


Developing ASP.NET MVC 5 Models

Exercise 1: Creating an MVC Project and Adding a Model

Task 1: Create a new MVC project

1. Copy starter folder from MNS/Kursus/→Today's Folder → Opgave to your local machine
2. Create a new ASP.NET Web Application (.net framework) Empty MVC project.
Name it PhotoSharingApplication 

Task 2: Add a new MVC model

1. In the Solution Explorer, Right-click Models, point to add, and then click Class, name Photo and click add.
2. Add another Class – named Comment,

Exercise 2: Adding Properties to MVC Models

Task 1: Add public scope properties to the Photo model class

1. open Photo.cs.
2. In the Photo class of the Photo.cs code window, add following Read and write properties

Property name: PhotoID

Data type: integer

Property name: Title

Data type: string

Property names: PhotoFile

Data type for the image: byte []

Property name ImageMimeType

Data type for MIME type: string

Property name: Description

Data type: String

Property name: CreatedDate

Data type: DateTime

Property name: UserName

Data type: string

Task 2: Add properties to the Comment model class

1. open Comment.cs.
2. In the Comment class of the Comment.cs code window, add following Read and write properties

Property name: CommentID

Data type: integer

Property name: PhotoID

Data type: integer

Property name: UserName

Data type: string

Property name: Subject

Data type: string

Property name: Body

Data type: string

Task 3: Implement a relationship between model classes

1. Add a new property to the Photo model class to retrieve comments for a given photo by using the following information:

Scope: public

Property name: Comments

Data type: a collection of Comments

Access: Read and write

Include the virtual keyword

2. Add a new property to the Comment model class to retrieve the photo for a given comment by using the following information:

Scope: public

Property name: Photo

Property type: Photo

Access: Read and write

Include the virtual keyword

Exercise 3: Using Data Annotations in MVC Models

Task 1: Add display and edit data annotations to the model.

1. Open Photo.cs.

2. Add a display data annotation to the Photo model class to ensure that the PhotoFile property is displayed with the name, Picture. → [DisplayName("Picture")]
3. you need to add → using System.ComponentModel; and using System.ComponentModel.DataAnnotations;
4. Add an edit data annotation to the Photo model class that ensures the Description property editor is a multiline text box. → [DataType(DataType.MultilineText)]
5. Add the following data annotations to the Photo model class to describe the CreatedDate property:


```
[DataType(DataType.DateTime)]
[DisplayName("Created Date")]
[DisplayFormat(DataFormatString="{0:MM/dd/yy}")]
```
6. Open Comment.cs.
7. Add an edit data annotation to the Comment model class that ensures that the Body property editor is a multiline text box. → [DataType(DataType.MultilineText)]
9. you need to add → using System.ComponentModel.DataAnnotations;

Task 2: Add validation data annotations to the model.

1. In Photo.cs.
2. In the Photo.cs, Title is required → [Required]
3. In Comment.cs, Subject is required and max string length 250
→ [Required]
[StringLength(250)]

Exercise 4: SQL Database

Task 1: Add an Entity Framework Context to the model

1. Click Project menu in Visual Studio and click Manage NuGet Packages.
2. In the navigation pane of the PhotoSharingApplication - Manage NuGet Packages window, click Browse, click EntityFramework, and then click Install.
3. On the License Acceptance page, click I Accept.
4. In the PhotoSharingApplication - Manage NuGet Packages window, click Close.
5. In the Solution Explorer pane, right-click Models, point to Add, and then click Class.
6. In the Name box write PhotoSharingContext, and then click Add.
7. add → using System.Data.Entity;
8. PhotoSharingContext inherits : DbContext
10. In the PhotoSharingContext class, add 2 properties

Property name: Photos
Data type: DbSet<Photo>
Access: Read and write

Property name: Comments
Data type: DbSet<Comment>
Access: Read and write

Task 2: Add an Entity Framework Initializer

1. In the Solution Explorer pane of the PhotoSharingApplication - Microsoft Visual Studio window, rightclick Models, point to Add, and then click Class.
2. In the Name box of the Add New Item - PhotoSharingApplication dialog box, type PhotoSharingInitializer, and then click Add.
3. add → using System.Data.Entity; and using System.IO;
4. PhotoSharingInitializer.cs inherits : DropCreateDatabaseAlways<PhotoSharingContext>
6. You need to add following getFileBytes method in this class to seed images

```
//This gets a byte array for a file at the path specified
//The path is relative to the route of the web site //It is used to seed images
private byte[] getFileBytes(string path)
{
    FileStream fileOnDisk = new FileStream(HttpRuntime.AppDomainAppPath + path, FileMode.Open);
    byte[] fileBytes;
    using (BinaryReader br = new BinaryReader(fileOnDisk))
    {
        fileBytes = br.ReadBytes((int)fileOnDisk.Length);
    }
    return fileBytes;
}
```

7. In a new line in the PhotoSharingInitializer class, type→ override
press Spacebar, and then click Seed(PhotoSharingContext context).
12. In the Seed method, place the mouse cursor after the call to base.Seed, press Enter twice, and then type the following code.

```
var photos = new List<Photo>
{
    new Photo {
        Title = "Test Photo",
        Description = "Your
        Description",
        UserName = "NaokiSato",
        PhotoFile = getFileBytes
        ("\\Images\\flower.jpg"), ImageMimeType = "image/jpeg",
        CreatedDate = DateTime.Today
    }
};
```

13. Place the mouse cursor at the end of the list of Photo objects, press Enter twice, and then type the following code.

```
photos.ForEach(s => context.Photos.Add(s));
context.SaveChanges();
```

14. Place the mouse cursor at the end of the Entity Framework context code, press Enter twice, and then type the following code.

```
var comments = new List<Comment>
{
    new Comment {
        PhotoID = 1,
        UserName = "NaokiSato",
        Subject = "Test Comment",
        Body = "This comment " +
            "should appear in " +
            "photo 1"
    }
};
```

15. Place the mouse cursor at the end of the list of Comment objects, press Enter twice, and then type the following code.

```
comments.ForEach(s => context.Comments.Add(s));
context.SaveChanges();
```

16. In the Solution Explorer pane, click Global.asax.

17. In the Global.asax code window, add

```
using System.Data.Entity;
using PhotoSharingApplication.Models;
```

18. In the Application Start method code block, type the following code.

```
Database.SetInitializer(new PhotoSharingInitializer());
```

Task 3: Add connection string to web config.

```
<connectionStrings>

    <add name=" PhotoSharingContext" connectionString="Data
Source=(LocalDB)\MSSQLLocaldb.0;AttachDbFilename=|DataDirectory|\
PhotoSharingContext.mdf;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>
```

Exercise 5: Testing the Model and Database

Task 1: Add a controller and views.

1. In the Solution Explorer pane of the PhotoSharingApplication right-click Controllers, point to Add, and then click Controller.
2. In the Template list, click MVC Controller with ~~read/write actions and views~~, using Entity Framework.
3. In the Model class list, click Photo (PhotoSharingApplication.Models).
4. In the Data context class list, click PhotoSharingContext (PhotoSharingApplication.Models).
5. In the Controller name box , type PhotoController.

Task 2: Add an image and run the application.

1. Find \Images folder , you copied from MNS/Kursus Folder
2. Drage drop Images folder on Project Name
3. Run the Application in browser if you get error then append the existing URL with photo/index, and press Enter.
4. On the Index page, click the Details link.
Note: The details of the added image such as the image type, image description, and image creation date are displayed on the Details page. The scaffold templates do not display the image itself. You will see how to display images in later labs.