

Современные платформы прикладной разработки

ПРЕДСТАВЛЕНИЯ (ЧАСТЬ 2)

Представление ASP.NET MVC

ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ HTML
(HTML HELPERS)

HTML helpers

Вспомогательные HTML-методы позволяют избежать написания лишнего кода.

Вспомогательные HTML - методы автоматически генерируют разметку.

Пример создания HTML-формы

```
@using (Html.BeginForm("Search", "Home", FormMethod.Get))
{
    <input type="text" name="q" />
    <input type="submit" value="Search" />
}
```

Или

```
@{Html.BeginForm("Search", "Home",
    FormMethod.Get);}
<input type="text" name="q" />
<input type="submit" value="Search" />
@{Html.EndForm();}
```

Результирующая разметка

```
@using (Html.BeginForm("Search", "Home", FormMethod.Get,  
    new {  
        target = "_blank",  
        @class = "editForm",  
        data_validatable = true  
    }  
{ }
```



```
<form action="/Home/Search" class="editForm"  
    data-validatable="true" method="get"  
    target="_blank"></form>
```

Пример HTML-кодирования

```
@Html.TextArea("text", "hello <br/> world")
```



```
<textarea cols="20" id="text" name="text" rows="2">  
  hello &lt;br /&gt; world  
</textarea>
```

HTML helpers

@Html.ValidationSummary()

@Html.AntiForgeryToken()

@Html.HiddenFor(model => model.AlbumId)

@Html.LabelFor(m=>m.Color,
"Цвет", new {@class="control-label col-md-2"})

Html.TextBox

```
@Html.TextBox("Title", Model.Color.Name)
```

```
<input id="Title" name="Title" type="text" value="Aqua" />
```

```
@Html.TextBox("Color")
```

```
<input name="Color" id="Color" type="text"  
value="Color [Aqua]" >
```


HTML helpers

Представление ASP.NET MVC

СТРОГО ТИПИЗИРОВАННЫЕ
ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ HTML

Строго типизированные вспомогательные методы Html

Строго типизированные вспомогательные методы используют рефлексиию и применяются, если указана модель представления @model

Html.xxxFor

```
@Html.TextBoxFor(m=>m.Color.Name)
```

```
@Html.DropDownListFor(  
    Model => Model.Color.Name,  
    ViewBag.Colors as SelectList)
```

Класс Car

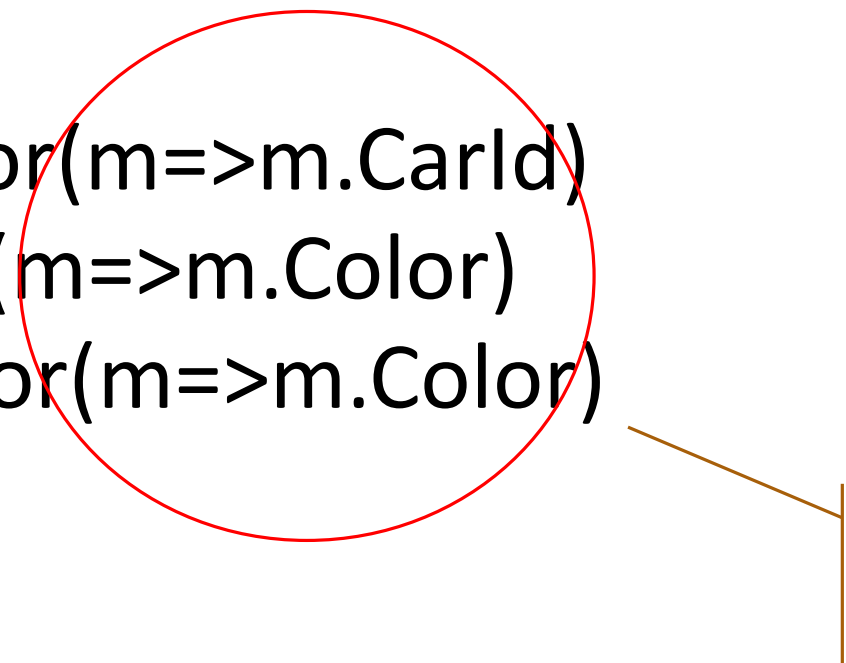
```
public class Car
{
    public int CarId { get; set; }
    public string Color { get; set; }
}
```

Строго типизированная разметка

`@model` WebApplication1.Models.Car

....

`@Html.HiddenFor(m=>m.CarId)`
`@Html.LabelFor(m=>m.Color)`
`@Html.TextBoxFor(m=>m.Color)`



Подставляется
объект Model

Результат

```
<input data-val="true"  
data-val-number="The field CarId must be a number." data-val-  
required="Требуется поле CarId."  
id="CarId" name="CarId" type="hidden" value="1" />
```

```
<label for="Color">Color</label>  
<input id="Color" name="Color" type="text" value="Aqua" />
```

Представление ASP.NET MVC

ПЕРЕДАЧА УПРАВЛЕНИЯ И ГЕНЕРИРОВАНИЕ
КОНТЕНТА

Передача управления и генерирование контента

@Html.ActionLink – генерирует гиперссылку

@Html.RouteLink – генерирует гиперссылку на основе маршрута

Html.ActionLink

```
@Html.ActionLink("Зарегистрироваться",  
    "Register",  
    "Home",  
    new { @class = "btn btn-default" })
```



```
<a class="btn btn-default" href="/Home/Register?Length=7">  
    Зарегистрироваться  
</a>
```

Html.Partial

Html.PartialAsync генерирует частичное представление в строку.

Html.RenderPartialAsync генерирует частичное представление в строку и направляет сразу в выходной поток.

```
@{Html.RenderPartial("AlbumDisplay");}  
@Html.Partial("AlbumDisplay")
```

Представление ASP.NET MVC

ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ URL (URL HELPERS)

URL.Action и @URL.RouteUrl

@URL.Action и @URL.RouteUrl – работают аналогично методам Html.ActionLink и Html.RouteLink , но вместо гиперссылки (тэг <a>) полученный адрес выводят в виде текста.

@Url.Action("Browse", "Store", new { genre = "Jazz" }, null)



/Store/Browse?genre=Jazz

Tag-helpers

Tag Helpers

Tag helpers – новая возможность, появившаяся в ASP.NET Core, - представляют собой классы C#, которые манипулируют элементами HTML, добавляя или изменяя контент элемента.

Tag Helpers

Tag helpers пришли на смену HTML-helpers, используемых в предыдущих версиях ASP.NET

Tag Helpers

Для общих целей существует много встроенных Tag Helpers - например, для создания форм, ссылок, загрузки и тд - и многие из них доступны в открытых GitHub репозиториях и в качестве NuGet пакетов.

Tag Helpers работают с HTML элементами, основываясь на имени элемента, имени атрибута или родительском тэге.

Встроенные Tag-helpers

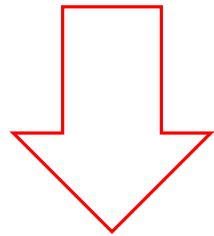
Tag-helpers ссылок

ВСТРОЕННЫЕ TAG-HELPERS

Tag-helpers ссылок

```
<a asp-action="Index"  
    asp-controller="Home"  
    asp-route-id="2"  
    asp-route-mark="4">  
    Домой
```

```
</a>
```



```
<a href="/home/index/2?mark=4">Домой</a>
```

Tag-helpers ссылок

`asp-area` : определяет зону в проекте

`asp-protocol` : определяет протокол передачи данных
например `asp-protocol="https"`

Tag-helpers для формы

ВСТРОЕННЫЕ TAG-HELPERS

Tag-helpers для тэга **form**

asp-controller: указывает на контроллер, которому предназначен запрос

asp-action: указывает на действие контроллера

asp-area: указывает на название области, в которой будет вызываться контроллер для обработки формы

asp-antiforgery: если имеет значение **true**, то для этой формы будет генерироваться antiforgery token

asp-route: указывает на название маршрута

asp-route-[имя параметра]: определяет значение для определенного параметра

```
<form asp-controller="Home"  
      asp-action="Demo"  
      asp-antiforgery="true"  
      asp-route-id="2">
```

• • •

```
</form>
```


Tag-helpers для тэга `input`

asp-for: указывает, для какого свойства модели создается элемент

asp-format: устанавливает формат ввода для элемента

```
<input class="form-control"
      asp-for="Expired"
      asp-format="{0:dd-MM-yyyy}" />
```

Tag-helpers для тэга `label`

asp-for: указывает, для какого свойства модели создается элемент

```
<label asp-for="Description"></label>
```

Tag-helpers для тэга `select`

asp-for: указывает, для какого свойства модели создается элемент

asp-items: указывает на объект коллекции `IEnumerable<SelectListItem>`, используемый для заполнения списка

Tag-helpers для тэга `select`

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;
namespace FormsTagHelper.ViewModels
{
    public class CountryViewModel
    {
        public string Country { get; set; }
        public List<SelectListItem> Countries { get; } =
            new List<SelectListItem> {
                new SelectListItem { Value = "MX", Text = "Mexico" },
                new SelectListItem { Value = "CA", Text = "Canada" },
                new SelectListItem { Value = "US", Text = "USA" }, };
    }
}
```

Tag-helpers для тэга select

```
<form asp-controller="Home" asp-action="Index" method="post">
  <select asp-for="Country"
    asp-items="ViewBag.Countries">
    <option disabled="disabled"
      selected="selected"
      value="">Выберите страну</option>
  </select>

  <br /><button type="submit">Register</button>
</form>
```

Использование перечислений для заполнения СПИСКОВ

```
public enum Directions
{
    [Display(Name = "Вверх")]
    Up,
    [Display(Name = "Вниз")]
    Down,
    [Display(Name = "Влево")]
    Left,
    [Display(Name = "Вправо")]
    Right
}
```

Использование перечислений для заполнения списков

```
public class ViewModel
{
    public Directions Direction { get; set; }
}
```

Использование перечислений для заполнения СПИСКОВ

```
@model ViewModel
```

```
<form asp-controller="Home" asp-action="Index" method="post">  
    <select asp-for="Direction"  
        asp-items="@Html.GetEnumSelectList<Directions>()">  
        <option disabled="disabled"  
            selected="selected" value="">  
            Выберите направление</option>  
    </select>  
    <br /><button type="submit">OK</button>  
</form>
```


Валидационные Tag-helpers

asp-validation-for : выводит сообщения валидации для указанного свойства модели

asp-validation-summary : выводит все сообщения валидации

Валидационные Tag-helpers

```
<div asp-validation-summary="All"
      class="text-danger"></div>
```

. . .

```
<div class="form-group">
  <label asp-for="Name"></label>
  <div><span asp-validation-for="Name"
            class="text-danger"></span></div>
  <input asp-for="Name" class="form-control" />
</div>
```

Значения атрибута `asp-validation-summary`

None: ошибки валидации не отображаются

ModelOnly: отображаются только ошибка валидации уровня модели, ошибки валидации для отдельных свойств не отображаются

All: отображаются все ошибки валидации

Другие Tag-Helpers

Cache

Вспомогательная функция тегов кэша позволяет повысить производительность приложения ASP.NET Core за счет кэширования его содержимого во внутренний поставщик кэша ASP.NET Core.

Cache

```
<cache>@DateTime.Now</cache>
```

Cache

Первый запрос к странице, содержащей вспомогательную функцию тегов, отобразит текущую дату. Последующие запросы будут показывать кэшированное значение, пока срок действия кэша не истечет (по умолчанию — 20 минут) или пока кэшированная дата не будет удалена из кэша.

Environment

```
<environment names="Staging,Production">  
    <strong>IWebHostEnvironment.EnvironmentName  
is Staging or Production</strong>  
</environment>
```


Environment

```
<environment names="Staging,Production">  
    <strong>IWebHostEnvironment.EnvironmentName  
is Staging or Production</strong>  
</environment>
```

Tag-helpers для CSS и JavaScript

ВСТРОЕННЫЕ TAG-HELPERS

Tag-helpers загрузки скриптов

asp-append-version: если имеет значение true, то к пути к файлу скрипта добавляется номер версии

asp-fallback-src: указывает вспомогательный путь к скрипту, который используется, если загрузка скрипта, указанного в атрибуте src пройдет неудачно

asp-fallback-test: определяет выражение, которое тестирует загрузку основного скрипта из атрибута src

asp-src-include: определяет шаблон подключаемых файлов, через запятую можно задать несколько шаблонов

Tag-helpers загрузки скриптов

asp-src-exclude: определяет через запятую набор шаблонов для тех файлов, которые следует исключить из загрузки

asp-fallback-src-include: определяет через запятую набор шаблонов файлов, которые подключаются в том случае, если загрузка основного скрипта из атрибута src прошла неудачно

asp-fallback-src-exclude: определяет через запятую набор шаблонов файлов, которые следует исключить из загрузки в том случае, если загрузка основного скрипта из атрибута src прошла неудачно

Tag-helpers загрузки скриптов

```
<script  
src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js"  
asp-fallback-src="~/lib/jquery/dist/jquery.min.js"  
asp-fallback-test="window.jQuery"  
crossorigin="anonymous"  
integrity="sha384-  
K+ctZQ+LL8q6tP7I94W+qzQsfRV2a+AfHIi9k8z8l9ggpc8X+Ytst4yBo/hH+8Fk">  
</script>
```

Шаблоны пути к файлу

? : шаблон соответствует любому одному символу, кроме **/**.

Пример:

шаблон **js/src?.js** соответствует любому файлу в каталоге **js**, с именем **src**, после которого может быть любой символ, заканчивающемся на **.js**.

Пути **js/src1.js** и **js/srcX.js** подходят под шаблон.

Пути **js/src123.js** или **js/mydir/src1.js** не подходят.

Шаблоны пути к файлу

***** : данный шаблон соответствует любому количеству символов, кроме /.

Пример

шаблон **js/*.js** соответствует любому файлу в каталоге **js**, имеющий расширение **.js**

Пути **js/src1.js** и **js/src123.js** подходят под шаблон.

Путь **js/mydir/src1.js** не подходит под шаблон.

Шаблоны пути к файлу

****** : данный шаблон соответствует любому количеству символов, включая /.

Пример:

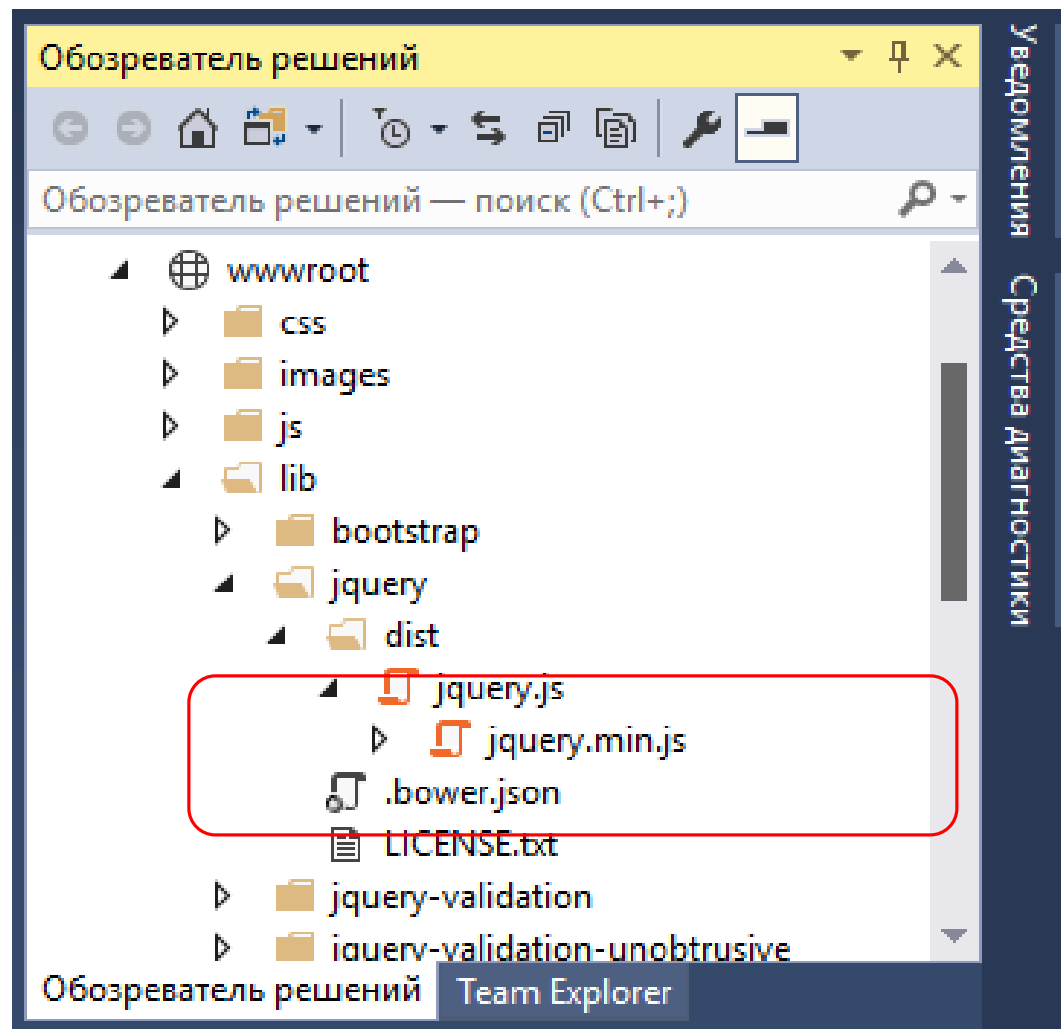
шаблон **js/**/*.**js соответствует любому файлу с расширением **.js** внутри каталога **js**, включая **все подкаталоги**.

Пути **/js/src1.js** and **/js/mydir/src1.js** подходят под шаблон.

Пример

```
<script  
    asp-src-  
include="~/lib/jquery/dist/**/*.*.js">  
</script>
```

```
<head>
  <meta charset="utf-8" />
  <title></title>
  <script
src="/lib/jquery/dist/jquery.js"></script>
  <script
src="/lib/jquery/dist/jquery.min.js">
    </script>
</head>
```



Исключение лишних файлов

```
<script  
include=~ /lib/jquery/dist/**/* .js"  
    asp-src-exclude="**/* .min.js">  
</script>
```

asp-src-

Сужение шаблона поиска

```
<script  
    asp-src-  
include=~ /lib/jquery/dist/**/jquery.js">  
</script>
```

Использование Hosting Environment

```
<environment names="Development">
  <script src="~/lib/jquery/dist/jquery.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>
</environment>
<environment names="Staging,Production">
  <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js"
    asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
    asp-fallback-test="window.jQuery"
    crossorigin="anonymous"
    integrity="sha384-
K+ctZQ+LL8q6tP7I94W+qzQsfRV2a+AfHII9k8z8l9ggpc8X+Ytst4yBo/hH+8Fk">
    </script>
</environment>
```

Tag-helpers для файлов CSS

`<link`

```
    asp-href-include="/lib/bootstrap/dist/**/*.min.css"  
    rel="stylesheet" />
```

Tag-helpers для файлов CSS

```
<link
href="https://maxcdn.bootstrapcdn.com/bootstra
p/3.3.6/css/bootstrap.min.css"
asp-fallback-href-include=

"/lib/bootstrap/dist/**/*.min.css"
asp-fallback-test-class="btn"
asp-fallback-test-property="display"
asp-fallback-test-value="inline-block"
rel="stylesheet" />
```

Использование CDN

CDN (Content Delivery Network) — это географически распределённая сетевая инфраструктура, обеспечивающая быструю доставку контента пользователям веб-сервисов и сайтов.

Пример CDN для Bootstrap:

<https://getbootstrap.com/docs/5.3/getting-started/download/#cdn-via-jsdelivr>

Подключение Tag-helpers

TAG-HELPERS

Подключение Tag-helpers

Подключение tag-helpers осуществляется в файле
_ViewImports.cshtml

```
@addTagHelper "*", Microsoft.AspNetCore.Mvc.TagHelpers"
```

```
@addTagHelper "*", TagHelperSamples.Bootstrap"
```

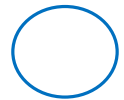
```
@addTagHelper TagHelperSamples.Web.TagHelpers.DemoTagHelper,  
TagHelperSamples.Web
```

Подключение Tag-helpers

@addTagHelper "*,Microsoft.AspNetCore.Mvc.TagHelpers"

@addTagHelper TagHelperSamples.Web.TagHelpers.DemoTagHelper,

TagHelperSamples.Web



Имя tag-хелпера



Библиотека, в которой описан tag-хелпер

Создание tag-helpers

Создание tag-helpers

Класс Tag-хелпера наследуется от базового класса `TagHelper`, определенного в пространстве имен `Microsoft.AspNetCore.Razor.TagHelpers`, и переопределяет метод

```
async Task ProcessAsync(TagHelperContext context,  
                        TagHelperOutput output)
```

или

```
void Process(TagHelperContext context,  
            TagHelperOutput output)
```

Создание tag-helpers

```
using Microsoft.AspNetCore.Razor.TagHelpers;

namespace TagHelperSamples.Web.TagHelpers
{
    public class DemoTagHelper : TagHelper
    {
        public override void Process(TagHelperContext context,
                                     TagHelperOutput output)
        {
            . . .
        }
    }
}
```


TagName, Attributes и Content

Пример: разметка

```
<email mail-to="gogogo"></email>
```

должна преобразоваться в:

```
<a href="mailto:gogogo@mail.ru">gogogo@mail.ru</a>
```

<https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/tag-helpers/authoring?view=aspnetcore-7.0>

Создание tag-helpers

```
public class EmailTagHelper : TagHelper
{
    private string domain = "@mail.ru";
    // ссылка на атрибут mail-to
    public string MailTo { get; set; }

    public override void Process(TagHelperContext context,
TagHelperOutput output)
    {
        output.TagName = "a";
        var address = MailTo + domain;
        output.Attributes.Add("href", "mailto:" + address);
        output.Content.SetContent(address);
    }
}
```

RemoveAll, PreContent.SetHtmlContent и PostContent.SetHtmlContent

Разметка

`<p bold>Жирный текст</p>`

Должна преобразоваться в:

```
<strong>  
    <p>Жирный текст</p>  
</strong>
```

```
[HtmlTargetElement(Attributes = "bold")]
public class BoldTagHelper:TagHelper
{
    public override void Process(TagHelperContext context,
                                TagHelperOutput output)
    {
        output.Attributes.RemoveAll("bold");
        output.PreElement.SetHtmlContent("<strong>");
        output.PostElement.SetHtmlContent("</strong>");
    }
}
```

Передача модели в tag-helper

СОЗДАНИЕ TAG-HELPERS

Передача модели в tag-helper

```
public class User
{
    public int UserId { get; set; }
    public string UserName { get; set; }
}
```

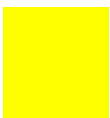
```
public class UserTagHelper:TagHelper
{
    public User props { get; set; }
    public override void Process(TagHelperContext context,
                                TagHelperOutput output)
    {
        . . .
    }
}
```

```
output.TagName = "table";
    TagBuilder div = new TagBuilder("div");
    foreach(var p in typeof(User).GetProperties())
    {
        TagBuilder tr = new TagBuilder("tr");
        tr.InnerHtml.AppendHtml(
            $"<td>{p.Name}</td><td>{p.GetValue(props)}</td>");
        div.InnerHtml.AppendHtml(tr);
    }
output.Attributes.Add("class", "table table-condensed");
output.TagMode = TagMode.StartTagAndEndTag;
output.Content.SetHtmlContent(div);
```


Разметка представления

```
<user props="@{new User { UserId = 1,  
                                UserName = "Goga" };  
}"
```

```
</>
```



Результирующая разметка

```
<table class="table table-condensed">
  <tbody>
    <tr><td>UserId</td><td>1</td></tr>
    <tr><td>UserName</td><td>Goga</td></tr>
  </tbody>
</table>
```

Генерирование адресов в классе tag-helper

Генерирование адресов в классе tag-helper

Задача: внутри tag-helper сформировать ссылку:

```
<a href="/home/index">Домой</a>
```

```
var path = urlHelper.Action("index", "home");
```

```
var link = @$@"<a href='{path}'></a>";
```

Вариант 1

```
public class DemoTagHelper : TagHelper
{
    public IUrlHelper UrlHelper { get; set; }

    public override void Process(TagHelperContext context,
                                TagHelperOutput output)
    {
        . . .
    }
}
```

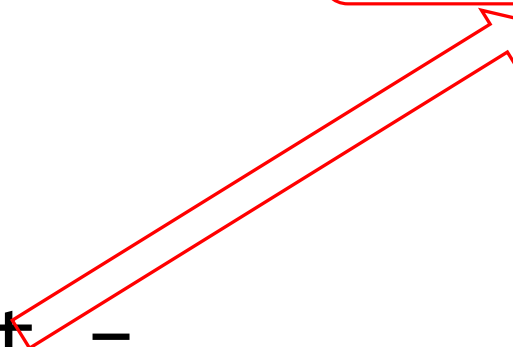
Вариант 1

`<demo url-helper="@Ur1"></demo>`

Вариант 2

```
IUrlHelper urlHelper =  
    urlHelperFactory.GetUrlHelper(actionContext);
```

```
ActionContext actionContext =  
    actionContextAccessor.ActionContext;
```



Вариант 2

В классе Program.cs

```
builder.services  
    .AddSingleton<IActionContextAccessor,  
                ActionContextAccessor>();
```

Вариант 2

```
public class DemoTagHelper : TagHelper
{
    private IUrlHelperFactory urlHelperFactory;
    private IActionContextAccessor actionContextAccessor;

    public DemoTagHelper(IUrlHelperFactory factory,
                        IActionContextAccessor accessor)
    {
        urlHelperFactory = factory;
        actionContextAccessor = accessor;
    }
}
```

Вариант 2

```
public override void Process(TagHelperContext context,  
                             TagHelperOutput output)  
{  
    var urlHelper = urlHelperFactory  
        .GetUrlHelper(actionContextAccessor.ActionContext);  
    . . .  
}
```

Вариант 3

```
public class DemoTagHelper : TagHelper
{
    private IUrlHelperFactory urlHelperFactory;

    [ViewContext]
    [HtmlAttributeNotBound]
    public ViewContext ViewContext { get; set; }

    public DemoTagHelper(IUrlHelperFactory factory)
    {
        urlHelperFactory = factory;
    }
}
```

Вариант 3

```
public override void Process(TagHelperContext
context, TagHelperOutput output)
{
    var urlHelper = urlHelperFactory
                    .GetUrlHelper(ViewContext);

    . . .
}
```

Вариант 4 (начиная с версии 2.2)

В ASP.Net Core, начиная с версии 2.2, имеется класс **LinkGenerator**, который по умолчанию зарегистрирован в качестве сервиса, а значит, его можно «внедрить» в конструктор класса Tag-helper

Класс LinkGenerator

Класс **LinkGenerator** предоставляет набор функций для генерирования url-адресов, аналогично интерфейсу **IUrlHelper**

```
var path = _lg.GetPathByAction("index", "Home");
```

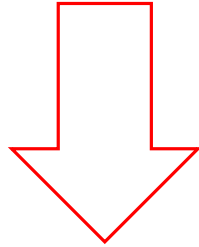
Пример

Создать Tag-Helper для тэга `` для формирования атрибута `src` с помощью атрибутов `img-controller` и `img-action`

```
[HtmlTargetElement(tag:"img",      Attributes      ="img-action,      img-  
controller")]  
    public class ImageTagHelper : TagHelper  
    {  
        public string ImgAction { get; set; }  
        public string ImgController { get; set; }  
        LinkGenerator _linkGenerator;  
  
        public ImageTagHelper(LinkGenerator linkGenerator)  
        {  
            _linkGenerator = linkGenerator;  
        }  
    }
```

```
public override void Process(TagHelperContext context,  
                             TagHelperOutput output)  
{  
    var uri = _linkGenerator  
                .GetPathByAction(ImgAction, ImgController);  
    output.Attributes.Add("src", uri);  
}
```

```
<img asp-controller="Home" asp-action="GetImage"  
alt="Alternate Texts" class="thumbnail" />
```



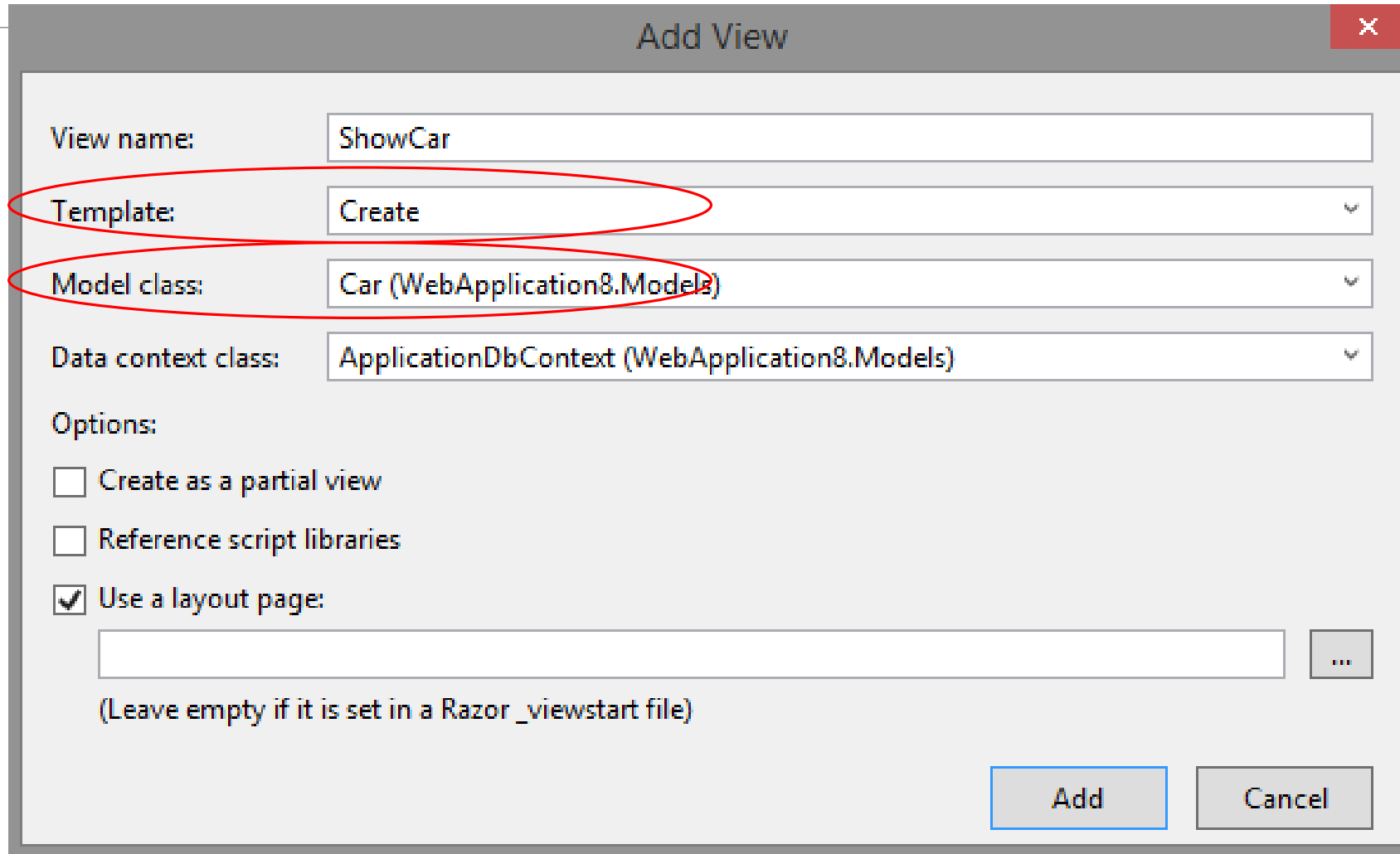
```
<img src= "Home/GetImage" alt="Alternate Texts"  
class="thumbnail" />
```

Представление ASP.NET MVC (ч.2)

SCAFFOLDING

Scaffolding — автоматическая генерация
представления на основании модели

Scaffolding



The image shows a screenshot of the 'Add View' dialog box in a web development application. The dialog has a title bar with the text 'Add View' and a close button (X) in the top right corner. The main area contains several input fields and checkboxes. The 'View name' field is set to 'ShowCar'. The 'Template' dropdown is set to 'Create'. The 'Model class' dropdown is set to 'Car (WebApplication8.Models)'. The 'Data context class' dropdown is set to 'ApplicationDbContext (WebApplication8.Models)'. Under the 'Options' section, there are three checkboxes: 'Create as a partial view' (unchecked), 'Reference script libraries' (unchecked), and 'Use a layout page:' (checked). Below the 'Use a layout page:' checkbox is an empty text box and a button with three dots (...). At the bottom of the dialog are two buttons: 'Add' and 'Cancel'. Red circles are drawn around the 'Template' and 'Model class' fields.

View name: ShowCar

Template: Create

Model class: Car (WebApplication8.Models)

Data context class: ApplicationDbContext (WebApplication8.Models)

Options:

- ☐ Create as a partial view
- ☐ Reference script libraries
- ☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

Scaffolding

<https://learn.microsoft.com/ru-ru/aspnet/core/fundamentals/tools/dotnet-aspnet-codegenerator?view=aspnetcore-7.0>