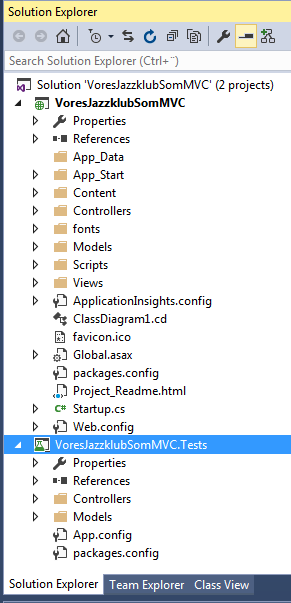
En del af opgaven er at undersøge, hvordan ASP.NET MVC fungerer, og holde det op mod andre måder at løse samme slags web-kommunikation på. Derfor bliver der oprettet nogle små projekter, der på ingen måde vil kunne anvendes til noget praktisk, men kun er til for at afsløre, hvordan centrale problemstillinger bliver adresseret.

Når man opretter et projekt i Visual Studio og vælger ASP.NET MVC med tilhørende testprojekt, får man en kørende applikation og tre fungerende test i en struktur som vist her; dog er mappen Models i testprojektet tilføjet efterfølgende.



Der er små 600 filer og over 800 mapper. Om det er en fordel eller ulempe, at få så meget, komme nok an på, hvor godt man kender IDE’et og den generelle tankegang i MVC-modellen.

I første omgang kan man nøjes med at koncentrere sig om mapperne Controllers, Models og Views i projektet, og de tilsvarende mapper for controllers og views i testprojektet. Der er ikke umiddelbart nogen test af webgrænsefladen, men i betalingsudgaven af Visual Studio er der mulighed for at få startet IE og der igennem optage nogle handlinger mod applikationen. Handlingerne kan så afspilles igen, uden anvendelse af en browser. Det er så kommunikationen mellem browseren og webserveren, der undersøges. Det er også muligt at tilføje web-tests, ved at skrive dem selv efter en optagelse [[[1]](#endnote-1)]. Alternativt kan Selenium anvendes, som er et produkt til test af web-sider [[[2]](#endnote-2)]. Det kræver så mere installation og applikationen, der skal testes, skal køre på en server uden for IDE’et

For komme i gang, oprettes først en controller-klasse med en public metode. Som udgangspunkt rammer man metoden ved at anvende en URL på følgende form: <domænenavn>/<controller>/<metode>. Det kan man konfigurere sig ud af, hvis der ønskes en anden logisk struktur set fra brugerens side.

Controlleren kan for så vidt godt anvendes alene, hvis metoden f.eks. returnerer en tekststreng eventuelt med et stykke html, men ellers kan browseren godt vise ren tekst uden markups. Det bliver dog nogle noget fattige websider, som kan laves på den måde, med mindre man koder sig frem til en webside med alt indhold; men det er absolut ikke at anbefale, da det er svært at overskue og hurtigt kan blive et meget omfattende stykke arbejde at vedligeholde koden.

Dernæst laves et view. Da controlleren blev oprettet, blev der dannet to mapper. En i controller-mappen og en i view-mappen. På den måde blive controller og view knyttet sammen. De enkelte metoder, der skal returnere et view, skal have et view med samme navn som metoden. Én metode returnerer ét view.

Der er et master-view, som man kan lade de øvrige views arve fra. På den måde kan man opnå ensartethed på tværs af siderne i applikationen.

Den tredje komponent er modellen. Her holdes de data, der skal vises på glaspladen subsidiært persisteres i databasen. Ved oprettelse af en controller blev der samtidigt oprettet en mappe til de tilhørende views. Det sker så ikke i relation til modellerne. Der er ikke den samme en-til-en relation, da man sagtens kan forestille sig, at en given model anvendes i flere sammenhænge. F.eks. kunne der være en side, som kun viser data, og en anden side, hvor data kan oprettes, vises, ændres og slettes. Den første side kan alle tilgå, og den anden side kan kun privilegerede få fat på. Det er de samme data, der skal anvendes på de to sider, og dermed samme model.

Modellen vil have adgang til forskellige metoder til læsning og skrivning på en tabel og i filsystemet. Her skal man så være opmærksom på konfigurationsfilerne. Til projektet er det Web.config og til test-projektet er det App.config. Her lægges connection strings til databaserne. Alt andet, der skiller test fra produktion, bør på lignende måde placeres i konfigurationsfiler.

En sidste ting, der kan være relevant på den funktionelle side af web-serveren, er RouteConfig.cs. Her kan man bestemme, hvor i applikationen man kommer hen ved at angive en sti efter domænenavnet i browserens URL input box.

1. https://msdn.microsoft.com/en-us/library/ms182538%28v=vs.90%29.aspx [↑](#endnote-ref-1)
2. https://blogs.msdn.microsoft.com/visualstudioalm/2016/01/27/getting-started-with-selenium-testing-in-a-continuous-integration-pipeline-with-visual-studio/ [↑](#endnote-ref-2)