Hand-1

Group 7

Anders Andreasen (201205925)

Peter Mai (201271054)

Magnus Schou Abildgren (MA90747)

- **Redegøre for principperne for web udvikling i et MVC framework, og vigtigste karakteristika i et MVC framework til webudvikling.**

MVC er et softwarearkitektur, hvori ideen er at separere domæne, applikation og business logikken, fra resten af brugerinterfacet. Dette gøres ved at have tre forskellige dele Model-View-Controller. Modellen er den, som skal håndtere den grundlæggende adfærd og de data der er i en applikation. Modellen kan besvare de anmodninger der skulle komme andre steder fra og ændre sin tilstand, hvilket kan gøres ved hjælp af en database, eller andre former for lagersystemer. Det korte og det lange er at modellen anvendes til datastyring for en applikation.

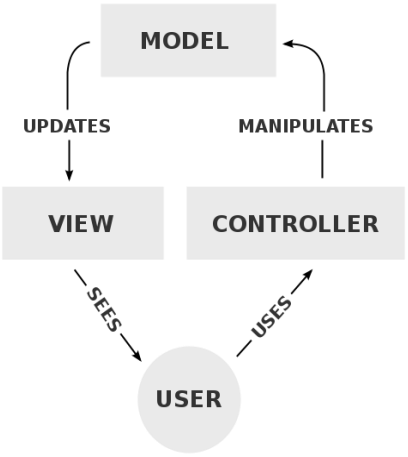
Implementering af en model er typisk en .cs fil, hvori man opretter en klasse med tilhørende properties med get og set metoder.

Viewet giver brugeren den effektive brugergrænseflade. Den viser data fra modellen til en form, som gør det meget brugervenligt at se på.

Implementering af et view foregår igennem layouts. Man kan oprette views på mange måder, manuelt eller igennem entity frameworket, som dermed også kan oprette en controller ud fra databasekontekst og model.

Controllerens opgave er at modtage brugerens input og anvende de data til at kalde modelobjekter, som skal sendes til viewet.

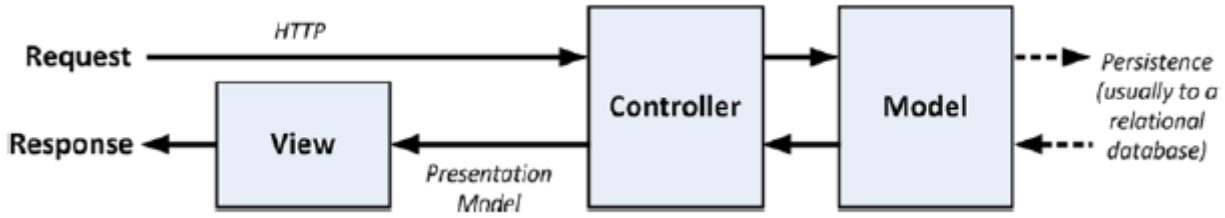
Implementering af en controller er ved at man først vælger en bestemt type controller. Heri er ideen, at der skal oprettes nogle metoder, som skal kunne håndtere businesslogikken. Eksempelvis hvis man arbejder med en database, ville det være fornuftigt, at anvende CRUD metoderne, til at manipulere modellaget fra viewet. Dette gøres ved at informere controlleren om hvilken, model og database der skal tilgås. På figur 1 ses flowet for MVC.



Figur 1 MVC Flow

Karakteristik for webudvikling i MVC framework:

Når vi arbejder med MVC frameworket i webudvikling, anvender vi controllers som er C# klasser, som er nedarvet fra System.Web.Mvc.Controller klasser, hvori hvert public metode i klassen er nedarvet fra Controller, som er en action metode, som er associeret med en konfigurerbar URL, igennem ASP.NET Routing system. Når der sendes et request til URL'en som er forbundet med action metoden, vil statementet i controller klassen blive eksekveret, for at udføre nogle operationer på domæne modellen samt vælge et view, der skal vises til brugeren. På figur 2 ses interaktionen for et MVC applikation.



Figur 2 Interaktion for et MVC applikation

ASP.NET frameworket bruger en view "motor", som er den komponent, som er ansvarlig for behandlingen af den visningen på viewet, for at generere et svar til browseren. I de tidligere ASP.NET blev der anvendt ASPX som syntax. Razor blev først introduceret i MVC3 som view "motor", hvilket blev modificeret i MVC 4 og uændret derefter i MVC 5. Razor er en syntaks der blandt andet understøtter, at man i viewets html kan anvende C# ved at indføre "@".

- Designe og implementere en webløsning, som omfatter persistering af data i en database.

Til persistering er der i dette tilfælde blevet brugt en lokal database, hvor alle informationer bliver gemt. Der CRUD til at gemme data til databasen, hvor hver tabel i databasen har sin egen controller til at håndter adgangen til databasen. Desuden har vi valgt at bruge code-first entryframeworked. Dog tilgåes databasen direkte fra HomeController, fordi HomeController skal presenter admin'en for oversigt af samtlige kategorier og komponenter i de relevante kategorier. Det er tiltænkt at brugeren skal have et view der minder om admin'ens overview. Dog uden redigerings mulighederne.

For at viewsne kan vise de relevante informationer, der skal vises er der blevet sendt models til viewet. Disse models kan så tilgåes i view.cshtml filen og manipuleres som normal C#-kode ved at skrive @ foran C#-koden. På denne måde er det muligt at opbygge html-filer med dataen fra databasen. Et eksemple kunne være at antalet af kategorier i kategori-listen afhænger af, hvor mange kategorier, der findes i databasen. Eller at komponent-listen afhænger af hvilken kategori, man har valgt, da det kun er komponenter for kategorien man har valgt, som bliver vist.

Til information imellem forskellige controllers og views er der blivet brugt parameter, til at fortælle, hvilket information man vil have frem.

Der var en del spild i begyndelsen, fordi vi arbejde med Azura database, men kunne den til at virke.

- Redegøre for og implementere authorisation and authentication i en webløsning.

[Authorize] attributen er blivet brugt til at sige tilgang til hjemmesiden skal forgå ved hjælp af login. Dette gøres for at kun admin skal have rettigheder, til at tilgå editor muligheder.

Der er blivet brugt athentication til at sikre dataen til databasen er rigtig. Desuden er det automatisk brugt i AccountControllers.

- Anvende relevant front-end framework i udviklingen af en webløsning med responsivt design, herunder også anvendelse og forståelse for CSS Preprocessors.

Der er i projektet kun blevet brugt bootstrap. Deres knapper, listviews, navigation-bar, mm.

Det ville også have været muligt at bruge LESS eller SASS til at designe CSS.

En ide kunne også være at bruge Autoprefixer, Gulp eller Grunt

- Redegøre for relevante sikkerheds- og performancemæssige udfordringer ved implementeringen og brugen af en webløsning.

Ved vores webløsning er der inden for sikkerhed ikke andet end at man først skal registrere sig som bruger og derefter kan man logge ind. Vi har anvendt MVC5's egen login system, hvilket er den eneste reelle form for sikkerhed til resten af komponent database.

For at tilgå en database er der i Web.config filen ved connectionStrings tagget ”Integrated Security = true ”, hvilket betyder at den nuværende Windows brugers tilladelse bliver brugt som authentication. Der kan anvendes SetAuthCookie metoden, som tilføjer en cookie til browseren, som gør at brugeren ikke behøver at verificere sig selv hvert gang man skal lave en request.

- Grundlæggende viden om hosting af webløsninger, her under også cloud baseret hosting

For at kunne få eksterne personer til at se ens webside er det nødvendigt at få siden lagt op på en webserver. Det er muligt at anvende sin egen computer til at agere som en webserver. Det er dog mere almindeligt at anvende web hosting, som kan gemme websiden på en offentlig server. Web hosting ordner også domain name registrering, som er en måde at referere til IP-adresser på med et andet navn i URL'en. Disse navne vil typisk være nemmere at huske og kan bruges til at identificere specifikke websider.

Alternativt kan man anvende cloud baseret hosting på virtuelle servers for at publishe sin webside. Cloud baseret hosting fungerer som en form for clustered hosting, hvor websider bliver hosted på flere servere ad

gangen, hvor cloud baseret hosting adskiller sig ved at have det hele på en server. Man kan fx have en host der kun kan tilgås ved at være på et specifikt netværk eller igennem VPN. Eller man kan fx anvende udbydere af cloud baseret hosting som Microsoft Azure.