Prova på-laboration i PHP

Johan Sjöholm
johsj@ida.liu.se
Institutionen för datavetenskap, Linköpings universitet
2009-08-09

1. Introduktion till webbprogrammering

Webbprogrammering består av ett antal olika områden. Till skillnad från exempelvis imperativ programmering är inte webbprogrammering en speciell programmeringsparadigm utan snarare en speciell tillämpning. Man pratar inom webbprogrammering mycket om server-side och client-side. Dessa begrepp är mycket viktiga och att veta vad man gör på respektive sida är vitalt för en programmerare som arbetar med webbapplikationer.

Saker som görs client-side är saker som kan köras på den enskilda klientdatorn, layout och grafik är exempel på sådant som ofta görs på klientdatorn, lite förenklat kan man säga att klientdatorn laddar ner ett litet program som gör saker direkt på datorn. Saker som däremot görs server-side är saker som helt och hållet görs på servern, som t.ex. att kommunicera med en databas eller hämta nya webbsidor. När en klientdator vill göra något server-side skickar den en instruktion till webbservern som utför arbetet och sedan levererar resultatet till klientdatorn.

Det är här som PHP-kommer in. PHP står för PHP: Hypertext Preprocessor och är ett av de populäraste språken inom server-side-programmering. Som namnet antyder används det som en preprocessor (ett program vars enda syfte är att producera input till ett annat program) som formaterar och ändrar HTML-kod innan den skickas till klientdatorn.

Vad PHP gör är alltså att mer eller mindre generera HTML-kod. I den här labben förutsätts därför att du har viss erfarenhet av HTML och känner till grundläggande begrepp som taggar och attribut, en kort introduktion finns i Introduktion till IDA:s datorsystem i STONE.

2. Exempel på PHP-program

Ett PHP-program består i princip av HTML-kod med HTML-genererande PHP-kod insprängt med hjälp av en speciell PHP-tag.

```
<html>
<head><title>Example</title></head>
<body>
<!-- Här är en HTML-kommentar -->
<?php
 //Här är en PHP-kommentar
 echo "<p>Detta är ett exempel.</p>"
?>
</body>
</html>
```

Ovanstående PHP-kod är mycket enkel och ur ett rent praktiskt perpektiv hade det varit bättre med ren HTML då PHP-koden är statisk och alltid gör samma sak. Funktionen `echo` skriver helt enkelt ut en HTML-snutt. Däremot visar det tydligt hur en typisk PHP-fil brukar se ut med en blandning av HTML och PHP. Den visar också en kommentar i HTML-koden och en kommentar i PHP-koden.

Nedan har vi innehållet i filen hej.php:

```
<html>
<body>

<?php
if (isset($_POST["fnamn"]) && isset($_POST["enamn"])) {
 echo "<p>Hej, " . $_POST["fnamn"] . " " . $_POST["enamn"]  . "!</p>";
}
?>

<form action="hej.php" method="POST">
Förnamn: <input type="text" name="fnamn" />
Efternamn: <input type="text" name="enamn" />
<input type="submit" />
</form>

</body>
</html>
```

Ovan har vi ett mer avancerat program som använder ett formulär och metoden POST. Det finns också en annan formulärs-metod som heter GET men skillnaderna är inte relevanta för den här labben så POST är vad vi använder.

När man använder ett formulär sparas datan man matar in undan i en *associationslista* och den skickas sedan till den PHP-sida som står efter action-attributet i formuläret, i det här fallet är det filen själv, `hej.php`. För metoden POST heter den associationslistan `$_POST`. I exemplet ovan sparas ett för- och ett efternamn undan i `$_POST`.

En associationslista är en lista med nyckel-värde-par där man får fram ett visst värde genom att använda dess nyckel. Man kan t.ex. tänka sig en associationslista med par av namn som nyckel och ålder som värde. Då kan man exempelvis ha namnet ”Adam” knutet till åldern 34. För att få fram personen Adams namn skriver man `associationslista["Adam"]`. För att skapa en egen associationslista skriver man `array(nyckel1=>värde1, nyckel2=>värde2)`. För att lägga till ett nyckel-värde-par i en associationslista skriver man `associationslista[nynyckel] = nyttvärde`.

Associationslistevariabeln `$_POST` behöver man dock inte skapa själv utan är inbyggd och fylls alltså med de nyckel-värde-par som bildats av att anropa ett formulär med POST-metoden. Nycklar blir det som står efter `name` i `input`-taggarna och värden blir det som matas in i input-fälten. I det här fallet är det textfält och värdena blir då strängar.

När sidan först visas undersöker funktionen `isset()` om `$_POST` innehåller nycklarna `"fnamn"` och `"enamn"` (symbolen && betyder ”och” vilket innebär att både uttrycket före och uttrycket efter måste vara sant för att `if`-satsen ska vara sann) och eftersom de inte gör det händer inget mer i PHP-koden. Sedan producerar HTML-koden ett formulär. När man sedan fyller i formuläret och klickar på ”Submit” anropas filen igen men nu med nycklar och värden från formuläret sparade i `$_POST`. Eftersom `isset()` nu kommer hitta nycklar och värden kommer koden i if-satsen att köras. Punkterna på rad 6 efter `echo` är så kallade konkateneringsoperatorer och används för att slå ihop strängar.

Sedan producerar HTML-koden åter samma formulär och man kan göra om det hela med nya värden.

3. Att använda PHP

PHP är ett *interpreterat* språk som alltså inte behöver kompileras innan körning. Däremot måste filerna med koden finnas på rätt ställe för att webb-servern ska kunna göra sitt jobb.

På IDA har alla studenter en mapp i sin hemkatalog vid namn `www-pub`. I den kan man lägga material som man vill ska vara nåbart via webben, exempelvis PHP-sidor.

För att testa om koden i filen fungerar som tänkt går man till `http://www-und.ida.liu.se/~abcde123/`*`filnamn`* där `abcde123` är det egna användar-namnet. Att få PHP att fungera kan vara lite tricktigt men på IDA är allt redan förkonfigurerat och det är bara att börja lägga PHP-filer i sin `www-pub`-katalog.

4. Övningar

Övning 1 – Få igång PHP

Skapa filen `index.php` i mappen `www-pub` i din hemkatalog. I filen ska texten ”Hello, World!” skrivas ut som rubrik.

Gå sedan in på www-und.ida.liu.se/~abcde123 (där abcd123 ska vara ditt användarnamn på IDA) och kontrollerar att det fungerar.

Skapa sedan filen hej.php i samma mapp med innehållet från exemplet ovan och testa att det fungerar genom att gå till www-und.ida.liu.se/~abcde123/hej.php och testa formuläret.

Om inget annat står ska resterande övningar bygga ut filen `index.php`.

Övning 2 – Att få hjälp

PHP är ett språk anpassat för webben och det är på webben mest hjälp finns att få om man kör fast. Använd webben för att ta reda på hur funktionen `date()` fungerar i PHP.

Fortsätt sedan bygga på `index.php` med texten ”Is it Friday?” och sedan en if-sats som på nästa rad skriver ut ”Yes, it is Friday!” om det är fredag och ”No, it is not Friday.” om det inte är fredag.

Övning 3 – Funktioner

I PHP, precis som i andra programmeringsspråk, är funktioner en grundläggande konstruktion. Skapa en funktion som tar ett namn som argument och skriver ut “Välkommen *namn*!” och anropa den sedan med några olika namn.

Använd webben för att ta reda på hur man skapar funktioner i PHP.

Övning 4 – Arrayer och stränghantering

För att skapa en array gör man på samma sätt som att skapa en associationslista, med den skillnaden att man inte lägger in kommaseparerade nyckel-värdepar utan bara kommaseparerade värden.

För att iterera genom en array kan man använda iteratorn foreach. Förutsatt att man har arrayen `$a` kan man iterera igenom den genom att skriva

```
foreach ($a as &$element) {
  //Gör saker med varje element i arrayen
}
```

I exemplet ovan kommer variabeln `$element` för varje iteration innehålla det element i arrayen som man just nu opererar på.

Skapa en array som innehåller namnen ”Adam Andersson”, ”Berit Bengtsson” och ”Christer Carlsson”. Generera sedan en HTML-sida där en tabell med dessa namn visas upp, ett per rad. Namnen ska hämtas genom att iterera igenom arrayen och konkatenera strängarna där med strängar av HTML-kod.

Tips: För att se hur tabeller fungerar kan man kolla i STONE.

Övning 5 – Formulär och associationslistor

Skapa ett formulär som tar en veckodag och hämtar ut en städansvarig och skriver ut på sidan. Veckodagarna och dess städansvarige ska lagras i en associationslista.

*Övning 6 – Inloggning och sessioner

På mer avancerade webbsidor kan man vilja lagra information om en användare mellan flera sidor, detta är t.ex. en förutsättning för alla sidor med inloggning. I PHP använder en speciell associationslista som heter `$_SESSION` för att hantera detta. Associationslistan `$_SESSION` är precis som `$_POST` inbyggd men till skillnad från `$_POST` är den inte tillgänglig hur som helst.

För att man ska kunna använda `$_SESSION` måste man först i filen, före all annan PHP- och HTML-kod, anropa funktionen `session_start()`. Då kommer filen sessionhanteras och `$_SESSION` vara tillgänglig. På IDA måste man också, redan innan man anropar `session_start()` informera om var `$_SESSION` ska lagras, det gör man genom att anropa funktionen `session_save_path()` med en passande sökväg, t.ex. `session_save_path(“/home/abcde123/tmp”)`.

Skapa ett formulär som tar ett användarnamn och ett lösenord och kollar det mot en associationslista med användarnamn och lösenord som du skapar. Om namnet och lösenordet finns och stämmer överens med varandra ska användarnamnet

läggas in i `$_SESSION`. Sedan ska en ny PHP-sida som hanterar sessioner anropas där texten "Välkommen *användarnamn*!" skrivas ut.

Tips: För att från PHP-gå till en annan sida använder man funktionsanropet `header('Location: annansida.php')` där `annansida.php` ska vara den sida man vill gå till.

*Övning 7 – PHP och MySQL

PHP är mycket populärt att använda tillsammans med databassystemet MySQL, bland annat i webbservrar av så kallad LAMP-modell (Linux, Apache, MySQL and PHP). Därför har PHP ett antal inbyggda funktioner för att hantera kommunikationen med MySQL-databaser.

För att ansluta till en MySQL-server använder man sig av funktionen `mysql_connect` som tar tre strängar som argument, adressen till en MySQL-server, ens egna användarnamn på databasen samt ens lösenord.

När man anslutit sig till databassystemet måste man välja vilken databas på servern man vill använda sig av, det gör man med hjälp av `mysql_select_db()` som tar en sträng med namnet på en databas som argument.

När detta är gjort skickar man SQL-frågor till databasen med funktionen `mysql_query()` som tar en sträng innehållande en SQL-fråga som argument och skickar frågan till den databas man valt. Sådana frågor genereras ofta dynamiskt med hjälp av formulär.

För att sedan stänga uppkopplingen mot databasen igen använder man `mysql_close()`.

Ofta kan det vara viktigt att kunna byta ut den databas och det databassystem man arbetar mot, därför är det viktigt att inte använda mysql-funktioner direkt i koden. Skapa en funktion `query_database()` som tar en sträng innehållande en SQL-fråga och ansluter till en databas, ställer frågan till den, stänger databasen och returnerar svaret. Om man en dag vill byta databassystem, eller adressen till databasen, eller något annat liknande, ska man bara behöva ändra i den funktionen.

5. Referenser

Framtida kurser

Programspråket PHP återkommer bland annat i kursen *Webbprogrammering och Interaktivitet* för IP, C- och D-programmen och *Avancerad webb-programmering* för C- och D-programmen.

Litteratur

PHP är ett språk för webbprogrammering och det bästa stället att få hjälp med PHP är just på webben. En av de bättre nybörjarsiterna för PHP är www.w3schools.com, ett svenskt alternativ är www.webdesignskolan.se.

Vill man ha en bok kan det vara svårt att hitta en bra bok specifikt för PHP men boken *Internet & World Wide Web: How To Program* tar upp PHP tillsammans med mycket annat inom webbprogrammering. Språket omnämns också som hastigast i Concepts of Programming Languages.

6. Frågor att fundera över

För dig som är nybörjare på programmering

Programmering är en process som består av flera faser. Först måste man förstå vad det är för problem som man ska lösa. Därefter ska man försöka designa en lösning. Denna lösning ska *implementeras*, dvs man ska skriva själva programmet. Sist men inte minst måste det färdiga programmet testas. Oftast går dock inte processen så här rakt och tydligt. Många gånger kan man tjäna på att experimentera lite, utan att ha förstått själva problemet. En del hävdar till och med att det är först när man har utformat lösningen som man verkligen har förstått problemet. För att bli en bra programmerare krävs lång träning – mycket längre än vad ens en högskoleutbildning kan ge. Man måste lära känna de olika byggstenarna som finns i programspråket, men också lära sig när och hur man ska använda dem.

Hur gjorde du när du löste övningarna i den här laborationen? Kan du känna igen de steg som beskrivs ovan? Känner du att du har förstått åtminstone lite grann av vilka byggstenar som finns i PHP? Kändes det svårt att försöka formulera lösningar på problem i ett formellt programmeringsspråk?

För dig som programmerat en del förut

Liknar PHP något annat språk som du har erfarenhet av? Vad känns svårare eller lättare att göra i PHP, enligt din bedömning?