Building Websites Using ASP.NET Core Razor Pages

Introduction

Imagine you're an employee of IT department in a University. Your manager has asked you to develop a web application for student, course and enrollment management

Course(CourseID, Title, Credits, CourseID)

Enrollment(EnrollmentID, CourseID, StudentID, Grade)

Student(ID IDENTITY(1,1), LastName, FirstMidName, EnrollmentDate)

The application has to support adding, viewing, modifying, and removing students—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

This lab explores creating an application using Razor Pages, ASP.NET Core, and C#. An **SQL Server Database** will be created to persist the product data that will be used for reading and managing product data by **Entity Framework Core.**

Lab Objectives

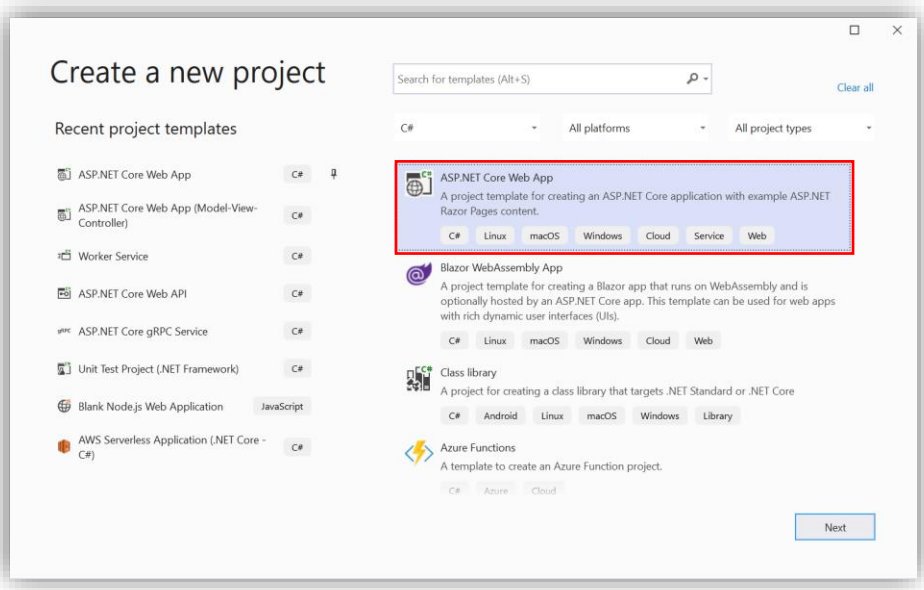
In this lab, you will:

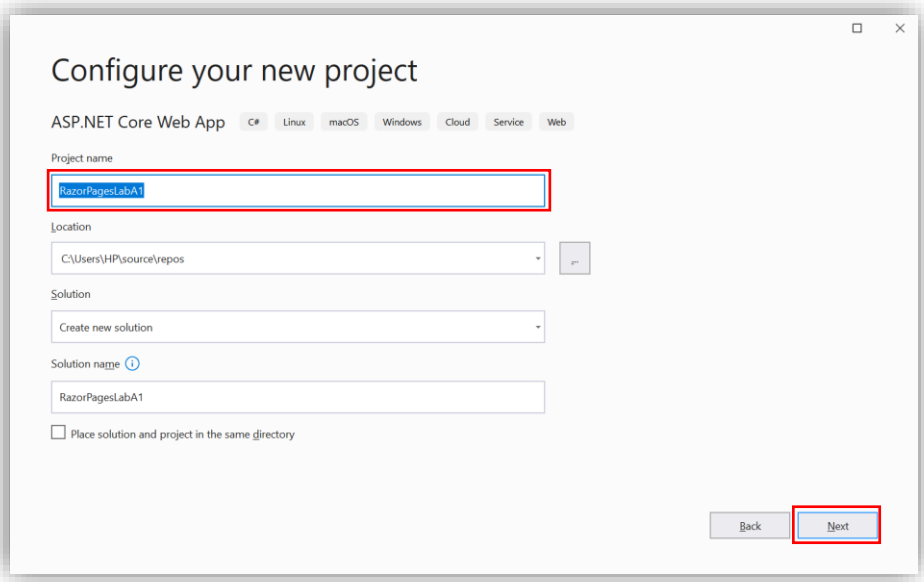
- Use the Visual Studio.NET to create ASP.NET Core Web Application Project.
- Develop application using Razor Pages.
- Apply annotations and custom validation for validating input data.
- Use Entity Framework to Create a SQL Server database named SchoolContextDB that has three tables: Student, Course, Enrollment.

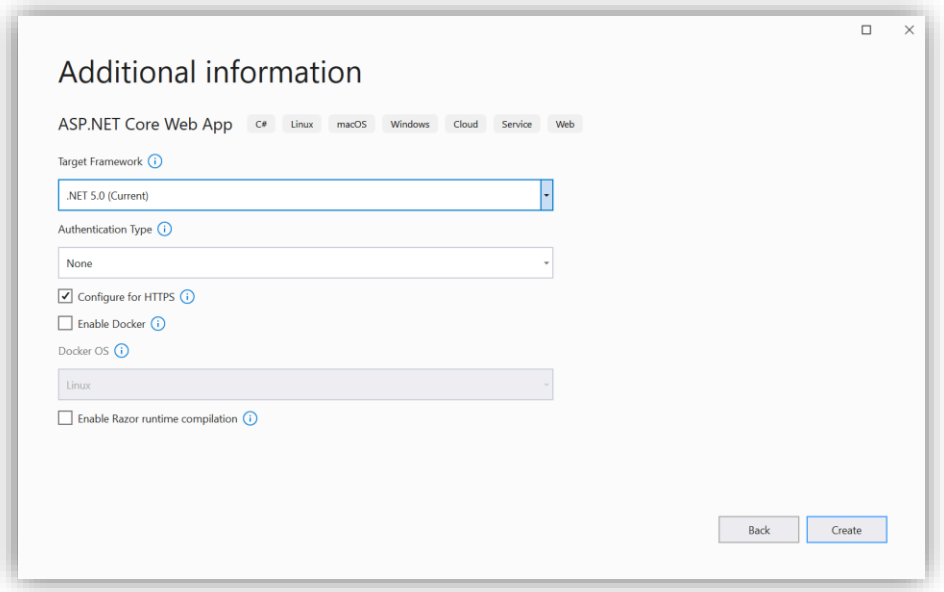
- Develop Entity classes a and DBContext class to perform CRUD actions using Razor Pages.
- Create Search and Paging functions for web application.
- Run the project and test the application actions.

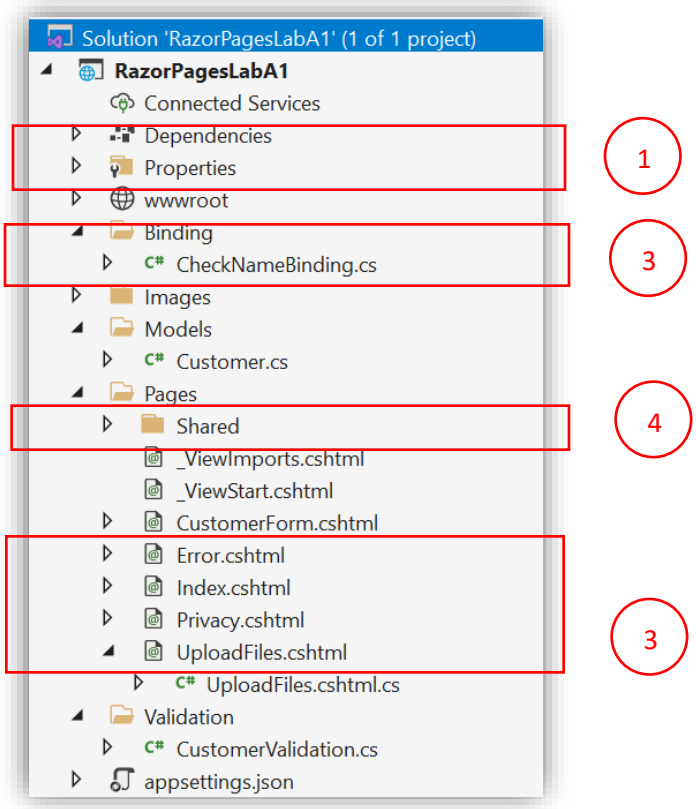
Activity 01: Simple Razor Pages with Validation and File(s) uploading

Step 01. Create ASP.NET Core Web App (A project template for creating an ASP.NET application with example ASP.NET Razor Pages content.)









Step 02. Using Model Binding in Razor Pages to takes values from HTTP requests and maps them to handler method parameters or PageModel properties.

Create class “CheckNameBinding” implements IModelBinder interface.

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Microsoft.Extensions.Logging;
using System;
using System.Threading.Tasks;

namespace RazorPagesLabA1.Binding
{
    4 references
    public class CheckNameBinding : IModelBinder
    {
        private readonly ILogger<CheckNameBinding> _logger;

        0 references
        public CheckNameBinding(ILogger<CheckNameBinding> logger)
        {
            _logger = logger;
        }
        0 references
        public Task BindModelAsync(ModelBindingContext bindingContext)
        {
            if (bindingContext == null)
            {
                throw new ArgumentNullException(nameof(bindingContext));
            }
            // Get ModelName
            string modelName = bindingContext.ModelName;
            ValueProviderResult valueProviderResult = bindingContext.ValueProvider.GetValue(modelName);

            if (valueProviderResult == ValueProviderResult.None)
            {
                return Task.CompletedTask;
            }

            // Set ModelState for the value bindinng
            bindingContext.ModelState.SetModelValue(modelName, valueProviderResult);

            string value = valueProviderResult.FirstValue;
            if (string.IsNullOrEmpty(value))
            {
                return Task.CompletedTask;
            }
            var s = value.ToUpper();

            if (s.Contains("XXX"))
            {
                bindingContext.ModelState.TryAddModelError(
                    modelName, "Cannot contain this pattern xxx.");
                return Task.CompletedTask;

            }
            bindingContext.Result = ModelBindingResult.Success(s.Trim());

            return Task.CompletedTask;

        }
    }
}
```

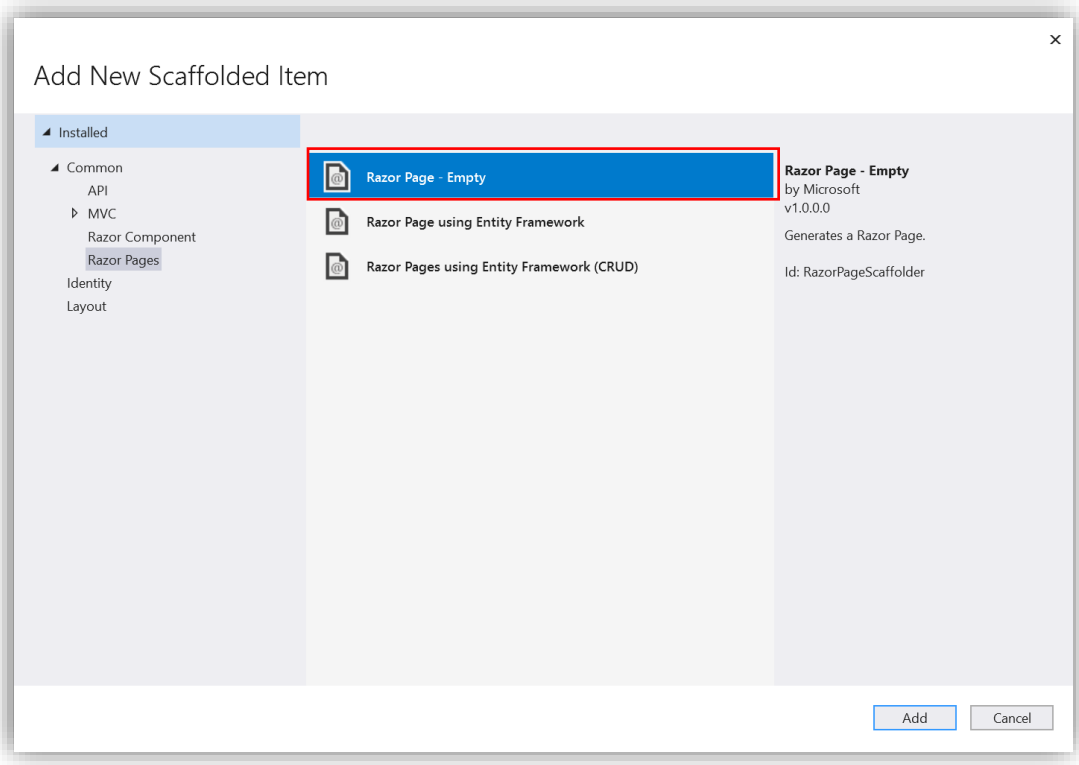
Step 03. Create custom validation class

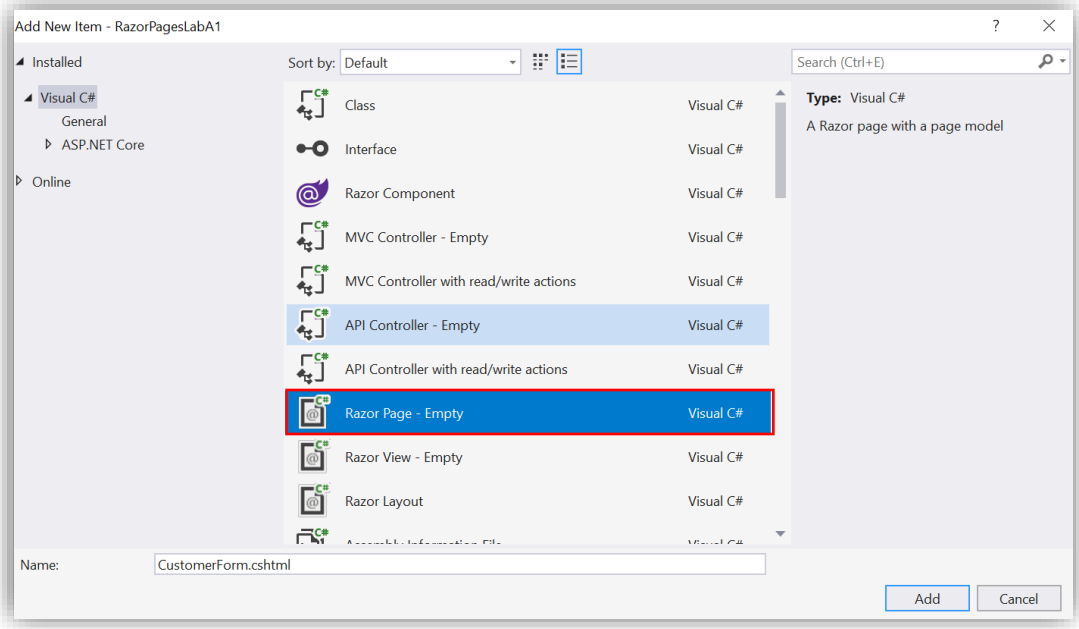
```
using System;
using System.ComponentModel.DataAnnotations;

namespace RazorPagesLabA1.Validation
{
    2 references
    public class CustomerValidation:ValidationAttribute
    {
        1 reference
        public CustomerValidation()
        {
            ErrorMessage = "The year of birth cannot greeter than current year (2021).";
        }
        0 references
        public override bool IsValid(object value)
        {
            if (value == null)
                return false;
            int number = Int32.Parse(value.ToString());
            return (number < 2021);

        }
    }
}
```

Step 04. Create a Model using DataAnnotations and custom validation.

```
using Microsoft.AspNetCore.Mvc;
using RazorPagesLabA1.Validation;
using RazorPagesLabA1.Binding;
using System;
using System.ComponentModel.DataAnnotations;

namespace RazorPagesLabA1.Models
{
    1 reference
    public class Customer
    {
        [Required(ErrorMessage = "Customer name is required!")]
        [StringLength(20, MinimumLength = 3, ErrorMessage = "The length of name is from 3 to 20 chara
        [Display(Name = "Customer name")]
        [ModelBinder(BinderType = typeof(CheckNameBinding))]
        4 references
        public string CustomerName { set; get; }

        [Required(ErrorMessage = "Customer email is required!")]
        [EmailAddress]
        [Display(Name = "Customer email")]
        4 references
        public string Email { set; get; }

        [Required(ErrorMessage = "Year of birth is required!")]
        [Display(Name = "Year of birth")]
        [Range(1960, 2000, ErrorMessage = "1960 - 2000")]
        [CustomerValidation]
        4 references
        public int? YearOfBirth { set; get; }
    }
}
```

Step 05. Create Razor Page (Empty) for form validation





Change information for viewing form (CustomerForm.cshtml)

Message form Model

```
@page
@model RazorPagesLabA1.Pages.CustomerFormModel
@{
    var customerinfo = Model.customerInfo;

    if (customerinfo != null)
    {
        <h3>@Model.Mesage</h3>
        <div asp-validation-summary="All"></div> if (ModelState.IsValid)
        {
            <div class="card">
                <div class="card-body">
                    <p><strong>Customer name: </strong> @customerinfo.CustomerName</p>
                    <p><strong>Customer email: </strong> @customerinfo.Email</p>
                    <p><strong>Year Of Birth: </strong> @customerinfo.YearOfBirth</p>
                </div>
            </div> }

    }
}

 <h3>Customer information</h3>
<form method="post" class="m-2 p-3 bg-light">
    <div class="form-group">
        <label asp-for="@customerinfo.CustomerName"></label>
        <input class="form-control" asp-for="@customerinfo.CustomerName" />
        <span asp-validation-for="@customerinfo.CustomerName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="@customerinfo.Email"></label>
        <input class="form-control" asp-for="@customerinfo.Email" />
        <span asp-validation-for="@customerinfo.Email" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="@customerinfo.YearOfBirth"></label>
        <input class="form-control" asp-for="@customerinfo.YearOfBirth" />
        <span asp-validation-for="@customerinfo.YearOfBirth" class="text-danger"></span>
    </div>
    <div class="form-group">
        <button class="btn btn-primary" asp-page="Form">Send</button>
    </div>
</form>
```

Change the code in CustomerForm.cshtml.cs

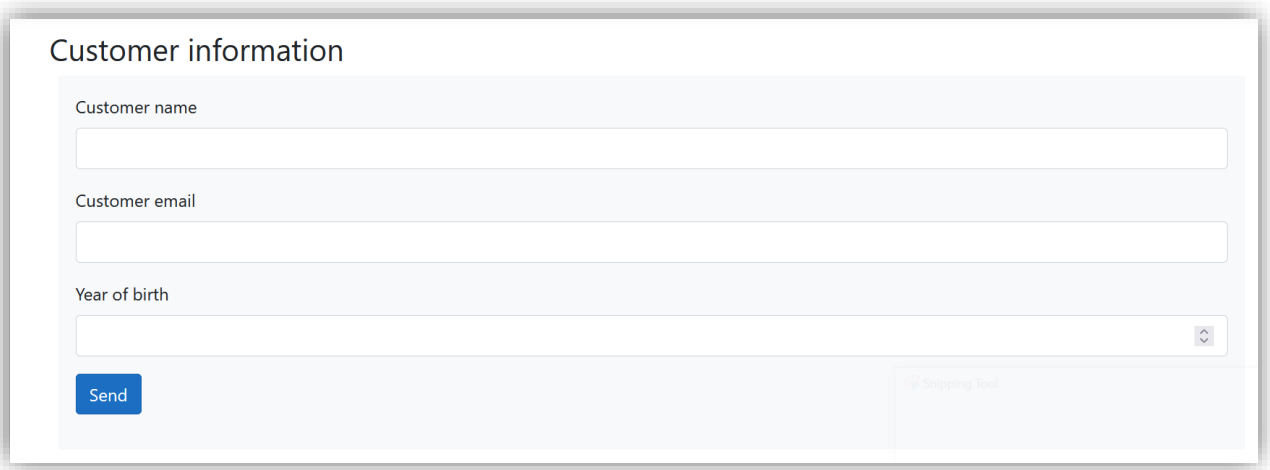
```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using RazorPagesLabA1.Models;

namespace RazorPagesLabA1.Pages
{
    5 references
    public class CustomerFormModel : PageModel
    {
        3 references
        public string Mesage { set; get; }
        [BindProperty]
        1 reference
        public Customer customerInfo { set; get; }

        0 references
        public void OnPost()
        {
            if (ModelState.IsValid)
            {
                Mesage = "Information is OK";
                ModelState.Clear();
            }
            else
            {
                Mesage = "Error on input data.";
            }
        }
    }
}
```

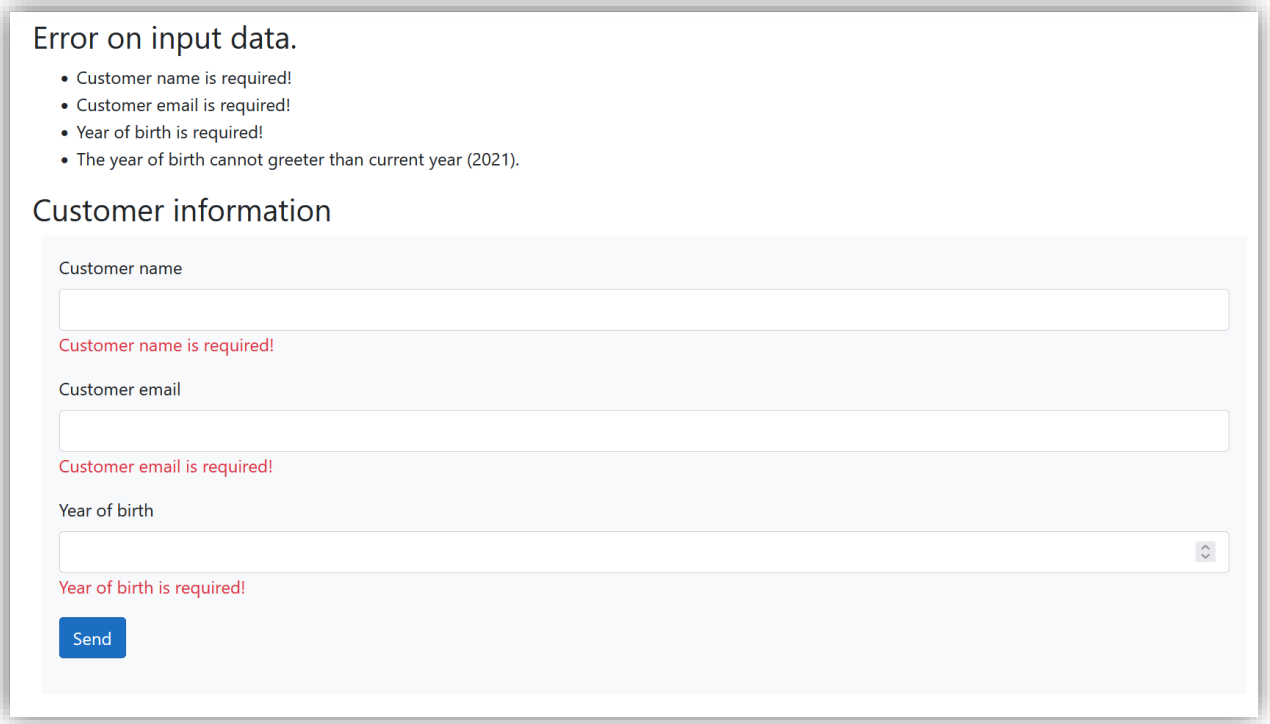
Step 06. Create Razor Page (Empty) for uploading files

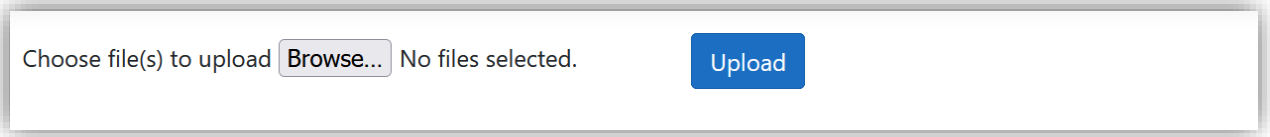
```
@page
@model RazorPagesLabA1.Pages.UploadFilesModel
@{
    <form method="post" enctype="multipart/form-data">
        <label asp-for="@Model.FileUploads"></label>
        <input asp-for="@Model.FileUploads" />
        <span asp-validation-formaction="@Model.FileUploads"></span>

        <button class="btn btn-primary" asp-page="UploadMulti">Upload</button>
    </form>
}
```

```
1  using System.ComponentModel.DataAnnotations;
2  using System.IO;
3  using System.Threading.Tasks;
4  using Microsoft.AspNetCore.Hosting;
5  using Microsoft.AspNetCore.Http;
6  using Microsoft.AspNetCore.Mvc;
7  using Microsoft.AspNetCore.Mvc.RazorPages;
8
9
10 namespace RazorPagesLabA1.Pages
11 {
       6 references
12     public class UploadFilesModel : PageModel
13     {
14         private IHostingEnvironment _environment;
           0 references
15         public UploadFilesModel(IHostingEnvironment environment)
16         {
17             _environment = environment;
18         }
19
20         [Required(ErrorMessage = "Please choose at least one file.")]
21         [DataType(DataType.Upload)]
22         [FileExtensions(Extensions = "png,jpg,jpeg,gif")]
23         [Display(Name = "Choose file(s) to upload")]
24         [BindProperty]
           5 references
25         public IFormFile[] FileUploads { get; set; }

           0 references
26         public async Task OnPostAsync()
27         {
28             if (FileUploads != null)
29             {
30                 foreach (var FileUpload in FileUploads)
31                 {
32                     var file = Path.Combine(_environment.ContentRootPath, "Images",
33                                                                 FileUpload.FileName);
34                     using (var fileStream = new FileStream(file, FileMode.Create))
35                     {
36                         await FileUpload.CopyToAsync(fileStream);
37                     }
38                 }
39             }
40
41         }
42     }
43 }
```

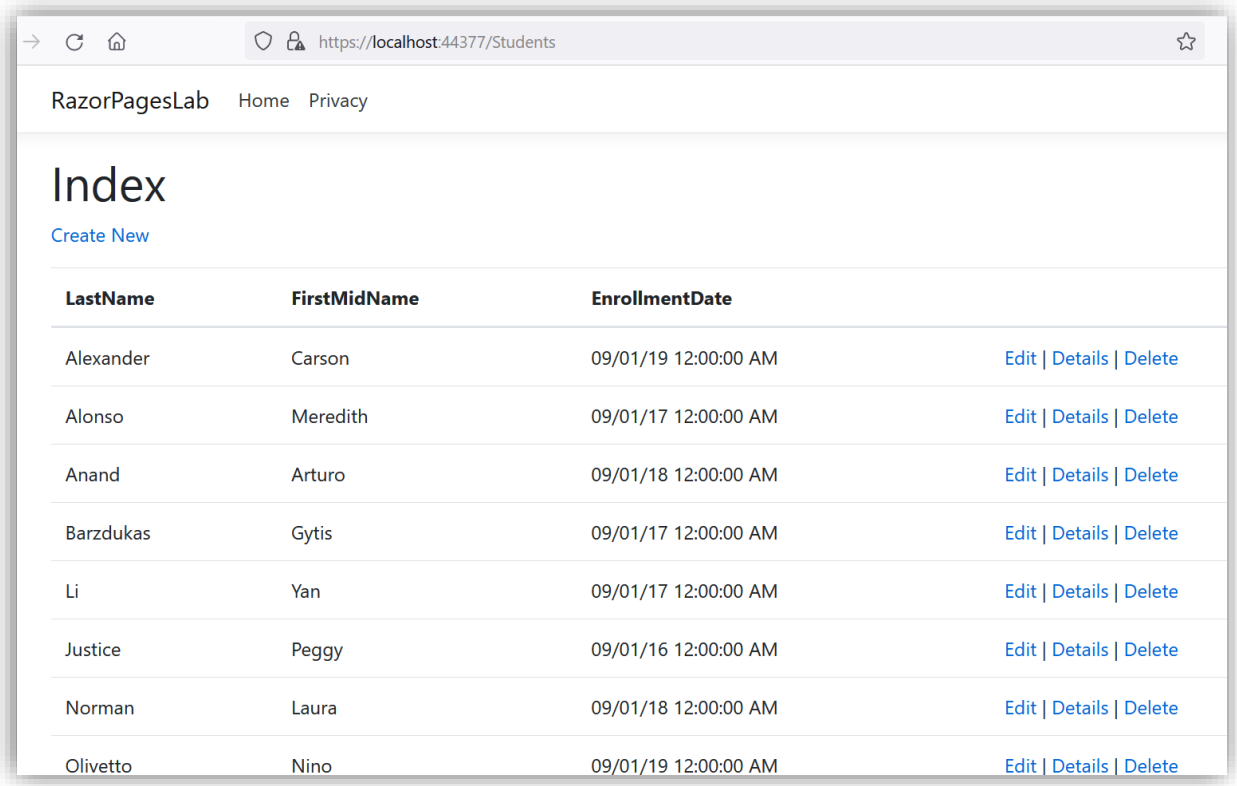
Step 07. Build and run Project.



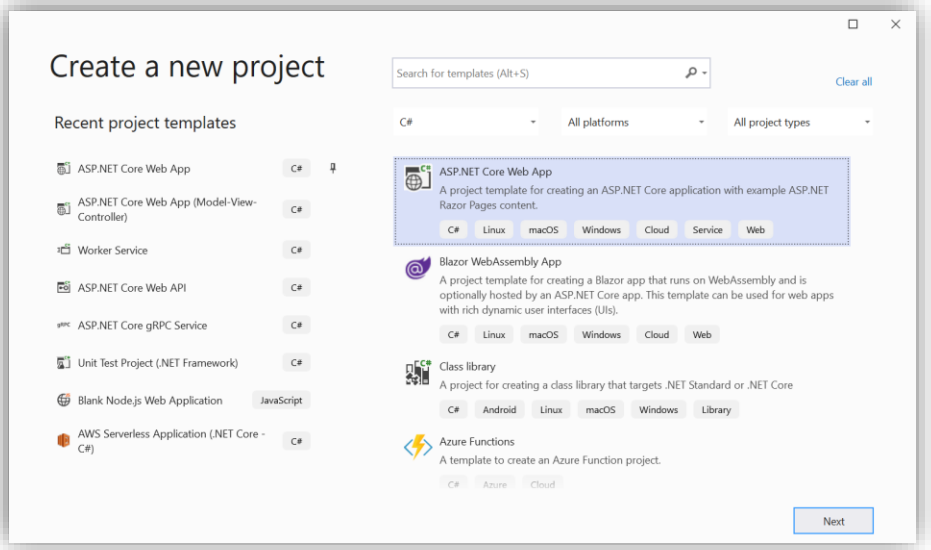


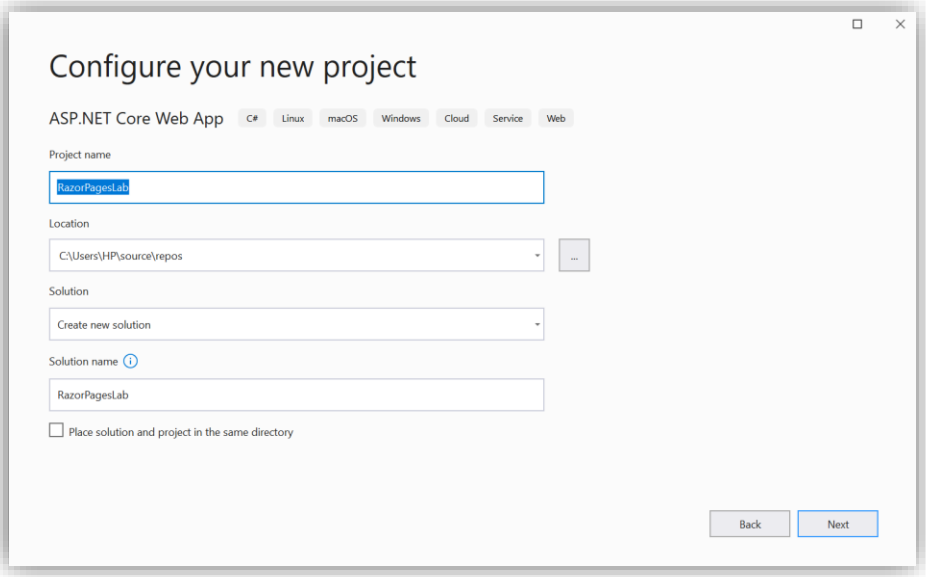


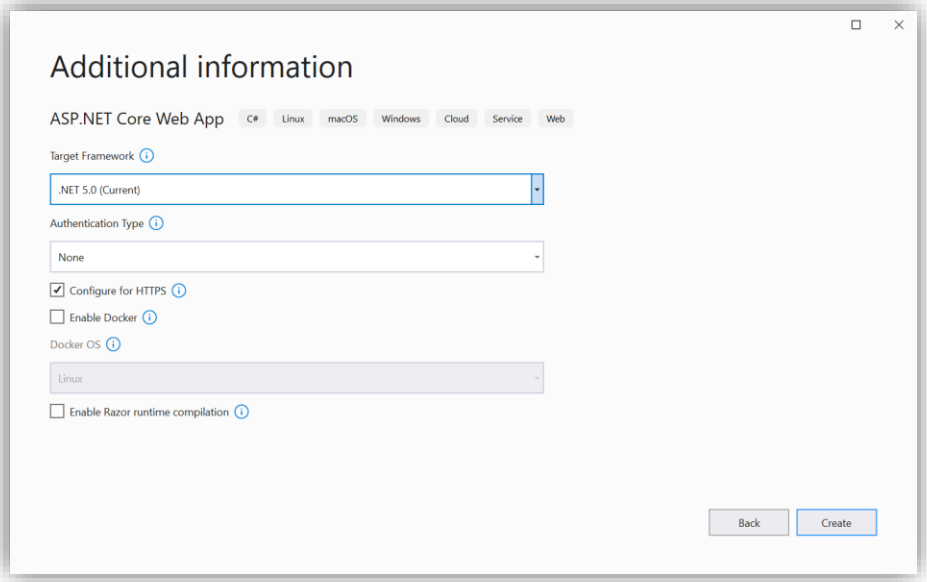
Activity 02: Razor Pages with Entity Framework CRUD

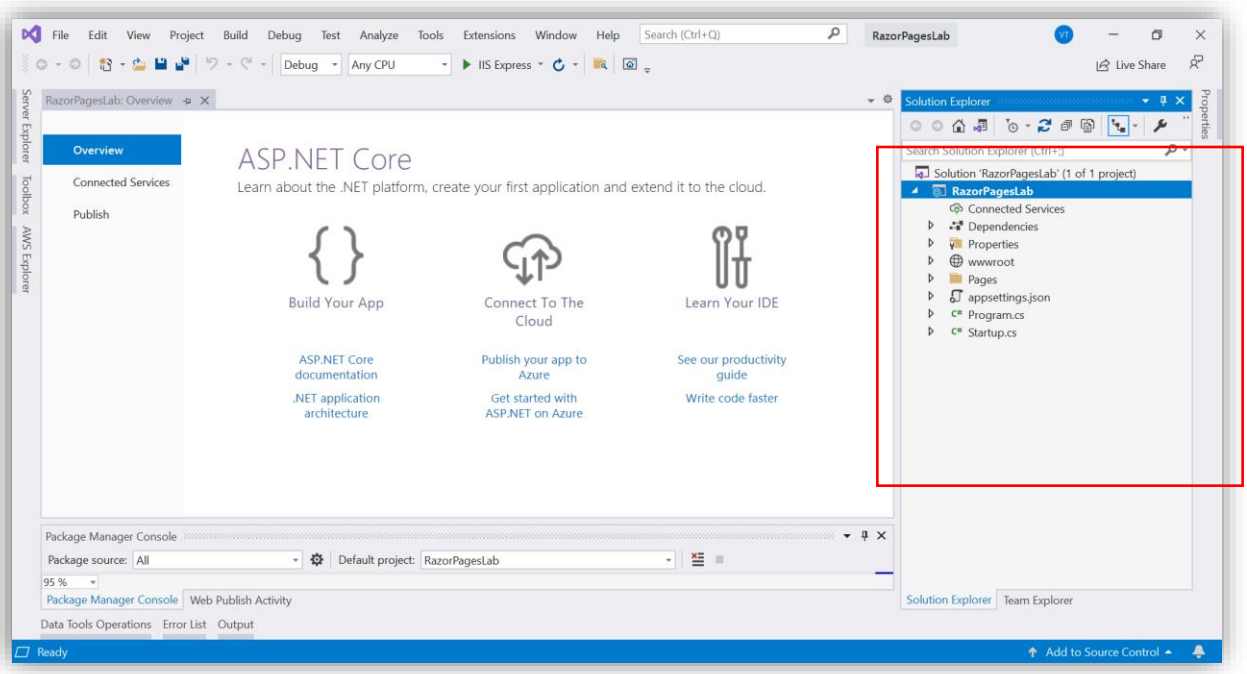


Step 01. Create ASP.NET Core Web Application




Configure your new project
ASP.NET Core Web App
C#
Linux
macOS
Windows
Cloud
Service
Web
Project name
RazorPagesLab
Location
C:\Users\HP\source\repos
...
Solution
Create new solution
Solution name
RazorPagesLab
Place solution and project in the same directory
Back
Next


Additional information
ASP.NET Core Web App
C#
Linux
macOS
Windows
Cloud
Service
Web
Target Framework
.NET 5.0 (Current)
Authentication Type
None
Configure for HTTPS
Enable Docker
Docker OS
Linux
Enable Razor runtime compilation
Back
Create



Step 02. Add model – Student, Course, Enrolment.

A student can enroll in any number of courses, and a course can have any number of students enrolled in it.

```
public class Course
{
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    7 references
    public int CourseID { get; set; }
    7 references
    public string Title { get; set; }
    7 references
    public int Credits { get; set; }

    0 references
    public ICollection<Enrollment> Enrollments { get; set; }
}

public class Student
{
    11 references
    public int ID { get; set; }
    21 references
    public string LastName { get; set; }
    21 references
    public string FirstMidName { get; set; }
    21 references
    public DateTime EnrollmentDate { get; set; }

    0 references
    public ICollection<Enrollment> Enrollments { get; set; }
}
```

```
public enum Grade
{
    A, B, C, D, F
}

17 references
public class Enrollment
{
    0 references
    public int EnrollmentID { get; set; }
    12 references
    public int CourseID { get; set; }
    12 references
    public int StudentID { get; set; }
    [DisplayFormat(NullDisplayText = "No grade")]
    9 references
    public Grade? Grade { get; set; }

    0 references
    public Course Course { get; set; }
    0 references
    public Student Student { get; set; }
}
```

The database context class SchoolContext.cs.

```
15 references
public class SchoolContext : DbContext
{
    0 references
    public SchoolContext(DbContextOptions<SchoolContext> options)
        : base(options)
    {
    }

    10 references
    public DbSet<Student> Students { get; set; }
    1 reference
    public DbSet<Enrollment> Enrollments { get; set; }
    1 reference
    public DbSet<Course> Courses { get; set; }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Course>().ToTable("Course");
        modelBuilder.Entity<Enrollment>().ToTable("Enrollment");
        modelBuilder.Entity<Student>().ToTable("Student");
    }
}
```

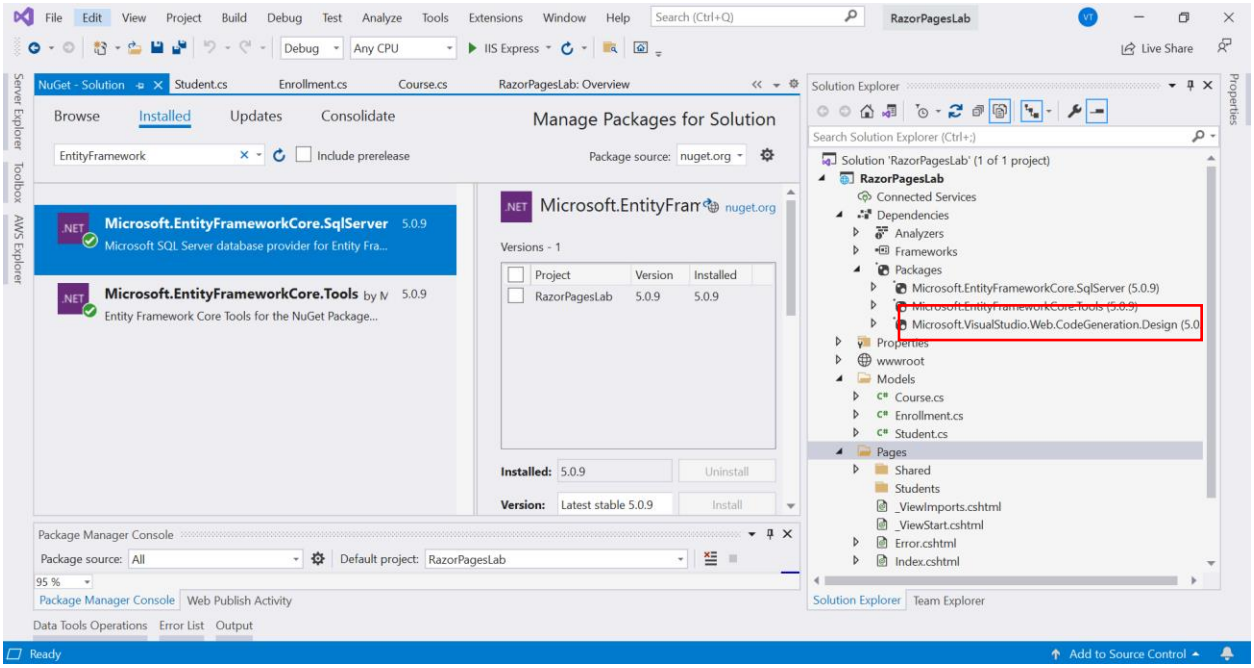
DbInitializer.cs

```
using System;
using System.Linq;
using RazorPagesLab.Models;
```

```
namespace RazorPagesLab.Data
{
    public class DbInitializer
    {
        public static void Initialize(SchoolContext context)
        {
            // Look for any students.
            if (context.Students.Any())
            {
                return;   // DB has been seeded
            }

            var students = new Student[]
            {
                new
Student{FirstMidName="Carson",LastName="Alexander",EnrollmentDate=DateTime.Parse("201
9-09-01")},
                new
Student{FirstMidName="Meredith",LastName="Alonso",EnrollmentDate=DateTime.Parse("2017
-09-01")},
                new
Student{FirstMidName="Arturo",LastName="Anand",EnrollmentDate=DateTime.Parse("2018-
09-01")},
                new
Student{FirstMidName="Gytis",LastName="Barzdukas",EnrollmentDate=DateTime.Parse("2017
-09-01")},
                new
Student{FirstMidName="Yan",LastName="Li",EnrollmentDate=DateTime.Parse("2017-09-
01")},
                new
Student{FirstMidName="Peggy",LastName="Justice",EnrollmentDate=DateTime.Parse("2016-
09-01")},
                new
Student{FirstMidName="Laura",LastName="Norman",EnrollmentDate=DateTime.Parse("2018-
09-01")},
                new
Student{FirstMidName="Nino",LastName="Olivetto",EnrollmentDate=DateTime.Parse("2019-
09-01")}
            };

            context.Students.AddRange(students);
            context.SaveChanges();

            var courses = new Course[]
            {
                new Course{CourseID=1050,Title="Chemistry",Credits=3},
                new Course{CourseID=4022,Title="Microeconomics",Credits=3},
                new Course{CourseID=4041,Title="Macroeconomics",Credits=3},
                new Course{CourseID=1045,Title="Calculus",Credits=4},
                new Course{CourseID=3141,Title="Trigonometry",Credits=4},
                new Course{CourseID=2021,Title="Composition",Credits=3},
                new Course{CourseID=2042,Title="Literature",Credits=4}
            };
```

```
            context.Courses.AddRange(courses);
            context.SaveChanges();

            var enrollments = new Enrollment[]
            {
                new Enrollment{StudentID=1,CourseID=1050,Grade=Grade.A},
                new Enrollment{StudentID=1,CourseID=4022,Grade=Grade.C},
                new Enrollment{StudentID=1,CourseID=4041,Grade=Grade.B},
                new Enrollment{StudentID=2,CourseID=1045,Grade=Grade.B},
                new Enrollment{StudentID=2,CourseID=3141,Grade=Grade.F},
                new Enrollment{StudentID=2,CourseID=2021,Grade=Grade.F},
                new Enrollment{StudentID=3,CourseID=1050},
                new Enrollment{StudentID=4,CourseID=1050},
                new Enrollment{StudentID=4,CourseID=4022,Grade=Grade.F},
                new Enrollment{StudentID=5,CourseID=4041,Grade=Grade.C},
                new Enrollment{StudentID=6,CourseID=1045},
                new Enrollment{StudentID=7,CourseID=3141,Grade=Grade.A},
            };

            context.Enrollments.AddRange(enrollments);
            context.SaveChanges();
        }
    }
}
```

Step 03. Manage NuGet packages for Solution/Project



Step 04. Add Connection string (appsettings.json file)

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
```

```
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Persist Security Info=False;User ID=sa;Password=1234567890;Initial
Catalog=SchoolContextDB;Data Source=.;Connection Timeout=100000"
  }
}
```

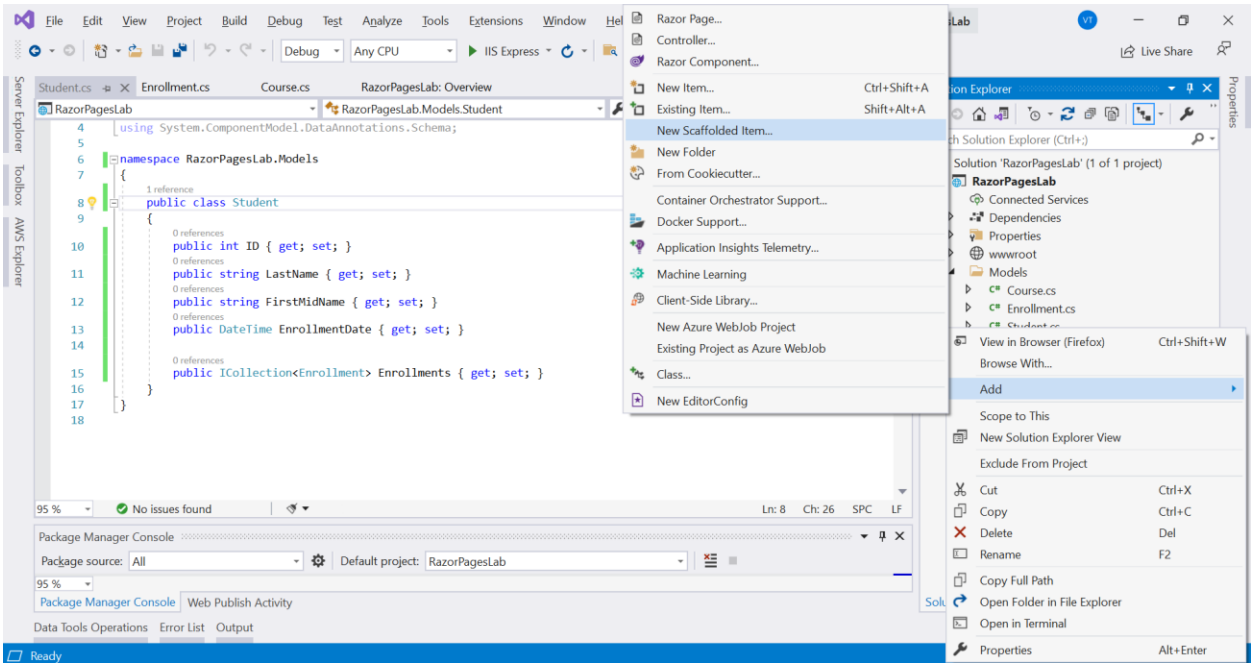
Step 05. Scaffold Student pages

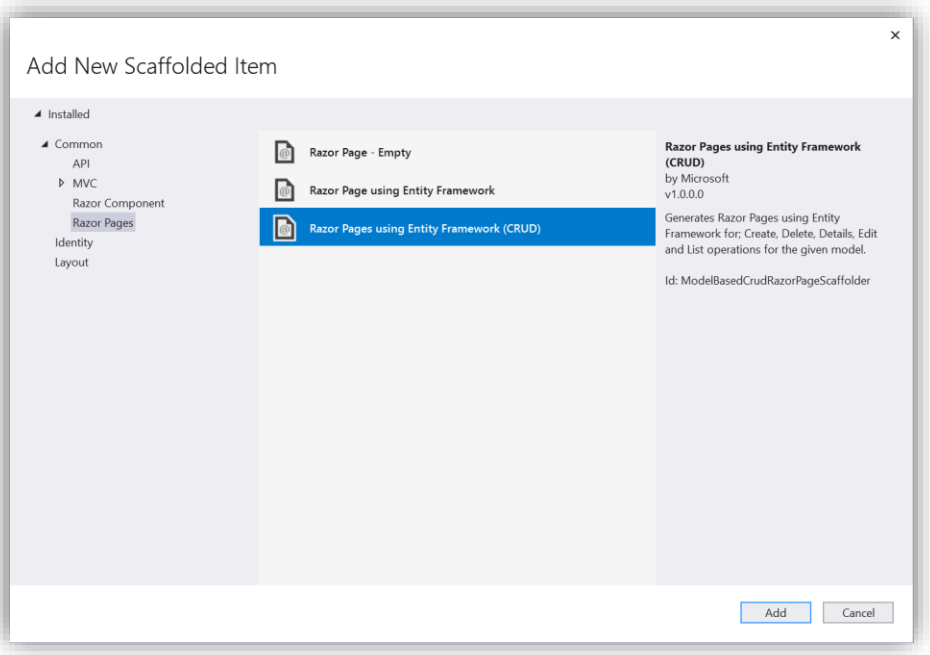
The following packages are automatically installed:

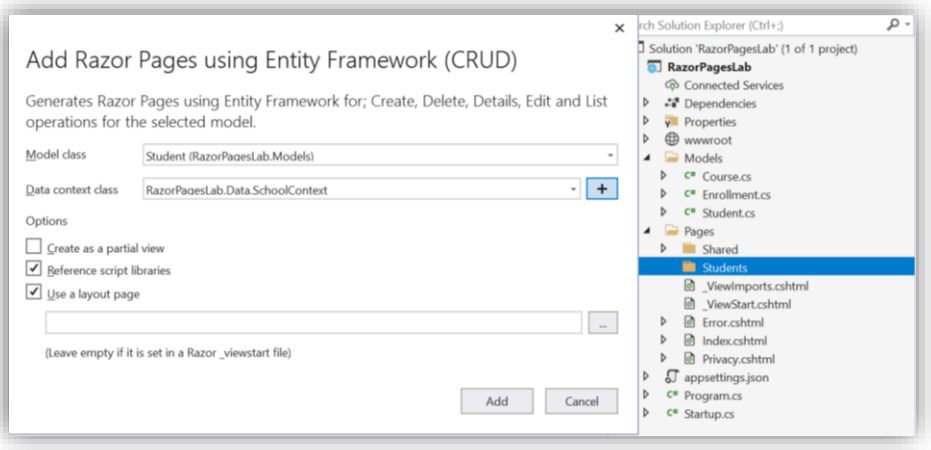
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- **Microsoft.VisualStudio.Web.CodeGeneration.Design**

The scaffolding process will provide these files (creates Razor pages in the *Pages/Students* folder)

- Create.cshtml and Create.cshtml.cs
- Delete.cshtml and Delete.cshtml.cs
- Details.cshtml and Details.cshtml.cs
- Edit.cshtml and Edit.cshtml.cs
- Index.cshtml and Index.cshtml.cs







Step 06. Change the code on Startup.cs and Program.cs

Add the context to dependency injection in *Startup.cs.*

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
```

```
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the
container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddRazorPages();

        services.AddDbContext<SchoolContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("SchoolContext")));


    }

    // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();

            app.UseDeveloperExceptionPage();
            app.UseMigrationsEndPoint();
        }
        else
        {
            app.UseExceptionHandler("/Error");
            // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapRazorPages();
        });
    }
}
```

Register the SchoolContext

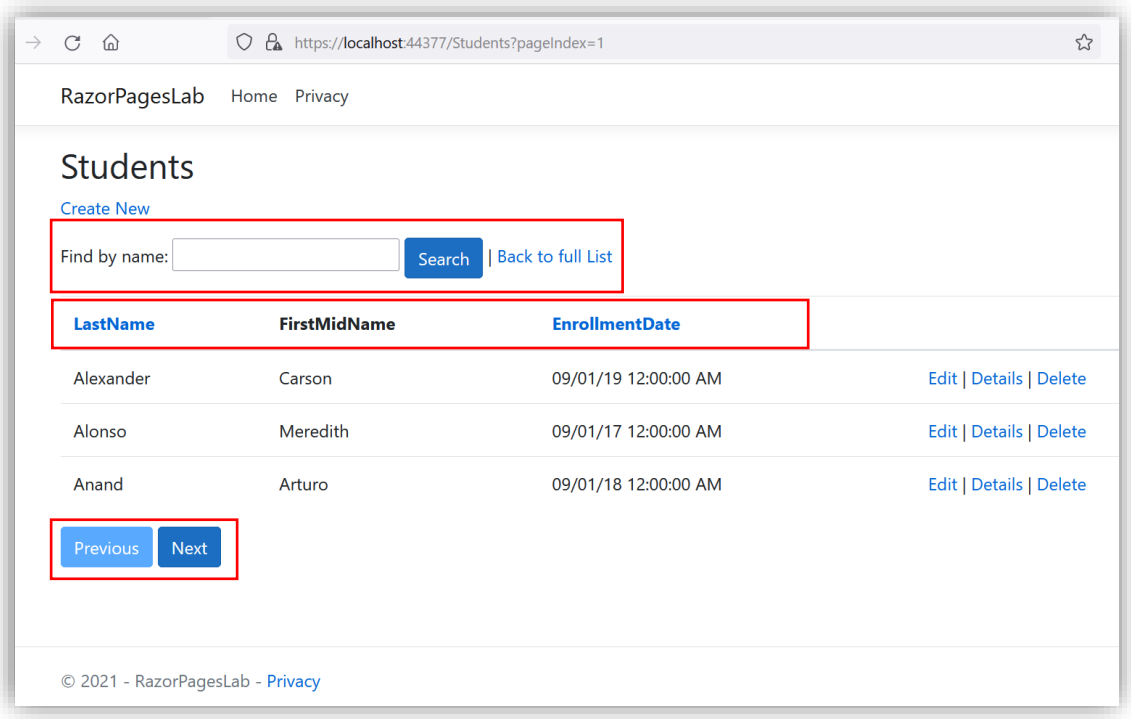
Check if the database does not exist in the *Program.cs*

```
public class Program
```

```
{
    public static void Main(string[] args)
    {
        var host = CreateHostBuilder(args).Build();
        CreateDbIfNotExists(host);
        host.Run();
    }

    private static void CreateDbIfNotExists(IHost host)
    {
        using (var scope = host.Services.CreateScope())
        {
            var services = scope.ServiceProvider;
            try
            {
                var context = services.GetRequiredService<SchoolContext>();
                context.Database.EnsureCreated();
                // DbInitializer.Initialize(context);
            }
            catch (Exception ex)
            {
                var logger = services.GetRequiredService<ILogger<Program>>();
                logger.LogError(ex, "An error occurred creating the DB.");
            }
        }
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

Step 07. Build and run Program.

Activity 03: Razor Pages with Entity Framework - Sort, Filter, Paging



Step 00. Using Project in Activity 2.

Step 01. Working with Index.cshtml.cs

The method OnGetAsync using the parameters

- the *sortOrder* as parameter from the query string in the URL
- the *currentFilter* as parameter for filtering and saves the parameter value in the CurrentFilter property.
- the *searchString* as parameter for search string
- the *pageIndex* as parameter for page index

Step 02. Add pagging. Create PaginatedList class to support paging. The PaginatedList class uses Skip and Take statements to filter data on the server instead of retrieving all rows of the table.

```
public class PaginatedList<T> : List<T>
```

```
{
    public int PageIndex { get; private set; }
    public int TotalPages { get; private set; }

    public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
    {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);

        this.AddRange(items);
    }
    public bool HasPreviousPage
    {
        get
        {
            return (PageIndex > 1);
        }
    }
    public bool HasNextPage
    {
        get
        {
            return (PageIndex < TotalPages);
        }
    }
    public static async Task<PaginatedList<T>> CreateAsync(
        IQueryable<T> source, int pageIndex, int pageSize)
    {
        var count = await source.CountAsync();
        var items = await source.Skip(
            (pageIndex - 1) * pageSize)
            .Take(pageSize).ToListAsync();
        return new PaginatedList<T>(items, count, pageIndex, pageSize);
    }
}
```

Add page size to configuration

```
{
  "PageSize": 3,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "SchoolContext":
"Server=.;Database=SchoolContextLab;Trusted_Connection=True;MultipleActiveResultSets=true
"
  }
}
```

```
12  public class IndexModel : PageModel
13  {
14      private readonly SchoolContext _context;
15      private readonly IConfiguration Configuration;
16
        0 references
17      public IndexModel(SchoolContext context, IConfiguration configuration)
18      {
19          _context = context;
20          Configuration = configuration;
21      }
22
        2 references
23      public string NameSort { get; set; }
        2 references
24      public string DateSort { get; set; }
        6 references
25      public string CurrentFilter { get; set; }
        3 references
26      public string CurrentSort { get; set; }
27
        9 references
28      public PaginatedList<Student> Students { get; set; }
29
        0 references
30      public async Task OnGetAsync(string sortOrder,
31          string currentFilter, string searchString, int? pageIndex)
32      {
33          CurrentSort = sortOrder;
34          NameSort = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
35          DateSort = sortOrder == "Date" ? "date_desc" : "Date";
36
37          if (searchString != null)
38          {
39              pageIndex = 1;
40          }
41          else
42          {
43              searchString = currentFilter;
44          }
45
46          CurrentFilter = searchString;
47
48          IQueryable<Student> studentsIQ = from s in _context.Students
49                                           select s;
50          if (!String.IsNullOrEmpty(searchString))
51          {
52              studentsIQ = studentsIQ.Where(s => s.LastName.Contains(searchString)
53                                    || s.FirstMidName.Contains(searchString));
54          }
55          switch (sortOrder)
56          {
57              case "name_desc":
58                  studentsIQ = studentsIQ.OrderByDescending(s => s.LastName);
59                  break;
60              case "Date":
61                  studentsIQ = studentsIQ.OrderBy(s => s.EnrollmentDate);
62                  break;
63              case "date_desc":
64                  studentsIQ = studentsIQ.OrderByDescending(s => s.EnrollmentDate);
65                  break;
66              default:
67                  studentsIQ = studentsIQ.OrderBy(s => s.LastName);
68                  break;
```

The method uses LINQ to Entities to specify the column to sort by.

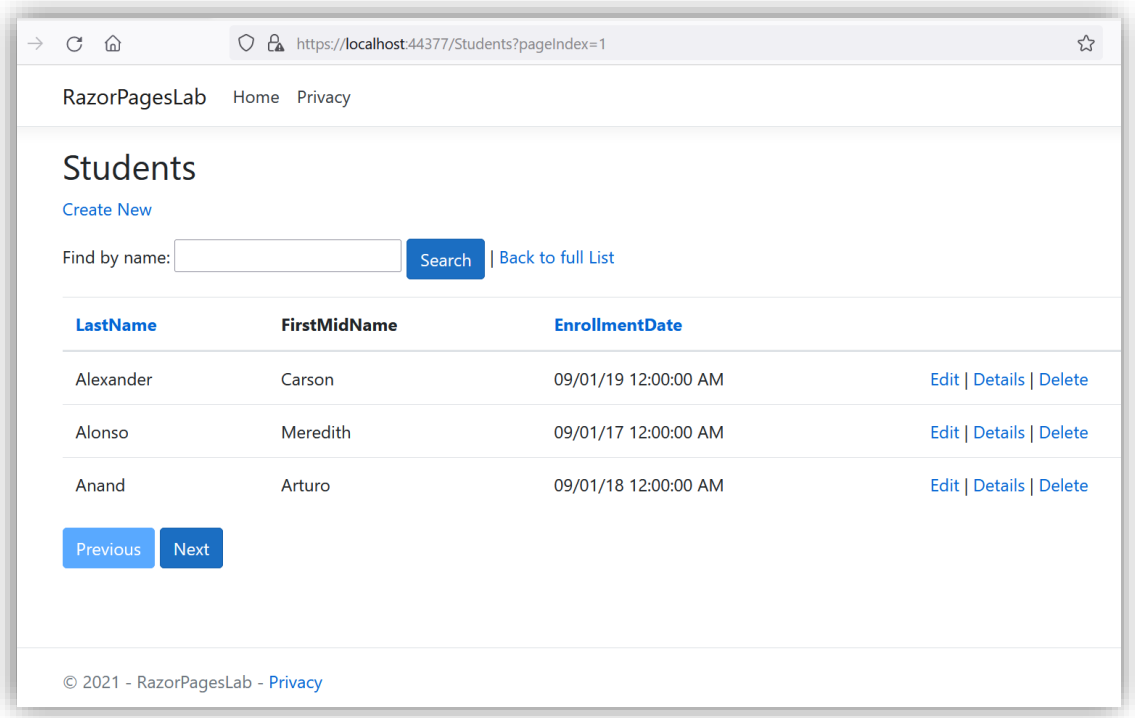
```
68              }
69
70              var pageSize = Configuration.GetValue("PageSize", 4);
71              Students = await PaginatedList<Student>.CreateAsync(
72                  studentsIQ.AsNoTracking(), pageIndex ?? 1, pageSize);
73          }
74      }
75  }
```

Step 03. Work with Student Index page Index.cshtml

```
1   @page
2   @model RazorPagesLab.Pages.Students.IndexModel
3
4   @{
5       ViewData["Title"] = "Students";
6   }
7
8   <h2>Students</h2>
9
10  <p>
11      <a asp-page="Create">Create New</a>
12  </p>
13
14  <form asp-page="./Index" method="get">
15      <div class="form-actions no-color">
16          <p>
17              Find by name:
18              <input type="text" name="SearchString" value="@Model.CurrentFilter" />
19              <input type="submit" value="Search" class="btn btn-primary" /> |
20              <a asp-page="./Index">Back to full List</a>
21          </p>
22      </div>
23  </form>
24
25  <table class="table">
26      <thead>
27          <tr>
28              <th>
29                  <a asp-page="./Index" asp-route-sortOrder="@Model.NameSort"
30                     asp-route-currentFilter="@Model.CurrentFilter">
31                      @Html.DisplayNameFor(model => model.Students[0].LastName)
32                  </a>
33              </th>
34              <th>
35                  @Html.DisplayNameFor(model => model.Students[0].FirstMidName)
36              </th>
37              <th>
38                  <a asp-page="./Index" asp-route-sortOrder="@Model.DateSort"
39                     asp-route-currentFilter="@Model.CurrentFilter">
40                      @Html.DisplayNameFor(model => model.Students[0].EnrollmentDate)
41                  </a>
42              </th>
43              <th></th>
44          </tr>
45      </thead>
```

```
46      <tbody>
47          @foreach (var item in Model.Students)
48          {
49              <tr>
50                  <td>
51                      @Html.DisplayFor(modelItem => item.LastName)
52                  </td>
53                  <td>
54                      @Html.DisplayFor(modelItem => item.FirstMidName)
55                  </td>
56                  <td>
57                      @Html.DisplayFor(modelItem => item.EnrollmentDate)
58                  </td>
59                  <td>
60                      <a asp-page="./Edit" asp-route-id="@item.ID">Edit</a> |
61                      <a asp-page="./Details" asp-route-id="@item.ID">Details</a> |
62                      <a asp-page="./Delete" asp-route-id="@item.ID">Delete</a>
63                  </td>
64              </tr>
65          }
66      </tbody>

67  </table>
68
69  @{
70      var prevDisabled = !Model.Students.HasPreviousPage ? "disabled" : "";
71      var nextDisabled = !Model.Students.HasNextPage ? "disabled" : "";
72  }
73
74  <a asp-page="./Index"
75     asp-route-sortOrder="@Model.CurrentSort"
76     asp-route-pageIndex="@(Model.Students.PageIndex - 1)"
77     asp-route-currentFilter="@Model.CurrentFilter"
78     class="btn btn-primary @prevDisabled">
79      Previous
80  </a>
81  <a asp-page="./Index"
82     asp-route-sortOrder="@Model.CurrentSort"
83     asp-route-pageIndex="@(Model.Students.PageIndex + 1)"
84     asp-route-currentFilter="@Model.CurrentFilter"
85     class="btn btn-primary @nextDisabled">
86      Next
87  </a>
```

Step 04. Run and test all functions of Project.


https://localhost:44377/Students?pageIndex=1
RazorPagesLab
Home
Privacy
Students
Create New
Find by name:
Search
| Back to full List
LastName
FirstMidName
EnrollmentDate
Alexander
Carson
09/01/19 12:00:00 AM
Edit | Details | Delete
Alonso
Meredith
09/01/17 12:00:00 AM
Edit | Details | Delete
Anand
Arturo
09/01/18 12:00:00 AM
Edit | Details | Delete
Previous
Next
© 2021 - RazorPagesLab - Privacy