Projekttips

Variabelnamn, kolumnamn

Ett av de allra vanligaste problemen vid utveckling av PHP-projekt är att man inte har full koll på vad databaser, tabeller och kolumner heter. Därför ska man alltid när man jobbar med projektet ha en bok bredvid datorn där det tydligt framgår:

Alla tabell och kolumnnamn
Primärnycklar och främmande nycklar
ER-diagram

Välj kolumnnamn med omsorg! Använd gärna engelska namn för att undvika åäö. Om möjligt väljer man namn på främmande nycklar så att det framgår vilka primärnycklar de refererar till.

Namn på PHP-variabler bör döpas så att de överensstämmer så väl som möjligt med kolumnnamnen om de innehåller värden som hämtats eller ska in i en tabell. I exemplet ovan har både **album** och **track** kolumnerna **id** och **title**. Det räcker inte att döpa varialerna till **$id** och **$track** för man vet inte om de tillhör **album** eller **track**. Vi kan då döpa en variabel som innehåller ett värde från kolumnen **id** i tabellen album för **$album_id**.
Eftersom det lätt blir rörigt med flera tabeller, flera kolumner som heter likadant mm är det mycket viktigt att vara konsekvent.

Namn på fält i formulär bör ha samma namn som de PHP-varibler där värdena hamnar.

Exempel

Om man vill göra ett formulär för att mata in namnet på ett album, bör variabeln som tillfälligt lagrar värdet som ska in i tabellen döpas till $album_title. Inmatningsfältet bör då se ut så här:

```
<input type="text" name="album_title" value="">
```

Ett mycket vanligt fel är att man glömt om man döpte kolumnen med namnet på skivan till **title** eller till **name**, och råkar döpa PHP-variabeln till **$album_name**. Det är visserligen tillåtet att göra det, men leder troligtvis till att man skriver fel i sin SQL-sats när det är dags att peta in namnet i databasen. Det är därför man ska ha en uppslagen bok bredvid datorn där man kan se alla tabell- och kolumnnamn.

Citationstecken

Problem med citationstecken har tidigare nämnts i samband med SQL injection attacks, där man tex försöker göra intrång i en databas genom att infoga SQL-satser mellan citationstecken. Citationstecken ställer till problem även då det inte är onda bakomliggande orsaker.

Exempel

Antag att vi vill mata i låten *That's the way* i kolumen **name**, nej, jag menar, vad hette den nu, **title** i tabellen **track**. Om variabeln **$track_title="That's the way"** och vi försöker med följande:

```
$sql="INSERT INTO track (title) VALUES ('$track_title')"; får vi
följande SQL-sats:

INSERT INTO track (title) VALUES ('That's the way')          FEL
```

Detta är ogiltig SQL på grund av citationstecknet mitt inne i en sträng. Databashanteraren tror att strängen slutar efter That.
Lösningen är att använda *escape* dvs ersätta citationstecken med någon kombination av specialtecken. Standardsättet i SQL om man vill ha ett citationstecken inne i en sträng är att skriva två citationstecken i följd:

```
INSERT INTO track (title) VALUES ('That''s the way')  KORREKT
```

MySQL tillåter även att man använder \ som escape-tecken. Följande fungerar därför också:

```
INSERT INTO track (title) VALUES ('That\'s the way')  KORREKT (MySQL)
```

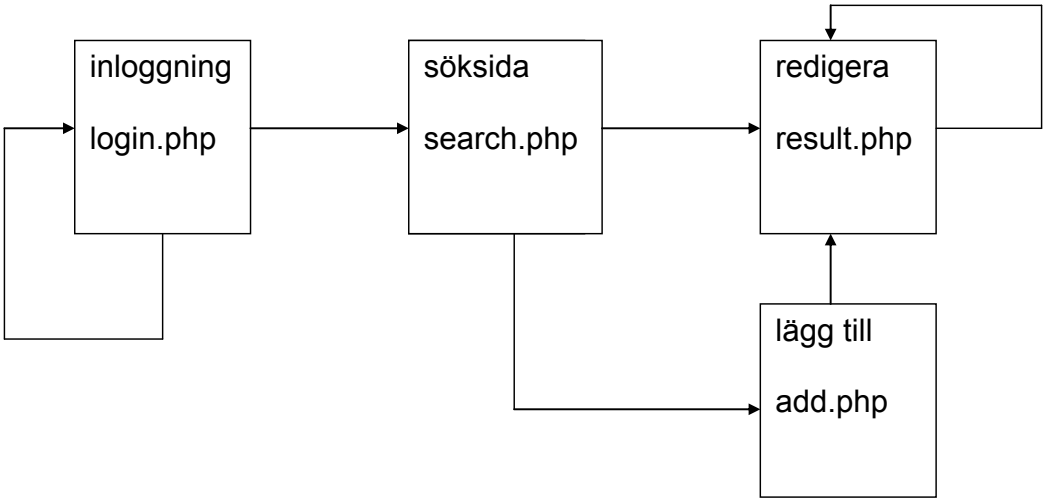
Detta fungerar dock inte med alla SQL-databashanterare.

Referensinformation om PHP och MySQL

Grundresurserna när man letar info om PHP och MySQL är www.php.net och www.mysql.com.

Flödesscheman

I en webbaserad databaslösning flyttar man sig ofta mellan olika sidor, till exemepel inloggningssidor, inmatningssidor, söksidor mm. Rita ett schema över hur de olika sidorna är länkade till varandra:



Testning

Använd många echo-satser i PHP-koden när ni utvecklar och testar, för att tex se att variabler har korrekta värden. Skriv i synnerhet ut SQL-satser:

```
$table="track";
$sql="SELECT * FROM $table";
//testutskrift som först skriver ut
//namnet på variabeln,
//sedan skrivs värdet på variabeln ut.
echo '$sql: '."$sql";
```

Utskriften blir:

$sql: SELECT * FROM track

Testa alltid med riktiga data! Det blir mycket lättare om man matar in rimliga data än om man matar in nonsenstexter.

Exempel

Antag att ni gjort ett formulär som ska skicka data till tabellen **catalog**:

namn	testing
nummer	lkjflköadjfölkj

Antag att man gjort fel i sin PHP-kod som lägger in detta i tabellen, så att kolumnerna förväxlas. Det blir då väldigt svårt att se:

catalog

name	**number**
lkjflköadjfölkj	testing

Om man istället testar med

namn	Sven Svensson
nummer	040-123456

och tittar in i tabellen ser man genast att det blivit fel:

catalog

name	number
040-123456	Sven Svensson

Inloggning

Inloggning krävs inte för projektet, men får gärna vara med.

Fritextsökning, fulltextsökning

När man söker i en databas vill man ofta söka i mer en ett fält. Man kan göra detta med OR i SQL-satser, men det blir otympligt om det är många fält.

```
SELECT registration FROM inventory WHERE
MAKE LIKE '%corvette%' OR
MODEL LILE '%corvette%' OR
DESCRIPTION LIKE '%CORVETTE%'
```

Förutom att SQL-koden blir otymplig om det är många fält att söka i, kan sökningarna också bli ineffektiva.

I MySQL-tabeller kan man definiera fulltext fält enligt följande exempel:

CREATE TABLE inventory (
registration CHAR(7) PRIMARY KEY NOT NULL,
make CHAR(50),
model CHAR(50),
description CHAR(255),
FULLTEXT (make, model, description))

Man kan nu söka i tabellen med följande:

SELECT registration FROM inventory
WHERE MATCH (make, model, description) AGAINST ('corvette')

Läs mer om fulltext på

http://devzone.zend.com/26/

Blandade tips

- Använd include-filer för PHP-kod som ofta återkommer: serverdata, funktioner, formulär som kan finnas med på olika sidor mm.
- Använd method="get" vid utveckling men method="post" i färdig applikation.

Method="get" gör att det man fyllt i i ett formulär syns i URL på en webbsida. Om man använder method="post" göms det man fyllt i.