

# *Databaskonstruktion*

*PHP*



# PHP

- Utveckla en applikatio
- PHP: Hypertext Preprocessor
- Serverside



# Hur

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8
9   Hej mitt namn är:
10  <?php
11    echo "Johan";
12  ?>
13
14 </body>
15 </html>
```

Hej mitt namn är: Johan

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8
9   Hej mitt namn är:
10  Johan
11 </body>
12 </html>
```

# Variabler i PHP

- Startar med ett dollartecken (\$)
- Deklareras när de får ett värde.
- Gör skillnad på gemener och versaler

```
Hej mitt namn är:
```

```
<?php
```

```
    $namn = "Johan";
```

```
    echo $namn;
```

```
?>
```

(måste starta med en bokstav eller understreck ( \_ ) och bara bestå av bokstäver, siffror och understreck)

- För att sätta ihop strängar/variabler används punkt

```
$namn = "Johan";
```

```
echo "Hej, mitt namn är " . $namn;
```



# Arrayer i php

"An array in PHP is actually an ordered map." - php.net

- Key -> value pairs
- Multidimensionella arrayer

```
$namn = array("Johan", "Henrik"); //initiera en array med 2 personer  
array_push($namn, "Marcus");      //lägga till en person till  
  
print_r($namn);
```

```
Array ( [0] => Johan [1] => Henrik [2] => Marcus )
```

```
$personer["Johan"]["age"] = 42;  
$personer["Johan"]["likes"] = "Burgers";  
$personer["Johan"]["electronics"] = "1kg";  
  
$personer["Henrik"]["age"] = 43;  
$personer["Henrik"]["likes"] = "Cheesedoodles";  
$personer["Johan"]["electronics"] = "341kg";  
  
print_r($personer);
```

```
Array  
(  
    [Johan] => Array  
        (  
            [age] => 42  
            [likes] => Burgers  
            [electronics] => 341kg  
        )  
    [Henrik] => Array  
        (  
            [age] => 43  
            [likes] => Cheesedoodles  
        )  
)
```

# Iteration med hjälp av PHP

- While
- For
- ...
- foreach

```
$aNumber = 3;

while($aNumber < 5){
    //do stuff
    $aNumber++;
}
```

```
for($i=0;$i<5;$i++){
    //do stuff 5 times
}
```

```
foreach($anArrayWithValues AS $key => $value){
    // loops through the array and foreach row sets
    // the key's value into the first parameter ($key)
    // and the value of that key into second parameter ($value)
}
```

# Foreach VS For med en array där nycklarna inte är 0-xx

```
$personer[0] = "Marcus";  
$personer[2] = "Johan";  
$personer[3] = "Henrik";
```

```
for($i=0;$i<count($personer);$i++){
```

I detta exemplet så fungerar `for` dåligt iom att vi har hoppat över/tagit bort en plats i arrayen. `Count` skulle säga att arrayen ä 3 element stor och iterationen skulle gå från 0-2 (0-1-2) och därmed missa Henrik.

```
$personer["Johan"] = 42;  
$personer["Henrik"] = 43;
```

När vi har text som nycklar så fungerar `for` dåligt, en i regel använder iteratoren för att ta fram en viss plats med nummer

```
foreach($personer AS $index => $namn){  
    echo "index: " . $index . " & namn är " . $namn . "<br>";  
}
```

```
index: 0 & namn är Marcus  
index: 2 & namn är Johan  
index: 3 & namn är Henrik
```

```
index: Johan & namn är 42  
index: Henrik & namn är 43
```

`Foreach` går igenom arrayen från första platsen till sista platsen i arrayen och fungerar med båda exemplen av arrayer ovan.

# formulär

- Form - element
- Action - Var ska skicka data
- Method - hur ska vi skicka data

```
<form method="post" action="this_document.php">  
  <label>  
    Förnamn  
    <input type="text" name="fnamn">  
  </label>  
  
  <input type="submit" value="Skicka">  
  
</form>
```

Förnamn

Skicka



# Ta emot data från formulär

- Fördefinierade variabler för att ta emot formulär-data
- `$_POST`
- `$_GET`
- Arrayer med data
- Name på skickad data blir nyckeln för arrayen.

```
<form method="post" action="this_document.php">
  <label>
    Förnamn
    <input type="text" name="fnamn">
  </label>

  <input type="submit" value="Skicka">
</form>

<form action="this_document.php" method="get">
  <form method="get" action="this_document.php">
    <label>
      Förnamn
      <input type="text" name="fnamn">
    </label>

    <input type="submit" value="Skicka">
  </form>
```

`$_POST["fnamn"]`

`$_GET["fnamn"]`

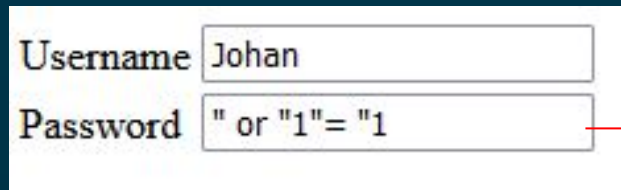
# Fråga mot databas i php

```
$sql_query = "SELECT * FROM users WHERE username=" . $_POST["username"] . " and password=" . $_POST["password"] . ";;";
```

# SQL injections

- Skickar in SQL-kod i så att detta skickas med okontrollerat mot vår databas i detta fall med hjälp av ett formulär

```
$sql_query = "SELECT * FROM users WHERE username=" . $_POST["username"] . " and password=" . $_POST["password"] . ";;";
```



Username

Password

```
$sql_query = 'SELECT * FROM users WHERE username="Johan" and password="\" or \"1\"= \"1\"";';
```

```
$sql_query = 'SELECT * FROM users WHERE username="Johan" and password="" or "1"="1";';
```

# Prepared statements (PDO)

- Lösningen för att inte bli drabbad av SQL injections var att använda sig av prepared statements

```
$pdo = new PDO('mysql:dbname=a00leifo;host=localhost', 'myusername', 'mypassword');  
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
$querystring='SELECT * FROM users WHERE username=:USERNAME and password=:PASSWD;';  
  
$stmt = $pdo->prepare($querystring);  
$stmt->bindParam(':USERNAME', $_POST['username']);  
$stmt->bindParam(':PASSWD', $_POST['password']);  
$stmt->execute();
```

```
$querystring='SELECT * FROM users WHERE username=:USERNAME and password=:PASSWD;';
```



```
$stmt = $pdo->prepare($querystring);  
$stmt->bindParam(':USERNAME', $_POST['username']);  
$stmt->bindParam(':PASSWD', $_POST['password']);  
$stmt->execute();
```

# Pdo Querys

- Kan ställa andra typer av frågor än bara SELECT i våra php-sql frågor

```
$querystring='SELECT * FROM users WHERE username=:USERNAME and password=:PASSWD;';  
$querystring='CALL MyProcedure(:MYINPUT);';  
$querystring='INSERT INTO aTable (col1,col2,col3) VALUES (:VALUE1,:VALUE2,:VALUE3);';
```

- Stora likheter med det vi skriver i MySQL workbench

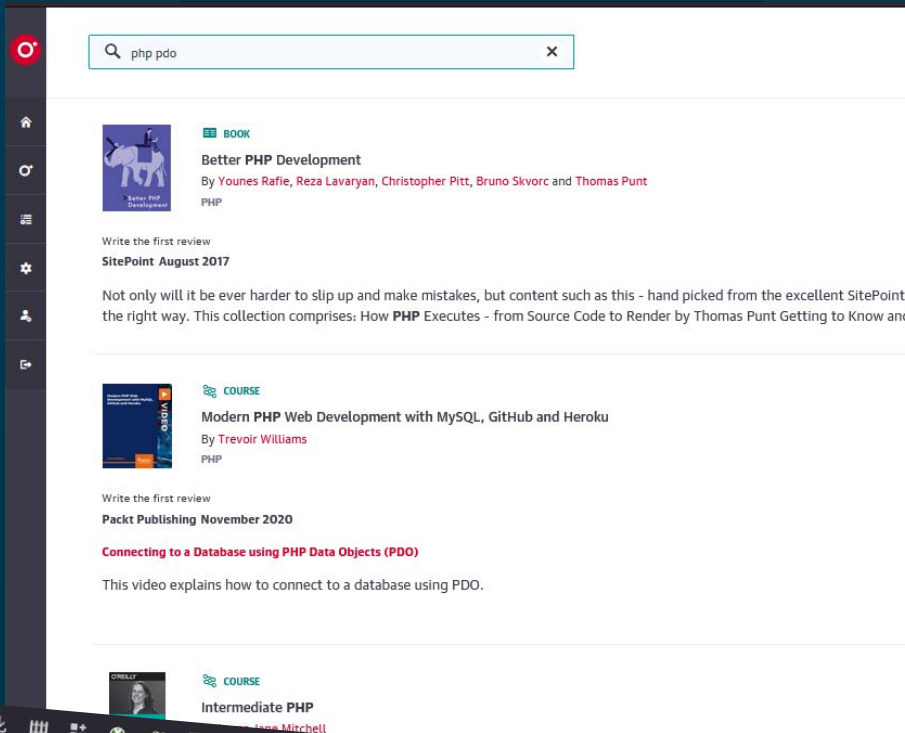
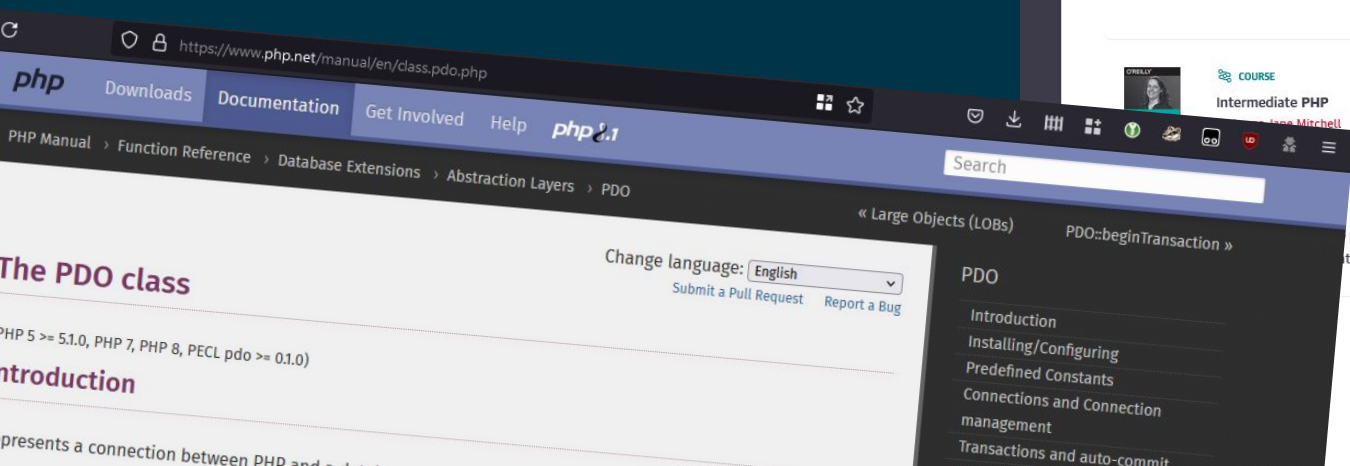
# Ta hand om resultatet från en Pdo-query

- Pdo-query likt innan
- Tar hand om resultatet som finns i variabeln `$stmt` med hjälp av `foreach`

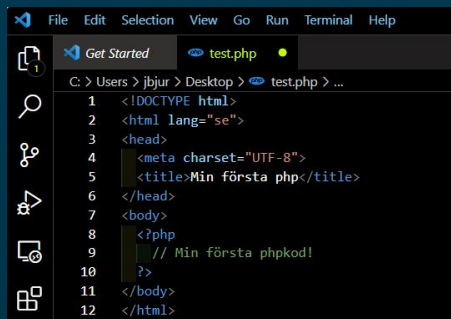
```
echo "<table>";  
$querystring='SELECT * FROM CUSTOMER WHERE SSN=:SSN';  
  
$stmt = $pdo->prepare($querystring);  
$stmt->bindParam(':SSN', $_GET['SSN']);  
$stmt->execute();  
  
foreach($stmt as $key => $row){  
    echo "<tr>";  
    echo "<td>".$row['CUSTNO']."</td>";  
    echo "<td>".$row['NAME']."</td>";  
    echo "<td>".$row['REGDATE']."</td>";  
    echo "</tr>";  
}  
echo "</table>";
```

# Resurser

- LenaSYS
- php.net
- Böcker om PHP och Pdo på O'reilly
  - Finns flera bra
- En sökmotor nära dig...

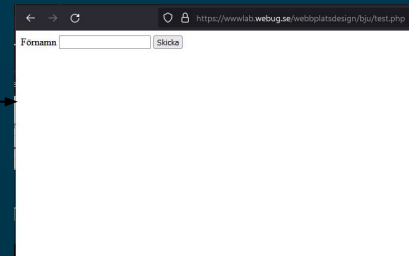
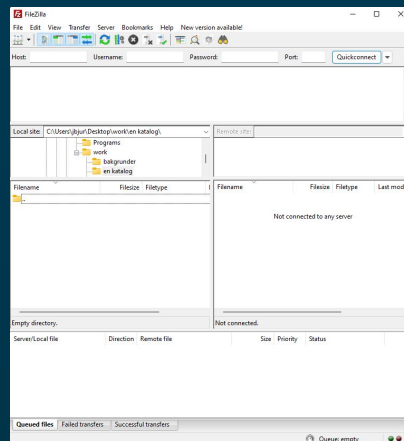


# Tips för att komma igång



```
1 <!DOCTYPE html>
2 <html lang="se">
3 <head>
4   <meta charset="UTF-8">
5   <title>Min första php</title>
6 </head>
7 <body>
8   <?php
9     // Min första phpkod!
10  >
11 </body>
12 </html>
```

→ filnamn.php →



Samma inloggningsuppgifter som ni hade på  
SSH/SFTP till MySQL-Workbench  
(Glömt? - lärarna kanske kan skriva det på tavlan)