Модель

public class Product

{

public int ProductId { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public string Category { get; set; }

}

Контроллер

public class ProductController : Controller

{

//

// GET: /Product/

public ActionResult Index()

{

List<Product> products = new List<Product>();

products.Add(new Product()

{

ProductId = 1,

Name = "Шариковая ручка",

Description = "Синяя шариковая ручка с колпачком и прозрачным корпусом.",

Price = 3m,

Category = "Канцтовары"

});

products.Add(new Product()

{

ProductId = 2,

Name = "Бумага A4",

Description = "Стандартная бумага для цветной и чёрно-белой печати.",

Price = 15m,

Category = "Канцтовары"

});

products.Add(new Product()

{

ProductId = 2,

Name = "Мобильный телефон",

Description = "Мобильный телефон с фотокамерой.",

Price = 250m,

Category = "Техника"

});

// Возвращаем представление из директории Views/Products/Index.cshtml

// Параметр передающийся в метод View() является моделью, которая будет доступна только на чтение в представлении Index

return View(products);

}

}

Представление

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

@foreach (var item in Model)

{

@\*Чтение свойства элемента модели\*@

<h3>Имя продукта: @item.Name</h3>

if (item.Category == "Канцтовары")

{

@\*Razor понимает, что оператор в теле if начинается с HTML дескриптора,

и поэтому относится к нему как к разметке, которую нужно генерировать\*@

<p>На канцтовары действует <b>скидка 10%</b></p>

}

if (item.Price < 5)

{

@\* Префикс строки @: указывает что строку нужно обработать так, как если она будет HTML разметкой\*@

@:стоимость доставки 10 грн.

}

<hr />

}

</body>

</html>

# RouteConfig

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

// Регистрация маршрута {controller}/{action}. После запуска приложения сделайте запрос по адресу /Home/Index или /Home/List

routes.MapRoute("Default", "{controller}/{action}");

}

}

\*\*\*

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default", // имя маршрута

url: "{controller}/{action}", // переменные сегментов

defaults: new { controller = "Home", action = "Index" } // значения по умолчанию

);

}

\*\*\*

# Href

<body>

<div>

Маршрутизация в MVC.<br />

Определение значений по умолчанию.

<br />

<a href="/Customer">/Customer</a><br />

<a href="/Customer/Index">/Customer/Index</a><br />

<a href="/Customer/List">/Customer/List</a><br />

<a href="/Customer/List/All">/Customer/List/All</a>

</div>

</body>

# StaticSegments

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}",

defaults: new { controller = "Home", action = "Index" }

);

routes.MapRoute(

name: "PublicRoute",

url: "public/{controller}/{action}",

defaults: new { controller = "Home", action = "Index" }

);

}

# RouteData and ViewBag

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = "DefaultId" }

);

// !!!!!!

// Некторые имена зарезервированы и не могут использоваться как имена для переменных сегментов.

// К таким именам относятся controller, action и area

}

}

Controller

public ActionResult Index()

{

// RouteData.Values - коллекция значений сегментов, доступных в момент обработки текущего запроса

// ViewBag - динамический словарь, который позволяет передавать данные в представление

ViewBag.Variable = RouteData.Values["id"];

return View();

}

//

// GET: /CustomVariable/

// Отображение переменной id на параметр метода доступа.

// MVC Framework будет пытаться преобразовать значение, полученное из URL, в тип указанный в параметрах.

public ActionResult CustomVariable(int? id)

{

ViewBag.Variable = id;

// Метод действия CustomVariable по умолчанию должен возвращать представление CustomVariable.cshtml из директории Views/Home

// Строковое значение, переданное в качестве параметра метода View, позволяет задать другое представление, которое вернется в ответ на запрос

return View("Index");

}

# Router Segments Variable

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{x}/{y}",

defaults: new { controller = "Home", action = "Index", x = 0, y = 0 }

);

}

Controller

// Значения сегментов с именами x и y автоматически будут присвоены параметрам метода Index с теми же именами.

// В случае, если значения, которые были получены из сегментов, не могут быть приведены к типу int, параметрам будет присвоено значение null, так как тип параметров int?,

// в случае использования типа int произойдёт ошибка этапа выполнения.

public ActionResult Index(int? x, int? y)

{

if (x != null && y != null)

{

ViewBag.Result = x + y;

}

return View();

}

View

<div>

Result = @ViewBag.Result<br />

<a href="/Home/Index/2/3">2 + 3 (/Home/Index/2/3)</a><br />

<a href="/Home/Index/10/20">10 + 20 (/Home/Index/10/20)</a>

</div>

# UrlParametr

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional } // переменная сегмента со значением UrlParameter.Optional является необязательной

);

# CatchAll

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}/{\*catchall}",

// \* перед именем переменной сегмента указывает на то что данная переменная получит значения всех оставшихся сегментов которые есть в URL

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

public ActionResult Index()

{

ViewBag.Id = RouteData.Values["Id"];

ViewBag.All = RouteData.Values["catchall"];

return View();

}

<body>

<div>

id = @ViewBag.Id

<br />

catchall = @ViewBag.All

<br />

<br />

<br />

<a href="/Home/Index/10/hello/world">/Home/Index/10/hello/world</a>

</div>

</body>

# Router Constraint

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

// Маршрут настроен таким образом, что для совпадения с входящим запросам, значения переменных сегментов

// должны подходить под шаблон, который задан через регулярное выражение.

// Данный маршрут требует, что бы переменная {controller} имела значение, которое начинается с символа H,

// {action} должна иметь значение либо Index либо About, переменная {id} должна быть числовым значением.

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },

constraints: new { controller = "^H.\*", action = "^Index$|^About$", id = "^\\d\*$" }

);

}

# Ignore Router

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

// Маршрут настроен таким образом, что для совпадения с входящим запросам, значения переменных сегментов

// должны подходить под шаблон, который задан через регулярное выражение.

// Данный маршрут требует, что бы переменная {controller} имела значение, которое начинается с символа H,

// {action} должна иметь значение либо Index либо About, переменная {id} должна быть числовым значением.

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

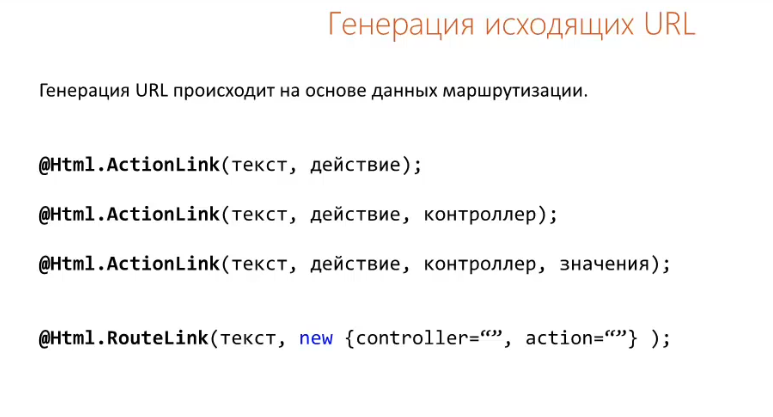
defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },

constraints: new { controller = "^H.\*", action = "^Index$|^About$", id = "^\\d\*$" }

);

}

# HTML.LINKS



<body>

<div>

<h2>Представление Index контроллера Home</h2>

@\*Генерация ссылки на действие в другом контроллере.

Результат работы метода ActionLink зависит от текущей схемы маршрутизации

результат: <a href="/Customer/Index">Работа с клиентами</a>\*@

@Html.ActionLink("Работа с клиентами", "Index", "Customer")

<br />

@Html.RouteLink("Работа с клиентами", new { controller = "Customer", action = "Index" })

<br />

@\*Передача дополнительных параметров действию\*@

@Html.ActionLink("Создать запись о клиенте", "Create", "Customer", new { id = "Ivanov-Ivan" }, null)

<br />

@\*Добавление HTML атрибутов\*@

@Html.ActionLink("Создать запись о клиенте", "Create", "Customer", new { id = "Ivanov-Ivan" }, new { style = "font-size:25px;" })

<br />

@\*Генерация URL без окружающей HTML разметки\*@

Адрес действия "Создать запись о клиенте" - @Url.Action("Create", "Cusomer", new { id = "customer-name" })

</div>

# Areas

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },

namespaces: new[] { "Areas.Controllers" } // разрешение конфликта пространств имен.

// в данном приложении находится несколько классов HomeController - один в основной области,

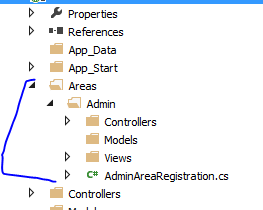
// второй в области Admin.

// массив, заданный в качестве последнего параметра, позволяет определить приоритеты

// пространств имен, в которых нужно искать контроллер.

);

}



public class AdminAreaRegistration : AreaRegistration

{

public override string AreaName

{

get

{

return "Admin";

}

}

public override void RegisterArea(AreaRegistrationContext context)

{

context.MapRoute(

"Admin\_default",

"Admin/{controller}/{action}/{id}",

new { action = "Index", id = UrlParameter.Optional }

);

}

}

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

<h2>Представление Index основная область</h2>

@\*Создание ссылки через метод RouteLink на действие в другой области\*@

@Html.RouteLink("Admin area", new { controller = "Home", action = "Index", area = "Admin" })

</div>

</body>

</html>

# Razor Syntax

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Razor Syntax</title>

</head>

<body>

<!--Блок кода---------------------------------------->

@{

int a = 123;

string message = "<b>Hello world</b>";

}

<!--------------------------------------------------->

<!--Выражение (HTML encoded)------------------------->

<span>@message</span>

<!--------------------------------------------------->

<!--Выражение (Unencoded)--------------------------->

<span>@Html.Raw(message)</span>

<!--------------------------------------------------->

<!--Комбинирование текста и разметки----------------->

@{

string[] items = { "Item 1", "Item 2", "Item 3" };

}

@{

foreach (string item in items)

{

<span>@item</span>

}

}

<!--------------------------------------------------->

<!--Комбинирование кода и текста--------------------->

@{

bool foo = true;

}

@if (foo)

{

<span>Hello world</span>

}

<!--------------------------------------------------->

<!--Комбинирование кода и текста (Вариант 2)--------->

@if (foo)

{

@:Hello world

}

<!-------------------------------------------------->

<!--Email адреса в разметке-------------------------->

admin@example.com

<!--------------------------------------------------->

<!--Использование символа at------------------------->

<p>Для получения доступа к значению переменной foo мы используем код @@foo</p>

<!--------------------------------------------------->

@\*

Серверный

многострочный

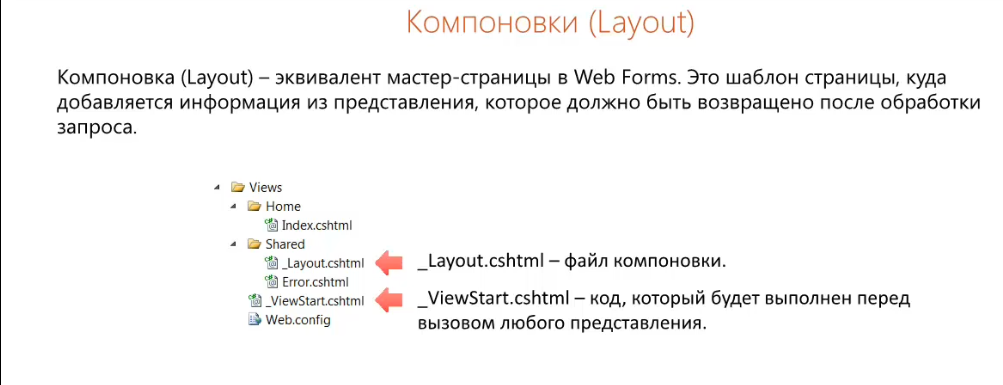
комментарий

\*@

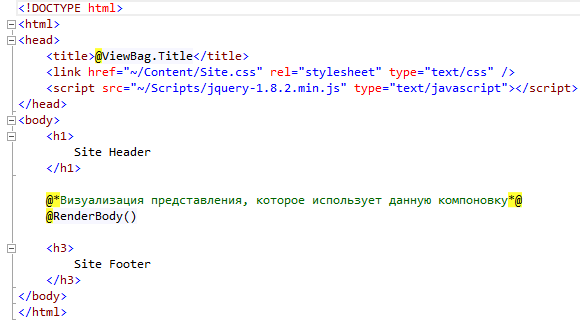
</body>

</html>

# Компоновка



## Layout



@{

@\*Определение файла компоновки для данного представления\*@

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce

varius nisl ac ipsum gravida vel pretium tellus.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce

varius nisl ac ipsum gravida vel pretium tellus.

</p>

## ViewStart



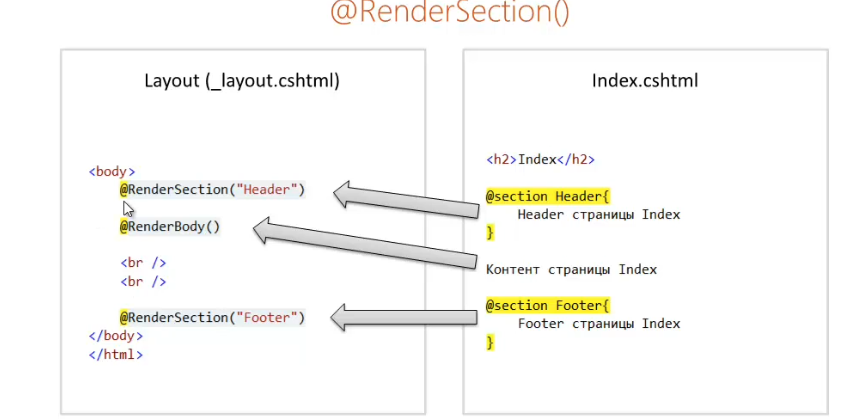
@\*Код в этом файле выполнится перед вызовом любого представления\*@

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

## RenderSection



<!DOCTYPE html>

<html>

<head>

<title>@ViewBag.Title</title>

<link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />

<script src="@Url.Content("~/Scripts/jquery-1.8.2.min.js")" type="text/javascript"></script>

</head>

<body>

@\*Визуализация содержимого секции с именем Header\*@

@RenderSection("Header")

@\*Визуализация контента представления, который не пренадлежит секциям\*@

@RenderBody()

<br />

<br />

@\*Визуализация содержимого секции с именем Footer\*@

@RenderSection("Footer")

</body>

</html>

Index

@{

ViewBag.Title = "Index Sections Sample";

}

<h3>Index View</h3>

@section Header{

<h1>

Header представления Index

</h1>

}

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce varius nisl ac ipsum gravida vel pretium tellus.

Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.

Placerat suscipit, orci nisl iaculis eros, a tincidunt nisi odio eget lorem nulla condimentum tempor mattis ut vitae feugiat augue cras ut metus a risus iaculis scelerisque eu ac ante.

</p>

@section Footer{

<h4>

Footer страницы Index

</h4>

}

## IsSectionDefine

<!DOCTYPE html>

<html>

<head>

<title>@ViewBag.Title</title>

<link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />

<script src="@Url.Content("~/Scripts/jquery-1.8.2.min.js")" type="text/javascript"></script>

</head>

<body>

<div>

@Html.ActionLink("Index", "Index") | @Html.ActionLink("Foo", "Foo")

</div>

<br />

<br />

@RenderSection("Body")

@\*Проверка наличия секции Footer в текущем представлении\*@

@if (IsSectionDefined("Footer"))

{

@RenderSection("Footer");

}

else

{

<h4>Default footer</h4>

}

</body>

</html>

Index chtml

@{

ViewBag.Title = "Index";

}

@section Body

{

<h2>Index View</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce varius nisl ac ipsum gravida vel pretium tellus.

Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.

Placerat suscipit, orci nisl iaculis eros, a tincidunt nisi odio eget lorem nulla condimentum tempor mattis ut vitae feugiat augue cras ut metus a risus iaculis scelerisque eu ac ante.

</p>

}

@section Footer

{

<footer>

All rights reserved. <a href="mailto:support@cbsystematics.com">support</a>

</footer>

}

Foo chtml

@{

ViewBag.Title = "Foo";

}

@section Body

{

<h2>Foo View</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit fusce vel sapien elit in malesuada semper mi, id sollicitudin urna fermentum ut fusce varius nisl ac ipsum gravida vel pretium tellus.

Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.

Placerat suscipit, orci nisl iaculis eros, a tincidunt nisi odio eget lorem nulla condimentum tempor mattis ut vitae feugiat augue cras ut metus a risus iaculis scelerisque eu ac ante.

</p>

}

RenderSections required

<div>

@Html.ActionLink("Index", "Index") | @Html.ActionLink("Foo", "Foo")

</div>

<br />

<br />

@RenderSection("Body")

@\*Визуализация необязательной секции Footer\*@

@RenderSection(name: "Footer", required: false)

</body>

</html>

BundleConfig

public class BundleConfig

{

// For more information on Bundling, visit http://go.microsoft.com/fwlink/?LinkId=254725

public static void RegisterBundles(BundleCollection bundles)

{

// Создание бандла ~/bundles/jquery в который войдут файлы jquery-\*

bundles.Add(new ScriptBundle("~/bundles/jquery")

.Include("~/Scripts/jquery-{version}.js")

.Include("~/Scripts/jquery.validate.js"));

// Создание бандла ~/bundles/modernizr для библиотеки modernizr

bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(

"~/Scripts/modernizr-\*"));

// Создание бандла для стилей

bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

}

}

Layout chtml

<title>@ViewBag.Title</title>

@\*

Бандл - набор CSS или JavaScript файлов, которые на этапе выполнения будут минимизированы и объеденены в один файл.

Минификация и бандлинг будут работать только если запустить веб приложение с настройкой

<compilation debug="false" targetFramework="4.5" /> в файле web.config

Настройка бандлов происходит в файле App\_Start/BundleConfig.cs

\*@

@\*

Бандл, который содержит все стили.

На этапе выполнения преобразовывается в <link herf="...." />

\*@

@Styles.Render("~/Content/css")

@\*

Бандл ссылается на JavaScrip библиотеку modernizr для определения поддержки HTML5 и CSS3 возможностей в браузере

На этапе выполнения преобразовывается в <script src="...."></script>

\*@

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

@RenderBody()

@\*Бандл с JavaScrip библиотекой jQuery\*@

@Scripts.Render("~/bundles/jquery")

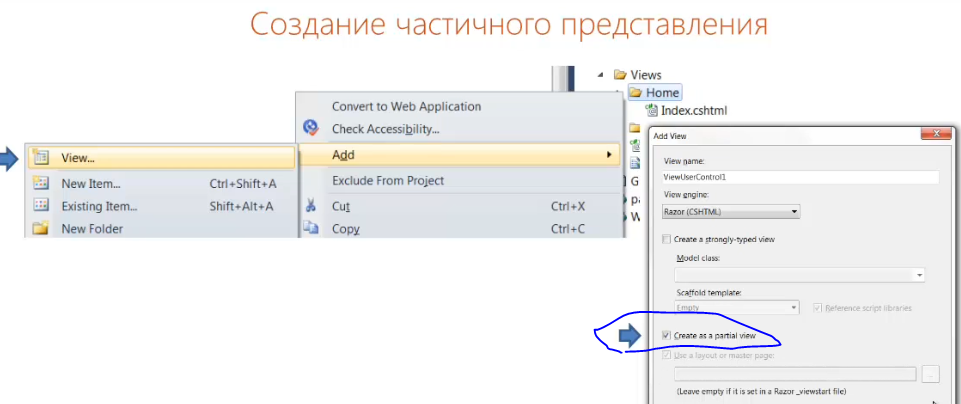
@\*Не обязательная секция для сценариев отдельных представлений\*@

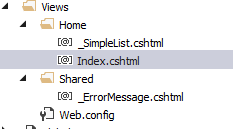
@RenderSection("scripts", required: false)

</body>

</html>

# PartialViews





## HtmlPartial

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Использование частичных представлений</title>

</head>

<body>

<div>

@\*Визуализация частичного представления (Views/Home/\_SimpleList.cshtml)\*@

@Html.Partial("\_SimpleList")

@\*Визуализация частичного представления (Views/Shared/\_ErrorMessage.cshtml)\*@

@Html.Partial("\_ErrorMessage")

</div>

</body>

</html>

## StronglyTypedPartialView

Index chtml

@model IEnumerable<\_02\_StronglyTypedPartialView.Models.Product>

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Strongly Typed Partial Views</title>

</head>

<body>

<div>

@foreach (var item in Model)

{

// Второй параметр метода Partial - модель, которая будет доступна в частичном представлении

@Html.Partial("\_ProductDesc", item)

}

</div>

</body>

</html>

ProductDecs.chtml

@model \_02\_StronglyTypedPartialView.Models.Product

<div>

<ul>

<li># @Model.Id</li>

<li>Name: @Model.Name</li>

<li>Price: @Model.Price $</li>

</ul>

</div>

HomeController

public ActionResult Index()

{

List<Product> products = new List<Product>();

products.Add(new Product()

{

Id = 1,

Name = "Item 1",

Price = 10

});

products.Add(new Product()

{

Id = 2,

Name = "Item 2",

Price = 5

});

products.Add(new Product()

{

Id = 3,

Name = "Item 3",

Price = 50

});

return View(products);

}

}

## Action Child вызов метода

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Using Child Actions</title>

</head>

<body>

<div>

@\*

Дочернее действие - метод действия контроллера, который можно выполнить вызвав метод Html.Action в представлении

@Html.Action("CurrentDate") - Вызов метода в контроллере и вывод результата в представление.

\*@

<p>

@Html.Action("CurrentDate")

</p>

</div>

</body>

</html>

Controller

[ChildActionOnly] // метод может быть запущен только как дочернее действие

public string CurrentDate()

{

return DateTime.Now.ToShortDateString();

}

## WorkWithPartialAction

public class ProductCollection

{

public static List<Product> All

{

get

{

List<Product> products = new List<Product>();

for (int i = 0; i < 20; i++)

{

products.Add(new Product()

{

Id = i + 1,

Name = "Item Name " + i,

Price = (i + 1) \* 2

});

}

return products;

}

}

Controller

public ActionResult ShowTable(int numberOfRows = 5)

{

IEnumerable<Product> products = ProductCollection.All.Take(numberOfRows);

return PartialView("\_Table", products);

}

\_Table.chtml

@model IEnumerable<\_04\_ChildActionUsing.Models.Product>

<table border="1">

<tr>

<th>

#

</th>

<th>

Name

</th>

<th>

Price

</th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>

@item.Id

</td>

<td>

@item.Name

</td>

<td>

$ @item.Price

</td>

</tr>

}

</table>

Index.chtml

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

@\*В отличии от частичных представлений дочерние действия позволяют выполнять дополнительные действия в контроллере\*@

@Html.Action("ShowTable", new { numberOfRows = 10 })

</div>

</body>

</html>

# Контроллеры

## IController

// В MVC классы контроллеров должны наследоваться от интерфейса IController

// Для получения доступа к данному контроллеру нужно выполнить запрос /Basic/имя\_действия

public class BasicController : IController

{

// Метод Execute вызывается, когда запрос направляется этому контроллеру.

public void Execute(System.Web.Routing.RequestContext requestContext)

{

string controller = (string)requestContext.RouteData.Values["controller"];

string action = (string)requestContext.RouteData.Values["action"];

requestContext.HttpContext.Response.Write(

string.Format("Controller: {0}, Action: {1}", controller, action));

}

}

## Свойства контроллера

public ActionResult Index()

{

// Свойства контроллера для доступа к информации о запросе.

// Request - данные о текущем HTTP запросе.

// Response - данные о текущем HTTP ответе.

// RouteData - данные маршрутизации для текущего запроса.

// HttpContext - получение специфической информации о текущем HTTP запросе.

// Server - объект с методами для обработки HTTP запроса.

string userName = User.Identity.Name;

string machineName = Server.MachineName;

string clientIp = Request.UserHostAddress;

string formData = Request.Form["data"];

string queryStringData = Request.QueryString["data"];

HttpCookie cookie = Request.Cookies["cookieName"];

## Отправка данных контроллеру

//

// GET: /Data/

public ActionResult Index()

{

return View();

}

//

// POST: /Data/DataInformation/

public ActionResult PostInformation()

{

// Чтение данных, которые передаются с помощью POST запроса.

ViewBag.Message = Request.Form["message"];

return View("Index");

}

//

// GET: /Data/RouteInformation/

public ActionResult RouteInformation()

{

// Чтение данных, которые передаются с помощью GET запроса, как данные в маршруте.

ViewBag.Message = RouteData.Values["id"];

return View("Index");

}

//

// GET: /Data/QueryInformation/

public ActionResult QueryInformation()

{

// Чтение данных, которые передаются в адресной строке.

ViewBag.Message = Request.QueryString["message"];

return View("Index");

}

View

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

<!--Данные отправляются с помощью POST запроса-->

<form action="/Data/PostInformation" method="post">

<input type="text" name="message" />

<input type="submit" value="Отправить на сервер" />

</form>

<!--Данные отправляются с помощью GET запроса, как данные в маршруте-->

<a href="/Data/RouteInformation/hello-world-routedata">Route Data</a>

<br />

<!--Данные отправляются с помощью GET запроса, как данные адресной строки-->

<a href="/Data/QueryInformation?message=hello-world-querystring">QueryString Data</a>

</div>

</body>

</html>

## Отправка в параметр действия

Названия параметра действия = названия елемента на форме

public ActionResult Index()

{

return View();

}

// Платформа предоставит значения для параметров, проверив соответствующие объекты контекста,

// включая Request.Form, RouteData.Values, Request.QueryString

//

// POST: /Data/PostInformation

public ActionResult PostInformation(string message)

{

//ViewBag.Message = Request.Form["message"];

ViewBag.Message = message;

return View("Index");

}

//

// GET: /Data/RouteInformation

public ActionResult RouteInformation(string id)

{

//ViewBag.Message = RouteData.Values["id"];

ViewBag.Message = id;

return View("Index");

}

//

// GET: /Data/QueryInformation

public ActionResult QueryInformation(string message)

{

//ViewBag.Message = Request.QueryString["message"];

ViewBag.Message = message;

return View("Index");

}

## Создание правильной формы

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost] // метод будет вызываться только при POST запросах

public ActionResult Index(int x, int y)

{

ViewBag.Result = x + y;

return View();

}

View

<body>

<div>

@\* BeginForm - создает элемент form с атрибутами в соответсвии с параметрами\*@

@using (Html.BeginForm("Index", "Home"))

{

@\*@Html.TextBox("x") - создает элемент input type="text" с атрибутам в соответсвии с настройками\*@

<label>X</label> @Html.TextBox("x") <br />

<label>Y</label> @Html.TextBox("y") <br />

if (ViewBag.Result != null)

{

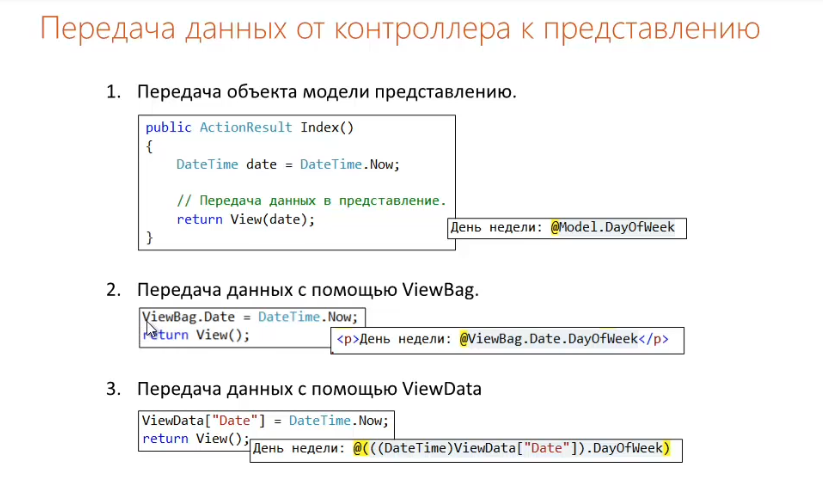
<p>@ViewBag.Result</p>

}

<input type="submit" name="operation" value="Get Result" />

}

# Передача данных в представления



//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult Action1()

{

DateTime date = DateTime.Now;

// Передача данных в представление.

return View(date);

}

public ActionResult Action2()

{

// Использование объекта ViewBag (это динамический тип данных).

ViewBag.Date = DateTime.Now;

return View();

}

public ActionResult Action3()

{

// Запись данных во ViewData для представления.

ViewData["Date"] = DateTime.Now;

return View();

}

View

@\*Указание типа модели\*@

@model System.DateTime

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

@\*Получение доступа к модели\*@

День недели: @Model.Year

</div>

</body>

</html>

# TempData

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult Action1()

{

// TempData удобно использовать, если необходимо сохранить данные между двумя запросами

// Объект TempData похож на Session, но после прочтения значения из TempData оно помечается на удаление

// и удаляется после обработки запроса.

TempData["info"] = "hello world from Action1";

return View();

}

public ActionResult Action2()

{

// При чтении значения по ключу info, после завершения обработки запроса, запись в TempData будет удалена.

//string info = TempData["info"] as string;

// Получение значения из TempData не помечая его на удаление.

string info = TempData.Peek("info") as string;

return View((object)info);

}

public ActionResult Action3()

{

// Объекта по ключу info не существует.

string info = TempData["info"] as string;

return View((object)info);

}

View

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<title>Action1</title>

</head>

<body>

<div>

<h2>Controller:Home, Action: Action1</h2>

Данные записаны в TempData

<br />

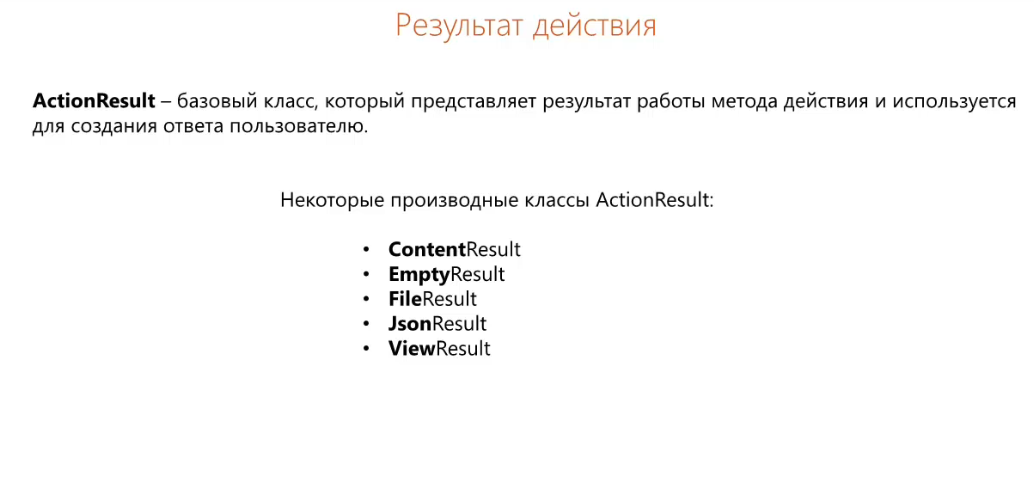
@Html.ActionLink("Действие #2", "Action2")

</div>

</body>

</html>

# Результаты действия



## Redirect methods

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult GoTo1()

{

// Переход со статус кодом 302 - временное перенаправление.

// Использование метода из базового класса Controller

return Redirect("http://itvdn.com");

}

public ActionResult GoTo2()

{

// Переход со статус кодом 302 - временное перенаправление.

// Использование экземпляра класса RedirectResult

return new RedirectResult("http://itvdn.com");

}

public ActionResult GoTo3()

{

// Переход со статус кодом 301 - постоянное перенаправление.

return RedirectPermanent("http://itvdn.com");

// Второй вариант постоянного перенаправления.

//return new RedirectResult("http://itvdn.com", true);

}

public ActionResult GoTo4()

{

return RedirectToRoute(new {

controller = "TestRedirect",

action = "Index",

id = "Hello from RedirectToRoute"

});

}

public ActionResult GoTo5()

{

return RedirectToAction("Index", "TestRedirect", new { id = "Hello from RedirectToAction" });

}

}

## Content and EmptyRezult

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult Action1()

{

// Content - создает ActionResult представляющий строковой контент отправляющийся в ответ пользователю.

// первый параметр - текстовые данные, которые необходимо отправить.

// второй параметр - MIME тип возвращаемых данных, по умолчанию text/html.

// третий параметр - кодировка.

return Content("Hello World", "text/plain", Encoding.UTF8);

}

public ActionResult Action2()

{

// Отправка пустого ответа.

return new EmptyResult();

}

## Json and Ajax

//

// GET: /Data/

[HttpGet] // атрибут устанавливает ограничение на метод Action1 и позволяет выполнять его только при GET запросе

public ActionResult Action1()

{

var product = new

{

Name = "Item 1",

Price = 10,

Description = "Dummy Text ....",

};

// По умолчанию MVC Framework не будет возвращать JSON в ответ на GET запрос по причнинам безопасности.

// Возвращать в ответ на GET запрос JSON безопасно если в ответе не содержатся чувствительные данные.

// http://haacked.com/archive/2009/06/25/json-hijacking.aspx/

return Json(product, JsonRequestBehavior.AllowGet);

}

[HttpPost] // атрибут устанавливает ограничение на метод Action2 и позволяет выполнять его только при POST запросе

public ActionResult Action2()

{

var product = new

{

Name = "Item 1",

Price = 10,

Description = "Dummy Text ....",

};

return Json(product);

}

View

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

@Html.ActionLink("Action #1", "Action1", "Data") <br />

<a id="sendRequest" href="#">Action #2 (JavaScript)</a>

<div id="result"></div>

<script src="~/Scripts/jquery-2.0.2.min.js"></script>

<script>

$(document).ready(function () {

$("#sendRequest").click(function () {

// Получение данных с помощью GET запроса

$.ajax({

url: "@Url.Action("Action2", "Data")",

type : "POST",

success: function (data) {

$("#result")

.append("Name: " + data.Name + "<br />")

.append("Price: " + data.Price + "<br />")

.append("Description: " + data.Description + "<br />");

}

});

});

});

</script>

</div>

</body>

## Загрузка файлов

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult DownloadFile()

{

string filename = Server.MapPath("/Content/File.pdf");

string contentType = "application/pdf"; // MIME Type image/png image/jpg

string downloadName = "PDF File";

// Если имя файла для скачивания не указано и если

// браузер поддерживает тип файла, файл откроется в самом браузере.

//downloadName = null;

return File(filename, contentType, downloadName);

}

public ActionResult DownloadBytes()

{

string filename = Server.MapPath("/Content/File.pdf");

string contentType = "application/pdf";

byte[] data = System.IO.File.ReadAllBytes(filename);

return File(data, contentType);

}

public ActionResult DownloadStream()

{

string filename = Server.MapPath("/Content/File.pdf");

string contentType = "application/pdf";

FileStream stream = System.IO.File.OpenRead(filename);

return File(stream, contentType);

}

}

View

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

@Html.ActionLink("Скачать файл 1", "DownloadFile")

<br />

@Html.ActionLink("Скачать файл 2", "DownloadBytes")

<br />

@Html.ActionLink("Скачать файл 3", "DownloadStream")

</div>

</body>

## Exception Status Code

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult Action1()

{

// Возвращение статуса 404 NotFound

return HttpNotFound();

}

public ActionResult Action2()

{

int result;

try

{

int a = 10, b = 0;

result = a / b;

}

catch (Exception)

{

// Возвращение статуса 500 Internal Server Error

return new HttpStatusCodeResult(500, "Internal Server Error");

}

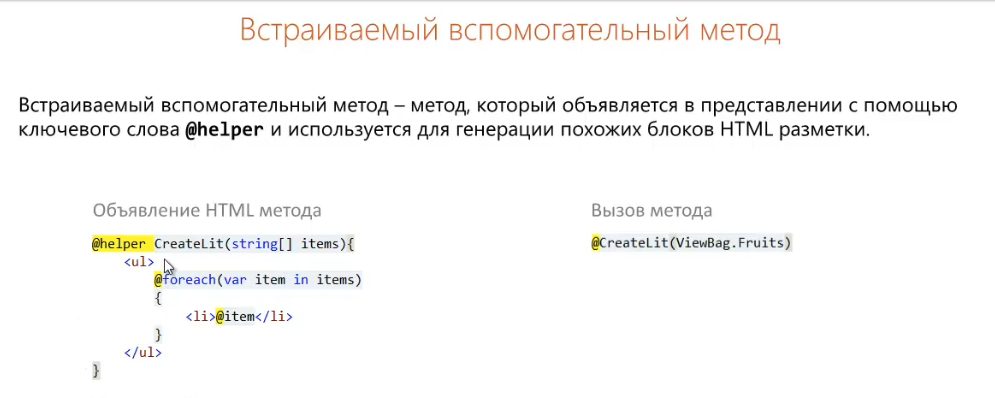
return Content(result.ToString());

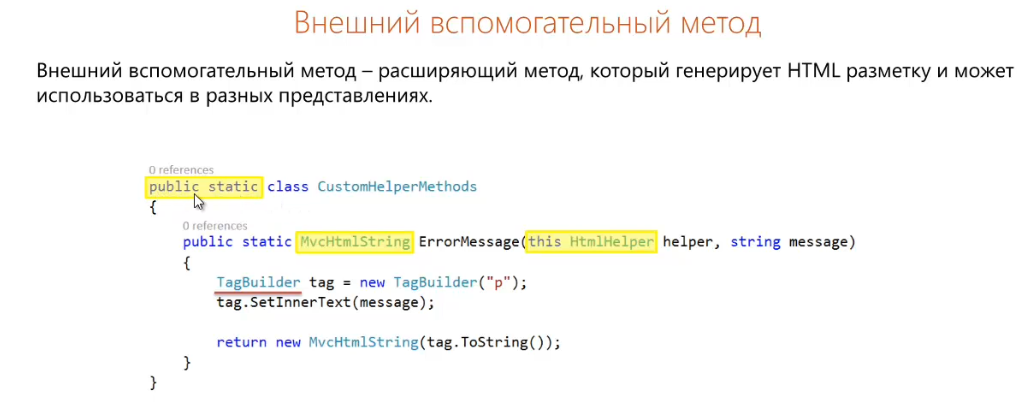
}

}

# Хелперы и методы

## Встраиваемые методы





//

// GET: /Home/

public ActionResult Index()

{

ViewBag.Days = new string[] { "Monday", "Tuesday", "Wednesday" };

ViewBag.Fruits = new string[] { "Mango", "Banana", "Apple" };

return View();

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@\*

Встраиваемый вспомогательный метод HTML.

Метод может использоваться только в этом представлении.

\*@

@helper CreateLit(string[] items)

{

<ul>

@foreach (var item in items)

{

<li>@item</li>

}

</ul>

}

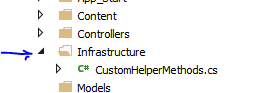
Дни недели:

@CreateLit(ViewBag.Days)

Фрукты:

@CreateLit(ViewBag.Fruits)

## Внешние методы



namespace \_02\_ExternalHelperMethods.Infrastructure

{

public static class CustomHelperMethods

{

// Для того чтобы использовать данный вспомогательный метод, необходимо импортировать пространство имен \_02\_ExternalHelperMethods.Infrastructure.

// В данном проекте, пространство имен импортировано с помощью файла Views/web.config

public static MvcHtmlString UnorderedList(this HtmlHelper helper, string[] items)

{

// TagBuilder - класс для создания HTML элементов.

TagBuilder tag = new TagBuilder("ul");

foreach (var item in items)

{

TagBuilder liTag = new TagBuilder("li");

liTag.SetInnerText(item);

tag.InnerHtml += liTag.ToString();

}

// MvcHtmlString - представляет HTML кодированную строку, которая не должна кодироваться повторно.

// Необходимо самостоятельно позаботится о кодировании тех данных, с которыми связан риск XSS атаки

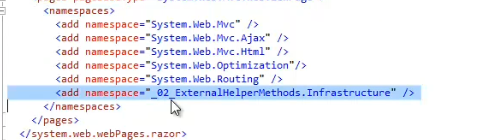
return new MvcHtmlString(tag.ToString());

}

}

}

Добавить пространство имен



Controller

//

// GET: /Home/

public ActionResult Index()

{

ViewBag.Days = new string[] { "Monday", "Tuesday", "Wednesday" };

ViewBag.Fruits = new string[] { "Mango", "Banana", "Apple" };

return View();

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

Дни недели:

@Html.UnorderedList(ViewBag.Days as string[])

Фрукты:

@Html.UnorderedList(ViewBag.Fruits as string[])

TagBuilder



namespace \_03\_ExternalHelperMethods.Infrastructure

{

public static class CustomHelperMethods

{

public static MvcHtmlString ErrorMessage(this HtmlHelper helper, string message)

{

TagBuilder tag = new TagBuilder("p");

tag.MergeAttribute("style", "color:red; padding:8px; border:1px solid red; border-radius:3px;");

tag.SetInnerText(message);

return new MvcHtmlString(tag.ToString());

}

}

}

View

@using \_03\_ExternalHelperMethods.Infrastructure;

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@Html.ErrorMessage("Unknown Error")

# HTML Form

## Способы создания форм

@{

ViewBag.Title = "Index";

}

<h2>Способы создания HTML форм</h2>

@\*

Html.BeginForm - создание формы с установкой значения атрибута action. Значение устанавливается с учетом механизма маршрутизации.

\*@

@{

Html.BeginForm("Index", "Home");

}

<p>Первая форма</p>

<input type="submit" />

@{

Html.EndForm();

}

@using (Html.BeginForm("Index", "Home"))

{

<p>Вторая форма</p>

<input type="submit" />

}

## HTML Elements

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@using (Html.BeginForm("Home", "index"))

{

@:Флажок

@Html.CheckBox("myCheckBox", false)

<br />

@:Скрытое поле

@Html.Hidden("myHiddenField", "val")

<br />

@:Переключатель

@Html.RadioButton("myRadioButton", "val", true)

<br />

@:Поле для ввода пароля

@Html.Password("myPassword", "val")

<br />

@:Текстовая область

@Html.TextArea("myTextArea1", "val", 5, 20, null);

<br />

@:Текстовое поле

@Html.TextBox("myTextArea2", "val")

<br />

}

## Пример работы с формой

Model

public class TaskModel

{

public string Name { get; set; }

public string StartDate { get; set; }

public bool Completed { get; set; }

}

}

Controller

// Возвращает главную страницу со ссылкой на форму создания новой задачи

// GET: /Home/

public ActionResult Index()

{

return View();

}

// Возвращает форму для создания новой задачи

// GET: /Home/Create

public ActionResult Create()

{

TaskModel model = new TaskModel();

model.Name = "Unknown";

model.StartDate = DateTime.Now.ToShortDateString();

model.Completed = false;

return View(model);

}

// Получает данные переданные через форму и создает задачу

// POST: /Home/Create

[HttpPost]

public ActionResult Create(TaskModel incomingData)

{

Debug.WriteLine("Name = " + incomingData.Name);

Debug.WriteLine("StartDate = " + incomingData.StartDate);

Debug.WriteLine("Completed = " + incomingData.Completed);

return View("Success");

}

}

Index.chtml

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@Html.ActionLink("Create New Task", "Create")

Create.chtml

@{

ViewBag.Title = "Create New Task";

}

@\*

При использовании перегрузки @Html.TextBox(string) создается input с типом text и атрибутом name и id указанном в параметре.

Важная особенность данного метода - поиск данных для атрибута value элемента при вызове.

При следующем вызове @Html.TextBox("Name") поиск будет происходить в такой последовательности:

1) ViewBag.Name

2) ViewData["Name"]

3) @Model.Name

\*@

@using (Html.BeginForm())

{

<fieldset>

<legend>Create New Task Form</legend>

<div>

<label>Task Name</label>

@Html.TextBox("Name")

</div>

<div>

<label>Start Date</label>

@Html.TextBox("StartDate")

</div>

<div>

<label>Is Completed</label>

@Html.CheckBox("Completed")

</div>

<input type="submit" value="Create" />

</fieldset>

}

Success.chtml

@{

ViewBag.Title = "Success";

}

<h2>Success</h2>

<p>Congratulations! Task has been successfully created.</p>

## Строго типизированное представление

@model \_04\_StronglyTypedHelperMethods.Models.TaskModel

@{

ViewBag.Title = "Create New Task";

}

@\*

@Html.TextBoxFor(лямбда выражение) - строго типизированный вспомогательный метод для создание элемента управления и

задания для него значения из модели (значение берется на основе лямбда выражения).

Может использоваться только в строго типизированных представлениях.

\*@

@using (Html.BeginForm())

{

<fieldset>

<legend>Create New Task Form</legend>

<div>

<label>Task Name</label>

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

</div>

<div>

<label>Start Date</label>

@Html.TextBoxFor(m => m.StartDate)

</div>

<div>

<label>Is Completed</label>

@Html.CheckBoxFor(m => m.Completed)

</div>

<input type="submit" value="Create" />

</fieldset>

}

## Выпадающие списки

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(string list1, string[] list2)

{

ViewBag.Selected1 = list1;

string result = string.Empty;

foreach (var item in list2)

{

result += item + ", ";

}

ViewBag.Selected2 = result.TrimEnd(',', ' ');

return View();

}

}

View

@{

ViewBag.Title = "Index";

}

@using (Html.BeginForm())

{

@:Раскрывающийся список

@Html.DropDownList("list1",

new SelectList(new[] { "Item 1", "Item 2", "Item 3" }))

<br />

@:Список с множественным выбором

@Html.ListBox("list2",

new MultiSelectList(new[] { "A", "B", "C" }),

new { style = "width:100px;" })

<br />

<input type="submit" name="button" value="Отправить" />

}

<br />

<br />

<div>

Раскрывающийся список - @ViewBag.Selected1

<br />

Список с множественным выбором - @ViewBag.Selected2

</div>

## Заполнение списка с контролла

//

// GET: /Home/

public ActionResult Index()

{

List<Product> products = new List<Product>()

{

new Product() { Id = 1, Name ="Product #1"},

new Product() { Id = 2, Name ="Product #2"},

new Product() { Id = 3, Name ="Product #3"},

new Product() { Id = 4, Name ="Product #4"}

};

ViewBag.Products = new SelectList(products, "Id", "Name", 2);

return View();

}

[HttpPost]

public ActionResult Index(string Products)

{

return View("Selected", (object)Products);

}

Index.chtml

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@using (Html.BeginForm())

{

@Html.DropDownList("Products")

<input type="submit" />

}

Selected.chtml

@{

ViewBag.Title = "Selected";

}

<h2>Selected</h2>

<p>Выбранный элемент - @Model</p>

# WebGrid

Добавить ссылку System.Web.Helpers

Model

public static class BrowsersDatabase

{

private static List<BrowserInfo> \_browsers = null;

public static List<BrowserInfo> Browsers

{

get

{

if (\_browsers == null)

{

\_browsers = new List<BrowserInfo>();

BrowserInfo info;

info = new BrowserInfo();

info.ID = 1;

info.Name = "Google Chrome";

info.ImagePath = "/Content/Images/chrome.png";

info.Url = "http://www.google.ru/chrome";

\_browsers.Add(info);

info = new BrowserInfo();

info.ID = 2;

info.Name = "Microsoft Internet Explorer";

info.ImagePath = "/Content/Images/internet\_explorer.png";

info.Url = "http://windows.microsoft.com/ru-RU/internet-explorer/products/ie/home";

\_browsers.Add(info);

info = new BrowserInfo();

info.ID = 3;

info.Name = "Mozilla Firefox";

info.ImagePath = "/Content/Images/mozilla.png";

info.Url = "http://www.mozilla.org/en-US/firefox/new/";

\_browsers.Add(info);

info = new BrowserInfo();

info.ID = 4;

info.Name = "Opera";

info.ImagePath = "/Content/Images/opera.png";

info.Url = "http://www.opera.com/download/";

\_browsers.Add(info);

info = new BrowserInfo();

info.ID = 5;

info.Name = "Safari";

info.ImagePath = "/Content/Images/safari.png";

info.Url = "http://www.apple.com/safari/download/";

\_browsers.Add(info);

}

return \_browsers;

}

}

}

public class BrowserInfo

{

public int ID { get; set; }

public string Name { get; set; }

public string ImagePath { get; set; }

public string Url { get; set; }

}

Controller

//

// GET: /Home/

public ActionResult Index()

{

return View(BrowsersDatabase.Browsers);

}

Views

@model IEnumerable<\_01\_WebGrid.Models.BrowserInfo>

@{

ViewBag.Title = "Index";

// WebGrid - генерирует разметку для отображения данных в таблице.

// Для проекта созданного по шаблону Basic необходимо добавить ссылку на сборку System.Web.Halpers и в свойствах сборки

// установить значение Copy Local = True для того чтобы использовать класс WebGrid

var grid = new WebGrid(Model);

}

<h2>WebGrid Sample</h2>

@\*Получение разметки\*@

@grid.GetHtml()

## WebGrid Styles

@{

ViewBag.Title = "WebGrid Styles";

var grid = new WebGrid(source: Model, rowsPerPage: 3);

}

@section styles

{

<link href="~/Content/GridStyles.css" rel="stylesheet" />

}

<h2>WebGrid Styles</h2>

@\*Стили для WebGrid находятся в файле /Content/GridStyles.css\*@

@grid.GetHtml(tableStyle: "grid",

headerStyle: "header",

rowStyle: "row",

alternatingRowStyle: "alt-row",

footerStyle: "footer",

htmlAttributes: new { cellpadding = "6px", cellspacing = "0px" })

css

/\*Стиль для всей таблицы\*/

.grid

{

background-color:black;

border:0;

}

/\*Стиль для заголовка таблицы\*/

.header

{

background-color:Red;

}

/\*Строка таблицы\*/

.row

{

background-color:White;

}

/\*Нечетная строка таблицы\*/

.alt-row

{

background-color:whiteSmoke;

}

/\*Последняя строка таблицы\*/

.footer

{

background-color:Yellow;

}

WebGrid настройка вывода

@{

ViewBag.Title = "WebGrid Columns";

var grid = new WebGrid(Model);

}

<h2>WebGrid Columns</h2>

@grid.GetHtml(

columns: grid.Columns(

grid.Column(columnName: "Name", header: "Имя"),

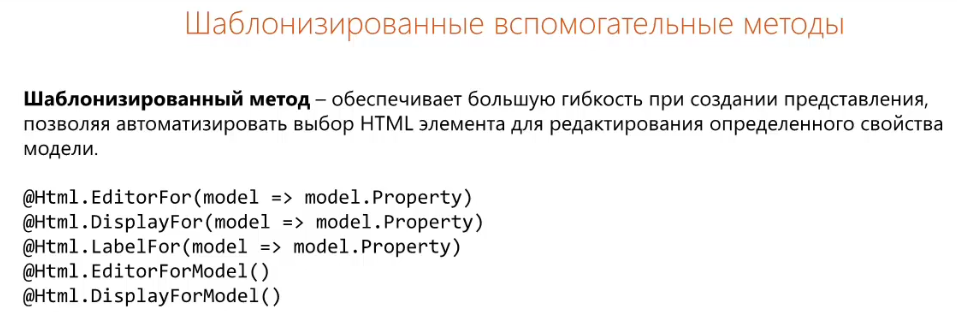
grid.Column(header: "Логотип", format: @<img src="@item.ImagePath" />, canSort:false),

grid.Column(header: "Скачать браузер", format: @<a href="@item.Url">Скачать @item.Name</a>)

)

)

# Шаблонизированные методы



## Editor and Display

//

// GET: /Home/

public ActionResult Index()

{

Product p = new Product()

{

Id = 1,

Name = "Test Product",

IsTangible = false

};

return View(p);

}

View

@model TemplatedHelperMethods.Models.Product

@{

ViewBag.Title = "Index";

}

<h2>Шаблонизированные вспомогательные методы</h2>

@\*

LabelFor - визуализирует элемент label соотвествующий определенному свойству модели.

EditorFor - визуализирует редактор для изменения определенного свойства модели.

DisplayFor - визуализирует представление только для чтения, определенного совйства модели.

\*@

@Html.LabelFor(x => x.Id)

@Html.EditorFor(x => x.Id)

<br />

@Html.LabelFor(x => x.Name)

@Html.EditorFor(x => x.Name)

<br />

@Html.LabelFor(x => x.IsTangible)

@Html.EditorFor(x => x.IsTangible)

<br />

<h2>Шаблонизированные вспомогательные методы (разметка только на чтение)</h2>

@Html.LabelFor(x => x.Id)

@Html.DisplayFor(x => x.Id)

<br />

@Html.LabelFor(x => x.Name)

@Html.DisplayFor(x => x.Name)

<br />

@Html.LabelFor(x => x.IsTangible)

@Html.DisplayFor(x => x.IsTangible)

<br />

## EditorForModel and DisplayForModel

@model EditorForModel.Models.Product

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@\*EditorForModel - визуализирует редакторы для целого объекта модели\*@

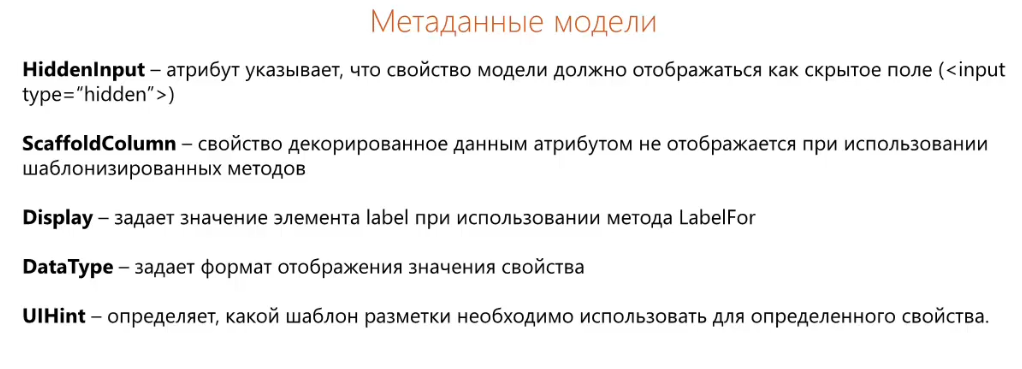
@Html.EditorForModel()

<br />

@\*DisplayForModel - визуализирует представление для чтения свойств целого объекта модели\*@

@Html.DisplayForModel()

# Метаданные модели



## Атрибуты модели

Модель

public class Customer

{

//[HiddenInput] // Указывает, что свойство должно быть отображено как hidden input и текстовый литерал.

[HiddenInput(DisplayValue = false)] // Указывает, что свойство должно быть отображено только как hidden input.

public int Id { get; set; }

[Display(Name = "Имя")] // Определение имени метки, которая находится слева от поля ввода.

public string FirstName { get; set; }

[Display(Name = "Фамилия")]

public string LastName { get; set; }

[DataType(DataType.Date)] // Инструкция для отображения значения свойства модели. Теперь выводиться дата без времени.

public DateTime RegistrationDate { get; set; }

public bool IsApproved { get; set; }

[ScaffoldColumn(false)] // Исключает свойство из генерируемой HTML разметки.

public int Foo { get; set; }

[UIHint("MultilineText")] // Выбор шаблона отображения

public string Description { get; set; }

}

}

Controller

//

// GET: /Home/

public ActionResult Index()

{

Customer customer = new Customer()

{

Id = 1,

FirstName = "Ivan",

LastName ="Ivanov",

IsApproved = true,

Description = "text... text... text...",

RegistrationDate = DateTime.Now,

Foo = 123

};

return View(customer);

}

View

@model ModelsMetadata.Models.Customer

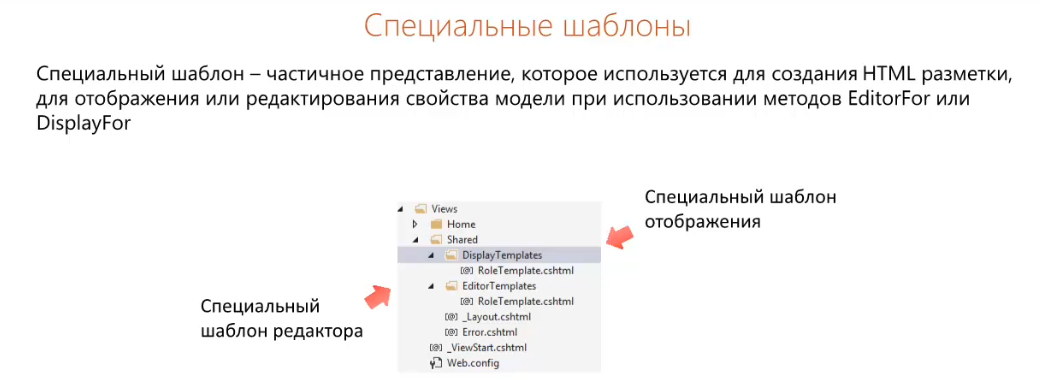
@{

ViewBag.Title = "Index";

}

@Html.EditorForModel()

# Специальные шаблоны



## Настройка EditorFor and DisplayFor

public class Person

{

public int PersonId { get; set; }

public string Name { get; set; }

public Role Role { get; set; }

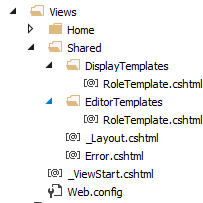
}

public enum Role

{

Guest, User, Admin

}



EditorTamplate

@\*Шаблон для редактирования значения свойства типа Role\*@

@using SpecialTemplateSample.Models

@model Role

<select id="Role" name="Role">

@foreach (Role value in Enum.GetValues(typeof(Role)))

{

<option value="@value" @(Model == value ? "selected=\"selected\"" : "")>

@value

</option>

}

</select>

@\*<select id="Role" name="Role">

<option value="Guest">Guest</option>

<option value="User">User</option>

<option value="Admin" selected="selected">Admin</option>

</select>\*@

DisplayTemplate

@\*Шаблон для отображения значений свойства типа Role\*@

@using SpecialTemplateSample.Models

@model Role

@foreach (Role value in Enum.GetValues(typeof(Role)))

{

if (value == Model)

{

<u>@value</u>@:&nbsp;

}

else

{

@value @:&nbsp;

}

}

@\*Guest&nbsp;User&nbsp;<u>Admin</u>&nbsp;\*@

View

@model SpecialTemplateSample.Models.Person

@{

ViewBag.Title = "Index";

}

<h2>EditorFor</h2>

@Html.LabelFor(x => x.PersonId)<br />

@Html.EditorFor(x => x.PersonId) <br />

@Html.LabelFor(x => x.Name)<br />

@Html.EditorFor(x => x.Name) <br />

@Html.LabelFor(x => x.Role)<br />

@\*Второй параметр указывает на название шаблона из директории EditorTempaltes \*@

@Html.EditorFor(x => x.Role/\*, "RoleTemplate"\*/) <br />

<h2>DisplayFor</h2>

@Html.LabelFor(x => x.PersonId)<br />

@Html.DisplayFor(x => x.PersonId) <br />

@Html.LabelFor(x => x.Name)<br />

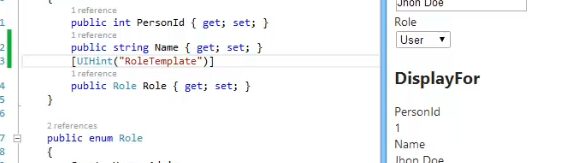
@Html.DisplayFor(x => x.Name) <br />

@Html.LabelFor(x => x.Role)<br />

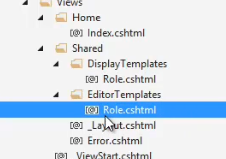
@\*Второй параметр указывает на название шаблона из директории DisplayTempaltes \*@

@Html.DisplayFor(x => x.Role/\*, "RoleTemplate"\*/) <br />

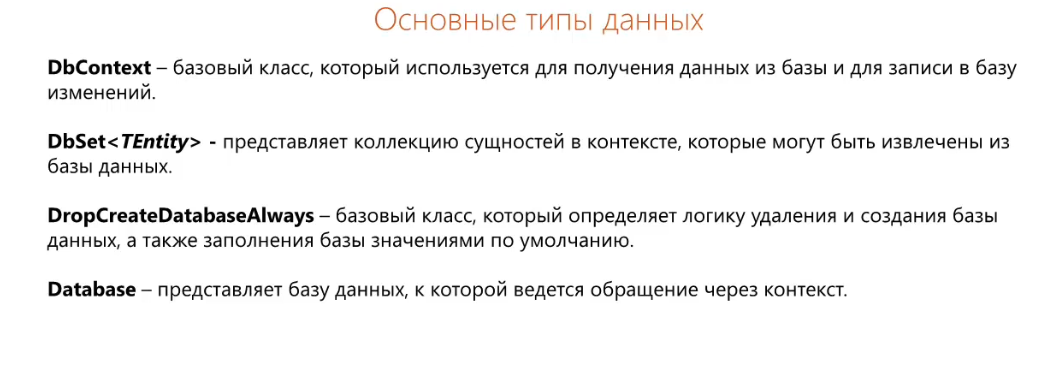
## UIHint



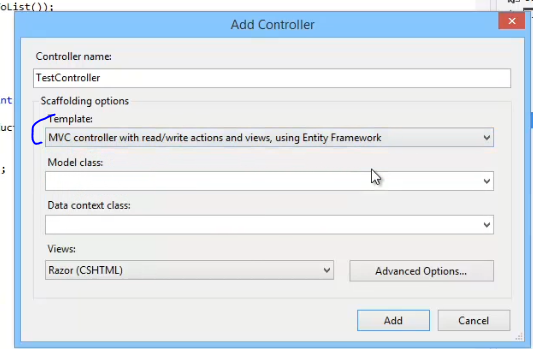
Или без слова Template



# EntityFramework



## Создания контроллера с CRUD – функционалом



## Работа с базой. CRUD.

Контекст

public class ProductsContext : DbContext

{

public ProductsContext()

: base("ProductConnection")

{

}

public DbSet<Product> Products { get; set; }

}

Controller

public class ProductsController : Controller

{

private ProductsContext db = new ProductsContext();

//

// GET: /Products/

public ActionResult Index()

{

return View(db.Products.ToList());

}

//

// GET: /Products/Details/5

public ActionResult Details(int id = 0)

{

Product product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

return View(product);

}

//

// GET: /Products/Create

public ActionResult Create()

{

return View();

}

//

// POST: /Products/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Product product)

{

if (ModelState.IsValid)

{

db.Products.Add(product);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(product);

}

//

// GET: /Products/Edit/5

public ActionResult Edit(int id = 0)

{

Product product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

return View(product);

}

//

// POST: /Products/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Product product)

{

if (ModelState.IsValid)

{

db.Entry(product).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(product);

}

//

// GET: /Products/Delete/5

public ActionResult Delete(int id = 0)

{

Product product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

return View(product);

}

//

// POST: /Products/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Product product = db.Products.Find(id);

db.Products.Remove(product);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

db.Dispose();

base.Dispose(disposing);

}

}

Index

@model IEnumerable<EntityFrameworkSamples.Models.Product>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table>

<tr>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Price)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

</tr>

}

</table>

Create

@model EntityFrameworkSamples.Models.Product

@{

ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

<fieldset>

<legend>Product</legend>

<div class="editor-label">

@Html.LabelFor(model => model.Name)

</div>

<div class="editor-field">

@Html.EditorFor(model => model.Name)

@Html.ValidationMessageFor(model => model.Name)

</div>

<div class="editor-label">

@Html.LabelFor(model => model.Price)

</div>

<div class="editor-field">

@Html.EditorFor(model => model.Price)

@Html.ValidationMessageFor(model => model.Price)

</div>

<p>

<input type="submit" value="Create" />

</p>

</fieldset>

}

<div>

@Html.ActionLink("Back to List", "Index")

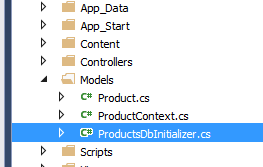
</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

## Инициализация базы данных



public class ProductsDbInitializer : DropCreateDatabaseAlways<ProductsContext>

{

protected override void Seed(ProductsContext context)

{

context.Products.Add(new Product() { Id = 1, Name = "Product 1", Price = 10 });

context.Products.Add(new Product() { Id = 2, Name = "Product 2", Price = 20 });

context.Products.Add(new Product() { Id = 3, Name = "Product 3", Price = 30 });

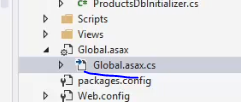
context.Products.Add(new Product() { Id = 4, Name = "Product 4", Price = 40 });

base.Seed(context);

}

}

Global.asax



public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

Database.SetInitializer(new ProductsDbInitializer());

AreaRegistration.RegisterAllAreas();

WebApiConfig.Register(GlobalConfiguration.Configuration);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

}

## Сложная модель

public class Customer

{

public int CustomerId { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public List<Order> Orders { get; set; }

}

public class Order

{

public int OrderId { get; set; }

public int CustomerId { get; set; }

public int ProductId { get; set; }

public DateTime Date { get; set; }

public Product Product { get; set; }

public Customer Customer { get; set; }

}

public class Product

{

public int ProductId { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

}

public class DatabaseContext : DbContext

{

public DatabaseContext()

: base("name=DatabaseContext")

{

}

public DbSet<Customer> Customers { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<Product> Products { get; set; }

}

Order Controller

private DatabaseContext db = new DatabaseContext();

//

// GET: /Orders/

public ActionResult Index()

{

var orders = db.Orders.Include(o => o.Product).Include(o => o.Customer);

return View(orders.ToList());

}

//

// GET: /Orders/Details/5

public ActionResult Details(int id = 0)

{

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

return View(order);

}

//

// GET: /Orders/Create

public ActionResult Create()

{

ViewBag.ProductId = new SelectList(db.Products, "ProductId", "Name");

ViewBag.CustomerId = new SelectList(db.Customers, "CustomerId", "Name");

return View();

}

//

// POST: /Orders/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Order order)

{

if (ModelState.IsValid)

{

db.Orders.Add(order);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.ProductId = new SelectList(db.Products, "ProductId", "Name", order.ProductId);

ViewBag.CustomerId = new SelectList(db.Customers, "CustomerId", "Name", order.CustomerId);

return View(order);

}

//

// GET: /Orders/Edit/5

public ActionResult Edit(int id = 0)

{

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

ViewBag.ProductId = new SelectList(db.Products, "ProductId", "Name", order.ProductId);

ViewBag.CustomerId = new SelectList(db.Customers, "CustomerId", "Name", order.CustomerId);

return View(order);

}

//

// POST: /Orders/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Order order)

{

if (ModelState.IsValid)

{

db.Entry(order).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.ProductId = new SelectList(db.Products, "ProductId", "Name", order.ProductId);

ViewBag.CustomerId = new SelectList(db.Customers, "CustomerId", "Name", order.CustomerId);

return View(order);

}

//

// GET: /Orders/Delete/5

public ActionResult Delete(int id = 0)

{

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

return View(order);

}

//

// POST: /Orders/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Order order = db.Orders.Find(id);

db.Orders.Remove(order);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

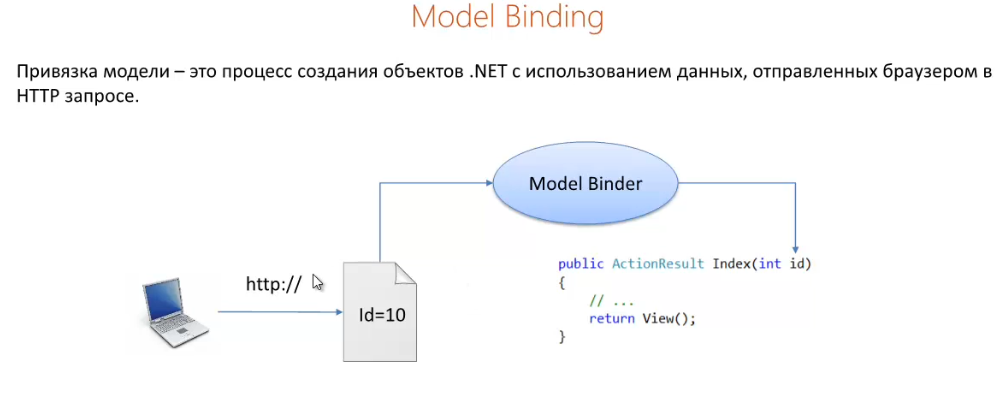
db.Dispose();

base.Dispose(disposing);

}

}

# Привязка модели





Простое звязывание

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

// Связыватель модели - объект который производит поиск значений в запросе в соответствии с именами параметров метода действия.

// Связыватель модели по умолчанию производит поиск значений в следующем порядке:

// 1) Request.Form

// 2) RouteData.Values

// 3) Request.QueryString

// 4) Request.Files

public ActionResult Action1(int id)

{

ViewBag.Message = "Метод действия Action1(int id)";

ViewBag.Data = id;

return View("ActionView");

}

public ActionResult Action2(int id = 1)

{

ViewBag.Message = "Метод действия Action2(int id = 1). При отсутствии значения для параметра id будет использоваться значение 1";

ViewBag.Data = id;

return View("ActionView");

}

public ActionResult Action3(int? id)

{

ViewBag.Message = "Метод действия Action3(int? id). При отсутствии значения для параметра id будет использоваться значение null";

ViewBag.Data = id;

return View("ActionView");

}

}

## Привязка к свойствам объекта

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

// В данном случае связыватель модели с помощью рефлексии получает список всех открытых свойств модели (класса MyModel)

// И осуществляет поиск и установку значений для каждого свойства по очереди, используя стандартный приоритет поиска данных в запросе.

[HttpPost]

public ActionResult Index(MyModel model)

{

Debug.WriteLine("Prop1 = " + model.Prop1);

Debug.WriteLine("Prop2 = " + model.Prop2);

Debug.WriteLine("Prop3 = " + model.Prop3);

return View();

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@using (Html.BeginForm())

{

<input type="hidden" name="Prop1" value="100" /> <p>100</p>

<input type="hidden" name="Prop2" value="Hello Woeld" /> <p>Hello world</p>

<input type="hidden" name="Prop3" value="true" /> <p>true</p>

<input type="submit" name="submit" />

}

## Привязка нескольких объектов

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(MyModel model1, MyModel model2)

{

Debug.WriteLine("model1.Prop1 = " + model1.Prop1);

Debug.WriteLine("model1.Prop2 = " + model1.Prop2);

Debug.WriteLine("model2.Prop1 = " + model2.Prop1);

Debug.WriteLine("model2.Prop2 = " + model2.Prop2);

return View();

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@using (Html.BeginForm())

{

// name="model1.Prop1" - model1 это префикс, который будет использовать связыватель модели для того,

// чтобы понять, для какого параметра метода действия адресовано значение

<input type="hidden" name="model1.Prop1" value="100" /><p>100 (model1)</p>

<input type="hidden" name="model1.Prop2" value="Hello" /><p>Hello (model1)</p>

<input type="hidden" name="model2.Prop1" value="200" /><p>200 (model2)</p>

<input type="hidden" name="model2.Prop2" value="World" /><p>World (model2)</p>

<input type="submit" name="submit" value="submit" />

}

## Передача массива

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(int[] numbers)

{

for (int i = 0; i < numbers.Length; i++)

{

Debug.WriteLine(numbers[i]);

}

return View();

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@using (Html.BeginForm())

{

<input type="hidden" name="numbers" value="1" />

<input type="hidden" name="numbers" value="10" />

<input type="hidden" name="numbers" value="100" />

<input type="hidden" name="numbers" value="1000" />

<input type="hidden" name="numbers" value="10000" />

<input type="submit" name="submit" value="submit" />

}

## Выгрузка файла

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Upload(HttpPostedFileBase file)

{

string fileName = Guid.NewGuid().ToString();

string extension = Path.GetExtension(file.FileName);

fileName += extension;

List<string> extensions = new List<string>() { ".txt", ".png", ".jpg", ".pdf", ".zip" };

if (extensions.Contains(extension))

{

file.SaveAs(Server.MapPath("/Uploads/" + fileName));

ViewBag.Message = "Файл сохранен";

}

else

{

ViewBag.Message = "Ошибка. Допустимые расширения файлов - '.txt', '.png', '.jpg', '.pdf', '.zip'";

}

return View("Index");

}

View

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<h3>Выбрите файл для отправки на сервер</h3>

<form action="@Url.Action("Upload")" method="post" enctype="multipart/form-data">

Файл - <input type="file" name="file" />

<br />

<input type="submit" />

<p>

@ViewBag.Message

</p>

</form>

# ModelBinder

public object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)

{

// создаем новую или работаем с готовой моделью.

MyModel model = (MyModel)bindingContext.Model ?? new MyModel();

bool hasPrefix = bindingContext.ValueProvider.ContainsPrefix(bindingContext.ModelName);

string searchPrefix = hasPrefix ? bindingContext.ModelName + "." : "";

model.Prop1 = int.Parse(GetValue(bindingContext, searchPrefix, "Prop1"));

model.Prop2 = GetValue(bindingContext, searchPrefix, "Prop2");

return model;

}

private string GetValue(ModelBindingContext bindingContext, string prefix, string key)

{

ValueProviderResult vpr = bindingContext.ValueProvider.GetValue(prefix + key);

return vpr == null ? null : vpr.AttemptedValue;

}

}

Global.asax

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

ModelBinders.Binders.Add(typeof(MyModel), new MyModelBinder());

WebApiConfig.Register(GlobalConfiguration.Configuration);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

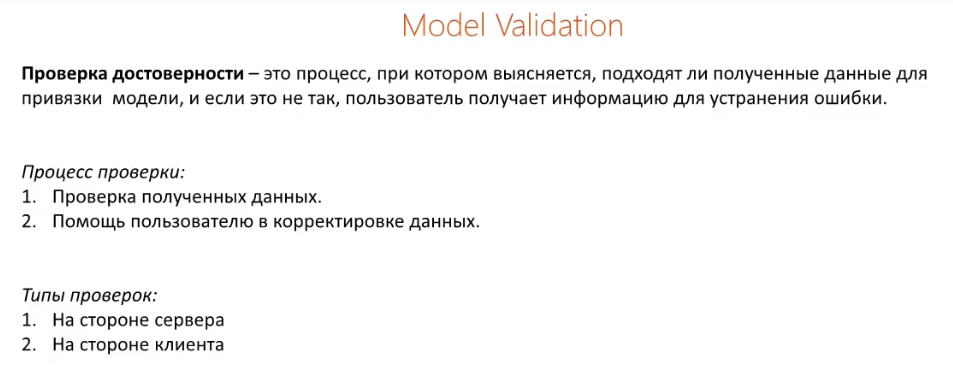
BundleConfig.RegisterBundles(BundleTable.Bundles);

}

}

}

# Валидация



## Валидация реестрации пользователя

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(AccountModel model)

{

if (string.IsNullOrEmpty(model.Login))

{

ModelState.AddModelError("Login", "Введите логин");

}

if (string.IsNullOrEmpty(model.Password))

{

ModelState.AddModelError("Password", "Введите пароль");

}

if (string.Compare(model.Password, model.PasswordConfirm) != 0)

{

ModelState.AddModelError("PasswordConfirm", "Пароли не совпадают");

}

if (model.Email != null && !new Regex(@"\b[a-z0-9.\_]+@[a-z0-9.-]+\.[a-z]{2,4}\b").IsMatch(model.Email))

{

ModelState.AddModelError("Email", "Email не правильный");

}

if(ModelState.IsValid)

{

// произвести сохранение модели

return View("Success");

}

else

{

return View();

}

}

}

Model

public class AccountModel

{

public string Login { get; set; } // обязательное значение

public string Password { get; set; } // Password и PasswordConfirm должны совпадать

public string PasswordConfirm { get; set; }

public string Email { get; set; } // должно быть валидным email адресом

}

Index.chtml

@model ExplicitDataValidation.Models.AccountModel

@{

ViewBag.Title = "Index";

}

<h2>Registration</h2>

@using (Html.BeginForm())

{

<p>

@Html.LabelFor(x => x.Login)

@Html.EditorFor(x => x.Login)

@Html.ValidationMessageFor(x => x.Login)

</p>

<p>

@Html.LabelFor(x => x.Password)

@Html.EditorFor(x => x.Password)

@Html.ValidationMessageFor(x => x.Password)

</p>

<p>

@Html.LabelFor(x => x.PasswordConfirm)

@Html.EditorFor(x => x.PasswordConfirm)

@Html.ValidationMessageFor(x => x.PasswordConfirm)

</p>

<p>

@Html.LabelFor(x => x.Email)

@Html.EditorFor(x => x.Email)

@Html.ValidationMessageFor(x => x.Email)

</p>

<input type="submit" value="Register" />

}

Css

.field-validation-error {

color: #f00;

}

.field-validation-valid {

display: none;

}

.input-validation-error {

border: 1px solid #f00;

background-color: #fee;

}

.validation-summary-errors {

font-weight: bold;

color: #f00;

}

.validation-summary-valid {

display: none;

}

## Ошибка уровня модели

Смотри сверху пример

if (model.Email != null && !new Regex(@"\b[a-z0-9.\_]+@[a-z0-9.-]+\.[a-z]{2,4}\b").IsMatch(model.Email))

{

ModelState.AddModelError("Email", "Email не правильный");

}

if (model.Login == model.Password)

{

// Добавление ошибки уровня модели.

ModelState.AddModelError("", "Пароль не может совпадать с логином");

}

if (ModelState.IsValid)

{

// произвести сохранение модели

return View("Success");

}

else

{

return View();

}

}

Index.chtml

<p>

@Html.LabelFor(x => x.Email)

@Html.EditorFor(x => x.Email)

@Html.ValidationMessageFor(x => x.Email)

</p>

@Html.ValidationSummary();

<input type="submit" value="Register" />

}

## Валидация с помощю атрибутов модели

using System.ComponentModel.DataAnnotations;

public class AccountModel

{

// Required - поле обязательное для заполнения

// StringLength - допустимая длина слова для поля ввода

// RegularExpression - регулярное выражение для проверки введенных данных

// Range - диапазон допустимых значений для поля ввода

// Compare - значения свойств должны иметь одинаковые значения

[Required(ErrorMessage = "Вы не ввели логин")]

[StringLength(10, ErrorMessage = "Логин не может привышать 10 символов")]

public string Login { get; set; }

[Required(ErrorMessage = "Поле пароль обязательно для заполнения")]

[StringLength(20, MinimumLength = 5, ErrorMessage = "Следует указать пароль от 5 до 20 символов")]

[Compare("PasswordConfirm", ErrorMessage = "Пароли не совпадают")]

public string Password { get; set; }

public string PasswordConfirm { get; set; }

[Required(ErrorMessage = "Поле Email обязательно для заполнения")]

[RegularExpression(@"(?i)\b[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b", ErrorMessage = "Email адрес указан не правильно")]

public string Email { get; set; }

[Required(ErrorMessage = "Поле Age обязательно для заполнения")]

[Range(18, 60, ErrorMessage = "Значение поля возраст должно попадать в диапазон от 18 до 60")]

public int Age { get; set; }

}

}

Controller

[HttpPost]

public ActionResult Index(AccountModel model)

{

if (ModelState.IsValid)

{

return View("Success");

}

return View();

}

View

@model MetadataValidation.Models.AccountModel

@{

ViewBag.Title = "Index";

}

<h2>Registration</h2>

@using (Html.BeginForm())

{

<p>

@Html.LabelFor(x => x.Login)

@Html.EditorFor(x => x.Login)

@Html.ValidationMessageFor(x => x.Login)

</p>

<p>

@Html.LabelFor(x => x.Password)

@Html.EditorFor(x => x.Password)

@Html.ValidationMessageFor(x => x.Password)

</p>

<p>

@Html.LabelFor(x => x.PasswordConfirm)

@Html.EditorFor(x => x.PasswordConfirm)

@Html.ValidationMessageFor(x => x.PasswordConfirm)

</p>

<p>

@Html.LabelFor(x => x.Email)

@Html.EditorFor(x => x.Email)

@Html.ValidationMessageFor(x => x.Email)

</p>

<p>

@Html.LabelFor(x => x.Age)

@Html.EditorFor(x => x.Age)

@Html.ValidationMessageFor(x => x.Age)

</p>

<input type="submit" value="Register" />

}

## Свой атрибут проверки

Model

[Required(ErrorMessage = "Поле Age обязательно для заполнения")]

[Range(18, 60, ErrorMessage = "Значение поля возраст должно попадать в диапазон от 18 до 60")]

public int Age { get; set; }

[MustBeTrue(ErrorMessage="Вы не согласились с условиями использования")]

public bool TermsAccepted { get; set; }

}

Класс в папке model

public class MustBeTrueAttribute : ValidationAttribute

{

// Если метод возвращает true - значение свойства допустимо.

// При значении false - возникнет ошибка на уровне свойства.

// Таким же способом можно создавать атрибуты для всей модели. В случае атрибута для всей модели,

// значения параметра value метода IsValid будет экземпляром модели, а не значением свойства, как в данном случае.

public override bool IsValid(object value)

{

// если тип проверяемого значения bool, а значение true - возвращаем true

return value is bool && (bool)value;

}

}

Index.chtml

<p>

@Html.LabelFor(x => x.Age)

@Html.EditorFor(x => x.Age)

@Html.ValidationMessageFor(x => x.Age)

</p>

<p>

@Html.LabelFor(x => x.TermsAccepted)

@Html.EditorFor(x => x.TermsAccepted)

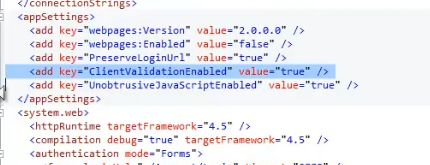
@Html.ValidationMessageFor(x => x.TermsAccepted)

</p>

<input type="submit" value="Register" />

}

## Валидация на стороне клиента



BundleConfig

public class BundleConfig

{

// For more information on Bundling, visit http://go.microsoft.com/fwlink/?LinkId=254725

public static void RegisterBundles(BundleCollection bundles)

{

bundles.Add(new ScriptBundle("~/bundles/jquery").Include(

"~/Scripts/jquery-{version}.js"));

bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(

"~/Scripts/jquery-ui-{version}.js"));

bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(

"~/Scripts/jquery.unobtrusive\*",

"~/Scripts/jquery.validate\*"));

Layout.chtml

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width" />

<title>@ViewBag.Title</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

@RenderBody()

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/jqueryval") @\*Подключение сценариев для валидации форм\*@

@RenderSection("scripts", required: false)

</body>

</html>

Index.chtml

@model ClientSideValidation.Models.AccountModel

@{

ViewBag.Title = "Index";

}

<h2>Registration</h2>

<p>

Для валидации данных на стороне клиента используется библиотека ~/Scripts/jquery.unobtrusive и ~/Scripts/jquery.validate

Данные библиотеки добавлены в файле компоновки через бандл ~/bundles/jqueryval</p>

@using (Html.BeginForm())

{

<p>

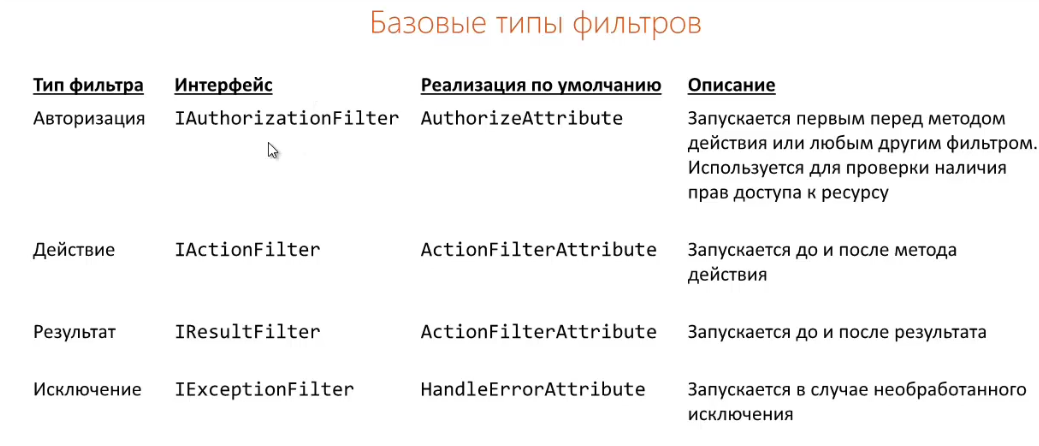
@Html.LabelFor(x => x.Login)

@Html.EditorFor(x => x.Login)

@Html.ValidationMessageFor(x => x.Login)

</p>

# Фильтры



## Exception filter



//

// GET: /Home/

public ActionResult Index()

{

return View();

}

// Для работы фильтра HandleError в файле web.config нужно добавить <customErrors mode="On"></customErrors>

// В случае возникновения исключения типа DivideByZeroException будет возвращено представление DivideError

[HandleError(ExceptionType = typeof(DivideByZeroException), View = "DivideError")]

public ActionResult Divide()

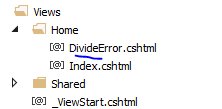
{

int a = 10, b = 0;

int result = a / b;

return Content(result.ToString());

}



Index.chtml

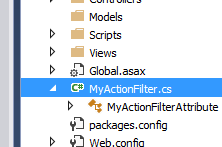
@{

ViewBag.Title = "Index";

}

@Html.ActionLink("Результат деления", "Divide")

## Собственный фильтер



// Атрибут, который реализует интерфейс IActionFilter, используется для декорирования действий в контроллере.

public class MyActionFilterAttribute : FilterAttribute, IActionFilter

{

// Метод запускается до выполнения кода действия.

public void OnActionExecuting(ActionExecutingContext filterContext)

{

// Проверка наличия значения в cookies

if (filterContext.HttpContext.Request.Cookies["test"] != null)

{

// Метод действия не выполняется, если в свойство Result записать значение отличное от null.

filterContext.Result = new HttpNotFoundResult();

}

}

// Метод запускается после выполнения кода действия.

public void OnActionExecuted(ActionExecutedContext filterContext)

{

}

}

Controller

// GET: /Home/

// Атрибут не дает запустить метод действия и возвращает ошибку 404

// если в cookie наборах запроса нет записи с именем test

[MyActionFilter]

public ActionResult Index()

{

return View();

}

## Замер скорости выполнения действия

public class ProfileAttribute : FilterAttribute, IActionFilter

{

private Stopwatch sw;

public void OnActionExecuting(ActionExecutingContext filterContext)

{

sw = Stopwatch.StartNew();

}

public void OnActionExecuted(ActionExecutedContext filterContext)

{

sw.Stop();

if (filterContext.Exception == null)

{

string message = string.Format("На выполнение метода действия {0} затрачено {1}",

filterContext.ActionDescriptor.ActionName, sw.Elapsed);

filterContext.HttpContext.Response.Write(message);

}

}

}

Controller

//

// GET: /Home/

[Profile]

public ActionResult Index()

{

return View();

}

[Profile]

public ActionResult SlowAction()

{

Thread.Sleep(2000);

return View();

}

Кеширование

//

// GET: /Home/

// OutputCache – указывает, что ответ нужно кэшировать на время указанное в параметре Duration

[OutputCache(Duration = 10)]

public ActionResult Index()

{

ViewBag.Message = "Операция выполнена в " + DateTime.Now.ToLongTimeString();

return View();

}

# Авторизация



## AccountModel

public class UsersContext : DbContext

{

public UsersContext()

: base("DefaultConnection")

{

}

public DbSet<UserProfile> UserProfiles { get; set; }

}

[Table("UserProfile")]

public class UserProfile

{

[Key]

[DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]

public int UserId { get; set; }

public string UserName { get; set; }

}

public class LocalPasswordModel

{

[Required]

[DataType(DataType.Password)]

[Display(Name = "Current password")]

public string OldPassword { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "New password")]

public string NewPassword { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm new password")]

[Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

}

public class LoginModel

{

[Required]

[Display(Name = "User name")]

public string UserName { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[Display(Name = "Remember me?")]

public bool RememberMe { get; set; }

}

public class RegisterModel

{

[Required]

[Display(Name = "User name")]

public string UserName { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm password")]

[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

}

}

## AccountCnotroller

[Authorize]

[InitializeSimpleMembershipAttribute]

public class AccountController : Controller

{

//

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

return View();

}

//

// POST: /Account/Login

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Login(LoginModel model, string returnUrl)

{

// WebSecurity.Login - аутентифицирует пользователя.

// Если логин и пароль введены правильно - метод возвращает значение true после чего выполняет добавление специальных значений в cookies.

if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))

{

return RedirectToLocal(returnUrl);

}

// Был введен не правильный логин или пароль

ModelState.AddModelError("", "The user name or password provided is incorrect.");

return View(model);

}

//

// POST: /Account/LogOff

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LogOff()

{

// Удаление специальных аутентификационных cookie значений

WebSecurity.Logout();

return RedirectToAction("Index", "Home");

}

//

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

return View();

}

//

// POST: /Account/Register

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Register(RegisterModel model)

{

if (ModelState.IsValid)

{

// Attempt to register the user

try

{

// Создание пользователя

WebSecurity.CreateUserAndAccount(model.UserName, model.Password);

// Аутентификация пользователя

WebSecurity.Login(model.UserName, model.Password);

return RedirectToAction("Index", "Home");

}

catch (MembershipCreateUserException e)

{

ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));

}

}

// If we got this far, something failed, redisplay form

return View(model);

}

//

// GET: /Account/Manage

public ActionResult Manage(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."

: message == ManageMessageId.SetPasswordSuccess ? "Your password has been set."

: message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."

: "";

ViewBag.HasLocalPassword = OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));

ViewBag.ReturnUrl = Url.Action("Manage");

return View();

}

//

// POST: /Account/Manage

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Manage(LocalPasswordModel model)

{

if (ModelState.IsValid)

{

bool changePasswordSucceeded;

try

{

// ChangePassword выбрасывает исключения в случае не удачной попытки смены пароля.

changePasswordSucceeded = WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword, model.NewPassword);

}

catch (Exception)

{

changePasswordSucceeded = false;

}

if (changePasswordSucceeded)

{

return RedirectToAction("Manage", new { Message = ManageMessageId.ChangePasswordSuccess });

}

else

{

ModelState.AddModelError("", "The current password is incorrect or the new password is invalid.");

}

}

// If we got this far, something failed, redisplay form

return View(model);

}

## SimpleMembershipPovider

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited = true)]

public sealed class InitializeSimpleMembershipAttribute : ActionFilterAttribute

{

private static SimpleMembershipInitializer \_initializer;

private static object \_initializerLock = new object();

private static bool \_isInitialized;

public override void OnActionExecuting(ActionExecutingContext filterContext)

{

// Инициализация ASP.NET Simple Membership происходит один раз при старте приложения

LazyInitializer.EnsureInitialized(ref \_initializer, ref \_isInitialized, ref \_initializerLock);

}

private class SimpleMembershipInitializer

{

public SimpleMembershipInitializer()

{

Database.SetInitializer<UsersContext>(null);

try

{

using (var context = new UsersContext())

{

if (!context.Database.Exists())

{

// Создание базы данных SimpleMembership без применения миграции Entity Framework

((IObjectContextAdapter)context).ObjectContext.CreateDatabase();

}

}

// Настройка ASP.NET Simple Membership

// 1 параметр - имя строки подключения к базе данных.

// 2 параметр - таблица, которая содержит информацию о пользователях

// 3 параметр - имя колонки в таблице, которая отвечает за хранение логина

// 4 параметр - autoCreateTables автоматическое создание таблиц если они не существуют в базе

WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTables: true);

SimpleRoleProvider roles = (SimpleRoleProvider)Roles.Provider;

SimpleMembershipProvider membership = (SimpleMembershipProvider)Membership.Provider;

// Проверка наличия роли Moderator

if (!roles.RoleExists("Moderator"))

{

roles.CreateRole("Moderator");

}

// Проверка наличия роли Admin

if (!roles.RoleExists("Admin"))

{

roles.CreateRole("Admin");

}

// Поиск пользователя с логином admin

if (membership.GetUser("admin", false) == null)

{

membership.CreateUserAndAccount("admin", "qwe123"); // создание пользователя

roles.AddUsersToRoles(new[] { "admin" }, new[] { "Admin" }); // установка роли для пользователя

}

if (membership.GetUser("user1", false) == null)

{

membership.CreateUserAndAccount("user1", "qwe123");

roles.AddUsersToRoles(new[] { "user1" }, new[] { "Moderator" });

}

}

catch (Exception ex)

{

throw new InvalidOperationException("The ASP.NET Simple Membership database could not be initialized. For more information, please see http://go.microsoft.com/fwlink/?LinkId=256588", ex);

}

}

}

## HomeController

public class HomeController : Controller

{

public ActionResult Index()

{

ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC application.";

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your app description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

[Authorize] // Запрещены анонимные обращения к данной странице

public ActionResult Cabinet()

{

ViewBag.Message = "Private Page.";

return View();

}

[Authorize(Roles = "Admin")] // К данному методу действия могут получать доступ только пользователи с ролью Admin

public ActionResult AdminPanel()

{

ViewBag.Message = "Admin Panel.";

return View();

}

[Authorize(Roles = "Admin, Moderator")] // К данному методу действия могут получать доступ только пользователи с ролью Admin и Moderator

public ActionResult ModeratorPanel()

{

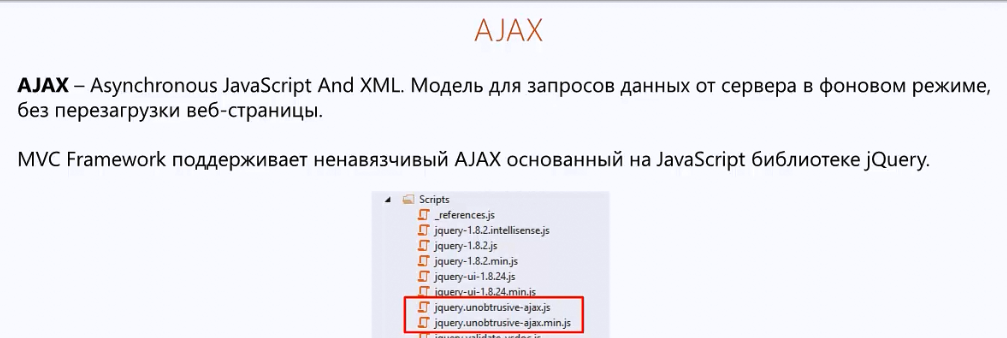
ViewBag.Message = "Moderator Panel.";

return View();

}

}

# Ajax



## Без аякса

public class Order

{

public int Id { get; set; }

public string Product { get; set; }

public string Customer { get; set; }

public int Quantity { get; set; }

}

class OrdersDatabase

{

public static IEnumerable<Order> Orders

{

get

{

Thread.Sleep(2000);

List<Order> list = new List<Order>();

list.Add(new Order() { Id = 1, Product = "Product 1", Customer = "Ivanov", Quantity = 1 });

list.Add(new Order() { Id = 2, Product = "Product 2", Customer = "Petrov", Quantity = 10 });

list.Add(new Order() { Id = 3, Product = "Product 2", Customer = "Fedorov", Quantity = 12 });

list.Add(new Order() { Id = 4, Product = "Product 3", Customer = "Fedorov", Quantity = 6 });

list.Add(new Order() { Id = 5, Product = "Product 1", Customer = "Petrov", Quantity = 7 });

list.Add(new Order() { Id = 6, Product = "Product 4", Customer = "Ivanov", Quantity = 11 });

list.Add(new Order() { Id = 7, Product = "Product 2", Customer = "Petrov", Quantity = 10 });

list.Add(new Order() { Id = 8, Product = "Product 5", Customer = "Petrov", Quantity = 2 });

list.Add(new Order() { Id = 9, Product = "Product 1", Customer = "Ivanov", Quantity = 11 });

list.Add(new Order() { Id = 10, Product = "Product 5", Customer = "Fedorov", Quantity = 3 });

list.Add(new Order() { Id = 11, Product = "Product 2", Customer = "Petrov", Quantity = 3 });

list.Add(new Order() { Id = 12, Product = "Product 1", Customer = "Ivanov", Quantity = 3 });

list.Add(new Order() { Id = 13, Product = "Product 4", Customer = "Petrov", Quantity = 7 });

return list;

}

}

Controller

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(string id)

{

// id - имя клиента, заказы которого необходимо выводить на странице.

return View("Index", (object)id);

}

public ActionResult OrdersData(string id)

{

var data = OrdersDatabase.Orders;

if (!string.IsNullOrEmpty(id) && id != "All")

{

// выполняем выборку по свойству Customer если значение id не пустое и не равное "All"

data = data.Where(e => e.Customer == id);

}

return PartialView(data);

}

}

Index.chtml

@{

ViewBag.Title = "Orders";

}

<h2>Orders</h2>

@using (Html.BeginForm())

{

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" />

</p>

}

## С аяксом

Layout

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width" />

<title>@ViewBag.Title</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

@RenderBody()

@Scripts.Render("~/bundles/jquery")

@RenderSection("scripts", required: false)

</body>

</html>

Index.chtml

@{

ViewBag.Title = "Orders";

// UpdateTargetId - элемент на странице, который будет обновляется после получения новых данных со стороны сервера.

// Url - адрес ресурса на сервере, который возвращает данные для обновления страницы

var options = new AjaxOptions()

{

UpdateTargetId = "tabledata",

Url = Url.Action("OrdersData")

};

}

<h2>Orders</h2>

@\*

Ajax.BeginForm(options) - при использовании данного метода, генерируется html элемент со специальными атрибутами.

Данные атрибуты будут использоваться библиотекой jquery.unobtrusive-ajax для того, чтобы определить форму, которая вместо

обычного postback запроса должна делать асинхронный запрос.

\*@

@using (Ajax.BeginForm(options))

{

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" />

</p>

}

@section scripts

{

@\*Ненавязивый JavaScript для выполнения запроса на сервер без полного обновления страницы\*@

<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>

}

## Индикатор загрузки

@{

ViewBag.Title = "Orders";

// LoadingElementId - элемент, который должен отображаться в процессе отправки и получения данных с сервера.

// Confirm - при наличии значения у данного свойства, перед отправкой запроса на сервер будет отображатся диалоговое окно с запросом.

var options = new AjaxOptions()

{

UpdateTargetId = "tabledata",

Url = Url.Action("OrdersData"),

LoadingElementId = "loadingIndicator",

Confirm = "Вы хотите запросить новые данные?"

};

}

<h2>Orders</h2>

@using (Ajax.BeginForm(options))

{

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" /> <img id="loadingIndicator" style="display:none;" src="@Url.Content("~/Content/loading.gif")" />

</p>

}

@section scripts

{

@\*Ненавязивый JavaScript для выполнения запроса на сервер без полного обновления страницы\*@

<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>

}

## Аякс ссылки

@{

ViewBag.Title = "Orders";

var options = new AjaxOptions()

{

UpdateTargetId = "tabledata",

Url = Url.Action("OrdersData"),

LoadingElementId = "loadingIndicator"

};

}

<h2>Orders</h2>

@using (Ajax.BeginForm(options))

{

<div id="loadingIndicator" style="display:none;">Loading...</div>

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" />

</p>

}

@\*Ajax.ActionLink - создание ссылки, клик по которой, будет обрабатываться кодом библиотеки jquery.unobtrusive-ajax\*@

@foreach (string str in new[] { "All", "Ivanov", "Petrov", "Fedorov" })

{

<div style="margin-right: 5px; float: left;">

@Ajax.ActionLink(str, "OrdersData", new { id = str },

new AjaxOptions

{

UpdateTargetId = "tabledata",

LoadingElementId = "loadingIndicator"

})

</div>

}

@section scripts

{

@\*Ненавязивый JavaScript для выполнения запроса на сервер без полного обновления страницы\*@

<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>

}

## Ajax callback function

@{

ViewBag.Title = "Orders";

// С помощью следующих свойств можно указать JavaScript функции, которые будут запускаться по мере выполнения запросов.

// OnBegin - начало асинхронного запроса на сервер.

// OnSuccess - успешное завершение обработки запроса на сервере.

// OnFailure - функция запустится в случае возникновения ошибки.

// OnComplete - функция запускается по завершению асинхронного запроса не зависимо от резальтата.

var options = new AjaxOptions()

{

UpdateTargetId = "tabledata",

Url = Url.Action("OrdersData"),

LoadingElementId = "loadingIndicator",

OnBegin = "OnBeginJs",

OnSuccess = "OnSuccessJs",

OnFailure = "OnFailureJs",

OnComplete = "OnCompleteJs",

};

}

<h2>Orders</h2>

@using (Ajax.BeginForm(options))

{

<div id="loadingIndicator" style="display:none;">Loading...</div>

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" />

</p>

}

@\*Ajax.ActionLink - создание ссылки, клик по которой, будет обрабатываться кодом библиотеки jquery.unobtrusive-ajax\*@

@foreach (string str in new[] { "All", "Ivanov", "Petrov", "Fedorov" })

{

<div style="margin-right: 5px; float: left;">

@Ajax.ActionLink(str, "OrdersData", new { id = str },

new AjaxOptions

{

UpdateTargetId = "tabledata",

LoadingElementId = "loadingIndicator"

})

</div>

}

<br />

@\*При нажатии на ссылку сервер вернет ответ 404\*@

@Ajax.ActionLink("ErrorLink", "OrdersDataError",

new AjaxOptions()

{

OnBegin = "OnBeginJs",

OnSuccess = "OnSuccessJs",

OnFailure = "OnFailureJs",

OnComplete = "OnCompleteJs",

})

@section scripts

{

@\*Ненавязивый JavaScript для выполнения запроса на сервер без полного обновления страницы\*@

<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>

<script type="text/javascript">

function OnBeginJs() {

alert("OnBegin Callback");

}

function OnSuccessJs(data) {

alert("OnSuccess Callback: " + data);

}

function OnFailureJs(request, error) {

alert("OnFailure Callback: " + error);

}

function OnCompleteJs(request, status) {

alert("OnComplete Callback: " + status);

}

</script>

}

## Json данные

Controller

public ActionResult JsonOrdersData(string id)

{

var data = OrdersDatabase.Orders;

if (!string.IsNullOrEmpty(id) && id != "All")

{

data = data.Where(e => e.Customer == id);

}

return Json(data, JsonRequestBehavior.AllowGet);

}

}

Index.chtml

@{

ViewBag.Title = "Orders";

var options = new AjaxOptions()

{

UpdateTargetId = "tabledata",

Url = Url.Action("JsonOrdersData"),

LoadingElementId = "loadingIndicator",

OnSuccess = "ParseResponse",

};

}

<h2>Orders</h2>

@using (Ajax.BeginForm(options))

{

<div id="loadingIndicator" style="display:none;">Loading...</div>

<table cellpadding="4">

<thead>

<tr>

<th>Id</th>

<th>Product</th>

<th>Quantity</th>

<th>Customer</th>

</tr>

</thead>

<tbody id="tabledata">

@Html.Action("OrdersData", new { id = Model })

</tbody>

</table>

<p>

@\*При отправке обратного запроса на сервер, в форме будет содержаться поле id которое методом действия будет использоваться для фильтрации данных\*@

@Html.DropDownList("id", new SelectList(new[] { "All", "Ivanov", "Petrov", "Fedorov" }, (Model ?? "All")))

<input type="submit" value="Submit" />

</p>

}

@section scripts

{

@\*Ненавязивый JavaScript для выполнения запроса на сервер без полного обновления страницы\*@

<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>

<script type="text/javascript">

function ParseResponse(data) {

var target = $("#tabledata");

target.empty();

for (var i = 0; i < data.length; i++) {

target.append("<tr><td>" + data[i].Id + "</td><td>"

+ data[i].Product + "</td><td>"

+ data[i].Quantity + "</td><td>"

+ data[i].Customer + "</td></tr>");

}

}

</script>

}

## Jquery

Layout

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width" />

<title>@ViewBag.Title</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

@RenderBody()

@Scripts.Render("~/bundles/jquery")

@RenderSection("scripts", required: false)

</body>

</html>

Index.chtml

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<button id="btn1">Articele 1</button>

<button id="btn2">Articele 2</button>

<div id="articleContent"></div>

@section scripts

{

<script>

$(function () {

$("#btn1").click(function () {

makeRequest(1);

});

$("#btn2").click(function () {

makeRequest(2);

});

function makeRequest(id) {

// Отправка асинхронного запроса на сервер.

// url - адрес ресурса, которому направляется запрос.

// type - HTTP метод используемый для отправки запроса.

// data - данные передаваемые на сервер.

// success - функция, которая запустится при успешной обработке запроса.

$.ajax({

url: "@Url.Action("GetArticle")",

type: "GET",

data: "id=" + id,

success: function (response) {

$("#articleContent").text(response);

},

});

}

});

</script>

}

Controller

public class HomeController : Controller

{

Dictionary<string, string> articles = new Dictionary<string, string>();

string article1 = "ARTICLE1 Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.";

string article2 = "ARTICLE2 Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel. Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.";

public HomeController()

{

articles.Add("1", article1);

articles.Add("2", article2);

}

//

// GET: /Home/

public ActionResult Index()

{

return View();

}

public ActionResult GetArticle(string id)

{

if (!string.IsNullOrEmpty(id) && articles.ContainsKey(id))

{

return Content(articles[id]);

}

return new EmptyResult();

}

}

## Безопасность

