TNMK30 Lab 4: Blogg med databas

Niklas Rönnberg, Stefan Gustavson, ITN-LiTH 2017

I förra labben gjorde du en enkel dynamisk blogg där posterna lades i en textfil som ändrades av PHP. Det finns flera nackdelar med att göra dynamiskt innehåll på det sättet. Dels måste man ge webservern skrivrättigheter i sin egen filkatalog, vilket inte bara är krångligt utan också kan innebära en säkerhetsrisk. Det blir dessutom svårt att sortera posterna och göra ett urval. I så fall måste man skriva kod som läser hela textfilen och tolkar vad det står i den, samt bara visar vissa delar.

En bättre lösning på alla sätt är att lägga posterna i en databas. Dels lägger man det dynamiska innehållet på en server som är gjord för att hantera uppdateringar av data på ett säkert och strukturerat sätt, dels blir det mycket enklare att söka, sortera och göra urval i data och att variera presentationen på olika sätt. Man kan till exempel visa de nyaste posterna först i stället för sist (vilket är det mest logiska för en blogg), man kan visa bara de senaste posterna, bara dem som är skrivna av en viss person, bara dem som innehåller ordet ”potatis”, som är skrivna på måndagar och är kortare än 100 tecken, för att bara nämna några möjligheter med sökningen. Urval görs smidigt genom att man ställer en specifik fråga i SQL till databasen, vilket gör att PHP-koden blir enkel och rättfram.

Databasen kommer också att vara tillgänglig från annat håll och för andra syften om man vill dela data med andra tillämpningar. En och samma databasserver kan sköta databaserna för ett stort antal webbtjänster, förutsatt att det inte är alltför hög trafik till dem.

Databas, användare, lösenord och tabellnamn

I den här kursen, och i några andra kurser på MT-programmet, används databasservern ”mysql.itn.liu.se”. Om du vill göra mer avancerade egna experiment kan du be om en egen användare och en egen databas på den servern. Kontakta examinator i så fall. Problemet med att starta från noll är att man måste skapa sina tabeller själv, och det är litet pilligt innan man lärt sig SQL ordentligt. För den här labben har vi därför gjort det litet enklare att komma igång genom att i den redan existerande databasen ”blog” skapa en tabell åt varje student i kursen. ”Din” tabell har samma namn som ditt student-ID, dvs fem bokstäver och tre siffror. Användaren som får läsa data heter också ”blog”, utan lösenord. För att skriva, uppdatera och radera poster (SQL-satserna INSERT, UPDATE, DELETE) behöver du i stället ange användaren ”blog_edit” och lösenordet ”bloggotyp”.

Notera alltså att alla tabeller för denna laboration har samma användare och samma lösenord, vilket är en klar säkerhetsbrist. Det är till exempel väldigt lätt att sabotera andras tabeller, men gör inte det är ni snälla.

Alla tabeller i databasen ”blog” är skapade enligt följande med SQL, men med ”kajsa123” utbytt mot kursdeltagarnas student-ID:

```
create table kajsa123 (
  entry_date timestamp,
  entry_author  varchar(100),
  entry_heading varchar(100),
  entry_text text);
```

Det finns dessutom en tabell som heter ”blog”, där det redan finns några testposter inlagda. Använd gärna den för att testa, men byt till er egen tabell när ni gör labuppgiften.

Hämta data

För att hämta ut data ur databasen kopplar man upp sig mot databasservern och ställer en fråga i SQL, och sedan läser man en post i taget i resultatet. Det går till så här i PHP:

```
<?php
 // Koppla upp mot databasen
 $connection = mysqli_connect("mysql.itn.liu.se","blog",
                      "", "blog");
 // Ställ frågan
 $result = mysqli_query($connection, "SELECT * FROM kajsa123");
 // Skriv ut alla poster i svaret
 while ($row = mysqli_fetch_array($result)) {
  $heading = $row['entry_heading'];
  print("<h2>$heading</h2>\n");
  $author = $row['entry_author'];
  $date = $row['entry_date'];
  print("<p>$author, $date</p >\n");
  $text = $row['entry_text'];
  print("<p>$text</p>\n");
  print("<hr/>");
 } // end while
?>
```

Tabellen i exemplet ovan (”kajsa123”) finns inte, utan du ska byta ut det namnet mot ditt eget student-ID. Tabellen ”blog” innehåller några få bloggposter som du kan använda för att testa visningen innan du har några poster i din egen databas. **Ändra din bloggsida från den förra laborationen så att den visar posterna från databasen i stället för från textfilen!**

Lagra data

Innan man kan visa några data måste man förstås lagra något i databasen. Det gör man också med en SQL-fråga, men frågan ska i det här fallet byggas upp som en sträng med data från formuläret. Eftersom den här kursen inte i första hand handlar om att lära er SQL så ger vi utseendet på frågan nedan. Notera att fältet ”timestamp” sätts till NULL, vilket kan tyckas underligt, men ett fält av den typen beter sig på ett speciellt sätt: om man inte lägger något annat i fältet så sätts det till dagens datum och klockslag när posten skapas, vilket är precis det vi vill ha.

```
$author = $_POST['author'];
$heading = $_POST['heading'];
$text = $_POST['entry'];
$query = "INSERT INTO blog VALUES(NULL, '$author', '$heading', '$text')";
// Nu har vi en fråga i $query som vi kan skicka till MySQL!
mysqli_query($connection, $query);
```

Ändra din blogg så att du kan skapa inlägg i din egen databas. Formuläret för att skriva in data kan vara exakt detsamma som i förra laborationen, men PHP-koden som tar hand om data ska koppla upp sig mot databasen i stället för att skriva i en textfil. Observera att du har flera användare på din databas med olika rättigheter. För att lägga till och ändra i poster (SQL-satserna INSERT, UPDATE och DELETE) måste du som sagt ange användarnamnet ”blog_edit” och ange lösenordet ”bloggotyp”. För att visa posterna räcker det däremot med användaren ”blog” som bara får göra SELECT och inte behöver ange något lösenord. Det kan tyckas krångligt att ha olika användare för olika ändamål, men i större sammanhang är det mycket viktigt att ha koll på säkerheten och inte ge användare tillgång till funktioner som de inte behöver. Databaser är avsedda att kunna användas till betydligt större och mer komplexa tillämpningar än den här laborationen.

SQL injection

Här är det läge att nämna en viktig säkerhetsaspekt. Om man bygger ihop en SQL-fråga från formulärdata på det här sättet så kan en illvillig användare skriva in SQL-kod i fälten som gör det möjligt att göra en massa dumheter. Detta kallas för ”SQL injection” och är tyvärr en väldigt vanlig metod för attacker på webplatser. Även av misstag kan det bli problem med vissa tecken i formulärfälten, framför allt enkla citationstecken som avslutar strängar i SQL och måste särbehandlas. För att undvika sådana problem finns färdiga funktioner i PHP. För MySQL heter funktionen `mysqli_real_escape_string()` och används så här:

```
$author = mysqli_real_escape_string($connection, $_POST["author"]);
$heading = mysqli_real_escape_string($connection, $_POST["heading"]);
$text = mysqli_real_escape_string($connection, $_POST["entry"]);
```

Förhindra SQL injection genom att använda funktionen enligt ovan! Strängar som behandlats på detta sätt är säkra att inkludera i en SQL-sats.

Urval av data

Vår enkla databas innehåller bara en enda tabell, så det går inte att göra så mycket spännande med den, men det går i alla fall att sortera, söka och göra urval av vilka poster som visas. För att visa posterna i ordning från den nyaste till den äldsta kan du formulera SQL-frågan så här:

```
SELECT * FROM blog ORDER BY entry_date DESC
```

För att bara visa de tio senaste posterna gör man så här:

```
SELECT * FROM blog ORDER BY entry_date DESC LIMIT 10
```

För att bara visa poster som är skrivna av användaren ”Analogsyntare”:

```
SELECT * FROM blog WHERE entry_author = 'Analogsyntare'
```

För mer information om hur man sorterar och gör urval med frågor i SQL hänvisar vi till MySQL-dokumentationen på mysql.com eller någon SQL-tutorial, till exempel den på w3schools.com.

Förändra din bloggsida så att den visar posterna i ordning från den nyaste till den äldsta, och inte visar fler än 5 poster om det finns fler än så i databasen.

Olika vyer med GET

En viktig och central funktion för många dynamiska websidor är att sidan kan visas på olika sätt beroende på hur man anropar den. En webadress (URI) innehåller inte bara adressen till en sida, utan kan även innehålla extra information i form av *argument*. En webadress med argument kan till exempel se ut så här:

```
http://www.student.itn.liu.se/~abcde123/TNMK30/blog.php?order=desc&limit=10
```

Det som står efter frågetecknet är en lista med så kallade *key-value pairs*: nyckelord och värden med likhetstecken emellan, avgränsade med &-tecken. Detta översätts i PHP till en array `$_GET[]` med motsvarande index och värden, på samma sätt som data från ett formulär översätts till arrayen `$_POST[]`. Resultatet av ovanstående länk skulle bli att `$_GET['order']` får värdet `'desc'` och `$_GET['limit']` värdet `10`.

Förändra din kod så att man kan bestämma med GET-argument hur posterna ska visas! Argumenten som PHP-koden ska känna igen ska vara de som visas i exemplet ovan, och deras värde ska påverka SQL-frågan. Om endera eller båda argumenten saknar värde (om de är en tom sträng i PHP), eller om de har ett värde som är orimligt eller inte känns igen av din kod, så ska koden använda rimliga standardvärden. **Gör några fasta länkar på bloggsidan som anropar sidan själv med några olika varianter av argument**, till exempel ”visa alla poster”, ”visa de tio senaste posterna” och ”visa alla poster med de äldsta först”.

`blog.php?order=desc`	Visa alla poster med de nyaste först
`blog.php?order=asc`	Visa alla poster med de äldsta först
`blog.php?order=desc&limit=10`	Visa max 10 poster med de nyaste först

Funktioner som saknas

I en riktig blogg bör man naturligtvis ha en mer flexibel presentation av posterna. Besökaren bör kunna bläddra mellan flera sidor för att se alla poster även om inte alla visas samtidigt, och även söka själv efter de poster som är av intresse. Det går litet utanför vad vi hinner med i den här laborationen, men du har nu provat på alla grundläggande verktyg som behövs för att göra sådana funktioner, och ni kommer att få gott om tillfällen att tänka mer på navigering, presentation och användbarhet senare i kursen, när ni gör era projekt.

Redovisning

Redovisa din lösning genom att presentera all kod du skrivit. Det blir totalt tre filer: en HTML-fil med formuläret, en HTML-fil med PHP-kod för att spara nya inlägg till fil och en HTML-fil med PHP-kod för att visa de befintliga inläggen, sorterat på några olika sätt beroende på hur man anropar sidan. Ditt arbete med att passa in sidan i strukturen på din webbplats och att formatera den med CSS bör kunna återanvändas i sin helhet från lab 4 och behöver inte redovisas igen.