

ASP.NET Core 2.0 MVC – Sources5

begincodingnow.com

File Creation Date: 2019-10-14

File Name: Sources5Documentation.docx

Some graphics (wireframes) were created with Figma.

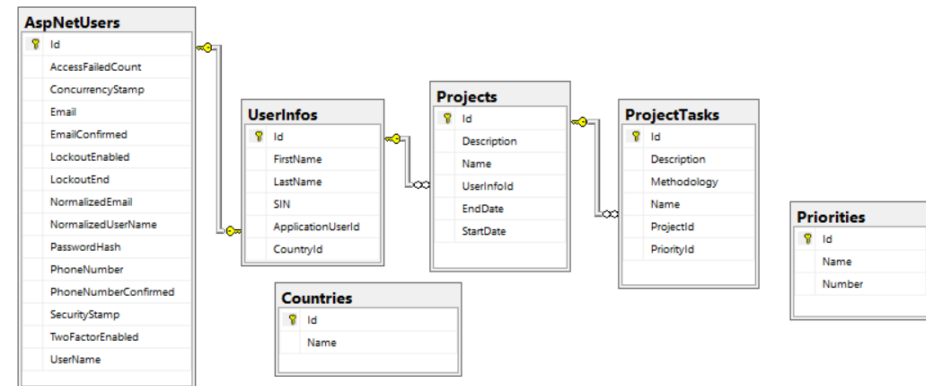
Project Name: Sources5

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

Requirements

This web app is a simple project management application that is really just a to-do list application. You need to Register with an email address and a password before you gain access to the ability to create projects and tasks for those projects. Also, you must create a “profile” after you register. This profile has some personal account information such as SIN number and your Country.

Database (in SQL Server)



Notice that there is no relationship in the database between ProjectTasks and Priorities. There is however a navigation property in the model ProjectTask:

```
public Priority Priority { get; set; }
```

The migration that creates the **link** from ProjectTasks to Priorities **fails**. The “solution” is to alter the migration and delete the code that tries to create the foreign key. Save it and update the database. Leave the code that creates the index.

Table	Relationship	Table
AppNetUsers	One to one	UserInfos
UserInfos	One-to-one	Countries
UserInfos	One to many	Projects
Projects	One to many	ProjectTasks
ProjectTasks	One to one	Priorities

ApplicationDbContext.cs file in the Data Folder

```
public DbSet<UserInfo> UserInfos { get; set; }
public DbSet<Project> Projects { get; set; }
public DbSet<ProjectTask> ProjectTasks { get; set; }
public DbSet<Country> Countries { get; set; }
public DbSet<Priority> Priorities { get; set; }
```

The following section lists the table names and their class names. When creating migrations, create the table(s) first, then create the relationships with their navigation properties and foreign key properties.

AppNetUsers

This table comes from installing Identity Framework.

UserInfos (class UserInfo)

```
[Required(ErrorMessage="Please enter a first name - max 100 characters")]
[MaxLength(100)]
[Display(Name = "First Name")]
public string FirstName { get; set; }

[Required(ErrorMessage = "Please enter a last name - max 100 characters")]
[MaxLength(100)]
[Display(Name = "Last Name")]
public string LastName { get; set; }

[MaxLength(12)]
public string SIN { get; set; }

public ApplicationUser ApplicationUser { get; set; }

public string ApplicationUserId { get; set; }

public Country Country { get; set; }

public int CountryId { get; set; }
```

Projects (class Project)

```
public int Id { get; set; }

[Required(ErrorMessage = "Please enter a project name - max 100 characters")]
[MaxLength(100)]
[Display(Name = "Project Name")]
public string Name { get; set; }

[MaxLength(2000)]
public string Description { get; set; }

[UIHint("Date")] // date without time component
[Display(Name="Start Date")]
public DateTime StartDate { get; set; }
```

```
[UIHint("Date")] // date without time component
[Display(Name = "End Date")]
public DateTime EndDate { get; set; }

public UserInfo UserInfo { get; set; }

public int UserInfoId { get; set; }
```

ProjectTasks (class ProjectTask)

```
public int Id { get; set; }

[Required]
[MaxLength(100)]
[Display(Name = "Task Name")]
public string Name { get; set; }

[MaxLength(1000)]
public string Description { get; set; }

[MaxLength(1000)]
public string Methodology { get; set; }

public Project Project { get; set; }
public int ProjectId { get; set; }

public Priority Priority { get; set; }

[Required(ErrorMessage = "Please choose a priority")]
[Range(1, 4, ErrorMessage = ("Please choose a priority from the list"))]
public byte PriorityId { get; set; }
// EF recognizes this one above as a foreign key ("Id")
// the Range is hard-coded here - not the best.
```

Countries (class Country)

```
public int Id { get; set; }
public string Name { get; set; }
```

Priorities (class Priority)

```
public byte Id { get; set; }
public string Name { get; set; }
public byte Number { get; set; }
```

View Models

This application requires us to create three view models. If a Form has a lookup table associated with it, we will need to create a view model.

```
public class UserInfoFormViewModel
{
    public UserInfo userinfo { get; set; }
    public IEnumerable<Country> countries { get; set; }
}
```

This view model is used by the Welcome.cshtml view in the Views/ProjectTask folder. This view uses a table to list the task for a particular project.

```
public class TasksViewModel
{
    // I need one project and a list of its tasks and Priorities
    // Used by Welcome.cshtml in Views/ProjectTask
    public Project project { get; set; }
    public IEnumerable<ProjectTask> tasks { get; set; }
    public IEnumerable<Priority> priorities { get; set; }
}
```

This view model is used by the ProjectTaskForm.cshtml view in the Views/ProjectTask folder.

```
public class TaskFormViewModel
{
    public Project project { get; set; }
    public ProjectTask task { get; set; }
    public IEnumerable<Priority> priorities { get; set; }
}
```

Points of Interest

This section highlights some interesting parts of this application.

- There are two reference (lookup) tables Countries and Priorities, which populate drop-downs on the GUI. Here is part of the GUI code that requires that we create a new SelectList object. `<select asp-for="userinfo.CountryId" class="form-control-sm" asp-items="@((new SelectList(Model.countries, "Id", "Name")))">`
- For the Project views and the ProjectTask views, there are only two views required: the Index.cshtml (or Welcome.cshtml in this project) and the Form view because both Create and Edit use the Form view. The controller contains the logic.
- All new users get added to the Users role. This is done in the Register action of the AccountController with this line: `result = await _userManager.AddToRoleAsync(user, "Users");`
-

Log In

Log in

Use a local account to log in.

Email

admin@example.com

Password

.....

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user?](#)

When the user clicks the Profile link on the main menu at the top of the application, they are taken to the Index action of the UserInfo controller. That gets the Id of the current user. We use that to get the Id of the current user in our UserInfos table. There is a one-to-one relationship between the AppNetUsers table provided by Identity and our table, UserInfos. If the user has already created a profile on this app, they are delivered the Welcome view, otherwise they are delivered the Create view.

Welcome Admin example.com

SIN: 123123123

Welcome.cshtml view in the UserInfo folder.

Edit Your Profile

If the user clicks the Edit Your Profile button,

Edit Your Profile

UserInfoForm.cshtml under Views/UserInfo

First Name

Admin

Last Name

example.com

SIN

123123123

Save

Cancel

Create Your Profile

First Name

Last Name

SIN

Save

Cancel

Projects Link in the Main Menu

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

_NavBar.cshtml in Views/Shared

```
<li><a class="nav-item nav-link text-dark" asp-controller="Project" asp-action="Index">Projects</a></li>
```

Controller: Project

Action: Index

Below is a screenshot of the main menu.

Company Inc. About Contact Profile Projects

Hello admin@example.com Logout

Project Controller

```
public IActionResult Index()
```

Get the current user (the assumption is that the user has already logged in.)

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.

List of Projects

```
IEnumerable<Project> projects = _context.Projects.Where(p => p.UserInfoId == usrinfo.Id).ToList();
```

Projects

The user has logged in and clicked the Projects link in the main menu. That takes them to the Index action which returns the Welcome view.

Welcome.cshtml under Views/Project



@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 object and initialize the UserInfoId column
4. Return the view **ProjectForm** and pass the **Project** model

```
public IActionResult Create()
{
    // The user has clicked the 'Create New Project' button in the Welcome view
    // This is just an anchor link <a> in Welcome.cshtml
    System.Security.Claims.ClaimsPrincipal currentUser = this.User;
    var Uid = userManager.GetUserId(User); // Get user id (string)
    // Get the row from our UserInfos table that is the current user's
    var userinfo = _context.UserInfos.SingleOrDefault(i => i.ApplicationUserId == Uid);
    var project = new Project
    {
        UserInfoId = userinfo.Id
    };
    return View("ProjectForm", project); // show the form
}
```

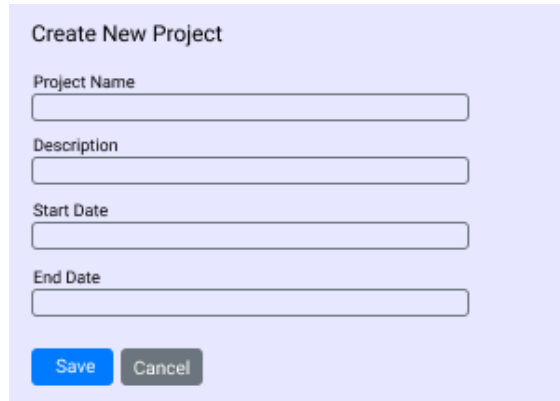
Improvements (optimization)

Could we make the Create New Project a <form>? We need the UserInfold to create a new project because we need to know which user we are working with. We have this. This way we don't need to get the current user from Identity and save at least one trip to the database.

Create New Project

ProjectForm.cshtml in Views/Project

The ProjectForm.cshtml is used for both creating a new project and editing an existing project.



@model: Project

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

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. This results in the same thing as simply clicking the Project link in the main menu.

Save

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, one-by-one, 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.

Cancel

The user has clicked the Cancel button in the Create New Project form. This is just an anchor link (<a>) that sends the user back to the list of their projects.

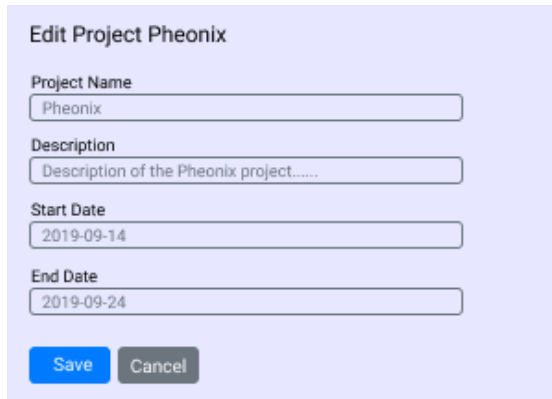
```
<a asp-controller="Project" asp-action="Index" class="btn btn-secondary">Cancel</a>
```

Edit Project

The user has clicked the **Edit** button in the Projects list. We need to know the project Id because there are a series of projects in the list.

ProjectForm.cshtml in Views/Project

The ProjectForm.cshtml is used for both creating a new project and editing an existing project.



@model: Project

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

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

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

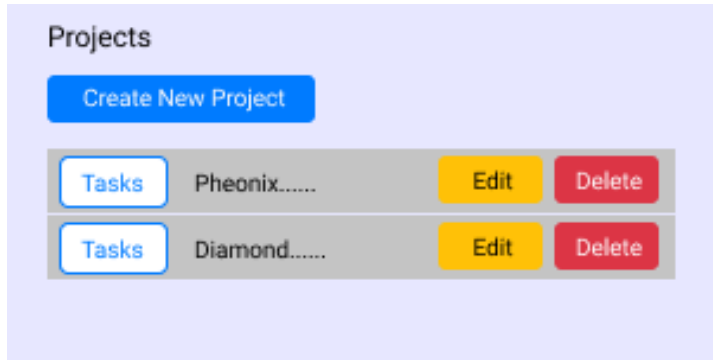
Remove the project

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.



`@model IEnumerable<Project>`

Tasks button

Controller: **ProjectTask**

Action: **TaskListForProject**

Method: post

Type: text

Name: pId

Value: `@p.Id`

hidden

The Tasks button is inside a form that POSTS to the server.

```
@foreach (var p in Model)...
<form asp-controller="ProjectTask" asp-action="TaskListForProject" method="post">
  <input type="text" name="pId" value="@p.Id" hidden />
  <button type="submit" class="btn btn-sm btn-outline-primary">Tasks</button>
</form>
```

The name, pId, must match the parameter pId. If they are named differently, it won't work.

[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

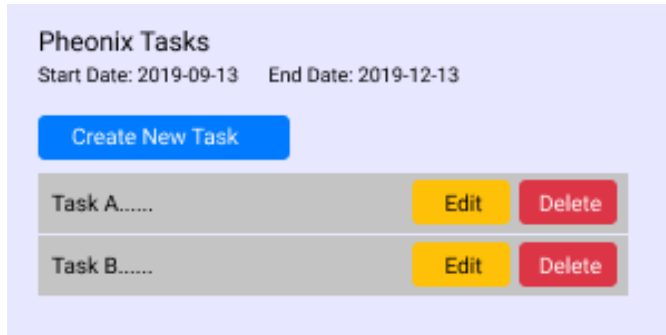
Initialize the view model to include the Project and the list of tasks

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

ViewModel: TasksViewModel

```
public Project project { get; set; }
public IEnumerable<ProjectTask> tasks { get; set; }
public IEnumerable<Priority> priorities { get; set; }
```

Welcome.cshtml in Views/ProjectTasks



@model ProjectTasksPrioritiesViewModel

This view model has three parts. One project, a list of tasks and a list of priorities.

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

<u>Create New Task button</u> Controller: ProjectTask Action: Create Method: post Type: text Name: pId Value: @Model.project.Id hidden	<u>Edit button</u> Controller: ProjectTask Action: Edit Method: post Type: text Name: Id Value: @t.Id hidden
	<u>Delete button</u> 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.

[HttpPost]

public ActionResult Create(int pId)

Get the project from the database using the pId

Initialize the ProjectId property to the pId passed in.

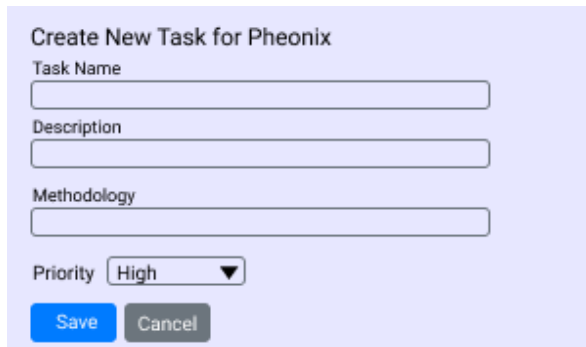
Create a new task object and initialize the ProjectId field

Get the list of Priorities from the database

Return the view ProjectTaskForm along with the view model

Create New Task

ProjectTaskForm.cshtml



@model TaskFormViewModel

```
public Project project { get; set; }
public ProjectTask task { get; set; }
public IEnumerable<Priority> priorities { get; set; }
```

```
<form asp-controller="ProjectTask" asp-action="Save" method="post">
Etc.
<select asp-for="task.PriorityId" asp-items="@((new SelectList(Model.priorities,
"Number", "Name")))">
  <option>Please select a priority</option>
</select>
Etc.
<button type="submit" class="btn btn-primary">Save</button>
```

When the user clicks Save, we are calling the Save action and passing in the TaskFormViewModel. However, when we inspect the code in the Save action, we notice that the only data coming in is the task, not the project and not the list of possible priorities. Why? The only data coming into the Save action is the data in the **form**. If for example, we wanted to include the project's name, we could include in this form (ProjectTaskForm.cshtml) code such as this:

```
<div class="form-group">
  <label asp-for="project.Name" hidden></label>
  <input asp-for="project.Name" class="form-control" hidden />
</div>
```

Since we don't need the project's name, we don't need to include the code above.

Save

The user has filled out the form correctly and clicked the Save button. The Save button is in the <form>.

```
[HttpPost]
public IActionResult Save(TaskFormViewModel tk)
```

If the model state is not valid re-build the view model and return **ProjectTaskForm**

If the task Id is zero, we know we need to add a new task

Create and initialize a **new** task object

Add it to the context

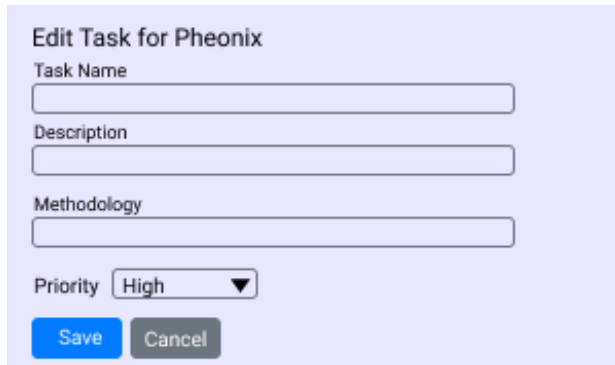
Save the changes to the database

Add the project Id to TempData

Redirect to action Welcome

Edit Task

ProjectTaskForm.cshtml



@model TaskFormViewModel

```
public Project project { get; set; }
public ProjectTask task { get; set; }
public IEnumerable<Priority> priorities { get; set; }
```

Save

The user has filled out the form correctly and clicked the Save button. The Save button is in the <form>.

```
[HttpPost]
public IActionResult Save(TaskFormViewModel tk)
```

If the model state is not valid re-build the view model and return **ProjectTaskForm**

If the task Id is **not zero**, we know we need to **edit** the task

Save the changes to the database

Add the project Id to TempData

Redirect to action Welcome

Delete Task

Pheonix Tasks
Start Date: 2019-09-13 End Date: 2019-12-13

Create New Task

Task A.....	Edit	Delete
Task B.....	Edit	Delete

The user has clicked one of the Delete buttons beside a task.

```
<form asp-controller="ProjectTask" asp-action="Delete" method="post">
  <input type="text" name="Id" value="@t.Id" hidden />
  <button type="submit" class="btn btn-sm btn-danger">Delete</button>
</form>
```

UserInfoController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Identity;
using System.Linq;
using Sources5.Data;
using Sources5.Models;
using Microsoft.AspNetCore.Authorization;
namespace Sources5.Controllers
{
    [Authorize(Roles = "Admin,Users")]
    public class UserInfoController : Controller
    {
        private UserManager<ApplicationUser> userManager;
        private ApplicationDbContext _context;

        public UserInfoController(ApplicationDbContext context, UserManager<ApplicationUser> usrMgr)
        {
            _context = context;
            userManager = usrMgr;
        }

        [HttpGet]
        public ActionResult Index()
        {
            System.Security.Claims.ClaimsPrincipal currentUser = this.User;
            bool isAdmin = currentUser.IsInRole("Admin"); // might use this in future
            var uid = userManager.GetUserId(User); // Get user id:
            UserInfo usrinfo = _context.UserInfos.SingleOrDefault(p => p.ApplicationUserId == uid);
            if (usrinfo != null) {
                //
                var countries = _context.Countries.ToList();
                var vm = new UserInfoFormViewModel
                {
                    userinfo = usrinfo,
                    countries = countries
                };
                return View("Welcome", vm);
            }
            else {
                // the user needs to Create a UserInfo profile
                var ui = new UserInfo {
                    //ApplicationUserId = uid
                };
                var countries = _context.Countries.ToList();
                var vm = new UserInfoFormViewModel
                {
                    userinfo = ui,
                    countries = countries
                };
                return View("UserInfoForm", vm);
            }
        }

        [HttpPost]
        public IActionResult Save(UserInfoFormViewModel uin) // uin is short for user information
        {
            if (!ModelState.IsValid)
            {
                var countries = _context.Countries.ToList();
                var vm = new UserInfoFormViewModel
                {
                    userinfo = uin.userinfo,
                    countries = countries
                };
                return View("UserInfoForm", vm);
            }
            if (uin.userinfo.Id == 0) // ADD NEW USER INFO (UserInfos table in Db)
            {
                System.Security.Claims.ClaimsPrincipal currentUser = this.User;
                var uid = userManager.GetUserId(User); // Get user id:

                uin.userinfo.ApplicationUserId = uid;
                _context.UserInfos.Add(uin.userinfo);
            }
            else // EDIT USER INFO
            {
                var uinfoInDb = _context.UserInfos.Single(p => p.Id == uin.userinfo.Id);
                // Mosh video 44 - manually set or use a DTO for higher level of security,
                // but since we've manually set these, DTO probably doesn't increase security.

                uinfoInDb.FirstName = uin.userinfo.FirstName;
                uinfoInDb.LastName = uin.userinfo.LastName;
                uinfoInDb.SIN = uin.userinfo.SIN;
                uinfoInDb.CountryId = uin.userinfo.CountryId;
                // do not edit the Id or the ApplicationUserId because they stay the same
            }
            _context.SaveChanges();
            return RedirectToAction("Index");
        }

        public ActionResult Edit()
        {
            System.Security.Claims.ClaimsPrincipal currentUser = this.User;
            var uid = userManager.GetUserId(User); // Get application user id:
            // get the User Info (Profile) from the database's UserInfos table
            var uinfoInDb = _context.UserInfos.Single(i => i.ApplicationUserId == uid);

            if (uinfoInDb == null)
            {
                return View("Error"); // change this, but leave it for now.
            }
            var ui = new UserInfo
            {
                Id = uinfoInDb.Id,
                FirstName = uinfoInDb.FirstName,
                LastName = uinfoInDb.LastName,
                SIN = uinfoInDb.SIN,
                CountryId = uinfoInDb.CountryId
                // No need to pass ApplicationUserId out to user because in the Save() action
                // we lookup the project anyway by project Id in Profiles table and then we
                // have the ApplicationUserId. It won't change. User cannot edit that. Security.
            };
            var countries = _context.Countries.ToList();
            var vm = new UserInfoFormViewModel
            {
                userinfo = ui,
                countries = countries
            };
            return View("UserInfoForm", vm);
        }

        [HttpGet]
        public ActionResult Create()
        {
            System.Security.Claims.ClaimsPrincipal currentUser = this.User;
            var uid = userManager.GetUserId(User); // Get application user id:
            var ui = new UserInfo
            {
                ApplicationUserId = uid
            };
            var countries = _context.Countries.ToList();
            var vm = new UserInfoFormViewModel
            {
                userinfo = ui,
                countries = countries
            };
            return View("UserInfoForm", vm); // show the form
        }

        [HttpPost]
        public IActionResult Delete(int Id)
        {
            // if Id is not null or zero...
            if (Id != 0)
            {
                var uInfo = _context.UserInfos.Find(Id);
                _context.UserInfos.Remove(uInfo); // value cannot be null
                _context.SaveChanges();
            }
            return RedirectToAction("Index");
        }
    }
}
```

ProjectController

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Sources5.Data;
using Sources5.Models;
using System;
using System.Collections.Generic;
using System.Linq;
namespace Sources5.Controllers
{
    [Authorize(Roles = "Admin,Users")]
    public class ProjectController : Controller
    {
        private UserManager<ApplicationUser> userManager;
        private ApplicationDbContext _context;

        public ProjectController(ApplicationDbContext context, UserManager<ApplicationUser> usrMgr)
        {
            _context = context;
            userManager = usrMgr;
        }

        public IActionResult Create()
        {
            // The user has clicked the 'Create New Project' button in the Welcome view
            // This is just an anchor link <a> in Welcome.cshtml
            System.Security.Claims.ClaimsPrincipal currentUser = this.User;
            var Uid = userManager.GetUserId(User); // Get user id (string)
            // Get the row from our UserInfos table that is the current user's
            var userinfo = _context.UserInfos.SingleOrDefault(i => i.ApplicationUserId == Uid);
            var project = new Project
            {
                StartDate = DateTime.Now,
                EndDate = DateTime.Now,
                UserInfoId = userinfo.Id
            };
            return View("ProjectForm", project); // show the form
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Save(Project project)
        {
            if (!ModelState.IsValid)
            {
                return View("ProjectForm", project);
            }
            if (project.Id == 0) // ADD NEW PROJECT (Projects table in Db)
            {
                // create a new Project object and use object initializer syntax to
                // initialize all of the properties except for:
                // Id - this is an Identity in Db - do not initialize it - the Db will
                // UserInfo - this is a navigation property
                var p = new Project()
                {
                    // Id = proj.Id, AN IDENTITY THAT THE DB WILL INITIALIZE
                    Name = project.Name,
                    Description = project.Description,
                    StartDate = project.StartDate,
                    EndDate = project.EndDate,
                    UserInfoId = project.UserInfoId // foreign key
                };
                _context.Projects.Add(p);
            }
            else // EDIT PROJECT (Projects table in Db)
            {
                var projInDb = _context.Projects.Single(p => p.Id == project.Id);
                // Mosh video 44 - manually set these (or use a DTO for higher level of security)
                projInDb.Name = project.Name;
                projInDb.Description = project.Description;
                projInDb.StartDate = project.StartDate;
                projInDb.EndDate = project.EndDate;
                // do not edit the Id or the ApplicationUserId because they stay the same
            }
            _context.SaveChanges();

            TempData["message"] = $"{project.Name} has been saved"; // A. Freeman p 310 of Pro ASP.NET
            return RedirectToAction("Index");
        }
    }
}
```

```
    }
    public IActionResult Index()
    {
        //
        System.Security.Claims.ClaimsPrincipal currentUser = this.User;
        var Uid = userManager.GetUserId(User); // Get user id (string)
        // get the user from our UserInfos table.
        UserInfo usrinfo = _context.UserInfos.SingleOrDefault(p => p.ApplicationUserId == Uid);
        // if the user has already created their Profile then go to list of their Projects
        // (Welcome) even if they don't have any projects yet.
        if (usrinfo != null)
        {
            // get the list of this user's projects, if any.
            IEnumerable<Project> projects = _context.Projects.Where(p => p.UserInfoId ==
            usrinfo.Id).ToList();
            return View("Welcome", projects); // pass project list to the Welcome view
        }
        else {
            // user must create a profile before they can create projects.
            return RedirectToAction("Create", "UserInfo"); // (action, controller)
        }
    }
    [HttpPost]
    public ActionResult Edit(int Id)
    {
        // get the project row from the database table Projects that has Id of Id
        var pjInDb = _context.Projects.Single(p => p.Id == Id);
        if (pjInDb != null) // project found in database
        {
            var pj = new Project
            {
                // object initialization syntax
                Id = pjInDb.Id, // changed this to Id = pjInDb.Id
                Name = pjInDb.Name,
                Description = pjInDb.Description,
                StartDate = pjInDb.StartDate,
                EndDate = pjInDb.EndDate,
                UserInfoId = pjInDb.UserInfoId
            };
            return View("ProjectForm", pj);
        }
        return View("Error"); // using Single() would throw an error making this code redundant
    }
    [HttpPost]
    public IActionResult Delete(int Id)
    {
        // if projectId is not null or zero...
        if (Id != 0)
        {
            var project = _context.Projects.Find(Id);
            _context.Projects.Remove(project); // value cannot be null
            _context.SaveChanges();
        }
        else
        {
            return View("Error");
        }
        return RedirectToAction("Index");
    }
}
```

ProjectTaskController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Identity;
using System.Linq;
using Sources5.Data;
using Sources5.Models;
using Microsoft.AspNetCore.Authorization;
namespace Sources5.Controllers
{
    [Authorize(Roles = "Admin,Users")]
    public class ProjectTaskController : Controller
    {
        private UserManager<ApplicationUser> userManager;
        private ApplicationDbContext _context;
        public ProjectTaskController(ApplicationDbContext context, UserManager<ApplicationUser> usrMgr)
        {
            _context = context;
            userManager = usrMgr;
        }

        [HttpPost]
        public IActionResult TaskListForProject(int pId)
        {
            // The user clicked one of the Tasks buttons in the list of projects.
            // The Id of the project is passed in as pId. How?
            // It is a form with method post.
            var project = _context.Projects.SingleOrDefault(p => p.Id == pId);
            if (project != null) // user has some projects.
            {
                var tasks = _context.ProjectTasks.Where(t => t.ProjectId == pId).ToList();
                var vm = new TasksViewModel()
                {
                    project = project,
                    tasks = tasks,
                    priorities = _context.Priorities.ToList()
                };
                return View("Welcome", vm);
            }
            else { return View("Error"); }
        }

        [HttpGet]
        public IActionResult Index()
        {
            // The user has clicked the Cancel, Delete or Save button.
            // (a) The user has clicked the Cancel button in the Task form.
            // (b) The user has clicked the Delete button to delete a task.
            // (c) The user has either created a new task or edited an existing task
            //     and clicked the Save button. We have already saved the task.
            // We need to display the Welcome page that shows all of the tasks
            // for THIS project. TempData contains the project Id.
            //
            var projectId = (int)TempData["projectId"];

            var project = _context.Projects.SingleOrDefault(p => p.Id == projectId);
            if (project != null) // user has some projects.
            {
                //
                var tasks = _context.ProjectTasks.Where(t => t.ProjectId == projectId).ToList();
                // _context.ProjectTasks.Include(t => t.Priority)
                //     .Where(t => t.ProjectId == projectId).ToList();
                var vm = new TasksViewModel()
                {
                    project = project,
                    tasks = tasks,
                    priorities = _context.Priorities.ToList()
                };
                return View("Welcome", vm);
            }
            else
            {
                return View("Error"); // User has no projects. This should never happen.
            }
        }

        [HttpPost]
        public IActionResult Edit(int Id)
        {
            // The Id passed in is the Id number of the TASK in the ProjectTasks table.
            // the user has clicked on a specific Task with this Id

            Id);

            // get the task (single row) in the database table ProjectTasks for this project
            //var taskInDb = _context.ProjectTasks.Include(t => t.Priority).SingleOrDefault(t => t.Id ==
            Id);

            var taskInDb = _context.ProjectTasks.SingleOrDefault(t => t.Id == Id);
            // notice above we are including priorities.
            // var taskInDbFind = _context.ProjectTasks.Include(t => t.Priority).Find(Id);
            // I don't think we can use Find() as Find gives error.
            var projectInDb = _context.Projects.Find(taskInDb.ProjectId);
            if (taskInDb != null)
            {
                var task = new ProjectTask
                {
                    Id = taskInDb.Id,
                    Name = taskInDb.Name,
                    Description = taskInDb.Description,
                    Methodology = taskInDb.Methodology,
                    PriorityId = taskInDb.PriorityId,
                    ProjectId = taskInDb.ProjectId
                };
                var vm = new TaskFormViewModel()
                {
                    project = projectInDb,
                    task = task,
                    priorities = _context.Priorities.ToList()
                };
                return View("ProjectTaskForm", vm);
            }
            else
            {
                return View("Error"); // this is really an error!
            }
        }

        [HttpPost]
        public ActionResult Create(int pId)
        {
            // The user has clicked the Create New Task button.
            // We need to display the form.
            // However, we need to know the project.
            // Create a new task for the project with id of pId.
            var projectInDb = _context.Projects.Find(pId);
            var task = new ProjectTask
            {
                ProjectId = pId
            };
            var vm = new TaskFormViewModel()
            {
                project = projectInDb,
                task = task,
                priorities = _context.Priorities.ToList()
            };
            //return View("ProjectTaskForm", t);
            return View("ProjectTaskForm", vm);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Save(TaskFormViewModel tk) //
        {
            if (!ModelState.IsValid)
            {
                // if not valid send data back and display again
                var vm = new TaskFormViewModel
                {
                    // need to re-build and send back vm, do not send back tk!
                    task = tk.task,
                    priorities = _context.Priorities.ToList(),
                    project = _context.Projects.Single(p => p.Id == tk.task.ProjectId)
                };
                return View("ProjectTaskForm", vm);
            }
            // if (pri == 0) {return View("ProjectTaskForm", tk); } // This code never reached probably
            // because it his above !ModelState.IsValid
            if (tk.task.Id == 0) // ADD NEW TASK (ProjectTasks table)
            {
                var task = new ProjectTask()
                {
                    Name = tk.task.Name,
                    Description = tk.task.Description,
                    Methodology = tk.task.Methodology,
                    PriorityId = tk.task.PriorityId,
                    ProjectId = tk.task.ProjectId // is in the form, but hidden
                };
            }
        }
    }
}
```



```

        //Id = projtask.Id    since this is NEW, the Db Identity will take care of this
    };
    _context.ProjectTasks.Add(task);
}
else // EDIT TASK
{
    var taskInDb = _context.ProjectTasks.Single(t => t.Id == tk.task.Id);
    // Mosh video 44 - can manually set or use a DTO for higher level of security,
    // but since we've manually set these, DTO probably doesn't increase security.
    taskInDb.Name = tk.task.Name;
    taskInDb.Description = tk.task.Description;
    taskInDb.Methodology = tk.task.Methodology;
    taskInDb.PriorityId = tk.task.PriorityId;
    //taskInDb.ProjectId = tk.task.ProjectId    we don't need this we already have it in Db
    //taskInDb.Id = tk.task.Id; we don't need to re-set this; cannot edit this
}
_context.SaveChanges();

TempData["message"] = $"{tk.task.Name} has been saved"; //page 312 in book called Pro ASP.NET
}

Core MVC 2
TempData["projectId"] = tk.task.ProjectId; // is in the form, but hidden
return RedirectToAction("Index");
}
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Delete(int Id)
{
    // Delete a task in the ProjectsTasks table that has an Id of the Id passed in.
    // if projectId is not null or zero...
    if (Id != 0)
    {
        var task = _context.ProjectTasks.Find(Id);
        TempData["projectId"] = task.ProjectId; // PROJECT Id not task Id
        _context.ProjectTasks.Remove(task); // value cannot be null
        _context.SaveChanges();
        return RedirectToAction("Index");
    }
    else { return View("Error"); }
}
[HttpPost]
public IActionResult Cancel(int Id)
{
    TempData["projectId"] = Id;
    return RedirectToAction("Index");
}
}
}
}

```