Programmeringsspråket PHP och webbprogrammering

Tobias Zetter

Referat

PHP är ett populärt skriptspråk, som huvudsakligen används för att generera dynamiska webbsidor. Denna avhandling behandlar hur en webbsida är uppbyggd och hur man konstruerar en webbapplikation med hjälp av PHP. Detta inkluderar en snabb genomblick på internetprotokoll som t.ex. HTTP och hur man generar den information som krävs för att en webbläsare skall visa en webbsida på korrekt sätt. PHP jämförs dessutom med andra alternativ för webbutveckling. Detta med fokus på själva utvecklingen, upprätthållande och prestanda. Avhandlingen behandlar även hur programmeringsspråket PHP i grunden fungerar, samt om dess extensioner som möjliggör kommunikation med till exempel databaser.
Nyckelord: PHP, webbapplikation

Innehåll

1 Introduktion

När internets popularitet ökade i början av 90-talet uppstod snabbt ett behov för dynamiska webbsidor.

Till en början ansågs det svårt att skapa sådana. Detta eftersom webbapplikationerna måste skrivas i programmeringsspråk som inte var utvecklade för eller ens hade ordentligt stöd för ändamålet.

När man konstruerar en webbapplikation finns en massa som bör tas i beaktan, som i sig ställer krav på programmeringsspråket.

Denna avhandling behandlar en webbsidas uppbyggnad, kommunikationen mellan webbserver och webbläsare, kraven som ställs på ett programmeringsspråk vid webbutveckling samt hur programmeringsspråket PHP tillfredsställer dessa.

Bakgrund

Webbteknologins rötter kan hittas i de ursprungliga Internet protokollen (även kända som TCP/IP) som utvecklades på 1980-talet. Dessa protokoll var en vidareutveckling på ARPANET, som utvecklades av USA:s försvarsdepartement (DoD) för att skapa en öppen, distribuerad och decentraliserad nätverksarkitektur.

Tidigare arkitekturer hade flera svaga punkter, såsom att det alltid måste finnas en central punkt där all kommunikation gick igenom.

Med hypertext avses text som man med hjälp av hyperlänkar kan navigera till önskat ställe i. Konceptet har existerat sedan 1940-talet, men själva ordet myntades år 1965 av Ted Nelson i samband med hans system Xanadu.

Med koncept från tidigare hypertext system föreslog Tim Berners-Lee 1989 skapande av system som senare ledde till the World Wide Web, även idag allmänt kallat webben.

Snabbt därefter skapades teknologierna HTML(*Hyper Text Markup Language)* och HTTP(*Hyper Text Transfer Protocol)*, som ännu i dag utgör grunder för internet och WWW.

2 Kommunikation med webbservrar

För att förstå kraven som ställs på ett programmeringsspråk vid webbutveckling krävs att man förstår grunderna bakom en webbsida och internet.

Processen som sker vid överföring av en webbsida är i grunden ganska simpel. Processen bygger på server-klient principen, vilken innebär att det finns en serverdator försedd med en webbserverapplikation (till exempel Apache) som väntar på en klient ansluter. Klienten skickar en förfrågan varefter servern ger en respons. Klienten, som till exempel kan vara en webbläsare behandlar därefter resultatet som kan innebära att rendera en webbsida.

2.1 URL

URL (Uniform Resource Locator) är den formella benämningen på en webbadress. URL uppfanns 1989 i samband med att World Wide Web lanserades. Webbadresser används exempelvis i webbläsare för att definiera platsen för en resurs.

Ett URL består av upp till sex delar: protokoll, domän, port, sökväg, query och ankare.

Exempel på ett URL som innehåller alla delar:

```
http://www.example.com:80/test/test.php?test=test&test2=test2#test
```

Även användarnamn och lösenord kan läggas direkt i webbadressen för autentiseringssyften i vissa situationer.

URL ger webbläsaren information om vad som skall visas som därefter görs om till en HTTP förfrågan.

2.2 TCP/IP

TCP/IP är en samling med protokoll. Namnet TCP/IP refererar till de två viktigaste protokollen inom samlingen: TCP (Transmission Control Protocol) och IP (Internet Protocol).

Protokollen kan delas in i följande lager:

- Applikationslagret
- Transportlagret
- Internetlagret
- Länklagret (bestående av ett datalänkslager och ett fysiskt lager)

2.3 HTTP

HTTP (HyperText Transfer Protocol) är ett kommunikationsprotokoll som används för att överföra information över internet. HTTP utvecklades ursprungligen med syftet att överföra webbsidor från server till klient. HTTP utgör grunden för webben (WWW) och hör till applikationslagret av TCP/IP.

HTTP fungerar allmänt genom att en klient skickar en förfrågan till en TCP-port på en server. Transportprotokollet TCP används vanligen eftersom HTTP kräver en pålitlighet som få andra protokoll kan garantera.

TCP kontrollerar de vanligaste fel som kan uppstå vid överföring: dubbla paket, paket i fel ordning, paket som saknas och korrupta paket. TCP kontrollerar dessa fel och begär en omsändning i fall situationen s kräver. Även andra protokoll som t.ex. UDP kan användas för vissa specifika tillämpningar där hastighet prioriteras över pålitlighet.

Standardportnumret för HTTP är 80. Information om portnumret kan därmed skippas i webbadresser som refererar till webbservrar som opererar på denna port.

Det finns flera versioner av HTTP, av vilka främst HTTP/1.0 och HTTP/1.1 numera är i användning. HTTP/1.1 kan till skillnad från sin föregångare hålla en anslutning i liv och kräver att man anger ett värdnamn vid anslutning. Denna skillnad är nödvändig för att kunna skilja olika domän-namn, ifall en serverdator fungerar som server för flera olika domännamn.

HTTP definierar 9 metoder som indikerar åtgärden som skall göras på en identifierad resurs: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT och PATCH.

GET används för att be servern skicka en utvald information till klienten. Detta är det absolut vanligast använda kommandot, och används oftast då en webbläsare hämtar information från en webbserver.

En simpel HTTP-förfrågan (request headers) utan extra information kan se ut såhär:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Responshuvudet (response headers) som mottas av klienten kan då se ut såhär:

```
HTTP/1.1 200 OK
Date: Fri, 25 Feb 2011 15:30:35 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.2.10
Connection: close
Content-Type: text/html; charset=UTF-8
```

Vanligen innehåller dock både förfrågan och responsen en del mera information. Om ”request headern” innehåller metoden GET eller POST, skulle förutom headern till exempel en webbsida returneras.

IETF (Internet Engineering Task Force) har definierat en lista över tillåtna headers som kan användas i kommunikation över HTTP. Dessa används aktivt i kommunikation mellan klient och server.

HTTPS är en variant på HTTP som även enkrypterar överföringen med antingen protokollen SSL eller TLS. Enkryptering av en överföring försvårar avlyssning och används därför ofta vid förmedling av känslig data såsom lösenord eller kreditkortsnummer.

Förutom att informationen är enkrypterad fungerar HTTPS på samma sätt som HTTP. HTTPS fungerar vanligen på port 443.

3 En webbsidas struktur

3.1 HTML och XHTML

Webbsidor är i dagens läge i grund uppbyggda med HTML (*Hypertext Markup Language).*

HTML är en standard för strukturering av hypertext och inbyggda objekt som primärt används för webbsidor.

XHTML (*Extensible HyperText Markup Language*) är en vidareutveckling av HTML som omformulerar HTML i enligt en striktare XML standard.

Trots att dessa är skilda märkspråk är de i grunden ganska lika och bygger på olika typer av element som framställs med taggar. Dessa taggar, som i sig förutom ett innehåll kan ha flera attribut ställer upp sidans innehåll på ett sätt som kan presenteras av en webbläsare.

Webbsidor innehåller vanligen även Javascript, som används för att ändra på sidan innehåll utan att ladda om sidan och på så sätt möjliggör direkt användarinteraktion.

Med Javascript, eller mera specifikt AJAX (*Asynchronous JavaScript and XML)* kan man även hämta och förmedla information mellan klient och server utan att behöva ladda en hel sida.

Javascript kan antingen bäddas in direkt i ett HTML dokument, eller refereras till som en extern resurs.

Samma sak gäller CSS (*Cascading Style Sheets),* sidans stilmall som utgör ett sätt att ändra på sidans utseende.

Även teknologier som Flash eller externa mediaspelare kan bäddas in i HTML kod.

3.2 Webbformulär

Webbformulär används för att förmedla information mellan klient och webbserver.

Det finns två metoder för att skicka formulär-data till servern: GET och POST. Metoderna skiljer sig från varandra genom att det som skickas med GET blir synligt i webbsidans URL.

Information som skickas med GET kan enbart innehålla text och rekommenderas inte att använda för att utföra ändringar på en sida. POST används ofta i situationer då man utför en åtgärd på en webbsida, som till exempel vid inloggning. POST kan även användas för att ladda upp filer till servern. Information som skickas med GET har ofta en max-längd som är definierad av webbläsare och vissa webbservrar.

Ett enkelt webbformulär:

```
<FORM action="form.php" method="post">
<INPUT type="text" name="test" value="Testdata">
<INPUT type="submit" value="Skicka">
</FORM>
```

Detta framställs av webbläsare ungefär på följande sätt:

När webbformuläret skickas ser dess ”request headers” ut såhär:

```
GET /form.php?test=Testdata HTTP/1.1
Host: www.example.com
```

Om POST-metoden istället skulle ha använts skulle det istället se ut såhär:

```
POST /form.php HTTP/1.1
Host: www.example.com
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

test=Testdata
```

3.3 Cookies

Cookies används för att spara information i en webbläsare. Cookies skapas vanligtvis på en webbsidas begäran, genom information i responshuvudet. Därefter levererar webbläsaren information om cookien till webbservern vid varje förfrågan, antingen tills den inte mera är i kraft eller användaren manuellt tar bort den.

En cookie är antingen i kraft till en viss tidpunkt eller till sessionens slut. Cookies används vanligen för att identifiera en specifik webbläsare för att kunna leverera användarspecifik information.

4 PHPs uppkomst

Webben bestod till en början nästan enbart av statiska webbsidor. Webbsidorna uppdaterades av sina skapare, men varierade allmänt inte i innehåll på basen av användarens interaktioner.

Ett behov för dynamiskt innehåll uppstod dock snabbt och begreppet webbapplikation skapades. På tiderna före PHP var sådana applikationer vanligen skrivna i de generella programmeringsspråken C eller Perl. (Hudson, PHP in a Nutshell, 2006)

Utvecklingen av PHPs började när Rasmus Lerdorf år 1994 skapade en sampling personliga Perl-skript. Deras syfte var att göra vanliga uppgifter inom webbprogrammering lättare och mindre upprepande (Hudson, PHP in a Nutshell, 2006). PHP stod ursprungligen för ”Personal Home Page”.

Lerdorf skrev senare om dessa skript som CGI binärer och lade till funktionalitet för att arbeta med webb-formulär och samverka med databaser. Denna version av PHP kallades PHP/FI (Personal Home Page/Forms Interpreter).

Lerdorf gjorde juni 1995 PHP tillgängligt för allmänheten för att få snabba upp utvecklingen. Denna version hade redan en betydande del av den bas-funktionalitet som PHP har idag. Ett utvecklingsteam började formas kring PHP varefter PHP/FI 2 släpptes i november 1997.

Israelerna Zeev Suraski och Anti Gutmans skrev därefter om PHP parsern år 1997. Denna omskrivna kod utgjorde grunden till PHP 3. Därefter skrev de om hela PHPs kärna som producerade Zend Engine. PHPs namn ändrades då till den nuvarande rekursiva akronymen ”PHP: Hypertext Preprocessor”. Den 22.5.2000 släpptes PHP 4, som nu använde sig av Zend Engine.

PHP 5, som är den senaste stabila utgåvan släpptes i juli 2004 innehållande en ny effektivare version av Zend Engine, kallad Zend Engine II.

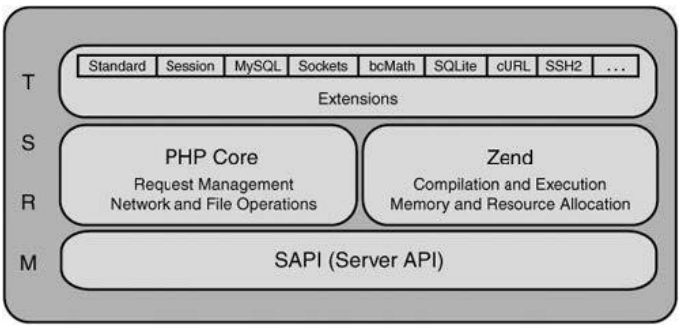
5 PHPs uppbyggnad

5.1 PHPs anatomi och livscykel

PHP består fem delar, alla nödvändiga för att systemet skall fungera på korrekt sätt.

Dessa delar är: SAPI, PHP kärnan,

Zend Engine, extensioner och TSRM som illustreras i Figur 1.

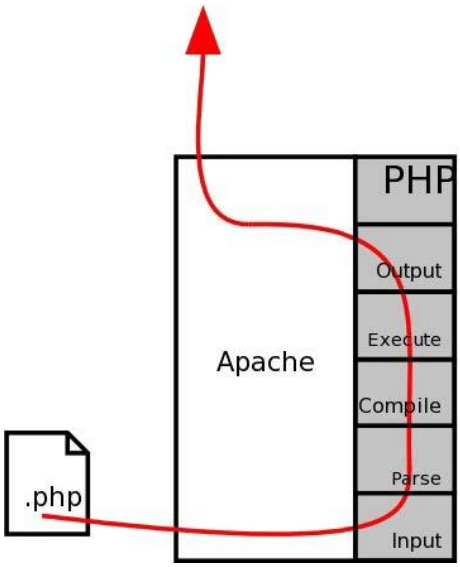


Figur 1: PHPs anatomi (Golemon, 2006)

Med SAPI avses det lager som kommunicerar med webb-servrar (som till exempel Apache). PHP kärnan behandlar grundläggande operationer som till exempel fil-operationer, fel hantering samt start och avslut.

PHPs kärna består av två separata delar. På lägsta nivå hittas Zend Engine. Zend tolkar (parsar) den läsbara koden och omvandlar den till maskinläsbar bytekod.

Zend exekverar därefter bytekoden i en virtuell maskin. Zend Engine sköter även om minneshantering, variabelutrymmet (variable scope) och skickande av funktionsanrop.



Figur 2: PHPs livscykel

Den andra delen, PHP kärnan sköter kommunikationen med och kopplingen till SAPI (Server Application Programming Interface) lagret, som även ofta refererar till värdmiljön – Apache IIS, CLI, CGI etc. Den bildar även en unifierad kontroll lager för *safe_mode* och *open_basedir* kontroller (checks), och stream-lager som associerar fil och nätverks I/O med userspace funktioner som *fopen(), fread(),* och *fwrite()*.

Ovanom kärnan och Zend ligger ett lager med extensioner som bidrar med nästan alla PHPs userspace funktioner.

Den sista delen är TSRM (Thread Safe Resource Management) lagret som möjliggör att en ensam PHP instans kan exekvera flera uppgifter samtidigt utan att orsaka problem.

5.2 PHP Extensioner

I stort sett varje userspace funktion i PHP kommer från en extension. PHP kommer som standardf med omkring 86 extensioner. Tillsammans bidrar dessa extensioner med över 2500 userspace funktioner. I PECL paketsamlingen (repository) finns ytterligare över 100 extensioner som bidrar med ytterligare funktionalitet.

Med en extension avses en kodsamling som kan kopplas till att fungera med PHP tolkaren för att ge mera funktionalitet till userspace skript. Extensioner ger i regel åtminstone en funktion, class, resurstyp eller stream implementation, ofta flera av dessa i kombination (Golemon, 2006).

Den mest utbrett använda extensionen *standard* definierar mera än 500 funktioner, 10 resurstyper, 2 klasser och 5 stream wrappers. Denna extension tillsammans

med *zend_builtin_functions* extensionen kompileras alltid med PHP, oberoende av andra konfigurationsinställningar. Därför ses denna liksom ett antal andra extensioner inte egentligen som en extension (Golemon, 2006).

Andra extensioner som till exempel session, spl, pcre, mysql och sockets aktiveras eller avaktiveras med konfigurationsinställningar eller genom phpize verktyget, som används för att aktivera extensioner.

Extensioner används ofta för att länka ihop PHP med ett annat bibliotek och förse PHP med dess funktionalitet.

Med hjälp av extensioner kan man avsevärt snabba upp sin PHP applikation, eftersom PHP inte erbjuder samma prestanda som kompilerad C eller C++.

5.2.1 PEAR

PEAR (PHP Extension and Application Repository) är ett distributionssystem för användbara PHP komponenter (Pear, 2011).

PEAR erbjuder ett strukturerat kodbibliotek för PHP användare där utvecklare kan distribuera sina egna paket som sedan kan installeras genom PEAR Package Manager och därefter finnas tillgänglig på alla ställen.

Koden som distribueras genom PEAR är trots allt enbart PHP-kod och inte egentliga extensioner. Detta efter att distributionskanalerna delades upp i PEAR och PECL.

Däremot används PEAR Package Manager även för installation av extensioner från PECL.

5.2.2 PECL

PECL (PHP Extension Community Library) är ett av PHPs systerprojekt som erbjuder ett sätt att distribuera PHP extensioner.

Flera extensioner börjar ofta i PECL för att senare bli en av PHPs standardextensioner.

6 Användning

PHPs popularitet kopplas ofta med hur enkelt programmeringsspråket är att använda och komma igång med.

Syntaxen är lik de ”klassiska” programmeringsspråken som till exempel C++ och är därmed lätt att komma igång med om man tidigare har programmerat.

PHP är även en av de få programmeringsspråk som uttryckligen har skrivits med tanke på webbutveckling och innehåller därmed massvis med funktionalitet lämpat för ändamålet.

6.1 Syntax

PHP är ett dynamiskt skripspråk. Ett skriptspråk är ett tolkat programspråk som körs i en speciell exekveringsmiljö. Skriptkod är ofta portabel och kan därmed köras i flera olika operativsystem utan redigering.

Att ett programmeringsspråk implementerar dynamiskt typning innebär att variabler i sig inte har en datatyp. Däremot har variablernas värden alltid en datatyp. Denna datatyp kan dock förändras under körningen av programmet.

Variabler av olika datatyper kan jämföras med varandra med logiskt resultat, och ett tredje likhetstecken har implementerats för att även kunna jämföra med specifik datatyp.

```
<?php

$a = 5;
$b = '5';

if($a == $b) {
    //true
}

if($a === $b) {
    //false
}

?>
```

Exemplet ovan visar skillnaden mellan dubbla och tredubbla likhetstecken. Heltalet med samma värde som strängen returnerar true vid normal jämförelse.

I PHPs syntax kan lätt märkas dess rötter. PHP började som en samling Perl-skript och skrevs sedan om i C. I PHPs syntax kan hittas klara likheter med bägge programmeringsspråken som illustreras i följande exempel:

```
#!/usr/bin/perl -w
use strict;
my $w = "world";
my $h = "Hello $w!\n";
print $h;
```

```
<?php

$w = "world";
$h = "Hello $w!\n";
print $h;

?>
```

Exemplen ovan visar likheter mellan PERL och PHP (nedanför).

```
#include <stdio.h>

int main(void) {
        char world[6] = "world";
        printf("Hello %s!\n", world);
        return 0;
}
```

Exempelkod i C.

```
<?php

$world = "world";
printf("Hello %s!\n", $world);

?>
```

Motsvarande kod i PHP.

Samtliga kodexempel skulle skriva ut ”Hello world!” och avsluta med en blank rad.

Något som hade betydande inverkan på PHPs framgång var möjligheten att bädda in koden direkt i HTML. PHPs parser identifierar vilken del av källkoden som ska tolkas som HTML och vad som endast direkt skall skickas ut med hjälp av startande och avslutande taggar. Detta är till stor fördel vid generering av webbsidor vars innehåll till stor del bestod av statisk text. I dagens läge med komplexa hemsidestrukturer, som sällan består av en enda fil är nyttan däremot liten.

6.2 Superglobala variabler och deras användning

För att kunna skapa ett dynamiskt innehåll är det vitalt för programmeringsspråk med syftet för webbutveckling att kunna hantera all data som får in från webbläsaren i samband med förfrågan till webbservern samt en del attribut specifika för webbservern.

Denna information finns tillgänglig i PHP i följande superglobala variabler:

- *$GLOBALS*
- *$_SERVER*
- *$_GET*
- *$_POST*
- *$_FILES*
- *$_COOKIE*
- *$_SESSION*
- *$_REQUEST*
- *$_ENV*

Variabeln $_SERVER innehåller information om vilket dokument som ska visas, hur man har kommit åt dokumentet: bland annat IP-adressen som har begärt dokumentet, porten som användes samt utvald information från request headern såsom webbadressen som refererat till sidan och webbläsarens egen information. Variabeln innehåller dessutom serverns alla vitala information som kan påverka på informationen som ska visas.

Information som förmedlats från webbformulär eller genom länkar till webbsidans URL kan hittas i variabeln $_GET. På motsvarande vis kan data som skickats med ett formulär med metoden ”post” hittas i räckan $_POST.

Kakor, d.v.s. cookies som sparas i webbläsaren skickas vid varje förfrågan till servern. Dessa är lätt åtkomliga i räckan $_COOKIE.

$_REQUEST innehåller kombinerad information från $_POST, $_GET och $_COOKIE.

De tre sistnämnda är fullt editerbara räckor. Ändring av data i $_COOKIE skapar dock inga cookies, utan en dylik ändring skall göras med funktionen setcookie.

6.3 Frameworks

PHP är känt för att ha en stor mängd färdiga skript som man kan utgår från vid webbutveckling. Exempel på dessa är bland annat Wordpress, Drupal, PHP-Nuke, Joombla och E-Commerce.

Alla dessa erbjuder en färdig kodbas som relativt lätt kan ändras utseendemässigt för att passa sina egna behov. Med dessa skapas lätt en färdig blogg, webbsida med lätta ändringsmöjligheter eller webbshop.

Samma situation finns även med frameworks (ramverk), som snabbar upp utvecklingen samt ger en färdig struktur för ens webbapplikation.

Flera webbutvecklare väljer att skapa sina egna frameworks. Det finns däremot ett stort mångfald med färdiga frameworks. Exempel på kändare frameworks är bland annat Cake PHP, Zend Framework och sympony.

Majoriteten av alla frameworks skrivna till PHP bygger på MVC konceptet, som går ut på att skilja ens kod i olika delar.

MVC konceptet blev speciellt populärt inom PHP kretsarna när PHP 5 släpptes år 2005, med full objektorientering med bekant syntax från C++.

6.4 Prestanda och optimering

6.4.1 Allmänt om webboptimering

När det kommer till webbsidor är hastighet alltid någonting som har prioriterats.

Idag, när webben till stor del består av dynamiska webbsidor genereras en webbsida på nytt för varje visning. När det kommer till total laddningstid av en normal webbsida är dock denna genereringstid ofta av mindre betydelse.

Detta beror på att en webbsida väldigt sällan endast består av enbart text, utan oftast även innehåller referenser till andra filer såsom bilder och externa filer för CSS (stilregler) och Javascript. Dessa resurser är så gott som alltid externa eftersom de ofta har ett statiskt innehåll och därför kan cachas av webbläsaren och enbart laddas om vi förändring.

Webbservrar returnerar allmänt en så kallat ”ETag” eller entitetsetikett med genom vilken webbservern i samarbete med webbläsaren kan avgöra om en resurs har ändrats sen senaste hämtning.

Om man realistiskt ser på optimering av en webbplats så går den mest betydande optimeringen ut på att minimera antalet förfrågningar till webbservern, genom att till exempel kombinera resurser och genom att ange rätt attribut i webbserverns responshuvud.

Detta eftersom varje anslutning för hämtning av en resurs har en bytande responstid. Samtidigt hämtar webbläsare i allmänhet högst 4 bilder åt gången från samma värdnamn och på motsvarande vis endast en Javascript eller CSS fil åt gången.

Statiska resurser bör få en expiration tid för att inte behöva laddas om varje gång. Om detta kombineras med avstängning av Etags för dessa resurser kommer inte webbläsaren att behövs ladda om dessa resurser och därmed spara tid vid hämtning av sidan.

Vanliga exempel på optimering förutom den uppenbara minimeringen och komprimeringen av resurser är att kombinera CSS eller Javascript filer. Bilder kan

även kombineras till såkallade ”image sprites”, som går ut på att kombinera flera småbilder till en större och därmed minska antalet förfrågningar.

Man kan även snabba upp laddningen av en webbsida genom att hålla anslutningar i liv, vilket är möjligt genom HTTP 1.1.

Dessa ändringar sker vanligen genom ändring av webbserverns inställningar. Däremot är de viktiga att komma ihåg om man genererar dylika resurser genom till exempel PHP.

Det finns flera användbara verktyg som hjälper webbutvecklare att snabba upp laddningen av deras webbsidor, till exempel Googles Page Speed och Yahoos YSlow.

Även om generationstiderna för webbsidor väldigt sällan utgör en större del av den totala laddningstiden bör man komma ihåg att övriga resurser inte kan börja laddas före själva HTML-sidan är laddad och definierar vilka externa resurser som bör hämtas eller kontrolleras efter förändring.

Därför utvecklas hela tiden verktyg för att snabba upp tiden som PHP använder för att generera sidorna.

6.4.2 Accelleratorer

När det kommer till prestanda har interpreterade språk alltid kritiserats för den långa process de går igenom vid varje sidgenerering. Att tolka och kompilera skript vid varje körning är kanske idealiskt i en utvecklingsmiljö (då man inte manuellt behöver kompilera om före varje körning), men inte det bästa alternativet i en produktionsmiljö där prestanda prioriteras.

Flera extensioner för att minska på arbetsbördan vid varje hämtning har dock utvecklats, bland annat eAccellerator, Zend Optimizer+, XCache och APC.

Accelleratorna funktionalitet varierar lite men bygger ofta på att cacha bytekoden som annars skulle kompileras vid varje körning.

EAccellerator började som en fork av Turck MMCache projektet av Dmitry Stogov. EAccellerator optimerar den kompilerade bytekoden och sparar den på

delat minne eller på hårddisk. Vid påföljande åtkomst till ett skript använder eAccellerator den cachade bytekoden istället för att kompilera om skriptet.

Med detta undviker man upprepad parsing och kompilering av skript och snabbar därmed upp processen.

Alternative PHP Cache (APC) är liksom eAccellerator ett ramverk som optimerar PHP kod och cachar PHP bytecode i delat minne. APC håller snabbt att bli standarden för PHP skript caching och kommer att inkluderas I PHP kärnan fr.o.m. PHP 6.

Användning av dessa optimeringsextensioner kan man minska körningstiden för genereringen av en webbsida flera gånger om.

6.4.3 Optimering av PHP kod

Precis som i alla andra språk finns det effektiva och mindre effektiva metoder att utföra saker. Trots att något skulle fungera som det ska, kan det i flera fall vara idé att gå utföra lite kodoptimering.

Det finns en massa exempel på simpla ändringar i programkod som kan utföras för att snabba upp körningen. Till stor del gäller samma kodoptimeringsregler som i andra programmeringsspråk, men det finns dessutom massvis med PHP specifika regler som kan snabba upp körningen.

Av egna experiment kan dock dömas att flera av dessa regler antingen är felaktiga, föråldrade eller av mycket liten betydelse.

När det kommer till mera krävande beräkningar eller komplex funktionalitet kan det god idé att skriva om funktionaliteten i C eller C++ och därefter köra den som PHP extension istället. Detta är speciellt allmänt bland större företag som vill snabba upp sin hemsida.

6.4.4 HipHop for PHP

Facebook, som ursprungligen är skrivet i PHP ogillade idén att skriva att använda sig av PHP extensioner eftersom det skulle minska antalet utvecklare som kan arbeta med hela kodbasen. (HipHop Team, 2010)

Som endast av Google överträffad i besökarantal (Alexa, 2011) ansåg de PHPs prestanda inte vara tillräcklig, trots använding av APC och började därmed utvecklingen av HipHop.

HipHop är en källkodsomvandlare som omvandlar PHP kod till optimerad C++. Efter omvandlingen kan C++ koden kompileras med g++ och bilda en webbserver.

HipHop utvecklarna har dessutom skrivit om flera av de vanligaste PHP extensionerna för att ta del av deras prestandaoptimeringar.

HipHop ställer dock några begränsningar i koden och stöder bara PHP upp till version 5.2, fast stöd för den nuvarande versionen 5.3 är planerad.

Nyttan med HipHop har diskuterats då prestandaskillnaderna mellan en uppdaterad PHP version med APC och HipHop inte alltid är av stor betydelse.

Facebook rapporterar dock att de använder 50 procent mindre CPU kraft med HipHop än med deras tidigare konfiguration med Apache och PHP.

7 Kritik

PHP har starkt kritiserats inom vissa kretsar. Den automatiska typkonverteringen anses av dålig och i vissa fall ologisk.

```
<?php

if((string)"false" == (int)0){
    //true
}

?>
```

Exemplet visar att PHP anser textsträngen ”false” vara ekvivalent med heltalet 0 (false). En textsträng returnerar i normala fall true.

Gammal funktionalitet som till exempel *register_globals*, som gör de superglobala variablerna tillgängliga som normala variabler och *magic_quotes_gpc* som escapar alla inputsträngar har ansetts osäkra. Dessa funktioner är däremot i nuläget föråldrade, ersatta av annan funktionalitet och planerade att tas bort i framtida versioner.

PHP har även kritiserats för bristen på namespaces och dåligt stöd för metaprogrammering. Dessa brister har däremot korrigerats i PHP 5.3.

PHP har även kritiserats på grund av sin tolerans mot programmeringsfel och felaktig kod. Flera även välanvända PHP skript (som t.ex. Wordpress) innehåller flera triviala syntaxfel som inte upptäcks av PHP.

En av de största brister PHP i nutida läge har är ett uselt stöd för olika teckenkodningar. Ovana PHP utvecklare stöter ofta på problem med teckenkodning. Detta även i vanliga uppgifter som att skicka epost eller parsa XML. Detta anses som ett stort problem då en stor del av programmerare inte ens vet vad teckenkodning innebär.

Bättre stöd för teckenkodning är dock en av prioriteterna i det kommande PHP 6.

8 Alternativ till PHP

Det finns förstås andra alternativ än PHP vid webbutveckling. Till de populärare hör: ASP.NET, Java, Perl, Python och Ruby.

Popularitet bland programmeringsspråk är svår eller så gott som omöjlig att på ett pålitligt sätt jämföra. PHPs popularitet har minskat de senaste åren, men är fortfarande enligt vissa källor det populäraste alternativet när det kommer till webbutveckling (TIOBE Software, 2011). Detta kan delvis bero på att majoriteten av alla webbhotell har PHP installerat på sina servrar. PHP är därmed ofta ett billigare alternativ när det kommer till serverkostnader. Detta är däremot inte relevant om man använder sig av en egen webbserver. PHPs popularitet var som högst i mitten av 2000-talet.

8.1 Ruby on Rails

Ruby on Rails är ett ramverk utvecklat av dansken David Heinemeier Hansson för programmeringsspråket Rails.

Ramverket som gjordes till open source år 2004 väckte snabbt ett intresse bland utvecklare på grund av snabb utvecklingshastighet.

RoRs prioriteringar har alltid varit ”convention over configuration” (Ruby on Rails, 2011), vilket innebär att återanvända färdig kod för att kunna utveckla snabbare och att inte behöva skriva om befintlig kod.

RoR har bland annat kritiserats för en lång inlärningströskel och liksom PHP dålig prestanda.

8.2 ASP.NET

ASP.NET är ett system för att skapa dynamiska webbsidor utvecklat av Microsoft. ASP.NET baserar sig på .NET ramverket. Ramverket är komponent- och eventbaserat och bygger på egentliga programmeringsspråk istället för skriptspråk.

ASP.NET applikationer kan skrivas i C#, Visual Basic.NET, J# och Ruby. Applikationen kompileras alltid före användning vilket i flera medför en stor prestandaskillnad jämfört med konkurrenterna.

8.3 Python

Till programmeringsspråket Python finns ett antal populära ramverk för webbutveckling, av vilket Django har väckt stort intresse.

Django utvecklades ursprungligen för nyhetssidor och genererar automatiskt ett administrationsgränssnitt (Django, 2011).

9 Avslutning

PHP är ett av de populäraste programmeringsspråken för webbutveckling.

Det finns som redan nämnt flera alternativ, av vilka alla har sina fördelar och nackdelar.

Stora webbplatser med höga krav beträffande funktionalitet och prestanda brukar inte begränsa sig till ett alternativ. Vanligt alternativ är att till exempel programmera användargränssnittet (webbsidorna) i PHP och sköta bakgrundsoperationer som ofta är tyngre med program skrivna i till exempel C eller Python.

Som uttryckligen utvecklat för ändamålet, med kort inlärningströskel och bekanta syntax utgör PHP utan vidare ett gott alternativ vid skapandet av en webbapplikation.

Litteraturförteckning

Alexa. (den 2 April 2011). *Alexa Top 500 Global Sites*. Hämtat från http://www.alexa.com den 2 April 2011

Django. (2011). *Django | The Web framework for perfectionists with deadlines*. Hämtat från http://www.djangoproject.com/ den 4 April 2011

Golemon, S. (2006). *Extending and Embedding PHP.* Sams.

HipHop Team. (den 8 Oktober 2010). *HipHop for PHP Wiki*. Hämtat från Github: https://github.com/facebook/hiphop-php/wiki/ den 2 Mars 2011

Hudson, P. (2006). *PHP in a Nutshell.* Sebastopol: O'Reilly Media, inc.

Hudson, P. (2006). *PHP in a Nutshell.* Sebastopol: O'Reilly Media, inc.

Pear. (den 26 3 2011). Hämtat från PEAR - PHP Extension and Application Repository: http://pear.php.net/ den 26 3 2011

PHP. (den 4 3 2011). *PHP at the Core: A Hacker's Guide to the Zend Engine.* Hämtat från http://fi2.php.net/internals2 den 26 3 2011

PHP. (den 4 3 2011). *PHP: History of PHP - Manual.* Hämtat från PHP: http://fi2.php.net/manual/en/history.php.php den 26 3 2011

Purer, K. (den 18 7 2009). *PHP vs. Python vs. Ruby { The web scripting.* Hämtat från https://klausi.fsinf.at/sites/klausi.fsinf.at/files/php-vs-python-vs-ruby.pdf den 29 3 2011

Ruby on Rails. (2011). *Ruby on Rails*. Hämtat från http://rubyonrails.org/ den 4 April 2011

ShklarLeon, & RosenRich. (2008). Web Application Architecture, Second Edition. Glasgow: Bell and Bain.

TIOBE Software. (Mars 2011). *TIOBE Programming Community Index for March 2011*. Hämtat från TIOBE Software: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html den 1 April 2011

Wikipedia. (den 24 3 2011). *PHP - Wikipedia, the free encyclopedia.* Hämtat från Wikipedia: http://en.wikipedia.org/wiki/PHP den 2011 3 24