Linux - Friheden til egen webserver

Version 2.5.20040516 - 2020-12-31

Peter Toft

Hans Schou

Linux - Friheden til egen webserverVersion 2.5.20040516 - 2020-12-31

af Peter Toft og Hans Schou

Skrevet af mange Linux-brugere til nye brugere, som vil hurtigt igang med at bruge Linux som web-server.

Indholdsfortegnelse

Tabelliste

Forord

1. Linux-bøgerne

Bogen er en del af en serie, som kan findes på http://www.linuxbog.dk/

- *Linux – Friheden til at vælge installation* – Om at installere Linux.
- *Linux – Friheden til at lære Unix* – Om hvordan man bruger Linux' (og Unix') kommandolinjeværktøjer.
- *Linux – Friheden til at vælge grafisk brugergrænseflade* – Om alle de grafiske brugergrænseflader, der findes til Linux.
- *Linux – Friheden til at vælge programmer* – Om de programmer du kan få til Linux.
- *Linux – Friheden til systemadministration* – Om at administre sit eget linuxsystem.
- *Linux – Friheden til at programmere* – Programmering på Linux
- *Linux – Friheden til at programmere i C* – Om at programmere i sproget "C".
- *Linux – Friheden til at programmere i Java* – Om at programmere i sproget "Java".
- *Linux – Friheden til sikkerhed på internettet* – Om at sikre dit Linuxsystem mod indbrud fra internettet.
- *Linux – Friheden til egen webserver* – Om at sætte en webserver med databaser, CGI-programmer og andet godt op.
- *Linux – Friheden til at skrive dokumentation* – Om at skrive dokumentation (og andet) i SGML/DocBook, LaTeX eller andre formater.
- *Linux – Friheden til at vælge kontorprogrammer* – Kontorfunktioner på et Linux/KDE/OpenOffice.org-system.
- *Linux – Friheden til at vælge IT-løsning* – Om muligheder, fordele og ulemper ved at bruge Linux i sin IT-løsning.
- *Linux – Friheden til at vælge OpenOffice.org* – Om at bruge OpenOffice.org, både på Linux og på andre styresystemer.
- *Linux – Friheden til at vælge digital signatur* – Digital signatur på Linux.

2. Ophavsret

Denne bog er skrevet af Linux-brugere til Linux-brugere. Store dele af bogen er skrevet eller redigeret af enkelte forfattere, hvilket er nævnt i revisions-historien til bogen.

Bogen kan findes i opdateret form på http://www.linuxbog.dk/, mens prøve-udgaver kan findes på http://cvs.linuxbog.dk/.

Figur 1. ÅDL

Kommentarer, ris og ros og specielt fejl og mangler bedes sendt til linuxbog@sslug.dk (mailto:linuxbog@sslug.dk), men er du medlem af SSLUG kan du i stedet for med fordel skrive til sslug-bog@sslug.dk (mailto:sslug-bog@sslug.dk).

3. Om forfatterne og bogens historie

Bogen startede som et bogprojekt i sommeren 1998, hvor Peter Toft, Kenneth Geisshirt og Snebjørn Andersen fik skrevet "Linux - friheden til at vælge". Bogen var i starten henvendt mod Linux-installation, at få lidt Unix-kendskab og noget om netværk. Bogen har altid været en fri "web-bog", der kunne læses enten i smådele eller som en helhed på http://www.sslug.dk.

Vi, forfatterne, har anvendt Linux i flere år og har stor erfaring med det. Vi er medlemmer af Skåne Sjælland Linux User Group (SSLUG), som er en uafhængig og non-profit svensk/dansk organisation, der søger at udbrede kendskabet til Linux. Det er gratis at være medlem af SSLUG, og organisationen findes på internettet på http://www.sslug.dk Vi har skrevet denne bog fordi den kunne være til glæde for andre.

Den oprindelige bog "Linux - Friheden til at vælge" blev udgivet som hæfte på 96 sider fra IDG Forlag i februar 1999 med en Red Hat 5.2 cd-rom, hvor den i lang tid lå på bestsellerlisten for edb-hæfter. I december 1999 udgav Forlaget Globe version 2.9 i bogform (med en del ærgerlige konverteringsfejl - se http://www.sslug.dk/misc/globe_fejl.html) på over 300 sider og med tre cd-rom'er (Red Hat 6.1, SuSE 6.2 og StarOffice 5.1).

I juli 2000 var bogen vokset til ca. 350 sider og blev delt ud til flere bøger. Alle kan læses på www.linuxbog.dk (http://www.linuxbog.dk/).

Fra version 1.0 af denne bog er det Hans Schou og Peter Toft, som vedligeholder bogen sammen med de mange, som skriver på dele af bogen.

4. Vi siger tak for hjælpen

Vi har haft stor glæde af mange SSLUG-medlemmers støtte, rettelser og forslag til forbedringer - bliv ved med dette. Specielt vil vi nævne:

- Jacob Laursen har hjulpet meget med at få lavet splittet af "Linux - Friheden til at vælge" på en god måde.
- Frank M.G. Jørgensen har skrevet om MySQL.
- Flemming Mahler Larsen og Dennis Lauersen har skrevet et afsnittet om webservere.
- Michael Rasmussen har skrevet kapitlet om PHP og eksemplerne der til.
- Carsten Svaneborg har skrevet et afsnittet om CGI.
- Gitte Wange har skrevet afsnittet om Zope.
- Rolf Larsen (gentagne gange) og Steen Jensen for at have læst bogen fra ende til anden og aflevere en stribe rettelser til hele bogen.
- Andre bidragydere er: Bjørn Bækkegaard, Christian Treldal, Keld Simonsen, Michael Lykke, Brian Hemstedt, Jacob Thamsborg, Søren Sjørup, Jens Stavnstrup, Poul Petersen, Palle E. Nielsen, Johan Myhre Andersen, Jan Michael Due, Kåre Løvgren, Morten Jensen, Jesper Krogh, Michael Skaarup, Erik Andresen, Jesper Louis Andersen, Flemming Bjerke, Niels Rasmussen, Kim Larsen, Anders Melchiorsen, Anders Damkjær Møller, Peter Andersen, Peter Christensen, Jørgen Ramskov, Christoffer Hall-Frederiksen, Henning Schou, Mads Tofte, Magnus Østergaard, Jon Loldrup, Svend Erik Venstrup-Nielsen, Morten Christensen, Jens Axelsen, Tue Hansen, Anders Pedersen, Jakob Hilarius, Martin Stenderup, Jan Hemmingsen, Flemming Mahler Larsen, René Seindal, Christian Borup, Finn Dorph-Petersen, Kim Futtrup Petersen, Peter B. Kvan, Brian Christensen, Jørgen Ramskov, Kevin Ilchmann Jørgensen, Stefan Klukowski, Henrik Johansen, Rask Ingemann Lambertsen, Kristian Støchkel, Jan Vestergaard Larsen, Niels Damgaard, Frank Damgaard, Henrik Lundquist, Poul Petersen, Anders Ebbesen, Allan Jacobsen, Jacob Sparre Andersen, Jesper Møller, Henrik Christian Grove, Anders Melchiorsen, Lars Scheele Jensen, Niels Kristian Bech Jensen, Jørgen I. Østergaard, Thomas Mørch, Olav Pedersen, Michael Jacobsen, Jens Bygum, Martin Lykke, Dan Windekilde, Jakob Hilmer, Morten Olsen, Niels Sandmann, Palle Arentoft, Peter Seidler, Poul-Erik Hansen, Thomas Petersen, Svend Erik Venstrup-Nielsen, Ingvar Blychert, Rune Tønnesen, Niels Damgaard, Jacob I. Christensen, Allan Olesen, Gunner Poulsen, Rune Tonnesen, Birger Langkjer, Danni Finne, Erik Søe Sørensen, Carsten Nordstrøm Jensen, Benny Bøtchiær Thomsen, Klaus Hebsgaard og Knud Haugaard Sørensen.

Du kan i Appendiks A finde en liste over alle de revisioner, som bogen har været igennem.

Hvis du har ord du ikke forstår, så kan http://www.whatis.com være interessant. Her kan du slå mange computerord op dog kun på engelsk.

5. Typografi

Vi vil afslutte indledningen med at nævne den anvendte typografi.

- Navne på filer og kataloger skrevet som `foo.bar`
- Kommandoer, du udfører ved at taste, skrives som **help**

- Der er flere steder i bogen, hvor vi viser, hvad brugeren taster, og hvad Linux svarer. Det vil se ud som:

```
[tyge@hven tyge]$ Dette taster brugeren
Dette svarer Linux.
```

- Der er tilsvarende flere steder i bogen, hvor vi viser, hvad systemadministratoren (brugeren "root") taster, og hvad Linux svarer. Det vil se ud som:

```
hven# Dette taster systemadministratoren
Dette svarer Linux.
```

  Det vigtige her er at kommandofortolkeren bruger nummertegnet (#) til at markere at man har systemadministratorrettigheder.

Kapitel 1. Webserver

I dette afsnit skal vi se nærmere på webservere - i praksis Apache, som i dag anvendes af ca. 70% af alle webservere på internettet.

Når du har fået din Linux-server op at køre, er en af de muligheder du kan gå i gang med, at installere en webserver. Du kender sikkert allerede webservere, i det de anvendes til websteder, intranet, ekstranet og meget andet.

Her kan du læse hvordan du selv kommer i gang med at være webmester på din egen webserver.

1.1. Hvad er en webserver?

En webserver var oprindelig en rimelig simpel server, hvis eneste funktion i livet var at sende filer fra sin harddisk til den klient, der bad om dem.

Der er sidenhen kommet et væld af ekstra funktionaliteter og faciliteter i webservere, så det i dag også er det en del af webserverens job at indsætte kode i sider når en bruger beder om den, f.eks. sidste gang filen blev rettet (Server Side Includes). Webserveren kan udføre programmer og returnere uddata i form af tekst, billed, lyd og animationer til browseren. Når man udfylder og indsender data i formularer på nettet, så er det også webserverens job at modtage data og filer, og at sende disse data videre til programmer, der bearbejder dem (CGI-programmer). Webserveren kan også udføre programmer indlejret i websiderne, så man f.eks. kan indsætte data fra en database direkte i sider (Perl/PHP/Python).

1.1.1. Lidt historie bag Apache

Apache serveren startede sit liv som et "patch-kit" (et sæt ændringer) til NCSA-serveren, og det er da også der navnet oprindeligt kommer - A PAtCHy Server.

Da udviklingen af NCSA's webserver stoppede, fortsatte udviklingen af Apache, og således er Apache-serveren nu blevet den markedsledende webserver.

Apache ejes og styres af Apache Software Foundation og er frit og gratis.

Apache er i skrivende stund nået til version 2.0.43. Apache er en af de mest stabile webservere. Dens markedsandel på ca. 70% af internettet er vist det mest sigende bevis på hvor god den er. De første versioner af Apache 2.0 er frigivet, mens 1.3-serien som p.t. er den mest udbredte kun får rettet fejl. Sammenlignet med 1.3-serien er 2.0-serien langt mere modulær. Denne bog dækker både Apache 1.3 og Apache 2.0.

Har du brug for en meget hurtig webserver (Apache er hurtig), så findes Tux-1.0, som er tunet meget kraftigt til en performance, der oftest ligger langt over en tilsvarende MS IIS.

1.2. Apache-installation

Når du skal installere en Apache webserver, skal du sikre dig, at du har seneste udgave. Du kan enten vælge at installere den gennem dit systems pakkehåndtering (Debian- eller RPM-pakker) eller oversætte den selv fra kildeteksten. Som udgangspunkt anbefales det, at du installerer fra kildeteksten, da det ikke er svært og det giver dig mulighed for selv at bestemme hvilke moduler, der oversættes ind i Apache.

1.2.1. Apache installation fra pakker

Du kan normalt finde pakker til Red Hat, Debian og andre Linux-distributioner med Apache. At anvende pakker kræver normalt at du har systemadministratorrettigheder på maskinen, du vil installere serveren på. Hvis du vælger at benytte en sådan, vil du normalt blot få en "standard Apache" (alt efter pakken), som installeres i f.eks. `/usr/local/apache` og som kører på port 80.

I Debian mv. installeres de binære filer under `/usr/sbin/` og opsætningsfiler findes i `/etc/apache/`. I Debian startes og stoppes Apache bedst hvis man anvender programmet i /etc/init.d/debian, der også kan anvendes til at genindlæse en opdateret opsætningsfil.

I Red Hat installeres Apache via (RedHat < 8.0) `apache*.rpm` (Redhat >= 8.0) `httpd*.rpm`, som følger med distributionen. Skal du have PHP4 med så skal du dernæst installere `mod_php-4.0*.rpm` eller `php-4.0*.rpm` afhængig af distributionen. Webserveren sættes op til at din dokument-root (der hvor filer gemmes) i `/var/www/html/`.

Dette kan være en løsning, hvis du blot har brug for at se hvorledes dine dokumenter ser ud når de kommer fra en browser, men hvis du for alvor vil lære Apache at kende, og vil være sikker på hvad der faktisk er oversat ind i serveren, er det ofte en god ide at installere den direkte fra kildeteksten.

1.2.2. Apache installation fra tar-pakker

For at installere (og anvende) Apache, behøver du ikke systemadministratorrettigheder på maskinen. Der vil blot være nogle begrænsninger af hvor du kan installere den og hvilke porte den kan køre på. Hvis du ikke vil installere den som systemadministrator, skal serveren køre på et portnummer over 1024 (ofte anvendes 8000, 8080 eller 8088).

Du kan hente seneste udgave af Apache serveren fra http://www.apache.org/dist/ (eller en nærmere spejling). Filen du skal bruge hedder `apache_<versionsnummer>.tar.gz`.

Filen pakkes ud med kommandoen **tar xzvf apache_<versionsnummer>.tar.gz**. Nu har du hele kildeteksten liggende i et katalog der hedder **apache_<versionsnummer>**. Gå ind i biblioteket (**cd apache_<versionsnummer>**).

At få installeret og startet en Apache webserver kræver normalt kun følgende trin:

```
[root@hven /root]# ./configure --prefix=PREFIX
[root@hven /root]# make
[root@hven /root]# make install
[root@hven /root]# PREFIX/bin/apachectl start
```

I linje 1 defineres opsætningen af Apache. Her er det kun hvor den installeres, der ændres. I stedet for PREFIX skal du skrive stien til det bibliotek hvor du vil have den installeret (f.eks. `/home/myuser/apache`). Apache har også nogle foruddefinerede præfikser såsom Red Hat-layout. I stedet skrives så **--with-layout=RedHat**. Se `config.layout` for andre layouts. Du kan til **configure**-kommandoen også angive en lang række andre parametre, som ændre på hvilke moduler der installeres, hvilken bruger den vil køre som osv. Du kan ændre på en stor del af de samme parametre efter Apache er blevet oversat og installeret.

Linje 2 bygger selve webserveren. På en standard Linux-maskine vil der normalt ikke ske andet end at hele byggeprocessen skrives ud på skærmen.

Linje 3 installerer serveren. Hvis du har valgt standardinstallationen skal du skifte til brugeren "root" for at installere den. Det samme gælder hvis du har valgt at installere Apache et sted hvor din bruger ikke har skrive adgang (trin 1 og 2 kan sagtens udføres uden systemadministratorrettigheder).

Linje 4 starter webserveren. Inden du gør dette vil du dog normalt skulle ændre i opsætningsfilerne, hvilket er beskrevet nærmere i næste afsnit. Hvis du starter den uden yderligere ændringer, kan du normalt finde din nye webserver på adressen: http://localhost:8080/. Husk igen at erstatte PREFIX med stien til installations-biblioteket.

1.3. Opsætning af Apache

Apaches opsætningsfiler kan du finde i `PREFIX/conf`. Den fil du skal have fat i er `httpd.conf`. På Red Hat og SuSE-systemer finder du filen i `/etc/httpd/conf/`. Denne fil indeholder et hav af valgmuligheder, så vi vil kun lige berøre de mest basale her. I Red Hat 8.0 findes desuden en række opsætningsfiler i `/etc/httpd/conf.d/` f.eks. php.conf og perl.conf.

1.3.1. Port

Denne parameter angiver hvilken port Apache skal lytte på. Normalt kører en webserver på port 80, hvilket den dog kun kan hvis du har installeret den som "root". Hvis du har installeret Apache som standard bør du nok vælge port 80.

1.3.2. Logfiler

Apache laver to logfiler som udgangspunkt — en tilgangslog, som fortæller hvem der har besøgt din server, og en fejllog, som fortæller dig om alle fejl der opstår mens serveren kører. Hvilke oplysninger, der står i din tilgangslog afhænger af dels hvilke du sætter serveren til at skrive og dels hvilke oplysninger klienten vil aflevere. Det første sætter du op i et LogFormat-direktiv. I standard `httpd.conf`-filen er der lavet fire skabeloner (combined, common, referer og agent), men du kan naturligvis også lave dit eget format.

For fejlloggen kan du også bestemme hvor alvorlige begivenheder skal være førend de skrives ned - dette defineres på 8 forskellige niveauer (fra debug til emergency).

En tredje type logfil som kan være meget nyttig hvis du vil afvikle CGI-programmer på din webserver, er ScriptLog. Denne logfil vil føre en log over de fejlbeskeder dine CGI-programmer afgiver. Det er dog vigtigt at der sættes en maksimum på størrelsen af denne logfil, da den ellers hurtigt kan blive for stor. Dette kan gøres med parameteren ScriptLogLength. Eksempel på anvendelse:

```
ScriptLog PREFIX/log/scriptlog
ScriptLogLength 5192880
```

Dette fortæller Apache at den skal lave en logfil der hedder scriptlog, og at denne fil må maksimalt være 5 Mb.

1.3.3. Options

Options er en af de vigtige direktiver at kende, idet den sætte nogle overordende regler for det område den gælder for (Ved Files-, Directory- og Location-direktiverne kan man have forskellige options, der gælder for forskellige dele af serveren).

De vigtigste parametre til Options er:

- Indexes - som bestemmer hvorvidt man må se indhold af kataloger uden en indeks-fil.
- ExecCGI - som bestemmer om det er muligt at køre CGI-programmer.
- FollowSymLinks - som bestemmer om symbolske links i området skal følges

1.3.4. Files, Directory og Location

Gennem Files- og Location-direktiverne kan du påføre specielle regler for dele af din webserver. Dette kunne f.eks. være at et givet område kun kunne ses, hvis man sad på serveren selv eller ligende.

Location anvendes til at implementere regler relativt i forhold til deres URL. Alt i området `/spoing/` må kun ses fra .dk-domæner eller ligende.

Directory anvendes til at implementere regler, men tager udgangspunkt i filsystemet. Således ville man f.eks. kunne implementere en regel der beskyttede alt der ligger i `/home/web/htdocs/spoing/`, så det kun kan ses fra .dk-domæner.

Files kan anlægge filtre der gælder på fil-niveau. Typiske anvendelser af Files, kunne f.eks. være at sikre at dine `.htaccess`-filer og opsætningsfiler ikke kan hentes af almindelige brugere. Hvis du bruger emacs vil den ofte efterlade sikkerhedskopier der hedder `FILNAVN~`. Disse kan man også sikre ikke kan ses gennem webserveren.

Det er muligt at kombinere Files med enten Directory eller Location, således vil en given regel kun gælde for nogle filer i et givet område. Et eksempel kunne være:

```
<Directory />
      Options +ExecCGI +FollowSymLinks
      <Files ~ "\.cgi$">
      SetHandler cgi-script
      </Files>
</Directory>
```

Her tillades at der køres CGI-programmer, hvis de har endelsen `.cgi`, overalt på maskinen, mens alle filer med en anden endelse vil blive leveret som normalt.

CGI står for "Common Gateway Interface". Det er en veldefineret måde at udføre scripts på webserveren, oftest ud fra brugerens id, eller ud fra brugerens menu-valg. Der kan hentes masser af information om CGI på http://cgi.resourceindex.com.

1.3.5. Adgangskode til websider

Med Apache er det muligt at sætte adgangskode på udvalgte dele af et websted hvis modulet `mod_access` er installeret. Dette kan bruges til at sikre at kun familie, venner og kollegaer får adgang til nogle bestemt sider på dit websted. Første gang adgangskode skal sættes op er det lidt besværligt at få overblikket. Derfor har vi sat en realistisk side op og taget filerne direkte der fra. Følgende filer indgår i opsætningen af adgangskode.

- `.htaccess` - denne fil ligger i det katalog der skal beskyttes

- `/home/chlor/htuser` - filen med brugernavne
- `/home/chlor/htgroup` - eventuelt en fil med grupper
- `/etc/httpd/conf/httpd.conf` - Apaches opsætningsfil (på Red Hat)

Først skal der oprettes en fil med få brugere. For nemheds skyld kaldes den ene bruger for 'demo' med adgangskoden 'demo'. Man kan selv vælge hvad filen skal hedde og hvor den skal ligge. Her hedder bruger-filen `/home/chlor/htuser`. Ved første bruger skal filen oprettes med **-c**. Herunder oprettes brugeren 'demo' og 'john'.

```
[chlor@sslug]$ htpasswd -c /home/chlor/htuser demo
[chlor@sslug]$ htpasswd /home/chlor/htuser john
```

Pas på ikke at bruge kommandoen **-c** anden gang, for så slettes filen. Filen med de to brugere ser nu således ud:

```
demo:2iYw72d6bnpxg
john:..HEJdavUser2
```

Begge adgangskoder er krypteret med 'crypt', så de kan ikke læses. Brugeren 'john's adgangskoder er ikke det du ser, men noget meget kryptisk. Med brugerfilen oprettet, kan den første `.htaccess` oprettes. I dette eksempel ligger filen i `/home/chlor/public_html/htaccess/` og ser således ud:

```
AuthType Basic
AuthName "Beskyttet webside"
AuthUserFile /home/chlor/htuser
Require valid-user
```

AuthType er sat til **Basic** da det er det enkleste. **AuthName** er den tekst, der vises når man skal logge ind på siden. Det nemmeste er at se på eksempel her: http://www.sslug.dk/~chlor/htaccess/ hvor du samtidigt kan prøve at logge ind. Alle brugere der er nævnt i `/home/chlor/htuser` har adgang ved at bruge kommandoen **Require valid-user**.

Dernæst ønsker vi et andet subdir hvor kun 'demo' og 'john' har adgang. Dette kan gøres enten ved at skrive alle de brugere der har adgang, eller ved at oprette en gruppe fil. Det nemmeste er at skrive navnene, og så ser filen `/home/chlor/public_html/htaccess/demojohn/.htaccess` således ud:

```
AuthType Basic
AuthName "Beskyttet demojohn"
AuthUserFile /home/chlor/htuser
Require user john demo
```

Har man mange forskellige sub-dir der nemmest administreres gruppevis, oprettes en gruppefil med navnet `/home/chlor/htgroup` hvor gruppen hedder **fortrolig** med følgende indhold:

```
fortrolig: john demo
```

Et nyt sub-dir oprettes hvor kun denne gruppe har adgang, i dette eksempel `/hemmelig/`. En ny `.htaccess` oprettes i `/home/chlor/public_html/htaccess/hemmelig/` med følgende indhold:

```
AuthType Basic
AuthName "Beskyttet gruppe"
AuthUserFile /home/chlor/htuser
AuthGroupFile /home/chlor/htgroup
Require group fortrolig
```

Ovenstående kan afprøves her: http://www.sslug.dk/~chlor/htaccess/hemmelig/. Er du allerede logget ind på siden med 'demo', må du enten prøve med programmet **lynx** eller lukke din browser ned og starte den igen for at blive spurgt om adgangskode igen, samt se teksten "Beskyttet gruppe".

I det sidste eksempel på en `.htaccess` fil, er det kun brugeren 'john' der kan udføre en **POST** kommando, hvor alle andre brugere kun kan udføre en **GET**. Dette kan bruges i et sub-dir hvor det kun er 'john' der må bruge en html-side med en submit-knap der udfører en **POST**.

```
AuthType Basic
AuthName "Beskyttet, kun john må skrive"
AuthUserFile /home/chlor/htuser
AuthGroupFile /home/chlor/htgroup

<LIMIT GET>
Require valid-user
</LIMIT>

<LIMIT GET POST>
Require user john
</LIMIT>
```

For at aktivere adgangskoder skal Apaches opsætningsfil `/etc/httpd/conf/httpd.conf` (eller `/etc/apache/httpd.conf` på Debian) rettes fra:

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/html">
....
AllowOverride None
....
</Directory>
```

til:

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/html">
....
AllowOverride All
```

```
....
</Directory>
```

1.3.6. Virtuelle værter med Apache

En Linux-maskine med Apache kan nemt være webserver for mange domæner på en gang. Man opretter såkaldte virtuelle værter på maskinen. Har du en DNS-server skal du sørge for at alle dine virtuelle webservere er oprettet med samme IP-adresse. Har du ikke DNS, så sørg for at dine `/etc/hosts`-filer har linjer med samme IP-nummer og hver sit webserver maskinnavn. Eksempel for et hjemmenetværk:

```
192.168.1.1     www.hjemmenet www
192.168.1.1     test.hjemmenet test
```

Med mindre du anvender NAT (bag en firewall) så kan du se din IP-adresse med en browser på http://myip.dk.

Dernæst skal webserveren sættes op, så de to virtuelle webservere tager fat i hver deres kataloger på harddisken. Lad os i eksemplet anvende `/home/www/www.hjemmenet` og `/home/www/test.hjemmenet`. Man kan tænke på at de to steder er til den egentlige webserver, og `/home/www/test.hjemmenet` er et test-område til nye hjemmesider.

Man skal redigere `httpd.conf` (Red Hat `/etc/httpd/conf/httpd.conf`). Led efter følgende tekst

```
### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your virtual host
# configuration.
```

Hvis webserveren ikke skal svar på alle boksens IP-adresser indsætter du IP-adressen på webserveren under `NameVirtualHost`, ellers sætter du en stjerne som markerer at den skal svare på alle IP-adresser.

```
NameVirtualHost *
```

eller

```
NameVirtualHost 192.168.1.1
```

```
<VirtualHost *>
  ServerName www.hjemmenet
  DocumentRoot /home/www/www.hjemmenet
```

```
  <Directory "/home/www/www.hjemmenet/">
    Options Indexes FollowSymLinks Includes
    AllowOverride None
    Order allow,deny
    Allow from all
    XBitHack full
  </Directory>
  DirectoryIndex index.shtml index.html index.htm index.php

  ServerAdmin root@localhost
  ErrorLog logs/www-error_log
  CustomLog logs/www-access_log common
</VirtualHost>

<VirtualHost *>
  ServerName test.hjemmenet
  DocumentRoot /home/www/test.hjemmenet
  <Directory "/home/www/test.hjemmenet/">
    Options Indexes FollowSymLinks Includes
    AllowOverride None
    Order allow,deny
    Allow from all
    XBitHack full
  </Directory>
  DirectoryIndex index.shtml index.html index.htm index.php

  ServerAdmin root@localhost
  ErrorLog logs/test-error_log
  CustomLog logs/test-access_log common
</VirtualHost>
```

Når du genstarter Apache så kan det være at du ikke kan gemme logfilerne (`logs/www-error_log`). Se efter *ServerRoot* i `httpd.conf` og lav kataloget/katalogerne til logfilerne. Eksempel

```
ServerRoot /etc/apache
```

og da *ErrorLog* og *CustomLog*, så skal man lige som root skabe kataloget til log-filerne:

```
[root@hven /root]# mkdir /etc/apache/logs
```

Som det kan ses har vi tilladt at man følger links "FollowSymLinks" (pas på hvis disse peger udenfor **DocumentRoot**-området). Vi har også sat at man leder efter indeks-filer med navne `index.shtml`, `index.html`, `index.htm` eller `index.php`. Vi har med "Includes" valgt, at webserveren skal fortolke hjemmesider, så du kan opdele siderne i forskellige filer, der så indlæses af serveren via linjer, såsom

```
<!--#include virtual="venstremenu.incl" -->
```

1.4. CGI-programmer

CGI står for Common Gateway Interface og tillader dig at kører programmer på din server, hvis indhold præsenteres af browseren (forudsat programmerne generer nogle data, browseren kan forstå). Det mest normale er at køre Perl-programmer gennem CGI, men det er også muligt at skrive programmer i Ada, C, C++, Python og mange andre sprog, som kan afvikles via CGI. I Afsnit 6.2.1 er der et eksempel på et CGI-program.

Som udgangspunkt vil Apache helst kun køre CGI-programmer fra `/cgi-bin/`-området (som placeres i `cgi-bin`-biblioteket under Apache-serverens installations bibliotek). Hvis du vil kunne køre CGI-programmer overalt, skal du fjerne udkommenteringen fra den linje der hedder "AddHandler cgi-script .cgi" og sikre at dine "Options" for det område CGI-programmerne skal kunne afvikles har parameteren ExecCGI. Også selv om du kun vil køre CGI-programmer fra `/cgi-bin/`-biblioteket, skal du fjerne udkommenteringen for linjen "AddHandler cgi-script .cgi".

Bemærk at CGI-programmer kan åbne for en række sikkerhedsrisici, som du skal overveje nøje. Prøv for eksempel at kigge nærmere på:

- http://www.w3.org/Security/Faq/wwwsf4.html
- http://www.go2net.com/people/paulp/cgi-security/

1.5. Apache-udvidelser

Der findes en række udvidelser i form af moduler til Apache. Disse kan integrere yderligere funktionalitet direkte ind i Apache. For at tilføje moduler skal du normalt genoversætte Apache.

Det kan dog lade sig gøre at oversætte moduler, som kan indlæses af Apache, så man ikke behøver at oversætte helt forfra. Om og hvordan et modul kan oversættes som et selvstændigt modul, kan du læse mere om i dokumentationen for de enkelte moduler.

I det følgende kan du finde en kort introduktion til nogle af de mest udbredte og anvendte moduler. Udover de her nævnte findes der en række andre. Den komplette liste findes i Apache Module Registry på http://modules.apache.org/.

1.5.1. mod_perl

mod_perl-projektet fusionerer webserveren Apache med Perl-oversætteren. Hvis man har lavet sine CGI-programmer i Perl, er det virkeligt et modul der tramper speederen i bund.

Udover fartforøgelsen, som mod_perl tilfører dine programmer, giver det tillige et komplet interface til selve Apache-serveren, og dermed mulighed for at lave nogle utroligt avancerede ting. Eksempler på dette kunne f.eks. været at køre adgangskontrol direkte op imod en database, eller bruge en database som adgangs- og fejllog for din webserver.

Du kan finde mere om mod_perl på http://perl.apache.org/

Hvis du ikke er interesseret i mod_perl eller synes det virker for kompliceret, kan du eventuelt også kigge nærmere på Fast_CGI.

1.5.2. mod_php

PHP er et programmeringssprog der er indlejret i HTML-koden. En del synes PHP er lettere at gå til end Perl. PHP har blandt andet en række funktioner i selve programmeringssproget, der giver adgang til databaser, post med mere.

1.5.3. mod_proxy

mod_proxy giver Apache-serveren mulighed for at agere stedfortræder (engelsk: "proxy"). Det betyder at du kan omdirigere forespørgsler fra en adresse til en anden (helt usynligt hvis du ønsker det).

Dette kunne f.eks. være, hvis du ville sprede belastningen mellem to servere eller ligende.

1.5.4. mod_ssl

mod_ssl giver mulighed for at Apache kan kommunikere med klienterne over en SSL-krypteret forbindelse. SSL bruges som krypteringsalgoritme på de fleste websteder som gerne vil beskytte informationen der sendes mellem server og klient.

Husk at SSL højest giver dig mulighed for at øge sikkerheden, men du skal dog selv forstå at implementere den gennem opsætning og fornuftig drift af serveren.

Hvis man vil køre https og dermed SSL på sin webserver skal man have et SSL-certifikat. Dette kan købes for eksempel hos Verisign.com, men man kan også blot benytte et selvunderskrevet certifikat, det er både billigere (gratis) og hurtigere at få. Installerer man Apache med SSL-modulet, kommer der ofte et certifikat med som er et generelt standard-certifikat. Dette er nok til at man kan køre https, og derved benytte den sikre kommunikationsprotokol, som ingen kan lytte med på.

Man kan også lave sit eget selvunderskrevne certifikat, som så vil indeholde eget firmanavn og kontaktoplysninger. Dette vil se en del pænere ud for brugerne af systemet. Et selvunderskrevet certifikat kan laves ved som brugeren root at køre:

```
[root@hven /root]# cd /etc/httpd/conf
[root@hven /etc/httpd/conf]# rm ssl.crt/server.crt
[root@hven /etc/httpd/conf]# make testcert
```

Hertil skal indtastes oplysninger om firma, kontaktadresse mv. Et lille tip: indtast "state, province" som et enkelt mellemrum. Amerikanerne tror alle bor i føderale nationer med delstater.

Man kan også lave sin egen krypterede nøgle, med **make genkey** men så skal man angive denne nøgle før Apache kan startes op, og det kan være noget besværligt, for eksempel ved automatisk opstart efter strømnedbrud.

1.6. Apache som WAP-server

Det seneste år er WAP-telefoner ved at blive en udbredt standard for at hente mobile internet-sider. Siderne er ikke formatteret i HTML, men i WML (*Wireless Markup Language*), som er en XML-variant.

I al sin enkelthed er WAP set fra serverens side blot et nyt filformat; WML. Det der skal tilføjes på din hjemmeside for at understøtte WAP, er så kun de samme data i WML, og så formateret i et format så de kan ses på en meget lille skærm.

Det er snyde nemt af få en Apache webserver til at fungere som en WAP-server. Det kræver basalt set kun at webserveren kan fortolke at filer der ender på `.wml` sendes som `Content-type: test/vnd.wap.wml`. Samtidig skal du måske også have `.wbmp`-billeder med over, så derfor skal du tilføje disse to linjer til `/etc/http/conf/httpd.conf`:

```
# For de almindelige WML-sider
AddType text/vnd.wap.wml      .wml
# For WML indlejret grafik
AddType image/vnd.wap.wbmp    .wbmp
```

Nu skal du bare genstarte Apache, og du har en WAP-server. Lad os lige tage to eksempler.

Eksempel 1-1. Det første WAP-eksempel

Det enkleste eksempel på en WAP-side er nok denne

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xm

<wml>
```

```
 <card id='home'>
  <p>
Her kommer teksten på min første WAP-side
  </p>
 </card>
</wml>
```

Det vigtigste, der skal bemærkes sammenlignet med HTML, er at næsten alle tags skal afsluttes, dvs. *skal* efterfølges af . Dog skal et billede ikke afsluttet, som det kan ses i næste eksempel

Eksempel 1-2. Et WAP-eksempel med et billede

<img> skal anvendes med / til sidst og alt-tag skal med.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  "http://www.wapforum.org/DTD/wml_1.1.x

<wml>
  <card id="demo" title="Peters søn">
    <p>
    Her er min søde søn Frederik
    <img src="F2.wbmp" alt="Frederik"/>
    </p>
  </card>
</wml>
```

WML er en ret stor standard, som man kan læse mere på på http://www.wap.com eller http://www.wapforum.com. Du kan på http://www.anywhereyougo.com/ayg/ayg/Content.po?name=wap/Wmlidx få mere hjælp om de enkelte tags. Har du ingen WAP-telefon kan du med fordel bruge http://gelon.net (http://gelon.net/) som har en Java-baseret WAP-simulator der er velegnet til test.

Er du interesseret i en kom-i-gang-guide, så prøv at læse http://www.zend.com/zend/tut/wap.php

I det følgende går vi dog lidt længere, idet vi viser hvordan man også får PHP-understøttelse til WAP. Vi antager at du har installeret Apache og PHP4.

Når et PHP-program starter med at sende data tilbage til modtageren, sender PHP først noget et *HTTP-hoved*. Hovedet indeholder dato, serverudgave og meget andet, men især hvilken type data der vil blive sendt. Hvis intet er anført i hovedet sendes *Content-type: text/html*, og dette kan ændres med en enkelt kommando: *Content-type: text/vnd.wap.wml*. Herunder er eksemplet fra før hvor filen hedder `test.php` i stedet for `test.wml`, hvorefter Apache vil overgive kontrollen med filen til PHP.

```
<?php
   // Filnavn: test.php
   header("Content-type: text/vnd.wap.wml");
```

```
?><?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"  "http://www.wapforum.org/DTD/wml_1.1.x
<wml>
 <card id='home'>
  <p>
  Dato idag er <?php echo date("Y-m-d"); ?>
  </p>
 </card>
</wml>
```

I Perl skrives HTTP-hovedet i starten og efterfølges af to '\n'. Dette eksempel skal placeres i `/cgi-bin/`-kataloget.

```
#!/usr/bin/perl -w
# Filnavn: test.pl

# Bemærk to gange '\n' som afslutter HTTP-hovedet
print "Content-type: text/vnd.wap.wml\n\n";

print "<?xml version=\"1.0\"?>\n";
print "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"";
print "  \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n";
print "<wml>\n";
print "  <card id='home'>\n";
print "  <p>\n";
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)
    = localtime(time);
my $today=sprintf("%04d-%02d-%02d",$year+1900,$mon+1,$mday);
print "  Dato idag er $today\n";
print "  </p>\n";
print "  </card>\n";
print "</wml>\n";
```

SSLUG har eksempelvis siden november 1999 vist sin kalender i WAP/PHP-format. Kildeteksten findes på http://www.sslug.dk/adict/wap.phps og kan ses med en almindelig browser. Resultatet af programmet findes på http://www.sslug.dk/adict/wap.php, hvor du så skal bede browseren om at gemme indholdet på din disk hvis du prøver med din browser. Alternativt kan http://gelon.net (http://gelon.net/) bruges til at se WAP-kalenderen.

1.7. Læs mere om Apache

Ben Laurie & Peter Laurie: *Apache: The Definitive Guide*, O'Reilly & Associates, Inc., ISBN: 1-56592-250-6, 246 sider + cd-rom

1.8. Alternative webservere

Udover Apache serveren er der et par andre, man også kan vælge mellem. De to mest interessante er beskrevet herunder. Ved at søge på http://www.linuxlinks.com/Software/Internet/WebServers/ (http://www.linuxlinks.com/Software/Internet/WebServers/) kan du finde mange andre.

1.8.1. Roxen

Et godt alternativ til Apache er Roxen, som er lavet i Sverige. Roxen har et veludviklet makrosprog, hvilket gør det let at understøtte flere sprog på sine websider. Roxen udmærker sig også ved at kunne sættes op direkte via internettet. Roxen kan hentes fra http://www.roxen.com. Roxen er også under GPL.

1.8.2. Netscape Webservere

En ting som ofte afskrækker nye webmestre er at Apache sættes op via tekstfiler. Dette er ikke tilfældet for Netscapes webservere, som har et komplet browser-baseret interface til opsætning, administration og anden vedligeholdelse af serverne.

I lighed med Apache, findes der en række moduler til Netscape Serverne, som f.eks. giver mulighed for at afvikle Java på serveren, et modul der svarer til mod_perl og ligende.

Netscapes webservere hedder i dag iPlanet, og kan kun erhverves på kommercielle vilkår.

Kapitel 2. Dynamiske websider

Den mest interessante feature for en webserver er i denne sammenhæng at webserveren ikke kun kan afsende statiske sider, men kan fortolke kode indlejret i websider så som Server-Side Includes, Mod_Perl og PHP, men også at webserveren kan udføre selvstændige programmer, programmets output kan være tekst, billeder eller lyd, og dette bliver af webserveren returneret til browseren. Denne sidste mulighed kaldes for CGI-programmer (Common Gateway Interface). CGI-programmer bruges typisk til at fortolke svaret fra webformularer, tællere af besøgende på en webside og til søgemaskiner.

2.1. CGI-programmer

Til at programmere CGI-programmer kan man vælge et vilkårligt programmeringssprog. Til små programmer der udskriver server information kan et kommandofortolkerprogram være passende. Til sider hvor svartiden er kritisk eller hvor der skal udføres komplekse matematiske operationer, kan oversatte sprog som Ada, C eller C++ med fordel anvendes. Men normalt skal kun tekst indlæses og udskrives, og det er en opgave som programmeringssproget Perl er designet til at løse. Perl er et fortolket sprog, der er beregnet til at behandle tekst med. Fordi det er et fortolket sprog, skal Perl-koden fortolkes hver gang et program udføres, (hvilket tager en del tid). Til gengæld behøver man ikke at bekymre sig om buffer overløb (eng. buffer overflows) i Perl, hvilket er den hyppigste fejl i C-programmer der anvender strenge. Hvis man skal udføre mange CGI-programmer kan det være nødvendigt at anvende Fast_CGI- eller Mod_Perl-modulet. Mod_Perl er et modul til Apache, som gør at du kan have inline Perl i HTML uden at det koster mange ressourcer per CGI-kald.

De følgende afsnit indeholder Perl-eksempler og kodestumper. Funktioner der allerede er anvendt i tidligere eksempler er slettet for at holde længden af programmerne nede, men der står også hvilke kodestumper der mangler og hvor de kan findes. Mange af disse eksempler kan hentes i et fuldt fungerende eksempel. Står der f.eks. `debug.cgi` i Perl-koden betyder det, at kode eksempel kan hentes på www.linuxbog.dk/web/eksempler/debug.cgi (http://www.linuxbog.dk/web/eksempler/debug.cgi), eller via den pakke af eksempler, som hører til webbogen og ligger under www.linuxbog.dk (http://www.linuxbog.dk/).

2.1.1. Opsætning af Apache til at udføre programmer.

Hvis du selv bestyrer en webserver skal den først sættes op til at udføre filer af typen `.cgi` som CGI-programmer, hvis du bruger et webhotel så vil du sikkert have adgang til noget information om hvor og hvordan programmer kan udføres. Følgende ændringer skal udføres i Apaches opsætningsfiler, der ligger normalt i `/etc/httpd/conf/` kataloget:

Filen `srm.conf` skal indeholde følgende linjer

```
AddHandler cgi-script .cgi
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
```

Filen `access.conf` definere web browseres rettigheder til at hente sider i kataloger, og 'Options ExecCGI' skal være slået til i alle kataloger hvor webserveren skal udføre programmer. Typisk er der i forvejen et `/home/httpd/cgi-bin` katalog hvor webserveren har rettigheder til at udføre programmer:

```
<Directory /home/httpd/cgi-bin>
  AllowOverride None
  Options ExecCGI
</Directory>
```

Det kan være nødvendigt at ændre adgangen til dette katalog, så du som normal bruger kan skrive til det. Dette gøres med kommandoen **chmod o+rwx /home/httpd/cgi-bin**.

Filen `httpd.conf` skal indeholde følgende linjer:

```
LoadModule dir_module         modules/mod_dir.so
AddModule mod_cgi.c
```

Mange af disse ændringer består i at fjerne udkommenteringen af '#' i starten af eksisterende linjer. Når disse ændringer er udført og webserveren genstartet med **/etc/rc.d/init.d/httpd restart**, så vil et program, `runme.cgi`, der ligger i `/home/httpd/cgi-bin/` kataloget, kunne udføres ved at man i en browser beder om `http://localhost/cgi-bin/runme.cgi`.

2.1.2. Kommunikation mellem webserver og CGI-programmer

Et program har altid tre datastrømme: stdin (standard input) der er input fra keyboard, stdout (standard output) der er output til skærmen, og stderr (standard error) der er fejlmeddelelser og også udskrives til skærmen. Dog skal det bemærkes at alle tre standard-datastrømme være omdirigerede.

Når webserveren udfører et CGI-program, vil alle data der udskrives til stdout blive returneret til browseren. Data der skrives til stderr vil blive gemt i webserverens fejllog `/var/log/httpd/error_log`. Programmet kan også modtage data på stdin fra webserveren, dette vil typisk være data der er indtastet i en webformular eller filer, der skal lægges op på serveren. CGI-programmer modtager også en række data igennem systemvariable (eng. environment variable).

I det følgende afsnit vil vi introducere CGI-programmer der returnerer et simpelt svar og ikke modtager data, forklare hvordan programmer kan returnere forskellige typer data, og gå igennem hvordan programmer kan modtage data fra webformularer og anvende systemvariable til at bestemme f.eks. IP addressen på den webbrowser som ønsker data. Desuden vil vi forklare, hvilke sikkerhedsproblemer, der er omkring programmer, og hvordan man debugger programmer.'

2.1.3. Simpelt svar

Det simpleste eksempel på et CGI-program er et, der bare returnerer en enkel oplysning, f.eks. hvad klokken er. Gem følgende program i `/home/httpd/cgi-bin/` som `klokken.cgi`:

```
#!/usr/bin/perl -w
#Filnavn: klokken.cgi

#Udskriv hoved
print "Content-Type: text/plain; charset=iso-8859-1\r\n";
print "\r\n";

#Udskriv indhold
print "Datoen er ".scalar(localtime);
```

Køres `klokken.cgi` fra kommandolinjen får man:

```
Content-Type: text/plain; charset=iso-8859-1

Datoen er Thu Jul 20 11:38:08 2000
```

Prøv nu at hente det i en webbrowser som **http://localhost/cgi-bin/klokken.cgi**. Du skulle gerne kunne se hvad datoen er. Prøv at genindlæse siden nogle gange, så du kan se sekunderne gå. Hvis dette ikke virker, så kig i afsnittet om »Typiske Fejl«.

Først udskriver programmet en linje med en "Content-Type", der fortæller webbrowseren, hvad den skal gøre med hvad de data der følger, feltet kaldes for medietypen. Her betyder »text/plain« det at indholdet er klar tekst, og »charset=iso-8859-1« at teksten er kodet i UTF-8. Der kan være flere linjer i hovedet og de afsluttes med en tom linje, alt under den tomme linje er data der vises i browseren.

Tekst er lidt kedeligt, så vi kunne istedet ønske at returnere HTML-formateret tekst. Dette kan let gøres ved at modificere programmet som følger:

```
#!/usr/bin/perl -w

#Filnavn: klokken2.cgi

#Udskriv hoved
print "Content-Type: text/html; charset=iso-8859-1\r\n";
print "\r\n";

#Vælg tilfældig farve
my $color=sprintf("%x",int rand(0x1000000));

#Udskriv HTML indhold
print "<H1>Datoen er: ";
print "  <font color=\"#$color\">", scalar(localtime) ,"</font>";
print "</H1>";
```

Programmet vil nu udskrive datoen med store bogstaver og med en tilfældig farve, der ændrer sig hver gang programmet kaldes/siden genindlæses.

2.1.4. Medietyper

Et CGI-program kan ikke kun returnere ren tekst og HTML-formatteret tekst, men også en lang række andre medieformater. I Netscape kan man i `Edit/Preferences/Navigator/Applications` se hvilke programmer, der udføres når browseren modtager en fil med en bestemt Medietype, hvis Netscape ikke kender filtypen bliver brugerne bedt om at gemme filen på harddisken.

Følgende er en liste af medietyper (eng. Media) for nogle hyppigt anvendte formater.

Tabel 2-1.

Media type	Beskrivelse
text/plain	ASCII-formateret tekst (.txt)
text/html	HTML formateret tekst (.html eller .htm)
image/jpeg	JPEG billed (.jpeg eller .jpg)
image/png	PNG billed (.png)
image/gif	GIF billed (.gif)
application/postscript	Postscript-fil (.ps)
application/x-dvi	DVI-fil (oversat LaTeX-fil - .dvi)
application/pdf	Portable Document Format fil (.pdf)
audio/x-mpeg	MPEG komprimeret lyd (.mp3)
audio/x-wav	Wave fil (.wav)
video/x-mpeg2	MPEG animation (.mpeg eller .mpg)
video/quicktime	Quicktime animation (.mov)
video/x-msvideo	Microsoft Video (.avi)

2.1.5. HTTP-hovedet

Hvis man returnere et svar med et CGI-program kan det ske at browseren gemmer en lokal kopi af det, og en genindlæsning vil derfor ikke køre programmet igen, men kun vise den lokale kopi. Følgende stump kode fortæller browseren og strømænd (som for eksempel Squid), at siden ikke må gemmes:

```
#Udskriv hoved
print "Content-Type: text/html\r\n";

#HTTP 1.1
```

```
print "Cache-Control: no-cache\r\n";

#HTTP 1.0 tilbage-kompatibilitet
print "Pragma: no-cache\r\n";
print "\r\n";

#Resten af programmet
```

Hvis siden må gemmes, men man ønsker at bestemme hvor længe, kan det gøres med **Expires**-feltet. Følgende stump kode viser hvordan:

```
# Indsættes: Udskriv hoved

use POSIX qw(strftime);

#må leve en time i cachen
my $levetid = 1*60*60;

#korrekt format laves med følgende linje
my $date = strftime "%a, %e %b %Y %H:%M:%S GMT", gmtime(time+$levetid);

print "Content-Type: text/html\r\n";
print "Expires: ",$date,"\r\n";
print "\r\n";

#Her udføres resten af programmet
```

Programmet vil f.eks. udskrive en "Expires: Sat, 29 Jul 2000 16:45:27 GMT" linje, der betyder at efter den dato er cache versionen forældet. Programmet fungerer ved at **time** returnerer antallet af sekunder, der er gået siden 1. januar 1970, til dette antal sekunder lægges dokumentets levetid, og dette konverteres af gmtime til time, dag, måned mv. **Strftime** konverterer derefter disse data til det angivne tidsformat, som er det, HTTP standarden forskriver.

Mere information om HTTP-hovedet kan findes i dokumenterne RFC 1945 og RFC 2068, der definerer HTTP 1.0- og HTTP 1.1-protokollerne.

2.1.6. Omdirigering

I mange situationer bruges et program til at sende browsereren en side som er dynamisk genereret. F.eks. kunne et program returnere forskellige sider afhængig af hvilken browser, man bruger, hvilket tidspunkt af døgnet det er, bruger et tilfældigt baggrundsbillede og/eller varierer indhold alt efter hvilken IP-addresse, man har.

Istedet for at returnere indholdet af filen eller outputtet af programmet til browseren kan et program returnere en redirektion, der fortæller browseren, at den skal finde indholdet på en anden URL.

Et eksempel:

Hver gang en browser forsøger at hente et katalog uden at specificere en index-fil, så afsendes `index.html` filen. Denne funktionalitet specificeres i `/etc/httpd/conf/httpd.conf` med linjen:

```
DirectoryIndex index.htm index.html index.shtml index.cgi
```

Dvs. at `index.cgi` køres hvis ikke nogen af de andre indeks-filer kan findes i kataloget. Så gem følgende program som `index.cgi` i et katalog med en masse HTML-filer, og hvor der ikke er nogen anden indeks-fil.

```
#!/usr/bin/perl -w
#Filnavn: index.cgi

#Lav en liste med indholdet af ./ kataloget
#dvs. samme katalog som index.cgi selv
opendir DIR, "./" or die "Can't list directory /.!\n";

#filtrér listen så den kun indeholder .html-filer
my @files = grep /\.html$/, readdir(DIR);

#vælg tilfældigt index i listen
my $no=int rand($#files+1);

#udskriv en location til dokument no $no
print 'Location: ', $files[$no] ,"\r\n\r\n";
```

Programmet vil finde alle `.html`-filer i samme katalog, og udskrive en redirektion til en filfældig `.html`-fil.

Den samme metode kan bruges til at returnere et tilfældigt baggrundbillede, hver gang en person besøger en side. Hvis der ligger en masse JPEG billeder i `/baggrund`-kataloget, kan ovenstående program gemmes i `/baggrund`-kataloget, og modificeres så '.html' ændres til '.jpeg', og HTML-koden for sider, der skal have en tilfældig baggrund er nu:

```
<body background="/baggrund/index.cgi">
```

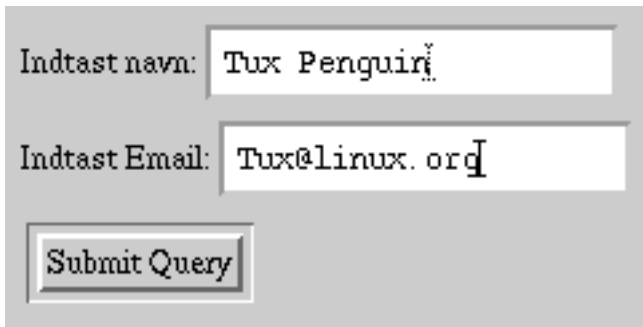
2.1.7. Webformularer (Get-metoden)

CGI-programmer bruges også til at fortolke data, der bliver indtastet i webformularer på nettet. Følgende eksempel er en simpel HTML-formular, hvor man kan indtaste et navn og en e-postadresse:

```
<form action="/cgi-bin/get_metoden.cgi" method="GET">
   Indtast navn: <input type="text" name="navn"><br>
   Indtast Email: <input type="text" name="adresse"><br>
   <input type="Submit">
</form>
```

I browseren vil det se ud som:

Figur 2-1. Illustration



Når et navn og en e-postaddresse er indtastet, og der trykkes på Submit knappen, så vil browseren sende de indtastede data til webserveren, der starter **get_metoden.cgi**. Dataene overfører webserveren igennem systemvariablen `QUERY_STRING`. (C/C++ programmer vil også modtage dem igennem argv-tabellen).

Gem følgende program som `get_metoden.cgi` i `/home/httpd/cgi-bin`:

```
#!/usr/bin/perl -w

print "Content-Type: text/plain\r\n\r\n";
print $ENV{'QUERY_STRING'};
```

I eksemplet vist vil get_data.cgi udskrive "navn=Tux+Penguin&adresse=Tux%40linux.org". Programmet modtager altså data fra webformularen, men de ser lidt mystiske ud! Formattet af data er "variabel1=indtastning1&variabel2=indtastning2&..". Hvor variabel1 og variabel2 svarer til indeholdet af name feltet i HTML-koden (altså "navn" og "adresse").

Både navne og indtastninger er URL-kodet. Det betyder at mellemrum skiftes ud med '+', ligeledes bliver en række tegn. specielt '=','%' og '&' ombyttet med den tilsvarende hexadecimale kode skrevet som %xx. I eksemplet blev '@' lavet om til %40.

For at URL-afkode en streng skal den altså først klippes i stykker ved alle &-tegnene. Dette giver en tabel af strenge af formen "variabel=indtastning". Hver streng skal så klippes ved '=' og man får variabel og indtastning hver for sig, og så skal alle '+' konverteres til mellemrum og bagefter skal alle %xx konverteres til det tilsvarende UTF-8-tegn i både variabel- og indtastningsstrengene.

En lille detalje er at %00 — tegnet "NULL" — skal fjernes. Det er vigtigt, at dette gøres netop i denne rigtige rækkefølge ellers vil alle "+"-tegnene, der er indtastet ændres til mellemrum.

Følgende Perl-program laver en URL-kodet streng om til en hashtabel `%data` af (variabelnavn->værdi)-par.

```
#!/usr/bin/perl

#Filnavn: get_metoden.cgi

sub URLdekrypt
#Udfører url dekryption på en streng.
{
    my ($input)=(@_);

    my ($variabel,$indtastning);
    my %data;

#Klip input strengen ved all '&'-tegn

    my @query=split /&/, $input;

#loop gennem tabellen af "variabel=indtastning" strenge

    foreach (@query)
      {

#Klip ved '='

        ($variabel,$indtastning)=split /=/, $_;

#fix '+' før %xy!

        $variabel    =~ tr/+/ /;
        $indtastning =~ tr/+/ /;

#null tegn er uønskede!
        $variabel    =~ s/%00//g;
        $indtastning =~ s/%00//g;

#substituter %xy med det tilsvarende tegn

        $variabel    =~ s/%([0-9A-Fa-f]{2})/pack("c",hex($1))/ge;
        $indtastning =~ s/%([0-9A-Fa-f]{2})/pack("c",hex($1))/ge;

#Hvis flere felter har samme navn så konkateneres deres indhold
```

```
#separeret af et "|"-tegn.

        if ($data{$variabel})
            { $data{$variabel}=$data{$variabel}."|".$indtastning; }
          else
            { $data{$variabel}=$indtastning; }
      }

   return %data;
}

my %data=URLdekrypt($ENV{'QUERY_STRING'});

#udskriv data
print "Content-Type: text/plain\r\n\r\n";
print "Jeg modtog følgende data:\n";

foreach (sort keys %data)
    {
       print "   $_ = \"$data{$_}\"\n";
    }
```

Programmet vil udskrive

```
Jeg modtog følgende data:
    adresse = "Tux@linux.org"
    navn = "Tux Penguin"
```

I stedet for at udskrive disse i browseren kunne programmet tilføjes navn og e-postaddresse til en database eller en besøgsbog, sende et brev til addressen, eller hvad nu man kan finde på.

2.1.8. Webformulare (Post-metoden)

Rent praktisk returnere get metoden data ved at konkatenere dem til URL'en til CGI-programmet afskildt af et spørgsmålstegn (se i browserens titel vindue). Browseren sender så hele denne URL til webserveren. Get metoden har den fordel at man kan lave bogmærker med den side, som programmet returnere på basis af de indtastet data, dette bruges typisk til at bogmærker af søge resulteter når man bruger en søgemaskine. Hvis denne streng er meget lang risikere man at få et bufferoverflow i browseren eller visse webservere, der så vil crashe. En overgang kunne enhver Microsoft Internet Server version 4.0 hackes på den måde. Get metoden kan altså kun bruges til programmer hvor mindre end ca. 200 bytes skal returneres.

Alternativt findes post metoden for at sende data tilbage til webserveren. Når et data Post'es så modtager CGI-programmet data på standard input, ligesom data var indtastet på tastaturet, men de er stadig URLenkrypterede, og systemvariablen `CONTENT_LENGTH` indeholder antallet af bytes der kan indlæses. Det har fordelen at meget store datamængder kan returneres uden problemer, men det er umuligt at lave

en bookmark til den side, der returneres efter en indtastning. Det tidligere eksempel kan let omskrives til at bruge post metoden:

```
<form action="/cgi-bin/post_metoden.cgi" method="POST">
   Indtast navn: <input type="text" name="navn"><br>
   Indtast e-postadresse: <input type="text" name="adresse"><br>
   <input type="Submit">
</form>
```

```
#!/usr/bin/perl

#filnavn: post_metoden.cgi
#URLdekrypt funktionen kommer fra get_metoden.cgi

sub HentPostData
#Returnere en variabel med strengen på stdin
{
#antallet af bytes der venter på stdin
   my $ContentLength = $ENV{"CONTENT_LENGTH"};

   my $input="";

#max 10 kb input.
   my $maxsize=10240;

   if($ContentLength)
    {
       if ($ContentLength<$maxSize)
          {
            read(STDIN,$input,$ContentLength);
          }
        else
          {
            print "Content-Type: text/plain\r\n\r\n";
            print "Script input exceeds acceptible size limit!";
            die "Error! $ENV{'REMOTE_ADDR'} attempted to submit $contentLength bytes!\n";
          }
     }

   return $input;
}

my %data=URLdekrypt(HentPostData());

#udskriv data
print "Content-Type: text/plain\r\n\r\n";
print "Jeg modtog følgende data:\n";

foreach (sort keys %data)
    {
       print "   $_ = \"$data{$_}\"\n";
    }
```

Post metoden kan modtage lige så store datamænger som man ønsker. Men antallet af programmer der kan kører parallelt og mængden af hukommelse på den computer de køre på sætter en grænse. I ovenstående eksempel er grænsen sat ved 10kb. Hvis en browser forsøger at sende flere data vil CGI-programmet returnere en fejl til browseren, programmet vil så dø efter at havde gemt en diagnostisk tekst i webserverens fejllog, den diagnostiske tekst indeholder $ENV{'REMOTE_ADDR'}, hvilket er IP addressen på den computer der har submittet et for stort svar. Det følgende afsnit forklarer hvilke andre data et CGI-program modtager.

2.1.9. Information tilgængelig for et CGI-program

Når et program udføres, så modtager det information om browserens IP address (typisk ikke dens DNS navn), og browserens navn og version. Kun hvis programet befinder sig i et adgangskodebeskyttet katalog kan man få information om brugeren, der har indtastet data.

I HTML-koden der kalder programmet kan der også indlejres information f.eks.:

```
<input type="hidden" navn="hemmelig" value="Jeg er en hemmelig streng">
```

Hvis HTML-koden til en webformular udskrives af et program, og det program ønsker at overføre information til det program der modtager de indtastede data, så kan skjulte felter anvendes. Webbrowseren viser ikke disse felter for brugeren, og deres indhold returneres uændret til programmet, der modtager de indtastede data. I det ovenstående eksempel vil programmet modtage en variabel med navnet "hemmelig", der har indholdet "Jeg er en hemmelig streng".

CGI-programmer kan også modtage en sti i systemvariablen `PATH_INFO`, og argumenter i QUERY_STRING. Disse kommer fra action=".." linjen i HTML-koden for webformularen. Følgende er en rækker eksempler på hvordan et program kan kaldes fra HTML-koden i en webformular:

Tabel 2-2. URL til CGI-program

URL til CGI-program	Forklaring
/cgi-bin/debug.cgi	Programmet modtager ikke nogen kommandolinjeargumenter.
/cgi-bin/debug.cgi?arg1&arg2&arg3	`QUERY_STRING`-systemvariablen vil indeholde strengen "arg1&arg2&arg3".
/cgi-bin/debug.cgi/foo/bar	`PATH_INFO`-systemvariablen vil indeholde strengen "/foo/bar".
/cgi-bin/debug.cgi/foo/bar?arg1&arg2&arg3	Både QUERY_STRING og PATH_INFO vil indeholde data.

Et eksempel på HTML-koden til en webformular der anvender kommandolinjeargumenter og skjulte variable:

```
<form action="/cgi-bin/debug.cgi/foo/bar?arg1&arg2" method="POST">
```

```
   Indtast navn: <input type="text" name="navn"><br>
   Indtast Email: <input type="text" name="adresse"><br>
   <input type="hidden" navn="hemmelig" value="Jeg er en hemmelig streng">
   <input type="Submit">
</form>
```

Følgende er en liste af de mest interessante systemvariable som CGI-programmer har til rådighed:

Tabel 2-3. Variable

Variabel	Indhold
REQUEST_METHOD	Indeholder GET eller POST. I eksemplet ovenover "POST".
PATH_INFO	Indeholder stien program URLen i webformularens HTML-kode. I eksemplet ovenover indeholder den "/foo/bar".
PATH_TRANSLATED	Indeholder PATH_INFO katalog data men relativt til WebRoot. Hvis webfiler ligger i /www så vil den i eksemplet indeholde "/www/foo/bar".
QUERY_STRING	Get metode: Indeholder URLenkrypted indtastningsdata. Post metode: Indeholder argumenter efter '?' i URL'en til CGI-programmet. I eksemplet ovenover indeholder den "Arg1&Arg2"
HTTP_ACCEPT	Indeholder de media typer, som browseren kan forstå. En tekst browser vil for eksempel ikke forvente grafik.
HTTP_ACCEPT_CHARSET	Tegnkodning som browseren kan forstå. Værdien kan for eksempel være "`iso-8859-1,utf-8,*`", hvilket betyder at browseren foretrækker UTF-8-tegnkodningen, sekundært UTF-8-tegnkodningen, men i øvrigt accepterer vilkårlige tegnkodninger.
HTTP_ACCEPT_LANGUAGE	Det sprog som browseren fortrækker et svar på. For en bruger der fortrækker Dansk men den indeholde "da, en". (I Netscape sættes sprog preferencer i Edit/Preferences/Navigator/Languages)
HTTP_ACCEPT_ENCODING	Hvis browseren er istand til at klare komprimerede data. f.eks. "gzip, compress".
HTTP_USER_AGENT	Identificere brugerens browser. f.eks. Netscape 4.73: "Mozilla/4.73 [en] (X11; U; Linux 2.2.16-3 i686)" Galeon (en gtk browser baseret på Gecko): "Mozilla/5.0 (X11; U; Linux 2.2.16-3 i686; en-US; Galeon) Gecko/20000713".

Variabel	Indhold
REMOTE_ADDR	IP addresse på computer hvor data er blevet indtastet.
REMOTE_HOST	DNS navnet der svarer til IP-adressen i REMOTE_ADDR. Denne returneres kun hvis HostNameLookups er slået til i Apache's kontrol filer. Det er den ikke som standard da det forsinker Apache at skulle lave et DNS opslag for hver forbindelse. (Den var slået til i den første netcraft sammenligning mellem Apache og MS Interne Server)
REMOTE_USER	Hvis programmet ligger i et kodeordsbeskyttet katalog, så indeholder REMOTE_USER navnet som brugeren har indtastet sammen med sit kodeord. Det er ikke nok at kodeordsbeskytte den html side, der indeholder webformularen!

2.1.10. Et udgangspunkt for CGI-programmer

Det følgende program, **echo.cgi**, kan bruges som udgangspunkt for egne CGI-programmer og for at debugge webformularer. Det kan modtage både Get og Post data, udskriver alle de data det modtager.

```
#!/usr/bin/perl

#Filnavn: echo.cgi

#URLdekrypt er den samme funktion som i get_metoden.cgi
#HentPostData kommer fra kodestumpen i afsnittet om Post metoden

my %data;

if ($ENV{'REQUEST_METHOD'} eq "POST")
   {
      %data= URLdekrypt(HentPostData());                    #Post data
   }
  else
   {
      %data= URLdekrypt($ENV{'QUERY_STRING'});              #Get Data
   }

#Indtastningsdata er nu indlæst i %data

#Udskrivning af data
print "Content-Type: text/plain\r\n\r\n";
print "Variable Modtaget:\n\n";

#udskriv systemvariable
```

```
print "Systemvariable:\n";
foreach (sort keys %ENV)
    {
       print "   $_ = \"$ENV{$_}\"\n";
    }

#udskriv webformular data

if (%data)
{
  print "\nWebformular data:\n";

  foreach (sort keys %data)
    {
       print "   $_ = \"$data{$_}\"\n";
    }
}

#Hvis POST metoden bruges og der er argumenter så udskriv disse.

if ($ENV{'QUERY_STRING'} and $ENV{'REQUEST_METHOD'} eq "POST")
 {
   print "\nArgumenter:\n";

   foreach (split /&/, $ENV{'QUERY_STRING'})
       {
          print "   \"$_\"\n";
       }
}
```

2.1.11. Typiske fejl

Apache returnerer "Not Found":

Årsagen er at Apache ikke kan finde CGI-programmet.

Hvis URL'en indeholder `/cgi-bin/` kan fejlen være at du ikke har tilladt cgi-programmer i `/etc/httpd/conf/httpd.conf`.

Apache returnerer "Forbidden":

Årsagen er, at Apache ikke kan udføre programmet.

Problemet skyldes "Option ExecCGI" ikke er slået til for det katalog som CGI-programmet ligger i, dette gøres i access.conf. Alternativt kan fejlen skyldes at Apache ikke har lov til at udføre programmet, det kan typisk kan det løses ved "chmod o+rx program.cgi".

Apache returnere "Internal Server Error"

Denne fejl betyder at det første der udskrives ikke er et korrekt formateret HTTP-hoved. Fejlen kan jo også have andre årsager.

Stderr og stdout output fra Perl-programmer kan findes i webserverens fejllog ofte lokaliseret i `/var/log/httpd/error_log`, og det gør det let at løse denne type fejl. Et hurtigt syntakstjek er at køre CGI-programmet på kommandolinjen og se om der kommer syntaks fejl.

En anden mulighed er at Apache ikke har rettigheder til at læse programmet, det kan typisk kan løses med "chmod o+r program.cgi", denne type fejl bliver også logged i error_log. Tager Perl-programmet for lang tid at udføre vil en timeout af webserveren også returnere denne fejl.

Fejlen kan også opstå hvis Perl-fortolkeren ikke ligger i `#!/usr/bin/perl`, hvilket er en meget lumsk fejl fordi når man kører programmet vil den svare `program.cgi: command not found` (tcsh) eller `bash: program.cgi: No such file or directory`, programmet findes og udføres, fejlen skyldes at den ikke kan finde fortolkeren.

Browseren viser en `The document contained no data.`. Denne fejl kommer hvis programmet kun udskriver HTTP-hovedet. Typisk sker der en fejl i programmet så det dør, og udskriver en fejl i Apaches fejllog, men undlader at printer en fejl først der kan vises i browseren. Istedet for die kan følgende stump kode bruges hvis man i forvejen har udskrevet HTTP-hovedet ($! er fejlteksten svarende til sidste fejl):

```
sub Fejl
#udskriver fejl til browser og error_log
{
  print "Fejl! $!";
  die "Fejl! $!";
}
```

CGI-program situationer, der typisk giver fejl:

Programmer køres af webserveren og ikke af dig. Typisk er webserver processen eget af bruger 'nobody' og group 'nobody', hvilket betyder at programmet skal have være udførbart og læsbart hvilket gøres med **chmod o+rx program.cgi**.

Hvis programmet skal gemme en fil i et katalog som du ejer, så skal det kataloget være skrivbart af webserveren. Det betyder at den bruger eller gruppe webserveren kører som (typisk »nobody« og »nobody«) skal have brugs og skrivetilladelse til kataloget. Det kan klares med kommandoen: **chmod**

o+rwx katalog. Hvis man ikke vil give alle og enhver adgang til at gemme filer i det katalog kan man bruge suExec som forklaret i afsnittet om sikkerhedsaspekter.

2.1.12. Debugging af CGI-programmer

Der opstår fejl i alle programmer. CGI-programmer er dog mere besværlige at debugge fordi de typisk skal køres fra en browser med et korrekt form input for at generere en bestemt fejl, der typisk viser sig som en "Internal Server Error" eller "The document contained no data." i browseren. Og problemet er nu at finde fejlen.

Alt tekst der udskrives til stderr vil udskrives i Apaches fejllog. Dvs. opstår der en fejl i programmet og det udføre en die "Fatal error occured!", så vil der ikke udskrives nogen information til browseren men "tail /var/log/httpd/error_log" vil udskrive fejlmeddelelsen. Det er derfor praktisk at udskrive en fejl med print før at die kaldes, så fejlen kan ses i browseren.

Den hurtigste måde at debugge et lille program på er at indsætte print-linjer på passende steder i koden, gerne med en informativ besked om, hvad koden skal til at udføre (’Åbner forbindelse til databasen...’), eller hvad den netop har konstateret (’Der blev angivet en e-postadresse, sender nu mail...’). (Det hænder dog, at et program er blevet debugget med kortere og knap så informative beskeder, så som ’A’ og ’ghjk’.)

Men er Perl-programmet meget stort eller komplekst kan det tage en del tid at finde fejlen ved at genstarte programmet gang på gang. Det er istedet mulig at fange input data og gemme dem i på harddisken, og så senere hente %data og %ENV hashes fra harddisken. CGI-programmet kan så køres i en debugger men med de samme data og systemvariable, som hvis det blev kørt af en webserver. Bemærk at det ikke køres med samme bruger og gruppe id!

Den følgende stump Perl-kode kan bruges til at debugge CGI-programmer med. Den anvender de funktioner der er erklæret i "Et udgangspunkt for et CGI-program" afsnittet. Køres programmet med $debug=0 vil det indlæse webformular data fra get eller post metoden og fortsætte, altså uden at der sker noget.

Køres programmet med $debug=1 og $gemdata=1 vil det indlæse webformular data og gemme dem i to filer. %data gemmes i en fil ved navn "grab.var", og systemvariable %ENV i en fil med navn "grab.env", programmet vil så stoppe.

Køres programmet bagefter med $debug=1 og $gemdata=0 vil programmet genoprette %data og %ENV, som da programmet kørte første gang og det vil så på en reproducibel måde behandle data, men med den fordel at dette kan ske på kommandolinjen eller i en debugger. De to filer er formateret i en syntaks, der ligner XML så det er også let at lave ændringer i datafilerne.

```
#!/usr/bin/perl
```

```
#filnavn: debug.cgi
#koden til URLdekrypt og HentPostData funktionerne er den samme som i echo.cgi

my %data;

#debug kode ------------------------------------------------

#basefilenavn
my $filename="grab";

#0 for at modtage data fra webserveren og bearbejde dem.
#1 for at slå debugging til

my $isDebug = 1;

#0 for at afspille data fra fil.
#1 for at gemme data

my $grabdata = 0;

sub Fejl
#udskriver fejl til browser og error_log
{
  print "Content-Type: text/plain\r\n\r\n";
  print "Fejl! $!";
  die "Fejl! $!";
}

sub GemHash
#Gemmer en hash til en fil i et XML lignende format.
{
   my ($hashref,$filnavn)=@_;
   my $tag,$streng;

   open OUT, $filnavn or die Fejl;
   while (($tag,$streng)=each %$hashref)
      {
#For at afkodning skal være entydig må '<' og '>' escapes.
         $streng =~ s/&/&amp;/g;
         $streng =~ s/</&lt;/g;
         $streng =~ s/>/&gt;/g;

         print OUT "<$tag>$streng</$tag>\n";
      }
   close OUT;
}

sub HentHash
#Henter en hash fra en fil.
{
   my ($hashref,$filnavn)=@_;
   my $content;
```

```
#backup af record separatoren.
   my $tmp=$/;

#indlæs hele filen på en gang
   open IN, $filnavn or die Fejl;
   $/=undef;
   $content=<IN>;

#loop gennem alle <tag>data</tag>'s.
   while ($content =~ /<(.*?)>([^<]*)<\/\1>/gc) { tohash($hashref,$1,$2); }

   $/=$tmp;
}

sub tohash
#addere hash data efter at escaped tegn er fixet
{
  my ($hashref,$key,$str)=(@_);

#Fix escaped tegn.
  $str =~ s/&lt;/</g;
  $str =~ s/&gt;/>/g;
  $str =~ s/&amp;/&/g;

  $$hashref{$key}=$str;
}

if (not $isDebug or $isDebug and $grabdata)
   {
#hvis ikke i fejlsøgningsmodus skal inddata fortolkes.
#hvis i fejlsøgningsmodus og grabdata så skal inddata også fortolkes.

        if ($ENV{'REQUEST_METHOD'} eq "POST")
          {
             %data= URLdekrypt(HentPostData());          #Post data
          }
         else
          {
             %data= URLdekrypt($ENV{'QUERY_STRING'});    #Get Data
          }
   }

if ($isDebug)
{
  if ($grabdata)
     {
#Gem systemvariable og data hash.
        GemHash(\%ENV, ">".$filename.".env");
        GemHash(\%data,">".$filename.".var");

#udskriv og stop programmet når data er gemt.
        print "Content-Type: text/plain\r\n\r\n";
        print "Data er opsamlet og gemt i \"$filename.*\".";
```

```
        print "Sæt \$grabdata=0 og kør programmet i en debugger\n";
        die;
     }
  else
     {
#Slet alle systemvariable
        %env=();

#hent systemvariable og data hash fra filer.
        HentHash(\%ENV,  $filename.".env");
        HentHash(\%data, $filename.".var");

#fortsæt CGI-programmet.
     }
}


#Her behandles webformular data.
```

Hvis programmet er skrevet i C eller C++ har man den mulighed at man kan debugge program processen mens det kører. Dette gøres ved at oversætte programmet med en option "-ggdb" for inkludere debug information, og indsætte en sleep(30) i starten af programmet (kræver unistd.h). Før programmet kører su'er man til webserver brugeren. Når man trykker Submit vil programmet pause i 30 sekunder, i den tid kan man udføre "ps aux | grep program.cgi" her kan man se programmets procesnummer og så udføre "ddd program.cgi <pid>" hvor <pid> er procesnummeret man fik fra **ps**-kommandoen. Man kan så single steppe programmet som det modtager, bearbejder og udskriver data. Dette trick virker desværre ikke med Java-, Pyton- og Perl-programmer da disse jdb, pydb og Perl-debuggere ikke kan debugge en kørende process. Og det kræver at man har nok rettigheder til at kunne attache til webserverens process.

2.1.13. Sikkerhedsaspekter

CGI-programmer modtager data fra en tvivelsom kilde, og kan bruge disse data som basis for hvilken som helst operation. I eksemplet der modtog data vha. post metoden blev problematikken omkring meget store datamængder omtalt, og koden der blev vist kan automatisk begrænse mængden af data programmet vil acceptere.

Og i afsnittet de data der var tilgængelig til et program blev skjulte variable præsenteret som en sikker måde at overføre skjult information i en webformular, men enhver kan i HTML-koden se at der er skjulte variable, og det er en let sag at ændre de skjulte variable (gem HTML-koden, ret i de skjulte variable, åben HTML-koden, indtast data, og submit dem). Er det vigtigt, at de skjulte variable ikke forfalskes må man bruge en eller anden tjeksums algoritme. Ligeledes kan en ondskabsfuld person returnere helt andre variable end dem der står i webformularen, man kan altså ikke stole på at fordi man har et name="navn" felt i formularen, at det URLenkrypteret svar vil indeholde det tilsvarende navn=.. felt. I perl vil dette svare til %data{'navn'} hvor "navn" ikke findes i hashen og den returnere "", så det er ikke noget stort problem.

En typisk problem kan være at data fra en webformular kan bruges som filnavn til at gemme information, og derfor overskrive filer der ikke burde overskrives, det bør derfor altid tjekkes om filen findes i forvejen, og hvad der så skal ske. I stedet for et filnavn kan programmet måske snydes til at anvende en kanal (engelsk "pipe"). For eksempel betyder `|program` at mens programmet tror det skriver til en fil, skriver det i virkeligheden til en kanal, der leder dataene videre til programmet **program**. På denne måde kan man altså udefra komme til at køre en vilkårlig kommando på systemet, hvis kanal-tegnet (|) ikke bliver fjernet fra filnavne.

Det kan også være farligt at udskrive for meget information om den server som programmet kører på, siden at den type information kan misbruges af hackere. Det følgende stump kode indeholder to sikkerhedsfejl:

```
#$filnavn er navnet på en fil og er indtastet i en webformular.

my $fil = "/www/filer/$filnavn";

#Indlæs og udskriv filen
```

Indtastes nu `../../etc/passwd` som filnavn i webformularen, så vil programmet vise alle brugere på serveren og deres enkrypterede passwords (med mindre shadow password bruges), og den information er tilgængelig for enhver på nettet. En hacker kunne så på sin egen computer lave et brute force angreb ved at prøve en masse passwords igennem, indtil at han rammer et password der giver det enkrypteret password, som det i password filen. Han kan så logge ind på serveren hvor CGI-programmet udføres. Moralen er at alle filnavne skal tjekkes for at sikre sig at de ikke indeholder katalog information. En anden fejl er at programmet selv udskriver siden, det betyder at webserverens adgangskontrol til sider bliver tilside sat. Den følgende kode stump tjekker om filnavnet indeholder '/' og istedet for at udskrive filen udskrives en Location til filen. På den måde virker Apaches adgangskontrol og ikke mindst adgangsbegrænsning stadig

```
#$filnavn er navnet på filen og kommer fra et en web formular.

die "Suspekt filnavn!\n" if  $filename =~ /\//;
my $fil = "/filer/$filnavn";

#udskriv en Location til filen.
```

Specielt farlige er de tre perl kommandoer eval, exec og system der gør det muligt at udføre programmer, der kører med samme rettigheder som Apache webserveren. System ved at der startes en shell (/bin/sh) og at denne modtager en streng af kommandoer, der skal udføres. Følgende program udskriver hvilke processer en bestemt bruger køre og indeholder en bagdør til webserver kontoen.

```
#$user er et bruger navn fra en webformular.

print "Content-Type: text/plain\r\n\r\n";
system("ps aux | grep $user");
```

En bruger kan nu taste "root" og se alle de tjenester der kører på maskinen. Det giver en idé om hvilke sikkerhedshuller der kan være i systemet. Men som om det ikke var nok så kan enhver taste "**tyge ; rm**

-rf /" i webformularen.

Sker dette vil programmet udføre "**ps aux | grep tyge ; rm -rf /**", hviket betyder at programmet efter at havde udskrivet alle tyge's processer vil fortsætter med at rekursivt slette alle filer på filsystemet. Typisk vil webdeamonen kun have adgang til at slette alle webfiler, og de data som andre CGI-programmer har gemt. Men det kan være slemt nok hvis serveren er en dedikeret webserver for et firma. Alternativt kunne en indtastning "**tyge ; ls -la /home/**" udskrive alle hjemmekataloger og deres adgangsrettigheder. Hvis der er et hjemmekatalog, der er læsbart for webserveren kan hackeren læse alle filer, som den bruger har, og skrive til alle de filer, der er skrivebare for alle brugere, hvis der findes en .rhosts fil og hackeren kan skrive til den, vil han kunne logge ind som den bruger uden password fra en vilkårlig server i verden. Hackeren ville også kunne udsende breve, der tilsyneladende kommer fra webserveren.

Men det er stadig kun småting en hacker kan med dette simple program oploade en terminal server (f.eks. netcat), oversætte den og køre den, således at han kunne logge ind på webserveren med samme rettigheder som webserveren og uden noget password. Og når en hacker har adgang til at køre vilkårlige kommandoer kunne han tjekker versionsnumre på alle deamons og sammenligne disse med kendte sikkerhedsfejl indtil han finder en bagdør til systemadministratorkontoen (root).

Derfor bør man i et CGI-program altid undgå exec og system kommandoer. Med eval kan ethvert perl udtryk udføres, dvs. ethvert perl udtryk der åbner filer, sletter filer, udfører system eller exec. Det er derfor umuligt at tjekke at et eval udtryk ikke indeholder farlige kommandoer.

2.1.13.1. Tainted variable

Perl har indbygget en smart funktion der gør det muligt at mærke variable, der kommer fra farlige kilder (en farlig variabel kaldes "tainted" der betyder noget i retning af mærket/beskidt/usikker), systemvariable, input, og output som andre programmer laver er automatisk tainted. Bruges en tainted variabel som filnavn eller som argument i exec eller system kald, vil Perl automatisk stoppe med en fejl. Tainted variable slås til ved at kalde Perl som "#/usr/bin/perl -T" og de slås automatisk til hvis programmet er setuid dvs. "**chmod ug+s program.cgi**", således at programmet kan køres af webserveren med dine rettigheder.

Hvis man tager en delstreng ud af en tainted streng vil delstrengen også være tainted, og konkateneres en tainted og en normal streng bliver resultatet også tainted. Man kan derfor være rimelig sikker på at der opstår en fejl, hvis man forsøger at bruge usikker information i en farlig operation. Den eneste måde at untainte en streng på er ved at ekstrahere den fra et regulært udtryk, f.eks. et der fjerner forbudte tegn som ';','|' og '/'. Se "man perlsec" for mere information om tainted variable.

Det følgende eksempel bruger tainted variable, og viser hvordan det untaintes.

```
#!/usr/bin/perl -T

#hent variable fra webformularen.
#$user er et brugernavn indtastet i webformularen.
#det er derfor automatisk tainted.
```

```
if ($user =~ /^(\w+)$/)
 {
   $user = $1;

#$user er nu untainted og kan kun indeholde
#følgende tegn A-Z,a-z,0-9, samt '_'
 }
else
 {
   die "Suspekt brugernavn!\n";
 }

die "Fy! fingerene væk!" if ($user =~ /root/);

print "Content-Type: text/plain\r\n\r\n";
system("ps aux | grep $user");
#da $user er untainted generere dette ikke en fejl.
```

Hvis flere bruger skal have adgang til at køre programmer på den samme webserver, så kan det være praktisk at anvende suEXEC (Swich User for Exec). suEXEC er et add-on til Apache, der tillader CGI-programmer at køre med samme rettigheder som den individuelle bruger, der har lavet programmet istedet for som webserveren. Det er langt sikrere end at setuid'e programmet med "**chmod ug+s program.cgi**", men kræver også at det bliver korrekt sat op for at undgå sikkerhedshuller og Apache genoversættes, det er derfor ikke inkluderet i den Apache version der normalt distribueres. Dokumentation hvordan Apache genoversættes med suEXEC indbygget er inkluderet i Apache dokumentationen og kan findes i /home/httpd/html/manual/suexec.html. suEXEC giver også mulighed for at hver bruger har sin egen error_log fil hvilket letter debugging af programmer.

2.1.14. FastCGI

En ulempe ved at skriver CGI-programmer i Perl er at hver gang de køres skal Perl-fortolkeren starte og den skal fortolke den samme Perl-kode igen og igen, og det tager en del tid hvilket kan være upraktisk hvis programmet skal køres mange gange i sekunded.

Bedre ville det være hvis CGI-programmet kørte hele tiden, og kunne modtage mange data uden at skulle genstartes. Dette er hvad fastCGI gør muligt ved at definere en protokol for hvordan en process kan kommunikere med webserveren's CGI-modul.

2.1.14.1. Installation af FastCGI

Installér apache-devel programpakken (indeholder apxs programmet).

Hent kildeteksten til FastCGI apache-modulet fra http://www.FastCGI.com (http://www.fastcgi.com)

Oversæt koden med:

```
apxs -c -o mod_fastcgi.so *.c
```

Installér Apache-modulet med:

```
apxs -i -a -n fastcgi mod_fastcgi.so
```

Tjek at mod_fastcgi er inkluderet i httpd.conf og at path til det dynamiske bibliotek er korrekt.

Adder følgende linje til srm.conf

```
AddHandler fastcgi-script fcg fcgi fpl
```

Genstart Apache:

```
[root@hven /root]# /etc/rc.d/init.d/httpd restart
```

Apache webserveren understøtter nu fastCGI.

På http://www.cpan.org kan man hente et FCGI-modul der tillader Perl-programmer at anvende fastCGI-modulet. Perl-modulet som andre Perl-moduler med:

```
[tyge@hven ~]$ tar xzvf FCGI-0.53.tar.gz
[tyge@hven ~]$ cd FCGI-0.53/
[tyge@hven ~/FCGI-0.53]$ perl Makefile.PL
[tyge@hven ~/FCGI-0.53]$ make && echo O.k.
...
O.k.
[tyge@hven ~/FCGI-0.53]$ su -c 'make install&& echo O.k.'
...
O.k.
[tyge@hven ~/FCGI-0.53]$ cd ../
[tyge@hven ~]$ rm -rf FCGI-0.53/
```

2.1.14.2. Brug af FastCGI

Gem følgende program som fastcgi_test.fcgi og kør det flere gange i browseren.

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

#filnavn: fastcgi_test.fcgi
my $count=0;

my $request = FCGI::Request();

while($request::accept() >= 0)
{
   print "Content-Type: text/html\r\n\r\n";
   print "<h1>FastCGI test</h1>\n";
   print "<p>Antal af forbindelser for dette program: ++$count</p>\n";

#her kan resten af CGI-programmet udføres.
}
```

Det specielle ved FastCGI-programmer er at de kan huske tidligere forbindelser. $count i eksemplet tælles op for hver gang programmet modtager en forbindelse. Efter at fcgi-programmet er udført kan man med "ps aux | grep perl" se at det venter på næste gang det skal modtage data. En forskel på FastCGI og normale Perl-programmer er at Perl-programmer løbende udskriver print bufferen til webserveren, mens FastCGI-programmer kun returnere data når de afsluttes, eller hvis bufferen flushes med $request->Flush().

Hvis man ændrer kildeteksten, og prøver programmet igen skal man være opmærksom på at man kan fælde et antal af gamle fcgi-programmer, der må dræbes manuelt med "kill <pid>". FastCGI-programmer kan også laves i C og Tcl se http://www.fastcgi.com for mere dokumentation om protokollen og hvordan den anvendes.

Kapitel 3. Server-Side Includes

Apache har en parser indbygget, så når en webside afsendes kan Apache indsætte stumper af HTML-kode for eksempel hvis alle sider skal have et fælles hoved eller fod. Apache kan også indsætte den dato hvor HTML-filen sidst blev rettet, og f.eks. inkludere eller udelukke HTML-kode alt efter hvilken browser der læses med.

Alle disse features kaldes under et for Server-Side Includes, og forkortes SSI. SSI gør mange trivielle ting lettere, men det betyder også at sider skal fortolkes før de afsendes, hvilket betyder at det tager længere tid at sende en webside.

3.1. Opsætning af Apache for at udføre SSI

For at Apache skal udføre SSI, skal følgende linjer indsættes eller udkommenteres i `/etc/httpd/conf/httpd.conf` eller en af de filer den inkludere f.eks. `/etc/httpd/conf/srm.conf`

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Alle filer med typen `.shtml` bliver så fortolket af Apaches SSI-modul (mod_include), og resultatet udskrives med mediatypen text/html, dvs. som en HTML-fil.

For at slå SSI til for et bestemt katalog f.eks. `/var/www/html` skal **"Options Includes"** inkluderes i katalogets opsætning, hvor disse befinder sig i enten filen `/etc/httpd/conf/httpd.conf` eller filen `/etc/httpd/conf/access.conf`, f.eks.:

```
<directory /var/www/html >
Options Includes
</directory>
```

Bruges **IncludesNOEXEC** istedet for Includes, så kan dynamisk genereret output ikke inkluderes, dvs. alle inkludes der anvender CGI-programmer eller udfører kommandofortolker negligeres.

Filer som ender på `.html` eller `.htm` vil ikke blive fortolket. Kun filer som slutter på `.shtml`. Det kan dog være praktisk også at fortolke visse `.html` og `.htm` filer. Dette kan vha. XBitHack kommandoen som følger

```
<directory /var/www/html >

# "*.shtml"-filer fortolkes
Options Includes
```

```
#filer med chmod u+x fil fortolkes
XBitHack on
</directory>
```

Alle filer hvor brugeren har rettigheder til at udføre dem vil nu blive fortolket (**chmod u+x fil.html**). Normalt bliver SSI-svar ikke cachet, fordi deres indhold kan variere fra gang til gang, for eksempel ved at dags dato indsættes et sted. Men hvis de HTML-stumper der indsættes er statiske, dvs. at man f.eks. indsætter den samme HTML-kode som hoved og fod hver gang, så kan siden jo sagtens caches uden problem.

Man kan gøre dette muligt med "XBitHack full". Alle filer hvor både brugeren og gruppen har rettighed til at udføre filen, vil nu blive afsendt med et "Last-Modified"-felt i HTTP-hovedet, som fortæller browseren, hvornår filen sidst blev ændret (**chmod ug+x fil.html**).

Bruger man "XBitHack full" og ændrer i de HTML-stumper der indsættes som hoved og fod, vil det derfor være hensigtsmæssig at 'touch'e alle filer der får hoved og fod indsat, således at den dato der returneres altid er nyere end HTML-stumpernes, og browseren er i stand til at cache svaret. Følgende kommandoer finder alle filer i og under kataloget `/var/www/html`, der kan udføres af brugeren og gruppen (ug+x), og opdaterer tiden på alle filerne:

```
[tyge@hven tyge]$ find /var/www/html -perm 110 | xargs touch
```

3.2. Inkludering af tekst i et HTML-dokument

Typisk bruges SSI til at inkludere et fælles hoved i alle .shtml-filer. Dette kan gøres med kommandoen **#include**, der har følgende syntaks:

```
<!--#include virtual="URL" -->
```

Dette er en SSI-kommando, og alt mellem **<..>** vil blive erstattet af indholdet i af den fil, som URL'en peger på. I alle SSI-udtryk er det meget vigtig, at der er styr på mellemrum. Der må ikke være mellemrum mellem **<!--#kommando** og der SKAL være et mellemrum før den afsluttende **-->**. Følgende er et eksempel på hvordan et hoved og en fod kan inkluderes i et HTML-dokument:

```
<body><!--#include virtual="/hoved.html" -->

<h1>Titel</h1>
<p>Dokumentets indhold</p>

<!--#include virtual="/fod.html" --></body>
```

Her ligger både "hoved.html" og "fod.html" i hjemmesidens rod `/`. Inkluderede filer kan dog ligge hvor som helst inden for hjemmesidens rod.

Hvis URL'en ikke starter med `/`, så findes filerne med html-stumperne relativt til den fil de skal indsættes i. En vigtig detalje er at URL'en til den html-stump, der skal inkluderes, ikke må starte med en protokol f.eks. http://. Den må kun være en absolut eller relativ sti til en fil.

3.3. Inkludering af uddata fra CGI-programmer og andre programmer

Uddata fra et CGI-program kan også inkluderes f.eks.

```
<!--#exec cgi="/cgi-bin/hoved.cgi" -->
```

Alt imellem **<..>** vil blive erstattet af de data som programmet `hoved.cgi` udskriver. Hvis CGI-programmet udskriver en omdirigering af formen

```
Location: URL\r\n\r\n
```

indsættes `URL` som et link istedet.

Det følgende eksempel bruger et CGI-program til at generere **<body bgcolor="#farve">**, hvor farven afhænger af tiden.

```
<!--#exec cgi="/cgi-bin/colorbody.cgi" -->
 <h1>Test</h1>
</body>
```

Og koden til `colorbody.cgi` er et lille Perl-program

```
#!/usr/bin/perl

#find RRGGBB farve på basis af tiden
my $color=sprintf("%x", ( time*25 % 0x1000000 ));

print "Content-Type: text/html\r\n\r\n";

#udskriver <body ..> tag
print "<body bgcolor=\"#$color\">\n";
```

Programmet fungerer ved at `time` er antallet af sekunder siden 1 jan. 1970. Dette ganges med 25 der er 'hastigheden' hvormed farven ændres og modulus 0x1000000 tages, det giver et resultat mellem 0x000000 og 0xffffff, dvs. alle mulige farver. Intensiteten af den blå komponent cykler med en periode på 10 sekunder, den grønne komponent med en periode på 43 minutter og den røde komponent varierer med en periode på omkring en uge.

Browseren modtager HTML-kode som:

```
<body bgcolor="#ff1adc78">
 <h1>Test</h1>
</body>
```

Reloades siden så vil baggrundfarven ændre sig fra gang til gang. NB. dette kunne gøres smartere med Javascript, men det kræver at browseren kan køre Javascript-programmer og at brugeren ikke har slået det fra.

Det er også muligt at udføre programmer direkte og inkludere deres output f.eks.

```
<!--#exec cmd="who" -->
```

Apache vil starte en shell og udføre **who**-programmet, der lister alle, der er logged ind på serveren.

Et mere avanceret eksempel er følgende:

```
<pre>
De sidste 50 sider du hentede på denne server var:
<!--#exec cmd="cat /var/log/httpd/access_log | grep ${REMOTE_ADDR} | tail -n50" -->
</pre>
```

Først udskrives Apache's log af alle filer den har afsendt. Dette filteres af grep der udskriver alle linjer der matcher IP addressen på den server der ønsker denne side, og tail -n50 udskriver de sidste 50 linjer. Dvs. de sidst 50 filer som den server hentede. Er access_log filen stor tager det dog en del tid at udføre.

Der er nogle sikkerhedaspekter i at give alle og enhver bruger på serveren lov til at bruge webserver-processerne til at udføre arbitrære kommandoer med arbitrære inddata. Med **Options IncludesNOEXEC** vil ovenstående #exec kommandoer ikke kunne udføres.

3.4. Inkludere datoer i HTML-sider

Enhver fil har indbygget en tid og dato for hvornår den sidst blev ændret, og mange på nettet vil være interesseret i at se hvornår den side de læser sidst blev opdateret. Istedet for at skrive dette direkte i HTML-koden, således at man manuelt skal justere datoen hvergang man retter i filen, så kan man anvende SSI til at automatisk indsætte fil datoen som følger:

```
<!--#echo var="LAST_MODIFIED" -->
```

Dette vil udskrive datoen som "Friday, 04-Aug-2000 21:00:41 CEST", hvilket ikke i et særligt anvendeligt format på dansk!

Formattet for hvordan tider udskrives kan ændres med **<!--#config timefmt="%d/%m %Y" -->** og indsættes før fil-datoen vil, så den udskrives som "04/08 2000". %d bliver substitueret med numret på dagen i måneden. En fuldstændig liste af alle koderne kan findes med **man strftime**.

Tabel 3-1. De mest interessante koder:

%S	Sekunder (00-61)
%M	Minut (00-59)
%H	Time (00-23)
%a	Forkortet navn på dag (Man)
%A	Navn på dag. (Mandag)
%e	Nummer dag i denne måned (1-31)
%j	Nummer dag i dette år. (001-365)
%u	Nummer dag i ugen. (1-7) 1 er Mandag.
%b	Forkortet månedsnavn (Jan)
%B	Månedsnavn (Januar)
%m	Månedsnummer (01-12)
%y	Årsnummer (00-99)
%Y	Årsnummer med 4 cifre (2000)

Alle navne er relative til det nuværende lokale, og der tages også hensyn til skudsekunder. Man kan også inkludere dagens dato ved at bruge **<!--#echo var="DATE_LOCAL" -->**, hvilket udskriver lokal tiden, mens **<!--#echo var="DATE_GMT" -->** udskriver datoen i Greenwich Mean Time (GMT).

3.5. Inkludere information om filer

Når man kan hente store billeder eller filer, er det praktisk hvis størrelsen på filen står et eller andet sted, så man ikke ender med at hente 20 Mb via en langsomt modemforbindelse. I stedet for at skrive dette i HTML-koden kan man med SSI automatisk indsætte størrelsen af filen:

```
<!--#fsize virtual="storfil.tgz" -->
```

Størrelsen af `storfil.tgz` bliver så indsat for eksempel 11k eller 8.2M.

Hvis man istedet vil have antallet af bytes, kan man bruge

```
<!--#config sizefmt="bytes" -->
```

Apache vælger automatisk at udskrive den mest fornuftige størrelse i kilo eller megabyte alt efter hvor stor filen er, dette svarer til følgende config kommando:

```
<!--#config sizefmt="abbrev" -->
```

På samme måde kan man indsætte tiden for, hvornår filen sidst blev ændret med

```
<!--#flastmod virtual="storfil.tgz" -->
```

Formatet for uddata fra denne kommando kan konfigueres ligesom i sidste afsnit.

3.6. Fejlmeddelelser

Hvis der opstår en fejl under afvikling af en SSI-kommando, vil den voldsomt informative fejlmeddelelse udskrives:

```
[an error occurred while processing this directive]
```

Hvis man ønsker at tilrette denne fejlmeddelelse kan det gøres med

```
<!--#config errmsg="(SSI-Fejl: Sand i maskineriet, støvsug mig!)" -->
```

Enhver tekststreng kan bruges som fejlmeddelelse. Fejlmeddelelser vil for eksempel blive udskrevet, hvis man forsøger at indsætte kodestumper der ikke findes, hvis man forsøger at indsætte uddata fra et CGI-program med #include og ikke #exec, eller at indsætte en fil i sig selv.

3.7. Variable

I virkeligheden betyder **<!--echo var="LAST_MODIFIED" -->** at variablen LAST_MODIFIED skal udskrives, så man spørger sig selv, hvilke andre variable findes der? Der findes en kommando der udskriver alle de variable, der er kendt:

```
<pre>
<!--#printenv -->
</pre>
```

Følgende er et udklip af hvad ovenstående eksempel udskriver:

```
HTTP_USER_agent=Mozilla/4.73 [en] (X11; U; Linux 2.2.16-3 i686)
REMOTE_addr=127.0.0.1
REMOTE_host=localhost
SCRIPT_filename=/www/cgibog/test.shtml
REQUEST_uri=/cgibog/test.shtml
DATE_local=Sunday, 06-Aug-2000 13:11:39 CEST
DATE_gmt=Sunday, 06-Aug-2000 11:11:39 GMT
LAST_modified=Sunday, 06-Aug-2000 13:11:38 CEST
DOCUMENT_uri=/cgibog/test.shtml
:
```

Mange af disse variable er de samme, som et CGI-program modtager. Normalt modtager man ikke DNS-navnet på den server der ønsker data (REMOTE_HOST), men kun dens IP-adresse, slå **HostNameLookups On** for et katalog definition i **`/etc/httpd/conf/access.conf`** så vil REMOTE_HOST indeholde server navnet. Miljø-variable oprettes, når Apache begynder at fortolke den .shtml-fil, som browseren beder om, og indsættes en .shtml-fil i en anden, vil den .shtml-fil der indsættes have de samme miljø-variable til rådighed.

Det er muligt selv at erklære variable, dette gøres med

```
<!--#set var="minvariabel" value="Dette er indholdet af min variabel" -->
```

Det er også muligt at anvende andre variable til at definere nye variabler f.eks.:

```
<!--#set var="ServerFil" value="${REMOTE_ADDR} -> ${DOCUMENT_URI}" -->
```

Variablen ServerFil vil så i ovenstående eksempel indeholder strengen "127.0.0.1 -> /cgibog/test.shtml".

3.8. Logiske udtryk

Det er muligt at anvende logiske udtryk i SSI. Dette minder lidt om #define, #ifdef,#else og #endif i C/C++. Syntaksen i SSI er:

```
<!--#if expr="betingelse" -->
<!--#elif expr="betingelse" -->
<!--#else -->
<!--#endif -->
```

Tabel 3-2. Betingelser:

Streng	Er sandt hvis strengen ikke er tom.
Streng1=Streng2	Er sand hvis Streng1 og Streng2 er identiske.
streng=/regularudtryk/	Er sand hvis regular udtrykket matcher strengen.
Streng1!=Streng2	Er sand hvis Streng1 og Streng2 er forskellige
betingelse1 && betingelse2	Er sand hvis BÅDE betingelse1 OG betingelse2 er sande, ellers falsk.
betingelse1 || betingelse2	Er sand hvis ENTEN betingelse1 ELLER betingelse2 er sande.
!betingelse	Er sand hvis betingelsen er falsk, og falsk hvis betingelsen er sand. (negation)
(betingelse)	Er sand hvis betingelsen er sand, ellers falsk.

Strenge må ikke indeholde whitespace, dvs. mellemrum, tabulartor oa. Skal strenge matches der inderholder whitespace må de skrives med aprostroffer som 'string1' = 'streng2'. Apaches SSI bruger den

samme dialekt til regulære udtryk som egrep. For mere information om regulære udtryk se **man egrep**.

Et eksempel:

```
<body>
<!--#if expr="${HTTP_USER_AGENT} = /Mozilla/" -->

  Du bruger Netscape!
     <!--#if expr="${HTTP_USER_AGENT} = /Gecko/" -->
       Og din browser er bruger Gecko rendering engine!
     <!--#endif -->

<!--#elif expr="${HTTP_USER_AGENT} = /MSIE/" -->

  Du bruger Explorer.

<!--#else -->
  Din browser kan ikke genkendes.
<!--#endif -->
</body>
```

I dette eksempel bruges SSI til at genkende hvilken browser der bruges, og sende specielle HTML-koder til de forskellige browsere. HTTP_USER_AGENT-linjen fortæller hviken browser brugeren anvender.

Tabel 3-3. HTTP_USER_AGENT for nogle typiske browsere

Netscape 4.73	Mozilla/4.73 [en] (X11; U; Linux 2.2.16-3 i686)
Galeon	Mozilla/5.0 (X11; U; Linux 2.2.16-3 i686; en-US; Galeon) Gecko/20000807
Mozilla M17	Mozilla/5.0 (X11; U; Linux 2.2.16-3 i686; en-US; m17) Gecko/20000807
Lynx	Lynx/2.8.2rel.1 libwww-FM/2.14

Slår man **HostnameLookups** til så modtager man DNS navnet på den server der ønsker data. Det giver en række interessante muligheder f.eks.:

```
<body>
<!--#if expr="${REMOTE_HOST} = /microsoft.com/" -->

Desværre - denne side er ikke tilgængelig for Microsoft-ansatte!

<!--#else -->

Dette er Tux - den mest magtfulde pingvin i verden!

<!--#endif -->
</body>
```

Programmet vil returnere én side til alle folk der surfer fra maskiner på Microsofts netværk og én anden side til resten af verden.

3.9. Mere information

Reference-information kan findes i Apache-dokumentationen, der typisk ligger i `/var/www/html/manual/mod/mod_include.htm`.

Kapitel 4. PHP: Hypertext Processor

PHP er et scriptsprog, der kan inkluders direkte i html sider. Til forskel fra JavaScript, der fortolkes af browseren, fortolkes PHP scripts af Apaches serverens PHP4 modul, og resultatet sendes så til brugerens klient. Et simpel script kunne se ud som følgende HTML fil:

```
<body>
<h1>PHP test eksempel.</h1>
<?php
  echo "Denne linje udskrives af PHP fortolkeren";
?>
</body>
```

Browseren vil så modtage følgende HTML kode

```
<body>
<h1>PHP test eksempel.</h1>
Denne linje udskrives af PHP fortolkeren
</body>
```

PHP har en lang række features, der gør det meget praktisk og anvendeligt til at lave dynamiske html sider, sammenlignet med CGI scripts. PHP kan udskrive HTML sider med data fra databaser, lave billeder, modtage data fra webformulare, kommunikere vha. XML formateret filer. PHP kan også kommunikere med e-post, FTP, POP3, IMAP, LDAP, FTP, MySQL Afsnit 6.3.1, PostgreSQL Afsnit 6.2.2 og kender til mange andre databaser, protokoler og akronymer.

Formålet med det følgende kapittel er en introduktion til variabler, syntax, og grundlæggende funktioner i PHP scriptsproget. Med PHP følger en excellent manual, der indeholder en masse reference information om PHP sprogets mange funktioner. Endnu mere information, kode eksempler m.m. kan findes på nettet et godt udgangspunkt er http://www.php.net (http://www.php.net/). Eller også kan du kigge lidt på kursmaterialet til "Frit PHP-kursus" på http://www.chbs.dk/kurser/php/.

På n

4.1. Installation

RPM pakker følger med Red Hat. Disse hedder mod_php*.rpm og php*.rpm, men Pentium optimerede versioner kan hentes fra Troels Arvins Php homepage, der kan findes på http://rpms.arvin.dk/php/. Den nyeste version af PHP er version 4. Apache configureres til at køre PHP scripts ved at inkludere følgende i `/etc/httpd/conf/httpd.conf`.

```
LoadModule php4_module /usr/lib/apache/libphp4.so
AddModule mod_php4.c
```

```
<IfModule mod_php4.c>
        AddType application/x-httpd-php .php .php3 .phtml
        AddType application/x-httpd-php-source .phps
</IfModule>
```

Her har vi ikke tilladt at brugere laver PHP-kode inden i almindelige .html-filer, men lagt om til at disse hedder .phtml. Vil vi også gerne have at .html-filer også godt kan indeholde PHP-kode, så anvendes følgende opsætning i stedet. Det vil være lidt langsommere, men det er et valg.

```
LoadModule php4_module /usr/lib/apache/libphp4.so
AddModule mod_php4.c

<IfModule mod_php4.c>
        AddType application/x-httpd-php .php .php3 .phtml .html
        AddType application/x-httpd-php-source .phps
</IfModule>
```

Disse linjer inkluderes sikket automatisk af installationsprogrammet. De første to linjer inkludere selve PHP4 modulet, mens <IfModule> definere hvilke typer af filer, der skal fortolkes af PHP modulet. PHP modulet har sin egen opsætningsfil der ligger i `/etc/php.ini`. Efter PHP er installeret kan Apache genstartes med følgende kommandoer:

```
[root@hven /root]# apachectl configtest
Syntax OK
[root@hven /root]# apachectl restart
/usr/sbin/apachectl restart: httpd started
```

Hver gang computeren bliver genstartet vil PHP-modulet automatisk blive indlæst.

Du kan nu eksperimentere med at lave PHP scripts. Start med at gemme ovenstående script som `test.php`, og prøv at hente scriptet i en browser.

4.2. Simple datatyper

PHP har de velkendte variabel typer af heltal, floating point tal, og strenge. Variabler deklereres ikke på forhånd som i C. Men defineres ved at tilskrive variablen en værdi. Som i Perl starter alle typer af variabler med dollartegn, men til forskel fra Perl bruges at og procent tegnene ('@' og '%') ikke til mærke variable, der er tabeller eller associative tabeller. Følgende eksempel demonstrere hvordan variable oprettes og siden udskrives.

```
<pre>
<?php
# Opret variabel
  $a= 42;    # et heltal
  $b= 43.0;  # et floating point tal
  $c= "44";  # en streng
```

```
# Udskriv
  echo "\$a=$a, \$b=$b og \$c=$c \n";

# Udregninger
  $d=$a+$b;
  $e=$a+$c;

# Udskriv resultat
  echo "\$d=$d og \$e=$e \n";
?>
</pre>

Udskriver:
$a=42, $b=43 og $c=44
$d=85 og $e=86
```

Som i Perl bliver værdien af variabler indsat automatisk når udskrives med som **echo "$a"**, mens **echo '$a'** bare ville udskrive $a (variabel interpolation). Som i Perl betyder koden "\n" linjeskift. Bemærk at output jo vises af browseren, og browseren ignorere linjeskift med mindre teksten vises som preformateret. Dette er grunden til at PHP scriptet er foldet ind i <pre>..</pre> tags, som vist i eksemplet.

Eksemplet viser også at det er muligt at tage summen af et tal og en streng for eksempel: 42+"44" og resultatet er 86. I artitmetiske operationer, der involvere strenge, vil strengen automatisk konverteres til en heltal eller reelt tal (se **man strtod** for konverteringen).

Følgende eksempel viser nogle grundlæggende operationer på strenge.

```
<pre>
<?php
   $a="Tux Penguin\n";

   print "Streng: $a";
   print "Længde:". strlen($a) ."\n";
   print "Tegn på tredie position: $a[2]\n";

#konkatenering:
   $b  = "Første linje\n";
   $b .= "Anden Linje\n";

   print $b;

#lange strenge med doc operatoren

   $c= <<<EOT
Lange strenge der fylder flere
linjer, skrives bedst ved at bruge
doc (<<<) operatoren.
EOT;
```

```
   echo $c;
?>
</pre>

Udskriver:
Streng: Tux Penguin
Længde:12
Tegn på tredie position: x

Første linje
Anden Linje

Lange strenge der fylder flere
linjer, skrives bedst ved at bruge
doc (<<<) operatoren.
```

Som i Perl konkateneres strenge ved at bruge punktum. Strenge i PHP opfører sig som en tabel, og man kan derfor ekstrahere enkelt tegn med **$a[position]** hvor position går fra 0 til strenglængde-1, strenge slutter med 0. Dette er årsagen til at længden 12 er et tegn længere end strengen, der er vist i eksemplet. Eksemplet demonstrerer også hvordan man kan skrive strenge, der fylder flere linjer med <<< operatoren. Bemærk at det er vigtigt at der ikke står nogen mellemrum før slut mærkatet (her EOT).

4.3. Tabeller

Dette PHP script demonstrere nogle grundlæggende egenskaber ved tabeller.

```
<?php
#simpel tabel
  $a[0]=1;
  $a[1]=2;
  $a[2]=3;

#adder streng til første frie index i tabel
  $b[]="ost";     # =$b[0]
  $b[]="smør";    # =$b[1]
  $b[]="brød";    # =$b[2]

#definer tabel med array funktionen
  $c=array("Pascal","Basic","Fortran");

#alfabetisk sortering
  sort($c);

#definer 2D tabel:
#1 index   $d[0]         ,$d[1]
#2 index         [0] [1]        [0] [1]

  $d=array(array("a","b"),array("c","d"));
```

```
#udskriv nogle af elementerne
  echo "$a[0] $b[1] $c[0] {$d[0][1]}\n";
?>

Udskriver:
1 smør Basic b
```

Tabeller begynder ved index 0, og en tabel defineres ved at man tilskriver elementerne i tabellen værdier, til forskel fra Perl bruges dollartegnet også til at referere til tabeller (Perl bruger @ for tabel kontekst). **$b[] =**streng notationen betyder adder strengen til den første element i tabellen. Sort funktionen sortere strengene i tabellen i alfabetisk rækkefølge. Det er også muligt at definere multi-dimensionale tabeller ved at bruge **array(array(),array())** koden. Når en 2D tabel skal udskrives bruges '{}', dette skyldes at $d[0][1] fortolkes både som variablen "$d[0][1]" eller variablen "$d[0]" fulgt af strengen "[1]". Bracket {..} viser at vi ønsker at udskrive det andet element i den første tabel. Et alternativ ville være at bruge konkateneringsoperatoren: **echo '\$d[0][1]='.$d[0][1]."\n".**

4.4. Associative tabeller

En associativ tabel er en tabel, der bruger strenge som indices, og de "associeres" med en streng eller talværdi.

```
<?php
#associativ tabel
   $Disney["fugl"]="Anders And";
   $Disney["hund"]="Pluto";

#array notation
#              nøgle      værdi
   $logo=array("Tux"   => "Penguin",
               "Gnome" => "Fod",
               "Billg" => "Snegl");

#multi-dimensional array
   $kunst=array(
         "impressionism" =>
               array("Monet" => "Åkander",
                     "Manet" => "Frokost i det Grønne"),
         "barok" =>
               array("Bach"  => "Brandenburg koncerterne",
                     "Hendel"=> "Messias")
           );

#udskriv

   echo $Disney["hund"] ."\n";
   echo $logo[Gnome] ."\n";
   echo $kunst["impressionism"]["Manet"] ."\n";

#sorteringseksempler
```

```
   udskrivtabel("Usorteret     : ",$logo);

   asort($b);
   udskrivtabel("Værdi sorteret: ",$logo);

   ksort($b);
   udskrivtabel("Nøgle sorteret: ",$logo);

#definer en udskrivningsfunktion:

function udskrivtabel($streng,$tabel)
  {
   echo "\n".$streng;

#loop gennem alle par af nøgler og værdier i tabellen
   foreach($tabel as $key=>$val)
      {
         echo "$key => $val\t";
      }
  }
?>

Udskriver:
Pluto
Fod
Frokost i det Grønne

Usorteret     : Tux => Penguin  Gnome => Fod    Billg => Snegl
Værdi sorteret: Gnome => Fod    Tux => Penguin  Billg => Snegl
Nøgle sorteret: Billg => Snegl  Gnome => Fod    Tux => Penguin
```

Eksemplet viser hvordan associeret tabeller kan oprettes og udskrives. Eksemplet viser også hvordan en associativ tabel kan sorteres efter nøgler og værdier med ksort og asort funktionerne. En funktion defineres til at simplifiere udskrivningen af den associative tabel. Funktionen bruger **foreach** kommandoen til at iterere gennem alle elementer i tabellen, og udskrive nøgle og værdi for alle elementer i tabellen. Koden "\t" betyder tabulator.

4.5. Funktioner

```
<pre>
<?php
# Funktion der udskriver en streng efterfulgt af en tilfældig type af fisk.
# rand(a,b) funktionen returnere et tilfældigt tal mellem a og b inklusiv.
  function AddAFish($str)
   {
      $fisk=array("Pirana","Elektrisk Ål","Haj","Guldfisk");

      $i=rand(0,count($fisk)-1);
      echo $str." ";
      echo $fisk[$i];
```

```
      echo "\n";
   }

# Returnere den $count gange konkatenerede streng.
  function StrengGange($str,$count)
   {
     for ($i=0;$i<$count;$i++) $retstr.=$str;
     return $retstr."\n";
   }


# Enkryptere strengen via en reference
  function Encrypt(&$s)
   {
    for ($i=0;$i<strlen($s);$i++)
     {
       $s[$i]=chr(ord($s[$i])+1);
     }
   }

  $str="Hello World";

  AddAFish($str);
  echo StrengGange("Test",4);

# Overfør reference til funktionen:
  Encrypt(&$str);
  echo $str;
?>
</pre>

Udskriver:
Hello World Guldfisk
TestTestTestTest
Ifmmp!Xpsme
```

AddAFish funktionen modtager en streng og udskriver strengen igen efter fulgt af en tilfældig fisk fra fisk tabellen (Helt klart en funktion ethvert website har brug for!). AddAFish funktionen returnere ikke nogle data. StrengGange funktionen modtager to argumenter og returnere en streng. Encrypt funktionen er mere interessant, den modtager en reference (&$str) til strengen $str. Det betyder at $s variablen i Encrypt er præcist den samme variabel som den globalt defineret $str variabel, og Encrypt funktionen ændrer derfor $str direkte, og behøver ikke at returnere en streng. Selve Encrypt funktionen enkryptere strengen ved at erstatte alle tegn med det næste tegn i ASCII-tabellen dvs. 'a' bliver til 'b' osv.

4.6. Logiske Kontrolstrukturer

Det følgende eksempel viser syntaxen for de logisk strukturer i PHP.

```
<pre>
```

```
<?php
  echo "If:\n";

   $a=1;
   $b=2;

   if ($a<$b)
       {
           echo "$a mindre end $b";
       }
    else
       {
           echo "$a ikke mindre end $b";
       }

  echo "\n\nIf elseif .. else:\n";

   $c="Tux";
   if
       ($c=="Tux")   {echo "En penguin";}
     elseif
       ($c=="ESR")   {echo "Eric Raymond";}
     elseif
       ($c=="RMS")   {echo "Richard Stallman";}
     elseif
       ($c=="BillG") {echo "Hvem?";}
      else
                     {echo "Ukendt";}


   echo "\n\nSwitch:\n";

   $d=2;

   switch ($d)
    {
     case 0:
       echo "zero"; break;
     case 1:
       echo "one";  break;
     case 2:
       echo "two";  break;
     default:
       echo "Unknown";
    }
?>
</pre>

Udskriver:
If:
1 mindre end 2

If elseif .. else:
```

```
En penguin

Switch:
two
```

4.7. Loops

Syntaxen for for, while og do-while loops er som i Perl og C, og er vist i det følgende eksempel:

```
<pre>
<?php
   echo "\n\nFor loop:\n";

   echo "2-tabellen: ";
   for ($i=0;$i<10;$i++) echo 2*$i." ";

   echo "\n\nWhile loop:\n";

   $i=42;
   while ($i>30)
       {
         echo $i." ";
         $i--;
       }

   echo "\n\ndo-while loop:\n";

   $i=22;
   do
       {
         echo $i." ";
         $i-=2;
       }
   while ($i>30);
?>
</pre>

Udskriver:
For loop:
2-tabellen: 0 2 4 6 8 10 12 14 16 18

While loop:
42 41 40 39 38 37 36 35 34 33 32 31

do-while loop:
22
```

Loops kan afbrydes med **break** . **break** hopper straks til slutningen af loopet uden at udføre koden indtil da, til forskel så vil **continue** straks hoppe til starten af loopet. Begge kommandoer kan modtage et heltal

argument, der er antallet af loops der skal brydes ud af. Følgende er et uhyre kryptisk eksempel der bruger både break og continue:

```
<pre>
<?php
   echo "Continue og break:\n";

   $i=42;
   while ($i!=119)
      {
        do
          {
            $i-=17;

            if ($i == 52) { break 2;}
            if (5*$i % 13==0) break;
          }
        while ($i>0);

        do
          {
            $i+=19;
            if (7*$i % 13==0) break;
          }
        while ($i<101);

        if ($i % 112 == 0) continue;

        echo $i." ";
        $i++;
      }
?>
</pre>

Udskriver:
105 101 114 108 104 119
```

En tricky opgave i ovenstående eksempel er at finde ud af om koden afsluttes ved at hoppe ud af et break udtryk, eller fordi while udtrykket ikke længere er sandt?

4.8. Infokager (cookies)

Infokager er små stykker information, der gemmes hos brugeren, og kan hentes frem igen af netlæseren ved det næste besøg. De kan bruges til mange formål, eksemplet nedenunder viser hvordan man kan spare en bruger for at indtaste den samme information igen i en formular.

Programmeringsmæssigt skal man først indsamle informationen, via en FORM-formular, denne udfører en anden HTML-side, hvor man så kan se den indsamlede information, og gemme informationen hos

brugeren. Gemningen af informationen, med PHP-funktionen `setcookie()` skal ske før der overhovedet er skrevet andet ud, incl. den begyndende DTD og <html>-mærket. Informationen gemmes som regel i en tekstfil et administrationskatalog for netlæseren i brugerens hjemmekatalog, for eksempel `~/.netscape/cookies`.

Eksemplet indeholder to sider: `bestilling.php` og `bestilt.php`. `bestilling.php` er siden brugeren kommer til og indtaster oplysningerne, og `bestilt.php` er den side som kvitterer for indtastningen. De to filer udgør et komplet eksempel, der kører.

```
Siden "bestilling.php":
<html><head></head><body>
<?
  $navn=$HTTP_COOKIE_VARS["navn"];
  $epost=$HTTP_COOKIE_VARS["epost"];
?>
<FORM method="post" action="bestilt.php">
  Navn: <INPUT name="navn" value="<? echo $navn; ?>" >
  Epost: <INPUT name="epost" value="<? echo $epost; ?>" >
  <INPUT type="submit" value="Bestil">
</FORM> </body>
```

"bestilling.php" indhenter information om navn og epostadresse, via indtastning. Hvis vi tidligere har fået informationen og gemt den i infokager hos brugeren, så vil eksemplet udfylde indtastningen på forhånd med disse værdier. Dette sker ved at hente informationen via PHPs systemvariabel HTTP_COOKIE_VARS, som ved aktiveringen af siden allerede indeholder alle infokager for netstedet, og give det som værdier med VALUE tildelinger, og brug af PHP echo-udskrivning.

```
Siden "bestilt.php":
<?
  setcookie("navn",$navn,time()+3600*24*365,"/",".ditfirma.dk");
  setcookie("epost",$epost,time()+3600*24*365,"/",".ditfirma.dk");

echo "<html><head></head><body>Dit navn: " . $navn . "<br>Din epostadresse: " . $epost . "<
?>
```

`bestilt.php` aktiveres med tryk på "Bestil"-knappen fra `bestilling.php`. Den får med over alle variable fra formularen i `bestilling.php`, og det første vi gør er at sende infokagerne til brugeren med `setcookie()`.

Parametrene er: 1: navnet på infokagen, 2: værdien, 3: hvor længe infokagen gælder, i eksemplet et år (i sekunder), 4: en sti hos værten, som regel "`/`" (og der er netlæsere, der ikke forstår andet), 5: domænenavnet for værten, "`.`" betyder incl alle underdomæner. Dette er navnet på de web-servere der kan få fat på infokagen igen. Hvis du leger med eksemplet hér, så husk at ændre domænenavnet til dit eget domænenavn.

Dernæst udskriver vi et lille HTML-skript med de indtastede oplysninger. Bemærk "`.`" bruges til at sætte flere strenge sammen til én streng i PHP, derved undgås mange `echo` sætninger.

Kapitel 5. PHP eksempler

Har du af og til spekuleret over hvordan de forskellige websteder, du besøger, kan genkende dig, når du vender tilbage? Forklaringen er enkel, og når du har læst denne kapitel, ved du hvorfor og hvordan, du selv kan gøre noget tilsvarende.

For at benytte dette eksempel, kræves der, så vidt jeg ved, ikke PHP version 4.0, da vi netop udvikler vores egen sessionsstyring, og derfor kræver det altså ikke understøttelse af sessions, som er nyskabelsen med PHP 4.0.

Du skal nu til at begive dig ind i en spændende verden, hvor næsten alt kan lade sig gøre mht. dynamiske websider. Hvad der gør dette muligt, vil du få en introduktion til i dette kapitel.

Dette kapitel viser dig hvordan, du kan holde styr på dine besøgende, og på baggrund af gemte oplysninger, har mulighed for, at dirigere dine besøgende rundt på sider, der er af relevans for netop dem. Det der gør det muligt, er, at vi har oprettet en database med oplysninger om de besøgende. Oplysninger indeholder i vores eksempel kun de enkelte brugeres status - gæst, medlem, eller administrator, men det kunne sagtes udbygges til at have oplysninger om alt muligt andet; kun fantasien sætter grænser.

5.1. Sessioner i PHP med brug af MySQL

Hvorfor nu udvikle sin egen sessionsstyring? Til det spørgsmål findes der mange svar, men de mest åbenlyse for mig er følgende:

- Faciliteten sessioner er først medtaget i PHP 4.x, og der er stadig mange websteder, som kun tilbyder PHP 3.x.
- Sessioner i standardversionen kræver understøttelse af cookies hos klienterne. Mange har fravalgt muligheden for at modtage cookies
- Man kan godt nok skrive en databasehandler til PHP, så den ikke kræver understøttelse af cookies hos klienterne, men af interesse ville jeg lave min egen, uden at skulle begrænse mig til PHP 4.x. Eksemplet er også samtidigt en glimrende indgangsvinkel til PHP's verden.

For at du skal kunne afvikle eksemplerne i artiklen, skal du have følgende software installeret:

- En webserver - jeg bruger apache version 1.3.12
- PHP installeret på webserveren - jeg bruger PHP version 4.0.1.pl2
- MySQL installeret på serveren - jeg bruger MySQL version 3.23.10-alpha
- Operativsystemet på serveren skal være enten Linux eller Windows - jeg bruger Linux, men eksemplerne er også testet på en Windows NT 4.0 Workstation

5.2. Sessioner - hvad er det

Sessioner er en metode til at holde rede på ens besøgende, for på den måde at kunne vise sider, som den pågældende person *ønsker* det, eller som middel til at kunne bestemme, hvilke sider en person må se.

Hvordan fungerer sessioner:

- En person besøger din hjemmeside, og bliver registreret som bruger. Registreringen har som resultat, at personen får tildelt et entydigt BrugerID.
- Når brugeren kommer tilbage, og logger sig på, får vedkommende tildelt et entydigt SessionsID, som han medtager rundt på dine sider.
- Ved hjælp af SessionID kan du krydsreferere brugerens profil, for på den måde at finde ud af hvilke præferencer eller adgangsmuligheder denne person har.

5.3. En gennemgang trin for trin

Denne tutorial er inddelt i 4 afsnit. I første afsnit viser jeg, hvordan det tager sig ud for brugeren, samt kort introducerer dig for problemstillingen. Afsnittet indeholder en kort gennemgang af begrebet sessioner samt en overordnet gennemgang af den database, vores eksempel bygger på.

I andet snit gennemgås, hvordan du opretter databasen i MySQL. I afsnittet vil du få vist, hvordan du opretter databasen og de nødvendige tabeller. For en egentlig gennemgang af MySQL kan du med fordel læse i Afsnit 6.3.

I tredje afsnit vil jeg så gennemgå, hvordan tingene hænger sammen, og det er her, jeg vil vise dig, hvordan man får PHP til at arbejde i baggrunden. I afsnittet vil du kunne se kildeteksten til eksemplet, og jeg vil prøve efter bedste evne, at forklare hvordan det virker.

I fjerde og sidste afsnit vil jeg kort komme ind på, hvad der endnu mangler i eksemplet samt komme med anvisninger på, hvor du kan finde yderligere materiale om PHP.

For at du ikke skal skrive det hele ind selv, kan du her hente en fil med alle eksempel-filerne. Adressen hvor de ligger er Linuxbogens hjemmeside hos SSLUG: www.linuxbog.dk/ (http://www.linuxbog.dk/)

5.4. Problemstillingen, og hvad brugeren ser

Det følgende afsnit er en kort introduktion til begrebet databaser, og hvordan du kan drage nytte af dem, når du skal lave dynamiske websider. Afsnittet gennemgår begrebet sessioner, og illustrerer det med et konkret eksempel.

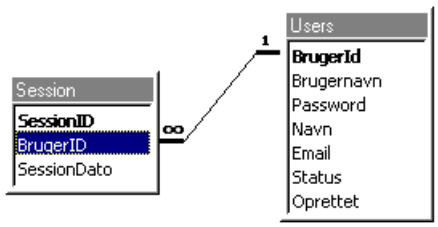
5.4.1. Databasen i vores gennemgående eksempel

Inden vi kan begynde, skal vi have oprettet vores database, og alle de tabeller vi skal bruge. Se figur 2-1.

Vores eksempel kræver to tabeller. En til at gemme brugerprofiler, og en anden til at gemme oplysninger om sessioner.

- Users: Brugernes profiler
- Session: Oplysninger om sessioner

Figur 5-1. Vores eksempeldatabase



Af vores eksempel kan vi se, at feltet Brugernavn i tabellen *Session* er en fremmednøgle til feltet Brugernavn i tabellen *Users* - det faglige ord er Foreign Key constraint (FK). At jeg har vist FK i den grafiske fremstilling, er en konsekvens af teorien om relationelle databaser, men da denne facilitet ikke er implementeret ordentligt i MySQL, og da konsortiet bagved også fraråder brugen af FK, benyttes FK ikke i mit eksempel. Hvis du f.eks. benytter Oracle i stedet for MySQL, vil jeg kraftigt opfordre dig til at benytte dig af FK, da de gør arbejdet med at vedligeholde en database væsentligt nemmere. Men til vores brug har det ikke den store betydning, udover at vi bare skal være klar over det og tage forbehold for det, når vi tilføjer eller opdaterer i databasen.

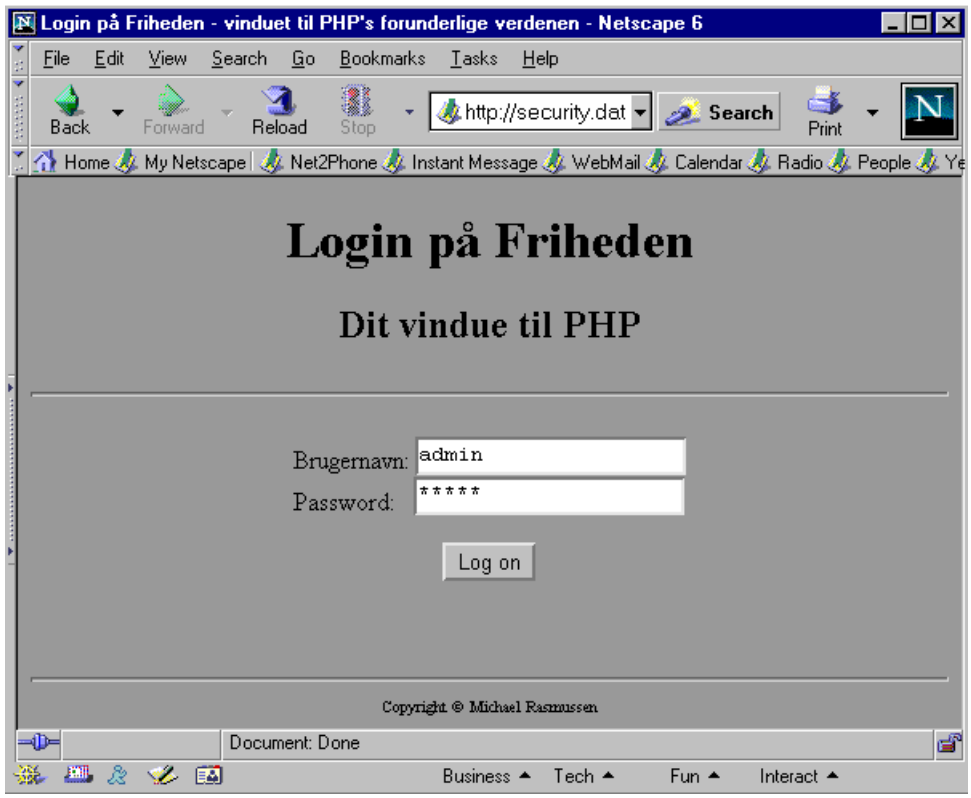
Dette eksempel er ikke særlig avanceret, da resultatet af brugerens indtastning kun er en side, hvor vedkommende bliver præsenteret for de oplysninger, vi har gemt om personen, men det har heller ikke været formålet med denne gennemgang, at vi skulle ende op med en stort eksempel, men mere har det været formålet at anskue hvilke muligheder, man har med kombinationen PHP og MySQL, og muligvis kunne være til inspiration for dig i din egen udvikling af dynamiske websider. Eksemplet illustrerer dog, hvilket kraftfuldt værktøj du har mellem hænderne, og kan let udbygges til en mere omfattende hjemmeside. Nu har du i hvertfald fået en skabelon, du kan bygge videre på.

5.4.2. Hvordan starter vi en session

Det første brugeren møder, når vedkommende kommer til vores websted, er en simpel login formular.

Her er, hvad brugeren ser.

Figur 5-2. Loginbilledet



Når brugeren har indtastet sit brugernavn og password, skal de indtastede oplysninger valideres, og alt efter resultatet af valideringen bliver brugeren præsenteret for et velkomstbillede eller en meddelelse om, at der er opstået en fejl.

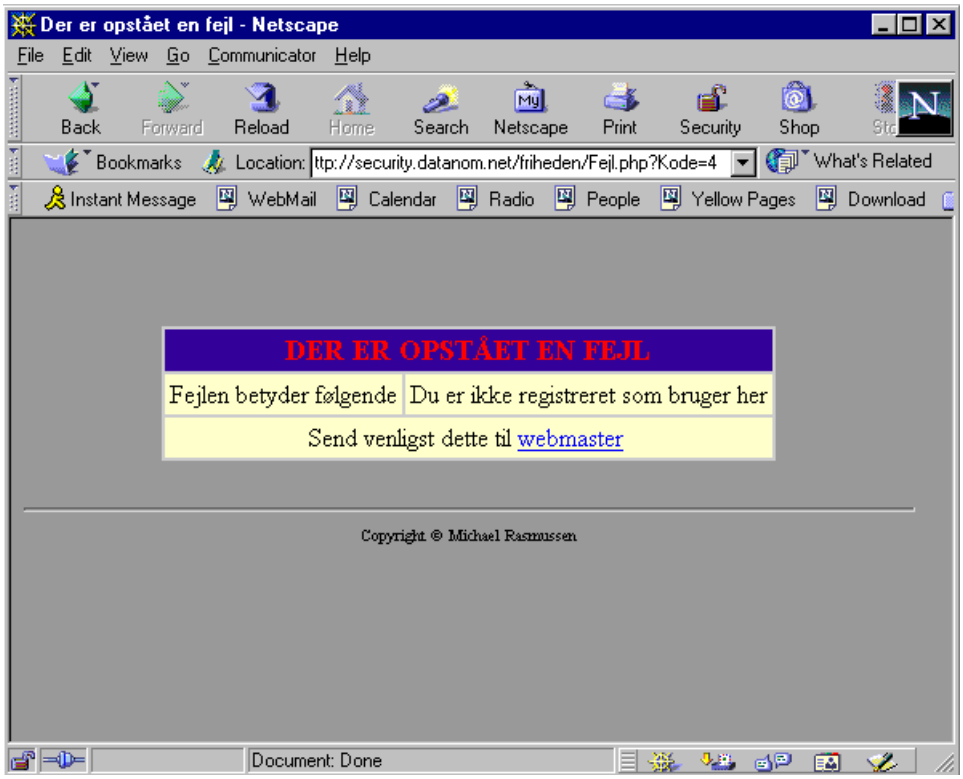
Hvis ellers det tekniske er i orden, vil der typisk være to udfald: Enten er den besøgende registreret, som bruger på vores system, eller også er vedkommende det ikke. Figur 2.3 viser det skærmbillede, der er resultatet af, at en registreret bruger har logget sig ind, mens figur 2.4 viser det skærmbillede, der er resultatet af at en bruger uden rettigheder, har forsøgt at logge sig ind.

Her er, hvad brugeren ser, hvis vedkommende godkendes.

Figur 5-3. Velkomstbilledet

Her er, hvad brugeren ser, hvis vedkommende ikke godkendes.

Figur 5-4. Fejlbilledet



5.5. Vores database oprettes i MySQL

5.5.1. Opret databasen og tildel en bruger rettigheder

Det første vi skal gøre, er at oprette vores database i MySQL, hvorefter vi skal have tildelt en bruger rettighed til, at indsætte, opdatere, slette og udtrække data fra databasen. For at gøre dette skal du logge dig ind på databaserveren og udføre følgende kommandoer:

```
[tyge@hven ~]$mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 254 to server version: 3.22.32

Type 'help' for help.
```

```
mysql>create database friheden;
Query OK, 1 row affected (0.00 sec)

mysql>use friheden;
Database changed
mysql>grant insert, delete, update, select on friheden.*
    -> to webuser@localhost identified by "WebVer1.00";
Query OK, 0 rows affected (0.12 sec)

mysql>
```

5.5.2. Opret tabellerne

Når vi nu har fået oprettet vores database, og har givet en bruger adgang hertil, mangler vi kun at få oprettet vores tabeller. Dette gøres med følgende kommandoer:

```
mysql>create table Users (
    -> Brugernavn varchar(50) primary key,
    -> Password varchar(50) not null,
    -> Navn varchar(50),
    -> Email varchar(30) unique not null,
    -> Status varchar(15) not null,
    -> Oprettet datetime);
Query OK, 0 rows affected (0.02 sec)

mysql>create table Session (
    -> SessionID char(32) primary key,
    -> Brugernavn varchar(50) not null,
    -> SessionDato datetime);
Query OK, 0 rows affected (0.01 sec)

mysql>
```

5.5.3. Indsæt de første brugere i databasen

Vi har nu oprettet databasen, og herefter mangler vi kun at indsætte en administrator og en almindelig bruger, som vi skal bruge som test. Kommandoerne kommer her:

```
mysql>insert into Users (Brugernavn, Password,
    -> Navn, Email, Status, Oprettet)
    -> values
    -> ("admin", "admin", "Michael Rasmussen",
    -> "admin@friheden.dk", "Administrator",
    -> "2001-02-06");
Query OK, 1 row affected (0.02 sec)
```

```
mysql>insert into Users (Brugernavn, Password,
    -> Navn, Email, Status, Oprettet)
    -> values
    -> ("sten", "sten", "Sten Larsen",
    -> "sten@larsen.dk", "Medlem",
    -> "2001-02-06");
Query OK, 1 row affected (0.01 sec)

mysql>
```

5.5.4. Kryptering

Vi har nu fået oprettet vores database, og fået lagt de første brugere ind i databasen. Der er dog en ting, du skal være opmærksom på her. I mine eksempler benytter jeg mig af funktionen *MD5*, som er en krypteringsfunktion, der er indbygget i PHP.

Nu vil du sikkert stille det spørgsmål: "Hvorfor det?". Forklaringen er ligetil: Uden kryptering er det “alt for let” for udenforstående at kompromitere dine sikkerhedsrutiner, og derfor har jeg medtaget faciliteten i mit eksempel. Send blot et indlæg til newslisten news.sslug.dk/sikkerhed, og du skal (måske) få svar på alle dine spørgsmål.

Det har den konsekvens, at hvis du har brugt mine programmer, vil alle de passwords, du har indtastet manuel, være "forkerte". Hvis du vil undgå denne fejl, skal du enten skrive alle PHP-programmer selv, eller fuldt ud benytte dig af mine - valget er dit.

En gennengang af hvordan du installerer mine programmer på din webserver, findes i afsnittet om Installation af eksemplerne.

5.6. Hvordan laver vi koden i PHP

I dette afsnit afslører jeg, hvordan vores eksempel kodes i PHP. Gennemgangen struktures på følgende måde:

Først viser jeg kildeteksten, hvorefter jeg vil gennemgå den. Jeg antager, at du er bekendt med HTML, men har ingen eller næsten ingen erfaring med PHP. Hvis du ikke har erfaring med HTML, vil jeg råde dig til, at hente denne viden et andet sted, inden du læser videre. Eksemplerne er dog ikke sværere end at man med et minimalt kendskab til HTML, vil kunne følge med i gennemgangen.

5.6.1. Login

Her har du koden til login billedet. Bemærk, at det er skrevet i rent HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
     "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
        <TITLE>Login på Friheden - vinduet til PHP's forunderlige verdenen</TITLE>
<META name="Generator" content="Stone's WebWriter 3">
</HEAD>
<BODY bgcolor="#999999" text="#000000" link="#0000FF" vlink="#660066"
  alink="#FF0000">
<CENTER>
<H1>Login på Friheden</H1>
<H2>Dit vindue til PHP</H2>
<HR>
<FORM action="login.php" method="POST" name="login">
Brugernavn: <INPUT type="text" name="user"><BR>
Password:   <INPUT type="password" name="password">
<P><INPUT type="submit" value="Log on" name="submit">
</FORM>
</CENTER>
<BR>
<P><HR>
<CENTER><FONT face="Times New Roman" size="-2">
Copyright © Michael Rasmussen</FONT></CENTER>
</BODY>
</HTML>
```

Brugeren indtaster sit brugernavn og password.

5.6.2. Brugervalidering

Her se koden for valideringen af brugerinput:

```
<?php
  require("Config.php");
  /*
  Denne fil indeholder definitionen på alle databasevariable. Variabelnavnene er skrevet me
  og første tegn begynder med "_". Henvisningen tilføjes i starten af alle filer, der tilgå
  */

  $link = @mysql_connect(_MYSQL_SERVER_NAME, _MYSQL_USER_NAME, _MYSQL_PASSWORD);
  if ($link == 0) {
     header("Location: Fejl.php?kode=1");
     exit;
  }
```

```
  mysql_selectdb(_MYSQL_DATABASE_NAME, $link);

  $password = md5($password);
  $SQLQuery = "SELECT Brugernavn FROM Users ";
  $SQLQuery .= "WHERE brugernavn='$user' and ";
  $SQLQuery .= "password='$password'";
  $result = mysql_query($SQLQuery);
  if ($result == 0) {
     header("Location: Fejl.php?kode=2");
     exit;
  }
  else{
       if (mysql_fetch_array($result)){
           $bruger_id = mysql_result($result,'Brugernavn');
           $session_id = md5(uniqid(rand()));
           $SQLQuery = "INSERT INTO Session (Brugernavn, SessionID, SessionDato) ";
           $SQLQuery .= "VALUES ('$bruger_id', '$session_id', '" . date("Y-m-d H:i:s") . "'
           $result = mysql_query($SQLQuery);
           if ($result == 0) {
                header("Location: Fejl.php?kode=3");
                exit;
           }
           mysql_close($link);
           header("Location: Velkommen.php?sessionid=$session_id");
           exit;
       }
  else header("Location: Fejl.php?kode=4");
  }
?>
```

Først skal vi erklære vores databasevariable. Dette foregår i filen *Configure.php* [1]. Herefter skal vi skabe en forbindelse til MySQL, og dette gøres med funktionen *mysql_connect(MySQL server, MySQL brugernavn, MySQL password)*. Hvis alt går vel, for vi returneret en handle til database serveren. For at finde ud af om alt gik godt, laver vi en test på den returnerede handle. Har den en værdi forskellig fra 0, er alt i den skønneste orden, ellers er der opstået en fejl. Til at håndtere fejlmeddelelser til brugerne har jeg skrevet en fil, som udskriver den rigtige fejlmeddelelse baseret på en medsendt kode. Dette har jeg gjort for at holde styr på alle fejlmeddelelser, og samtidig sikrer man sig også mod at udskrive en forskellig fejlmeddelelse hver gang. Efter vi har fået forbindelse til MySQL, forestår blot at bede den om at aktivere vores database. Dette gøres med funktionen *mysql_selectdb(MySQL database, databasehandle)* .

Vi har nu fået forbindelse til databasen, og kan herefter begynde at trække oplysninger ud fra den. Det første vi skal gøre, er, at få krypteret det indtastede password med *MD5*. Herefter skal vi blot bede om at få brugernavnet i tabellen Users, hvis password er lig med, det indtastede. Det gøres med funktionen *mysql_query(SQL forespørgsel)* [2]. Hvis det returnerede resultat er <> 0, er der opstået en fejl, i modsat fald får vi en handle til de(n) række(r), som opfylder vores betingelser - SQL forespørgslen. Da vi ved, at vi kun vil få returneret en række, behøver vi kun at læse indholdet af den første række. Til det formål benytter vi os af funktionen *mysql_fetch_array(handle)* [3]. Havde vi kunnet få flere rækker returneret,

måtte vi for at se alle rækkerne, benytte os af en løkkestruktur [4]. Funktionen returnerer den første række i vores handle, og de enkelte felter hentes med funktionen ***mysql_result(handle, feltnavn)***.

Det vi mangler, er blot at generere et SessionID og indsætte det i tabellen Sessions sammen med vores brugernavn. For at generere et SessionID skal vi bruge funktionen *uniqid(værdi)*. For at gøre det endnu mere tilfældigt bruger jeg funktionen *rand([min [, max]])*. Da jeg ikke har nogen krav til uddata, bruger jeg funktionen uden angivelse af min og max. Det returnerede tal skal nu konverteres til et SessionID, og derfor sender vi resultatet fra uniqid til funktionen MD5. At jeg bruger kryptering her, hænger sammen med, at antallet af mulige værdier `SessionID` kan have bliver forøget fra (9+31^10) til (32^37). Det er derfor meget sværere at gætte `SessionID`'s værdi, og hvis du udvider systemet med en funktion, som rydder op i `Session`-tabellen, så vil det være næsten umuligt at gætte en anden brugers `SessionID`.

Vi har nu et brugernavn og et SessionID, og vi kan derfor tildele det til vores ny besøgende, og indsætte vedkommende i tabellen over tildelte sessions. Herefter forestår kun at lukke vores forbindelse til MySQL, og sende brugeren videre til den første side sammen med personens tildelte SessionsID. Dette gøres med kommandoen *header("Location: Velkommen.php?sessionid=$session_id");*

5.6.3. Hvordan henter vi information på baggrund af et SessionID

For at hente oplysninger ud af databasen, baseret på et aktuelt SessionID, bruges følgende fremgangsmåde:

```
<?php
  include("session.php");

  if ($SessionID == "") {
       header("Location: login.html");
       exit;
  }
  $SQLQuery = "SELECT Brugernavn FROM Session WHERE SessionID = '$SessionID'";
  $result = mysql_query($SQLQuery);
  if ($result == 0) {
    header("Location: Fejl.php?kode=2");
    exit;
  }
  else {
    $field = mysql_fetch_array($result);
    $bruger_id = $field['Brugernavn'];
  };

  $SQLQuery = "SELECT Brugernavn, Navn, Email, Status, Oprettet from Users ";
  $SQLQuery .= "WHERE brugernavn='$bruger_id'";
  $result = mysql_query($SQLQuery);
  if ($result == 0) {
    header("Location: Fejl.php?kode=2");
    exit;
```

```
  }
  else {
    $field = mysql_fetch_row($result);
    $bruger = $field['Brugernavn'];
    $navn = $field['Navn'];
    $email = $field['Email'];
    $status = $field['Status'];
    $oprettet = $field['Oprettet'];
  }
?>
<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN \">
<HTML>
<HEAD>
        <TITLE>Websted Friheden</TITLE>
<META name=Generator content=Stone's WebWriter 3>
</HEAD>
<BODY bgcolor="#999999" text=#000000 link=#0000FF vlink=#660066 alink=#FF0000>
<BR><BR><BR><BR>
<CENTER>
<H2>Hvilke oplysninger har vi registreret om den pågældende person</H2>
<TABLE border=1 bgcolor=#CCFFFF>
<TH colspan="2" bgcolor="YELLOW">Følgende er registreret om <?php print "$bruger" ?></TH>
</TR>
<TR>
        <TD>Navn</TD>
        <TD><?php echo "$navn" ?></TD>
</TR>
<TR>
        <TD>Email adresse</TD>
        <TD><?php echo "$email" ?></TD>
</TR>
<TR>
        <TD>Status</TD>
        <TD><?php echo "$status" ?></TD>
</TR>
<TR>
        <TD>Oprettet</TD>
        <TD><?php echo "$oprettet" ?></TD>
</TR>
</TABLE>
<BR>
<BR>
<?php
if ($bruger == 'admin'){
    print
    "<BR>
    <A href='NyBruger.php?sessionid=$SessionID'>Opret ny bruger</A>";
}
?>
</CENTER>
<BR>
<P><HR>
<CENTER><FONT face="Times New Roman" size="-2">
```

```
Copyright © Michael Rasmussen</FONT></CENTER>
</BODY>
</HTML>
```

Vi er nu nået til det punkt, hvor vi skal til at se resultatet af vores anstrengelser. Det første vi selvfølgelig skal gøre, er, at kontrollere om vores bruger har logget sig ind. Har han ikke det, bliver han sendt til login siden. Denne kontrol foregår i filen *session.php*, og resultatet af kontrollen har to udfald: Vi har en godkendt bruger - 1) SessionID medsendt, 2) SessionID er valid, eller vi har en ikke godkendt bruger - 1) SessionID er ikke medsendt, 2) SessionID er ikke valid. Mere om denne fil senere. For nu antager vi blot, at vi har at gøre med en godkendt bruger.

I filen *session.php* fandt vi ud af, om brugeren var godkendt. Ved samme lejlighed hentede vi samtidig denne brugers brugernavn, og på baggrund af dette brugernavn har vi nu mulighed for, at hente de resterende oplysninger i tabellen Users. Opbygningen af forespørgslerne og udtrækningen af informationerne foregår på præcis samme måde, som beskrevet under Brugervalidering.

Her kommer den lovede gennemgang af filen *session.php*. Først kommer kilden:

```
<?php
  include("Config.php");

  $link = mysql_connect(_MYSQL_SERVER_NAME, _MYSQL_USER_NAME, _MYSQL_PASSWORD);
  if ($link == 0) {
     $SessionID = "";
  }
  else {
     mysql_selectdb(_MYSQL_DATABASE_NAME, $link);
     if (!isset($SessionID) or ($SessionID == "")) {
     $SessionID = "";
      }
     else {
        $SQLQuery = "SELECT Brugernavn from Session ";
        $SQLQuery .= "WHERE sessionid='$SessionID'";
        $result = mysql_query($SQLQuery);
        if ($result == 0) {
             $SessionID = "";
        }
        else {
              if (mysql_fetch_array($result)) {
                  $bruger_id = mysql_result($result,'Brugernavn');
              }
              else {
                   $SessionID = "";
              };
              mysql_free_result($result);
        }
     }
  }
?>
```

Først undersøges der, om vi kan etablere en forbindelse til databaseserveren. Kan vi ikke det, sættes SessionID = "" - hvis der ikke er forbindelse til databasen, vil vi af sikkerhedsgrunde, ikke tillade login til sikrede sider. Hvis vi kan få etableret en forbindelse, undersøges der, om brugeren har logget sig på - **if (!isset($SessionID) or ($SessionID == ""))**: Hvis SessionID ikke er aktiveret, eller hvis det ikke er tildelt nogen værdi, sættes SessionID = "". Hvis SessionID er tildelt en værdi, skal vi hente, det tilknyttede brugernavn. Hvis vi får en tom række tilbage, er det fordi, der ikke er blevet medsendt et valid SessionID, og derfor sættes SessionID = "". Inden vi afslutter, frigives vores handle.

5.7. Hvad mangler, og hvordan kommer vi videre med vores eksempel

Vores eksempel er nu færdigt, og nu henstår blot en perspektivering af vores eksempel. En ting er åbenbar: Vi mangler faciliteter til at administrere vores database. Administrationen kunne selvfølgelig foregå manuelt, men det bliver temmelig besværligt i længden, så min anbefaling ville være, at dette problem løses via en passende side, hvortil kun administratoren har adgang. Jeg har anvist, hvordan man kan løse problemet i mit eksempel, hvor man, hvis man har status af administrator, får en mulighed for at oprette nye brugere, når velkomstbilledet vises. I et konkret eksempel kunne man forestille sig, at når en administrator loggede sig på, ville han blive præsenteret for en særlig side, hvorfra han kunne gå til en administrationsside. Her skulle det være muligt at

- oprette og slette brugere
- Ændre de oplysninger vi har om brugerne
- Læse og udskrive statistik for vores websted
- .... Mulighederne er legio, kun din egen fantasi - og dine evner, sætter grænser

En anden ting jeg lige må påpege, er, at vores database vil vokse uafladeligt, da vi gemmer alle oplysninger om sessioner, så hvis du vil undgå dette fænomen, må du finde ud af hvordan, og hvornår, du vil slette sessionsoplysninger. Du kan eventuelt klare det med et cron-job, eller på anden hvis automatisere det. Har du ikke brug for historik, kan du lade webserveren klare det selv, ved at du sørger for at udarbejde et program, som sletter alle oplysninger i sessionstabellen, når brugeren forlader dit websted.

Hvis PHP har fanget din interesse, kan du med stor fordel finde yderligere informationer her:

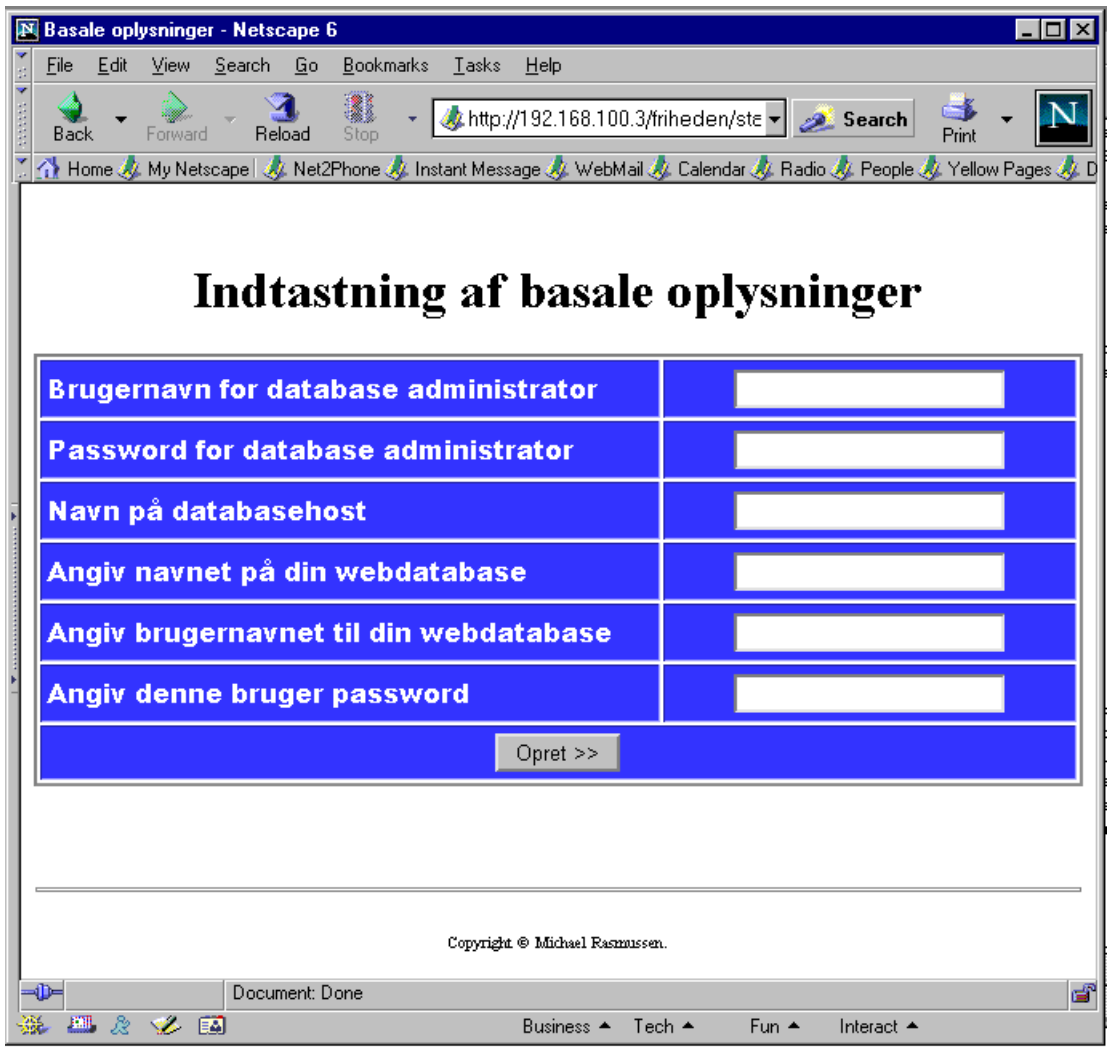
- http://webcafe.dk
- http://netcoders.dk
- http://weberdev.com
- http://phpbuilder.com
- Sidst men ikke mindst: http://php.net

5.8. Installation af mit eksempel

Den letteste måde at afprøve mine eksempler på er, hvis du benytter mine programmer i kataloget admin til at oprette databasen, tabellerne og configurationsfilen *Config.php*. Programmerne skal installeres i et katalog for sig selv under det katalog, hvor du har kopieret de andre programmer. Du skal huske at tildele skriverettigheder til kataloget, da du ellers vil få en fejlmeddelelse, når oprettelsesprogrammet prøver at skrive opsætningsfilen. En anden ting du skal huske på, er, at du efterfølgende sletter kataloget admin eller fjerne læserettighederne, da alle brugere ellers vil kunne aktivere installationsproceduren.

Når du har kopieret alle installationsprogrammerne til kataloget admin, skal du starte din browser og kalde filen *start.html* for at påbegynde installationen. Når filen start.html er blevet indlæst i din browser, vil du blive mødt af følgende billede:

Figur 5-5. Start.html



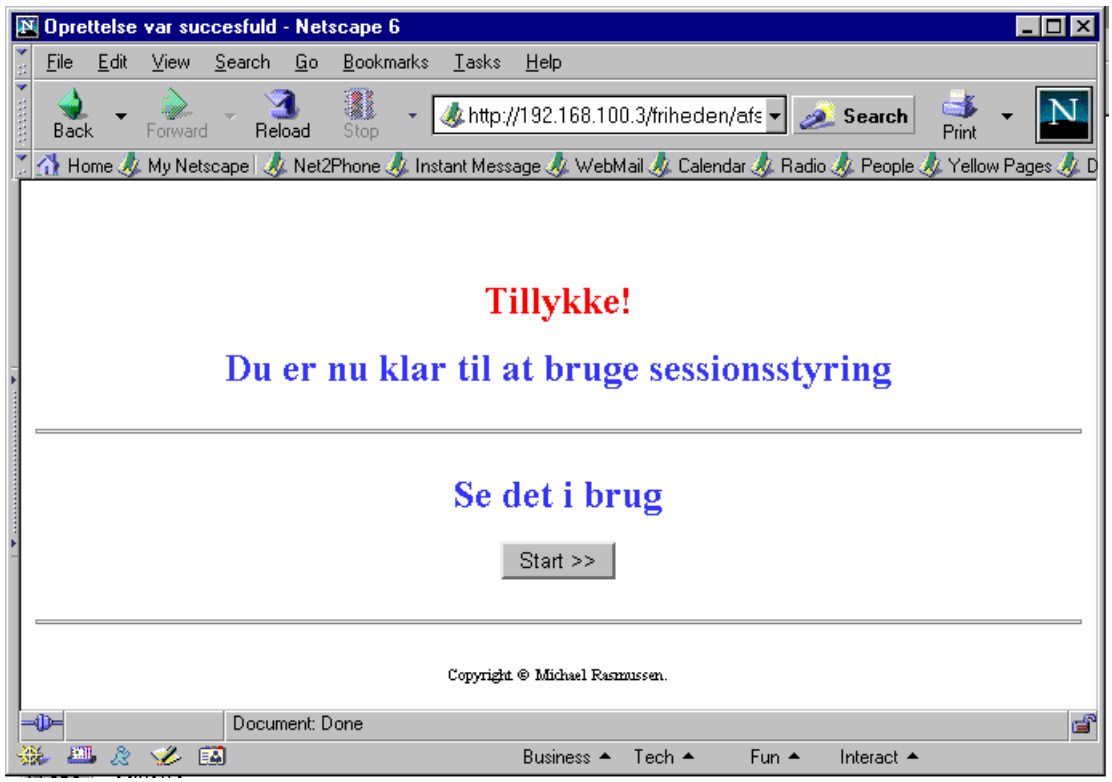
I billedet er der seks oplysninger, du skal indtaste. Disse er følgende:

- Brugernavn for database administrator: Den bruger, der har administratorrettigheder i MySQL. Default i MySQL er brugeren root.
- Password for database administrator: Passwordet for valgte bruger.
- Navn på database host: Hostnavnet for database serveren. Default i MySQL er localhost.
- Det ønskede navn for din webdatabase: I mit eksempel friheden.
- Brugernavn for brugeren af din webdatabase: I mit eksempel webuser.

- Passwordet for valgte brugernavn: I mit eksempel WebVer1.0.

Forløber alt fejlfrit, vil du se nedenstående billede:

Figur 5-6. Afslut.html



5.9. Afsluttende bemærkninger

Jeg håber, du har fået interesse for PHP gennem denne artikel, men inden du slipper den, mangler jeg lige at forklare forskellen mellem *include()* og *require()*. Forskellen er umiddelbart ikke stor, men i virkelighedens verden er der en væsentlig forskel. *require()* fungerer meget lig metoden #include <file> i C, og betyder blot, at nævnte fil skal indsættes på det kaldte sted. Indsættelsen foregår altid, og betingelsesløst. Med include() forholder det sig anderledes, da filen kun indsættes, når den pågældende linje udføres, og dermed har man mulighed for at includere forskellige filer alt efter programforløbet.

Hvis du ønsker at diskutere PHP med andre ligesindede, kommr her en liste med diskussionsgrupper. Som du ser, findes der ikke en diskussionsgruppe på news.sslug.dk, men her må du i stedet ty til gruppen prog, men hvem ved om ikke der snart kommer en for diskussioner omkring PHP? Du kan jo tilslutte dig gruppen af personer, som ønsker det, og når vi engang bliver lige så store som C-folket, kan det jo være SSLUG bøjer sig for kravet :-):

- news://alt.php
- news://dk.edb.internet.webdesign.serverside.php
- news://dk.worldonline.mysql-php

Slutbemærkning:

```
<?php
   //Database configs
   define("_MYSQL_SERVER_NAME", "localhost");
   define("_MYSQL_USER_NAME", "webuser");
   define("_MYSQL_PASSWORD", "WebVer1.00");
   define("_MYSQL_DATABASE_NAME", "friheden");
?>
```

2. Et glimrende opslagsværk, til den overkommelige pris af 69 kr, der kan købes i alle boghandler, er et lille hæfte fra IDG Forlag. (Rolland, 1999: Start på SQL)
3. Af uforklarlige årsager fungerer funktionen **mysql_fetch_row(handle)** ikke, som den skal, når man afvikler commandoen i PHP på en Windows-server - fejlen opstår, hvis man skal hente oplysninger fra flere felter i vores uddata, så derfor har jeg af kombatilitetshensyn valgt, konsekvent at benytte mig af funktionen **mysql_fetch_array(handle)**
4. F.eks. while (mysql_fetch_array(result))

Kapitel 6. Databaser

Linux er på grund af sin stabilitet et godt valg som databaseserver. Inden for databaser er de facto standarden sproget SQL, hvor data kan tilgås fra tekst-, grafisk-, og webbaserede løsninger samtidigt.

De to absolut mest brugte SQL-databaser på Linux er PostgreSQL og MySQL. Begge databaser udgives under en fri licens og er af høj teknisk kvalitet. For begge databaser gælder at du frit kan vælge de programmeringssprog og omgivelser du kender fra Linux. Perl, PHP, Python, C, Tcl/Tk, Java og foretrækker du shell-programmer, er det også en mulighed.

PostgreSQL er med her fordi den er meget avanceret og meget udbredt. PostgreSQL har SQL-funktionerne sub-select, stored procedures, transactions og ikke mindst brugerdefinerede objekter i databasen. MySQL er lidt hurtigere end PostgreSQL i mange situationer og er den mest udbredte af de to databaser.

De helt store kommercielle SQL-database producenter, Oracle, Sybase, Informix, IBM med DB2 og Inprise med InterBase er også kommet med en Linux-version af deres SQL-databaser. Programmeringsmulighederne er de samme, så det er nemt at flytte sit program den ene eller den anden vej, eller have et blandet miljø. Disse databaser på Linux anvendes i høj grad i produktionsmiljø, og i særdeleshed i udviklingsmiljø.

Anvender man Oracle og Linux sammen kan følgende artikler være meget interessante http://www.linuxjournal.com/article/5840 og http://www.linuxjournal.com/article/5841, idet de beskriver tuning af Linux så ydelsen nogle gange kan komme voldsomt i vejret.

Anvender du Microsoft Access databaser og overvejer at flytte dem til Linux, så er følgende artikel af interesse. http://www.nusphere.com/products/access2mysql.pdf.

6.1. SQL

SQL er en forkortelse for Structured Query Language og er et generelt sprog til at tilgå data i en SQL-database. De basale kommandoer i SQL er nemme at lære, så man kommer hurtigt igang med sin opgave. Har man kun een tabel at arbejde med, er SQL lige så nemt at bruge som andre konventionelle databaser, men når man ønsker at hente data fra flere tabeller samtidigt, så er der rigtigt mange fordele ved SQL. Har du prøvet at arbejde med ASCII filer der opdateres af flere personer samtidigt, så kender du til problemer som SQL-databaser løser for dig.

Et nemt forståeligt eksempel med to tabeller, eller 'relationelle tabeller' som det retteligt hedder, er en tabel med firmaer og en tabel med postnumre. For at sikre os at alle bynavne er stavet på samme måde, er postnummer og bynavn i en tabel for sig. I tabellen med firmaer er til gengæld kun angivet postnummeret og ikke bynavnet.

Skematisk kunne tabellerne se således ud

Tabel 6-1. Firma-tabel

FirmaNavn	Vej	PostNr
Vagabondos	Tagensvej 100	2200
DKUUG	Fruebjergvej 3	2100
Niels Bohr Institutet	Blegdamsvej 17-21	2100

Tabel 6-2. Postnumre-tabel

PostNr	ByNavn
2100	København Ø
2200	København N

Opgaven er nu at få en liste af alle firmaer med fuld adresse. Der er en relation i mellem de to tabeller ved feltet PostNr og SQL-kommandoen er så: **SELECT FirmaNavn, Vej, Firma.PostNr, Bynavn FROM Firma, Postnumre WHERE Firma.postnr=Postnumre.PostNr** Resultatet af ovenstående ser således ud:

Tabel 6-3. Søgeresultat: Firma+Postnumre

FirmaNavn	Vej	PostNr	ByNavn
Vagabondos	Tagensvej 100	2200	København N
DKUUG	Fruebjergvej 3	2100	København Ø
Niels Bohr Institutet	Blegdamsvej 17-21	2100	København Ø

Det der skete med SQL-kommandoen, var at der blev udført en 'join' imellem de to tabeller, således at postnumrene blev parret fra hver tabel.

Dette eksempel er selvfølgelig nemt at lave med ASCII tekst-filer, men prøv at forestil dig at der var tre eller flere tabeller.

Det vil være for vidt at komme ind på alle de SQL-kommandoer man kan lave i denne bog, da der findes mange gode bøger om emnet på dansk, og ikke mindst på engelsk. Er bøger uden for rækkevidde, er der god hjælp at hente på nettet. Se MySQL og PostgreSQL's hjælpesider.

Læsning af data er nok den mest brugte SQL-kommando, så den bliver lige beskrevet i korte træk. I sin simpleste form kan man skrive **SELECT * FROM Firma**, der giver alle felter fra tabellen 'Firma' i tilfældig rækkefølge. Er det kun 'FirmaNavn' og 'PostNr' man ønsker udskrevet, skrives **SELECT FirmaNavn,PostNr FROM Firma**. Data kommer i tilfældig rækkefølge, så der må lige sortering på **SELECT FirmaNavn,PostNr FROM Firma ORDER BY PostNr**.

Senere i bogen er nogle eksempler på hvordan man praktisk bruger SQL til noget der har med rigtige brugere at gøre. Til disse eksempler anvendes følgende liste af SQL-kommandoer som basis, og listen giver samtidigt de mest brugte SQL-kommandoer. Kommandoerne gør følgende:

- Opretter en tabel med to felter
- Tilføjer en post til tabellen
- Retter en post i tabellen
- Udskriver alle poster
- Sletter en post
- Sletter tabellen

```
CREATE TABLE linuxbog(kapitel INT, titel VARCHAR(40));
INSERT INTO linuxbog(kapitel, titel) VALUES(0, 'Indledning');
UPDATE linuxbog SET kapitel=1 WHERE kapitel=0;
SELECT * FROM linuxbog ORDER BY kapitel;
DELETE FROM linuxbog WHERE kapitel=1;
DROP TABLE linuxbog;
```

De viste SQL-kommandoer er helt basale og virker med alle SQL-databaser.

Læs mere om SQL-kommandoer på http://www.thomasheinen.de/courses/sql/A_SQL_Primer.htm, http://www.sqlcourse.com/ og
http://riki-lb1.vet.ohio-state.edu/mqlin/computec/tutorials/SQLTutorial.htm.

6.2. PostgreSQL

PostgreSQL er den mest avancerede frie SQL-database der findes til Linux. PostgreSQL startede som et studieprojekt, og har udviklet sig til fuldt professionel database med alle de avancerede teknologier som kræves idag: views, triggers, stored-procedures, transactions sub-select etc.

Installation af PostgreSQL fra en RPM-fil er den nemmeste måde at installere på. Efter installationen startes PostgreSQL.

```
[root@hven /root]# rpm -ivh postgresql-7.*.rpm
[root@hven /root]# rpm -ivh postgresql-server*.rpm
[root@hven /root]# /etc/rc.d/init.d/postgresql start
```

En nem måde at bruge PostgreSQL fra en webserver er ved at oprette en bruger med navnet `nobody`. Apache starter webserverne som brugeren `nobody` og så behøver man ikke angive adgangskoder i sine webprogrammer, idet PostgreSQL ser at man er `nobody` når man åbner en database.

Før `nobody` kan få adgang skal `nobody` oprettes som PostgreSQL bruger. Login som brugeren `postgres` og opret brugeren `nobody`. Kender du ikke `postgres` passwordet, kan det gøres således:

```
[root@hven /root]# su - postgres
```

```
[postgres@hven /pgsql]$ createuser nobody
```

`nobody` skal også have en database. Login som `nobody`. Ved ikke at angive databasenavnet nobody er det default navn når **createdb** køres. Kender du ikke `nobody` passwordet, kan det gøres således:

```
[root@hven /root]# su - nobody
[nobody@hven /]$ createdb
```

Databasen kan nu anvendes med de ganske få kommandoer SQL har. Der er få forskelle i de forskellige SQL-databaser der findes, men du lærer hurtigt de få forskelle der er, hvis du skifter over til en anden database. Her er brugt SQL-kommandoer som virker med alle de SQL-databaser som du vil komme i nærheden af. Følgende liste af kommandoer opretter en tabel, indsætter en post, retter den samme post, udlæser indholdet af tabellen, sletter en post og sletter tabellen.

```
-- fil: create.sql
-- PostgreSQL/LinuxBog
-- Kør dette program:
--  psql nobody < create.sql
CREATE TABLE linuxbog(kapitel INT, titel VARCHAR(40));
INSERT INTO linuxbog(kapitel, titel) VALUES(0, 'Indledning');
UPDATE linuxbog SET kapitel=1 WHERE kapitel=0;
SELECT * FROM linuxbog ORDER BY kapitel;
DELETE FROM linuxbog WHERE kapitel=1;
DROP TABLE linuxbog;
```

SQL-kommandoerne kan nemt indlæses fra en fil med kommandoen

```
[nobody@hven /]$ psql nobody < create.sql
```

Hvis du lige undlader at slette posterne (DELETE) og tabellen (DROP), kan eksemplet bruges sammen med de efterfølgende programmer i Shell, PHP, Perl og Python. Bemærk at hver linje er afsluttet med semi-kolon ';'. Semi-kolon bruges kun med programmet `psql` og ikke i andre fortolkede eller oversatte sprog.

Den interaktive SQL-fortolker, **psql**, kan anvendes til meget, men skal man have et rigtigt program, bør man bruge et mere generelt programmeringssprog end SQL og så snakke med databasen gennem et bibliotek. Linux har et bredt udvalg af sprog der kan kommunikere med de SQL-databaser der findes til Linux. Næsten alle sprog kan bruges sammen med alle databaserne. Vi har her valgt at bruge PostgreSQL som basis, og vise eksempler på hvordan man anvender disse sammen med:

- Kommandofortolkerprogrammering Afsnit 6.2.1
- PHP Afsnit 6.2.2
- Perl Afsnit 6.2.3
- Python Afsnit 6.2.4

I `/usr/doc/` har du hele PostgreSQL-manualen, typisk her: file:/usr/share/doc/postgresql-7.4.2/html/. Har du en anden version af PostgreSQL, så start her file:/usr/share/doc/ eller file:/usr/doc/. Du kan også

bruge kommandoen **locate app-postgres.html** til at lokalisere stedet, hvor html-manualen er installeret. Desuden ligger manualen også på nettet http://www.postgresql.org/docs/7.4/static/index.html.

PostgreSQL har sin hjemmeside på http://www.postgresql.org/.

PostgreSQL-HOWTO giver et godt overblik
http://www.linuxdoc.org/HOWTO/PostgreSQL-HOWTO.html.

6.2.1. Kommandofortolkerprogrammering + PostgreSQL

Kommandofortolkerprogrammer er ikke det mest avancerede, man kan lave, men er alligevel særdeles anvendeligt til at man hurtigt kan få lavet lidt database-tilgang alligevel. For at køre de følgende eksempler, kræves kun at PostgreSQL er installeret. **psql** har en option så man kan skrive SQL-kommandoer som parameter direkte fra kommandolinjen.

```
[nobody@hven /]# psql nobody -c "SELECT * FROM linuxbog"
```

En anden option giver output som HTML. Har du Apache installeret kan der nu laves et simpelt udtræk til web, ved at lægge et lille kommandofortolkerprogram i `cgi-bin`. Apache kan have dette sub-dir forskellige steder, så den nemmeste måde at finde det er med kommandoen **locate cgi-bin**. Placér programmet `pgsql.sh` i `cgi-bin` og gør det kørbart med kommandoen **chmod +x pgsql.sh**.

```
#!/bin/sh
echo Content-type: text/html
echo
echo "<HTML><TITLE>PostgreSQL CGI</TITLE>"
echo "<H1>PostgreSQL CGI</H1>"
psql nobody -H -c "SELECT * FROM linuxbog"
echo "</HTML>"
```

CGI-programmet kan nu startes fra din browser med URL'en http://localhost/cgi-bin/pgsql.sh. Til test kan programmet også køres fra kommandolinjen: **./pgsql.sh** hvorved output kommer ud på skærmen.

Ovenstående SQL-kommando er nok ikke lige det du vil, men nok mere noget i retning af:

```
psql nobody -H -T "cellspacing='0'" -c "SELECT kapitel AS \"Bog kapitel\", titel FROM linux
```

I ovenstående kommer der en pænere ramme uden om med option `-T "cellspacing='0'"`. Så er hver kolonne angivet, f.eks. `kapitel`. Typisk har hver kolonne et forkortet navn, hvilket ikke ser godt ud i en tabel. Ved at skrive `AS \"Bog kapitel\"` får kolonnen et andet navn. `ORDER BY kapitel DESC` sortere på 'kapitel' og DESC gør det i modsat orden. `LIMIT 10` gør at der kun kommer 10 rækker, og `OFFSET 0` fortæller at der skal startes med række 0.

Kommandofortolkerprogrammer er gode til små simple test, men man skal nok holde sig fra at lave større programmer. Muligheden er der, og det virker! I de næste afsnit vises det samme eksempel i PHP, Perl og Python, som er bedre til at håndtere de større programmer.

6.2.2. PHP + PostgreSQL

PHP er rigtigt godt sprog til fremstilling af webprogramme. Har du lidt erfaring med programmering i andre sprog, kommer du hurtigt igang med PHP og PostgreSQL.

PHP med PostgreSQL interface installeres nemmest fra RPM-filer. Mandrake 7.0 har modulerne færdigpakket, som kan hentes på http://rpmfind.net/linux/RPM/mandrake/7.0/Mandrake/RPMS/System_Environment_Daemons.html. PHP i en nyere version kan oftest findes hos Troels Arvin http://www.fsr.ku.dk/people/troels/rpms/php/. PostgreSQL har du installeret, så du skal kun hente følgende:

- apache-1.3.9-17mdk.rpm 2.6MB
- mod_php3-3.0.13-6mdk.rpm 850KB
- mod_php3-pgsql-3.0.13-6mdk.rpm 30KB

```
[root@hven /root]# rpm -ivh apache-1.3.9-17mdk.i586.rpm
[root@hven /root]# rpm -ivh mod_php3-3.0.13-6mdk.i586.rpm
[root@hven /root]# rpm -ivh mod_php3-pgsql-3.0.13-6mdk.i586.rpm
```

Hvis ikke Apache er startet nu, så gør det.

```
[root@hven /root]# /etc/rc.d/init.d/httpd start
```

Eksemplet `pgsql.php3` bruger den tabel der blev beskrevet i Afsnit 6.1 om SQL.

```
<html><head><TITLE>LinuxBog PostgreSQL PHP</TITLE></head>
<?  // Her skiftes der fra HTML til PHP-kode med tegnet '<?'
  // Åben en forbindelse til PostgreSQL
  // Ved ikke at angive brugernavn, er det 'nobody' der er brugeren
  $conn = pg_connect("");
  // Udfør en læsning fra tabellen 'linuxbog'
  $res = pg_exec($conn, "SELECT * FROM linuxbog ORDER BY kapitel");
  // For alle poster (linjer), udskriv indholdet
  for ($n = 0; $n < pg_numrows($res); $n++ ) {
    $post = pg_fetch_array($res, $n);
    print( $post["kapitel"]." ".$post["titel"]."<br>\n" );
  }
  // Frigiv hukommelse
  pg_freeresult($res);
  // Luk forbindelsen
  pg_close($conn);
?></html>
```

Resultatet af ovenstående ser således ud i en browser

```
1 Indledning
```

PHP er frit programmel og kan hentes på http://dk.php.net/.

6.2.3. Perl + PostgreSQL

Perl er effektivt programmeringssprog til mange ting. Perl anvendtes i starten ofte til behandling af tekstfiler, men efter de mange år Perl har været fremme, kan man arbejde direkte med databaser. Om det drejer sig om et kommandolinje-program eller et webprogram, så kan Perl klare den opgave.

Ved en normal Linux-installation får du automatisk installeret Perl. Der findes mange moduler til Perl, så du får ikke dem alle installeret. For at bruge PostgreSQL fra Perl, kræves at du installere DBI og DBD-Pg. Det forudsættes at PostgreSQL er installeret. Modulerne kan hentes fra http://sunsite.dk/CPAN/modules/by-category/07_Database_Interfaces. Dernæst finder du de to pakker her:

- /DBI/DBI-1.13.tar.gz 160KB
- /DBD/DBD-Pg-0.93.tar.gz 40KB

Installationen er meget nem og står beskrevet i begge pakker, men lad os lige tage den med her for en sikkerheds skyld.

```
[root@hven /root]# tar xzvf DBI-1.13.tar.gz
[root@hven /root]# cd DBI-1.13
[root@hven DBI-1.13]# perl Makefile.PL
[root@hven DBI-1.13]# make
[root@hven DBI-1.13]# make test
[root@hven DBI-1.13]# make install
```

Fremgangsmåden er næsten den samme for DBD-Pg-0.93.tar.gz.

```
[root@hven /root]# tar xzvf DBD-Pg-0.93.tar.gz
[root@hven /root]# cd DBD-Pg-0.93
[root@hven DBD-Pg-0.93]# export POSTGRES_include=/usr/include/pgsql
[root@hven DBD-Pg-0.93]# export POSTGRES_lib=/usr/lib/pgsql
[root@hven DBD-Pg-0.93]# perl Makefile.PL
[root@hven DBD-Pg-0.93]# make
[root@hven DBD-Pg-0.93]# make test
[root@hven DBD-Pg-0.93]# make install
```

Her er et lille eksempel i Perl der læser fra PostgreSQL. Eksemplet bruger den tabel der blev oprettet i Afsnit 6.2 om PostgreSQL. Tabellen linuxbog åbnes og alle poster udskrives.

```
#!/usr/bin/perl
# Inkluder DBI-modulet
use DBI;
```

```
# Åben en DatabaseHandle til PostgreSQL
$dbh = DBI->connect("DBI:Pg:dbname=nobody");
# Forbered en SQL-kommando for læsning af een tabel
$sth = $dbh->prepare("SELECT * FROM linuxbog ORDER BY kapitel");
# Udfør SQL-kommandoen
$sth->execute;
# For alle poster (linjer), udskriv indholdet
while (($kapitel,$titel) = $sth->fetchrow) {
  print "$kapitel $titel\n";
}
# Nedlæg StatementHandle
$sth->finish;
# Luk forbindelsen til PostgreSQL
$dbh->disconnect;
```

DBI og DBD har mange flere kommandoer som kan findes i online-manualen

```
[tyge@hven tyge]$ perldoc DBI
[tyge@hven tyge]$ perldoc DBI::FAQ
[tyge@hven tyge]$ perldoc DBD::Pg
```

Perl har hjemmesiden http://www.perl.com/ og http://www.perl.org/

6.2.4. Python + PostgreSQL

Python er et af de nyere fortolkede sprog der vinder kraftigt frem. Dette afsnit vil kun berøre Python's forbindelse til PostgreSQL. Du kan læse mere om Python i bogen *"Linux - friheden til at programmere"*.

For at bruge Python sammen med PostgreSQL skal du først have installeret selve Python, og dernæst et PostgreSQL-modul til Python. Python kommer med de fleste distributioner og måske allerede installeret på din maskine. PostgreSQL-modulet findes sammen med PostgreSQL, f.eks. her: ftp://ftp.sunsite.dk/disk1/www.postgresql.org/pub/binary/v7.0/redhat-RPM/RPMS/redhat-6.x/. Installationen er følgende:

```
[root@hven /root]# rpm -ivh python-1.5.1*.rpm
[root@hven /root]# rpm -ivh postgresql-python-7*.rpm
```

Her er et lille eksempel i Python der læser fra PostgreSQL. Eksemplet bruger den tabel der blev oprettet i Afsnit 6.2 om PostgreSQL. Tabellen linuxbog åbnes og alle poster udskrives.

```
#!/usr/bin/python
# Inkluder fra postgresql (pg) modulet
from pg import DB
# Åbne en forbindelse til databasen "nobody"
dbc = DB("nobody")
# Udfør SQL kommando
dbqo = dbc.query("SELECT * FROM linuxbog ORDER BY kapitel")
```

```
# For alle poster (linjer), udskriv kapitel og titel
for data in dbqo.dictresult():
    print str(data["kapitel"]) + " : " + data["titel"]
# Luk forbindelsen til databasen
dbc.close()
```

Gem filen som `pgsql.py` og gør den kørbar med kommandoen **chmod +x pgsql.py**. Og prøv så programmet.

```
[tyge@hven tyge]$ ./pgsql.py
```

Læs mere om Python hjemmesiden http://www.python.org/.

6.3. MySQL

MySQL er et stærkt SQL-database program, som folkene bag ikke tøver med at udråbe som "en af de hurtigste databaser, overhovedet"

Siden MySQL i sommeren 2000 ændrede sine licensbetingelser, sådan at alle fremtidige versioner af MySQL bliver udgivet under GPL-licensen, er MySQL efterhånden blevet et fast indslag i alle de store Linux-distributioner.

Hos MySQL anbefaler man dog, at Mandrake-(8.x) og RedHat(7.x)-brugere installerer MySQL ved hjælp af rpm-pakker hentet fra MySQL's websted, da distributions-pakkerne er kompileret med en version af gcc, der er kendt for at være problematisk i forhold til MySQL. Det kan derfor have ført til fejlbehæftede rpm-pakker, og det sikreste er derfor at hente pakkerne hos MySQL.

Hvis formålet med øvelsen er at "prøvekøre" en MySQL-installation, kan der dog næppe ske noget ved at installere pakkerne fra distributions-cd-rommen.. Hvis du vælger, at installere fra cd-rom'en - og har plads nok - så installér blot alle MySQL-pakkerne, så skulle du være kørende.

Hvis du vælger at hente de anbefalede pakker fra nettet, kan de hentes på adressen: www.mysql.com (http://www.mysql.com), eller fra sunsite.dk's danske spejling, sunsite.dk/mysql/ (http://sunsite.dk/mysql/) De nyeste versioner af MySQL er udviklingsversionerne, og bør derfor undgås. Den version, der er benævnt "stable release", eller "recommended" (i skrivende stund version 3.23.49) er den du ønsker at hente.

For at opnå en brugbar installation skal du som minimum hente og installere følgende pakker:

- MySQL-X.XX.XX.i386.rpm
- MySQL-client-X.XX.XX.i386.rpm

Når du nu er i gang, så vil følgende pakker sikkert gøre livet nemmere på lang sigt (de er for eksempel krævede i forbindelse med installationen af PHP med MySQL-understøttlse - læs herom senere)

- MySQL-devel-X.XX.XX.i386.rpm
- MySQL-shared-X.XX.XX.i386.rpm

Pakkerne installeres på følgende måde

```
[root@hven /downloads]# rpm -ivh MySQL-X.XX.XX.i386.rpm
[root@hven /downloads]# rpm -ivh MySQL-client-X.XX.XX.i386.rpm
[root@hven /downloads]# rpm -ivh MySQL-devel-x.XX.XX.i386.rpm
[root@hven /downloads]# rpm -ivh MySQL-shared-X.XX.XX.i386.rpm
```

Hvor X.XX.XX erstattes med versionsnummeret.

Nu er MySQL installeret, og der er blevet skabt de relevante opstartsfiler således, at MySQL startes automatisk under opstart af maskinen. For at tjekke om programmet allerede kører, kan du prøve at indtaste

```
[root@hven /root]# mysqladmin version
```

Hvis du får vist versionsnummer, uptime mm., kører MySQL allerede. Hvis MySQL ikke kører, kan du starte den med:

```
[root@hven /root]# /etc/rc.d/init.d/mysql start &
```

Nu er du parat til at gå i gang. Der er allerede under installationen oprettet en bruger, "root" (denne "root" har ikke noget med maskinens "root" at gøre - der er tale om MySQL's "superbruger") som har adgang til at foretage ændringer i databaserne. Denne bruger skal nu have et password:

```
[root@hven /root]# mysqladmin -u root password 'hemmeligt'
```

Lad os nu prøve at oprette en database, som vi kalder "bog". Først starter vi klientprogrammet:

```
[root@hven /root]# mysql -u root --password=hemmeligt
```

Nu er vi i forbindelse med MySQL og prompten skifter til en MySQL-prompt.

Her følger så lidt (My)SQL-gymnastik. Først skal vi oprette en database

```
mysql> CREATE DATABASE bog;
```

Dernæst vælger vi den nye base

```
mysql> USE bog
```

Hvorpå vi opretter en simpel tabel i basen

```
mysql> CREATE TABLE linuxbog(kapitel INT, titel varchar(40));
```

En database uden indhold er der ikke meget sjov ved, så vi fylder lidt data på tabellen

```
mysql> INSERT INTO linuxbog (kapitel,titel) VALUES (0,'Forord');
mysql> INSERT INTO linuxbog (kapitel,titel) VALUES (2,'Introduktion');
```

Der er ikke noget "kapitel 0", så vi retter vores indtastning til "kapitel 1"

```
mysql> UPDATE linuxbog SET kapitel=1 WHERE kapitel=0;
```

Nu vil vi gerne se, hvilke data, der rent faktisk er i vores tabel

```
mysql> SELECT * FROM linuxbog ORDER BY kapitel;
```

Og så sletter vi kapitel 1 (hvis du undlader at udføre de to sidste trin, kan databasen bruges sammen med PHP-eksemplet i Afsnit 6.3.1.

```
mysql> DELETE FROM linuxbog WHERE kapitel=1;
```

Og her sletter vi tabellen

```
mysql> DROP TABLE linuxbog;
```

Hvis du synes, at kommandolinje-klienten er lidt tung at danse med, så er du ikke den eneste. Heldigvis findes der flere alternativer. På MySQL's websted kan du under »download«-sektionen finde en lang liste over "contributed software", og på freshmeat.net (http://freshmeat.net) kan du ligeledes finde mange forskellige grafiske klientprogrammer. Et program, der ser meget lovende ud er MySQL's eget MyCC, som i skrivende stund kan hentes i en alpha-test-udgave fra MySQL's »download«-sektion. Har du mod på at installere apache webserveren og PHP (se nedenfor) kan du hente det glimrende webbaserede program, phpMyAdmin, som gør livet ikke så lidt nemmere for den nye bruger (også for den øvede - for den sags skyld). Pakken kan hentes i forskellige komprimerede formater hoswww.phpwizard.net/projects/phpMyAdmin/ (http://www.phpwizard.net/projects/phpMyAdmin/)

MySQL er mere end blot et database program; det er en database-server. Det betyder, at man - hvis man er oprettet som bruger og er blevet tildelt de passende rettigheder - ved hjælp af et klient-program kan koble sig på serveren via internet, lave opslag, intaste data, ændre data etc. Disse muligheder falder udenfor dette kapitels rammer, men der findes et par tutorials på bl.a http://www.devshed.com, der er værd at læse desangående.

6.3.1. PHP + MySQL

PHP er et fortolket sprog, der kan bruges til at skabe dynamiske websider. En af de helt store fordele ved PHP er evnen til at kommunikere med forskellige databaseprogrammer, og PHP "taler" usædvanlig godt med MySQL.

Før du kan bruge PHP, skal din webserver understøtte sproget. På et RedHat-system (7.x) opnås dette nemmest ved at installere følgende rpm-pakker fra distributions-cd-rommen:

- apache-1.3.xx
- php-4.0.x.i386.rpm
- php-mysql-4.0.x.i386.rpm

```
[root@hven /root]# rpm -ivh apache_1.3.XX.i386.rpm
[root@hven /root]# rpm -ivh php-4.0.x.i386.rpm
[root@hven /root]# rpm -ivh php-mysql-4.0.x.i386.rpm
```

Start Apache med kommandoen

```
[root@hven /root]# /etc/rc.d/init.d/httpd start
```

Nu kan det ske, at Apache fortæller, at den ikke kan "determine local host...". I så fald skal du åbne /etc/httpd/conf/httpd.conf i din favorit-editor og finde frem til linjen

```
#ServerName localhost
```

Fjern #-symbolet, gem og start Apache igen.

Her er et lille eksempel på, hvorledes Apache, PHP og MySQL kan arbejde sammen om at hente data fra en database i MySQL og præsentere dem på en webside. Databasen der bruges, er basen "bog", som blev oprettet i Afsnit 6.3.

```
<HTML>
 <HEAD>
  <TITLE>LinuxBog MySQL PHP</TITLE>
 </HEAD>
<BODY>
<?php  // Her skiftes der fra alm. HTML til PHP-kode

$conn = mysql_connect("localhost","root","hemmeligt");  // Åben en forbindelse til MySQL

$sql="SELECT * FROM linuxbog ORDER BY kapitel";  // Udfør en læsning fra tabellen linuxbog

$res = mysql_db_query("bog",$sql);

echo "<TABLE border=1><TR><TD colspan=2 bgcolor=#CCCCCC>";  //start en tabel
echo "<CENTER>Linuxbog</CENTER></TD></TR>";
```

```
  for ($n = 0; $n < mysql_numrows($res); $n++ ) {  // For alle poster (linjer), udskriv ind
                                                                                  // i en lille tabel
    $post = mysql_fetch_array($res);

    echo "<TR><TD>$post[kapitel]</TD><TD> $post[titel]</TD></TR>\n";
  }

echo "</TABLE>";  // afslut tabel

mysql_freeresult($res);  // Frigiv hukommelse

mysql_close($conn);  // Luk forbindelsen

?>

</BODY></HTML>
```

Kopier eventuelt dette eksempel, og indsæt det i et tomt dokument, som du gemmer som "test.php" i Apaches dokumentrod (/home/httpd/html/test.php). Kald filen igennem en webbrowser (http://localhost/test.php), og se hvad der sker.

Du kan læse meget mere om PHP på www.php.net (http://www.php.net), og du kan finde tutorials og programmer på www.devshed.com (http://www.devshed.com), www.hotwired.com (http://www.hotwired.com) og mange, mange andre steder på nettet.

Kapitel 7. Administration af web-indhold

Der er flere muligheder for at lave web-portaler med Linux, dvs. systemer som hjælper med at få en infrastruktur op omkring det indhold, man selv skal levere. phpNuke er en mulighed, Zope en anden og der er flere andre.

7.1. Zope

Zope er et Open Source *Content Management System*. Det bruges til - på en let og overskuelig måde - at opbygge en hierakisk organiseret hjemmeside.

Zope er bygget op over forskellige *Products*, som man kan importere i sin Zope, for derefter at lægge dem ind forskellige steder i hjemmesidens hieraki. Disse produkter varierer lige fra forskellige databaseforbindelser til nyhedslister, chats, gæstebøger og kalendere. Alt sammen nyttige ting til hjemmesiden.

I praksis kan man importere en pakke (et product) ved at placere dens filer under `~/zope/lib/python/Products` og genstarte Zope. Herefter optræder dens elementer i listen med objekter, der kan anvendes i foldersystemet.

Zope giver mulighed for at kontrollere adgangen til de enkelte dele af hjemmesiden. Har man fået adgang i et niveau, gælder denne for alle niveauer herunder, indtil der evt. sættes et nyt adgangs object ind. Fremgangsmåden er enkel, man placerer blot et acl_users object i den folder, man vil lukke med en adgangskode, og opretter de brugere, der skal have adgang. Forskellige andre objecter i samme folder eller underfoldere, kan derefter forholde sig til de rettigheder, der er sat op.

Man kan i sin adgang for eksempel give alle (anonyme brugere) adgang til at læse, og en eller nogle få adgang til at redigere.

Zope er skrevet i Python. Internt bruges markup-sproget DTML som man kan bruge til at manipulere dynamiske data med på forskellige måder. Man kan også vælge at lave sine egne python scripts, hvis man hellere vil dette. De implementeres lige så let som DTML dokumenter.

Zope har en indbygget webserver (ZServer) som man kan nøjes med at bruge til en mindre trafikeret hjemmeside. Der skal i det tilfælde blot rettes i filen ~/zope/z2.py, idet standardværdien for HTTP_PORT er 8080 (rettes til 80), mens den for FTP_PORT er 8021 (rettes til 21). Har du et startscript under `/etc/init.d/` skal du også kontrollere dette for portnumrene 8080 hhv. 8021. Skal Zope køre parallelt med Apache, skal der tilføjes lidt til httpd.conf, hvilket forholdsvis let kan lade sig gøre. Vil du vide mere om Zope, henvises til http://www.zope.org.

En af Zope's største styrker er produktet CMF (Content Management Framework). Dette kan bl.a. bruges til at bygge en portal. Med CMF er det nemt at styre brugere. CMF sørger for at holde indhold, data og kode adskilt. Har en indbygget skin-funktionalitet, som man kan bruge til at lave forskellige udseender til sin hjemmeside, lave den flersproget osv. Læs mere på http://cmf.zope.org

7.1.1. Installation af Zope

Her gives en beskrivelse af hvordan man kan få en fungerende (binær) Zope-2.6.1 til at anvende mysql-3.23-52 og mysql-4.0. Man skulle også kunne anvende rpm-pakker og debian-pakker, men hvis der opstår problemer, kan man falde tilbage på nedenstående og hente en binær pakke med zope fra http://www.zope.org:

Zope-mysql connections kræver to at man henter MySQL-python-0.9.2.tar.gz og ZMySQLDA-2.0.8.tar.gz (eller nyere stabile pakker) fra: http://sourceforge.net/project/showfiles.php?group_id=22307.

Læg de to tar pakker yderst i din zope-mappe (dér hvor du starter zope). Udpak pakkerne med: **tar xvzf PAKKENAVN**.

Nu lægger den udpakkede mappe MySQL-python-0.9.2 sig i zope-mappen, hvorimod ZMySQLDA lægger sig i `lib/python/Products/` under ZOPE-mappen og er sådan set klar til at blive anvendt inde fra zope - idet man kan tilføje en MySQL-connection. Dog kræver det først at python kan kommunikere med MySQL.

Til det formål skal man gå ind i mappen MySQL-python-0.9.2 og køre følgende kommando: **../bin/python setup.py build**.

Nu er der bygget de programmer som kræves. Det bemærkes at man anvender den python-udgave der følger med zopes binære programpakke. Dette er vist ikke mindst vigtigt, hvis ens python ikke er python2.1.3, men f.eks. python2.2, fordi zope bruger 2.1.3. Herudover er det afgørende at der med zopes python følger biblioteket distutils som skal bruges ved kompileringen. (Hvis man selv har kompileret zope, skal man angiveligt blot skrive: python setup.py build, men så må man have distutils-biblioteket i sin python.)

Man skal også sørge for at c-biblioteket mysql hvori mysql.h ligger, findes et sted hvor gcc-compileren kan finde den, f.eks. `/usr/include/mysql`. Biblioteket kommer med i mysql-4.0-standard-tgz-pakken som kan hentes fra http://www.mysql.com. Kopier det så til `/usr/include`.

Man skal også have en af mig ukendt delmængde af følgende biblioteker:

```
/usr/lib/libmysqlclient.a
/usr/lib/libmysqlclient.la
/usr/lib/libmysqlclient.so
```

```
/usr/lib/libmysqlclient.so.10
/usr/lib/libmysqlclient.so.10.0.0
/usr/lib/libmysqlclient_r.so
/usr/lib/libmysqlclient_r.so.10
/usr/lib/libmysqlclient_r.so.10.0.0
```

Mangler de relevante biblioteker, går oversættelsen i stå med beskeden:

```
/usr/bin/ld: cannot find mysqlclient_r
```

Det kan også være at oversættelsen går i stå med:

```
/usr/bin/ld: cannot find -lz
```

I så fald går du ind på: http://www.gzip.org/zlib/ og henter `zlib` som kildetekst. (Tager du f.eks. en rpm-pakke, risikerer du uendelige afhængighedsproblemer).

Du udpakker zlib med **tar xvzf PAKKENAVN**. (hvis du har hentet tar.gz-filen) og går ind i den resulterende zlib-mappe hvor du foretager:

```
./configure
make test
make install
```

Når du er færdig med at installere relevante biblioteker, bør du formentlig køre: **/sbin/ldconfig** så bibliotekerne bliver fundet under oversættelsen.

Læs iøvrigt README-filen.

Nu mangler man blot at installere MySQL-python. Det gøres med følgende kommando, inde fra mappen MySQL-python-0.9.2:

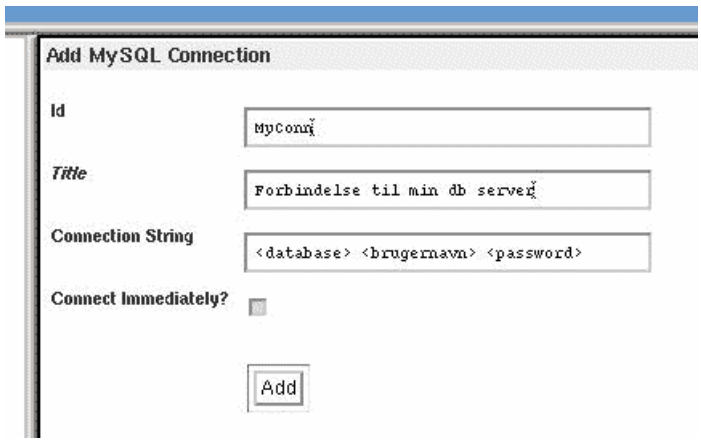
```
../bin/python setup.py install
```

Du skulle nu kunne starte zope som du plejer og adde en ZMySQL-Connection inde i zope.

Når du inde i zope har valgt en mysql connection (vi forudsætter at mysqld kører), skriver du i feltet "Database Connection String", f.eks.:

```
<databasenavn> <bruger> <password>
```

Figur 7-1. Tilføj MySQL forbindeler



Så kommer zope sikkert ud og siger: "`Can't connect to local MySQL server through socket ....`" I så fald går du ud i en bash-terminal og skriver:

```
mysqladmin -u root -p variables | grep socket
```

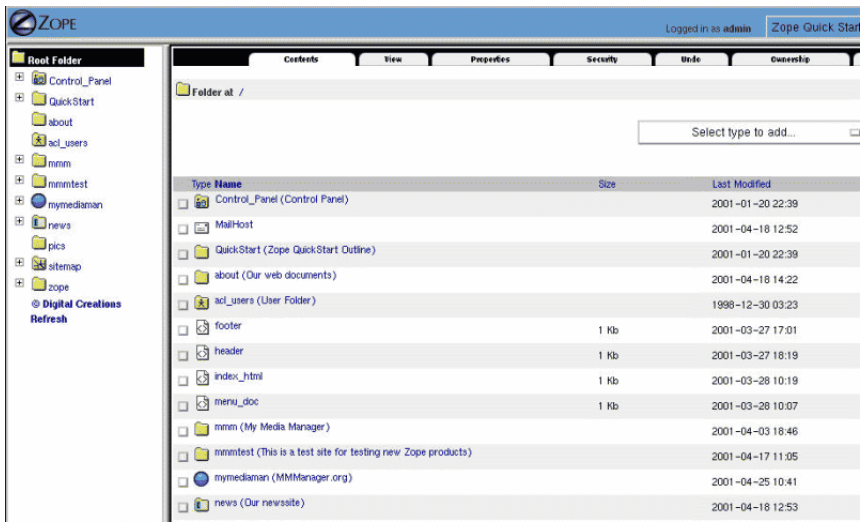
har du ikke sti til mysqladmin, så tilføj fuld sti foran den kommando.

Nu kan du skrive den rigtige "Database Connection String":

```
<databasenavn> <bruger> <password> <sti til socket>
```

Derefter går du formentlig fuldstændig i ekstase over at din mysql-forbindelse virker. Du kan i stedet flytte de to tar-pakker fra zope-mappen - og sikkert også mappen MySQL-python-0.9.2 - væk fra zope-mappen.

Figur 7-2. Zope i aktion



7.1.2. Zope koblet til MySQL

Lad os nu se på hvordan Zope kobles til en database i MySQL.

7.1.2.1. Få Zope til at bruge MySQL i stedet for Zope's egen Z Gadfly database

Zope har sin egen indbyggede database (Z Gadfly), og er dit behov for datalagring ikke særlig stort, kan du sagtens klare dig med denne. Dels har den dog en øvre begrænsning på et par gigabyte (det kan de fleste nøjes med), og dels kan man ikke komme i kontakt med den fra andre programmer. Eftersom der gemmes en del i denne database i forvejen (bl.a. brugerdata) er dette nok ikke den optimale løsning hvis du skal lave et stort websted med mange brugere, eller med databaseindhold, som kommer fra helt andre kilder end Zope selv.

Der er lavet forskellige adaptorer til de mest brugte databaser.

7.1.2.2. Vise MySQL data i Zope

Sådan henter du data fra f.eks. en MySQL database og viser dem. Dette gøres ret enkelt i Zope. Gå ned i din rodfolder og tilføj et "Z MySQL Database Connection"-objekt. Udfyld formen med de korrekte data og tilføj den. Tilføj nu en ZSQL Method. "Arguments"-feltet udfylder du med udvælgelseskriterier til

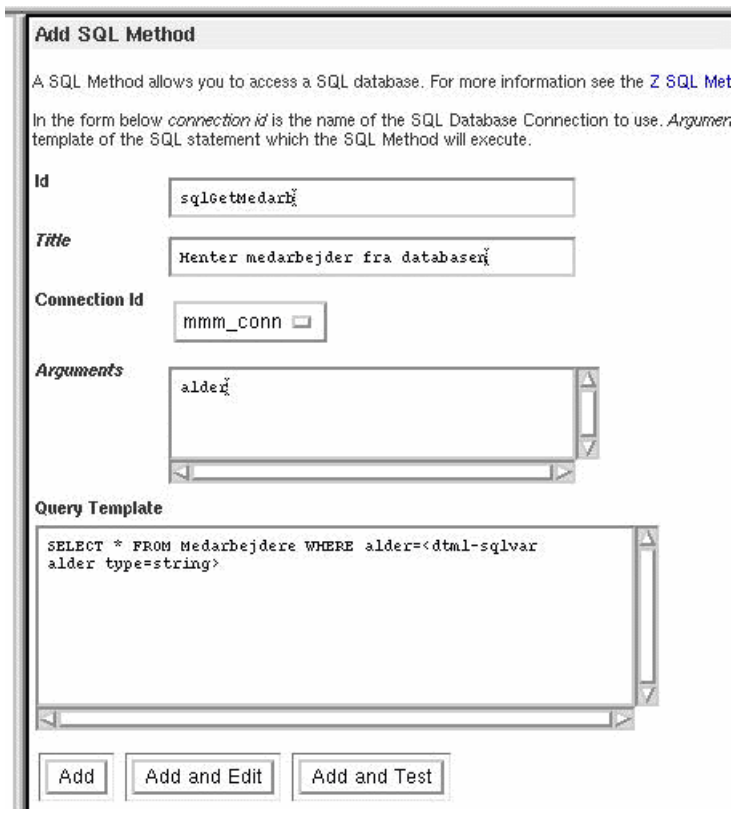
query-strengen. Hvis du nu har en tabel i din database der hedder Medarbejdere. Den har følgende kolonner: Navn, Afdeling, Alder. Du vil gerne liste alle medarbejdere der er født i 1980. Fra det DTML-Document der kalder din ZSQL Method har du adgang til variablen alder. Derfor kan du i din "Argument"-liste skrive alder. Din query-streng skal så se ud som følger: `SELECT * FROM Medarbejdere WHERE alder=<dtml-sqlvar alder type=string>`.

Fra dit DTML Document kalder du din ZSQL Method på følgende måde:

```
<dtml-in expr="MinSQLMethod(alder=alder)">
        Navn: <dtml-var Navn><BR>
        Afdeling: <dtml-var Afdeling><BR><BR>
</dtml-in>
```

Forklaring: <dtml-in> er en nem måde at iterere gennem f.eks. SQL resultater. Den fungerer på samme måde som f.eks. en for-løkke i PHP. For at være sikker på at alder optræder i REQUEST-objektet, kalder man sin ZSQL-Method med alder-variablen. <dtml-var (navn)>-tag'erne indsætter de variabler, der returneres fra ZSQL-metoden.

Figur 7-3. Tilgang til MySQL



7.1.3. Andre Zope-produkter

Som tidligere nævnt findes der mange forskellige produkter til Zope, som er udviklet af andre Zope-brugere. Det smarte ved at bruge disse produkter er at i stedet for selv at lave f.eks. en chat, kan man hente en fra http://www.zope.org/Products.

Det er ret nemt at installere disse produkter. Der findes to måder de fleste produkter installeres på. (Andre måder kan forekomme - læs derfor altid installations vejledningen). Filer er som regel pakket som tar.gz, eller også har den en `.zexp` extension. I det første tilfælde skal filen udpakkes så det udpakkede dir ender med at ligge i `/usr/local/Zope-xx/lib/python/Products/`. Herefter genstartes Zope og dit nye produkt er klar til brug. Ender filen derimod på `.zexp` skal filen lægges i `/usr/local/Zope-xx/import`. Du skal så gå ind i Zope's administration interface i Control Panel - Products og klikke på "import/export"-knappen. En ny side kommer nu frem. I "import"-tekstfeltet skriver du navnet på den fil du netop har placeret i import-dir'et og klikker på "import"-knappen. Produktet vil nu være synligt i listen over produkter (Control Panel - Products).

Du bruger produkterne ved at tilføje dem i de relevante mapper i dit administrationsinterface.

7.1.4. CMF

CMF er et meget kraftfuldt værktøj som man kan udvide sin Zope med. CMF står for Content Mangement Framework og bruges til at opbygge - på en meget nem måde - portaler med. CMF benytter sig også af Zope's rolle hieraki. Derudover bliver man introduceret for "skins". Zope.org har dedikeret et helt site til deres portal-produkt. Adressen er http://cmf.zope.org. Det er her du how-to's, nye portal-produkter, nyheder og hvad der ellers er værd at vide om CMF.

7.1.4.1. Sådan installeres CMF

Gå ind på Zope's CMF-site. På den første side er der et link til deres »download«-side. Hent den af versionerne der hedder 1.0 og ikke er alpha eller beta. Din tarball skal nu udpakkes i `<Zopedir>/lib/python/Products`. Du får lavet en folder der hedder CMF1.0. Gå ned i det og flyt de tre biblioteker der er dernede et niveau op. Der skulle nu i din Products folder ligge 3 nye foldere: CMFCore, CMDDecor, CMFDefault. Genstart din Zope og CMF er klar til brug !

7.1.4.2. Kom godt igang med CMF

For at kunne lave sin portal, skal man gå ind i Zope's administrationsinterface. I den folder du gerne vil placere din portal (som regel rod-folderen) vælger du at tilføje et CMF Site. Du kalder din portal noget passende og portalen er nu klar til brug. (Et passende navn vil være indec.html, idet de fleste browsere søger efter en fil med det navn som standard startsiden).

For at komme igang med at bruge din portal, skal du klikke på "View"-tabben. Der er et par ting der skal konfigures, før din portal er klar til offentligheden. Du kan enten vælge at bruge portalen egen opsætningsside, eller du kan konfigurere den via administrationssiden. Vælger du det sidste skal du gå tlbage til `http://www.mitdomæme.dk:8080/manage` og vælge din portal. (portnummeret :8080 er standarden, men har du ændret det til port 80, kan du helt udelade :8080 i foregående link). Du skal herefter klikke på "Properties"-tabben. Indtast eller udskift de eksisterende værdier og klik på "Save". Nu er portalen klar til at se dagens lys :-)

Der er masser af måder du kan ændre designet af din portal på. Men inden du begynder, er det måske en meget god idé at kigge lidt på hvordan portalen hænger sammen. Der findes som standard tre slags brugere; Anonymous, Member og Reviewer. Man er Anonymous indtil man bliver medlem. Når man så er medlem, kan man bidrage med forskellige ting til portalen. Prøv at opret en bruger ved at klikke på "Join" og log ind. Du vil nu få nogle flere valgmuligheder i den venstre menu. Hvis du klikker på "My Stuff" kommer indholdet af din folder frem i vinduet. Du kan nu vælge enten at editere filen "index_html" (som er den side andre kan se hvis din profil er listet) eller du kan vælge at tilføje mere til folderen. Klik på "Add" og prøv at tilføje en nyhed. Først udfylder du objektet's metadata og derefter vælger du at editere teksten. Når du har gjort dette, vil du selvfølgelig gerne have at andre skal kunne læse din nyhed, så du klikker på "Publish". Din nyhed er dog ikke helt publiceret endnu. For at den kan blive dette, er der en bruger med rollen "Reviewer" eller "Manager" der skal godkende din nyhed. Du kan nu vælge enten at oprette en anden bruger, du kan tildele rollen "Reviewer" eller du kan vælge at logge ind med din administrator bruger (som jo har rollen "Manager"). Vælger du det første skal du logge ud, klikke endnu engang på "Join", oprette brugeren og gå ind i administrationsinterfacet (`http://www.mitdomæne.dk:8080/minportal/manage`). Her skal du klikke på den folder dine medlemmer ligger i (acl_users) og klikke på den bruger du gerne vil tildele rollen "Reviewer". Marker "Reviewer" i select-boxen og klik på "Save". Gå nu tilbage til din portal og log ind som denne review-bruger. Du vil nu se nederst i den venstre menu at der står "Pending (1)". Klik her. Læs og godkend nyheden så vil denne blive publiceret. Gå tilbage til portalens forside og kig i højre side i nyheds-kassen. Nyheden optræder nu her. På samme måde som du lige tilføjede en nyhed, kan alle dine portal-brugere tilføje dokumenter, nyheder, billeder, links etc. etc. Den eneste der begrænser dette, er portal-administratoren.

7.1.4.3. Lav portalen som du selv vil have den

Det er nu på tide at kigge på de lidt mere konfigurerbare ting af portalen, der ikke kræver den store programmør. Portalens udseende bliver styret fra den folder der hedder "portal_skins". Gå tilbage til administrations-interfacet og klik på "+" på skins-folderen så den folder sig ud. Der ligger en række foldere, hvoraf en er en "almindelig" folder. De andre er det der kaldes for "FileSystem" foldere. Indholdet i disse kan ikke ændres i skins-folderen, men kan kopieres til den folder der hedder "Custom" og ændres der. Hvis du klikker på folderens "Properties"-tab, kan du se hvordan de forskellige skins er defineret. Ud for hvert enkelt skin, står navnene på de foldere der bliver inkluderet i skin'et. Folderne bliver gennemsøgt fra venstre mod højre. D.v.s. at hvis man kopiere en fil fra f.eks. "generic"-folderen til "custom"-folderen, bliver den kopi af filen der ligger i "custom" brugt før den der ligger i "generic". Hvis man nu byttede om på rækkefølgen af "custom" og "generic" ville den tilrettede kopi af filen ikke blive brugt. På denne måde kan man definere forskellige former for skins (med forskellige stylesheets f.eks.) til sin portal og dine brugere har nu forskellige udseender at vælge imellem. Du kan også bruge skins til at definere forskellige sprog med. En anden god ting man kan lave er et "Printable" skin som gør det

muligt at vise en print-venlig side af alle dine dokumenter. Du laver bare en folder der hedder "Printable" i "portal_skins" folderen, kopierer de mest nødvendige filer ned i (såsom "index_html" etc.). Herefter laver du på "portal_skin"-folderens properties side et nyt skin der hedder "Printable" og tilføjer de skin-foldere der er nødvendige (som oftes skal de samme foldere inkluderes som også er i de andre skins på nær dem der sætter farver ind, laver topbaren, sidebarerne etc.). At tilrette de eksisterende skins og lave nye er en hel videnskab for sig, men jo mere man arbejder med det, jo bedre forståelse får man for hvordan det hænger sammen.

Der er efterhånden blevet udviklet flere og flere portal-typer end dem din portal blev født med. Disse finder du på http://cmf.zope.org. Det er ikke helt let at tilføje nye portal-typer til portalen, da der er mange forskellige måder man kan gøre det på. Men som regel har udvikleren af portal-typen lavet en installationsvejledning og så er det bare med at følge den.

Løber du ind i problemer du ikke umiddelbart kan løse selv, er der hjælp at hente. Du kan tilmelde dig mailinglisten Zope-CMF. Klik ind på http://www.zope.org/Resources/MailingLists og tilmeld dig denne liste. Så er du i hvert fald godt på vej !

7.2. PHP-Nuke - start en portal på få timer

Hvis man vil have en portal startet på få timer og ikke regner med en ektrem belastning på maskinen, så et PHP-Nuke en glimrende måde at komme i gang på. Langt det meste vil konfigureres fra en web-browser, og man får typisk en fin site med en beskeden indsats. Ideen med PHP-Nuke er at der anvendes en SQL-database til at gemme indhold mens integrationen mellem Apache-webserveren sker med en række PHP-scripts, der ikke ændres. Systemet er lavet så man ikke skal rette i filer for at håndtere den daglige drift.

Eneste negative ting angår sikkerheden. PHP-Nukes programmører er ikke gode til at højne sikkerheden - hvilket kan ses på http://www.securityfocus.com hvis der søges på PHP-Nuke. F.eks. er der problemer med PHP-Nuke-filerne hvis man har en masse brugere på samme maskine, idet alle kan læse og skrive de filer der danner PHP-Nuke. Er det kun system-administratoren som anvender maskinen kan man se bort fra dette.

Start med at hente PHP-Nuke fra http://www.phpnuke.org. Vælg »Downloads« i øverste menu og "PHP-Nuke". Dette burde lede til http://phpnuke.org/modules.php?name=Downloads&d_op=getit&lid=225 (http://phpnuke.org/modules.php?name=Downloads&d_op=getit&id=225).

Start med at installere en apache-webserver og en MySQL-database. På en Red Hat 7.2 skal man indsætte Red Hat-cd-rom nummer 1 og skrive

```
[tyge@hven ~]$ su
[root@hven /root]# mount /mnt/cdrom
```

```
[root@hven /root]# rpm -ivh /mnt/cdrom/RedHat/RPMS/apache-1.3.20-16.i386.rpm
[root@hven /root]# rpm -ivh /mnt/cdrom/RedHat/RPMS/php-4.0.6-7.i386.rpm
[root@hven /root]# rpm -ivh /mnt/cdrom/RedHat/RPMS/php-mysql-4.0.6-7.i386.rpm
[root@hven /root]# umount /mnt/cdrom
```

Indsæt derefter cd-rom nummer to (stadig som brugeren root)

```
[root@hven /root]# mount /mnt/cdrom
[root@hven /root]# rpm -ivh mysql-3.23.41-1.i386.rpm mysql-server-3.23.41-1.i386.rpm
[root@hven /root]# umount /mnt/cdrom
[root@hven /root]# exit
```

Nu skal Apache-opsætningen sættes klar til en web-server. I det følgende antages at du vil have web-serveren http://www.linux-portal.dk i luften på IP-adresse 192.168.1.3 (denne IP-adresse virker kun på private netværk). Sørg for at dit servernavn www.linux-portal.dk er kendt i DNS (navneserveren) eller i det mindste nævnt i `/etc/hosts` hvis serveren kun skal servicere din hjemmenetværk.

Ret med en editor filen `/etc/httpd/conf/httpd.conf` og find afsnittet "Section 3: Virtual Hosts" omkring linje 1173. I linjen "NameVirtualHost" ændres IP-adressen og nedenfor indsættes en virtuel maskine som vist nedenfor - vi har her valgt at PHP-Nuke skal installeres i `/usr/local/PHPNuke`. Erstat gerne root@localhost med din egen email-adresse (anvendes til at rapportere fejl).

```
NameVirtualHost 192.168.1.3

<VirtualHost 192.168.1.3>
    ServerAdmin root@localhost
   DocumentRoot /usr/local/PHPNuke/html
  <Directory "/usr/local/PHPNuke/html">
    Options Indexes FollowSymLinks Includes
    AllowOverride None
    Order allow,deny
    Allow from all
    XBitHack full
  </Directory>
    ServerName www.linux-portal.dk
    ErrorLog logs/Nuke-error_log
    CustomLog logs/Nuke-access_log common
</VirtualHost>
```

Vi skal også huske at starte både Apache webserveren og MySQL serveren op.

```
[root@hven /root]# /etc/init.d/httpd restart
[root@hven /root]# /etc/init.d/mysqld restart
```

Pak nu PHP-Nuke ud

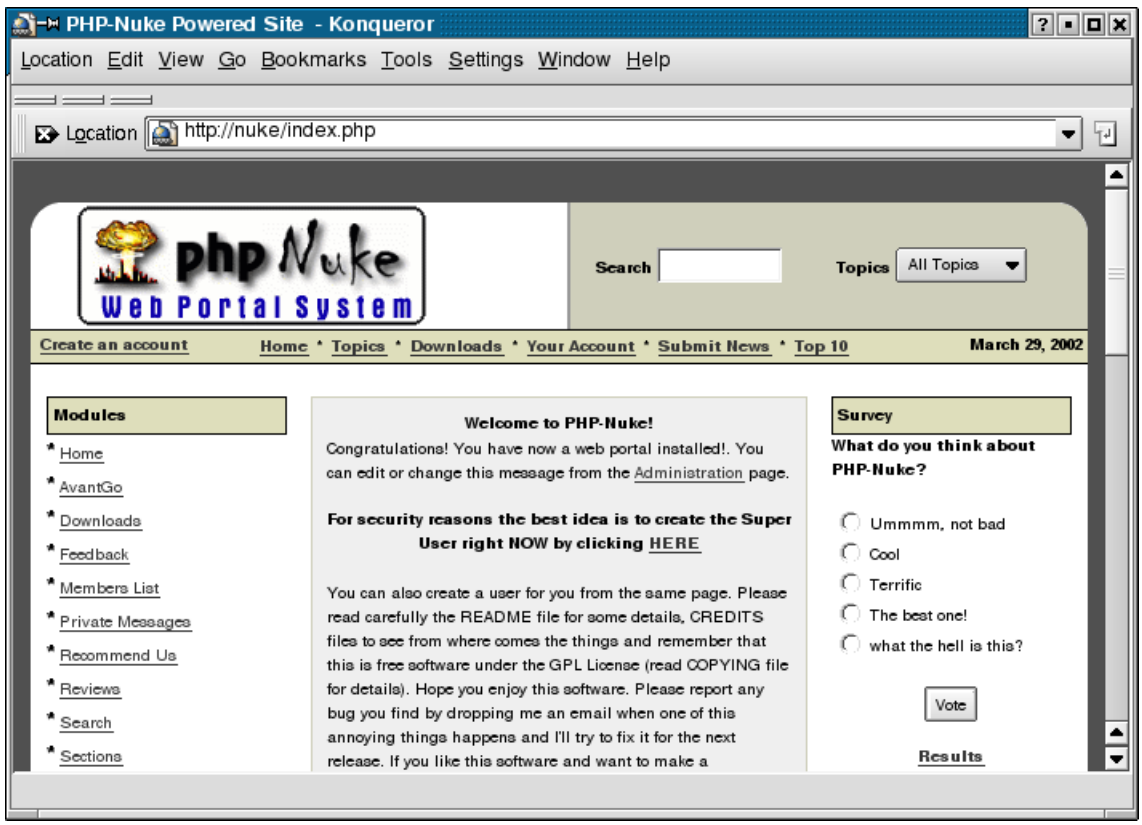
```
[root@hven /root]# mkdir /usr/local/PHPNuke
[root@hven /root]# mv PHP-Nuke-5.5.tar.gz /usr/local/PHPNuke
[root@hven /usr/local/PHPNuke]# tar xzvf PHP-Nuke-5.5.tar.gz
```

Læs gerne filen `INSTALL` som indeholder installations-noter. I store træk skal man køre følgende kommandoer. Først oprette en nuke-bruger i SQL-databasen, køre en opsætning ind. Rette alle filer og kataloger så alle kan læse og skrive alt. Flytte `config.php` ud fra web-serverens søgesti og endeligt få Apache til at læse `config.php` fra den nye sti.

```
[root@hven /usr/local/PHPNuke]# mysqladmin create nuke
[root@hven /usr/local/PHPNuke]# mysql nuke < sql/nuke.sql
[root@hven /usr/local/PHPNuke]# find . -type f -exec chmod 666 {} \;
[root@hven /usr/local/PHPNuke]# find . -type d -exec chmod 777 {} \;
[root@hven /usr/local/PHPNuke]# mv html/config.php .
[root@hven /usr/local/PHPNuke]# echo '<?php include("../config.php"); ?>' > html/config.php
```

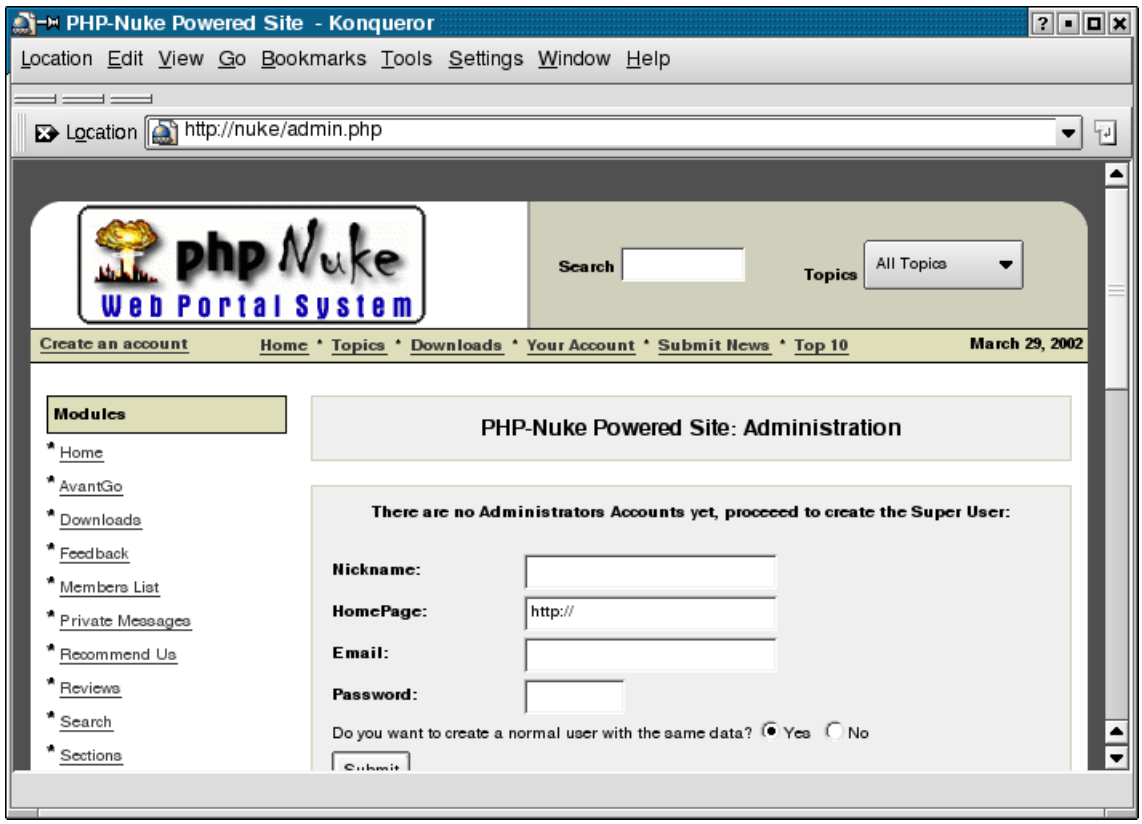
Nu er vi faktisk allerede klar til at starte med opsætningen via en web-browser. Du bør nu kunne få http://www.linux-portal.dk vist hvis du prøver at hente denne hjemmeside.

Figur 7-4. PHP-Nuke admin.php



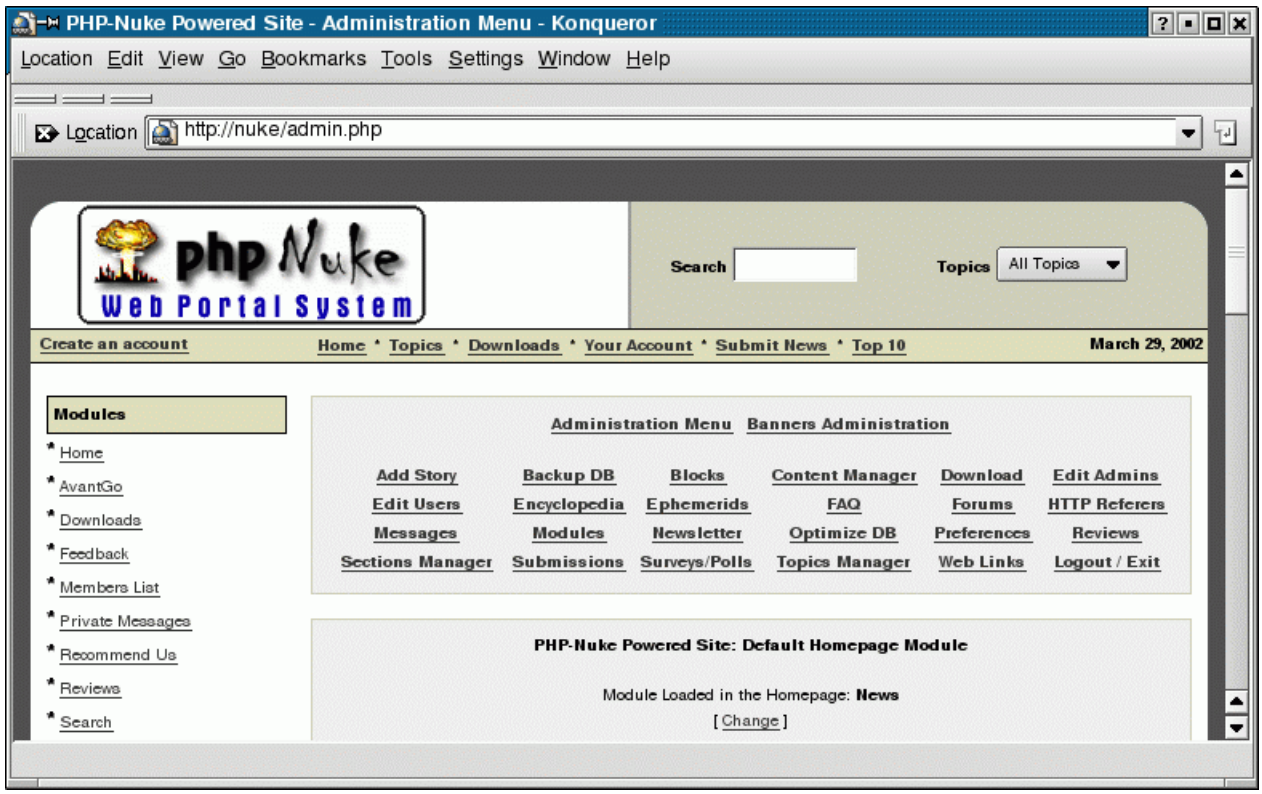
Hent straks derefter http://www.linux-portal.dk/admin.php. Da dette er første gang skal du nu indtaste dit system-administrator opsætning - dvs. dit login-navn (Nickname), og som minimum også password og email-adresse.

Figur 7-5. PHP-Nuke admin.php



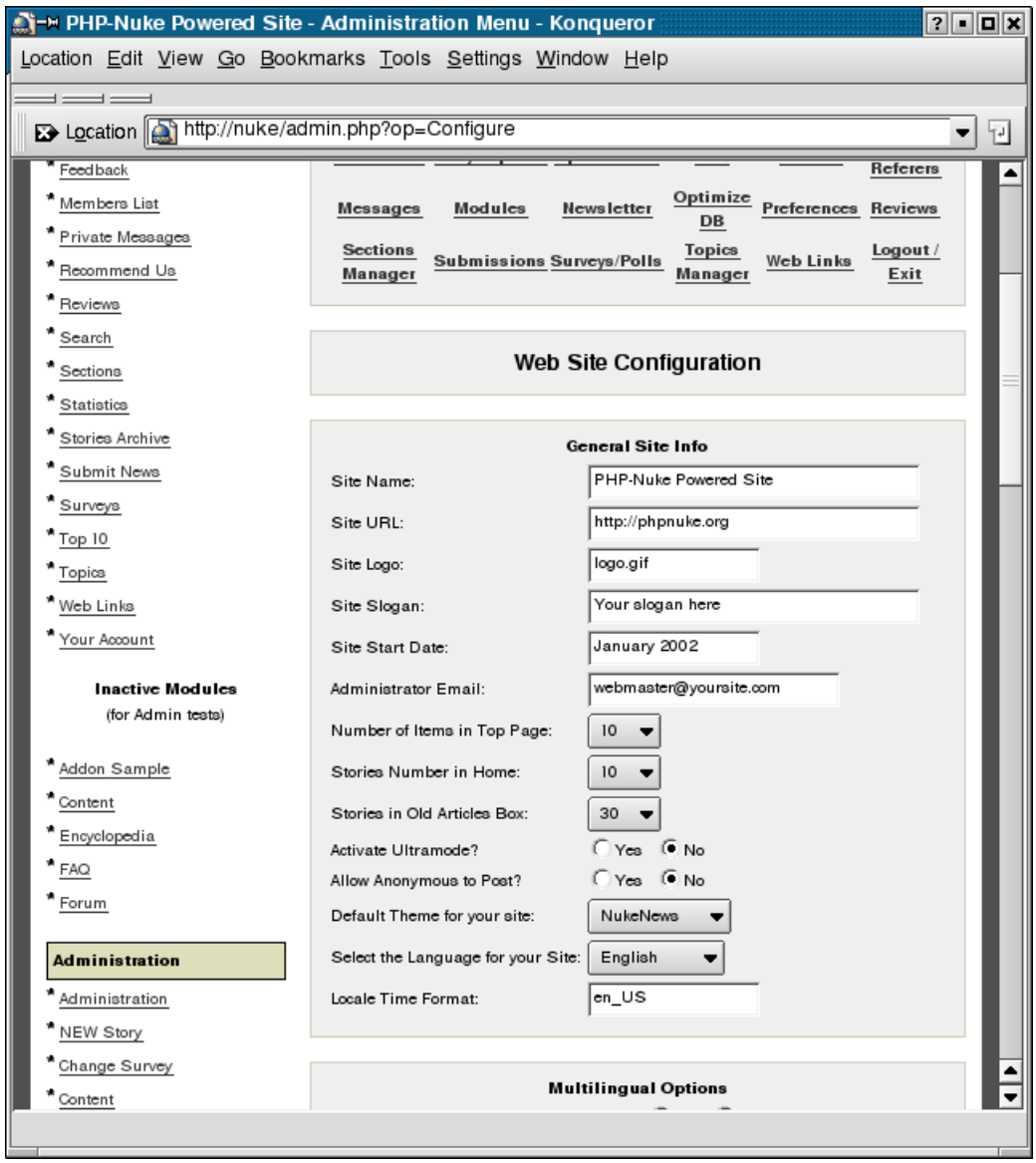
Derefter kan man logge ind i PHP-Nuke med den nye konto. Man kommer derefter ind på følgende side, hvorfra man kan styre hele ens portal.

Figur 7-6. PHP-Nuke admin.php



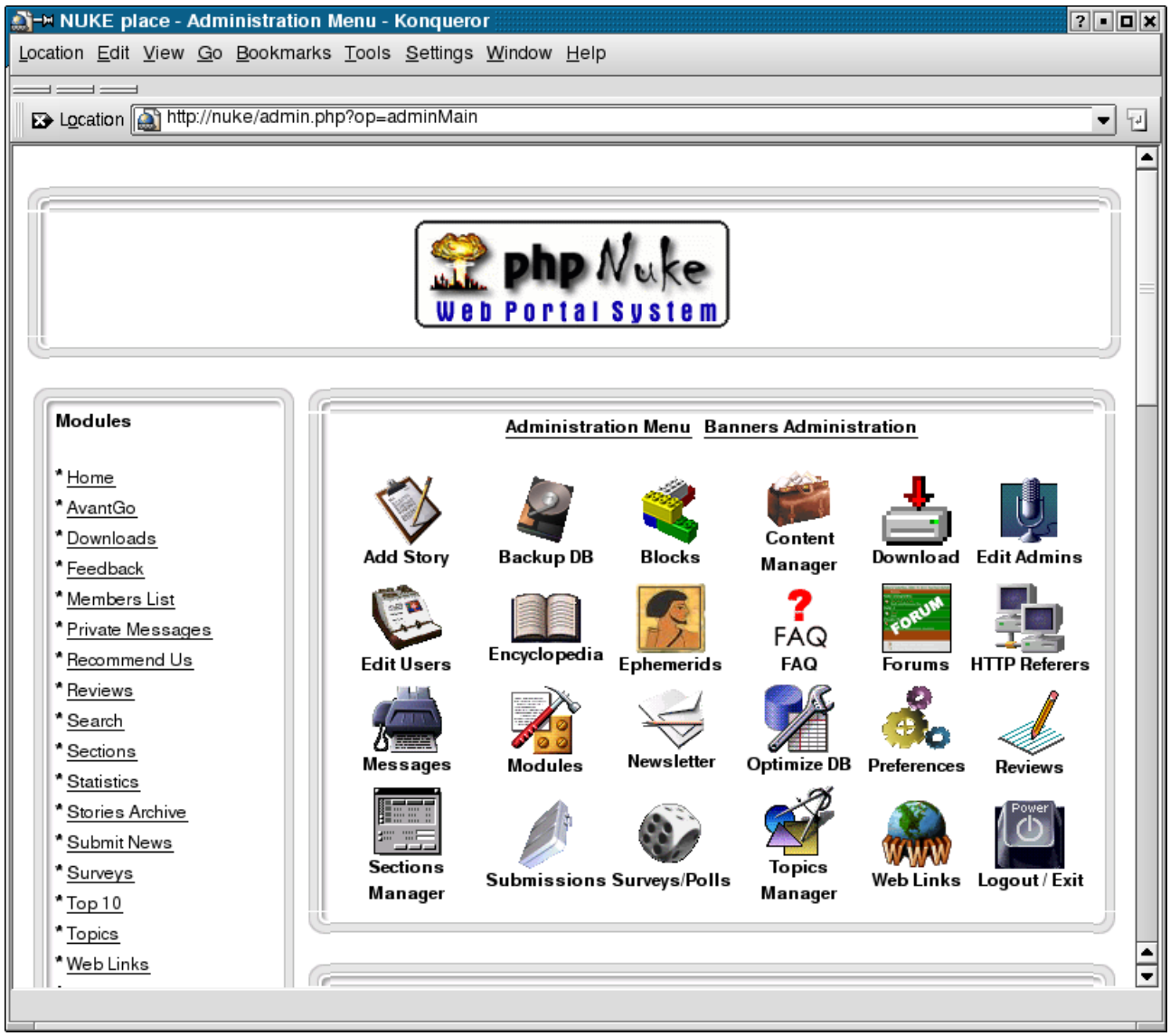
Man kan f.eks. vælge "preferences" og få fyldt reelle information ind om selve hjemmesiden.

Figur 7-7. PHP-Nuke preferences



Man kan også sætte temaer (themes), som ændrer hele udseendet af hjemmesiden. Dette er vist på næste figur.

Figur 7-8. PHP-Nuke admin.php



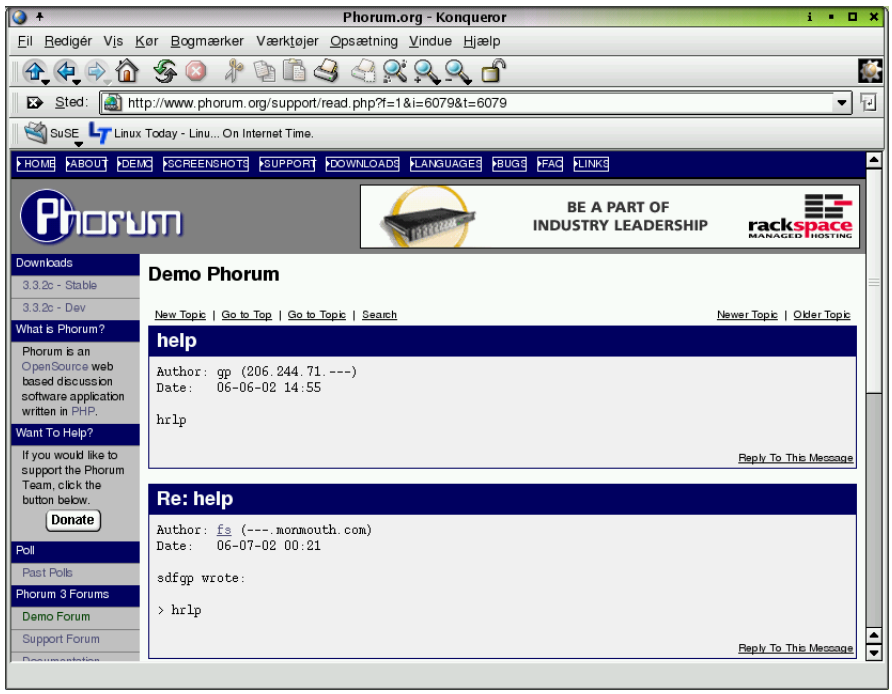
Som system-administrator kan man via admin.php sætte nye historier ind, sætte afstemninger ind osv. Læs mere om dette på http://www.nuketutorials.com/. Se også http://nukesupport.com/.

7.3. Phorum - web diskussionsforum

En nem måde at få lavet et diskussionsforum er at installere Phorum, der kan hentes fra

http://phorum.org. Det hele er skrevet i PHP, dvs. det kræver kun en Apache webserver med PHP installeret.

Figur 7-9. Phorum

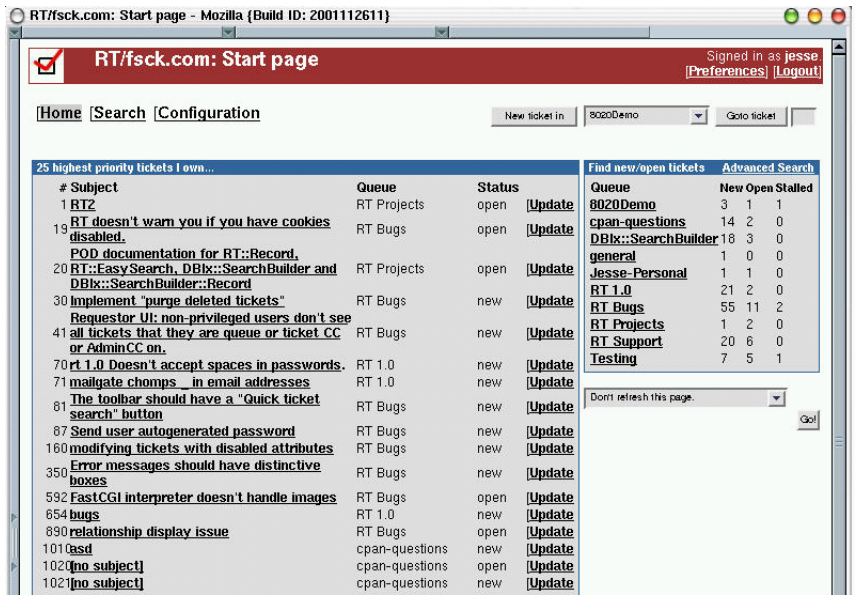


7.4. Helpdesk ticketsystem

Driver men en support-organisation med input, hvor man har en arbejdsgang ala: Der kommer en email til "helpdesken" med en rapport om et problem. Der oprettes automatisk en "billet" på opgaven, og et "billet-nummer" sendes tilbage til opgave-stilleren, så han eller hun kan referere til det nummer. "Billetten" sendes af det automatiske system videre til en gruppe af medarbejdere. En af gruppens medlemmer tager ejerskab af "billetten" og der skabes i det automatiske system et udkast til svar til opgave-stilleren. Udkastet udfyldes med det komplette svar efter opgaven er løst, og alle er glade.

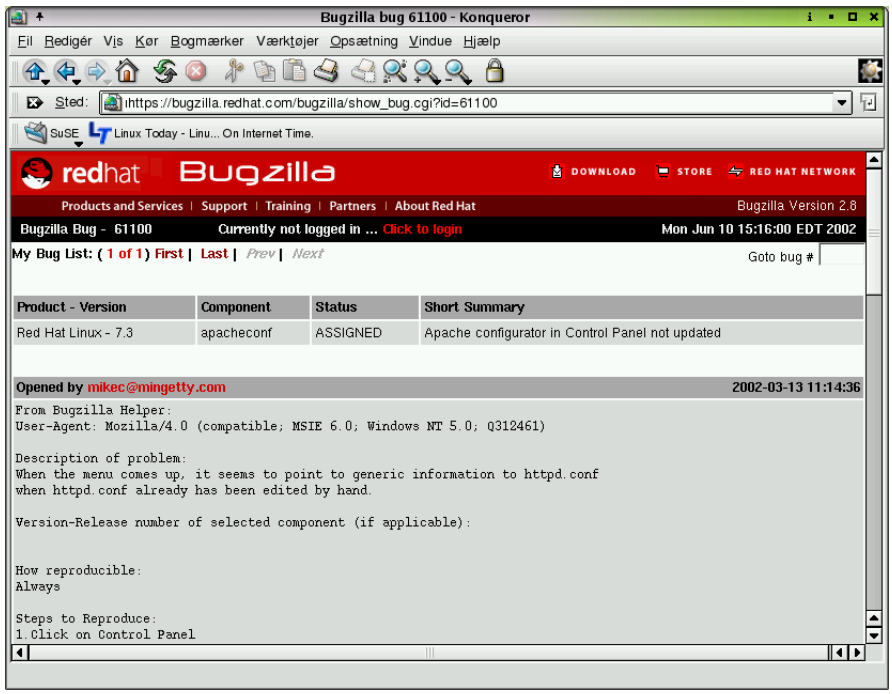
RT også kaldes Request Tracker kan hentes fra http://www.fsck.com/projects/rt/.

Figur 7-10. rt



Det vel nok mest anvendte program til fejl-håndtering i Open Source-verdenen er Bugzilla, som f.eks. Red Hat anvender til deres fejl-håndtering. Alt foregår via internettet. Et rigtigt live-eksempel kan findes på http://bugzilla.redhat.com. Bugzilla kan hentes fra http://www.bugzilla.org/.

Figur 7-11. Bugzilla



Appendiks A. Revisionshistorie for bogen

Denne bog er skabt ud fra version 3.9 af "Linux - friheden til at vælge", som blev splittet i mindre dele.

Igennem tiden har bogen "Linux - friheden til programmere" udviklet sig meget. Vi frigiver ofte nye versioner, når der er kommer en del rettelser ind, eller nye afsnit er blevet skrevet. Kommentarer, ris og ros, og specielt fejl og mangler bedes sendt til linuxbog@sslug.dk (mailto:linuxbog@sslug.dk), men er du medlem af SSLUG, så skriv til sslug-bog@sslug.dk (mailto:sslug-bog@sslug.dk). Alle kan bidrage - se om tilmelding se på http://www.sslug.dk/tilmeld.

Her er en liste over, hvad der er ændret i bogen.

- Version 2.5.20040516 - 16. maj 2004: Donald Axel: Retter sproglig fejl. Jacob Sparre Andersen: Retter sproglige fejl og opmærkning.
- Version 2.5 - 25. januar 2003: Jacob Sparre Andersen: Retter sproglige fejl.
- Version 2.4 - 7. oktober 2003: Claus Sørensen retter kapitlet om SSI. Flemming Bjerke skriver afsnit om installation at Zope helt om. Jacob Sparre Andersen retter mindre sproglige fejl. Peter Toft - tilføjer link til myip.dk.
- Version 2.3 - 6. april 2003: Peter Toft: Dræber dødt link - retter lidt sprog. Skriver at man selv skal lave log-fil kataloger til Apache. Tilføjer link fra Michael Rasmussen om konvertering fra Access databaser til MySQL. Keld Simonsen: Skriver nyt afsnit om infokager/cookies
- Version 2.3 - 1. december 2002: Poul-Erik Hansen - Har rettet adgangskontrol under Apache og skrevet en del om Apache 2.0. Troels Leth Petersen - Fanget en fejl i Python+PostgreSQL-afsnittet. Ole Kofoed Hansen - Forklarer hvorfor MD5-baserede sessionskoder er en fordel. Jacob Sparre Andersen - Opdateret henvisninger til SQL-kurser. Bjørn Bækkegaard retter tekst om WAP.
- Version 2.2 - 1. september 2002: Jacob Sparre Andersen - Harmoniseret brugen af **tar**. Udbedret logiske fejl i SGML-koden. Rettet sproglige fejl. Henrik Christian Grove retter en sproglig fejl fundet af Christian Treldal. Peter Toft tilføjer lidt mere om PHP og Apache-opsætning. Henrik Midtiby har tilføjet PNP/MySQL eksempler.
- Version 2.1 - 14. juni 2002: Kecia Hansen, en SQL-statement fejl. Peter Toft, nyt afsnit om PHP-Nuke (se Afsnit 7.2), Phorum (se Afsnit 7.3), og ticket-systemer til behandling af defekter i en support-organisation (se Afsnit 7.4). Jacob Sparre Andersen: Rettet sproglige småfejl.
- Version 2.0 - 10/3 2002: Ny licens for bogen - Åben dokumentlicens. Frank Jørgensen har rettet kraftigt op på kapitlet om databaser. Keld Simonsen tilføjer nyt om at generere egne SSL-certifikater. Claus Sørensen tilføjer link til mere PHP-information. Peter Toft retter trykfejl og tilføjer et par gode linke om tuning.
- Version 1.9 - 29. november 2001: Peter Toft: Link til mere om Linux og WAP. Jacob Sparre Andersen: Rettet enheder til (MHz = megahertz, Mbit = megabit, Mb = megabyte). Sproglige smårettelser.
- Version 1.8 - 9. juli 2001: Peter Toft: Tilføjet en henvisning til ny bog om Office. Rettet SGML-fejl.
- Version 1.7 - 24. maj 2001: Peter Toft: Rettet en lille SGML-fejl. Nyt kapitel om administration af web tilføjet. Gitte Wange har skrevet nyt afsnit om Zope og CMF. Rettelser fra Erik Søe Sørensen. Jacob Sparre Andersen: Rettet lidt op på sproget i databasekapitlet.

- Version 1.6 - 15. april 2001: Jacob Sparre Andersen har rettet sproglige fejl og rettet lidt op på SGML-koden.
- Version 1.5 - 12. marts 2001: Michael Rasmussen har skrevet et ny kapitel om PHP. Erik Sørensen og Peter Toft har luget mange sproglige fejl ud. Peter Toft har skrevet om at få sin Apache webserver til at fungere som WAP-server, herefter har Hans Schou tilføjet og rettet. Benny Bøtchiær Thomsen fandt en del fejl, som Jacob har rettet op på. Jacob Sparre Andersen stødte også på nogle sproglige fejl. Henrik Christian Grove fandt en dum trykfejl i oversigt over bøger.
- Version 1.4 - 4. februar 2001: Ny bog om Docbook nævnt i serien. Tommy Mogensen fandt en sprogbøf. Jacob Sparre Andersen: Rettet diverse sproglige fejl.
- Version 1.3 - 29. december 2000: Hans Schou; nyt om .htaccess. Peter Toft; nyt om virtuelle webservere på samme maskine. Har også rettet flere småfejl. Link til den nye bog "Linux - friheden til at programmere i C". Jacob Sparre Andersen har lavet et stort arbejde for at få rettet sprog op i bogen.
- Version 1.2 - 26. oktober 2000: Sproglige rettelser fra Frank Damgaard og Michael Lykke.
- Version 1.1 - 30. august 2000: Stort afsnit om CGI og tilsvarende om SSI af Carsten Svaneborg. Mindre rettelser fra Frank Damgaard.
- Version 1.0 - 30. juli 2000: Første offentlige version af bogen. Frank M.G. Jørgensen har skrevet om MySQL.

Stikordsregister

Symboler

A

C

D

H

I

K

M

O

P

R

S

T

V

W

Z