UserStories Part 4– Saving with json & generics

Du skal fortsætte med projektet scrumboard.
I stedet for at benytte MockData, skal du nu kunne gemme dine userStories i json format i en fil.

Refaktorer (modificer) din UserStoryService

- I mappen 'services' skal du lave et interface 'IUserStoryService' med metoderne
  (se evt. Razor Note 1 : kap 11 Repository Design Pattern ss. 157)

  a. List<UserStory> GetAll()
  b. UserStory GetById(int id)
  c. UserStory Create(UserStory newUserStory)
  d. UserStory Delete(int id)
  e. UserStory Modify(UserStory modifiedUserStory)

  Husk at lave interfacet public

- Du skal nu lade din 'UserStoryService' implementere ovenstående interface
  dvs. du kommer til at implementere et par metoder i forhold til det du allerede har
  Du skal stadig benytte MockUserStories.

  Vær opmærksom på at interfacet kan være lidt anderledes end din implementering af Delete (DeleteUserStory) og Get (GetUserStory) – så må du lave det som interfacet siger

- Refaktorer i StartUp-klassen så det benytter interfacet:
  services.AddSingleton<**IUserStoryService**, UserStoryService>();
  ^^^^^^^^^^^^^^^^

- Refaktorer i de to ModelView (controller) dvs. index.cshtml.cs og DeleteUserStory.cshtml.cs filer

  Instans feltet skal nu være (benytte interfacet)
  private IUserStoryService _userStoryService

  samt konstruktøreren (benytte interfacet)
  public IndexModel(IUserStoryService userStoryService)

- Du skal evt. i OnGet hhv. OnPost ændre i navnet på metoderne kaldet på _userStoryService

- Prøv at køre projektet – det skulle gerne virke lige så fint ny efter refaktoreringen.

Understøt Json

Du skal nu til at under støtte json i stedet for MockUserStories.
(se Razor Note 1 : kap 11 Repository Design Pattern ss. 157-160)

- I mappen 'Services skal du implementere en ny klasse 'UserStoryServicesJson', der ligeledes skal implementere interfacet 'IUserStoryService'

- I ScrumBoard projektet lav endnu en mappe (folder) 'util' til hjælpe klasser
  højre-klik på projektet of add new folder

  I den nye mappe opretter du en klasse 'JsonFileReader' (se i noten s. 158), der har en metode 'ReadJson' a.la.

```
public static List<UserStory> ReadJson(string fileName)
{
   /*
    * if first time (no file exists) - create a file with an empty list
    */
   if (!File.Exists(fileName))
   {
      JsonFileWriter.WriteToJson(new List<UserStory>(), fileName);
   }

   using (var jReader = File.OpenText(fileName))
   {
      return JsonSerializer.Deserialize<List<UserStory>>(jReader.ReadToEnd());
   }
}
```

  I den nye mappe opretter du en klasse 'JsonFileWriter' (se i noten s. 158), der har en metode 'WriteToJson' a.la.

```
public static void WriteToJson(List<UserStory> liste, string JsonFileName)
      {
         using (FileStream fs = File.Create(JsonFileName))
         {
            var writter = new Utf8JsonWriter(fs, new JsonWriterOptions()
               {
                  SkipValidation = false,
                  Indented = true
               }
            );
            JsonSerializer.Serialize<UserStory[]>(writter, liste.ToArray());
         }
      }
```

- Når du implementerer UserStoryServiceJson skal du gøre brug af dr to hjælpe klasser.
  Husk at definere en sti til din fil a.la
  private const string fileName = @" … fil navnet …";

- I StartUp-klassen gør brug af din nye klasse UserStoryServiceJson – du beholder interfacet men udskifter servicen
  services.AddSingleton<IUserStoryService, **UserStoryServiceJson**>();

- Prøv at køre projektet – det skulle gerne virke lige så fint ny efter refaktoreringen.
  Selvom applikationen nu gemme i en fil i json format

Understøt Generisk Service

Dette gøres i to trin

- Json hjælpe klasser laves generisk
- Ekstra: Servicen laves generisk

Json Service Generic

I stedet for at dine Json hjælpe klasser kun virker til UserStories skal du nu lave en generisk version som kan benyttes til at læse alle slags klasser.

I JsonFileReader

- Lav en ny metode som i stedet for **List<UserStory> ReadJson(string fileName)** gør brug af det generiske type oftest kaldet T: **List<T> ReadJsonGeneric<T>(string fileName).**
  Læg mærke til 1) returtypen er nu List<T> altså returnere lister af forskellige typer afhængig af kaldet og 2) efter metode navnet gøres plads til at angive typen (Det kender du fra JsonSerializer.Deserialize<…>)

- De to steder i metoden, hvor der står UserStory erstattes typen med den generiske T.

I JsonFileWriter

- Lav en ny metode **void WriteToJsonGeneric<T>(List<T> liste, string JsonFileName)**
- Det sted i metoden, hvor der står UserStory erstattes typen med den generiske T.

Ændre din UserStoryServiceJson til at benytte de generiske metoder

- I konstruktøren skal du anvende den nye ReadJsonGeneric:
  JsonFileReader.ReadJsonGeneric<UserStory>(FileName);

- Når du gemmer skal du anvende
  JsonFileWriter.WriteToJsonGeneric(_dataBuffer, FileName);

  Læg mærke til at du ikke her direkte skriver typen – Den er givet ved typen af _dataBuffer (her list<UserStory>)

Extra : Lav Servicen generisk

Kig på UserStoryService, det meste er lige til at lave generisk, dog kræver det at de klasse der håndteres af servicen alle har et id (GetById).

Sikre alle model-klasser har et id

- Lav et interface fx **IModelWithId**, der definere en property Id (int) med en get hhv. set metode. Dette interface skal laves i ScrumBoardLib i mappen model

  Du skal lade UserStory implementere dette interface.

Lave et Generisk interface og service

- Du lavede et interface til IUserStoryService tidligere, Nu skal du lave et nyt interface, som er generisk fx **IServiceGeneric,** der ligger i service mappen
  Alle hvor der stod UserStory skal der nu stå T og selve klassen skal have markeret at det er en generisk klasse:
  public interface IServiceGeneric<T> where T : IModelWithId

  Læg mærke til der står where T : IModelWithId dvs. der er begrænsninger på hvilke typer der kan anvendes, de skal have implementeret IModelWithId, dvs klasserne skal have et Id (hvilket netop er hvad vi ønsker)

- Lav en ny klasse i service mappen **ServiceGeneric**, der implementere Interfacet **IServiceGeneric**

  Lav klassen i det store og hele som UserStoryServiceJson blot hvor UserStory er erstattet af den generiske type T.

Afprøv den generiske klassse

- Afprøv klassen ved at erstatte UserStoryServiceJson med ServiceGeneric<UserStory> i startup-klassen.

- Tilpas de to page-controllere (viewmodel) index.cshtml.cs og DeleteUserStory.cshtml.cs til at modtage IServiceGeneric i konstruktøren