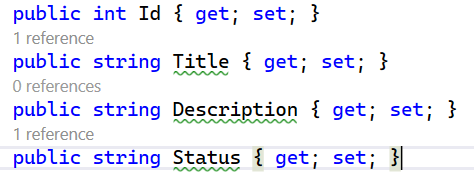
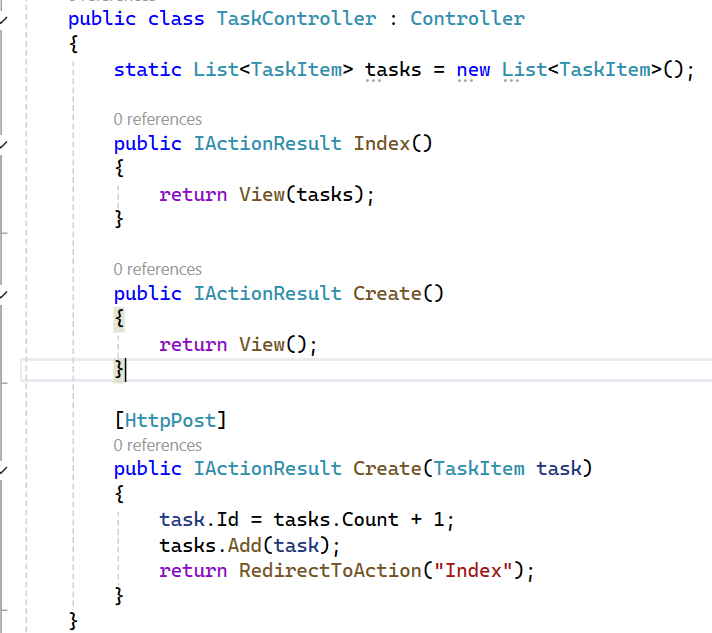
**Phase 1**

1. Create a new project → ASP.NET Core Web Application (.NET Core) MVC
2. Create a class TaskItem.cs inside Models folder -> Just define the fields.



1. Create a new controller TaskController inside Controllers folder -> Code to dummy task list.



1. Inside Views, create folder Task. Inside Task folder, create Index.cshtml and create.cshtml.

|  |  |
| --- | --- |
|  |  |

1. Run and set your browser to start with URL: /Task/Index.
2. Add model validations – Required and StringLength in Models file, along with using System.ComponentModel.DataAnnotations. Then, @Html.ValidationSummary() and @Html.ValidationMessage("Title") in Views files. Then, if (!ModelState.IsValid) in Controller Post method.
3. Add Update and Delete Functionalities.
4. Push this initial code to GitHub –

* View -> Git Changes -> Create Git repository -> Repository location: Same as your project folder, Add .gitignore for Visual Studio (Recommended) -> Sign in with GitHub -> Create and Publish -> Check your GitHub account.
* After any changes -> All changes will automatically be staged in VS -> Just write commit message -> Commit All and Push.

Note:

* User -> /Task/Index -> Controller (Index) -> View (Index.cshtml) shows list.

User clicks "Add Task" -> /Task/Create (GET) -> Controller (Create) -> View (Create.cshtml).

User fills form -> Form submits (POST) -> Controller (Create - POST) -> Adds to tasks -> Redirect to Index.

In model, we are creating a class 'TaskItem', in that we are defining properties instead of variables due to their advantages. In Controller, in that we are importing the models class, and we are creating another class and inherting it from the controller base class for it to work with http stuff. In that, we are temporarily creating a list to store the tasks instead of db. Then, since we want to display the tasks and add tasks, we have to use get() and post(). And for display - we have to display list of tasks and also an empty form. So, we created 2 views for each, and in get() we put those views with an IActionResult type. In post we gave the action to be done and redirected to index page.

* wwwroot is the root folder for static files (CSS, JS, Images).

**Phase 2**