Basics of ASP.NET

Ilya Loshkarev 2024

Basics of ASP.NET

- Common Gateway Interface
- IIS & ASP
- Принципы архитектуры ASP.NET
- Конфигурация запуска ASP.NET
- Razor Pages

Common Gateway Interface

CGI — спецификация интерфейса, используемого внеш программой для связи с веб-сервером.

RFC 3875

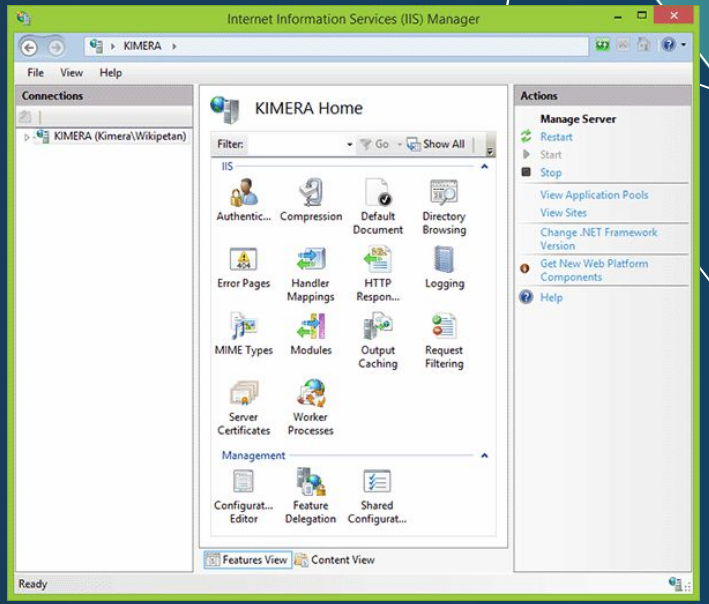
Шлюзом тут является веб-сервер, который получает запрос от клиента, преобразует в CGI-форму, вызывает обработчик и конвертирует его ответ из CGI-формы в форму HTTP-ответа клиенту.

http://www.example.com/cgi-bin/accept.cgi

Internet Information Services(Server)

IIS — проприетарный набор серверов для нескольких служб Интернета от компании Microsoft.

IIS поддерживает протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP



Технологий создания веб-приложений на IIS

CGI — стандартная межплатформенная низкоуровневая технология создания динамических веб-страниц

FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения

ISAPI — низкоуровневая технология, предоставляющая полный доступ ко всем возможностям IIS, возможность разработки веб-приложений в машинном коде и возможность переопределения части функций IIS и добавления к нему функций

Подсистема исполнения скриптов ASP и подсистема ASP.NET выполнены как модули ISAPI

Active Server Pages

Технология, предложенная компанией Microsoft в 1996 году для создания Web-приложений. Эта технология основана на внедрении в обыкновенные веб-страницы специальных элементов управления, допускающих программное управление

```
<%  Response.write "Hello World!"  %>
```

ASP.NET

It was first released in January 2002 with version 1.0 of the .NET Framework

In 2016, Microsoft released ASP.NET Core as ASP.NET's successor. This new version is a re-implementation of ASP.NET as a modular web framework

Шаблоны приложений ASP.NET

MVC - базовая модель страничных веб-приложений

Razor Pages - упрощенная модель для создания страничных веб-приложений

Web API - MVC без пользовательских страниц

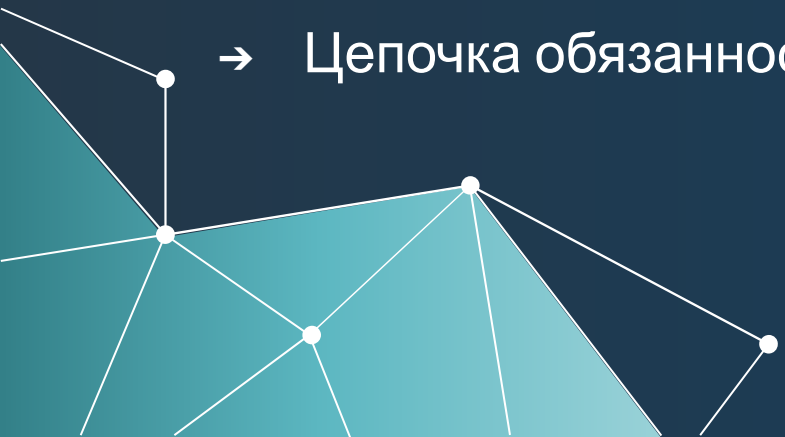
SignalR - MVC с использованием библиотеки обратной связи сервера с клиентом

React/Angular SPA - готовые шаблоны одностраничных приложений с подключением клиентских JS библиотек к серверу на ASP.Net Core

Blazor - создание клиентских приложений на C# (WebAssembly)

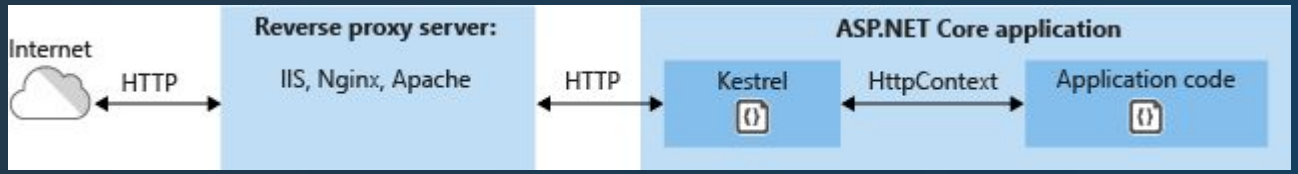
Принципы архитектуры ASP

- Универсальный сервер запросов (Generic Host)
- Внедрение зависимостей (Dependency Injection)
- Цепочка обязанностей (Chain of Responsibility)



Универсальный сервер запросов (Generic Host)

Единая точка доступа к ресурсам приложения: управляет конфигурацией, логированием и запуском веб-сервера



https://docs.microsoft.com/ru-ru/aspnet/core/fundamentals/host/generic-host

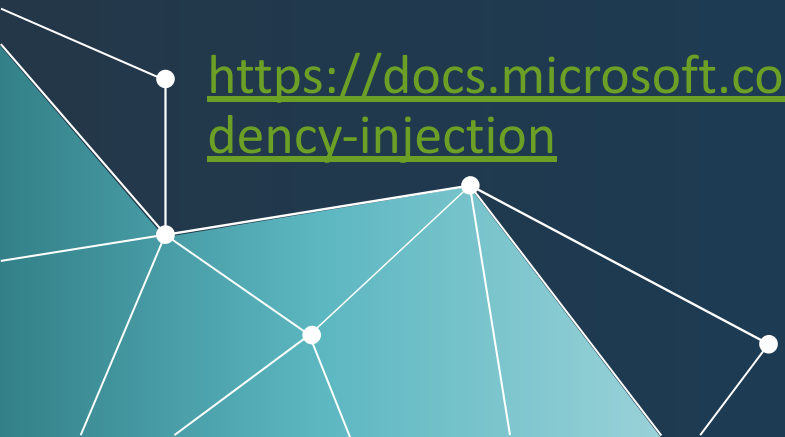
Универсальный сервер запросов

```
Host.CreateDefaultBuilder(args)
    .ConfigureWebHostDefaults(webBuilder =>
    {
        webBuilder.UseStartup<Startup>();
    })
    .Build()
    .RunAsync();
```

Внедрение зависимостей (Dependency Injection)

Приложение поддерживает внедрение зависимостей для поддержки принципа инверсии зависимостей

Позволяет подменять реализацию на лету, и упрощает разделение уровней приложения



https://docs.microsoft.com/ru-ru/aspnet/core/fundamentals/dependency-injection

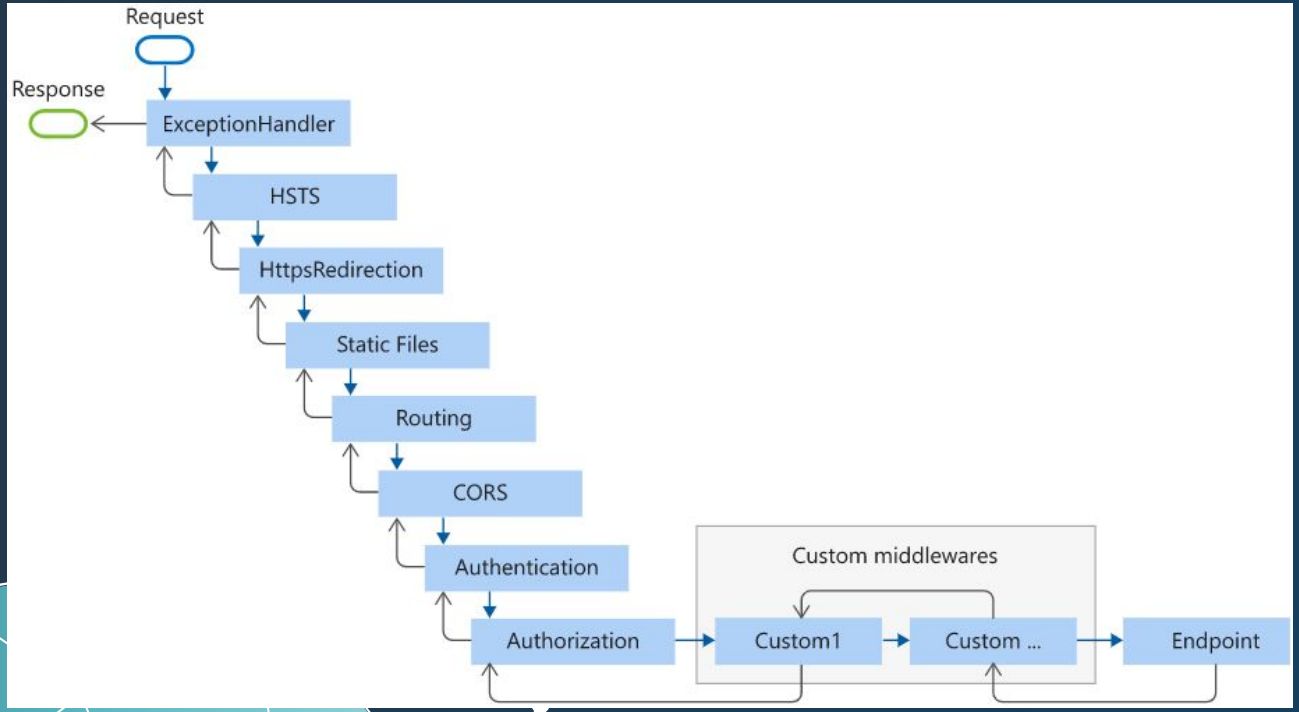
Цепочка обязанностей (Chain of Responsibility)

Обработка запросов осуществляется конвейером обработчиков, которые добавляются при инициализации приложения

Позволяет гибко настраивать необходимый набор обработчиков внутри базового веб-сервера

https://docs.microsoft.com/ru-ru/aspnet/core/fundamentals/middleware

Цепочка обязанностей



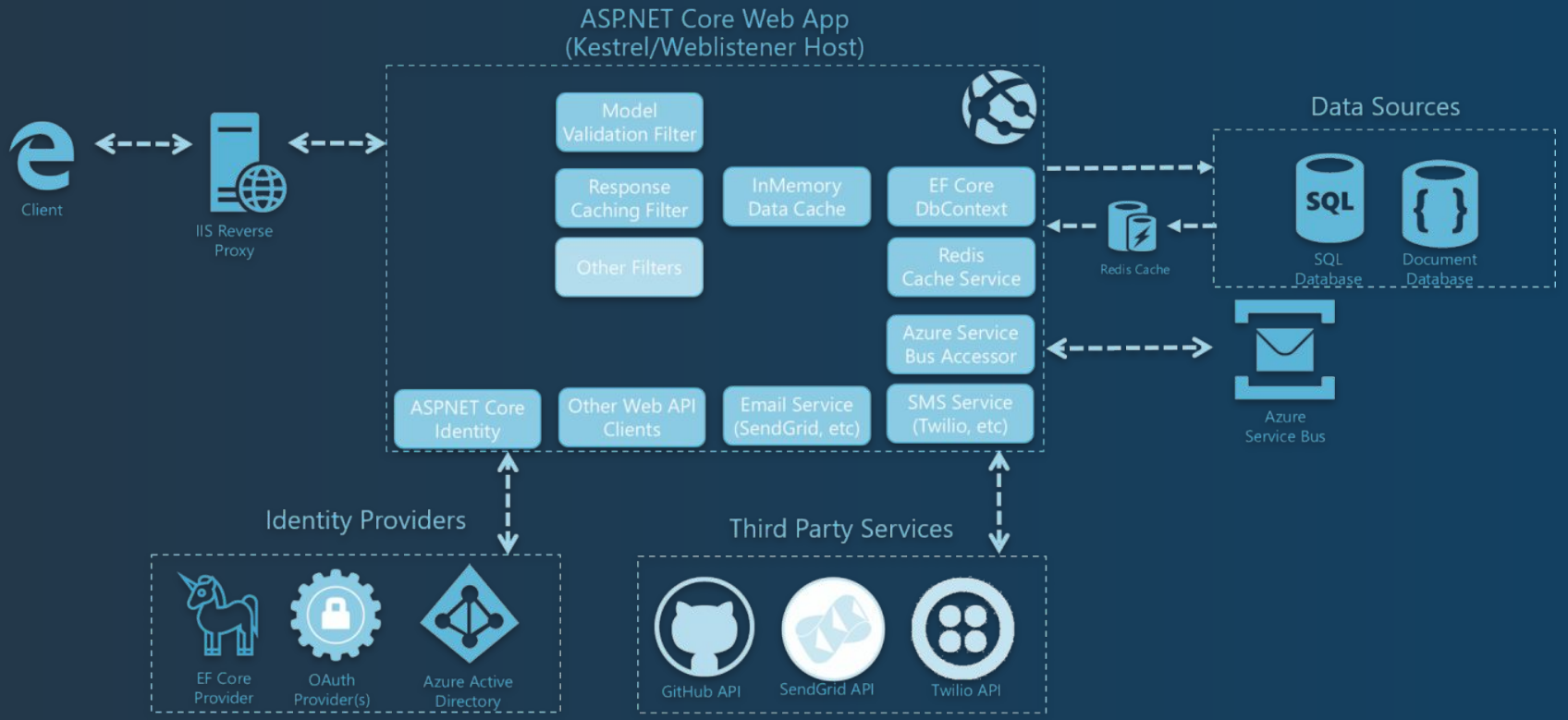
Middleware

Компоненты конвейера, которые отвечают за обработку запроса.

Middleware может либо передать запрос далее следующему в конвейере компоненту, либо выполнить обработку и закончить работу конвейера.

Также возможно передать запрос дальше по конвейеру, а затем вернуться в код компонента, для завершения обработки

ASP.NET Core Architecture



Конфигурация ASP приложения / Настройка хоста

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

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

Конфигурация ASP приложения / Настройка конвейера обработчиков

```
public class Startup
{
    public void Configure(IApplicationBuilder app,
                          IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Error");
            app.UseHsts();
        }
...
```

Конфигурация ASP приложения / Настройка конвейера обработчиков

```
…

app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapGet("/", async context =>
                {
                    await context.Response.WriteAsync("Hello World!");
                });
});

}
}
```

Middleware /2

Компоненты middleware конфигурируются с помощью методов расширений ***Run***, ***Map*** и ***Use*** объекта ***IApplicationBuilder***

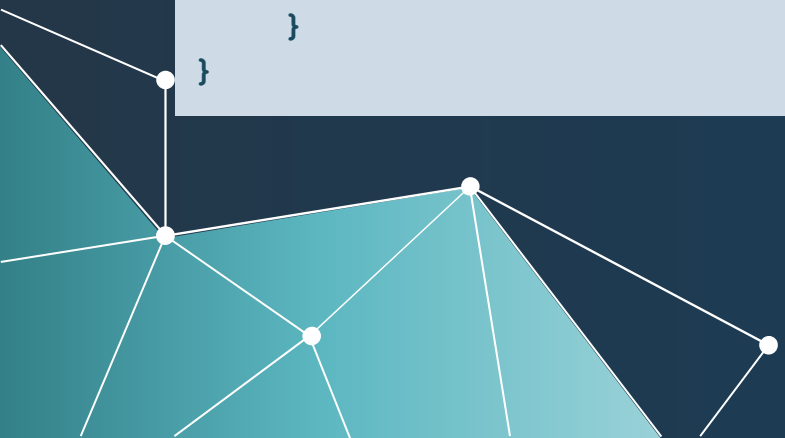
Run добавляет компонент в конвейер, но такие компоненты не вызывают никакие другие компоненты и не передают дальше обработку запроса

Use добавляет компоненты, которые также обрабатывают запрос, но в нем может быть вызван следующий в конвейере запроса компонент

Map (и методы ***Map###***) применяется для сопоставления пути запроса с определенным делегатом, который будет обрабатывать запрос по этому пути

Middleware /Run

```
public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        app.Run(async (context) =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    }
}
```



Middleware /Use

```
public void Configure(IApplicationBuilder app)
{
    int x = 5;
    int y = 8;
    int z = 0;
    app.Use(async (context, next) =>
    {
        z = x * y;
        await next.Invoke();
    });

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync($"x * y = {z}");
    });
}
```

Middleware /Map

```
public void Configure(IApplicationBuilder app)
{
    app.Map("/index", Index);
    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Page Not Found");
    });
}

private static void Index(IApplicationBuilder app)
{
    app.Run(async context =>
    {
        await context.Response.WriteAsync("Index");
    });
}
```

Конфигурация ASP приложения / Инициализация сервисов (DI)

```
public class Startup
{

...
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddRazorPages();
    }

}
```

Razor Pages

Модель страницы Razor сочетает в себе обязанности контроллера и модели представления.

Razor Pages упрощает процесс создания отдельных страниц в приложения ASP.NET Core, предоставляя при этом все архитектурные компоненты ASP.NET Core MVC.

Razor Pages и MVC

Razor Pages встроены в ***ASP.NET Core MVC*** и используют те же функции для маршрутизации, привязки модели, фильтров, авторизации и т. д. Но вместо отдельных папок и файлов для контроллеров, моделей, представлений и т. д., а также использования маршрутизации на основе атрибутов, Razor Pages помещаются в одну папку ("/Pages"), выстраивают маршрут на основе своего относительного расположения в этой папке и обрабатывают запросы с помощью обработчиков, а не действий контроллера.

https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps

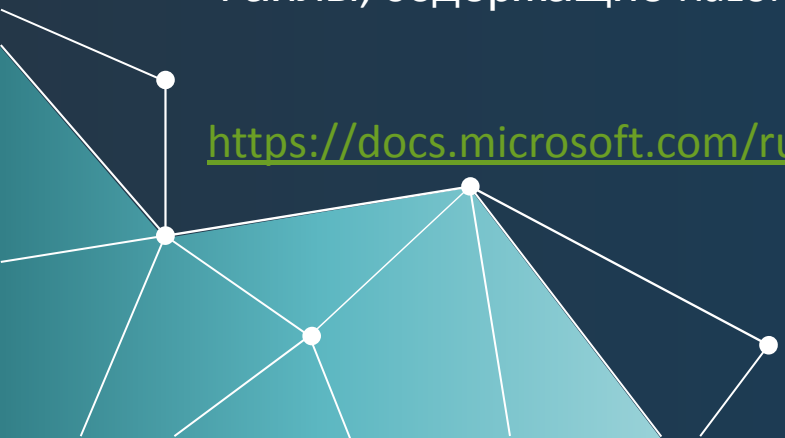
Razor-разметка

Razor — это синтаксис разметки для внедрения серверного кода на веб-страницы

Синтаксис ***Razor*** состоит из *Razor-разметки, C# и HTML*

Файлы, содержащие Razor обычно, имеют расширение ***.cshtml***

https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor



Пример Razor-разметки

```
@{
    var quote = "Getting old ain't for wimps! - Anonymous";
}

<div>Quote of the Day: @quote</div>
```

Сгенерированный Razor-контроллер

```
public class _Views_Something_cshtml : RazorPage<dynamic>
{
    public override async Task ExecuteAsync()
    {
        var output = "Getting old ain't for wimps! - Anonymous";

        WriteLiteral("/r/n<div>Quote of the Day: ");
        Write(output);
        WriteLiteral("</div>");
    }
}
```