

Labb 3

E-tjänster och webbprogrammering 7.5 hp VT-14

Introduktion

I denna laboration kommer vi att arbeta i två pass. I det första kommer vi att refaktorera vår applikation (alltså resultatet av tidigare labbar). För att öka nivån av “kontroll”, struktur och läsbarhet. Vi kommer att refaktorera vår `HTML`, `CSS`, `PHP`, `JavaScript` såväl som vår `SQL`. Med andra ord all kod vi skrivit :)

Föreslagen struktur

Denna laboration består endast av 1 fas. Detta är en vidareutveckling av den applikation vi byggt i labb 1 och 2. Denna laboration kräver ingen inlämning. Men för att få en uppfattning om hur ett färdigt projekt efter denna labb skulle kunna se ut – beakta nedan filstruktur.

- lab3-fornamn-eternamn (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 javascriptfiler)
 - * css (mapp med stylesheets)

Er faktiska filstruktur kan förstås markant variera beroende på vilken strategi ni väljer att lösa problemet med. Samt vilka refaktoreringar ni faktiskt genomför. Läs vidare i instruktionerna för klarare förståelse.

Uppgifter

Nedan följer uppgifterna som resulterar i inlämningen ovan.

Uppgift 1 Refaktorering

Refaktorera din kod så att vi **åtminstone** uppfyller nedan.

- Organisera dina filer i en tydlig och konsistent mappstruktur.
- Eliminera duplikation av liknande kod **inom ett dokument** (i.e. generalisera).
- Eliminera duplikation av liknande kod **emellan dokument** (i.e. generalisera).
- Bryt ut långa block av procedurell kod till **metoder**.
- Dela upp metoder som gör flera saker i separata metoder, och ge de tydligare namn.
- Låt alla **variabler**, **metoder** och **klasser** ha beskrivande och tydliga namn.
- Bryt ut varje databas-query till en egen **metod**. Använd metodargument för att skicka värden. Exempel: `createPost($author, $message);`
- Om flera databas-query-metoder är liknande. Försök slå ihop dem.
- Separera “printing” och “processing”. Låt **endast** de “yttersta” filerna hantera utskrivning av `html`. Låt inte “process-filerna” hantera `html`. Låt de istället hantera data-processning och business logic. Se förslagen inlämningsstruktur (högst upp) för ett förslag på denna indelning. I förslaget så har man valt att låta alla “visuella” filer bo i mappen `partials`. Sedan inkluderar `index.php` “rätt fil” beroende på vad som ska göras vid det givna tillfället. Vyerna (alltså mappen `partials`) samt `index.php` använder i sin tur filer ifrån `include` när komplexare operationer (såsom att skriva till databasen eller validera input) behöver utföras.
- Blanda `html` och `php` på ett tydligt och konsistent sätt. Du måste inte följa strukturen som är föreslagen i detta dokument. Men det är viktigt att du **har** en struktur.
- Nu när vi fått ordning på struktur och förhoppningsvis fått det lättarbetat och öppen för förändring skall ni nu byta ut javascripten till att använda er av jQuery istället.