

## Лабораторная работа №2

### Знакомство с приложением ASP.NET MVC (2 часа)

#### 1. Цель работы.

Знакомство со структурой приложения ASP.NET MVC Core и языком Razor. Создание простого приложения ASP.NET MVC Core. Использование макетов.

#### 2. Общие сведения.

Приложение ASP.NET MVC Core содержит следующие стандартные папки:

**/Controllers** – содержит классы контроллеров, принимающие URL-запросы;

**/Models** – содержит классы, описывающие и управляющие данными и бизнес-объектами;

**/Views** – содержит файлы шаблонов интерфейса, ответственных за создание представления, например в виде HTML

**/wwwroot** – содержит статические файлы стилей, скриптов, изображений и т.д.

**/Areas** – определяет области

Основные понятия:

**Контроллер (Controller)** отвечает за управление моделью и представлением в зависимости от запроса клиента. Имя класса контроллера **должно** заканчиваться словом Controller. Например, для контроллера **Home**, имя класса контроллера – **HomeController**.

**Представление (View)** отвечает за пользовательский интерфейс. Файлы представления имеют расширение **“.cshtml”** и содержат HTML-разметку и блоки кода на языке **Razor**. Файлы представлений располагают в папке **~/Views/<имя контроллера>** или **~/Views/Shared**

***Макет (Layout)*** – это шаблон, содержащий базовую разметку веб-страницы (скрипты, CSS, структурные элементы, например, навигацию и контейнеры содержимого) и определяющий места разметке, куда представление должно поместить содержимое веб-страницы.

Для взаимодействия с представлением макет использует вспомогательные методы:

**@RenderSection([Имя секции])**

и

**@RenderBody()**

***Вспомогательные HTML-методы*** позволяют избежать написания лишнего кода при написании разметки страницы. Вспомогательные методы имеют синтаксис @Html.ИмяМетода. Примеры вспомогательных методов: @Html.BeginForm(...), @Html.TextArea(...), @Html.RadioButton(...), @Html.ActionLink(...) и др.

**tag-helper** – выполняют те же функции, что и вспомогательные методы Html, но в более естественной для верстки форме - в виде атрибутов html.

***Для передачи данных*** представлению используются объекты ViewData, ViewBag и TempData. Они представляют собой словарные коллекции, формируемые динамически, и доступные как свойства в контроллере и в представлении

### 3. Выполнение работы

Используйте проект из предыдущей лабораторной работы.

#### 3.1. Знакомство с проектом

Найдите в проекте место размещения статических файлов (файлов css, js и др.)

Найдите в проекте папки для размещения контроллеров и представлений.

Откройте файл startup.cs. Найдите методы ConfigureServices и Configure.

Откройте файл appsettings.json. Ознакомьтесь с его содержимым.

В методе ConfigureServices найдите строки:

```
services.AddControllersWithViews();
```

```
services.AddRazorPages();
```

(если нет, то допишите)

### 3.2. Подготовка проекта

Из папки Controllers удалите файл HomeController.cs.

Из папки Views удалите папку Home.

### 3.3. Задание для выполнения

- a. Создайте контроллер с именем Home;
- b. Создайте представление для метода Index созданного контроллера, без использования страницы шаблона и выводющую статический текст “Hello World!”; (предварительно создайте папку Home в папке Views).

*c. Запустите проект и проверьте полученный результат*

- d. В папке Views/Shared найдите страницу макета (layout page) с именем \_Layout.cshtml.

В тэге <head> добавьте подключение библиотеки font-awesome:

В тэге <body> удалите всю разметку, кроме загрузки скриптов (в конце тэга).

Поместите в тэг <body> разметку тэга <body> из лабораторной работы №1, за исключением следующего: внутри тэга <main> удалите всю разметку и поместите @RenderBody()

- e. Удалите представление Index и создайте новое, с использованием созданного шаблона. В разметку страницы поместите текст “Hello, World!”.

*f. Запустите проект и проверьте полученный результат*

- g. Измените разметку главного меню на шаблоне, используя tag-helpers для формирования ссылок. Пункты меню должны ссылаться на следующие методы (или страницы):

Пункт меню	Area	Controller	Action	Page
Lab 2		Home	Index	
Каталог		Product	Index	
Администрирование	Admin			/Index

Измените разметку информации пользователя с использованием tag-helper. Ссылки оформите согласно таблице:

Ссылка	Area	Controller	Action	Page
Корзина		Cart	Index	
Мои товары		Product	UserProducts	
Log-off	Identity			/Account/Logout

- h. Запустите проект и проверьте полученный результат
- i. Откройте в папке Views представление с именем \_ViewStart.cshtml и убедитесь, что в нем указано использование макета \_Layout.cshtml для всех страниц проекта;
- j. Поместите в представление Index разметку из тэга <main> из лабораторной работы 1.
- k. Оформите список (внутри тэга <ol>) с помощью цикла “for”.
- l. Запустите проект и проверьте полученный результат
- m. С помощью ViewData передайте в представление Index текст «Лабораторная работа №2».
- n. Создайте на странице Index внутри формы список (элемент формы select). В качестве источника данных списка используйте объект SelectList.

Для формирования списка опишите класс:

```
public class ListDemo
{
    public int ListItemValue { get; set; }
```

```

        public string ListItemText { get; set; }
    }

```

Список должен отображать значение ListItemText, а передавать при отсылке формы значение ListItemValue (свойство value элемента select).

При создании элемента select используйте tag-helper “asp-items”

о. *Запустите проект и проверьте полученный результат*

#### 4. Контрольные вопросы

Где в проекте будет находиться файл представления List контроллера Product?

Как на странице шаблона (Layout Page) указать место, в котором будет помещаться разметка страницы представления?

Как в представлении указать, куда на страницу шаблона поместить разметку?

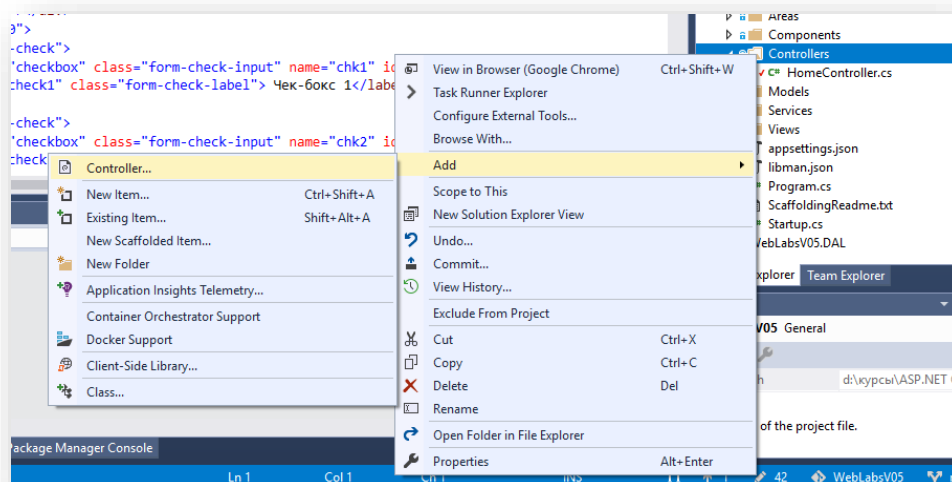
Для чего используется tag-helper «asp-action»?

#### 5. Пример выполнения работы

##### 5.1.1. Создание контроллера

Кликните правой кнопкой мыши по папке Controllers и выберите Добавить – Контроллер (Рисунок 1). Назначьте контроллеру имя Home.

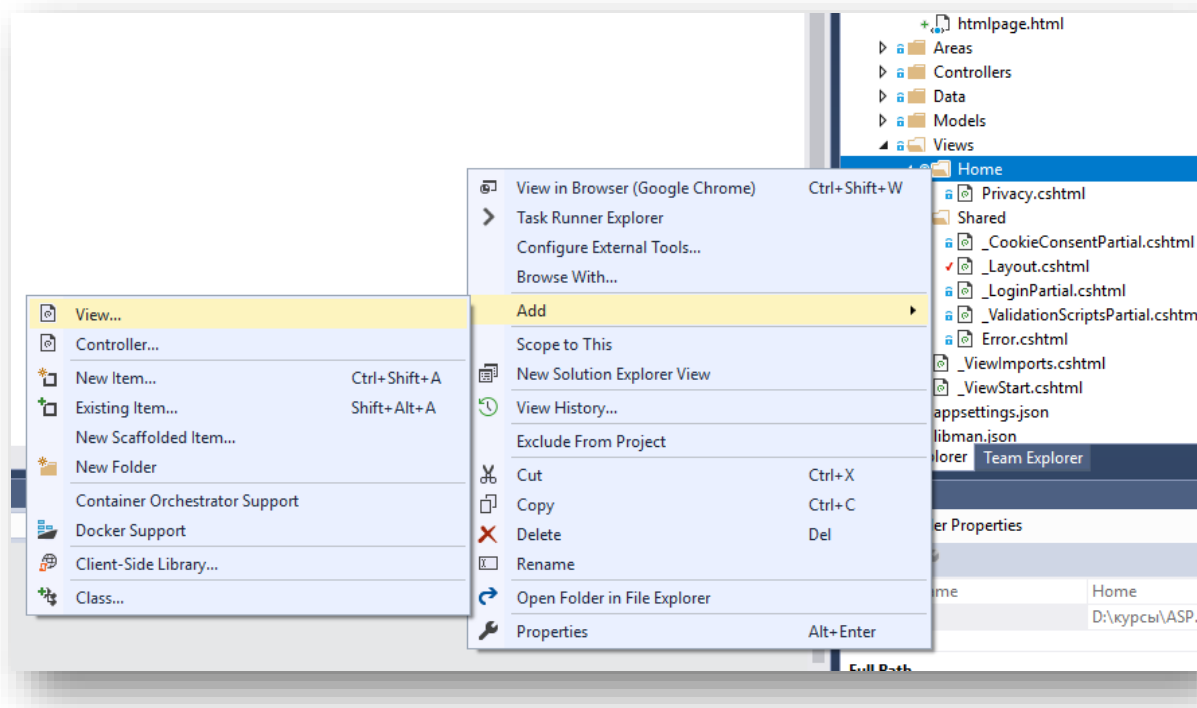
Выберите шаблон «Пустой MVC контроллер».



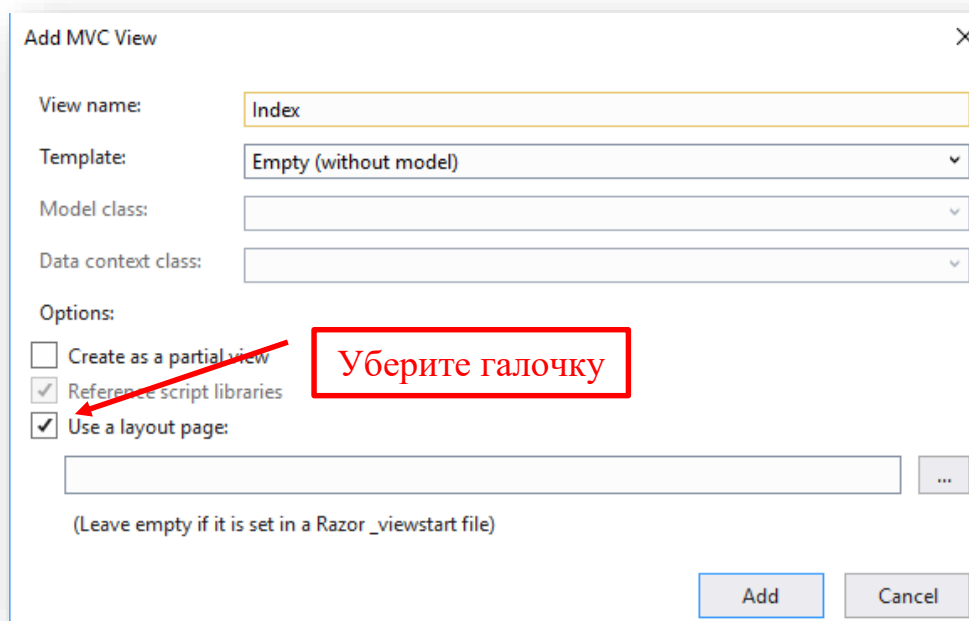
### 5.1.2. Создание представления

В папке Views создайте папку Home.

Кликните правой кнопкой мыши по папке Home и выберите «Добавить представление» (Add View) (Рисунок 2).



Уберите галочку с поля «Использовать шаблон». Создается представление с именем Index (как и название метода контроллера) и помещается в папку Views/Home (имя Home соответствует имени контроллера)



Внутри тега `<body>` выполните разметку

```
<body>
  <div>
    <h1>Hello World!</h1>
  </div>
</body>
```

Запустите проект на выполнение и посмотрите результат.

## 5.2. Работа с макетом (layout page)

Найдите в папке Views/Shared файл `_Layout.cshtml`.

В тэге `<head>` добавьте подключение библиотеки font-awesome:

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
  <title>@ViewData["Title"] - WebLabsV06</title>
  <link rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link href="~/lib/font-awesome/css/all.min.css" rel="stylesheet" />
  <link rel="stylesheet" href="~/css/site.css" />
</head>
```

В тэге `<body>` удалите всю разметку, кроме загрузки скриптов (в конце тэга).

Поместите в тэг <body> разметку тэга <body> из лабораторной работы №1, за исключением следующего: внутри тэга <main> удалите всю разметку и поместите @RenderBody();

Измените разметку главного меню на шаблоне, используя html-helpers для формирования ссылок:

```
<div class="navbar-nav">
  <a class="nav-item nav-link active" asp-controller="Home" asp-
action="Index">Lab 2</a>
  <a class="nav-item nav-link" asp-controller="Product" asp-
action="Index">Каталог</a>
  <a class="nav-item nav-link" asp-area="Admin" asp-
page="/Index">Администрирование</a>
</div>
```

**Примечание:** контроллеры Admin и Product пока не созданы, поэтому ссылки работать не будут.

Измените разметку информации пользователя с использованием tag-helper:

```
<!-- Информация пользователя -->
<a asp-controller="Cart" asp-action="Index" class="navbar-text ml-
auto">
  00,0 руб <i class="fas fa-shopping-cart nav-color"></i> (0)
</a>

<div class="dropdown ml-4 nav-color">
  <div class="dropdown-toggle" id="dropdownMenuButton"
    data-toggle="dropdown"
    aria-haspopup="true" aria-expanded="false">
    
  </div>
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <div class="dropdown-item-text">
      
      user@mail.ru
    </div>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" asp-controller="Cart" asp-
action="Index">Мои товары</a>
    <a class="dropdown-item" asp-area="Identity"
asp-page="/Account/Logout">Log off</a>
  </div>
</div>
<!-- Информация пользователя - конец -->
```



**Примечание:** контроллеры Cart и Product пока не созданы, поэтому ссылки работать не будут.

#### 5.2.1. Изменение представления Index

Удалите файл Index.cshtml и создайте новое представление Index, но установите галочку «Использовать макет» и выберите макет \_Layout.cshtml.

Результирующая разметка должна быть такой:

```
@{
    ViewBag.Title = "Главная страница";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Hello World!</h2>
```

Запустите проект и посмотрите результат.

#### 5.2.2. Привязка макета ко всем представлениям.

Найдите в папке Views представление с именем \_ViewStart.cshtml. Убедитесь, что используется макет \_Layout.cshtml. Файл должен содержать разметку:

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Удалите такую же разметку в файле Index.cshtml

Проверьте результат

#### 5.3. Знакомство с RAZOR

Скопируйте содержимое тэга <main> из Лабораторной работы 1 и поместите его на страницу Index.cshtml.

Оформите разметку списка <ol> в цикле for:

```
<ol type="A">
    @for (int i = 1; i < 5; i++)
    {
        <li>элемент @i списка</li>
    }
</ol>
```

#### 5.4. Передача данных представлению

В контроллере Nome поместите код:

```
public IActionResult Index()
{
    ViewData["Text"] = "Лабораторная работа 2";
    return View();
}
```

или

```
[ViewData]
public string Text { get; set; }

public IActionResult Index()
{
    Text = "Лабораторная работа 2";
    return View();
}
```

На странице Index поместите:

```
<h2>@ViewData["Text"]</h2>
```

Проверьте результат.

## 5.5. Формирование списка

Опишите класс

```
public class ListDemo
{
    public int ListItemValue { get; set; }
    public string ListItemText { get; set; }
}
```

(Можно описать класс в файле HomeController.cs, т.к. больше он нам не понадобится)

В контроллере Home опишите переменную, содержащую список объектов класса listDemo:

```
private List<ListDemo> _listDemo;
```

Создайте конструктор класса HomeController. В конструкторе выполните заполнение коллекции \_listDemo двумя-тремя объектами:

```
public HomeController()
{
    _listDemo = new List<ListDemo>
    {
        new ListDemo{ ListItemValue=1, ListItemText="Item 1"},
        new ListDemo{ ListItemValue=2, ListItemText="Item 2"},
        new ListDemo{ ListItemValue=3, ListItemText="Item 3"}
    };
}
```

В методе Index создайте коллекцию SelectList на основе коллекции \_listDemo. В качестве dataValueField укажите свойство ListItemValue, в качестве dataTextField укажите свойство ListItemText. Передайте полученную коллекцию в представление с помощью ViewData:

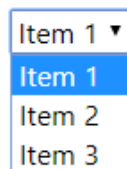
```
ViewData["Lst"] =  
    new SelectList(_listDemo, "ListItemValue", "ListItemText");
```

В представлении Index внутри формы поместите тэг <select>. С помощью tag-helper «asp-items» укажите коллекцию из ViewData (не забудьте сделать явное приведение типа данных):

```
<select asp-items="@ViewData["Lst"] as SelectList"></select>
```

Запустите проект и проверьте результат:

## Форма



Item 1 ▾  
Item 1  
Item 2  
Item 3

В браузере кликните правой клавишей мыши по созданному списку и выберите «Инспектировать элемент». Убедитесь, что значение «value» элементов (элементы «option») списка указано правильно:

