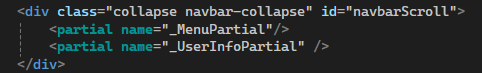
1. Для чего используется tag-helper «asp-action»?

asp-action: указывает на действие контроллера

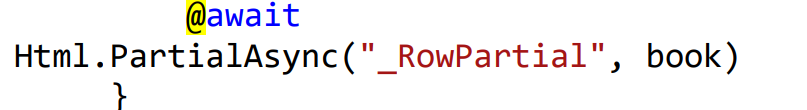
2. Что такое частичное представление?

Частичное представление – это файл на языке Razor, который генерирует разметку HTML внутри другой разметки.



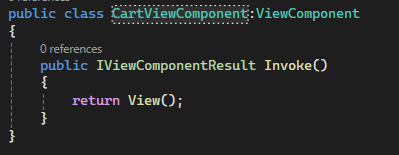


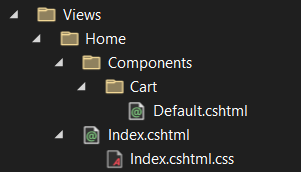
3. Как разместить частичное представление в разметке представления?



4. Чем отличается компонент представления от частичного представления?

Компоненты представлений - это классы, которые обеспечивают логику приложения для поддержки частичных представлений или для внедрения небольших фрагментов данных HTML или JSON в родительское представление.





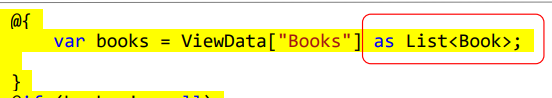


5. Как разместить компонент представления в разметке представления?

6. Где располагается файл представления компонента?

7. Чем отличаются объекты ViewData и ViewBag?

ViewData представляет собой словарь, формируемый динамически, и доступный как свойство в контроллере и в представлении. ViewData наследуется от ViewDataDictionary, следовательно доступ к данным осуществляются с помощью пары «ключ-значение».



ViewBag – это «обертка» ViewData, позволяющая создавать динамические свойства. Использование ViewBag аналогично ViewData, за исключением того, что доступ к данным осуществляется через свойства: ViewBag.SomeData = "Данные во ViewBag";

8. Что такое модель представления?

Наиболее надежный подход при передаче данных представлению — указание типа модели в представлении. Такая модель называется viewmodel. Экземпляр типа viewmodel передается в представление из действия контроллера.

В представлениях Razor для использования свойства Model необходимо указать, к какому классу относится модель. Для этого используется ключевое слово @model, например: @model IEnumerable

9. Как указать в представлении класс модели?

10. Как получить доступ к членам класса модели представления?

Лаба 3:  
1. Как зарегистрировать сервис в ASP.Net Core?

* AddTransient<service, implType>()
* AddTransient<service>()
* AddTransient<service>(factoryFunc)
* Поставщик услуг должен создать новый экземпляр типа **всякий раз**, когда ему необходимо разрешить внедрение зависимости
* AddScoped<service, implType>()
* AddScoped<service>()
* AddScoped<service>(factoryFunc)
* Этот жизненный цикл создает **один объект** класса реализации, который используется **для разрешения всех зависимостей**, связанных с одной областью, что обычно означает **один HTTP-запрос**
* AddSingleton<service, implType>()
* AddSingleton<service>()
* AddSingleton<service>(factoryFunc)
* AddSingleton<service>(Instance)
* Жизненный цикл Singleton гарантирует, что **один объект** используется для разрешения **всех зависимостей** для данного типа службы.

2. Чем отличается Transient сервис от Scoped сервиса?

3. Как внедрить сервис в метод (Action) контроллера?

Через использование dependency injection  
private readonly IUserService \_userService; // Внедрение зависимостей через конструктор public UsersController(IUserService userService) { \_userService = userService; } // Пример метода Action public IActionResult GetUser(int id) { var user = \_userService.GetUser(id); return View(user); // Возвращаем представление с данными пользователя }

4. Как передать данные от клиента в метод контроллера?

[HttpGet("users/{id}")] public IActionResult

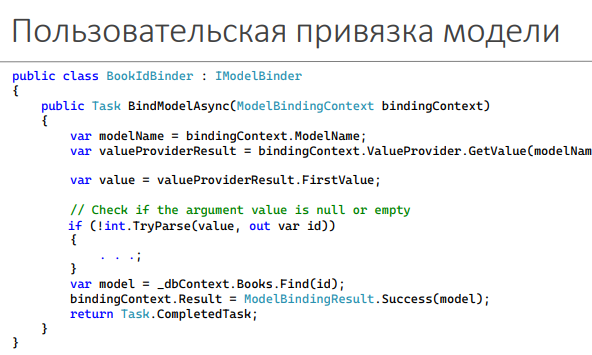
GetUser(int id)

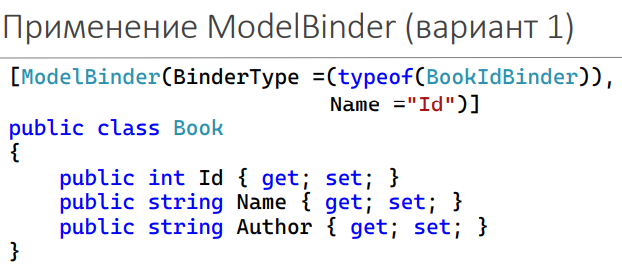
{

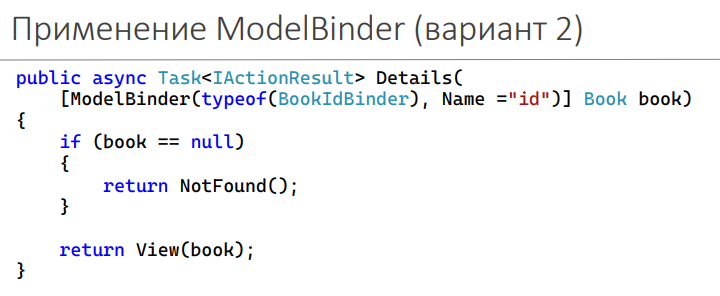
// Логика получения пользователя по ID return View();

}

5. Где механизм привязки (Model binding) ищет нужные значения?







6. Как получить данные из файла appsettings.json?

public MemoryProductService(

[FromServices]IConfiguration config,

ICategoryService categoryService)

{

//double diff = \_phones.Count/config.GetValue<int>("ItemsPerPage");

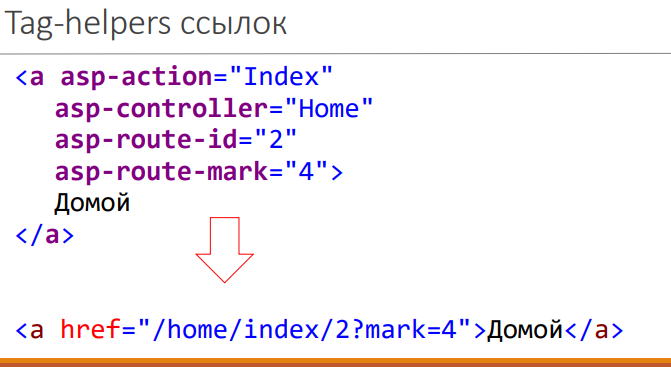
//\_amountOfPages = (int)Math.Ceiling(diff);

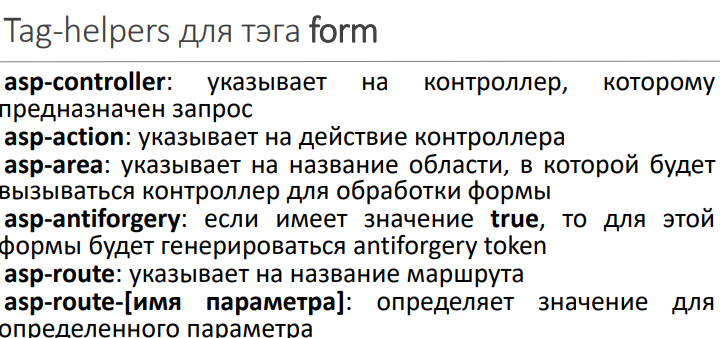
AmountOfPages = config.GetValue<int>("ItemsPerPage");

7. Как передать данные с помощью IOptions?

8. Как в коде прочитать значение, передаваемое в строке запроса?   
string? category = request.RouteValues["category"]?.ToString();

9. Как в тэге передать в запрос дополнительные данные с помощью tag-helper?





10. Что такое «Явное выражение Razor»?

Явные Razor выраженияЛюбое содержимое в скобках @() вычисляется и отображается в выходных данных. Без явного выражения <p>Age@joe.Age</p> обрабатывается как адрес электронной почты, и на выходе отображается <p>Age@joe.Age</p> . Если же текст написан как явное выражение, то вы получите <p>Age33</p>

Лаб 4:  
4. Контрольные вопросы

1. Чем контроллер API отличается от обычного контроллера?

Контроллеры API похожи на обычные контроллеры, за исключением того, что ответы, полученные их методами действия, являются объектами данных, которые отправляются клиенту без разметки HTML

1. Как осуществляется выбор метода (Action) контроллера API при обработке запроса Http?

θ POST - добавить ресурс

θ GET - для извлечения данных, но без изменения, с определенного URL-адреса

θ PUT - для сохранения или обновления ресурсов (URI)

θ DELETE - удаление указанного ресурса

θ PATCH - внести изменения в запрос

// GET: api/Dogs/5

[HttpGet("{id}")]

3. Где в запросе и в каком виде передаются данные в контроллер API?

4. Как в коде получить Scoped сервис? Используя внедрение зависимостей передать через конструктор

5. Что могут возвращать методы контроллера API?

Строку или объект C#, который потом сериализуется в формат json

6. Что такое Minimal API?

Основная идея Minimal API заключается в том, чтобы устранить некоторые из церемоний при создании простых API. Это означает определение лямбда-выражений для отдельных вызовов API. Например:

app.MapGet("/", () => "Hello World!");

7. Как зарегистрировать конечную точку для Minimal API?

Конечные точки можно указать имена для создания URL-адресов конечной точки. Использование именованной конечной точки позволяет избежать сложных путей кода в приложении:

C#Копировать

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.MapGet("/hello", () => "Hello named route")

.WithName("hi");

app.MapGet("/", (LinkGenerator linker) =>

$"The link to the hello route is {linker.GetPathByName("hi", values: null)}");

app.Run();

8. Как зарегистрировать группу конечных точек для Minimal API?

Для регистрации группы конечных точек в Minimal API в ASP.NET Core, вы можете использовать метод MapGroup. Этот метод позволяет организовать конечные точки с общим префиксом URL-адреса, что упрощает управление и структуру вашего API.

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

var apiGroup = app.MapGroup("/api");

apiGroup.MapGet("/hello", () => "Hello from /api/hello endpoint!");

apiGroup.MapPost("/create", (MyModel model) => {

// Логика для обработки POST-запроса

return Results.Ok(model);

});

app.Run();

**Лабораторная работа N5**

1. Чем сценарий Razor Pages отличается от MVC?

Razor Pages и MVC (Model-View-Controller) — это разные подходы в ASP.NET Core для создания веб-приложений:

* **Razor Pages**: организованы вокруг страниц, каждая страница может обрабатывать как GET, так и POST запросы. Это упрощает разработку, особенно для простых приложений или страниц с одной функциональностью.

Каждая страница Razor представляет собой пару файлов: • файл .cshtml, содержащий разметку HTML с кодом C # с использованием синтаксиса Razor. • файл .cshtml.cs – модель страницы - , содержащий код C #, который обрабатывает события страницы.

* **MVC**: использует контроллеры для обработки запросов, которые связываются с представлениями (Views). MVC хорошо подходит для более сложных приложений с разделением логики между контроллерами и представлениями.

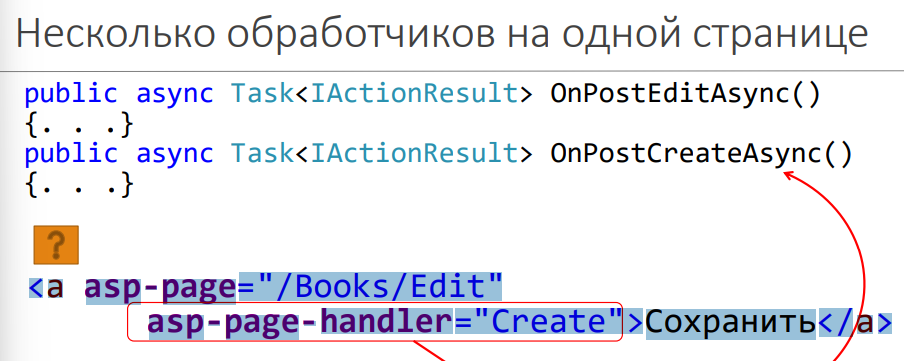
1. Что такое модель страницы Razor?

**Директива @model определяет класс модели страницы (описывается в файле .cshtml.cs )**

Модель страницы Razor (Page Model) — это класс, который связан с Razor Page (страницей Razor). Он содержит методы для обработки запросов (GET, POST и т.д.), а также свойства, которые используются для передачи данных между сервером и представлением. Модель страницы наследует от PageModel и играет ту же роль, что и контроллер в MVC.

1. Как обрабатываются запросы к страницам Razor?

Запросы к Razor Pages обрабатываются через методы, связанные с жизненным циклом запросов: OnGet(), OnPost(), и другие, такие как OnPut(), OnDelete(). Эти методы определяются в модели страницы и соответствуют HTTP-методам. Если приходит GET-запрос, вызывается OnGet(), если POST — OnPost().



1. Как в разметке страницы Razor получить доступ к свойствам модели страницы?

В разметке Razor Page можно получить доступ к свойствам модели страницы через директиву @model, которая указывает тип модели. Для доступа к свойствам просто используйте @Model.PropertyName. Например:

@page

@model MyPageModel

<p>@Model.SomeProperty</p>

1. Как привязать данные формы к модели страницы?

Для привязки данных формы к свойствам модели страницы используются атрибуты привязки, такие как [BindProperty]. Это позволяет автоматически привязывать данные формы к свойствам модели страницы при отправке формы:

public class MyPageModel : PageModel

{

[BindProperty]

public string Name { get; set; }

public void OnPost() { /\* обработка данных формы \*/ }

}

**Привязка осуществляется для всех методов, кроме Get. Если нужна привязка по Get, то дополнительно указывается: [BindProperty(SupportsGet = true)]**

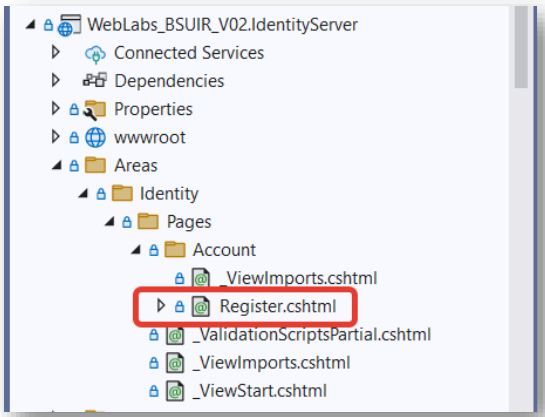
1. Для чего используется директива @page ?

Директива @page указывает, что файл .cshtml является страницей Razor. Она делает страницу доступной напрямую по URL. Без этой директивы файл не будет обрабатываться как самостоятельная страница.

**@page превращает файл в действие MVC - это означает, что он обрабатывает запросы напрямую, без прохождения через контроллер. @page должна быть первой директивой Razor на странице. @page влияет на поведение других конструкций Razor.**

1. Какой Url будет у страницы Register, которая находится в папке **Areas/Identity/Pages/Account/Register.cshtml**?

Путь будет: /Identity/Account/Register. В Razor Pages URL формируется на основе структуры каталогов, где /Areas/Identity/Pages соответствует /Identity.



1. Чем отличается адрес страницы «/index» от «./index»?

 /index: это абсолютный путь к странице index, начиная с корня сайта.

Используется для создания URL-адресов на страницу, например, Pages/Customers/Index.cshtml.

 ./index: это относительный путь к странице index относительно текущего маршрута.

Используется для создания URL-адресов на страницу, например, Pages/Index.cshtml

1. Как указать дополнительный сегмент маршрута к странице (например, id)?

Можно указать дополнительный сегмент маршрута прямо в директиве @page, добавив параметр в фигурных скобках:

@page "{id:int}"

Это позволит принимать id как часть URL, например, /MyPage/5.  
  
**Ограничение маршрутизации "{id: int}" указывает странице принимать запросы к странице, которые содержат данные о маршруте int. Если запрос к странице не содержит данных о маршруте, которые можно преобразовать в int, среда выполнения возвращает ошибку HTTP 404 (не найдена).**

1. Какой Middleware используется для передачи статического контента клиенту?

Middleware для передачи статического контента — это UseStaticFiles. Он отвечает за обработку запросов к статическим файлам, таким как изображения, CSS, JavaScript.

1. Какой интерфейс реализует объект, полученный сервером, при передаче файла от клиента на сервер?

Интерфейс IFormFile используется для представления файлов, загруженных с клиента на сервер. Он позволяет работать с переданными файлами, например, сохранять их на диск.

1. Что такое поставщики файлов (FileProvider)?

FileProvider — это интерфейс, который абстрагирует работу с файловой системой и другими источниками файлов. Он позволяет получать файлы, не полагаясь на физическую файловую систему, и использовать альтернативные источники, такие как сборки (Embedded Resources).

1. Как получить путь к папке «wwwroot»?

В ASP.NET Core путь к папке wwwroot можно получить через объект IWebHostEnvironment. Пример:

var wwwRootPath = \_hostingEnvironment.WebRootPath;

1. Форма передает файл. В каком виде этот файл будет представлен на сервере?

Файл будет представлен на сервере как объект IFormFile, который можно обработать в коде, например, сохранить на диск или прочитать его содержимое.

15) Какой атрибут нужно установить в тэге

Начало формы

, чтобы можно было передавать файлы?

Чтобы форма могла передавать файлы, нужно установить атрибут enctype="multipart/form-data":

<form method="post" enctype="multipart/form-data">  
  
  
  
  
  
 **Лабораторная работа N6:**

1. Какой механизм аутентификации имеет встроенную поддержку в ASP.Net Core?

(На основе куки)

 В ASP.NET Core встроена поддержка различных механизмов аутентификации, включая **Cookie Authentication** и **JWT Bearer Authentication**. Также можно использовать **OpenID Connect** и **OAuth2** для интеграции с внешними провайдерами аутентификации.

2. Что описывают классы ClaimsPrincipal и ClaimsIdentity?

*  **ClaimsPrincipal**: представляет собой объект, который содержит информацию о пользователе (например, его идентификационные данные, роли и другие утверждения). Он может содержать несколько ClaimsIdentity, представляющих разные источники аутентификации.

- позволяет работать с объектами claim (утверждения о пользователе, которая используется в контекста операций аутентификации и авторизации)

* **ClaimsIdentity**: описывает конкретную личность (identity) пользователя и содержит набор утверждений (claims), связанных с этой личностью. Каждый Claim представляет собой ключ-значение, которое предоставляет дополнительную информацию о пользователе.

3. Как подключить Middleware аутентификации и авторизации?

Чтобы подключить Middleware аутентификации и авторизации в ASP.NET Core, в методе Configure в файле Startup.cs нужно вызвать методы UseAuthentication() и UseAuthorization():

builder.Service.AddAuthentication();

builder.Service.AddAuthorization();

4. Приведите пример использования свойства HttpContext.User.

Можно использовать HttpContext.User для получения информации о текущем пользователе. Например:

public IActionResult Index()

{

var userName = HttpContext.User.Identity.Name; // Имя пользователя

var isAuthenticated = HttpContext.User.Identity.IsAuthenticated; // Аутентифицирован ли пользователь

return View();

}

5. Как в коде проверить, что пользователь прошел аутентификацию?

Чтобы проверить, что пользователь аутентифицирован, можно использовать свойство IsAuthenticated:

if (HttpContext.User.Identity.IsAuthenticated)

{

// Пользователь аутентифицирован

}

6. Как получить значение Claim пользователя?

Для получения значения конкретного Claim пользователя можно использовать метод FindFirst:

var id = User.FindFirst(ClaimTypes.NameIdentifier);

var emailClaim = HttpContext.User.FindFirst(ClaimTypes.Email)?.Value;

7. Как получить Id пользователя, прошедшего аутентификацию?

Идентификатор пользователя можно получить из Claims, используя FindFirst:

var id = User.FindFirst(ClaimTypes.NameIdentifier);

var userId = HttpContext.User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

8. Как разрешить доступ к контроллеру только для пользователей с ролью «manager»?

Для ограничения доступа к контроллеру можно использовать атрибут Authorize с указанием роли:

[Authorize(Roles = "manager")]

public class ManagerController : Controller

{

// Действия контроллера...

}

9. Как создать политику авторизации с помощью Claim?

Политику авторизации можно создать в методе ConfigureServices

services.AddAuthorization(options =>

{

options.AddPolicy("RequireAdministratorRole",

policy => policy.RequireClaim("Admin", "True"));

});

10. Как создать куки аутентификации с помощью объекта HttpContext?

Куки аутентификации можно создать с помощью метода SignInAsync:

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, user.UserName),

// другие claims...

};

var claimsIdentity = new ClaimsIdentity(claims, "MyCookieAuth");

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,

new ClaimsPrincipal(claimsIdentity));

11. Как добавить в проект использование системы членства Microsoft.AspnetCore.Identity?

Для добавления системы членства в проект, нужно установить пакет Microsoft.AspNetCore.Identity.EntityFrameworkCore и зарегистрировать Identity в Startup.cs:

services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<IdentityUser, IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddDefaultTokenProviders();

12. Как с помощью системы членства Microsoft.AspnetCore.Identity создать нового пользователя?

Для создания нового пользователя можно использовать UserManager:

var user = new IdentityUser { UserName = model.Email, Email = model.Email };

var result = await \_userManager.CreateAsync(user, model.Password);

13. Как с помощью системы членства Microsoft.AspnetCore.Identity осуществить вход пользователя в систему?

Для входа пользователя в систему можно использовать SignInManager:

var result = await \_signInManager.PasswordSignInAsync(user.Email, model.Password, isPersistent: false, lockoutOnFailure: false);

14. Как с помощью системы членства Microsoft.AspnetCore.Identity добавить Claim пользователю?

Для добавления Claim пользователю используется метод AddClaimAsync:

await \_userManager.AddClaimAsync(user, new Claim("YourClaimType", "YourClaimValue"));

15. Какой интерфейс используется в Microsoft.AspnetCore.Identity для доступа к хранилищу пользователей?

Интерфейс IUserStore<TUser> используется для доступа к хранилищу пользователей в ASP.NET Core Identity, где TUser — это класс пользователя, обычно IdentityUser.