Institutt for informasjonssystemer

Organisering av kode

IS-115 Webprogrammering i PHP | Peter André Busch

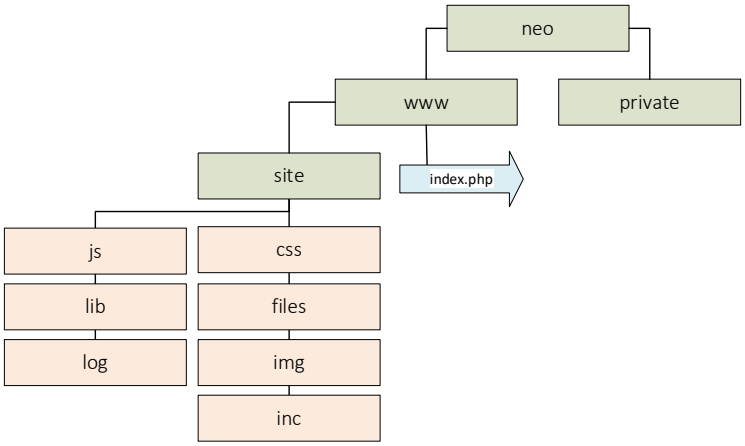
Innhold

Leksjon 5: organisering av kode

I denne leksjonen ser vi på ulike aspekter på hvordan koden kan organiseres og gjøres mer ryddig. Du har kanskje allerede fulgt flere av disse praksisene. Da blir det mindre jobb nå. Du bør gå gjennom det du har gjort hittil for å rydde litt opp i både oppgavekoden og koden for medlemsprosjektet dersom det er nødvendig. Vi presenterer dere også for objektorientert programmering i PHP i denne modulen. PHP var opprinnelig et proseduralt språk, men fungerer like fint som et objektorientert språk. Du velger derfor selv om du vil bruke objektorientert programmering (OOP). OOP skal vi gjennomgå i kapittel 6. Kapittel 5 i læreboka behandler organisering av kode.

1. Filstruktur

Det finnes ingen fasit på hvordan filstrukturen for ditt nettsted skal se ut. Her må du prøve deg frem selv for å finne ut det du foretrekker. Vi skal imidlertid komme med noen tips til deg. Når du oppretter en mappe for ditt nettsted (f.eks. 'neo'), så bør du antakeligvis ha en mappe inni den som heter noe sånt som 'www'. Da kan du definere 'www' som rotmappe for SFTP-klienten og du kan lagre ting som har med nettstedet å gjøre utenfor 'www' (du vil ikke at dette lastes opp på verdensveven). De som bruker Docker har trolig public som rotmappe. Alt som er utenfor rotmappen blir ikke tilgjengelig for folk på web. Her er et eksempel på en filstruktur:



Mappen 'site' inneholder andre ressursmapper igjen (andre kaller denne f.eks. for 'assets' istedenfor 'site'). Dette er for eksempel mappen 'css' (inneholder alle css-filer jeg bruker for nettstedet jeg utvikler), 'lib' (inneholder alle klasser), 'img' (inneholder alle bildefiler) og 'inc' (inneholder alle include-filer). Jeg har alltid en del filer som jeg inkluderer når jeg trenger dem (noen trenger jeg mer eller mindre alltid, slik som filen som håndterer databasetilkoblingen). Jeg synliggjør hvilke filer som tilhører hvilken kategori så langt det lar seg gjøre. For eksempel heter filen som håndterer databasetilkoblingen db.inc.php (slik ser jeg at dette er en include-fil). Det er også mulig å lagre noen (alle?) disse filene utenfor webroten ettersom de ikke skal være direkte tilgjengelige på web. En katalog som inneholder loggfiler bør absolutt ikke være tilgjengelig fra weben.

2. Kommentering

Det er god praksis å kommentere koden underveis. Selv om alt er klart som blekk for deg mens du programmerer, kan du oppdage at du selv(!) eller dine kolleger/medstudenter ikke skjønner så mye av

koden senere. Jeg har selv sittet og grublet på hva jeg har gjort et halvt år etter at kode ble skrevet. Derfor kommenterer vi underveis hva vi gjør. Det finnes to måter å kommentere på som vi har presentert tidligere i leksjon 2. Det fins mange former for kommentarer. Bruk av // og /* */ er vanligst, men også # er mulig.

```
// dette er en kommentar som gjelder til neste linjeskift.

/* her er en kommentar
over flere
linjer */
```

Her er et eksempel fra et nettsted hvor koden for å koble til databasen ligger i en include-fil (mer om det senere):

```
<?php
    //Connect to database
    require_once 'site/inc/db_connection.inc.php';
```

Det går også an å kommentere etter kode slik vi gjorde med `date()`-funksjonen:

```
$today = date("m.d.y"); // 03.31.20
```

Gjør det til en god vane å kommentere koden du skriver. Du kan kommentere midt inni en kodesekvens, inni `while()`-løkker osv. Det finnes ingen fasitsvar på hva som er den beste måten å gjøre dette på – det viktigste er at du forstår koden din og at ev. andre kolleger kan forstå hva du har gjort! Det er viktig med god dokumentasjon selv om det ikke er den gøyeste delen av programmeringen.

Dette må ikke forveksles med kommentering av HTML-kode. Dette gjøres på sin egen måte. Her er et eksempel på hvordan det fungerer:

```
<!--
<li class="no-padding<?php if( $path_parts['dirname'] == '/cpanel/setup' ) { echo ' active'; } ?>">
</li>
-->
```

Her er det **<!--** som brukes som åpningstag og **-->** som slutt-tag. All HTML-kode imellom blir utkommentert. Legg merke til PHP-koden i eksemplet ovenfor som ikke er kommentert ut. Den vil altså kjøres uavhengig av om HTML-koden er utkommentert. I dette tilfellet burde også PHP-koden vært utkommentert, f.eks. med // foran if slik at heller ikke den kjøres.

3. God programmeringsskikk

Det er god praksis å følge standard programmeringsskikk. Det er som med språk ellers, det blir enklere å forstå hverandre om vi følger felles regler. Det ser også finere ut når standard programmeringsskikk brukes konsekvent. Det finnes ulike «sett» med skikker. Se https://phptherightway.com/ for god programmeringspraksis med PHP. Det er mye stoff her som du kan se på innimellom. Til denne modulen anbefaler vi å se spesielt på de som har med kodestil å gjøre. Du finner mye interessant lesestoff, forslag til videre lesning, ressurser m.m. her: https://phptherightway.com/#code_style_guide.

Vi introduserer imidlertid noen enkle regler her som du kan følge:

1. PHP-koder

PHP-kode MÅ bruke de lange `<?php ?>`-kodene eller de korte `<?= ?>`-kodene; de andre tagvariantene må ikke brukes.

2. Navngivning av variabler, konstanter og funksjoner

Det mest vanlige er å bruke store og små bokstaver for å adskille hvert nye ord, f.eks.:

```
<?php $camelCase = ‘test’; ?>
```

Det går også an å bruke understrek, f.eks.

```
<?php $under_score = ‘test’; ?>
```

Konstanter skal ha store bokstaver og ev. understrek ved flere ord:

```
<?php define("CLASS_TABLE", "is115_users"); ?>
```

3. Navngivning av klasser

Klasser skal være substantiv eller substantivsuttrykk. Hvert nye ord i navnet skal ha stor forbokstav og være så korte som mulig:

```
class User {
```

4. Navngivning av filer og språkinnhold

Ikke bland norsk og engelsk i kode. Bruk suffix som inc.php, lib.php e.l. for include-filer, klassefiler etc.

5. Intendering

Bruk tabulator for hver nye underordnede kodeblokk:

```
if ( empty( $_POST['fornavn'] ) )
{
      $feilmeldinger[] = "Fornavn mangler";
}
```

6. Linjelengde

Vær linje med kode bør absolutt ikke gå over 120 tegn og bør helst være mindre enn 80 tegn.

7. Bruk av mellomrom

- naturlig mellomrom mellom hovedkodeblokker
- ikke mellomrom mellom et metodenavn og åpningsparantesen
- Ikke mellomrom mellom en unær operator og variabelen den virker på
- ikke etter en åpningsparentes og før en lukkeparentes (dette gjelder også []).
- ikke før et et semikolon

8. Kommentarer

Kommentarer skal være så korte (gjerne 1-2 ord), konsise (spot on) som mulig. Ingen blanding av norsk og engelsk.

9. Bruk av klammeparentes

Konsekvent plassering av parenteser (f.eks. alltid på ny linje for funksjoner):

```
if ( empty( $_POST['fornavn'] ) )
{
      $feilmeldinger[] = "Fornavn mangler";
}
```

eller:

```
if ( empty( $_POST['fornavn'] ) ) {
      $feilmeldinger[] = "Fornavn mangler";
}
```

10. Vær konsekvent!

For alle reglene gjelder er det viktigste at dere er konsekvente. Ikke bland stiler. Å følge vedtatte programmeringsskikker er bra fordi de gir dere beste praksis, god oversikt over egen kode, gjør at andre kan forstå deres kode og gjør koden «kompatibel» med annen kode (f.eks. om dere koder plugins e.l. for Microsoft-programvare).