**ASP.NET Core 2.0 MVC – Sources5**

begincodingnow.com

Date: 2019-10-14

File Name: Sources5Documentation.docx

**Project Name:**

ASP.NET Core 2.0 MVC with Identity Framework Individual User Accounts

**Requirements**

**Sources of Information**

**Steps**

➤

**Projects Link**

The user has clicked the **Projects** link in the main menu. They have already logged in.

Controller: Project

Action: Index

**public IActionResult Index()**

Get the current user

If not NULL, then get user’s list of projects and return **Welcome** view with list of projects.

Otherwise redirect to action **Create** in the **UserInfo** controller.

**Welcome.cshtml under Views/Project**

**A screenshot of a cell phone

Description automatically generated**

@model: IEnumerable<Project>

**Create New Project button**

Controller: Project

Action: Create

This button is just a link <a>, not a form.

@foreach (var p in Model)

The Tasks, Edit and Delete buttons are in their **own forms**, inside the table.

|  |  |
| --- | --- |
| **Tasks button**  Controller: ProjectTask  Action: TaskListForProject  Method: post  Type: text  Name: pId  Value: @p.Id  hidden | **Edit button**  Controller: Project  Action: Edit  Method: post  Type: text  Name: Id  Value: @p.Id  hidden |
|  | **Delete button**  Controller: Project  Action: Delete  Method: Post  Type: text  Name: Id  Value: @p.Id  hidden |

**Create New Project**

The user has clicked the Create New Project button in the Welcome view.

Controller: Project

Action: Create

This button is just a link <a>, not a form.

public IActionResult Create()

1. Get the current user Id from Identity
2. Get the user Id from our table UserInfos
3. Create a new Project and initialize the UserInfoId column
4. Return the view **ProjectForm** and pass the **Project** model

**ProjectForm.cshtml in Views/Project**

A screenshot of a cell phone

Description automatically generated

@model: Project

If the Model.Id is zero then we are creating a new project, otherwise we are editing an existing project.

**HTML Form**

The Save and Cancel buttons are inside the <form>.

**Save button**

Type: submit

Controller: Project

Action: Save

Method: post

**Cancel button**

Controller: Project

Action: Index

This is just an anchor link inside the form.

**Save Project**

The user has clicked the **Save** button in the Create New Project form. The Save button is a submit button that is posting to the server.

**public IActionResult Save(Project proj)**

If the ModelState is **not** valid, go back and return the ProjectForm and the Project model. We received the project model so we can simply send it back.

If the ModelState is valid, then we will either add a new project or edit an existing project based on the value of the **Id** field in the Project object. If the Id is zero, then this is a new project. If the Id is not zero then we are editing an existing project.

**New Project (Id is 0)**

Create a new Project object and use object initializer syntax to initialize all the properties except for these two:

* Id - this is an Identity in Db - do not initialize it - the Db will
* UserInfo - this is a navigation property

Add the object to the database

Save Changes

**Edit Project Id is not 0**

Get the project from the database based on the Project Id. (Single())

Manually set each of the properties in the Project

Save Changes

**Set TempData**

We got here because the model state was valid.

TempData["message"] = $"{proj.Name} has been saved"; // A. Freeman p 310 of Pro ASP.NET

**Return**

return RedirectToAction("Index");

We are in the Project controller, so we’ll go to the Index action which displays the list of projects for the current user.

**Edit Project**

The user has clicked the Edit button in the Projects list.

**A screenshot of a cell phone

Description automatically generated**

**Edit button**

Controller: Project

Action: Edit

Method: post

Type: text

Name: Id

Value: @p.Id

hidden

**Edit Project**

**public ActionResult Edit(int Id)**

Get the project row from the database table Projects

Create a new Project object and pass the project to the ProjectForm

**ProjectForm.cshtml**

A screenshot of a cell phone

Description automatically generated

The Save button is the save as the previous listing.

**Delete Project**

The user has clicked the Delete button in the list of Projects.

**Delete button**

Controller: Project

Action: Delete

Method: Post

Type: text

Name: Id

Value: @p.Id

hidden

**public IActionResult Delete(int Id)**

If the Id is not 0 then Find the project in the database

Remove the project

Save the Changes

Redirect to Action: Index

**Tasks**

The user has clicked the Tasks button beside the Pheonix project on the list of Projects page.

**A screenshot of a cell phone

Description automatically generated**

**Tasks button**

Controller: ProjectTask

Action: TaskListForProject

Method: post

Type: text

Name: pId

Value: @p.Id

hidden

**[HttpPost]**

**public IActionResult TaskListForProject(int pId)**

Get the Project from the database using the Id (SingleOrDefault())

If the project is null then return an Error.

If the Project was found then get the list of Tasks for the Project

Create a new view model object

Initialize the view model to include the Project and the list of tasks

Return the view **Welcome** along with the **view model**

**Welcome.cshtml in Views/ProjectTasks**

A screenshot of a cell phone

Description automatically generated

@model ProjectProjectTaskViewModel

The Create New Task, Edit and Delete buttons are in their **own forms**.

|  |  |
| --- | --- |
| **Create New Task button**  Controller: ProjectTask  Action: Create  Method: post  Type: text  Name: pId  Value: @Model.project.Id  hidden | **Edit button**  Controller: ProjectTask  Action: Edit  Method: post  Type: text  Name: Id  Value: @t.Id  hidden |
|  | **Delete button**  Controller: ProjectTask  Action: Delete  Method: Post  Type: text  Name: Id  Value: @t.Id  hidden |

**Create New Task**

The user has clicked the Create New Task button. It calls the Create action in the ProjectTask Controller.

**public ActionResult Create(int pId)**

Create a new ProjectTask object

Initialize the ProjectId property to the pId passed in.

Return the view ProjectTaskForm along with the Task object

We only need to know the project that we are creating the task for. We get the project Id passed in.

A screenshot of a cell phone

Description automatically generated