Rapport Oblig 1 - IMT 2571 - Datamodeller og Databaser

Av Jonas J. Solsvik, Studnr: 473193

Innledning:

Hvordan prosjektet skal brukes er beskrevet i /README.md.
Ellers har jeg forsøkt å dokumentere godt hva alt gjør, så det meste av dokumentering finnes i koden. Her prøver jeg å gi en oversikt med tanke på de spesifikke oppgavene som var gitt i oppgaveteksten.

Krav til innlevering:

" Format: Fila dere laster opp på Fronter skal være i PDF. Koden legges ved som vedlegg i PDF-fila.
ENDRET til koden lastes opp som egen .zip-fil, jamfør Fronter-innlegg - 2016-08-29.

Navnekonvensjon: Fila dere laster opp på Fronter skal ha navnet *Oblig1_etternavn_studentnummer*, med ditt etternavn og studentnummer.

Programkode: Programkode skal være ryddig og oversiktlig og skal følge konvensjonene i http://www.tutorialspoint.com/php/php_coding_standard.htm. "

Kommentar:
Har lastet opp zip - fil som inneholder all kildekode + .sql-fil + .pdf.
Den heter Oblig1_solsvik_473193.zip.

Del 1 og 2 skulle ikke skrives rapport på. Derfor er ikke dette gjort.

DEL 3 -- LAG DATABASEAPPLIKASJON

Oppgavebeskrivelse:

" I denne delen av oppgaven skal dere lage deres egen webapplikasjon ved bruk av PHP, PDO og MySQL. Oppgaven går ut på å lage en blog der enhver nettbruker kan legge inn et innlegg, og der alle nettbrukere kan se alle innlegg som er lagt inn. Til denne oppgaven skal dere lage en rapport der dere kort beskriver databasen og applikasjonen dere har laget. Dere skal legge ved databasen eksportert til SQL-fil (med noen få eksempelinnlegg) og dere skal legge ved all PHP-kode dere har laget. Dere kan velge om dere ønsker å bruke MVC-mønsteret for applikasjonen. "

Oppgave 1 - Lag database for bloggaplikasjonen.

Oppgavetekst - utdrag:
“Opprett en tabell for bloginnlegg i din database der du definerer disse kolonnene…...”

Oppretter sql-tabellen som har denne formen:

Databasenavn	oblig_1
tabellnavn	blogginnlegg
charset	utf-8-general-ci

Har satt opp følgende attrtibutter:

ID	int (11)	AUTO_INCREMENT
Tittel	varchar (255)	
Innhold	text	
ForfatterNavn	varchar (255)	
ForfatterEpost	varchar (255)	
PubTime	timestamp	CURRENT_TIMESTAMP
EditTime	timestamp	

Oppretter og eksporterer en database vha. phpMyAdmin verktøyet. Har lagret databasen i /blogginnlegg.sql

Har også lagt inn et par felter, som skal brukes til testing.
Noen av innleggene har ‘htmlspecialchars’ slik at det går ann å teste mot eventuell ‘escaping’ av disse.

Oppgave 2 - Lag PHP-kode til å vise Blogginnleggene

“Krav til programkoden:

- *Dere må bruke htmlspecialchars i PHP når genererer HTML for data hentet fra databasen for å unngå at spesielle tegn lagret i databasen (spesielt tegnene < og &) gir HTML-problemer.*
- *Dere skal presentere bloginnleggene med de nyeste innleggene først. (Tips: dere kan hente ut bloginnleggene sortert med nyeste innlegg først ved å legge til ORDER BY PubTime DESC i SQL-setningen dere bruker mot databasen.)*

- *Dere skal navngi alle kolonnene dere benytter i SQL-setningen – dvs. vi vil ikke akseptere SELECT * i SQL-setningen dere bruker.*
- *Dere skal lage ei egen php-fil der databasenavn og passord ligger slik at påloggingsinformasjon ikke er blandet med applikasjonskode og slik at påloggingsinformasjon kan importeres inn – og ikke dupliseres - i de ulike skriptene. “*

Fremgangsmetode:

1. /model/db_connect.php

For å vise blogginnleggene så har jeg først og fremst opprettet et db_connect.php - script. Dette scriptet opprettet et PDO-objekt som holder på tilkoblingen til databasen.
Her kan en også konfigurert innloggingsdetaljer til sin egen spesifikke database.

Oppfyller krav 4:

Informasjon om databasenavn og passord ligger lagret i egen fil.

2. /model/BlogModel.php

PDO objektet som opprettes av db_connect.php brukes flittig av klassen BlogModel som holder til i BlogModel.php.
BlogModel - klassen har ansvar for å være en interface, som resten av kodenbasen kan bruke for å komme i kontakt med blogginnlegg - databasen.

I BlogModel brukes disse medlemmene til å løse oppgave 2:

$db - PDO objekt;
$getPosts() - en metode som henter alle blog-innleggene vha. SQL setningen

```
"SELECT Tittel, ForfatterNavn, ForfatterEpost, PubTime, Innhold"
. " FROM"
. " blogginnlegg ORDER BY ID DESC";
```

Oppfyller krav 2 og 3:

- Her spesifiserer at de skal sorteres etter tabell-attributten ID, i synkende rekkefølge. Altså at nyeste POST alltid er først i resulat-arrayen.
- Her navngis alle attributter som skal hentes ut fra tabellradene.

3. /view/viewBlog.php

Kaller på metoden til BlogView som heter getPosts, og får tilbake en tabell med assosiative tabeller med alle innleggene til bloggen, i synkende rekkefølge, nyeste først.

Dette kjøres igjennom en foreach () loop som ved hjelp av echo og litt HTML strukturering, printer ut alle blogginnleggene.

Opfyller krav 1: Her brukes htmlspecialchars() til å escape innlegg-stringen.

Oppgave 3 - Lag PHP-kode for å legge inn et innlegg i bloggen

---

"Krav til programkoden:

- *Dere skal bruke PDO (ikke mysql- eller mysqli-bibliotekene).*
- *Dere skal avvise blog-innlegg dersom ikke brukeren har fylt inn alle feltene i formen.*
- *Dere skal håndtere mulige databasefeil (f.eks. skal ikke koden kræsje men gi en fornuftig feilmelding dersom databasen av en eller annen grunn er nede).*
- *Dere må bruke prepared statements i PDO for å unngå at sluttbrukeren forsøker å legge inn data som skaper problemer for systemet deres."*

Fremgangsmetode:

1. /view/viewBlog.php

Prosessen starter med en knapp id="nypostlinker" i viewBlog.php.
Denne sender serveren videre til viewNewpost.php

2. /view/viewNewpost.php

Her brukes det en HTML-form til å sende en POST request, som router serveren tilbake til viewBlog.php.

3. /view/viewBlog.php

Når POST requesten har ankomment viewBlog.php igjen så undersøkes det om
“ !empty($_POST[‘tittel] && !empty([‘innhold’])) “. Dersom dette er sant så brukes BlogModel sin metode createPost() for å sende tittel, innhold, forfatternavn og forfatterEpost inn i en ny rad i databasen.

Oppfyller krav 2:

Her avvises det nye blogginnlegget dersom brukeren ikke har fylt inn begge feltene.

Oppfyller krav3:

Her håndteres database-feil med en fornuftig tilbakemelding vha. en try-catch statement.

4. /model/ModelBlog.php

Denne metoden tar inn all ny informasjon som det nye blogginnlegget skal inneholde fra parameterene sine.

I BlogModel brukes disse medlemmen til å løse oppgave 3:

$db = PDO objekt
$createPost() - en metode som lager en ny post vha. SQL setningen:

```
"INSERT INTO blogginnlegg"
. "(Tittel, Innhold, ForfatterNavn, ForfatterEpost) "
. "VALUES"
 . "(:tittel, :innhold, :forfatternavn, :forfatterepost)
```

Oppfyller krav 1: PDO objekt brukes.
Oppfyller krav 4:

Her brukes en prepared-statement for å sende SQL request til databasen.

SLUTT PÅ RAPPORTEN