Slutgiltig laborationsinlämning

E-tjänster och webbprogrammering 7.5 hp VT-14

Introduktion

Laborationerna i sig är inte obligatoriska. Således finns det inte heller någon inlämning per laboration. Den som utför laborationerna kommer dock att ha ett **signifikant försprång** när det kommer till denna uppgift. Denna uppgift är obligatorisk och ska (1) lämnas in, (2) rättas, och sedan (3) redovisas vid en individuell dugga.

Uppgiften

Uppgiften går ut på att skapa en sida med kommentarsfunktionalitet. Endast inloggade användare ska kunna posta kommentarer. Således behöver användare kunna registrera sig och logga in. Användare ska även kunna logga ut. Kommentarer listas med avsändare och meddelande. Den användare som är inloggad är den som står som avsändare för en kommentar. Användare måste registreras med e-post och lösenord.

All formulärdata ska (på ett adekvat sätt) valideras på både klient-, och serversidan. Användares lösenord får **inte** sparas som plain-text, utan ska hash:as och saltas. Alla sidor måste vara skyddade ifrån `SQL injections`.

Kommentarer ska postas genom AJAX (alltså utan att sidan laddas om). Mer funktionalitet **får** utföras genom AJAX.

Sidan ska se professionell ut. Oseriösa eller slarviga inlämningar godtas ej.

Övriga krav

- Två användare ska inte kunna registrera sig med samma e-post.

Föreslagen struktur

Denna filstruktur är bara ett av otaliga sätt att angripa detta problem. Du bör alltså endast tolka nedan som inspiration till en lösning. Kanske ser din lösning ut så här. Kanske inte. Det viktiga är att du uppfyller kraven.

- lab-handin-fornamn-efternamn (mapp)
  - `index.php`
  - `register.php`
  - `register-process.php`
  - `login.php`
  - `login-process.php`
  - `logout-process.php`
  - `posts.php`
  - `posts-create.php`
  - `include` (mapp som innehåller filer som inte bör navigeras till)
    - `bootstrap.php` (Fil som include:ar alla klasser så att det räcker med att ladda in den här filen. Samt utför vitala aktiviteter såsom att t.ex. starta sessionen.
    - `views` (mapp som innehåller `html`-fokuserade filer)
      - `_header.php`
      - `_footer.php`
      - `_posts-list.php`
      - `_posts-new.php`
      - `login.php`
      - `posts.php`
      - `register.php`
    - `models` (mapp som innehåller `klasser`)
      - `db.php`
      - `user.php`
      - `post.php`
      - `authorizer.php`
  - `assets` (mapp)
    - `img` (mapp med bilder)
    - `js` (mapp med javscriptfiler)
    - `css` (mapp med stylesheets)

Notiser om strukturen

I mappen `views` används underscore (_) för att denotera en "partial view". Med andra ord så kommer en få en fullvärdig `html`-sida (ändå ifrån `<html>` till `</html>`) om en väljer att `include`:a t.ex. `posts.php`. Men inte om en inkluderar t.ex `_header.php`. Detta är endast konvention, men en god konvention att följa.

Mappen `models` heter just så för att denotera att den innehåller filer som korresponderar till klasser. Filen `post.php` kan t.ex. således antas innehålla en definition för klassen `Post`. Vi rekommenderar starkt att ni arbetar med `php` objektorienterat. Men om ni **inte** gör det hade mappen lika gärna kunnat heta `functions.php`. Isåfall skulle mappen rimligen innehålla ett gäng `php`-filer (t.ex. `db-connect.php`) som i sin tur definierar en mängd metoder som vi sedan kan anropa. Såsom t.ex. `connect_to_db();` eller `create_post($author, $message);`.

Om du istället blandar `oop` och `procedurell` kod så kanske du t.o.m. både vill ha en mapp som heter `functions` och en som heter `models`.

Mappen `include` innehåller alltså filer som inte bör navigeras till. Vi bör alltså inte låta användaren skriva in `url`:en `www.example.com/include/bootstrap.php`. Denna mapp innehåller alltså istället filer som `include`:as av övriga "topp-nivå"-filer. Dessa "topp-nivå"-filer kan alltså ses som spindlarna i nätet som ser till att alla delar surras ihop korrekt.

php	2	3
css	5	6
Javascript	8	9