Razor Pages Summary

Nederfor finder du et sammendrag af de vigtigste filer i et *Razor Pages* projekt (*Visual Studio 2022*).

Projektet er af typen **ASP.NET Core Web App**, med brug af **.NET 6.0** (NB: IKKE 3.1).

Bemærk, at et *Razor Pages* projekt vil indeholde flere filer af typen **.cshtml**. Disse filer rummer en form for mix af C#-kode og HTML, som Razor-frameworket kan forstå (vi vil fremover kalde dette for **Razor-kode**). Razor-kode kan således IKKE kompileres i et rent C# projekt, ejheller vises direkte i en browser.

Man bør også bemærke, at enhver **.cshtml**-fil hører sammen med en tilsvarende **.cshtml.cs**-fil (klik f.eks. på den lille trekant ud for **Index.cshtml**-filen, for at folde den ud). Disse to filer hører uløseligt sammen, og udgør tilsammen en enkelt *Razor Page*. Denne fil indeholder "ren" C#-kode, og repræsenterer den "model-klasse", som hører sammen med siden (dette begreb bliver gennemgået).

I den nedenstående tabel er der benyttet et farve-skema til at indikere vigtighed:

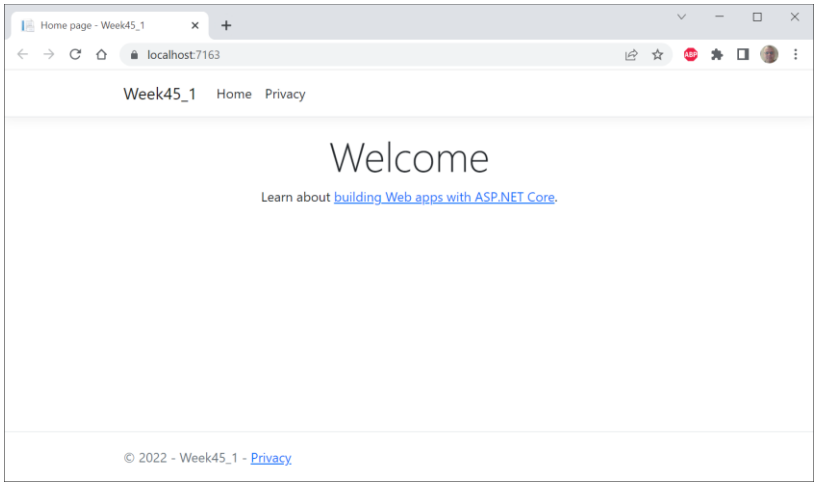
Grå tekst	Ikke vigtigt. Ikke noget vil komme til at berøre.
Normal tekst	Moderat vigtigt. Noget som vi måske vil rette i på et tidspunkt, men ikke vigtigt til at starte med.
Fed tekst (række fremhævet)	Vigtigt! Noget vi kommer til at ændre/tilføje til fra starten.

Oversigt over fil-struktur

Connected Services og Dependencies (folders)	Indeholder ikke noget der er relevant at se på eller ændre.
Properties (folder)	Indeholder kun en enkelt fil **launchsettings.json**, som vi ikke kommer til at berøre i første omgang.
wwwroot (folder)	Indeholder subfolders **css**, **js** og **lib**. Hvis man vil ændre på styling i projektet, kan man tilføje til **site.css** i **css**-folderen. Alternativt kan man lave en helt ny **.css** fil, og henvise til den i **_Layout.cs** (i **Pages/Shared** folderen). De øvrige filer kommer vi næppe til at berøre i første omgang.
Pages (folder)	Dette er en meget central folder, idet alle vores egne *Razor Pages* vil blive lagt i denne folder. Den indeholder fra starten flere centrale filer.
Pages\Shared\ _ValidationScriptsPartial.cshtml	Indeholder detaljer i.f.t. validering. Det kommer vi ikke til at røre ved i første omgang.
Pages\Shared\ _Layout.cshtml	Dette er en meget central fil, idet den rummer top-level layoutet for vores *Razor Pages*. Det er f.eks. her vi **definerer top-level links** i vores app, i en top-level navigation.-bar. Fra starten kan man finde to links ”Index” og ”Privacy”. I praksis vil vi kun ændre her, hvis vi vil ændre/tilføje links til denne top-level navigation-bar, eller tilføje noget til **script**-sektionen i bunden.
Pages\ _ViewImports.cshtml Pages\ _ViewStart.cshtml	Vi kommer ikke til at berøre disse filer i første omgang.
Pages\Error.cshtml	Rummer definition af en ”Error page”. Vi ændrer ikke på den i første omgang, men senere – f.eks. i jeres egne projekter – kan det måske være relevant at tilrette denne side.
Pages\Index.cshtml	Rummer definition af ”Index page”, d.v.s. den side, man ser som det første i web-appen.. Vi ændrer ikke på den i første omgang, men senere – f.eks. i jeres egne projekter – vil det være relevant at tilrette denne side.
Pages\Privacy.cshtml	Rummer definition af en ”Privacy page”. Vi ændrer ikke på den i første omgang, men senere – f.eks. i jeres egne projekter – kan det måske være relevant at tilrette denne side.
appsettings.json	Rummer definition af nogle få settings for appen, som vi ikke kommer til at pille ved i første omgang.
Program.cs	Her bliver selve appen konfigureret.

At køre en Razor Pages App

En *Razor Pages* app startes på helt normal vis fra *Visual Studio*, ved at klikke på det grønne trekant-ikon. MEN i modsætning til en konsol-app, startes vores **browser** i stedet (så hvis *Chrome* er din default-browser, bliver den startet)! Det vil se nogenlunde således ud:



Der kan muligvis stå et andet tal efter **localhost:**, men hvis der gør det, er det ikke noget problem. Lidt mere nøjagtigt sker der det at den såkaldte *Internet Information Server Express (IIS Express)* server startes, og den kører så appen for os. Det behøver vi ikke tænke videre over 😊.

At tilføje en domæne-klasse (simpel)

Hvis vi gerne vil tilføje en ny domæne-klasse til vores projekt – med det mål at kunne vise, oprette, ændre og slette instanser af denne klasse, d.v.s. udføre CRUD-operationer – går man typisk gennem disse trin:

1. Tilføj en ny folder med navnet **Models** til dit *Razor Pages* projekt, helt ude på top-niveau (d.v.s. på samme niveau som **Pages**-folderen). Hvis du allerede har oprettet **Models**-folderen, springer du dette trin over.
2. Højreklik på **Models**-folderen, og vælg **Add > Class**. Tilføj den nye klasse. Bemærk, at den nu vil ligge i namespacet *(dit projektnavn).***Models**. Lad blot dette stå.
3. Udfyld den nye klasse, med hvad den måtte have af properties, constructors, metoder, m.v..
4. Opret en ny folder under **Pages**-folderen, med samme navn som din klasse. Hvis din klasse f.eks. hedder **Customer**, får du således en mappe **Pages\Customer**.
5. Vi vil nu tilføje en eller flere *Razor Pages* knyttet til denne nye klasse. Disse pages vil ligge i den mappe, vi lige har lavet. Den første side man laver vil ofte have til formål at vise en liste over alle de instanser af klassen, systemet p.t. rummer. Man kan vælge at kalde denne side for Index, men man kan også vælge at kalde den for f.eks. GetAll(min klasses navn). Hvis klassen hedder **Customer**, vil man således kalde den for **GetAllCustomers**. Men HUSK NU: det er helt op til dig, hvad du kalder din page 😊. Hvis du synes bedre om f.eks. **ShowAllCustomers**, så kald den blot det.
6. Denne *"se alle"* page – som vi her vælger at kalde **GetAll…** – laves nu ved at højreklikke på **Pages/(***min klasse)*-folderen, og vælge **Add > Razor Page**. Vælg *Razor Page – Empty*, og klik på **Add**. Indtast navnet på din page, og klik på **Add**.
7. Nu er skelettet til den nye page lavet. Vi skal oftest hente noget data et sted fra – præcist hvordan dette skal gøres kan variere fra situation til situation. Hvis vi henter data fra en form for repository, skal der oftest tilføjes et *instance field* til model-klassen (den nede i **.cshtml.cs**-filen) af typen for repository-klassen. I simple tilfælde kan model-klassen selv initialisere dette repository i sin constructor, men oftest vil vi først skulle registrere vores repository son en *Service*, hvilket gøres i **Program.cs** (se også næste side). Det vil oftest ligne noget a la **builder.Services.AddSingleton<…>**. Detajler om dette bliver vist og gennemgået i timerne.
8. Når vores model-klasse er på plads, kan vi udfylde view-definitionen (den i **.cshtml-filen**), på den vis den nu skal udfyldes i den konkrete situation. Husk at man skal bruge **@** før enhver brug af C#-kode i view-definitionen, og at man kan få fat på properties/metoder fra model.klassen ved at skrive **Model**.*(navn på property/metode)*.
9. Endelig skal vi have tilføjet et link til vores nye page, det gøres i **_Layout.cs**. Typisk kan man bare copy-paste f.eks. **<li>**-elementet for Index, og så tilpasse **asp-page**-værdien og selve titlen. Husk at man skal angive den fulde sti (fra **Pages**), f.eks. **asp-page="Event/GetAllEvents"**.
10. Herefter kan vi definere flere pages for hørende til vores domæne-klasse. Disse pages vil typisk ligge nede i den samme folder som den page vi lige har lavet. Typisk vil man have separate sider for oprettelse, sletning og ændring.

At tilføje en service-klasse

Ofte vil vi have brug for også at tilføje et antal *Services* til vores app. En *Service* er i sig selv blot et interface og (mindst) en klasse som implementerer interfacet. Når vi har defineret dette, kan vi registrere vores *Service* i vores App (se nedenfor), og dermed kan alle vores model-klasser nu gøre brug af vores *Service*!

1. Tilføj en ny folder med navnet **Services** til dit *Razor Pages* projekt, helt ude på top-niveau (d.v.s. på samme niveau som **Pages**-folderen). Hvis du allerede har oprettet **Services**-folderen, springer du dette trin over.
2. Opret og implementer interface-definitionen og klassen som implementerer interfacet i **Services**-mappen.
3. Registrér den nye service i Program.cs, ved brug af **builder.Services.Add…** NB: Vær opmærksom på, om **AddTransient** eller **AddSingleton** er det rette valg. Hvis servicen er f.eks. en repository-agtig service, vil det som regel være **AddSingleton** der er det rette valg. Ved at bruge **AddSingleton** sikres det, at alle brugere af servicen har fat i den samme instans af servicen!
4. Nu kan alle model-klasserne få fat i en service gennem *Dependency Injection* i deres constructor. Mere konkret betyder det, at man nu kan tilføje en eller flere parametre til en model-klasses constructor; disse nye parametre skal have den interface-type, der svarer til den service man gerne vil benytte. Som regel vil man så i model-klassen have et instance-field af same interface-type, som man så kan initialisere i constructoren. Derved har model-klassen en permanent reference til servicen. Bemærk, at man kun behøver tilføje parametre for netop de services, man har brug for. Selve om der f.eks. er defineret fem services, kan man godt nøjes med kun f.eks. at tilføje de to, man har brug for.