Om include-filer i PHP

För att få bättre struktur och slippa skriva vanligt förekommande kod flera gånger använder man ofta include-filer i PHP. Här kommer en kort beskrivning av hur include fungerar. Grunden är att include-uttrycket infogar en fil i en annan.

För att förstå hur include fungerar är det viktigt att förstå att en php-fil kan innehålla flera avsnitt php-kod, som i följande exempel:

example1.php:

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5  <title>Untitled Document</title>
6  </head>
7  <body>
8  <h1>Ett PHP-exempel</h1>
9  <p>Vi börjar med lite text i html-koden.</p>
10 <?php
11 $text1= "Sedan kommer ett avsnitt text som skapas av PHP-kod";
12 $text2= "Ett nytt avsnitt som skapas av PHP-kod";
13 echo "<p>";
14 echo $text1;
15 echo "</p>";
16 ?>
17 <p>Här kommer text i html-koden igen, mittemellan php-avsnitten. Snart ska vi se hur vi kan få
   in denna text med include-uttrycket.</p>
18 <?php
19 echo "<p>";
20 echo $text2; //här använder vi variabeln $text2 som fick sitt värde i första php-avsnittet
21 echo "</p>";
22 ?>
23 <p>Avslutande text i html-kod</p>
24 </body>
25 </html>
```

Första php-avsnittet finns på raderna 10-16 och andra php-avsnittet finns på raderna 18-22. Resultatet av koden blir

Ett PHP-exempel

Vi börjar med lite text i html-koden.

Sedan kommer ett avsnitt text som skapas av PHP-kod

Här kommer text i html-koden igen, mittemellan php-avsnitten. Snart ska vi se hur vi kan få in denna text med include-uttrycket.

Ett nytt avsnitt som skapas av PHP-kod

Avslutande text i html-kod

Notera hur variabler som definieras i första php-avsnittet "lever kvar" i andra avsnittet.

Vi ska nu se hur vi kan få samma resultat med en include-fil. Vi delar upp innehållet i två filer, example2.php och shorttext.php enligt följande:

example2.php

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5  <title>Example 2</title>
6  </head>
7  <body>
8  <h1>Ett PHP-exempel</h1>
9  <p>Vi börjar med lite text i html-koden.</p>
10 <?php
11 $text1= "Sedan kommer ett avsnitt text som skapas av PHP-kod";
12 $text2= "Ett nytt avsnitt som skapas av PHP-kod";
13 echo "<p>";
14 echo $text1;
15 echo "</p>";
16
17 include "shorttext.php";
18
19 echo "<p>";
20 echo $text2; //här använder vi variabeln $text2 som fick sitt värde i första php-avsnittet
21 echo "</p>";
22 ?>
23 <p>Avslutande text i html-kod</p>
24 </body>
25 </html>
```

shorttext.php:

```
1  <p>Här kommer text i html-koden igen, mittemellan php-avsnitten. Snart ska vi se hur vi kan få
   in denna text med include-uttrycket.</p>
```

Vad blir reslultatet när man öppnar example2.php? Jo, all text i shorttext.php infogas i example2.php. **Men inte bara det! Det infogas även en php-sluttagg före den infogade filen, och en php-starttagg efter den infogade filen!**

Resultatet av example2.php med shorttext.php infogad blir alltså en sammanfogning av följande delar:

- raderna **1-16** i example2.php
- en php-sluttagg, alltså **?>**
- **hela** shorttext.php
- en php-starttagg, alltså **<?php**
- raderna **18-25** i example2.php

Resultatet blir som om allt skrivits i en fil med följande innehåll:

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5  <title>Example 2</title>
6  </head>
7  <body>
8  <h1>Ett PHP-exempel</h1>
9  <p>Vi börjar med lite text i html-koden.</p>
10 <?php
11 $text1= "Sedan kommer ett avsnitt text som skapas av PHP-kod";
12 $text2= "Ett nytt avsnitt som skapas av PHP-kod";
13 echo "<p>";
14 echo $text1;
15 echo "</p>";
16
17 ?>
18 <p>Här kommer text i html-koden igen, mittemellan php-avsnitten. Snart ska vi se hur vi kan få
   in denna text med include-uttrycket.</p>
19 <?php
20
21 echo "<p>";
22 echo $text2; //här använder vi variabeln $text2 som fick sitt värde i första php-avsnittet
23 echo "</p>";
24 ?>
25 <p>Avslutande text i html-kod</p>
26 </body>
27 </html>
```

Observera att sluttaggen på rad 17 och starttaggen på rad 19 varken finns med filen example2.php eller shorttext.php. De skapas automatiskt i samband med att shorttext.php infogas. Om de inte hade skapats, skulle det inte gå att infoga en fil med ren html-kod. Då skulle det blivit så här:

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5  <title>Example 2</title>
6  </head>
7  <body>
8  <h1>Ett PHP-exempel</h1>
9  <p>Vi börjar med lite text i html-koden.</p>
10 <?php
11 $text1= "Sedan kommer ett avsnitt text som skapas av PHP-kod";
12 $text2= "Ett nytt avsnitt som skapas av PHP-kod";
13 echo "<p>";
14 echo $text1;
15 echo "</p>";
16
17 <p>Här kommer text i html-koden igen, mittemellan php-avsnitten. Snart ska vi se hur vi kan få
   in denna text med include-uttrycket.</p>
18
19 echo "<p>";
20 echo $text2; //här använder vi variabeln $text2 som fick sitt värde i första php-avsnittet
21 echo "</p>";
22 ?>
23 <p>Avslutande text i html-kod</p>
24 </body>
25 </html>
```

Detta skulle inte fungera eftersom det då skulle finnas ren html-kod i php-avsnittet (rad 17 i det felaktiga exemplet ovan). En förutsättning för att det ska fungera är alltså att det automatiskt skapas en php-sluttagg före den infogade filen och en starttagg efter den.

Vi har nu sett hur man kan infoga en fil med endast html-kod i en php-fil. Nu ska vi se hur man även kan infoga en fil med php-kod.

Antag att vi har en include-fil som definierar värdena i en array. Själva huvudfilen används sedan för att skriva ut alla värden. Det skulle kunna se ut så här:

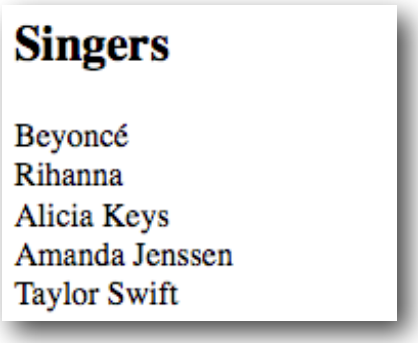
arraydef.php:

```php
<?php
$singers=array('Beyoncé','Rihanna','Alicia Keys','Amanda Jenssen','Taylor Swift');
?>
```

main.php:

```php
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Singers</title>
</head>
<body>
<h2>Singers</h2>
<?php
include "arraydef.php";
$numberofsingers=count($singers);
for ($i=0;$i<$numberofsingers;++$i)
{
    echo $singers[$i];
    echo "<br />";
}
?>
</body>
</html>
```

Utskriften blir följande om vi öppnar main.php i en webb-läsare:



Här är det viktigt att notera att filen arraydef.php som infogas i main innehåller php-kod, och att den därför måste inledas med php-starttagg och avslutas med sluttagg. Precis som i exemplet innan, läggs det automatiskt till en sluttagg före include-filen, och en starttagg efter den. Resultatet blir alltså som om följande hade stått i en enda fil:

main.php och arraydef.php ihopslagna:

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5  <title>Singers</title>
6  </head>
7  <body>
8  <h2>Singers</h2>
9  <?php
10 ?>
11 <?php
12 $singers=array('Beyoncé','Rihanna','Alicia Keys','Amanda Jenssen','Taylor Swift');
13 ?>
14 <?php
15 $numberofsingers=count($singers);
16 for ($i=0;$i<$numberofsingers;++$i)
17 {
18     echo $singers[$i];
19     echo "<br />";
20 }
21 ?>
22 </body>
23 </html>
```

Starttaggen på rad 9 kommer från main.php. Sluttaggen på rad 10 skapas automatiskt när arraydef.php infogas. Därför är det viktigt att arraydef.php innehåller en php-starttagg och det är den som syns på rad 11. Rad 13 innehåller sluttaggen från arraydef.php. På rad 14 ser vi en starttagg som automatiskt skapats efter include-filen.

Det kan tyckas onödigt att php automatiskt skapar en php-sluttag (den på rad 10 ovan) före include-filen och en starttagg efter include-filen (på rad 14). Sluttaggen på rad 10 och starttaggen på rad 11 tar ut varandra, och kan plockas bort utan att resultatet förändras. Det samma gäller för taggarna på rad 13 och 14. Om de inte hade skapats, hade vår fil arraydef.php inte behövt innehålla starttagg och sluttagg men då hade vi inte kunnat infoga html-kod som vi gjorde i första exemplet.

Sammanfattningsvis: innan en fil infogas skapas en php-sluttagg, för att tala om att nu kommer det kanske inte php-kod längre, utan eventuellt html-kod. När filen har infogats kommer en starttagg för att tala om att nu är det php igen. Om filen som infogas innehåller php-kod måste den infogade filen ha inledande starttagg och avslutande sluttagg, annars kommer den infogade filen att tolkas som html-kod.