Kaffeplaneten og Administrasjon-kaffeplaneten

Beskrivelse av prosjektene:

Prosjektet Kaffeplaneten en nettside med nettbutikk funksjonalitet. Nettsiden har funksjonalitet for brukerregistrering, innlogging, oppdatering av brukerdata, visning og kjøp av produkter, og visning av tidligere kjøp. Prosjektet er laget med Visual Studio 2015 i MVC .NET.

Prosjektet Administrasjon er en nettside med administrativ funksjonalitet som tar for seg alt av endringer tilknyttet nettsiden kaffeplaneten. Her kan administrator ut ifra sine rettigheter, lese alt av brukerinterkasjon i databaselogg, endre produkter, endre/kansellere ordrer og slette ansatte. Når man registrerer en ansatt kan man her sette hvilke av disse rettighetene den respektive ansatte skal kunne få tilgang til.

Vi har valgt å legge disse som to separerte prosjekter da det skaper mer ryddighet og oversikt over de forskjellige modulene, samt ved muligheten til å legge prosjektene på forskjellig serverer/områder for å spre belastningen.

Til opplysning:

Oppstart-siden for administrason-nettsiden viser en innloggingsside. Derfor trenger man på forhånd en hardkodet/forhåndsdefinert "superbruker" for å logge inn som vi har her:

Brukernavn:

sjefledersen@kaffeplaneten.no

Passord:

Sjefesen123

Denne trenger man å bruke første gangen for å kunne ha mulighet til å registrere nye ansatte og sette rettigheter.

Prosjektstruktur

Database:

CustomerContext.cs innholder Code First filen til databasen. Filen ligger i Models mappen. Databasen heter DatabaseKaffeplaeneten.

Database lag:

DBCustomer.cs, DBUser.cs, DBOrder.cs og DBProduct.cs tar seg av all aksess til databasen.

MVC lag

Model:

Alle modelklasser ligger i Models mappen. Disse representerer de ulike databaseentitetene.

Controller:

Alle kontrollerklassene ligger i Controllers mappen. Diss styrer dataflyten mellom databasen og viewene.

View:

De ulike viewene er organisert i undermapper i Views mappen.

Annet

Versjonskontroll:

Vi har benyttet Github og opprettet issues(merkelappproblemer) for alt av funksjonalitet, design ol. som er tilknyttet oppgaven. Vi har brukt git gjennom Visual til å push'e, pull'e og merge mellom oss. Ettersom vi benytter .Net-serveren til skolen med innlogging har vi ikke ønsket å gjøre repoet åpent for offentligheten da det er mange bots der ute som kan misbruke dette. Hvis det ønskes tilgang er det bare å si fra til en av oss slik at vi kan invitere.

DataCreator:

DataCreator har metode for å generere data til databasen. Brukes til å fylle databasen med produkter. Klassen har også metoder for å generere testdata.

SuperControllere:

Vi valgte å ha en SuperControllere i begge prosjektene som alle kontrollerene arver fra. Dette gjør det mulig å ha felles metoder alle kontrollerene har tilgang til. Vi valgte også å opprette en const string for hver Session variabel vi bruker. Dette sikrer mot skrivefeil i Session stringen. Disse ligger også i SuperController.cs.

Kodespråk:

Vi har valgt å kun bruke engelsk i koden. Dette er god kodepraksis da man ikke skal behøve å være norsk for å forstå koden. Kommentarer er på norsk.

Unit Tester

- Alle metoder, med unntak av konstruktør, er gjennomtestet i Admininstrsjonprosjektet.
- Metoder som er avhengig av Sessions er testet ved hjelp av Moq. MockHttpSession.cs inneholder get metoder for kontrollere som implementerer Moq slik at Session kan brukes for testing. Kilde til MockHttpSession koden:http://www.dontpaniclabs.com/blog/post/2011/03/22/testing-session-in-mvc-in-four-lines-of-code/ Valgte å bruke dette da det gir en ryddig løsning og gir støtte for å teste på Sessions i både kontroller og test metode. Så ikke løsningen på Fronter før etter denne var implementert og ser ingen grunn til å bytte.
- ViewBag verdier er ikke testet. Finner ingen enkel måte å kontrollere disse på, ser heller ikke behovet da de brukes til sende data til viewet og ikke har noen funksjonell betydning for programflyten.
- Kontrollerne i Kaffeplaneten prosjektet er ikke testet da disse hører til forrige oppgave. Dette fører til at enkelte metoder i BLL og Stub versjonene av DAL som kun brukes av Kaffeplaneten, ikke blir testet.
- Stub klassene ligger i et eget prosjekt, Stubs. Interfacene ligger i DAL.

- Alle DAL og Stubs implementerer et interface slik at dependency injection kan brukes. Unntaket er LoggingDAL.cs og LoggingDALStub.cs. Disse bruker et par const stringer og er arver derfor den abstrakte klassen ALoggingDAL.cs.

Laget av:

Magnus Johan Knalstad, s198760
Christer Bang, s198737
Sondre Husevold. s198755
Magnus Tønsager, s198761