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 :)

I det andra passet av denna laboration kommer vi att arbeta med att bygga om vår sida till en Single Page Application. Vi kommer alltså använda det tillvägagångssätt som ofta benämns `AJAX` för att se till att användaren kan interagera med sidan som vanligt, men utan att någon sida någonsin fullständigt "laddas om".

Föreslagen struktur

Denna laboration består av 2 faser. Dessa är inte separata uppgifter, utan föreslagna steg för att bygga ovan nämnd applikation. Som alltså är en vidarutveckling av den applikation vi byggt i labb 1, 2 och 3. 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.

- `labb3-fornamn-efternamn` (mapp)
  - `index.php`
  - `include` (mapp)
    - `db-connect.php`
    - `db-queries.php`
    - `process-register.php`
    - `process-login.php`
    - `process-post-comment.php`
  - `assets` (mapp)
    - `img` (mapp med bilder)
    - `js` (mapp med javscriptfiler)
    - `css` (mapp med stylesheets)

Hur många filer ni har i varje mapp beror förstås 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**.
- 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.
- 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.
- Blanda `html` och `php` på ett tydligt och konsistent sätt.

Uppgift 2 Single-page application

När vi ifrån en webbsida, navigerar till en annan sida, så behöver webbläsaren "ladda om". Detta gäller alltså förstås då även när vi t.ex. postar formulär. I vårat fall – när användaren postar kommentarer.

Den teknikfamilj som ofta benämns `AJAX` används alltså för att kunna skicka data fram och tillbaka emellan användarens klient och servern utan att användarens webbläsare ska behöva ladda om sidan.

Din uppgift är nu att se till att eliminera alla sidomladdningar i din applikation. Om användaren med andra ord postar en kommentar, så ska denna kommentar postas utan att hela sidan laddas om. Ett anrop till servern, och ett hanterande av ett response måste alltså göras i bakgrunden med hjälp av `JavaScript`.