung von Effizienzfragen bei der Administration (wir wollen hier keine Serverkonsolidierung betreiben)
- Damit bleiben:
  - Installation
    - incl. Kompilierung, im Rahmen der automatisierten Abläufe
  - Konfiguration
    - Grundkenntnisse
  - Protokolle eines Web-Servers
    - Grundkenntnisse
  - Ergänzung eigener Module (!)
    - Hinreichende Kenntnisse zum Einrichten der für den Kurs benötigten Module

Ziele

- Teilziele
  - Fertigstellung einer Entwicklungsumgebung für dynamische Web-Seiten.
  - Vertraut werden mit dem CGI bzw. seinen Alternativen (später)
- Übungen
  - Performance-Vergleich: CGI vs. Alternativen (FastCGI) / Option
  - Umgang mit Formularen (später)
  - Umgang mit SSI und Cookies (eventuell)
- Übergeordnetes Ziel
  - Einübung von Grundlagen-Fertigkeiten für den späteren Einsatz von Rails unter Produktionsbedingungen
  - Vorbereitungen für die Projektarbeit

Vorgehen / Methodik

- Keine „Kochrezepte“
  - Vorgegeben werden die Ziele sowie Informations- und Materialquellen
  - Ihre Aufgabe besteht in wesentlichen Teilen in der Entdeckung des „Wegs“

- Hilfe zur Selbsthilfe
  - Trainieren der Selbständigkeit
  - Hilfestellung bei Bedarf (aber nicht früher)
  - Raum für eigene Experimente und Entscheidungen!

Installation

Apache

„Lighty“

Serverauswahl

- WEBrick
  - \+ Mit Ruby bereits installiert
  - \+ Plattformunabhängig, rein in Ruby geschrieben
  - – Zu langsam für Produktionszwecke
  - – Keine SSL-Unterstützung

- Mongrel(s)
  - \+ Als Ruby Gem leicht nachinstallierbar – ohne Konfigurationen!
  - \+ Schnell genug für Produktionszwecke, da größtenteils in C implementiert
  - \+ Unterstützt von Capistrano
  - \+ Zwecks Lastverteilung kombiniert mit z.B. Apache („a pack of Mongrels“)
  - – Keine (direkte) SSL-Unterstützung

Serverauswahl

- Apache
  - \+ Für alle Einsätze geeignet, incl. SSL-Unterstützung; bewährt & bekannt
  - \+ Als Lastverteiler und Lieferant statischer Seiten mit u. ohne SSL
  - o Für Rails via FastCGI kombinierbar, aber: Qualität des FastCGI-Moduls?
  - \+ NEU: „mod_rails“ bzw. „mod_passenger“ – einfach installierbare, performante und skalierbare Rails-Unterstützung!

- LightTPD („lighty“)
  - \+ Sehr schnell (ca. 25% schneller als Apache bei statischen Seiten)
  - \+ Weniger komplex, schlanker, daher gut im Embedded-Bereich geeignet
  - \+ Einfacher konfigurierbar als Apache
  - \+ FastCGI bereits integriert
  - \+ SSL-Unterstützung
  - \+ „Lighty“ + FastCGI war bis 2007 die für Rails empfohlene Prod.-Plattform
  - – Stabilität?

Serverauswahl

- Empfehlung für Ihr Rails-Projekt

  – Entweder: Apache2 + Phusion Passenger („mod_rails“)

  – Oder: Lighty + FastCGI

  – Während der Entwicklung:
    - Nachteile von Apache und Lighty: Re-starts bei Codeänderungen!
    - Immer wieder den Code von der Entwicklungs- auf die Produktionsinstallation übertragen (von Hand oder z.B. mit capistrano) und auch dort testen

  – Bequemer: Ruby-native Serverkomponenten
    - Sofern kein SSL erforderlich, genügt WEBrick
    - Mongrel in der Testumgebung installieren zur Beschleunigung:

      ```
      $ sudo gem install mongrel
      ```

    Das Rails-Kommando „script/server“ erkennt dann automatisch, of mongrel installiert ist, und startet entsprechend

Installation

- Vorbemerkung
  - Dies ist eine Einzelübung - bitte eine Installation pro Person!
  - Erfahrungsaustausch innerhalb der Teams ist ok.
- Quellen
  - Material in `~werntges/lv/wba/02`, sofern nicht anders erwähnt. Dies sei der Wert von `$srcdir`.
  - Dokumentation: Entweder in den Paketen oder im Internet unter URLs, die z.B. in den Paketen erwähnt sind.
- Apache
  - Apache ("*a patchy server*") in der getesteten Version 2.0.53 oder in der aktuellen Version 2.2.10
  - Dateien: httpd-2.0.53.tar.bz2 oder httpd-2.2.10.tar.gz
- „Lighty“
  - Datei: lighttpd-1.4.20.tar.gz
- Vorbereitungen: **Schaffen Sie Platz!**
  - Löschen unbenutzter Dateien
  - Leeren Ihrer Browser-Caches
  - Archivierung + Kompression noch benötigter Daten

Installation (Apache)

1. Auspacken (Bsp.):

```
$ cd                  # ggf. anpassen
$ tar tvjf $srcdir/httpd-2.0.53.tar.bz2 # Inhalte listen
# alternativ: tar tvxf $srcdir/httpd-2.2.10.tar.gz
# Weiter, wenn ok; sonst Verzeichnis wechseln, dann:
$ tar xjf $srcdir/httpd-2.0.53.tar.bz2
# oder: tar xzf $srcdir/httpd-2.2.10.tar.gz
$ cd httpd-2.0.53 # oder: cd http-2.2.10
```

2. Doku lesen, insbesondere:

```
README, INSTALL, Internet-Seiten zu "Install"
Rufen Sie die Optionen von "configure" ab
```

3. PREFIX festlegen
   - Dies ist der zukünftige Installationsort
   - Legen Sie eine rein lokale Version an, d.h. unter Ihrem Verzeichnis und mit Ihren Berechtigungen.
   - Vorschlag: `$HOME/apache2`

Installation (Lighty)

1. Auspacken (Bsp.):

```
$ cd                # ggf. anpassen
$ tar tvxf $srcdir/lighttpd-1.4.20.tar.gz # Inhalte listen
# Weiter, wenn ok; sonst Verzeichnis wechseln, dann:
$ tar xzf $srcdir/lighttpd-1.4.20.tar.gz
$ cd lighttpd-1.4.20
```

2. Doku lesen, insbesondere:

   `README, INSTALL, Internet-Seiten (http://www.lighttpd.net)`

3. PREFIX festlegen
   - Dies ist der zukünftige Installationsort
   - Legen Sie eine rein lokale Version an, d.h. unter Ihrem Verzeichnis und mit Ihren Berechtigungen.
   - Vorschlag: `$HOME/lighttpd`

Installation (Apache)

4. Probeinstallation
   - Die folgenden Schritte sind bei Bedarf zu wiederholen, ggf. mit verschiedenen Optionen von `configure` und/oder angepassten Umgebungsvariablen

```
$ ./configure --prefix=$HOME/apache2 # bzw. Ihre Wahl
$ make
$ make install
```

5. Bemerkungen
   - Im Idealfall ist der Server nun bereits startklar. Im Allgemeinen muss er aber noch konfiguriert werden, was mehr Arbeit ist als die Installation.
   - Zur Installation neuer Module ist manchmal eine komplette Neuinstallation notwendig. Mit

     ```
     $ make distclean
     ```

     können Sie die ausgepackten Verzeichnisse in den Anfangszustand zurücksetzen.

Installation (Lighty)

4. Probeinstallation
   - Die folgenden Schritte sind bei Bedarf zu wiederholen, ggf. mit verschiedenen Optionen von `configure` und/oder angepassten Umgebungsvariablen

```
$ ./configure --prefix=$HOME/lighttpd # bzw. Ihre Wahl
$ make
$ make install
```

5. PATH-Anpassung

```
$ export PATH=$PATH:~/lighttpd/bin:~/lighttpd/sbin
# Besser: USERPATH in ~/.bashrc analog anpassen!
```

6. Bemerkungen
   - Im Idealfall ist der Server nun bereits startklar. Im Allgemeinen muss er aber noch konfiguriert werden, was mehr Arbeit ist als die Installation.
   - Zur Installation neuer Module ist manchmal eine komplette Neuinstallation notwendig. Mit

     ```
     $ make distclean
     ```

     können Sie die ausgepackten Verzeichnisse in den Anfangszustand zurücksetzen.

Konfiguration

Konfiguration (Apache 2.0.x)

- Wechseln Sie zu `$HOME/apache2`
  - Im Unterverzeichnis `conf` befinden sich verschiedene Versionen einer Konfigurationsdatei.
  - Die Datei "httpd.conf" ist die tatsächlich wirksame. Sie ist anfangs identisch mit "httpd-std.conf"

- Editieren Sie **httpd.conf**
  - Machen Sie sich vertraut mit den Inhalten dieser Datei,
  - diskutieren Sie die Einträge mit Ihrem Projektpartner.
  - Vorgabe: Der Server soll auf "localhost", Port 8888 laufen.
    - Ein ungenutzter Port mit hoher Nummer steht Ihnen auch ohne Adminrechte zur Verfügung. Port 80 ist auch schon belegt.
    - "localhost" ist ausreichend (und sicher) für isolierte Tests. Für teamübergreifende Tests ist statt dessen der Name Ihres Servers zu verwenden, z.B. "lx2-03".
  - Ändern Sie den Eintrag in "Listen" entsprechend.

Konfiguration (Lighty)

- Legen Sie ein Arbeitsverzeichnis für den Betrieb Ihrer Lighty-Installation an, z.B. `$HOME/var/lighttpd`
  - Legen Sie dort Unterverzeichnisse `conf` und `logs` an.
  - „conf“ nimmt die Konfigurationsdatei auf, „logs“ die Protokolldateien für Fehler und allgemeine Zugriffe sowie für die Process ID.

- **lighttpd.conf**
  - Folgen Sie der Anleitung: Legen Sie in „conf“ eine Kopie der Konf.-Datei aus dem Installationsbereich an und passen Sie diese an.
    - Machen Sie sich vertraut mit den Inhalten dieser Datei.
    - Diskutieren Sie die Einträge mit Ihrem Projektpartner.
    - Beschäftigen Sie sich mit den Beispielen und Erläuterungen auf der Projektseite von Lighttpd.
  - Ziel: Umsetzung der folgenden Vorgaben, Testbetrieb

Konfiguration (Lighty)

- **lighttpd.conf: Vorgaben**
  - Der Server soll auf "localhost", Port 8888 laufen.
    - Ein ungenutzter Port mit hoher Nummer steht Ihnen auch ohne Adminrechte zur Verfügung. Port 80 ist auch schon belegt.
    - "localhost" ist ausreichend (und sicher) für isolierte Tests. Für teamübergreifende Tests ist statt dessen der Name Ihres Servers zu verwenden, z.B. "lx2-03".
  - Logs, PID-Datei
    - Verwenden Sie die Standardnamen für diese drei Einträge
    - Konfigurieren Sie als Verzeichnis das soeben angelegte „logs“ (s.o.)
  - Module
    - Aktivieren Sie mindestens folgende Module:
      `mod_rewrite, mod_redirect, mod_access, mod_fastcgi, mod_cgi, mod_compress, mod_ssi, mod_ssl, mod_accesslog`
  - Server-Root
    - Legen Sie das Bezugs-Verzeichnis für HTTP-Request fest
    - Beispiel: $HOME/public_html

Konfiguration

- Bemerkungen
  - Es gibt zahlreiche Gründe für Änderungen der Konfigurationsdatei
  - Die Administration eines Web-Servers besteht zu erheblichen Teilen aus der Optimierung dieser Datei (zumindest in der Aufbauphase).
  - Sie werden daher gelegentlich zu dieser Datei zurückkehren.

Erste Tests

Server starten

- Apache

```
$ cd ~/apache2
$ ./bin/apachectl start
# Fehlermeldungen? Kontrolle der Prozesse:
$ ps -ef  # Laufen httpd-Prozesse unter Ihrem Account?
$ ./bin/apachectl stop
```

  - Tipp:
    - Verwenden Sie "`apachectl restart`" nach Änderungen in httpd.conf

- Lighty

```
cd ~/var/lighttpd/conf
$ lighttpd -t -f lighttpd.conf        # Syntax prüfen
# Ggf. Fehler in conf-Daten beseitigen, dann:
$ lighttpd -f lighttpd.conf
# Fehlermeldungen? Kontrolle der Prozesse:
$ ps -ef  # Laufen httpd-Prozesse unter Ihrem Account?
$ kill `cat ../logs/lighttpd.pid`     # Prozess beenden
```

Erste Tests (Apache)

- Erste statische Dokumente abrufen
  - Server starten
  - Rufen Sie den URL "http://localhost:8888/" bzw. "/" ab.
  - Verwenden Sie dazu anfangs "telnet" wie in der Vorlesungs-Demo, dann Ihren Browser.
  - Rufen Sie dann einen nicht vorhandenen URL ab, etwa "http://localhost:8888/nosuchfile".
  - Erklären Sie, woher die Antworten stammen.
    - Das ist bei Apache komplizierter als zunächst vermutet!
  - Ändern Sie diese Antworten testweise (zum Beweis)!

- Lighty
  - Sollten Sie noch keine HTML-Datei zum Testen zur Verfügung haben, richten Sie eine in Ihrem „server-root“-Verzeichnis ein.
  - Verwenden Sie z.B. die HTML-Seiten aus Ihrer Praktikumsübung zu (X)HTML aus „Einführung in die Informatik“, oder kopieren Sie eine der FH-Seiten als Grundlage für eigene Varianten.

Erste Tests (nur Apache)

- Erste virtuelle Dokumente abrufen
  - Fordern Sie "/" erneut an, diesmal aber **auf deutsch** (wenn Sie es vorhin auf englisch erhielten, sonst umgekehrt)
  - Nutzen Sie dazu HTTP-Header zur Angabe der von Ihnen bevorzugten Sprache!
  - Was macht Ihr Server dabei?
    - Stichwort in der Apache-Doku: „Content negotiation“
  - Finden Sie Hinweise auf den Mechanismus der Dokumentenzuordnung in httpd.conf?

Erste Tests

- Die Protokolldateien:
  - **`logs/access.log`**
  - **`logs/error.log`**
  - **`httpd.pid`** bzw. **`logs/lighttpd.pid`**
- Sichten Sie die Inhalte
  - Erklären Sie deren Zustandekommen.
- Tipp:
  - Verwenden Sie zum Debugging immer diese Dateien.

Einbau von Zusatzmodulen

... am Beispiel FastCGI

Zusatzmodul FastCGI (Apache)

- Quelle
  - `$srcdir/mod_fastcgi-2.4.2.tar.gz.` (oder neuer)
- Installation
  - Quellpaket entpacken, in das entstehende Unterverzeichnis mod_fastcgi-2.4.2.tar.gz wechseln.
  - Anweisungen in INSTALL.AP2 des Pakets befolgen
    - Makefile anpassen (Installationsverzeichnis / „top_dir“!)
- Konfiguration
  - Legen Sie ein Verzeichnis "fcgi-bin" an, analog zu "cgi-bin" (gleiches Elternverzeichns, gleiche Rechte)
  - In httpd.conf:
    - Eintrag „LoadModule …“ gemäß Anleitung einfügen
    - Einträge „ScriptAlias“ und „Directory“ für fcgi-bin anlegen, Fälle für „cgi-bin“ als Vorbilder nehmen.
  - Neues Verzeichnis mit "fastcgi-script" assoziieren:
    - <Location /fcgi-bin> SetHandler fastcgi-script </Location>

Zusatzmodul FastCGI (Apache)

- Installation von FCGI-Anwendungsprogrammen
  - FCGI-Skript/-Programm in „fcgi-bin“ kopieren
  - Execute- und Leserechte!
  - Programm bei Apache anmelden – Eintrag in httpd.conf:
    - `AppClass` *path-to-my_fcgi_module.fcgi*

- Wirkung:
  - Bereits beim Start des Servers kann Apache die angemeldeten FCGI-Module als Prozesse starten
  - Bei Aufruf per URL sind die Prozesse bereits in Betrieb und initialisiert, können also direkt mit der Aufgabe beginnen.
  - Beachte: Es gibt mehrere FCGI-Betriebsarten:
    - Statische Anwendungen (Direktive „FastCgiServer“)
    - Dynamische Anwendungen (Direktive „FastCgiConfig“)
    - Externe Anwendungen (Direktive „FastCgiExternalServer“)

- Dazu später mehr …

Zusatzmodul FastCGI (Lighty)

- Installation
  - Dieses Modul ist bereits Teil von Lighttpd und muss nicht separat installiert werden.
- Konfiguration
  - Legen Sie ein Verzeichnis "fcgi-bin" an, analog zu "cgi-bin" (gleiches Elternverzeichns, gleiche Rechte)
  - In lighttpd.conf:
    - Eintrag „fastcgi.server“ ent-kommentieren und ausfüllen
    - Für jedes zu startende FCGI-Script einen Eintrag wie folgt einrichten:

```
"anmelden.fcgi" =>
( "localhost" =>
  (
     "socket" => "/tmp/anmelden.socket",
     "bin-path" => "/pfad/zu/ihrem/anmelden.fcgi"
  )
)
```

Zusatzmodul FastCGI (Lighty)

- Installation von FCGI-Anwendungsprogrammen
  - FCGI-Skript/-Programm in „fcgi-bin“ kopieren
  - Execute- und Leserechte!
  - Programm bei Lighttpd anmelden – Eintrag in lighttpd.conf, s.o.

- Wirkung:
  - Bereits beim Start des Servers kann Lighty die angemeldeten FCGI-Module als Prozesse starten (auch mehrere zur Lastverteilung!)
  - Bei Aufruf per URL sind die Prozesse bereits in Betrieb und initialisiert, können also direkt mit der Aufgabe beginnen.

Einbau weiterer Module

Weitere Module

- Ausblick (vorläufig nicht benötigt): Lastverteilung
  - Das Apache-Modul „mod-proxy-balancer“
  - Das Lighty-Modul „mod-proxy“

- Hinweis für Rails-Anwender
  - Mongrel: Eine schnelle HTTP-Bibliothek + Server für Ruby
    - In Ruby geschrieben + C-Extension, schneller als WEBrick, ohne fcgi
  - Mongrel (http://mongrel.rubyforge.org/“) ist inzwischen recht beliebt als Server für Rails
  - Er lässt sich per Lastverteilungsmodul gut mit httpd bzw. lighttpd kombinieren:
    - Die „schnellen“ Webserver Apache bzw. Lighty liefern statische Seiten, Dateidownloads, Bilder etc.
    - Sie übernehmen ebenfalls https-Austausch (nicht von Mongrel unterstützt)
    - Dynamische Seitenabrufe werden zwecks Lastverteilung an eine Reihe von Mongrel-Prozessen durchgereicht.

Weitere Module

- **Nur Apache2: Phusion Passenger („mod_rails“)**
  - Noch neu: V 1.0.1 erschien 04/2008, aktuell ist V 2.0.3 08/2008
  - Ein Ruby-Interpreter integriert in einem Apache-Modul, analog zu mod_perl, mod_php etc.
  - Quellen:
    - Projektseite: http://www.modrails.com,
    - Sourcecode: http://rubyforge.org/frs/?group_id=5873
  - Sehr einfache Konfiguration
    - „DocumentRoot“ einfach auf „public“-Verzeichnis des Rails-Projekts lenken
    - Einfacher Neustart des Server-Moduls nach Änderungen:

```
$ touch restart.txt        # im public-Ordner des Rails-Projekts
```

  - Vorsicht – gut genug für unser Projekt, aber wahrscheinlich noch nicht so ausgereift wie andere Apache-Module.
  - Lastverteilung auf mehreren Servern? Noch zu klären …