Lektion 1 Webbserver och PHP

Innehållsförteckning

Översikt

Som vi tidigare lär oss så är HTML ett notations språk för att skapa dokument för att utnyttjas bl.a på WWW-servrar för att publicera information. I början så användes endast HTML dokument. Informationen blev då statisk, och man var tvungen att manuellt redigera dokumenten för att ändra/uppdatera informationen. Behovet att dynamiskt kunna påverka innehållet i informationen uppstod tidigt. Att skriva ut en tabell med innehåll som läses från en databas är ett sådant exempel. Det man då gjorde var att skapa standard för CGI (Common Gateway Interface)

CGI är ett protokoll för hur en webbserver kan köra program med argument och hur dessa anropas från en webbläsare via HTTP. CGI är alltså inte ett eget programspråk. I början skrevs CGI program t.ex. i C/C++, Perl. Det är dock ganska så omständligt att administrera en sådan webbapplikation då innehåll ökar i omfång.

Alternativ lösning till CGI blev att skapa *”Hypertext Preprocessor”*, dvs skriptspråk som exekveras på webbservern för att skapa dynamisk HTML kod. Exempel på skriptspråk är PHP, ASP (Active Server Pages) från Microsoft. Mer om PHP kommer nedan.

Denna lektion omfattar:

- Apache server
- PHP
- Formulär
- Cookies
- Sessions

Mål

Målet med avsnittet är att ni ska få kunskap och kännedom om Apache HTTP server.

Att ni ska kunna grundläggande phpkonstruktioner såsom deklarationer , datatyperna skalär, array och hash, styrsatser, funktionsdeklarationer, uppdelning av phpapplikationer i olika filer.

Att ni ska kunna utnyttja fomulär för att skicka data till en webbapplikation.

Att ni ska känna till skillnaderna mellan GET och POST.

Att ni ska förstå när det är lämpligt att använda POST. När man bör använda GET respektive POST.

Att ni ska förstå hur formulärdata representeras i GET (URLEnconding) och POST (attr. enctype).

Att ni ska förstå hur cookies distribueras mellan webbläsare och målwebbplats.

Att ni ska förstå vad en session är och hur man utnyttjar den för att lagra information under en session.

Läsanvisningar

Utgåva 5 Kap 17 Web Servers (IIS and Apache) (s637-648), Skumma igenom 17.7 då vi inte kommer att gå igenom IIS server i denna kurs.

Utgåva 5 Kap 19 PHP (s696-739) Kapitlet är kortfattat och avsedd som introduktion. Ni kommer att behöva referensmaterialet på http://php.net/ för att kunna nyttja php fullt ut. I fördjupningslitteratur hittar ni länkar till två ”Open Source” böcker om PHP som går att utnyttja som hjälp också.

Fördjupningslitteratur

Bra som nybörjarbok PHP Reference: Beginner to Intermediate PHP5

Mera avancerad bok för den erfarne PHP programmeraren, Kapitel 2 är dock bra läsning för grunder i PHP 5. PHP 5 Power Programming

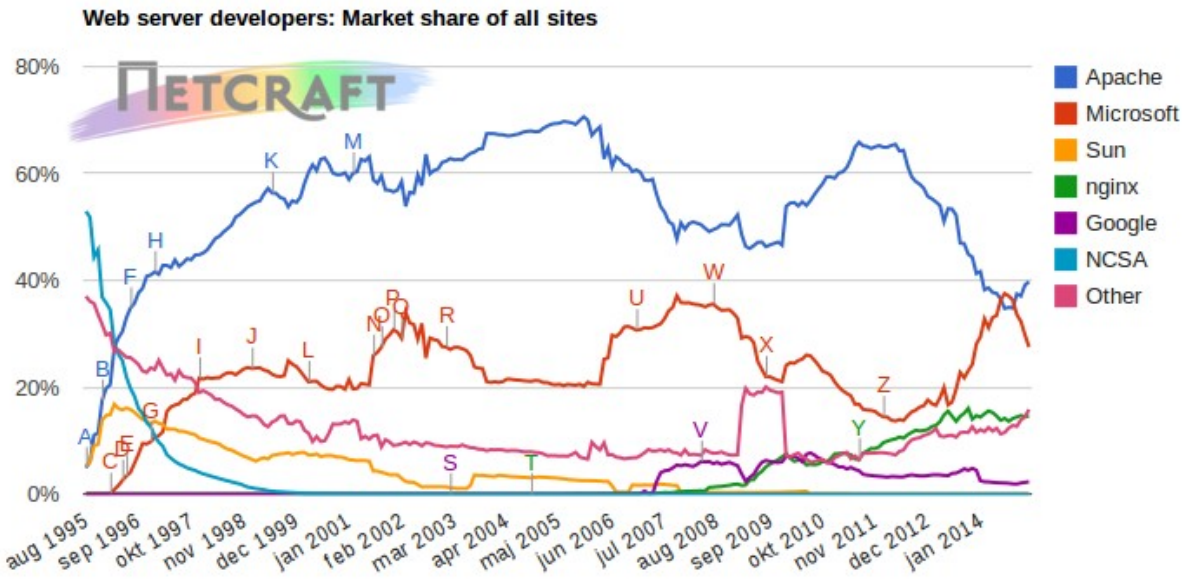
Det finns också en bok som omfattar PHP och PostgreSQL server som vi kommer att använda senare i kursen. Den boken kan ni hitta som pdf på nätet om ni söker på Google med följande sträng. ”Beginning PHP and PostgreSQL 8 pdf it-ebooks.info”. Den har några år på nacken men går bra att använda när man vill lära sig grunderna.

Innehåll

Apache server

I denna kurs kommer vi enbart att jobba med Apache HTTP server. Varför Apache HTTP server? Därför att Apache HTTP server är störst, den är världens mest använda server. Den är grattis att använda och källkoden är släppt som open source. Apache HTTP server utvecklas av The Apache Software Foundation. Några av anledningarna till att Apache HTTP server är störst är att den har ett väldigt stort stöd för olika scriptspråk, samt att den går att köra på många olika plattformar.

Nedan statistik över marknadsandelar för olika HTTP servrar taget från http://news.netcraft.com/



Developer	December 2014	Percent	January 2015	Percent	Change
Apache	358,159,405	39.11%	348,460,753	39.74%	0.63
Microsoft	272,967,294	29.81%	241,276,347	27.52%	-2.29
nginx	132,467,763	14.47%	128,083,920	14.61%	0.14
Google	20,011,260	2.19%	20,209,649	2.30%	0.12

Det finns många sätt att mäta andelar för HTTP servrar på internet, men gemensamt för nästa alla oberoende mätningar är att Apache HTTP server är störst på internet, med över 50% andel.
Värt att notera, om ni kollar på http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html

Den server som växer mest de senaste åren är nginx. Nginx är en HTTP server som är värt att lägga på minnet för framtiden, men det är ett helt annat kapitel. Läs mer om nginx på http://wiki.nginx.org

PHP

PHP (*PHP: Hypertext Preprocessor)* är ett skriptspråk som utvecklades av Rasmus Lerdorf 1995. (PHP History) Språket är ett mycket populärt språk för att skapa dynamiska sidor på internet.

En statisk HTML sida skickas så som den är skriven från servern till klienten (filer med .htm och .html ändelse). En dynamisk sida bearbetas på servern innan den den skickas till klienten. När en klient begär en PHP sida från servern så kommer den sidan att köras genom en PHP tolk. Tolken kommer att exekvera den kod som finns och producera en textström som utdata. Denna ström skickas då till klienten, oftas då i form av HTML kod.

En PHP sida är oftast en blandning av HTML kod och PHP kod. Se nedan på exempel.

```
<?php
$page_title = 'My first Hello World in PHP';
?>
<!DOCTYPE html>
<html lang="en"”>
  <head>
    <meta charset="utf-8">
    <title><?php echo $page_title; ?></title>
  </head>
  <body>
    <p><?php echo ”Hello World”; ?></p>
  </body>
</html>
```

Lathund för språkreferensen på php.net.

- Introduktion Hello World(php.net)
- Introduktion php script(php.net)
- Datatyper. Samtliga avsnitt under länken innehåll är av vikt, ett undantag är -Objects om du är obekant med OOP. Avsnittet Arrays är centralt och kommer att underlätta dina serverscript om du lär diog att använda arrayer och associativa arrayer(hashtables).
- Variabler. Värt att notera är avsnittet om scope. php är tämligen ensamt att vara så frikostig med synligheten hos sina variabler. En variabel som deklareras i ett script, är synlig i alla filer som detta script inkluderar. Funktioner ser inte "globala" variabler om de inte explicit deklareras i funktionen. Undantag är sk superglobals ex $_POST. Tänk på att php och perl skiljer sig väsentligen från varandra vad gäller $ symbolen. Php använder $ för att uttrycka en variabel, inte nödvändigtvis en skalär, och har ingen motsvaighet till perls filhandle, array, hash etc.
- Uttryck. Snarlikt många andra språk, kan läsas kursivt.
- Operatorer. Snarlikt många andra språk. Värt att titta extra på är avsnittet om arrayoperatorer och stringoperatorer.
- Styrsatser. Du märker snabbt att styrsatserna fungerar på samma sätt som i de flesta andra språk. Se särskilt på require och include-varianterna. Notera också att en styrsats kan vara utberdd över flera %lt;?php avsnitt.
- Funktioner. php kräver ingen strikt signatur (argument och returtyper) av deklarationerna utan kan returnera och ta emot olika datatyper, beroende på sammanhang. Det är dock sällan som beteendet är önskvärt.

Formulär

Som en första ansats att konstruera meningsfulla webbapplikationer ser vi först till att förstå den mekanism som förmedlar data mellan användargränssnittet (webbbläsaren/webbsidan) och webbapplikationen. Webbläsaren och webbapplikationen körs på två separata datorer, därför behövs ett protokoll för överföringen av data.

Get och POST, när var hur?

Förutsätter att du är bekant med de semantiska skillnaderna.

Det enkla svaret är att:

- Använd GET om din sida är meningsfull att lägga som ett bokmärke(Favorit/Shortcut) i webbläsaren.

Svaret är dock med en del förbehåll:

- För det första ska det vara samma sida (dock inte nödvändigtvis statisk) som visas varje gång en GET ställs, webbläsare cachar GET-sidor. Betrakta ett forum där man kan läsa inlägg på separata sidor. Om man läser ett inlägg och gör ett bokmärke på sidan. Vid nästa besök (via bokmärket) ska man då ta sig till det meddelande man tidigare läste. Om detta inte är möjligt (som regel fullt möjligt i ett forum) ska man inte använda GET.
- För det andra finns en begränsning i antalet tecken som GET (URLfrågan) kan innehålla. Den ligger kring något tusental tecken. Detta kan tyckas mycket, men om man inte skickar ren ascii text kommer tecknen i frågan att url kodas och därmed växa dramatiskt i storlek.
- För det tredje finns det gott om information som användaren inte ska komma i kontakt med via urlraden. Typiskt är lösenord och sessionsdetaljer (sessions ID), och andra saker som är olämpliga att lagra i urlraden. På samma sätt måste man tänka på att samtliga urlrader som visas på webbplatsen är potentiella länkar i massdistribuerade mail. Det finns flera exempel på mail som cirkulerat, innehållande sessionsid till webbplatser där sessionen varit en mer eller mindre oskyldigt inloggad användare (Sök: session hijacking). Nuförtiden är sessionsskyddet i php förbättrat.

Det finns uppenbara tillfällen då post är ett idealt sätt. Dessa finns uppräknade i rfc2616 under avsnittet POST. Typscenario är då man inte bör skicka samma fråga två gånger. Exempelvis vid foruminlägg, databasinsättningar, biljettbeställningar osv.

Cookies

Eftersom HTTP protokollet är ett stateless protokoll. Det innebär att information inte sparas efter det att en förfrågan avslutats. Med en förfrågan menas i detta fall en förfrågan från klienten, HTTP request och ett svar från servern, HTTP response. Ett sätta att bibehålla information mellan förfrågningar är att spara en Cookie hos klienten med information. Det är servern som skickar ut ett önskemål om att spara en cookie hos klienten med information. Informationen sparas då hos klienten (under förutsättning att klienten accepterar att servern får spara en cookie). Varje gång som en ny förfrågan sker till samma server (samma domän) så kommer innehållet i cookien att skickas med i HTTP request. På server sidan kan man då extrahera denna information.

I PHP använder man funktionerna `setcookie()`. Viktigt att tänka på som det står i

dokumentationen är att `setcookie()` måste anropas innan någon text skickas till klienten. Detta pga att cookies är en del av HTTP header.

Informationen i en cookie finner man i det globala PHP objektet `$_COOKIE`.

Sessions

Som vi tidigare nämnt så är HTTP protkollet *stateless*. Hur gör man då om man vill komma ihåg information utan att behöva spara det i en cookie hos klienten? T.ex. om jag behöver logga in på en sida hur skall jag då komma ihåg att besökaren är inloggad?

I PHP så finns en funktion för att använda något som kallas för SESSION.

När använda sessions?

När man vill spara information på servern för en klient under en viss tidsrymd då den gör förfrågningar. Detta sker genom att servern sparar en cookie hos klienten med ett session_id. På servern finns då en mekanism att kunna spara information och koppla ihop det med ett session_id. Om inte klienten stödjer cookies kan session_id skickas med i URL.

I PHP lagras sessions variabler oftast i temporära filer på servern. En session har en förbestämd livslängd (som går att konfigurera i php konfigurationen). När tiden har förlupit kommer den information som sparats i en sessions variabel att raderas.

Det går att konfigurer PHP att ha sessions att starta automatiskt för alla HTTP request. Om så inte är gjort så måste man i börja på ett PHP skript tala om att man vill använda sessions. Detta görs genom att anropa funktionen `session_start()`. För att spara sessions variabler används funktionen `session_register()`. Sessions variabler kommer man åt genom det globala PHP objektet `$_SESSION`.

Länkar