# Vad vi har gjort med dokumentet.

**A**

I mappen site skapade vi ännu en mapp vid namn incl.

I root’en skapade vi en mapp vid namn src.

Vi skapade filen demo.php och sparade i incl/site

Vi tog bort hela tabellen vi byggt upp men lämnade kvar den första raden som vi kommenterade bort.

Under div id=”contentsection” inkluderades demo.php.

Div id=”buttonlink” flyttade vi från content.php till demo.php

Provkörde och och validerade filen.

Sedan skapade vi MyArray.php och sparade den i src mappen.

**B**

Vi skapade filerna default.php, hidden.php, modurl.php, cookie.php och session.php och sparade dom i incl/site mappen.

I varje fil av dessa skrev vi ut filens namn med echo(”filnamn.php”);

I nav.php gjorde vi så att samtliga länkar pekade på content.php.

Vi testade detta och det fungerade, sedan ändra vi länkarna enligt modifierad url enligt följande:

Gömda fält – content.php?p=h

Modifierad url – content.php?p=m

Kakor – content.php?p=c

Sessioner – content.php?p=s

Vi provkörde filen och det fungerade och gick igenom validering.

**C**

I div id=”contentsection” inkluderade vi rätt fil för den länk användaren tryckt på. Vi kontrollerade om variabeln p hade ett värde och vilket värdet var. Vi använde detta för att kontrollera vilken fil vi skulle inkludera. Om variabeln p inte hade något värde inkluderade vi default.php och strängen ”Ingen metod vald ännu!” med echo.

Slutligen inkluderades den valda filen som fanns i variabeln $filetoinclude. Här skulle också MyArray.php inkluderas men vi valde att göra det i varje fil i incl/site istället.

Vi testkörde och validerade filen utan problem.

**D-G**

Nu började vi skriva kod i hidden.php, modurl.php, cookie.php och session.php för att lösa problemet med tillståndslös miljö.

Vi kopierade innehållet från demo.php och div id=”buttonlink” till hidden.php, cookie.php och session.php. Vi bytte ut värdet i actionattributet till content.php och den modifierad url som länkarna i content.php hade.

För modurl.php kopierade vi också in div id=”buttonlink” med bytte ut formuläret mot två länkar som pekar på content.php och använde där modifierad url för att länkarna skulle peka rätt.

I hidden.php lade vi till ett gömt fält i formuläret och döpte det till hidArray och gav det value=””.

I hidden.php skapade vi sedan ett objekt ur klassen MyArray och kontrollerade om någon knapp är tryckt i formuläret. Om någon knapp blev tryckt anropade vi lämplig metod i objektet.

Trycker man på slumpa så slumpade vi 100 tal med randomArray() och skrev ut det i en tabell med array2XHTML() och skrev ut vektorns innehåll som en sträng i value för det gömda fältet med array2String().

Trycker man istället på Sortera så hämtade vi värdet i det gömda fältet och placerade det i ett objekt med string2array(). Vi sorterade sedan vektorn med sort() och skrev ut den som en tabell med array2XHTML och avslutade med att skriva ut den sorterade vektorn som en sträng till det gömda fältet på nytt, med array2string().

Det blev en liknande lösning för modurl.php med istället för ett gömt fält använde vi länkar och placerade vektorns innehåll i en variabel i länken.

För cookie.php och session.php använde vi funktionsanrop.

I cookie.php använde vi samma funktioner som för hidden- och modurl.php. Det som tillkom är setcookie() för att skapa och ändra en värde för en kaka. I sessioner bytte vi ut array2string() och string2array() mot setArray() och getArray(). Vi använde också ob\_start();, session\_start(); och session\_regenerate\_id(true);.

För att hämta data i samband med cookies.php använde vi $\_COOKIE[] och för sessioner $\_SESSION.

**H**

I avslutade med att testa alla våra sidor och olika slumpa och sortera metoder. Vi validerade också varje sida innan och efter varje knapptryck. Vi gjorde också browsershots för att ta skärmdumpar men fick till slut timeout på sidan, men vi fick ihop 7 browsershots i olika webbläsare innan det hände.