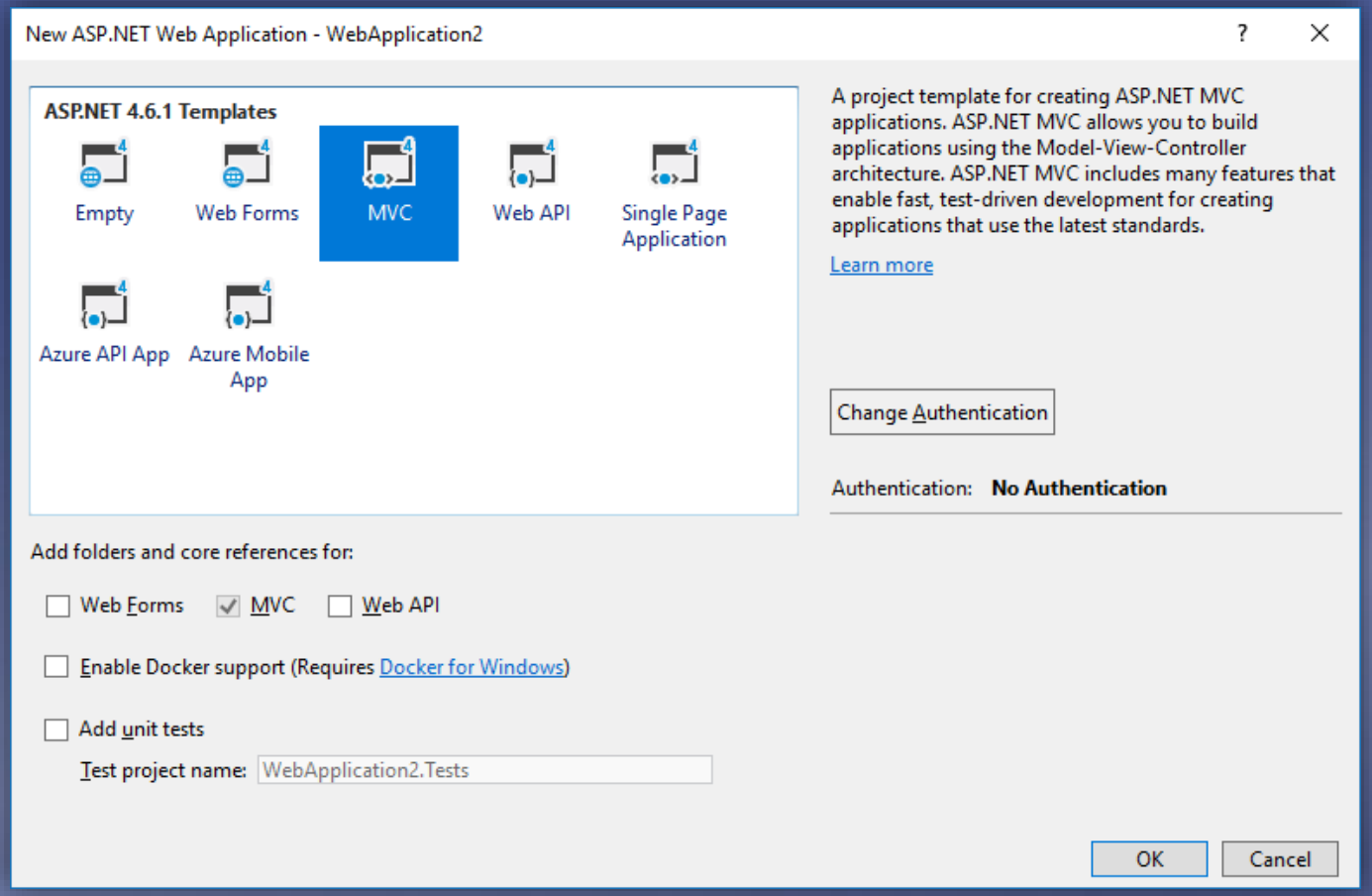
KznDotNet Meetup #1

KZN
DOT
NET

Razor Pages: New page-based framework in ASP.NET Core

Ладохин Станислав
Ведущий .NET разработчик, ICL

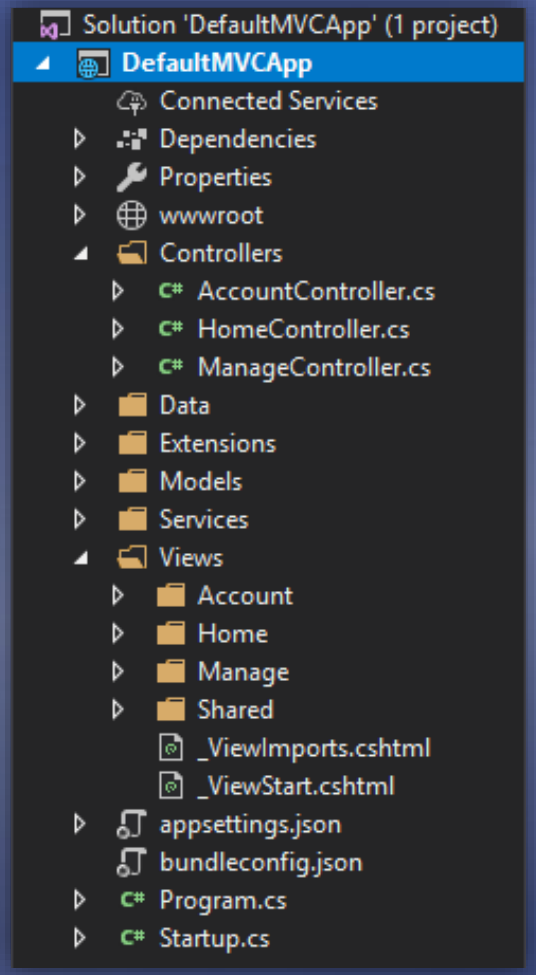
ASP.NET Web Application Templates



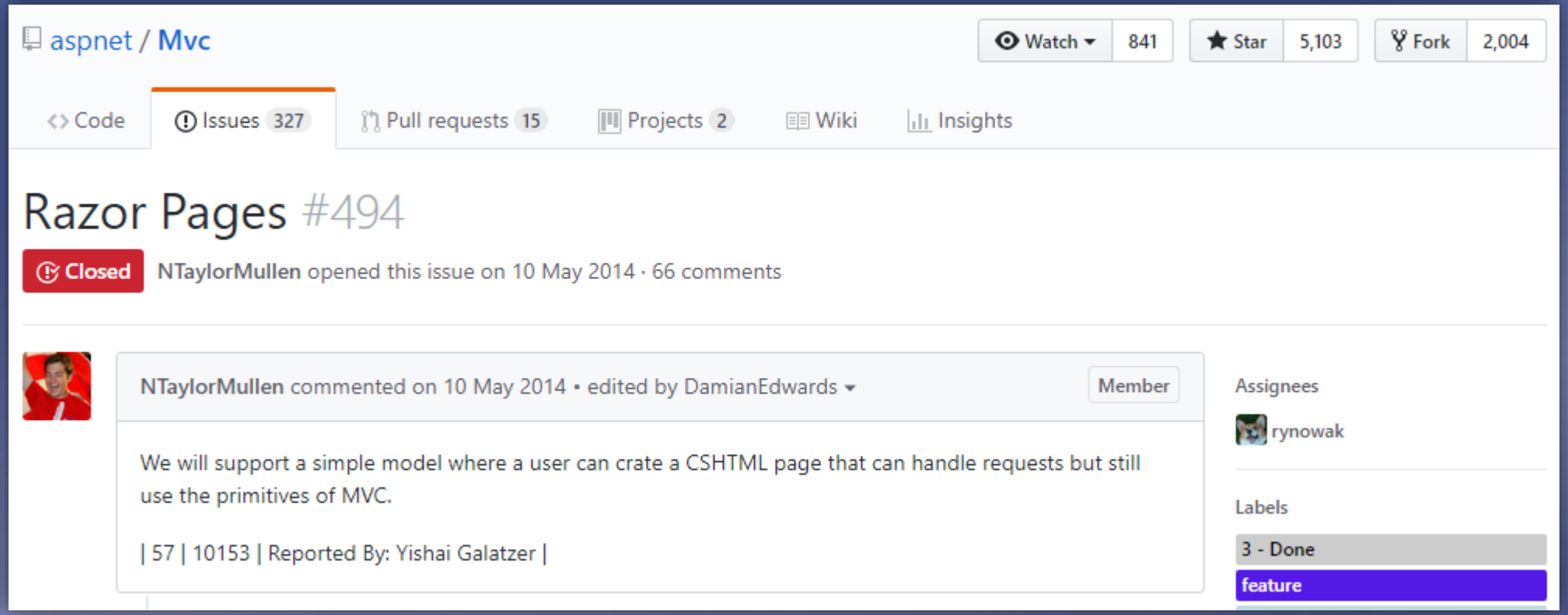
MVC – рекомендуемый и выбираемый по умолчанию шаблон проекта

ASP.NET MVC. Недостатки

- Большое количество файлов и папок
- Сложная маршрутизация
- Бесконечно разрастающиеся классы контроллеров
- Небольшие изменения в рамках одного компонента могут затронуть большое количество файлов
- Трудности в реализации «page-based» сценариев (статичные представления, CRUD, POST-Redirect-GET)

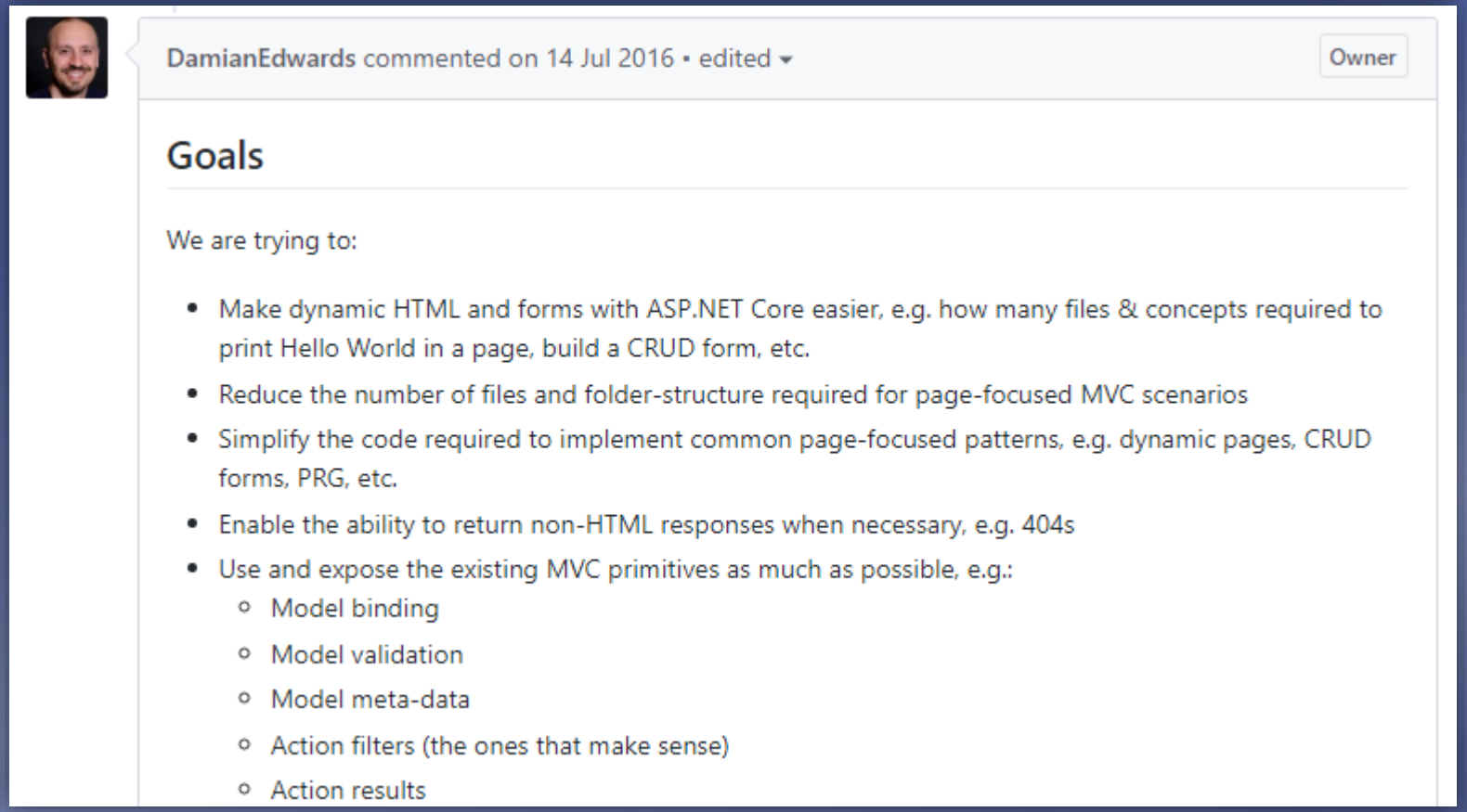


«Community requests»



https://github.com/aspnet/Mvc/issues/494

ASP.NET Team goals



DamianEdwards commented on 14 Jul 2016 • edited ▾ Owner

Goals

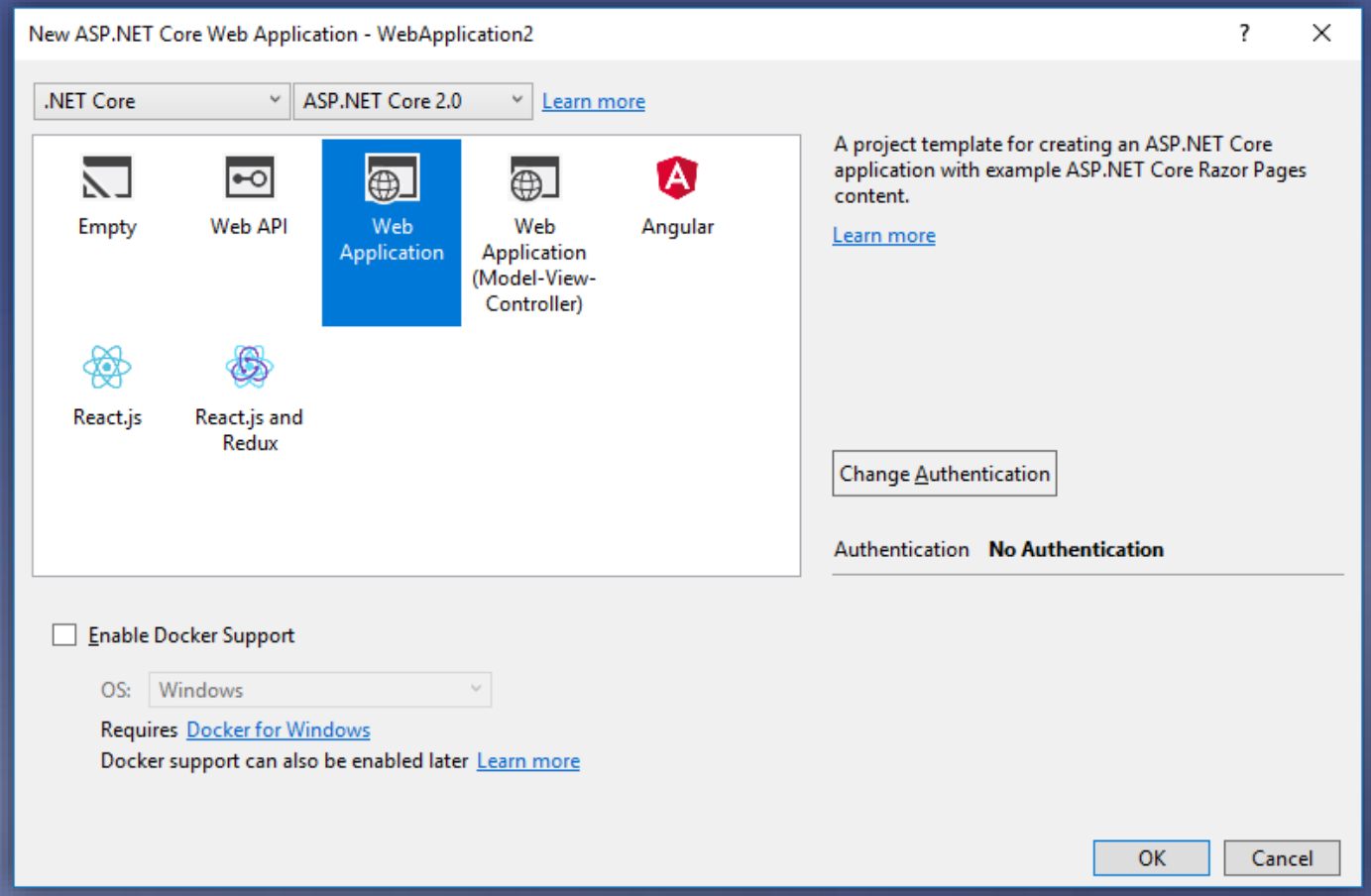
We are trying to:

- Make dynamic HTML and forms with ASP.NET Core easier, e.g. how many files & concepts required to print Hello World in a page, build a CRUD form, etc.
- Reduce the number of files and folder-structure required for page-focused MVC scenarios
- Simplify the code required to implement common page-focused patterns, e.g. dynamic pages, CRUD forms, PRG, etc.
- Enable the ability to return non-HTML responses when necessary, e.g. 404s
- Use and expose the existing MVC primitives as much as possible, e.g.:
  - Model binding
  - Model validation
  - Model meta-data
  - Action filters (the ones that make sense)
  - Action results

Non-goals:

- Create a scripted page framework to compete with PHP, etc.
- Hide C# with a DSL in Razor or otherwise
- New primitives should be usable in traditionally structured MVC apps too

ASP.NET Core Web Application Templates

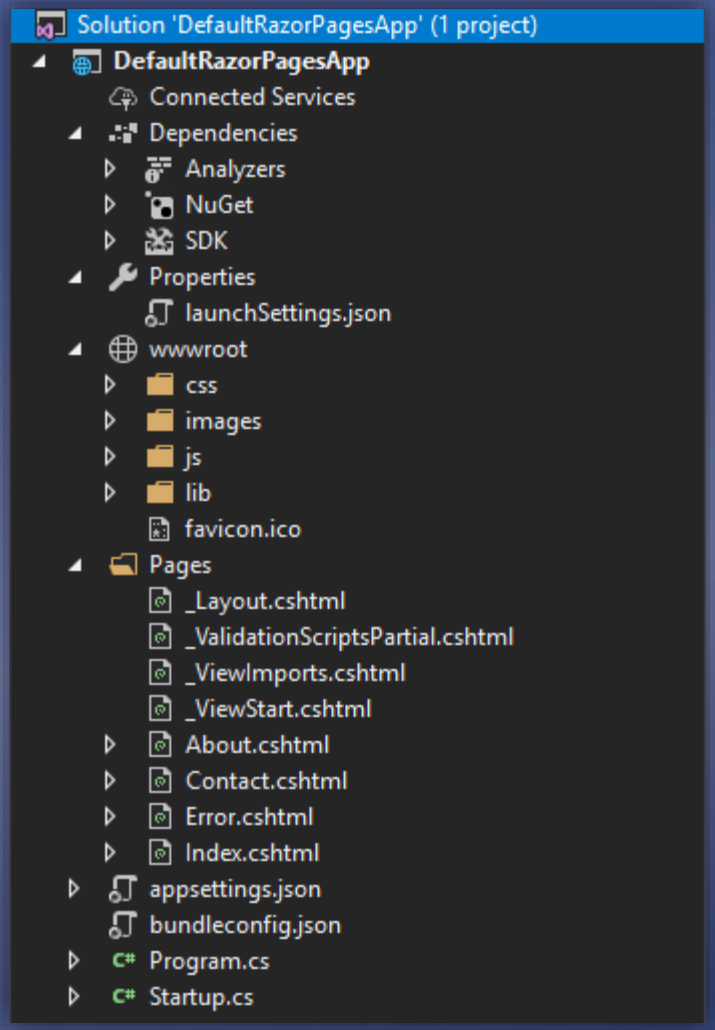


ASP.NET Core Razor Pages (Web Application) – рекомендуемый и выбираемый по умолчанию шаблон проекта

KZN DOT NET

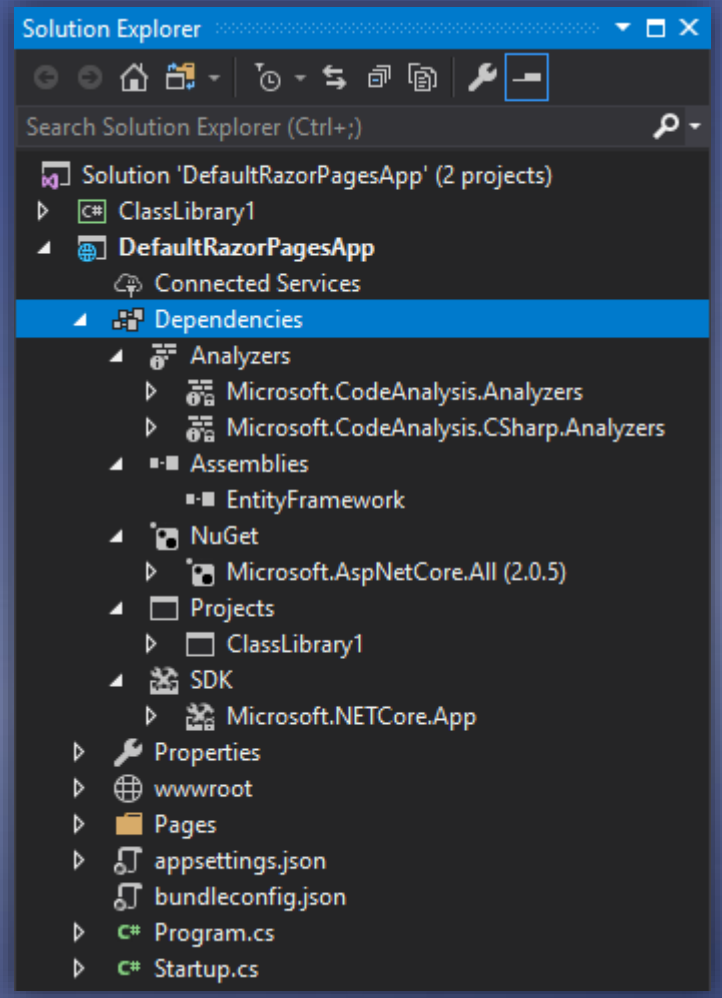
Структура приложения Razor Pages

- Project file (*.csproj)
- Dependencies
- Configuration files:
  - launchSettings.json
  - appSettings.json
  - bundleconfig.json
- «wwwroot» (static files)
- Program.cs
- Startup.cs
- Pages folder

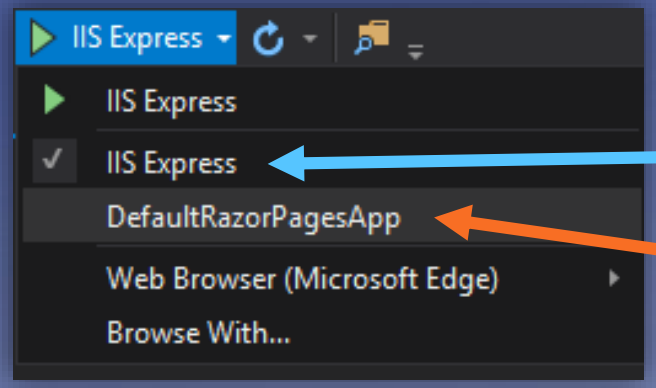


Dependencies

- Analyzers
- Assemblies
- Projects
- NuGet packages
- SDK



Configuration files. «launchSettings.json»

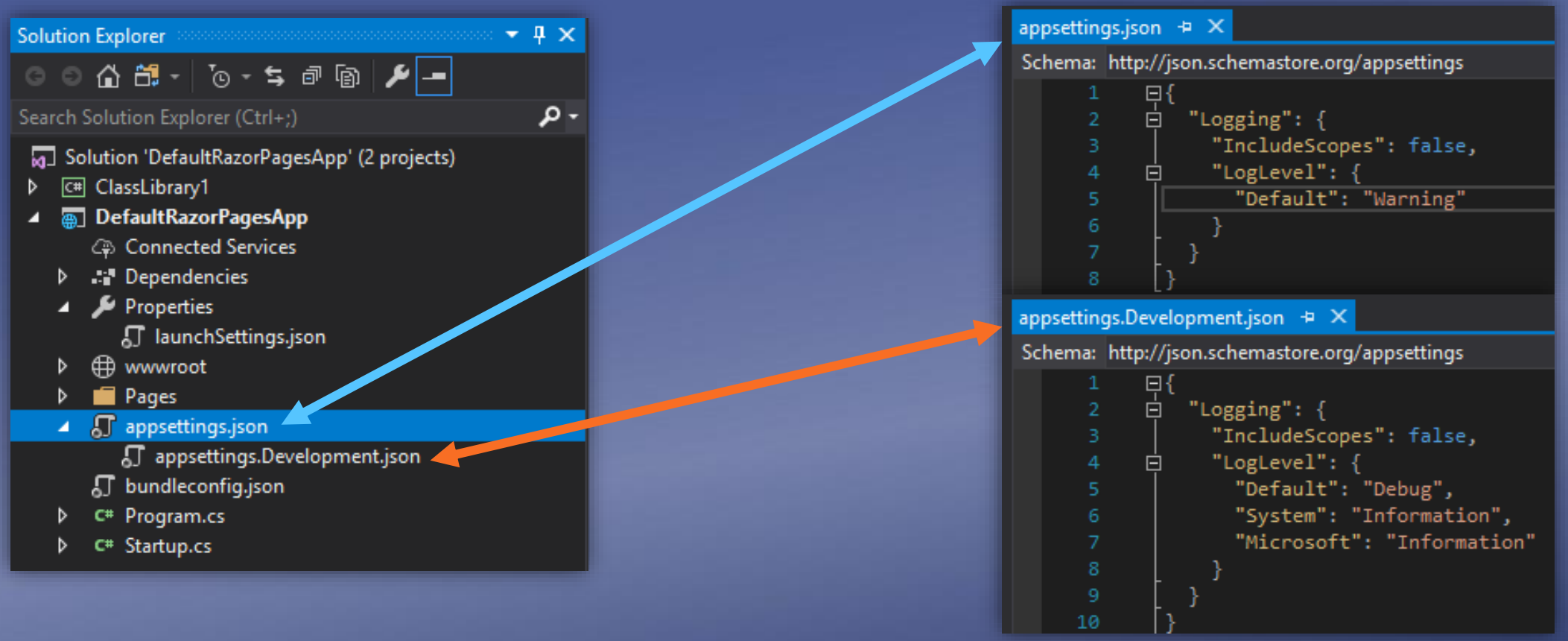


launchSettings.json

Schema: http://json.schemastore.org/launchsettings

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:46223/",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "DefaultRazorPagesApp": {
      "commandName": "Project",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      },
      "applicationUrl": "http://localhost:46224/"
    }
  }
}
```

Configuration files. «appSettings.json»



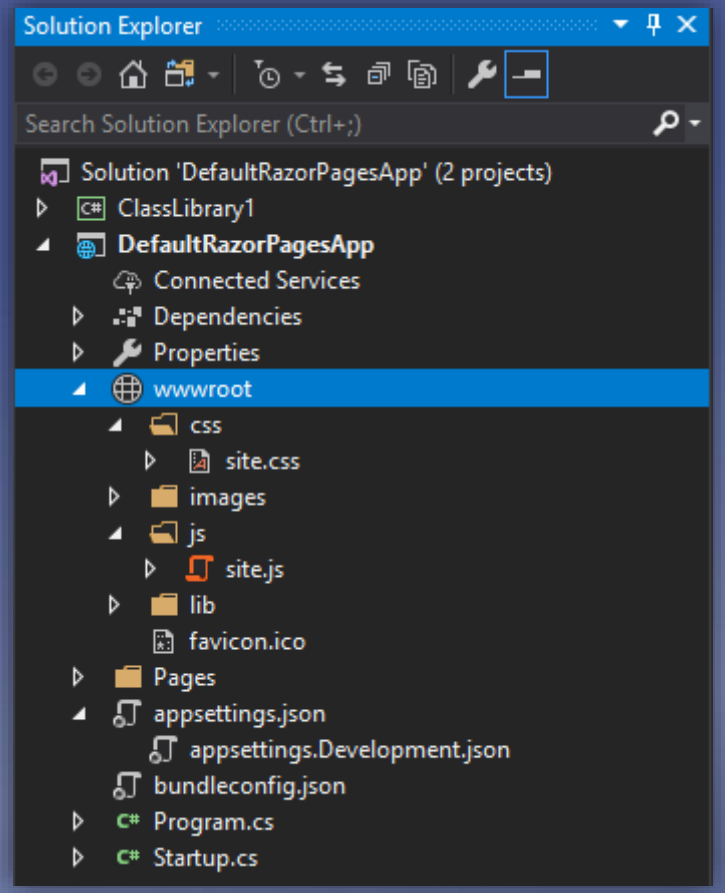
Configuration files. «bundleconfig.json»

bundleconfig.json

Schema: http://json.schemastore.org/bundleconfig

```
1   // Configure bundling and minification for the project.
2   // More info at https://go.microsoft.com/fwlink/?LinkId=808241
3   [
4     {
5       "outputFileName": "wwwroot/css/site.min.css",
6       // An array of relative input file paths. Globbing patterns supported
7       "inputFiles": [
8         "wwwroot/css/site.css"
9       ]
10    },
11    {
12      "outputFileName": "wwwroot/js/site.min.js",
13      "inputFiles": [
14        "wwwroot/js/site.js"
15      ],
16      // Optionally specify minification options
17      "minify": {
18        "enabled": true,
19        "renameLocals": true
20      },
21      // Optionally generate .map file
22      "sourceMap": false
23    }
24  ]
```

Static files («wwwroot»)



Для включения «фичи» необходимо вызвать «app.UseStaticFiles()» в классе «Startup.cs»

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseStaticFiles();

    app.UseMvc();
}
```

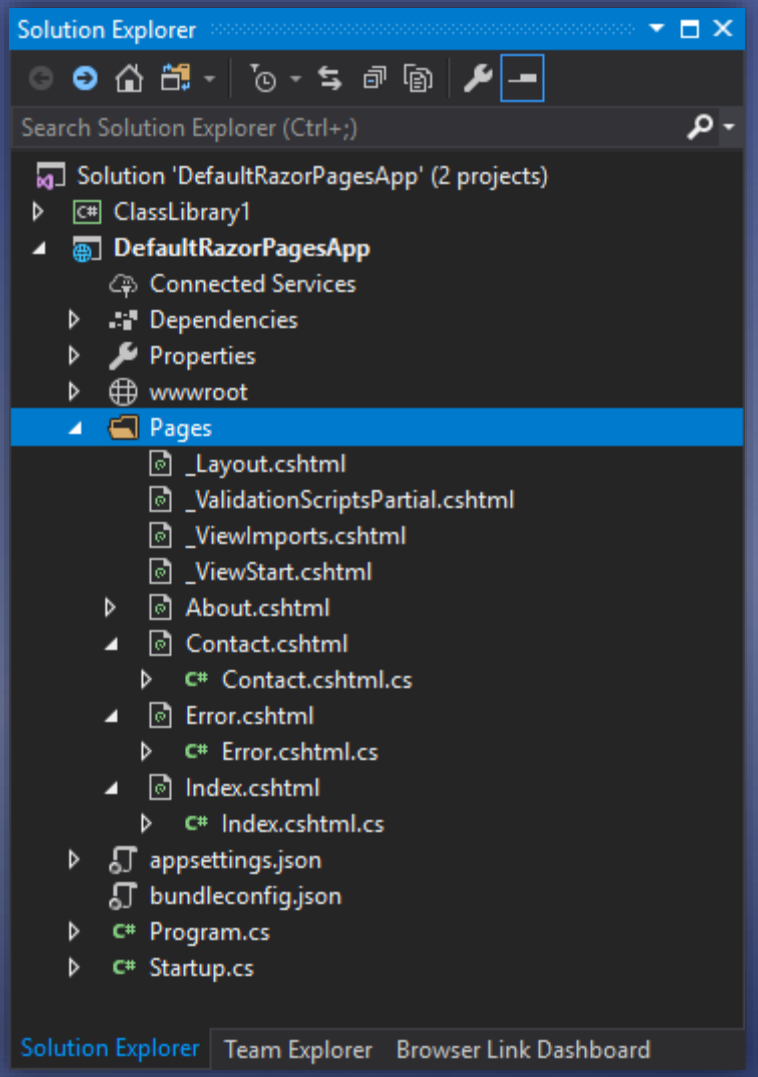
После ее включения запросы такого вида будут перенаправляться в «wwwroot»
Например:
http://<server_address>/css/site.css ->
/wwwroot/css/site.css

Pages folder. Razor Page

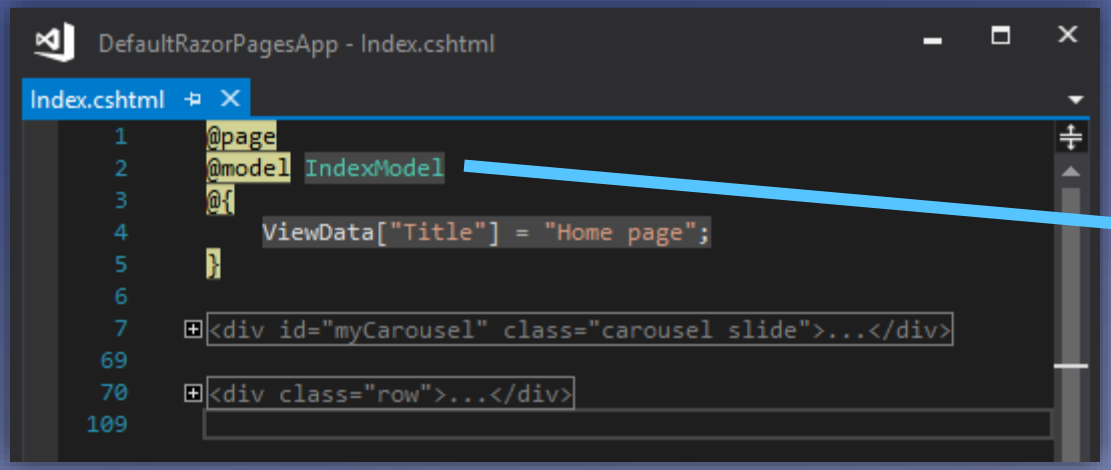
Признаки того, что перед нами Razor page:

- Отсутствует `_` в названии файла
- Расширение файла «cshtml»
- @page в первой строке файла

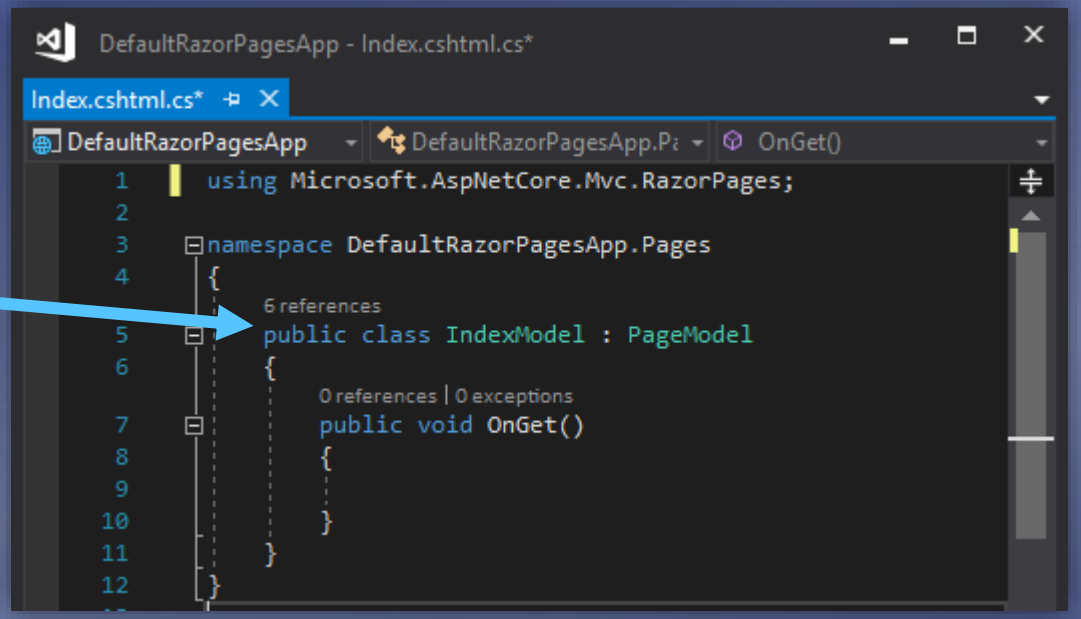
Page Path (‘/’ - Pages folder)	URL
/About.cshtml	/About
/SomeFolder/SomePage.cshtml	/SomeFolder/SomePage
/Index.cshtml	“/Index” OR “/”
/SomeFolder/Index.cshtml	/SomeFolder/



Razor Page with «Page Model»



«Index.cshtml» (Razor content Page)



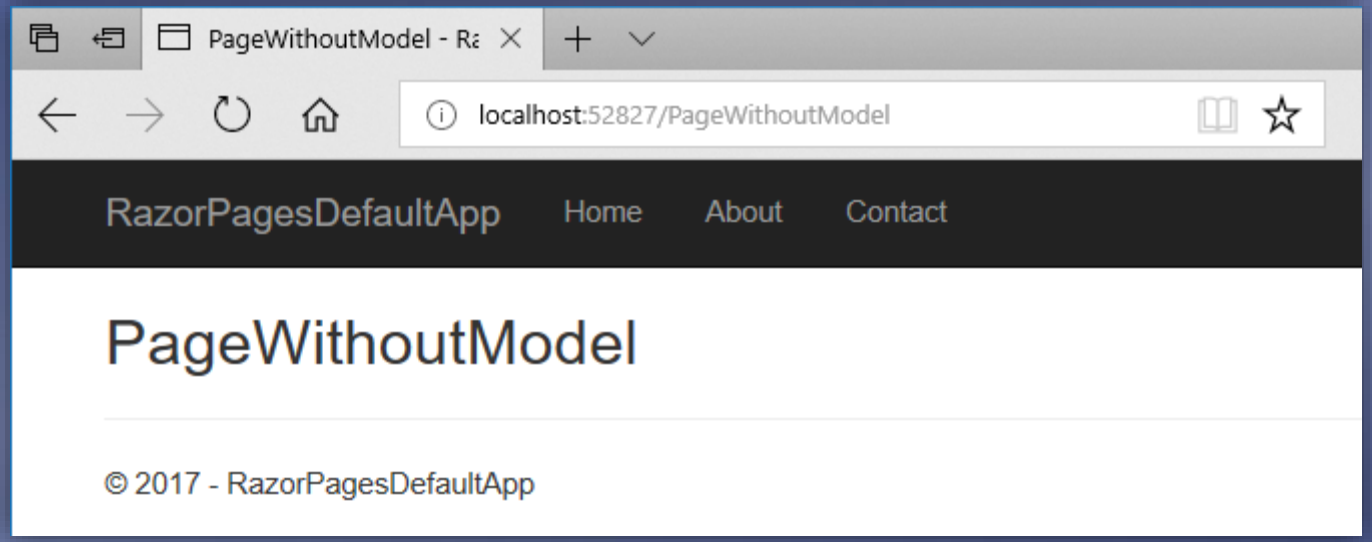
«Index.cshtml.cs» (Page Model)

Razor Page without «Page Model»

PageWithoutModel.cshtml

```
1   @page
2   @{
3       ViewData["Title"] = "PageWithoutModel";
4   }
5
6   <h2>PageWithoutModel</h2>
7
```

«PageWithoutModel.cshtml»
(Razor Page without @model directive)



Используется внутренняя реализация «PageModel» с «OnGet» обработчиком

Startup configuration

Startup.cs

RazorPagesTemplateApp | RazorPagesTemplateApp.Startup | ConfigureServices(IServiceCollecti

```
10  namespace RazorPagesTemplateApp
11  {
        1 reference
12      public class Startup
13      {
            0 references | 0 exceptions
14          public void ConfigureServices(IServiceCollection services)
15          {
16              services.AddMvc();
17          }
18
            0 references | 0 exceptions
19          public void Configure(IApplicationBuilder app, IHostingEnvironment env)
20          {
21              app.UseMvc();
22          }
23      }
```

It is MVC

- Routing
- Model Validation
- Action Descriptors
- Action Results
- Filters
- Model Binding
- Value Providers

- Layout & Partials
- HTML Helpers
- Tag Helpers
- ViewContext
- View Components

DEMO

DEMO

JobManagerContext.cs

DAL — DAL.JobManagerContext

```
1   using DAL.Models;
2   using Microsoft.EntityFrameworkCore;
3
4   namespace DAL
5   {
        18 references
6       public class JobManagerContext : DbContext
7       {
            0 references | 0 exceptions
8           public JobManagerContext(DbContextOptions<JobManagerContext> options)
9               : base(options)
10          {
11          }
12
            10 references | 0 exceptions
13          public DbSet<Job> Jobs { get; set; }
14      }
15  }
```

«DAL/JobManagerContext.cs»
(EF Core DbContext)

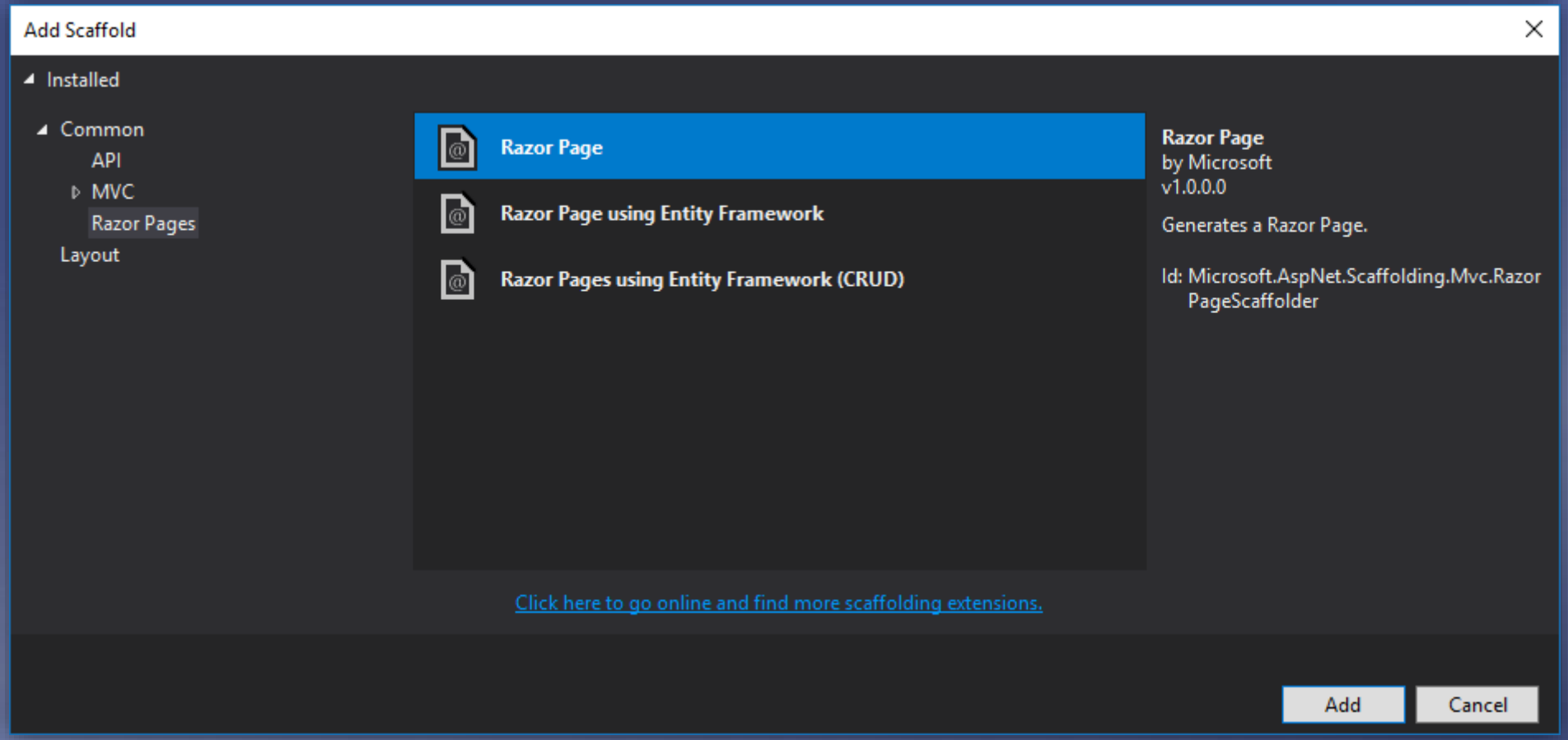
Job.cs | JobManagerContext.cs | Index.cshtml.cs | Index.csht

DAL

```
1   using System.ComponentModel.DataAnnotations;
2
3   namespace DAL.Models
4   {
        8 references
5       public class Job
6       {
            8 references | 0 exceptions
7           public int Id { get; set; }
8           [Required]
            4 references | 0 exceptions
9           public string Title { get; set; }
            4 references | 0 exceptions
10          public string Description { get; set; }
            4 references | 0 exceptions
11          public Priority Priority { get; set; }
            4 references | 0 exceptions
12          public bool IsActive { get; set; }
13          [Required]
            4 references | 0 exceptions
14          public string Assignee { get; set; }
15      }
16  }
```

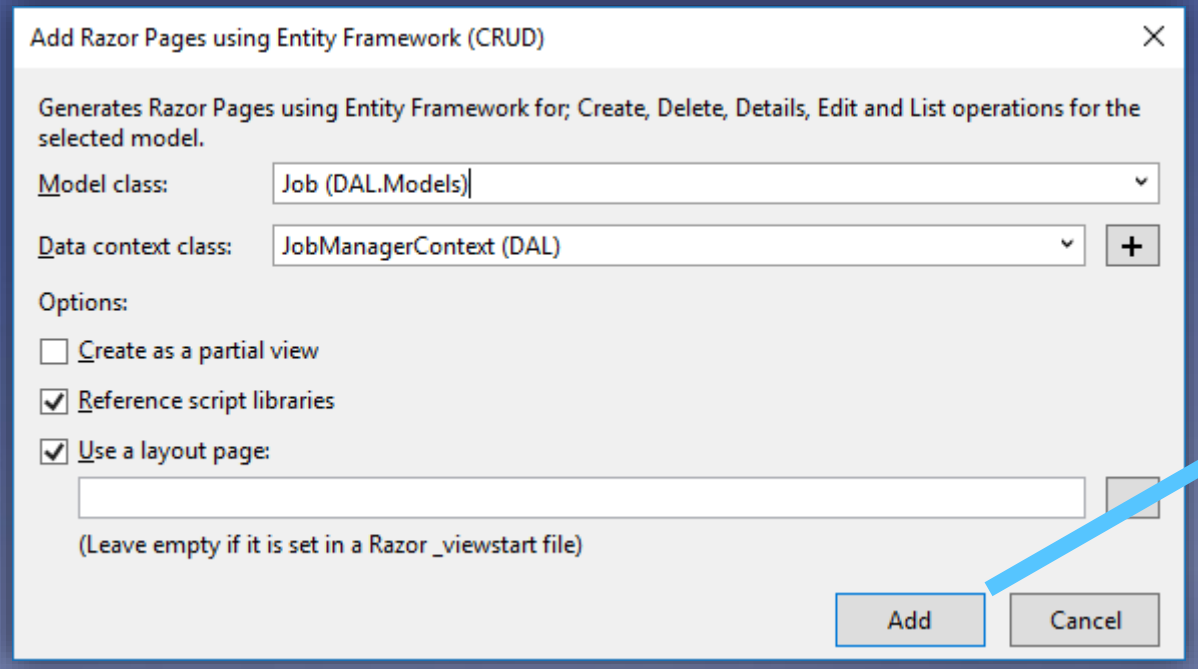
«DAL/Models/Job.cs»
(Entity)

DEMO

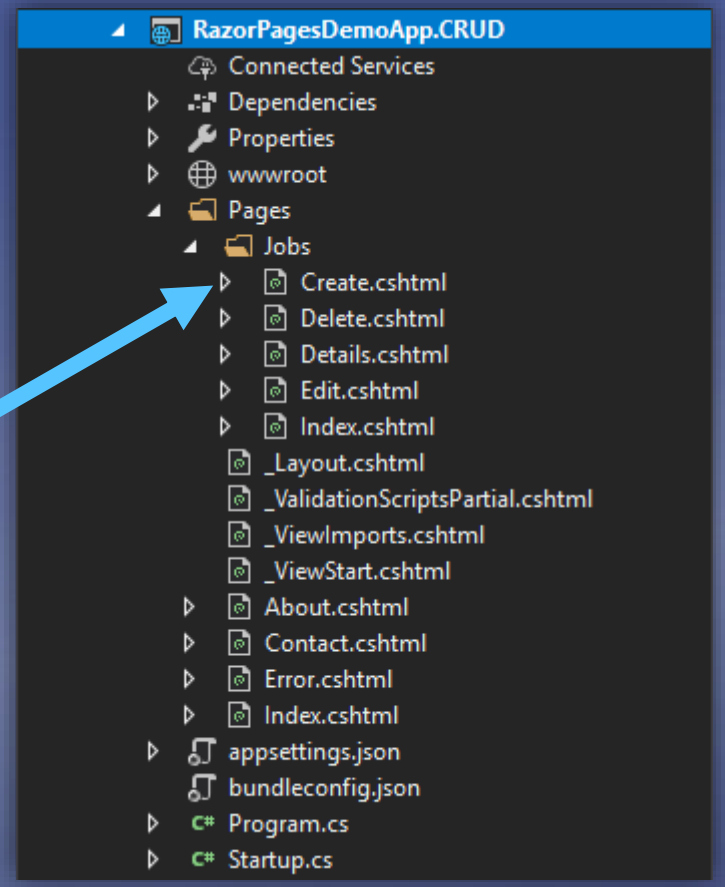


Add Razor Page dialog

DEMO



Generate Razor Pages using EF (CRUD) dialog



After generating

DEMO. Index

Job.cs | Index.cshtml.cs | Index.cshtml

RazorPagesDemoApp.CRUD

```
1   using ...
7
8   namespace RazorPagesDemoApp.CRUD.Pages.Jobs
9   {
        7 references
10      public class IndexModel : PageModel
11      {
12          private readonly JobManagerContext _context;
13
            0 references | 0 exceptions
14          public IndexModel(JobManagerContext context)
15          {
16              _context = context;
17          }
18
            7 references | 0 exceptions
19          public IList<Job> Job { get; set; }
20
            0 references | 0 exceptions
21          public async Task OnGetAsync()
22          {
23              Job = await _context.Jobs.ToListAsync();
24          }
25      }
26  }
```

Index.cshtml.cs | Index.cshtml

```
@page
@model RazorPagesDemoApp.CRUD.Pages.Jobs.IndexModel
@{ ViewData["Title"] = "Index"; }

<h2>Index</h2>
<p><a asp-page="Create">Create New</a></p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Job[0].Title)
            </th>
            ...
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Job)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Title)
                </td>
...
                <td>
                    <a asp-page="./Edit" asp-route-id="@item.Id">Edit</a> |
                    <a asp-page="./Details" asp-route-id="@item.Id">Details</a> |
                    <a asp-page="./Delete" asp-route-id="@item.Id">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

Razor Pages: New page-based framework in ASP.NET Core

KZN DOT NET

Handlers

Требования к методам:

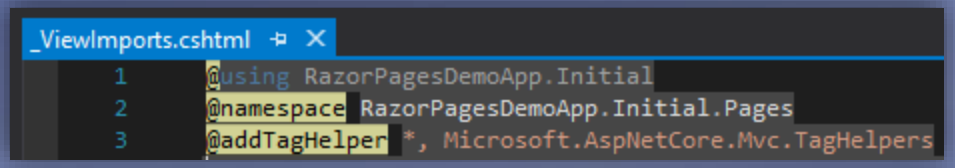
- Должны быть публичными
- Могут иметь любой тип возвращаемого значения, однако обычно либо возвращают void (или Task, если метод асинхронный), либо action result.

Default Convention:

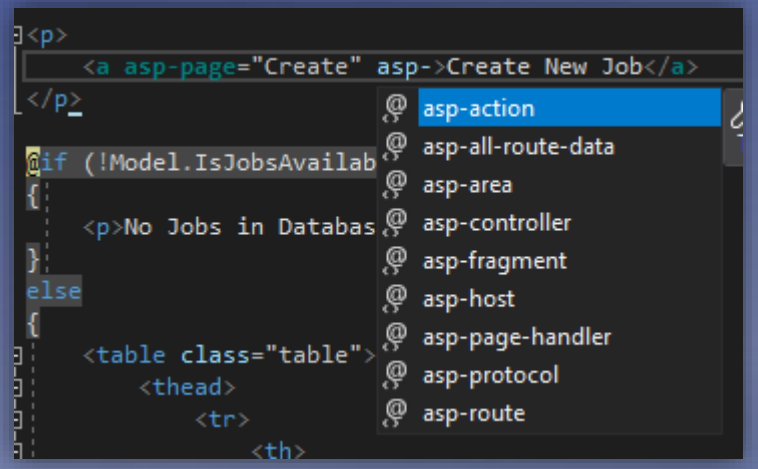
- On<HTTP Verb> || On<HTTP Verb>Async,
  HTTP Verb = Get, Post, Put, Delete

Tag Helpers

- Генерируют HTML разметку на сервере
- Большая схожесть с HTML (нет C# кода)
- Поддержка IntelliSense
- Не являются заменой HTML хелперам
- Можно создавать свои собственные «Tag Helper-ы»



Подключение всех «Tag Helper-ов» через «_ViewImports.cshtml»



«Кастомные» атрибуты «Anchor Tag Helper»

DEMO. Create

```cshtml
@page
@using DAL.Models;
@model RazorPagesDemoApp.CRUD.Pages.Jobs.CreateModel

@{
    ViewData["Title"] = "Create Job";
}

<h2>@ViewData["Title"]</h2>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Job.Title" class="control-label"></label>
                <input asp-for="Job.Title" class="form-control" />
                <span asp-validation-for="Job.Title" class="text-danger"></span>
            </div>
            <div class="form-group">...</div>
            <div class="form-group">...</div>
            <div class="form-group">...</div>
            <div class="form-group">...</div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </form>
    </div>
</div>
```

```cshtml
public class CreateModel : PageModel
{
    private readonly DAL.JobManagerContext _context;

    0 references | 0 exceptions
    public CreateModel(DAL.JobManagerContext context) => _context = context;

    [BindProperty]
    1 reference | 0 exceptions
    public Job Job { get; set; }

    0 references | 0 exceptions
    public async Task<IActionResult> OnPostAsync()
    {
        if (!ModelState.IsValid)
            return Page();

        _context.Jobs.Add(Job);
        await _context.SaveChangesAsync();

        return RedirectToPage("./Index");
    }
}
```

TempData

«CreateModel»:

1. Добавляем свойство «Message», помечая его при этом атрибутом «TempData»
2. Задаем значение свойству перед перенаправлением на другу страницу

```
[TempData]
1 reference | 0 exceptions
public string Message { get; set; }

0 references | 0 exceptions
public async Task<IActionResult> OnPostAsync()
{
    if (!ModelState.IsValid)
        return Page();

    _context.Jobs.Add(Job);
    await _context.SaveChangesAsync();

    Message = $"Job with title '{Job.Title}' is created now";
    return RedirectToPage("./Index");
}
```

TempData

«IndexModel»:

3. Добавляем свойство «Message», помечая его при этом атрибутом «TempData»

```
[TempData]
2 references | 0 exceptions
public string Message { get; set; }

1 reference | 0 exceptions
public bool IsMessageExists => !string.IsNullOrEmpty(Message);
```

«Index.cshtml»:

4. Выводим значение свойства «Message» на страницу

```
@if (Model.IsMessageExists)
{
    <p><div class="alert alert-success">@Model.Message</div></p>
}
```

Route Data

Default settings:
http://<server>/Jobs/Edit?id=1
RouteData: {page: /Object/Edit}
QueryString: {id: 1}

После изменения «Route Template» страницы «Edit» при помощи добавления параметра маршрута «id»:
http://<server>/Jobs/Edit/1
RouteData: {page: /Object/Edit; id: 1}
QueryString: {}

```
@page "{id}"
@model RazorPagesDemoApp.CRUD.Pages.Jobs.EditModel

@{
    ViewData["Title"] = "Edit";
}

<h2>Edit</h2>
```

«Edit.cshtml»

Named Handler Methods

```
public class EditModel : PageModel
{
    private readonly DAL.JobManagerContext _context;

    0 references | 0 exceptions
    public EditModel(DAL.JobManagerContext context) => _context = context;

    [BindProperty]
    7 references | 0 exceptions
    public Job Job { get; set; }

    [TempData]
    2 references | 0 exceptions
    public string Message { get; set; }

    0 references | 0 exceptions
    public async Task<IActionResult> OnGetAsync(int? id)...

    0 references | 0 exceptions
    public async Task<IActionResult> OnPostAsync()...

    0 references | 0 exceptions
    public async Task<IActionResult> OnPostAssigneeToAdminAsync(int id)...
```

«EditModel»

Naming convention:

- On<HTTP_Verb><Handler_Name>
- On<HTTP_Verb><Handler_Name>Async

Named Handler Methods

Способ 1: Несколько «обработчиков» на одну форму

```
<form method="post">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <input type="hidden" asp-for="Job.Id" />
    <div class="form-group">...</div>
    <div class="form-group">...</div>
    <div class="form-group">...</div>
    <div class="form-group">...</div>
    <div class="form-group">...</div>
    <div class="form-group">
        <button class="btn btn-default">Save</button> |
        <button asp-page-handler="AssigneeToAdmin" class="btn btn-default">Assignee To Administrator</button>
    </div>
</form>
```

Способ 2: Для каждого «обработчика» своя форма

```
]<form asp-page-handler="AssigneeToAdmin" method="post">
    <button class="btn btn-default">Assignee To Administrator By Form</button>
</form>
```

Request Verification. XSRF/CSRF

Традиционный подход:

- «@Html.AntiForgeryToken»
- Применение атрибута «ValidateAntiForgeryToken» там, где нужно (Action, Controller, Globally)

Razor Pages:

- Для генерации токена достаточно использовать «Form Tag Helper»
- Проверка происходит автоматически

```
<form asp-page-handler="AssigneeToAdmin" method="post">
    <button class="btn btn-default">Assignee To Administrator By Form</button>
</form>
```

```
<form action="/Jobs/Edit/4003?handler=AssigneeToAdmin" method="post">
  <button class="btn btn-default">Assignee To Administrator By F…</button>
  <input name="__RequestVerificationToken" type="hidden" value="CfDJ8JMnC35bAbxDtke-
  1aM44VYGeq77Wr1nW7q70nF4m0nCwt13_iszfwgvHtL48lWM9w_2qkyRsG4VZ0X8W43zT77c" />
</form>
```

Filters

- Authorization filters
- Resource filters
- Exception filters
- Action filters (не поддерживаются в Razor Pages)
- Result Filters
- Page Filters (поддерживаются только в Razor Pages)

```
public class MyPageFilter : IPageFilter
{
    // Called after the handler method executes, before the action result.
    0 references
    public void OnPageHandlerExecuted(PageHandlerExecutedContext context)
    {
    }

    // Called before the handler method executes, after model binding is complete.
    0 references
    public void OnPageHandlerExecuting(PageHandlerExecutingContext context)
    {
    }

    // Called after a handler method has been selected, but before model binding occurs.
    0 references
    public void OnPageHandlerSelected(PageHandlerSelectedContext context)
    {
    }
}
```

Вызываемые методы в рамках «Page Filter»

Configuration

- Изменить корневую папку для Razor Pages на папку Content

```
services.AddMvc().AddRazorPagesOptions((options) =>
{
    options.RootDirectory = "/Content";
});
```

- Задание соглашений по авторизации

```
services.AddMvc().AddRazorPagesOptions((options) =>
{
    options.Conventions.AuthorizePage("/Contact");
    options.Conventions.AuthorizeFolder("/Private").AllowAnonymousToPage("/Private/Public");
    options.Conventions.AllowAnonymousToPage("/Private/PublicPage");
    options.Conventions.AllowAnonymousToFolder("/Private/PublicPages");
});
```

- Добавление «Route Template» к странице

```
services.AddMvc().AddRazorPagesOptions((options) =>
{
    options.Conventions.AddPageRoute("/Jobs/Edit", "Jobs/Update/{id?}");
});
```

MVC контроллеры. Как подключить в рамках приложения Razor Pages

Необходимо добавить маршрут со следующим шаблоном вида: «{controller=Home}/{action=Index}/{id?}"»

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
// Or
app.UseMvcWithDefaultRoute();
```

«Startup.cs»

Что будет нового в версии ASP.NET Core 2.1 Razor Pages?

- Razor Pages in an «Area»
- Razor Pages in a class library
- Support «/Pages/Shared»
- «BindPropertyAttribute» on page model or controller
- Implement «IPageFilter» on page models
- Support absolute routes on Razor Pages

Полезные ссылки:

- https://github.com/staslado/kzndotnet1.Razor-Pages

- https://docs.microsoft.com/en-us/aspnet/core/mvc/razor-pages/
- https://www.learnrazorpages.com/
- https://github.com/aspnet/Mvc
- https://msdn.microsoft.com/en-us/magazine/mt842512.aspx
- http://www.bipinjoshi.net/articles/16ecfe49-98df-4305-b53f-2438d836f0d0.aspx

Спасибо за внимание☺