Генерация исходящих URL. Создание URL для ссылок в HTML разметке.

При построении представления использовать разметку <a href=”имя_контроллера/имя_метода”> не рекомендуется, так как при изменении настроек системы маршрутизации такие ссылки перестанут работать. Для того, чтобы значение атрибута href всегда правильно ссылалось на контроллер и метод действия в соответствии с текущими настройками маршрутизации, используется метод ***ActionLink*** или ***RouteLink*** класса ***HtmlHelper***.

```
@Html.ActionLink("Создать запись о клиенте", "Create");
@Html.ActionLink("Создать запись о клиенте", "Create", "Customer");

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


@Html.RouteLink("Создать запись о клиенте", new { controller = "Customer", action = "Create" });
```

Рассмотрим кнопку Bootstrap

```
<div>
    <a href="#" class="btn btn-success btn-lg">
        <span class="glyphicon glyphicon-print"></span> Print
    </a>
</div>
```

Чтобы указать контроллер, метод действия и параметр – используем

```
@*Генерация URL без окружающей HTML разметки    Адрес действия "распечатать данные о клиенте" – *@
@Url.Action("Print", "Cusomer", new { id = "customer-name" })
```

Цель урока – Изучить основы синтаксиса Razor. Научиться пользоваться компоновками, частичными представлениями и дочерними действиями.)

Объявление переменной во вью

```
@{
    var messageRasor = @"<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  asddasdasd pretium tellus.  </ p > ";
}
@*Выражения (HTML Encoded)*@
       <div>
           @messageRasor
       </div>
       <div>
          @Html.Raw(messageRasor)
       </div>
```

Компоновка – файл определяющий общий шаблон разметки веб приложения. Путь компоновки по умолчанию находится – Views/Shared/_Layout.cshtml
Методы для визуализации разметки представлений в файле компоновки –

- **RenderBody** – помещает в компоновку всю размету текущего представления без учета той разметки которая находится в отдельных секциях.
- **RenderSection** – метод, помещает в компоновку разметку определенной секции из текущего представления.

```
@*Визуализация содержимого секции с именем Footer*@
@RenderSection("Footer", false)
```

Второй параметр – обязательна ли секция во вью ! Если параметр равен `true`, то при отсутствии секции будет сгенерировано исключение

Действия (Action Methods)

В контроллере определяются методы, которые по умолчанию являются «методами действия». Контроллеры могут иметь много методов.

Определение публичных методов как NonAction

По умолчанию фреймворк MVC воспринимает все публичные методы как action методы. Если ваш класс контроллера содержит публичный метод и вы не хотите, чтобы он был так воспринят, вы должны определить его с атрибутом NonActionAttribute.

```
[NonAction]
private void DoSomething()
{
   // Method logic.
}
```

Методы обычно связываются с пользовательским действием связью один-к-одному. Пример пользовательского действия: пользователь вводит URL в браузер, нажимает ссылку и отправляет форму. Каждое из этих действий отправляет запрос на сервер. В каждом случае URL запроса содержит информацию, которую фреймворк использует для вызова метода.

Возвращаемые типы результатов

Большинство методов возвращаются экземпляр класса, наследуемого от **ActionResult**, класса, являющегося базовым для всех результатов методов. Однако существуют другие типы результатов, которые можно использовать в зависимости от задачи метода. Например, самый популярный метод – вызов метода View. Метод View возвращает экземпляр класса ViewResult, наследуемый от ActionResult.

Можно создавать методы, возвращаемые объект любого типа (строку, число, булево). Эти типы будут обернуты в подходящий тип ActionResult перед тем, как будет сгенерированы в ответное действие.

Action Result	Метод	Описание
ViewResult	View	Генерирует представление как веб-страницу.
PartialViewResult	PartialView	Генерирует частичное представление, определяемое часть представления, которая будет сгенерирована внутри другого представления.
RedirectResult	Redirect	Перенаправление на другой метод с указанием его URL.
RedirectToRouteResult	RedirectToAction RedirectToRoute	Перенаправление на другой метод.
ContentResult	Content	Возвращение определенного пользователем типа контента.
JsonResult	Json	Возвращение сериализованного объекта JSON.
JavaScriptResult	JavaScript	Возвращение скрипта, выполняемого на стороне клиента.
FileResult	File	Возвращение бинарного потока.
EmptyResult	(None)	Возвращение результата в том случае, если метод возвращает null (void).

PartialView

Частичные представления –это отдельные файлы, содержащие фрагменты разметки, которые могут быть включены в другие представления.

Визуализация частичного представления не вызывает метода действия.

Частичное представление – не задается layout page. Создадим два частичных представления

```
public ActionResult Partial()
{
    return PartialView();
}
public PartialViewResult GetDate()//так наглядней. ActionResult базовый класс
{
    var date = DateTime.Now;
    return PartialView(date);
}
```

Чтобы отобразить их в другом представлении используются методы

```
@Html.Partial("GetDate", DateTime.Now)
```

```
<br />
@{Html.RenderPartial("Partial");}
<br />
```

Отличие! Метод Partial возвращает MvcHtmlString, тогда как RenderPartial возвращает void и обрабатывается прямо во вью!
Без приставки Render Html.Partial (как и Html.Action) - возвращает html разметку. Разметка хранится в буфере, а потом также встраивается в выходной поток. С приставкой `Render` - немного эффективнее.
Также частичное представление можно вызвать с помощью `@Html.Action!`

```
<div>
    @Html.Action("GetDate", new { id = DateTime.Now.ToString() })
    <br />
    @{Html.RenderAction("GetDate", new { id = DateTime.Now.ToString() });}
    <br />
</div>
```

EmptyResult

Иногда надо НИЧЕГО не возвращать… `return new EmptyResult();`

FileResult

Иногда надо вернуть файл определенного формата… Например

```
public FilePathResult FileShow(string filename, string contenttype)
{
    string path = Server.MapPath("~");
    string name = string.Format(@"Files\{0}", filename);

    string filepath = Path.Combine(path, name);
    return File(filepath, contenttype);
}
```

Второй параметр – это content Type файла. Например, для изображений "image/png" , "image/jpeg"

Типизированные представления

Добавим модель

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 { set; get; }

}


public class ProductController : Controller
{
    Product product = new Product()
    {
        ProductID = 1,
        Name = "Notebook HP 2011",
        Category = "Notebook",
        Description = "OS Windows 8, I5 5000, 8Gb memory",
        Price = 11500M
    };
    public ActionResult Index()
    {
        ViewBag.ProductCount = 1;
        ViewBag.ExpressShip = true;
        ViewBag.ApplyDiscount = false;
        ViewBag.Supplier = null;
```

```
            return View(product);
        }
    }
```

Views/Product/Index.cshtml

```
@model WebAppRazor.Models.Product

@{
    ViewBag.Title = "Index";
}
<h2>This Page About @Model.Name</h2>
<p>
    Price this model @Model.Price
</p>
<div data-discount="@ViewBag.ApplyDiscount" data-express="@ViewBag.ExpressShip" data-
supplier="@ViewBag.Supplier">
    The containing element has data attributes
</div>
Discount:<input type="checkbox" checked="@ViewBag.ApplyDiscount" />
Express:<input type="checkbox" checked="@ViewBag.ExpressShip" />
Supplier:<input type="checkbox" checked="@ViewBag.Supplier" />
```

Установка значений атрибутов **data** элемента div – полезная вещь!!!!
data-атрибуты, которые являются атрибутами, чьи имена начинаются с data- неформальный способ создания пользовательских атрибутов в HTML5. Мы используем значения свойств ViewBag ApplyDiscount, ExpressShip и Supplier, чтобы установить значения этих атрибутов.
Затем можем получить эти значения в jQuery …

Установка значений атрибутов data элемента div – полезная вещь!!!!
data-атрибуты, которые являются атрибутами, чьи имена начинаются с data- неформальный способ создания пользовательских атрибутов в HTML5. Мы используем значения свойств ViewBag ApplyDiscount, ExpressShip и Supplier, чтобы установить значения этих атрибутов.
Затем можем получить эти значения в jQuery …

Использование условных операторов

```
    @switch ((int)ViewBag.ProductCount)
    {
        case 0:
            @: Out of Stock
            break;
        case 1:
            <b>Low Stock (@ViewBag.ProductCount)</b>
             break;
        default:
            @ViewBag.ProductCount
            break;
    }
```

Если вы хотите вставить в представление просто текст, когда он не содержится в HTML элементе, вы должны помочь Razor и обозначить строку следующим образом:

```
...
@: Out of Stock
...
```

Символ **@:** предотвращения Razor от интерпретации строки как C# выражения, что является поведением по умолчанию, когда он сталкивается с текстом.

```
    @if (ViewBag.ProductCount == 0)
    {
        @:Товаров нет
    }
    else if (ViewBag.ProductCount == 1)
    {
        <b>Low Stock (@ViewBag.ProductCount)</b>
    }
    else
    {
        @ViewBag.ProductCount
    }
```

Домашнее задание

```
class Employee
 {
     public int EmployeeId { get; set; }
     public string FirstName { get; set; }
     public string LastName { get; set; }
     public System.DateTime DateBirthday { get; set; }
     public System.DateTime HireDate { get; set; }//дата приема на работу

     public string INN { get; set; }
 }
```

Создать контроллер Home, Employee.
Добавить кнопки навигации между страницами.

Создать представление List – список (5 сотрудников) и частичное представление Get(int id), которое отображает информацию об одном сотруднике.