**ASP.NET MVC with Razor Exercise: To-Do List Application**

**Objective**:

Build a basic to-do list application using ASP.NET MVC with Razor views.

**Instructions:**

**Model:**

Define a model named TodoItem with the following properties:

1. Id (int): Unique identifier for the task.
2. Task (string): Description of the task.
3. IsCompleted (bool): Flag indicating whether the task is completed.

**Controller:**

Create a controller named TodoController to manage the to-do list. Implement the following actions:

1. Index: Display the list of tasks.
2. Add: Add a new task to the list. (HTTP POST method)
3. Complete: Mark a task as completed. (HTTP POST method)
4. Delete: Remove a task from the list. (HTTP POST method)

**Views:**

Create Razor views to implement the user interface for the to-do list application. Follow these guidelines:

1. Display the list of tasks with appropriate UI elements for completion and deletion.
2. Include a form for adding new tasks.

**Layout Master Page:**

Implement a layout master page to maintain a consistent UI across all views. Include:

1. Header section with application title.
2. Navigation bar or menu (optional).
3. Footer section with copyright information.

**Partial View:**

Utilize a partial view to render the list of tasks. Create a separate partial view for displaying individual tasks within the list. Include appropriate UI elements for completion and deletion.

**Custom HTML Helper:**

Create a custom HTML helper to generate HTML elements for displaying and interacting with to-do list items. The helper should simplify the rendering of tasks and reduce code duplication.

**Custom Routing:**

Implement custom routing to define user-friendly URLs for the application. Configure routes to handle requests for different actions within the TodoController.

**Anti-Forgery Token:**

Integrate Anti-Forgery Token mechanism to prevent Cross-Site Request Forgery (CSRF) attacks in the application. Ensure that the token is included in all forms that submit data to the server.

**Entity Framework Code First Approach:**

Implement Entity Framework Code First approach to manage the database. Define a DbContext class and configure it to work with the TodoItem model. Ensure proper database migrations and seed data.

**Entity Framework Data First Approach:**

Implement Entity Framework Data First approach to manage the database. Generate the database schema from an existing database and configure the DbContext class accordingly. Ensure proper database migrations and seed data.

**Data Validation:**

Implement data validation for the TodoItem model using Data Annotations. Use attributes such as Required, StringLength, etc., to enforce validation rules and improve data integrity.

**DataType Attributes:**

Utilize DataType attributes such as **DataType.Date**, **DataType.EmailAddress**, etc., to specify the type of data expected for certain model properties. This enhances data validation and improves user input handling.

**HTTP Methods - Best Practices:**

Implement HTTP methods best practices in the **TodoController**. Ensure that:

**Add**, **Complete**, and **Delete** actions are implemented as HTTP POST methods to prevent unintentional actions through GET requests.

**Functionality:**

Ensure the following functionality works correctly:

1. Users can add new tasks to the to-do list.
2. Users can mark tasks as completed.
3. Users can delete tasks from the list.

**Styling and UI:**

Apply basic styling to improve the appearance of the application. Ensure the UI is user-friendly and intuitive.

**Testing and Debugging:**

Thoroughly test the application to ensure all features work as expected. Debug any issues that arise during testing.

**Submission:**

Submit the following for evaluation:

* Complete source code files (Models, Controllers, Views, Layout Master Page, Partial View, Custom HTML Helper).
* Screenshots or a video demonstrating the functionality of the application.
* Any additional notes or documentation.
* A README file explaining how to run the application, any prerequisites, and any additional notes or documentation.

**Note:**

* Pay attention to code organization, naming conventions, and best practices.
* Creativity and additional features are encouraged for bonus points.

**Deadline:** [15-02-2024]