Screenshots ▾

Module 2: Designing ASP.NET Core Web Applications

Lab: Designing ASP.NET Core Web Applications

Scenario

Your team has chosen ASP.NET Core MVC as the most appropriate ASP.NET programming model to create the photo sharing application for the Adventure Works web application. You need to create a detailed project design for the application and have been given a set of functional and technical requirements with other information. You have to plan:

- An MVC model that you can use to implement the desired functionality.
- One or more controllers and controller actions that respond to users actions.
- A set of views to implement the user interface.
- The locations for hosting and data storage.

Exercise 1: Planning Model Classes

Scenario

You need to recommend an MVC model that is required to implement a photo sharing application. You will propose model classes based on the results of an initial investigation into the requirements.

The main tasks for this exercise are as follows:

- Examine the initial investigation
- Plan the photo model class
- Plan the comment model class

Task 1: Examine the initial investigation

□ 1. Go to **D:\Allfiles\Mod02\Labfiles\01_DesignProject_begin** and open **InitialInvestigation.docx**.

□ 2. In the **InitialInvestigation - Microsoft Word** window, in the **View** tab, in the **Show** section, ensure that the **Navigation Pane** check box is selected.

□ 3. In the **Navigation** pane, click **Introduction** and read the content.

□ 4. In the **Navigation** pane, click **General Description of the Photo Sharing Web Application** and

□ 5. In the **Navigation** pane, click **Use Cases** and read the content.

□ 6. In the **InitialInvestigation - Microsoft Word** window, review **Figure 1: Use Case Summary**.

□ 7. In the **InitialInvestigation - Microsoft Word** window, click **Close**.

Task 2: Plan the photo model class

□ 1. Go to **D:\Allfiles\Mod02\Labfiles\01_DesignProject_begin** and open **DetailedPlanningDocument.docx**.

□ 2. In the **DetailedPlanningDocument - Microsoft Word** window, locate the **MVC Model** section.

□ 3. In the **MVC Model** section, locate **Table 1: MVC Model**.

□ 4. In **Table 1: MVC Model**, in row 1, in the **Model Class** column, type Photo.

□ 5. In row 1, in the **Description** column, type:

```
The photo model class represents a photo that authenticated users can upload to the
```

□ 6. In row 1, in the **Properties** column, type PhotoID.

□ 7. In row 1, in the **Data Types** column, type Integer.

□ 8. In row 2, in the **Properties** column, type Title.

□ 9. In row 2, in the **Data Types** column, type String.

□ 10. Add the following additional properties and data types in the same way: **PhotoFile - Binary**, **Description - String**, **CreatedDate - DateTime**, **Owner - String**.

□ 11. In **Table 1: MVC Model**, in the **Model Class** column, select cells 1 to 6, right-click, and then select **Merge Cells**.

□ 12. In **Table 1: MVC Model**, in the **Description** column, select cells 1 to 6, right-click, and then select **Merge Cells**.

> **Note**: Keep the **DetailedPlanningDocument - Microsoft Word** window open for future tasks in this lab.

□ 13. Start **Microsoft Visual Studio**.

□ 14. In the **Start Page - Microsoft Visual Studio** window, on the **File** menu, point to **New**, and then click **Project**.

□ 16. In the **Create a new project** dialog box, choose **ASP.NET Core Web App (Model-View-Controller)** and click **Next**.

□ 17. In the **Project Name** text box, type **DesignProject** and click **Next**.

□ 18. In the **Additional information** dialog box, select **.NET 6.0 (Long-term support)** from the dropdown at the top of the dialog box.

□ 19. Click on **Create**.

□ 20. In the **DesignProject - Microsoft Visual Studio** window, in Solution Explorer, right-click the **Models** folder, point to **Add**, and then select **Class...**.

□ 21. In the **Add New Item - DesignProject** dialog box, in the **Name** box, type **Photo**, and then click **Add**.

□ 22. In the **Photo.cs** file, after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public int PhotoID { get; set; }
public string Title { get; set; }
public byte[] PhotoFile { get; set; }
public string Description { get; set; }
public DateTime CreatedDate { get; set; }
public string Owner { get; set; }
```

> ℹ You have now created a sample model class based on your **Photo** class design.

Task 3: Plan the comment model class

□ 1. Go back to the **DetailedPlanningDocument.docx** file, which should still be open in Microsoft Word.

□ 2. In the **DetailedPlanningDocument - Microsoft Word** window, locate the **MVC Model** section.

□ 3. In the **DetailedPlanningDocument - Microsoft Word** window, locate the **Table 1: MVC Model** table.

□ 4. In **Table 1: MVC Model**, in row 7, in the **Model Class** column, type **Comment**.

□ 5. In row 7, in the **Description** column, type:

```
The comment model class represents a comment that authenticated users can add to ph
```

□ 6. In row 7, in the **Properties** column, type **CommentID**.

□ 8. In row 8, in the **Properties** column, type **User**.

□ 9. In row 8, in the **Data Types** column, type **String**.

□ 10. Add the following additional properties and data types in the same way: **Subject - String**, **Body - String**, **PhotoID - Integer**.

□ 11. In **Table 1: MVC Model**, in the **Model Class** column, select cells 7 to 11, right-click, and then select **Merge Cells**.

□ 12. In **Table 1: MVC Model**, in the **Description** column, select cells 7 to 11, right-click, and then select **Merge Cells**.

□ 13. Go back to **DesignProject**, which should still be open in Microsoft Visual Studio.

□ 14. In the **DesignProject - Microsoft Visual Studio** window, in Solution Explorer, right-click the **Models** folder, point to **Add**, and then select **Class...**.

□ 15. In the **Add New Item - DesignProject** dialog box, in the **Name** box, type **Comment**, and then click **Add**.

□ 16. In the **Comment.cs** file, after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public int CommentID { get; set; }
public int User { get; set; }
public string Owner { get; set; }
public string Subject { get; set; }
public string Body { get; set; }
public int PhotoID { get; set; }
```

> You have now created a sample model class based on your **Comment** class design. We now need to connect the two classes to create a one-to-many relationship between **Photo** and **Comment** classes.

□ 17. In the **Photo.cs** file, add a further property at the end of the list of properties and before the **}** (closing brace) sign, by typing the following code:

```
public List<Comment> Comments { get; set; }
```

□ 18. on the **File** menu of the **DesignProject - Microsoft Visual Studio** window, click **Save All**.

configure the properties and data types of the model classes.

Exercise 2: Planning Controllers

Scenario

You need to recommend a set of MVC controllers that are required to implement a photo sharing application. You will propose controllers based on the results of an initial investigation into the requirements.

The main tasks for this exercise are as follows:

- Plan the photo controller
- Plan the comment controller

Task 1: Plan the photo controller

□ 1. On the taskbar, click **DetailedPlanningDocument - Microsoft Word**.

□ 2. In the **DetailedPlanningDocument - Microsoft Word** window, locate **Table 2: MVC Controllers** in the **MVC Controllers** section.

□ 3. In **Table 2: MVC Controllers**, in row 1, in the **Controller** column, type PhotoController.

□ 4. In row 1, in the **Action** column, type DisplayGallery (GET).

□ 5. In row 1, in the **Description** column, type **The action runs when the user requests the Photo Gallery page. The action obtains all the photos from the database and passes them to the DisplayGallery view.**

□ 6. In row 2, in the **Action** column, type DisplayRecent (GET).

□ 7. In row 2, in the **Description** column, type:

```
This action is similar to the DisplayGallery action except that only the most recen
```

□ 8. In row 3, in the **Action** column, type DisplayPhoto (GET).

□ 9. In row 3, in the **Description** column, type:

```
This action runs when the user clicks a photo's "Details" link in a gallery. The ac
```

□ 10. In row 4, in the **Action** column, type AddPhoto (GET).

```
This action runs when the user clicks the "Add a Photo" link. The action creates a
```

□ 12. In row 5, in the **Action** column, type AddPhoto (POST).

□ 13. In row 5, in the **Description** column, type:

~~~
This action runs when the user clicks "Save" in the AddPhoto view. The action saves
        ```
~~~

□ 14. In row 6, in the **Action** column, type DeletePhoto (GET).

□ 15. In row 6, in the **Description** column, type:

```
This action runs when the user clicks a "Delete this Photo" link in the DisplayPhot
```

□ 16. In row 7, in the **Action** column, type DeletePhoto (POST).

□ 17. In row 7, in the **Description** column, type:

```
This action runs when the user clicks "Delete" in the DeletePhoto view. The action
```

□ 18. In **Table 2: MVC Controllers**, in the **Controller** column, select cells 1 to 7, right-click, and then select **Merge Cells**.

Task 2: Plan the comment controller

□ 1. In the **Table 2: MVC Controllers** table, in the **Controller** column, in row 8, below **PhotoController**, type CommentController.

□ 2. In **Table 2: MVC Controllers**, in row 8, in the **Action** column, type DisplayComments (GET).

□ 3. In row 8, in the **Description** column, type:

```
This action runs when the DisplayPhoto view is displayed. The action requires the c
```

□ 4. In row 9, in the **Action** column, type AddComment (GET).

□ 5. In row 9, in the **Description** column, type:

```
This action runs when the user clicks the "Add a Comment" link in the DisplayPhoto
```

□ 6. In row 10, in the **Action** column, type AddComment (POST).

□ 7. In row 10, in the **Description** column, type This action runs when the user clicks "Submit" in the

□ 8. In **Table 2: MVC Controllers**, in the **Controller** column, select cells 8 to 10, right-click, and then select **Merge Cells**.

□ 9. on the **File** menu of the **DetailedPlanningDocument - Word** window, click **Save**.

✓ **Results**: After completing this exercise, you will be able to create proposals for controllers and configure their properties and data types.

Exercise 3: Planning Views

Scenario

You need to recommend a set of MVC views that are required to implement a photo sharing application. You will propose views based on the results of an initial investigation into the requirement.

The main tasks for this exercise are as follows:

- Define the views
- Design the single photo view
- Design the gallery view

Task 1: Define the views

□ 1. In the **DetailedPlanningDocument**, which should still be open in Microsoft Word, locate **MVC Views** section, and then locate **Table 3: MVC Views**.

□ 2. In row 1, in the **Controller** column, type PhotoController.

□ 3. In row 1, in the **View** column, type DisplayGallery.

□ 4. In row 1, in the **Description** column, type:

```
This view displays a collection of photos in the thumbnail size. For each photo the
```

□ 5. In row 2, in the **View** column, type DisplayPhoto.

□ 6. In row 2, in the **Description** column, type:

```
This view displays a single photo in full size. The Title and Owner values appear a
```

□ 7. In row 3, in the **View** column, type AddPhoto.

```
This view displays a form that the user can use to upload and describe a new photo.
```

□ 9. In row 4, in the **View** column, type DeletePhoto.

□ 10. In row 4, in the **Description** column, type:

```
This view displays a form that the user can use to confirm the deletion of a photo.
```

□ 11. In **Table 3: MVC Views**, in the **Controller** column, select cells 1 to 4, right-click, and then select **Merge Cells**.

□ 12. In row 5, in the **Controller** column, type CommentController.

□ 13. In row 5, in the **View** column, type DisplayComments.

□ 14. In row 5, in the **Description** column, type:

```
This partial view, which is used on the DisplayPhoto form, displays all the comment
```

□ 15. In row 6, in the **View** column, type AddComment.

□ 16. In row 6, in the **Description** column, type:

```
This view displays a form that the user can use to create a new comment for a photo
```

□ 17. In **Table 3: MVC Views**, in the **Controller** column, select cells 5 to 6, right-click, and then select **Merge Cells**.

□ 18. In the **DetailedPlanningDocument - Word** window, in the **FILE** menu, click **Save**.

Task 2: Design the single photo view

□ 1. Go to **D:\Allfiles\Mod02\Labfiles\01_DesignProject_begin** and open **Wireframe.pptx**.

□ 2. Notice that the first slide in the **Wireframe.pptx** slide deck is a wireframe mock-up of the Adventure Works photo sharing site landing page. Use this as an example.

> ℹ **Note:** There are many website design tools for wireframing. The intention is to design the layout and flow of the site without details of the graphical design, to facilitate technical development. If you already have access to a preferred wireframing tool such as Figma, Sketch, Balsamiq, etc., you can use that instead. Similarly, if you prefer to sketch freehand using pencil and paper or a graphics tablet, that is also a great way of completing the exercise. The purpose of the exercise is to come up with a rough page (view) design.

Note: You can use the editing tools, text boxes, shapes, etc. in PowerPoint, or re-use some of the assets on slide 4 in the deck.

□ 4. If needed, you can look at the corresponding slide in the **Wireframe.pptx** file in **D:\Allfiles\Mod02\Labfiles\01_DesignProject_end**. Remember, this is just one possible design.

□ 5. Leave the **Wireframe.pptx** PowerPoint deck open for the next task.

Task 3: Design the gallery view

□ 1. In **Wireframe.pptx**, edit the third slide in the deck to create the gallery view, based on the requirements in the **DetailedPlanningDocument**.

□ 2. If needed, you can look at the corresponding slide in the **Wireframe.pptx** file in **D:\Allfiles\Mod02\Labfiles\01_DesignProject_end**. Remember, this is just one possible design.

✓ **Results**: After completing this exercise, you have created a proposal for views and their layouts in a form that can be presented to stakeholders for further input.

Exercise 4: Architecting an MVC Web Application

Scenario

You need to recommend a web server and database server configuration that is required to implement a photo sharing application. You will propose details based on the results of an initial investigation into the requirements.

The main tasks for this exercise are as follows:

- Hosting options.
- Choosing a data store.

Task 1: Hosting option

□ 1. On the taskbar, click the **DetailedPlanningDocument - Microsoft Word** icon.

□ 2. In the **DetailedPlanningDocument - Microsoft Word** window, locate the **Hosting Recommendations** section, and then locate the **Web Server** section.

□ 3. Under **Web Server**, type the follwing text:

go deploy ☰

Task 2: Choose a data store

□ 1. In the **DetailedPlanningDocument - Microsoft Word** window, in the **Database** section, type the following text:

```
The author recommends using a SQL Database, within Microsoft Azure, to host the Pho
```

□ 2. In the **DetailedPlanningDocument - Microsoft Word** window, click **Close**.

□ 3. In **Microsoft word** dialogue box, click **Save** to save the changes.

✓ **Results**: After completing this exercise, you will be able to create proposals for hosting arrangements.