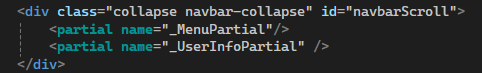
1. Для чего используется tag-helper «asp-action»?

asp-action: указывает на действие контроллера

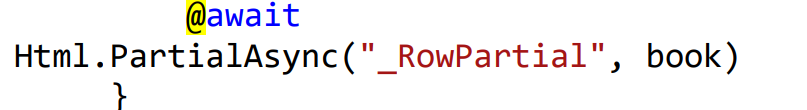
2. Что такое частичное представление?

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



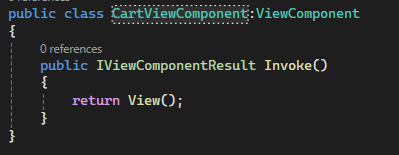


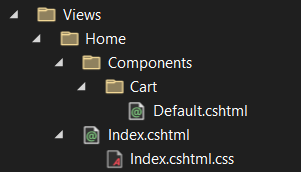
3. Как разместить частичное представление в разметке представления?



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.





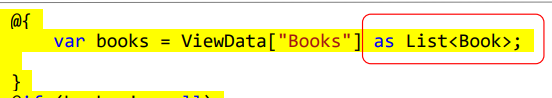


5. Как разместить компонент представления в разметке представления?

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

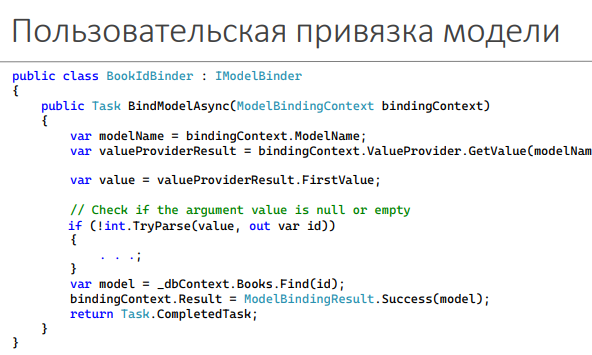
GetUser(int id)

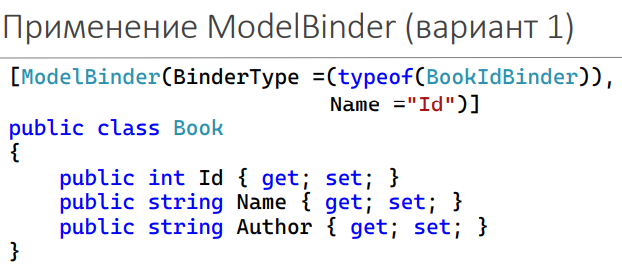
{

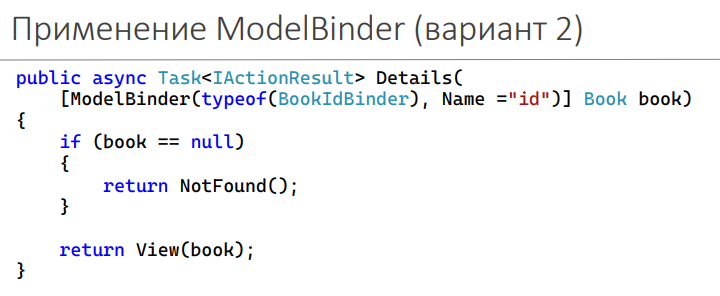
// Логика получения пользователя по ID return View();

}

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");

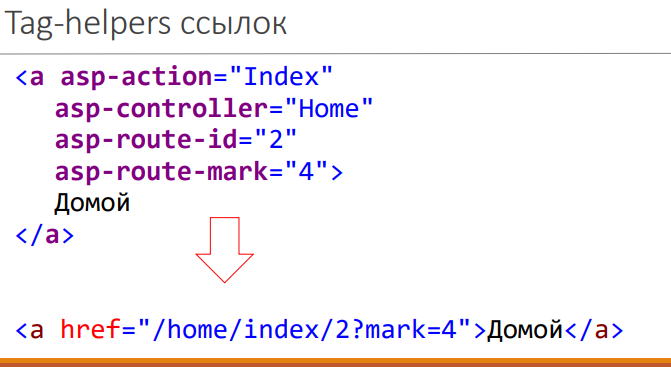
//\_amountOfPages = (int)Math.Ceiling(diff);

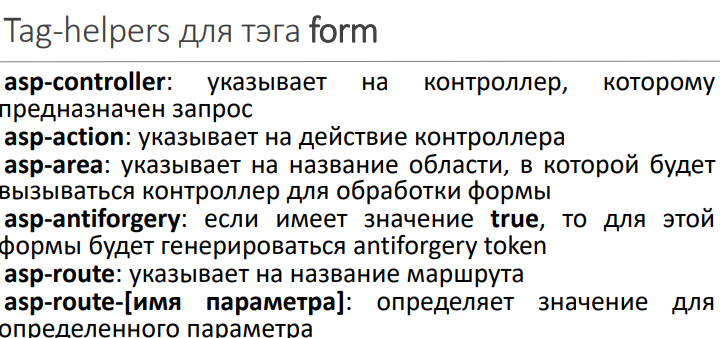
AmountOfPages = config.GetValue<int>("ItemsPerPage");

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();