Bootstrap в ASP.NET MVC 5

Введение в Bootstrap

Bootstrap представляет собой css-фреймворк для создания адаптивных веб-приложений. Проект Bootstrap изначально был создан и развивался разработчиками из Twitter для собственных нужд, однако затем вышел за границы твиттера и в настоящее время развивается как opensource-проект и является одним из самых популярных фреймворков для создания веб-приложений. Что в принципе и неудивительно: в настоящее время все растет количество пользователей мобильного интернета, все больше людей для веб-серфинга используют смартфоны, планшеты. Что в итоге привело к увеличению числа сайтов с адаптивным дизайном и соответственно к увеличению популярности средств для разработки подобных сайтов, в том числе фреймворка Bootstrap.

Проекты ASP.NET MVC 5 по умолчанию уже содержат все необходимые файлы Bootstrap:

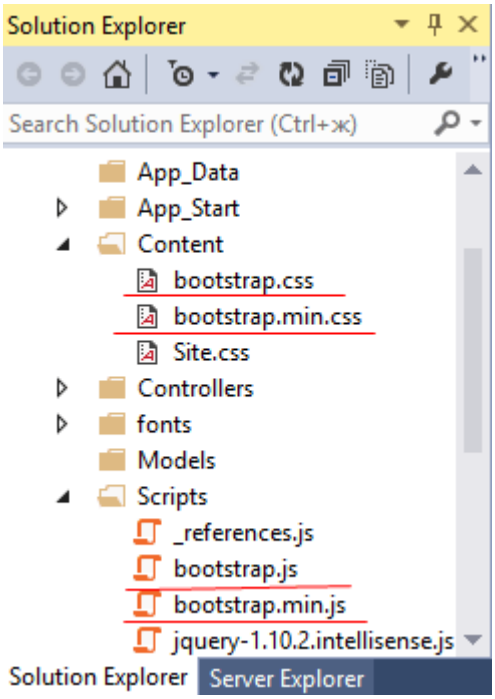

Рисунок 1 – Проект ASP.NET MVC 5

Функциональность Bootstrap заключена в файле стилей bootstrap.css и скрипте bootstrap.js. И если мы откроем мастер-страницу _Layout.cshtml, которая идет по умолчанию в проекте, то мы увидим подключение соответствующих библиотек:

```
<!DOCTYPE html>
<html>
<head>
    <metacharset="utf-8"/>
    <metaname="viewport"content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <divclass="navbarnavbar-inverse navbar-fixed-top">
        <divclass="container">
            <divclass="navbar-header">
                <buttontype="button"class="navbar-toggle"data-
toggle="collapse"data-target=".navbar-collapse">
                    <spanclass="icon-bar"></span>
                    <spanclass="icon-bar"></span>
                    <spanclass="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", null,
new { @class = "navbar-brand" })
            </div>
            <divclass="navbar-collapse collapse">
                <ulclass="navnavbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
            </div>
        </div>
    </div>
    <divclass="container body-content">
        @RenderBody()
        <hr/>
        <footer>
            <p>© @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Выражение @Styles.Render("~/Content/css") подключает необходимые стили, в том числе и bootstrap, а выражение @Scripts.Render("~/bundles/bootstrap") подключает скрипт bootstrap. Мы можем это увидеть, открыв файле BundleConfig.cs:

```
bundles.Add(newScriptBundle("~/bundles/bootstrap").Include(
            "~/Scripts/bootstrap.js",
            "~/Scripts/respond.js"));

bundles. Add(newStyleBundle("~/Content/css").Include(
            "~/Content/bootstrap.css",
                  "~/Content/site.css"));
```

Также на мастер-странице мы можем увидеть использование таких классов css, как navbar, navbar-inverse, navbar-fixed-topит.д. - все это классыbootstrap, которыепозволяютадаптироватьинтерфейсстраницыкразличнымустройствам. Например, запустим страницу в веб-браузере на ПК:

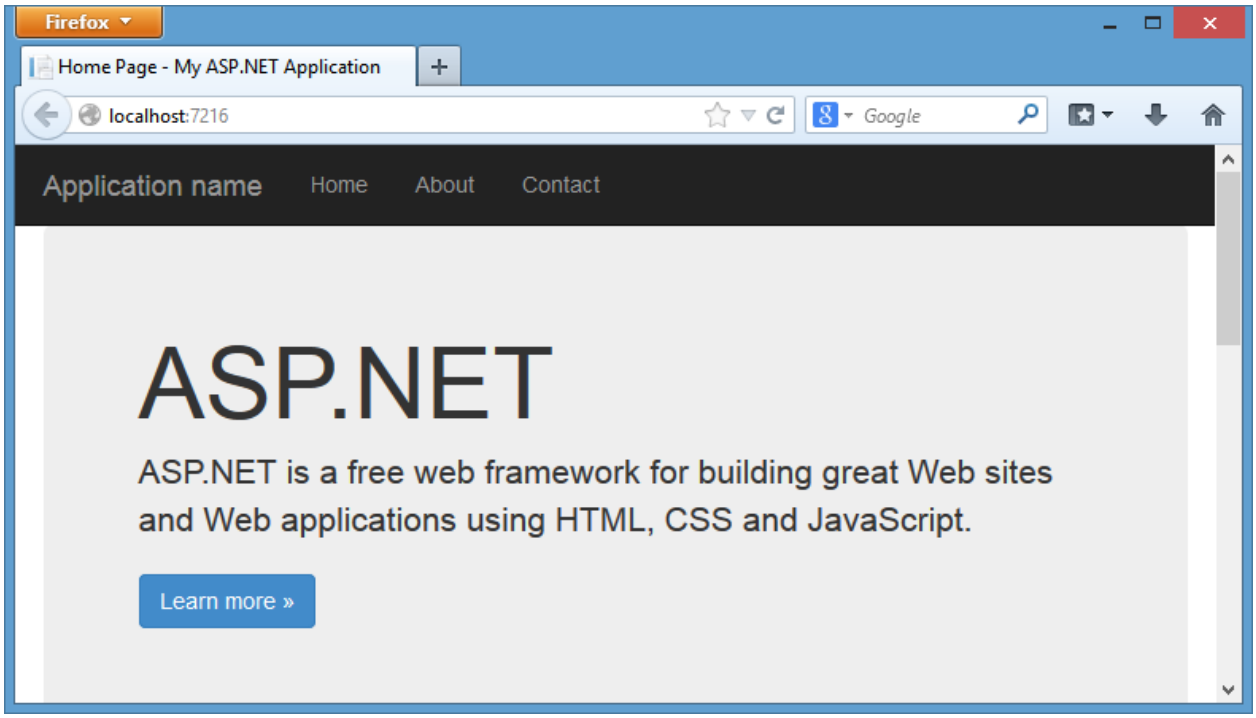

Рисунок 2 – Пример первого запуска приложения

И если мы также запустим данное приложение на мобильном телефоне или эмуляторе, то мы увидим, что интерфейс по-прежнему выглядит довольно неплохо:

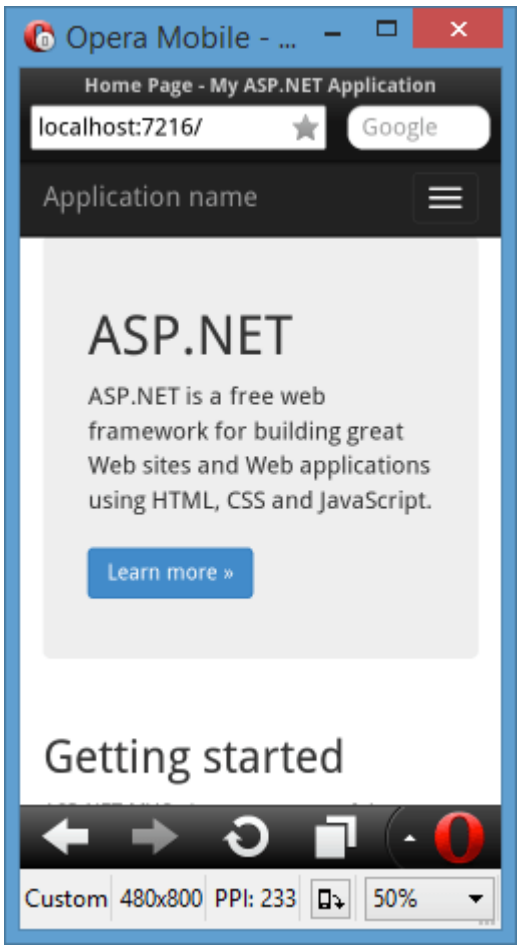

Рисунок 3 –Запуск приложения на эмуляторе

Собственно, в этом и преимущество использования Bootstrap. Основой для адаптивного дизайна стали правила mediaqueries, которые описывают стилевые свойства элементов для различных разрешений экрана. С точки зрения разработчиков Bootstrap все устройства можно поделить на следующие группы:

Очень маленькие, ширина экрана которых меньше 768 пикселей

Небольшие - их ширина от 768 пикселей и выше

Средние с шириной от 992 пикселя и выше

И большие с шириной от 1200 пикселей и выше

Это деление мы можем увидеть, если откроем файл bootstrap.css и найдем там правила media, например:

```
@media (min-width: 768px) {
  ..................
}
```

Позиционирование элементов

При использовании Bootstrap очень удобно позиционировать интерфейс в виде таблицы или сетки, используя строки и столбцы. Так, если мы посмотрим в браузере на стандартное представление Index.cshtml, которое идет по умолчанию, то мы увидим подобное позиционирование: элементы под заголовком расположены как бы в три столбца и составляют одну строку:

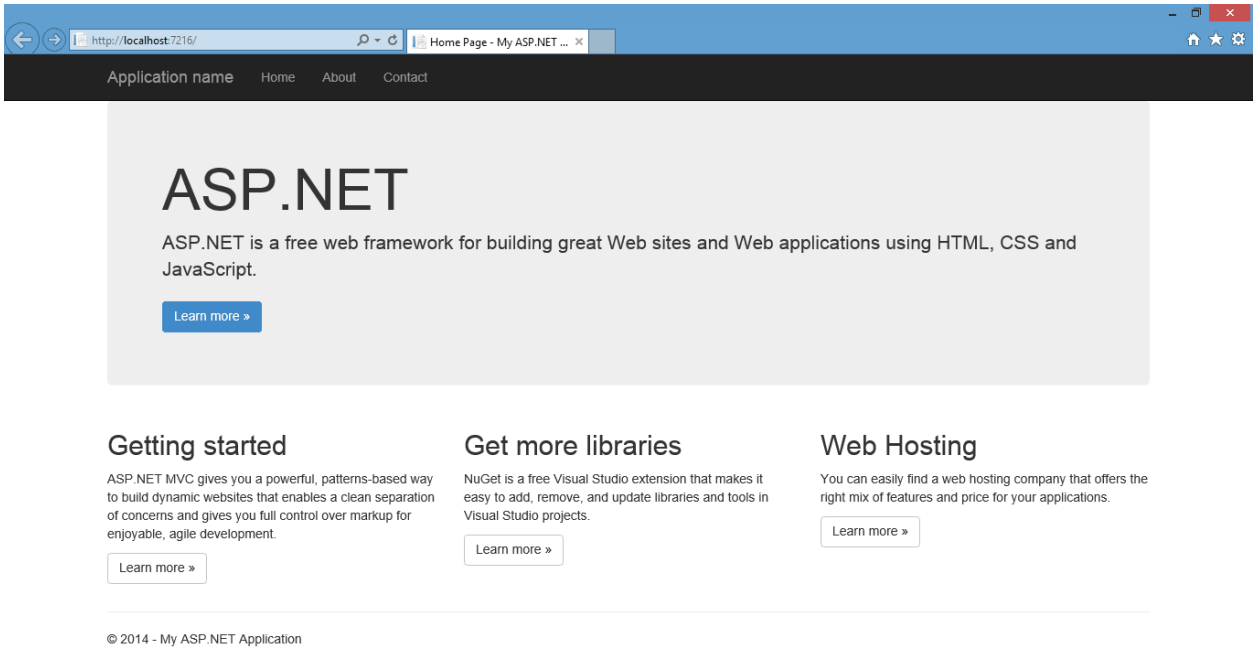

Рисунок 4 – Позиционирование элементов по столбцам

Если мы откроем код представления, то мы можем увидеть использование классов bootstrap:

```
<divclass="row">
    <divclass="col-md-4">
        <h2>Getting started</h2>
        <p>.............</p>
    </div>
    <divclass="col-md-4">
        <h2>Get more libraries</h2>
        <p>.............</p>
    </div>
    <divclass="col-md-4">
        <h2>Web Hosting</h2>
        <p>.............</p>
    </div>
</div>
```

Класс row задает расположение отдельных блоков в виде одной строки. Строка в Bootstrap может иметь до 12 столбцов. Строк может быть сколько угодно, но в данном случае у нас одна строка.

Для создания отдельного столбца строки используется класс `col-md-4`. `col`, как ясно из названия, обозначает столбец.

Дальше идет `md` - идентификатор устройства. Как мы увидели в прошлой теме, bootstrap делит все устройства условно на четыре группы в зависимости от ширины экрана. md, в частности, соотносится со средними устройствами (то есть которые имеют ширину от 992 пикселя и выше). А число 4 указывает, сколько условных единиц в строке будет занимать данный блок. Таким образом, получается, что класс `col-md-4` означает, что данный блок будет занимать 4 условных единицы из 12 в строке, то есть треть ширины экрана устройства с экраном шириной от 992 пикселей.

И поскольку у нас есть четыре группы, то для каждой группы имеются свои классы. Например, для очень маленьких устройств с экраном меньше 768 пикселей (то есть мобильных телефонов), подобный класс мог бы быть таким col-xs-4.

Все типы классов:

- col-xs-*: для устройств с шириной экрана меньше 768 пикселей;
- col-sm-*: для устройств с шириной экрана от 768 пикселей и выше;
- col-md-*: для устройств с шириной экрана от 992 пикселя и выше;
- col-lg-*: для устройств с шириной экрана от 1200 пикселей и выше.

Хотя даже на мобильных устройствах блок с классом `col-md-4` будет выглядеть вполне неплохо, но мы можем установить сразу два класса, чтобы еще больше детализировать отображение на различных устройствах, например:

```
<div class="col-xs-12 col-md-4">
```

И хотя в представлении Index.cshtml по умолчанию все три блока имеют относительную ширину в 4 единицы, составляя в целом 12 единиц, мы можем задать любую другую ширину, позиционируя их по собственному усмотрению. Например:

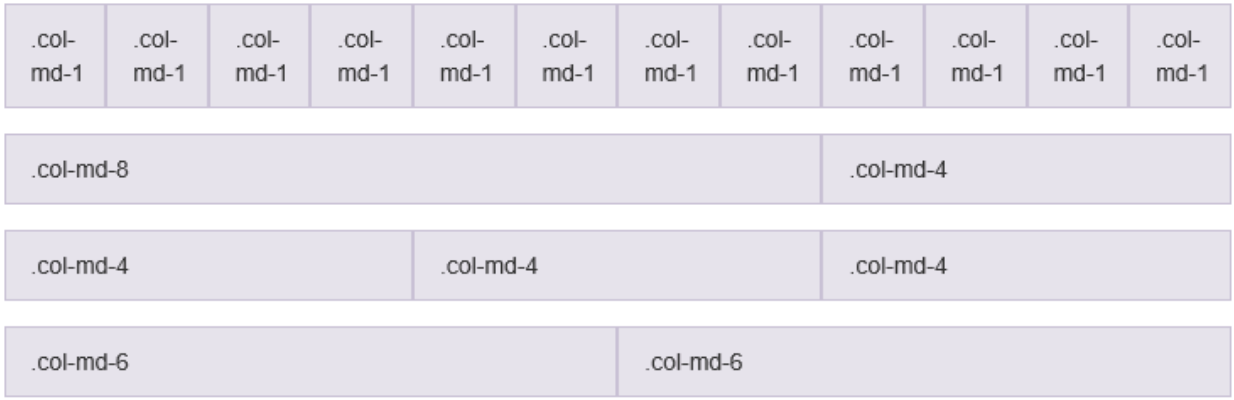

Рисунок 5 –

Отступы

Специальные классы `col-md(sm|lg)-offset-*` позволяют задать смещение относительно левого блока или начала строки в условных единицах. Например, у нас есть такая строка:

```
<divclass="row">
    <divclass="col-md-4">
        <h2>Левый блок</h2>
        <p>.................</p>
    </div>
    <divclass="col-md-4 col-md-offset-4">
        <h2>Правый блок</h2>
        <p>.................</p>
    </div>
</div>
```

Класс `col-md-offset-4` будет смещать правый блок на 4 условных единицы вправо:

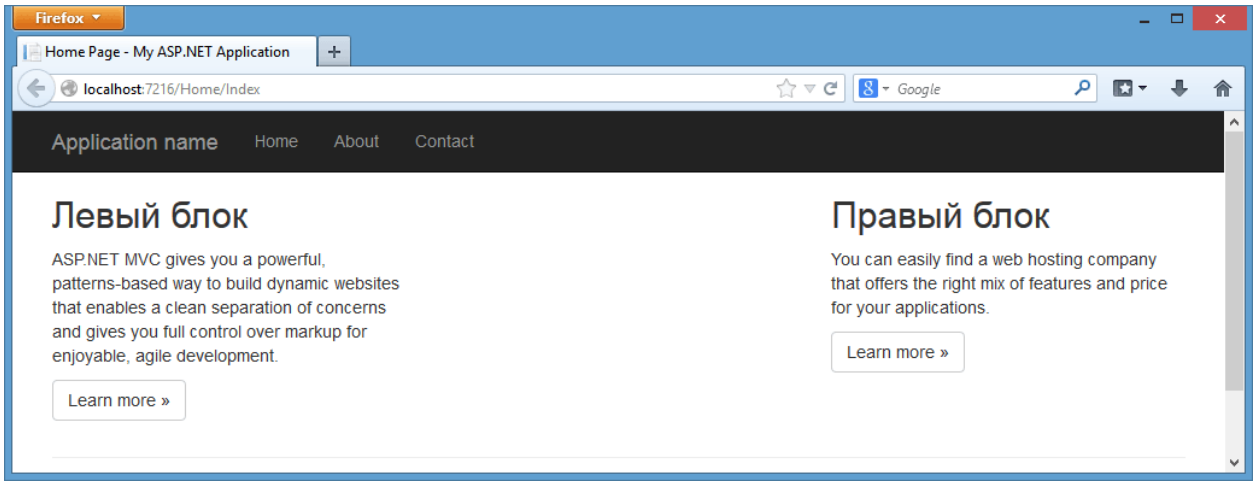

Рисунок 6 – Результат применения класса col-md-offset-4

Некоторые примеры по использованию смещения:

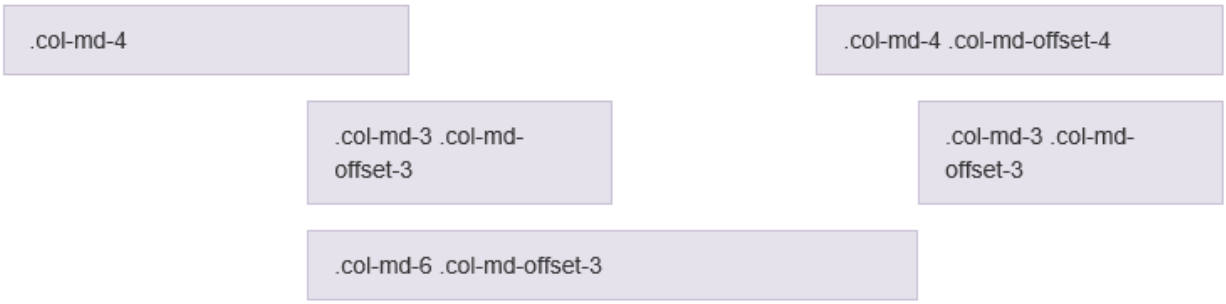

Рисунок 7 – Пример использования смещений

Порядок столбцов

С помощью классов `col-md(sm|lg)-push-*` и `col-md(sm|lg)-pull-*` мы можем переопределить порядок блоков в строке. Например, класс `col-md-push-4` сдвигает блок на четыре единицы вправо от текущего положения. И наоборот, класс `col-md-pull-4` сдвигает блок влево от текущей позиции. Так, предыдущий пример со смещением мы могли бы переписать следующим образом:

```
<divclass="row">
    <divclass="col-md-4">
        <h2>Левый блок</h2>
        <p>.................</p>
    </div>
    <divclass="col-md-4 col-md-push-4">
        <h2>Правый блок</h2>
        <p>.................</p>
```

```
    </div>
</div>
```

И у нас был бы тот же эффект. Но мы можем также полностью переупорядочить порядок следования блоков:

```
<divclass="row">
    <divclass="col-md-4 col-md-push-8">
        <h2>Левый блок</h2>
        <p>.................</p>
    </div>
    <divclass="col-md-4 col-md-pull-4">
        <h2>Правый блок</h2>
        <p>.................</p>
    </div>
</div>
```

Теперь правый блок сместится влево на четыре единицы, а левый блок - вправо на 8 единиц:

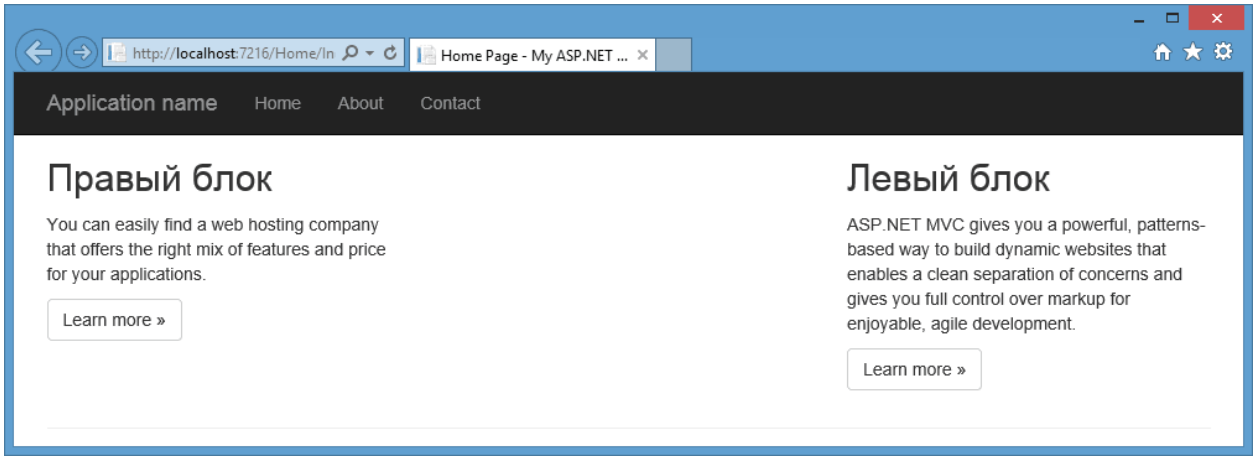

Рисунок 8 – Результат смещения блоков

Таким образом, используя класс row и классы столбцов, мы можем задать общее расположение элементов, а адаптивность bootstrap гарантирует, что на любых устройствах подобная сетка элементов будет выглядеть вполне нормально. Теперь рассмотрим некоторые компоненты, которые предлагает нам Bootstrap.

Компоненты Bootstrap

Bootstrap имеет ряд компонентов, которые не сводятся к стандартным кнопкам или текстовым полям, а представляют более сложные элементы.

Полный список компонентов Bootstrap можно найти на официальной странице проекта http://getbootstrap.com/components/. Рассмотрим вкратце некоторые из них.

Навигационные панели

Первый компонент Bootstrap, с которым мы сталкиваемся в проекте, это навигационная панель. Данный компонент располагается на мастер-странице _Layout:

```
<divclass="navbarnavbar-inversenavbar-fixed-top">
        <divclass="container">
            <divclass="navbar-header">
                <buttontype="button"class="navbar-toggle"data-
toggle="collapse"data-target=".navbar-collapse">
                    <spanclass="icon-bar"></span>
                    <spanclass="icon-bar"></span>
                    <spanclass="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", null,
new { @class = "navbar-brand" })
            </div>
            <divclass="navbar-collapse collapse">
                <ulclass="navnavbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
            </div>
        </div>
    </div>
```

И даже если мы сузим границы веб-браузера или запустим сайт на мобильном устройстве, то мы увидим, что панель навигации остается достаточно функциональной:

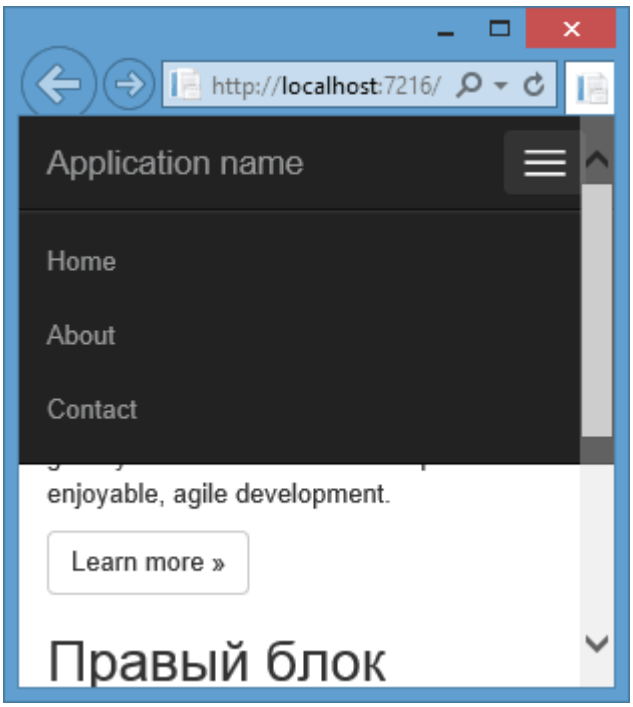

Рисунок 9 - Панель навигации

Компонент представляет класс navbar. Чтобы панель была фиксирована по верху страницы, используется также класс navbar-fixed-top. Если бы мы, наоборот, захотели фиксировать панель по низу, то в этом случае использовали бы класс navbar-fixed-bottom

И также в объявлении блока навигации используется класс navbar-inverse, который инвертирует цвета по умолчанию. Вместо этого класса мы могли бы использовать navbar-default, тогда в этом случае у нас бы было меню стандартных светлых тонов.

Для создания ссылок навигации применяется класс nav. Bootstrap представляет несколько классов для оформления ссылок навигации. По умолчанию используется класс navbar-nav, но мы можем использовать и другие возможности.

Для создания навигации по типу вкладок применяется класс nav-tabs. Так, например следующее меню:

```
<ulclass="navnav-tabs">
    <liclass="active">@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
```

даст такой эффект:

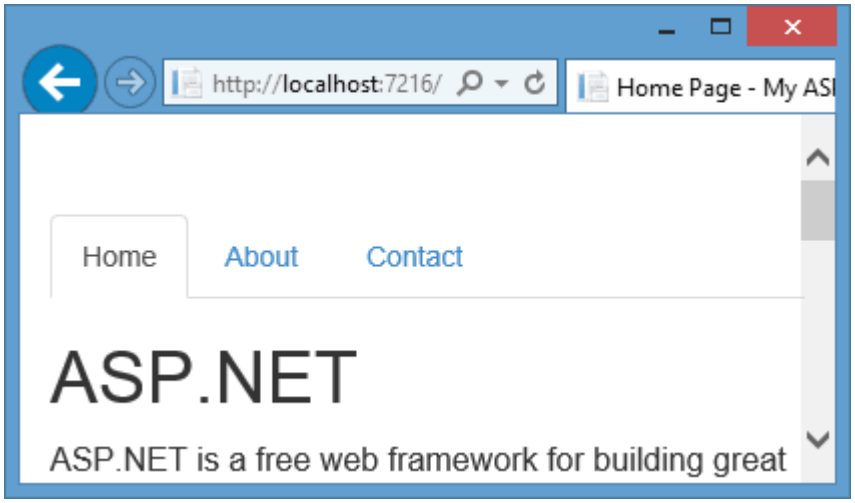

Рисунок 10 - Применение класса nav-tabs

Еще один вариант создания панели ссылок представляет класс nav-pills:

```
<ulclass="navnav-pills">
    <liclass="active">@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
```

Рисунок 11 - Применение класса nav-pills

А комбинируя классы nav-pills и nav-stacked, мы можем создать вертикальное меню:

```
<ulclass="navnav-pills nav-stacked">
    ..............................
```

```
</ul>
```

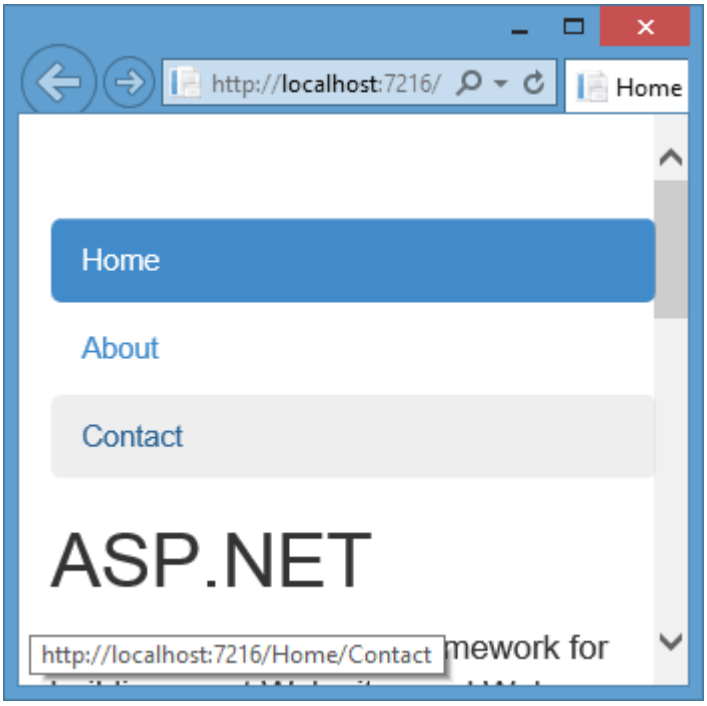

Рисунок 12– Применение класса nav-pills и nav-stacked

Пагинация

Класс pagination позволяет создать панель ссылок в виде постраничной навигации:

```
<ulclass="pagination">
  <li><ahref="#">«</a></li>
  <li><ahref="#">1</a></li>
  <li><ahref="#">2</a></li>
  <li><ahref="#">3</a></li>
  <li><ahref="#">4</a></li>
  <li><ahref="#">5</a></li>
  <li><ahref="#">»</a></li>
</ul>
```

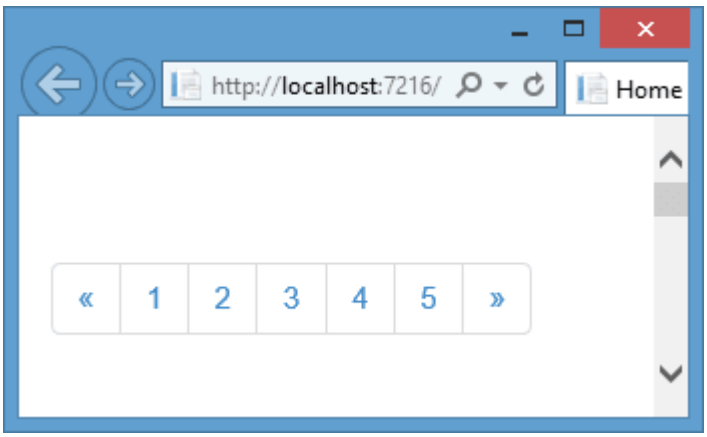

Рисунок 13 – Пример применения класса pagination

Заглавия

Для создания ссылок заголовков применяется класс breadcrumb:

```
<ulclass="breadcrumb">
    <liclass="active">@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
```

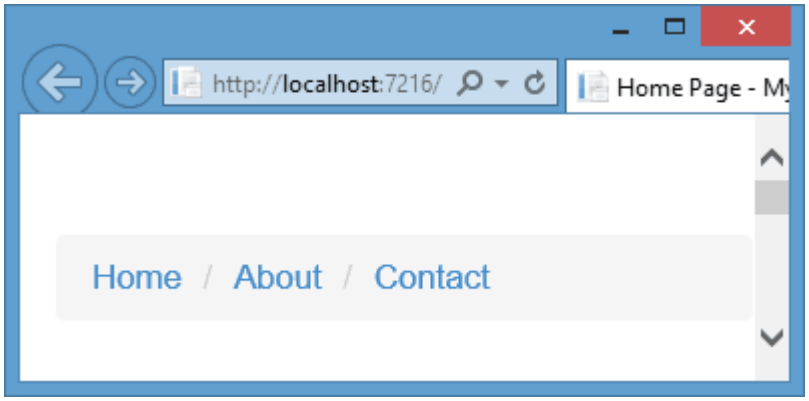

Рисунок 14 – Применение класса breadcrumb

Кнопки

Для создания кнопок Bootstrap имеет класс btn. Как правило, кнопки офрмляются в группу с помощью класса btn-group:

```
<divclass="btn-group"role="group">
  <buttontype="button"class="btnbtn-default">Left</button>
  <buttontype="button"class="btnbtn-default">Middle</button>
  <buttontype="button"class="btnbtn-default">Right</button>
</div>
```

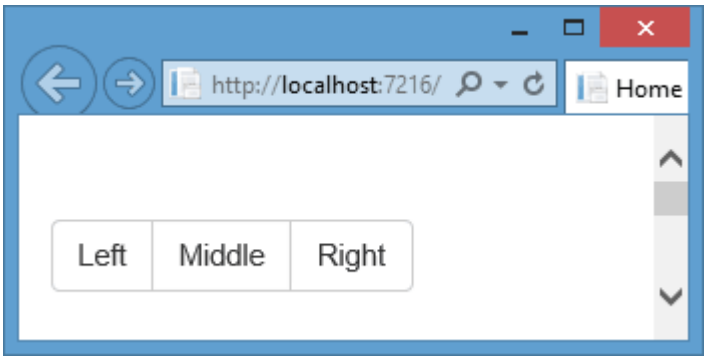

Рисунок 14 – Применение класса btn

Кнопка с выпадающим списком

Для создания выпадающего списка по примеру элемента нам надо использовать кнопку вместе со списком, который должен иметь класс dropdown-menu:

```
<divclass="btn-group">
    <buttontype="button"class="btnbtn-default dropdown-toggle"data-
toggle="dropdown">
        Языкпрограммирования<spanclass="caret"></span>
    </button>
    <ulclass="dropdown-menu"role="menu">
        <li><ahref="#">JavaScript</a></li>
```

```
        <li><ahref="#">C#</a></li>
        <li><ahref="#">VB.NET</a></li>
        <liclass="divider"></li>
        <li><ahref="#">Java</a></li>
    </ul>
</div>
```

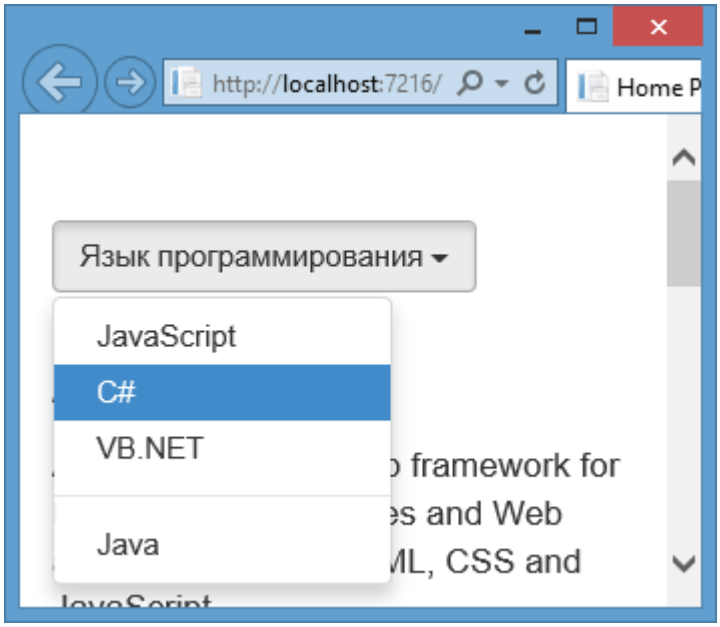

Рисунок 15 – Пример реализации выпадающего списка

Метки

Для оформления кусков текста в качестве меток мы можем использовать класс label. Также мы можем использовать дополнительные классы меток, чтобы конкретизировать тип сообщения:

```
<spanclass="label label-default">Default</span>
<spanclass="label label-primary">Primary</span>
<spanclass="label label-success">Success</span>
<spanclass="label label-info">Info</span>
<spanclass="label label-warning">Warning</span>
<spanclass="label label-danger">Danger</span>
```

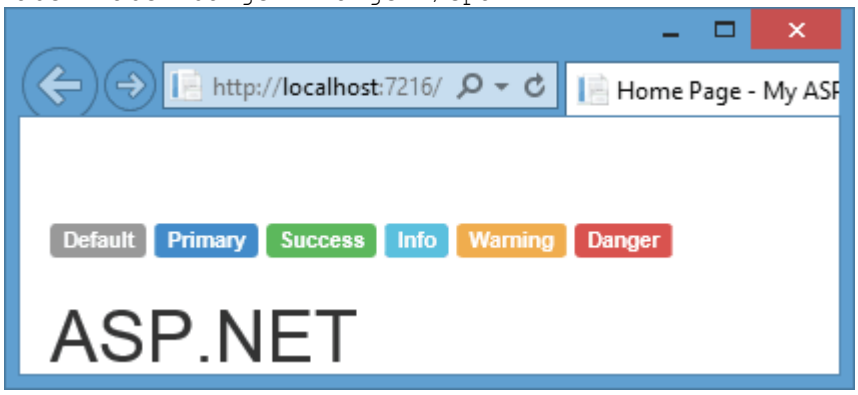

Рисунок 16 – Пример применения меток

Похожую функциональность предлагает класс alert:

```
<divclass="alert alert-success">Задачауспешнозавершена</div>
```

```
<divclass="alertalert-info">Дополнительная информация</div>
<divclass="alert alert-warning">Внимание!</div>
<divclass="alert alert-danger">Опасно!</div>
```

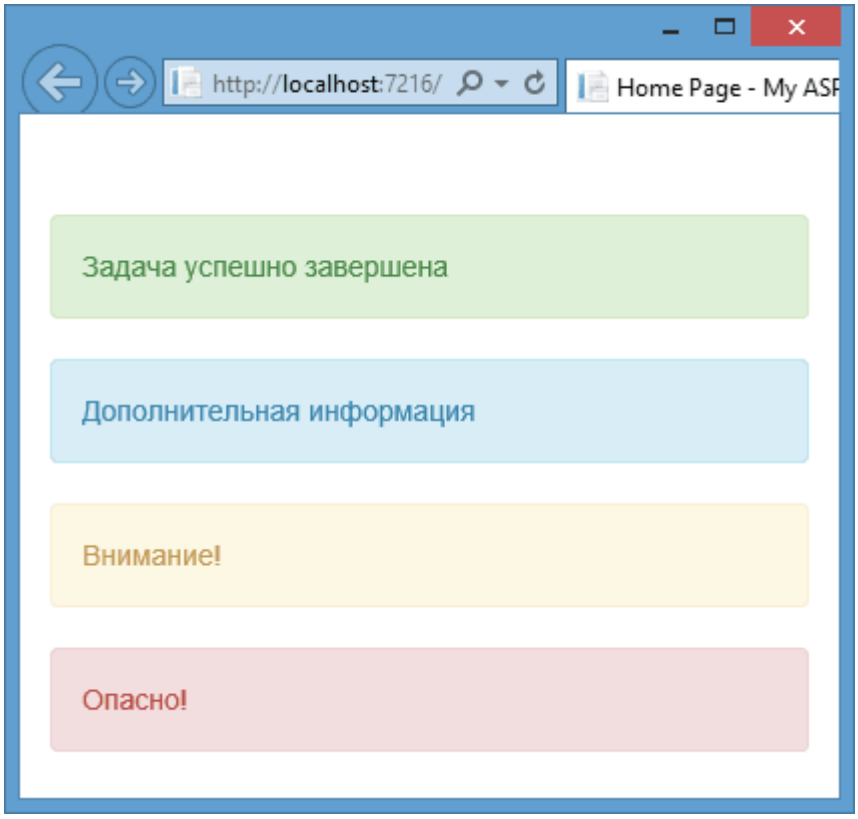

Рисунок 17 – Пример применения класса alert

Это только некоторые компоненты, которые предлагает Bootstrap. Но уже по ним можно увидеть, что Bootstrap довольно гибок, и даже сложные по структуре компоненты может легко адаптировать и приспособить под конкретные устройства.

Работа с формами в Bootstrap

Кроме отдельных компонентов Botstrap поддерживает также работу с формами как с единым целым. Если мы возьмем стандартное представление для регистрации Register.cshtml, то мы увидим, что к элементам формы уже применяют классы Bootstrap:

```
@{
    ViewBag.Title = "Register";
}

<h2>@ViewBag.Title.</h2>
```

```
@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr/>
    @Html.ValidationSummary()
    <divclass="form-group">
        @Html.LabelFor(m =>m.UserName, new { @class = "col-md-2 control-
label" })
        <divclass="col-md-10">
            @Html.TextBoxFor(m =>m.UserName, new { @class = "form-control" })
        </div>
    </div>
    <divclass="form-group">
        @Html.LabelFor(m =>m.Password, new { @class = "col-md-2 control-
label" })
        <divclass="col-md-10">
            @Html.PasswordFor(m =>m.Password, new { @class = "form-control"
})
        </div>
    </div>
    <divclass="form-group">
        @Html.LabelFor(m =>m.ConfirmPassword, new { @class = "col-md-2
control-label" })
        <divclass="col-md-10">
            @Html.PasswordFor(m =>m.ConfirmPassword, new { @class = "form-
control" })
        </div>
    </div>
    <divclass="form-group">
        <divclass="col-md-offset-2 col-md-10">
            <inputtype="submit"class="btnbtn-default"value="Register"/>
        </div>
    </div>
       }
```

К форме по умолчанию добавляется класс **form-horizontal**, который устанавливает выравнивание меток label по правой стороне связанных элементов ввода:

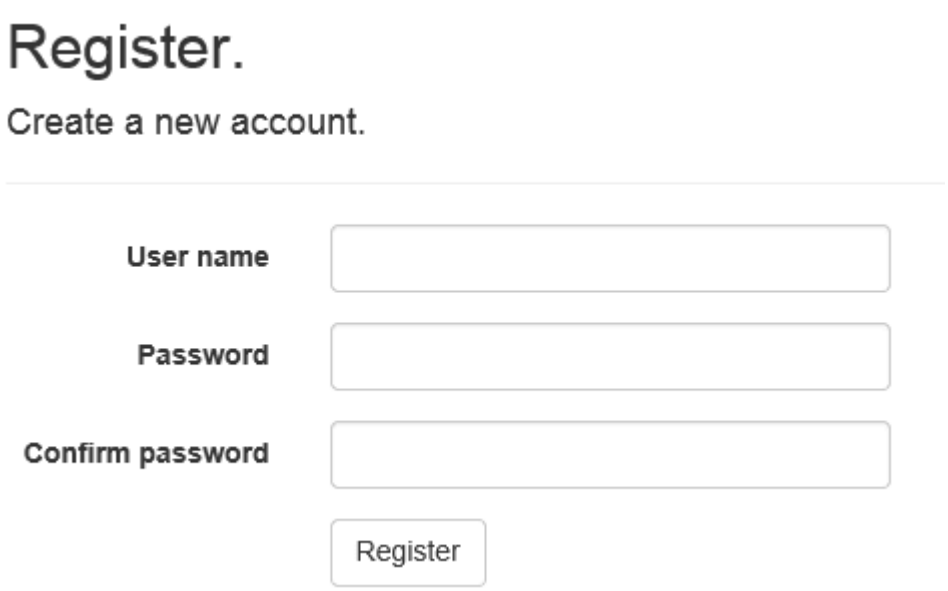

Рисунок 18 –Пример применения класса form-horizontal

Кроме того, элементы визуально отличаются от стандартных, так как к ним применяются стили из Bootstrap. Для стилизации элементов форм служит класс **form-control**. Здесь также используется класс **form-group**, который задает расстояние между элементами форм.

Кроме `form-horizontal` мы можем применить к форме другой класс: **form-inline**, который выстраивает все элементы в линию. Например:

```
@using (Html.BeginForm("Login", "Account", FormMethod.Post, new { @class = "form-inli
role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Use a local account to log in.</h4>
    <hr/>
    @Html.ValidationSummary(true)
    <divclass="form-group">
        <divclass="col-md-10">
            @Html.TextBoxFor(m =>m.UserName, new { @class = "form-control", @placehol
"Логин" })
            @Html.ValidationMessageFor(m =>m.UserName)
        </div>
    </div>
    <divclass="form-group">
        <divclass="col-md-10">
            @Html.PasswordFor(m =>m.Password, new { @class = "form-control",
@placeholder="Пароль" })
            @Html.ValidationMessageFor(m =>m.Password)
        </div>
    </div>
    <divclass="form-group">
        <divclass="col-md-offset-2 col-md-10">
            <divclass="checkbox">
                @Html.CheckBoxFor(m =>m.RememberMe)
```

```
                @Html.LabelFor(m =>m.RememberMe)
            </div>
        </div>
    </div>
    <divclass="form-group">
        <divclass="col-md-offset-2 col-md-10">
            <inputtype="submit"value="Log in"class="btnbtn-default"/>
        </div>
    </div>
}
```

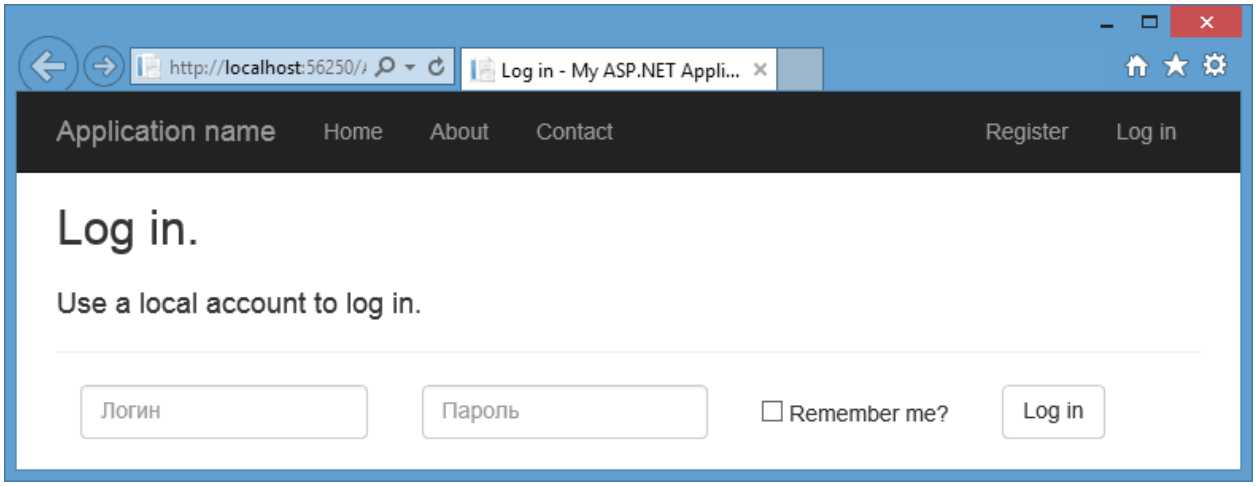

Рисунок 19 – Результат применениякласса form-inline, который выстраивает все элементы в линию

И в заключение разговора о формах следует сказать, что при использовании шаблонов формирования, когда VisualStudio автоматически генерирует вам разметку для представлений для создания и редактирования моделей, а создаваемые автоматически формы уже применяют классы Bootstrap, поэтому вам не придется вручную добавлять к ним вышеописанные классы.

Работа с таблицами

Для оформления таблиц в Bootstrap предназначен класс table. Например, пусть у нас есть такая таблица:

```
<divclass="row">
    <divclass="col-md-6">
        <tableclass="table">
            <tbody>
                <tr>
                    <th>
                        Язык
                    </th>
```

```
                    <th>
                        Процент
                    </th>
                </tr>
                <tr><td>C</td><td>17.5%</td></tr>
                <tr><td>Java</td><td>16.4%</td></tr>
                <tr><td>Objective-C</td><td>12.1%</td></tr>
                <tr><td>C++</td><td>6.3%</td></tr>
                <tr><td>C#</td><td>5.6%</td></tr>
            </tbody>
        </table>
    </div>
</div>
```

Тогда применение класса table будет давать следующий вывод:

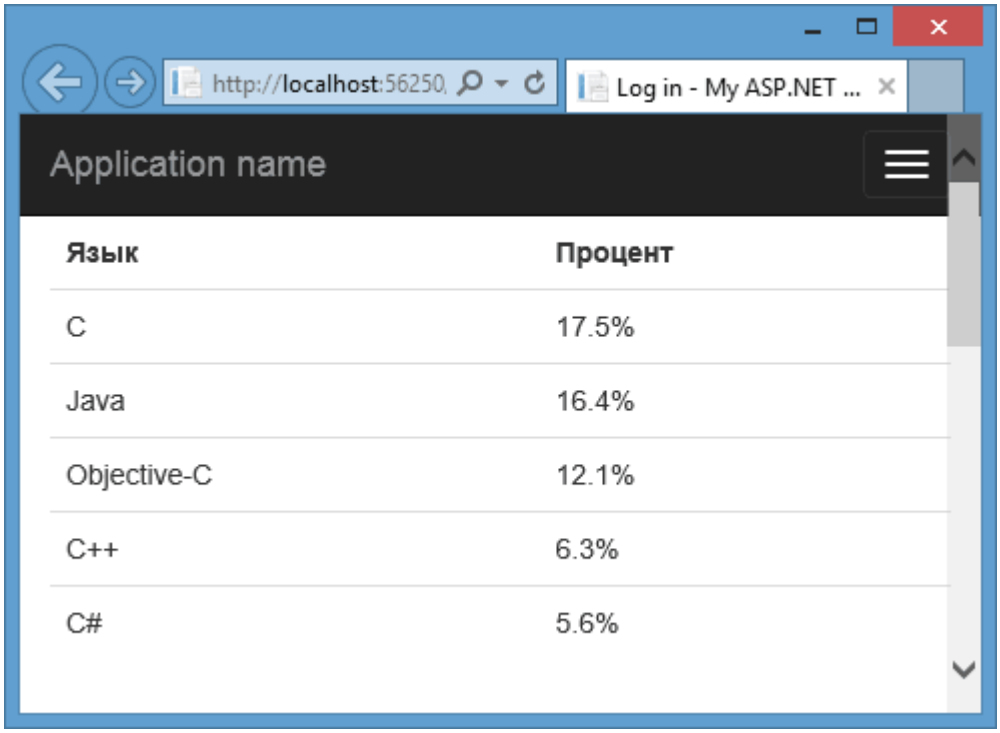

Рисунок 20 – Пример реализации таблицы

Кроме этого класса мы можем использовать в дополнение к нему еще ряд классов. Например, класс table-striped позволяет выделить четные и нечетные строки в таблице:

```
<tableclass="table table-striped">
    <tbody>
      ..........................
    </tbody>
</table>
```

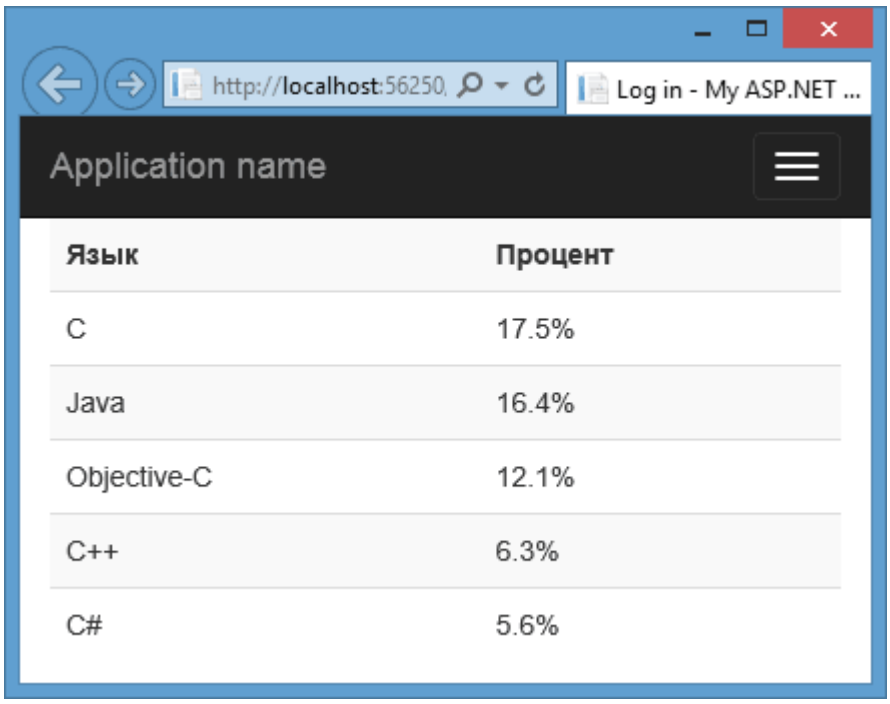

Рисунок 21 – Пример применения классatable-striped

Класс table-bordered создает границы для таблицы и ее ячеек, а класс table-hover позволяет выделять строку при наведении на нее курсором:

```
<tableclass="table table-bordered table-hover">
    <tbody>
      ..........................
    </tbody>
</table>
```

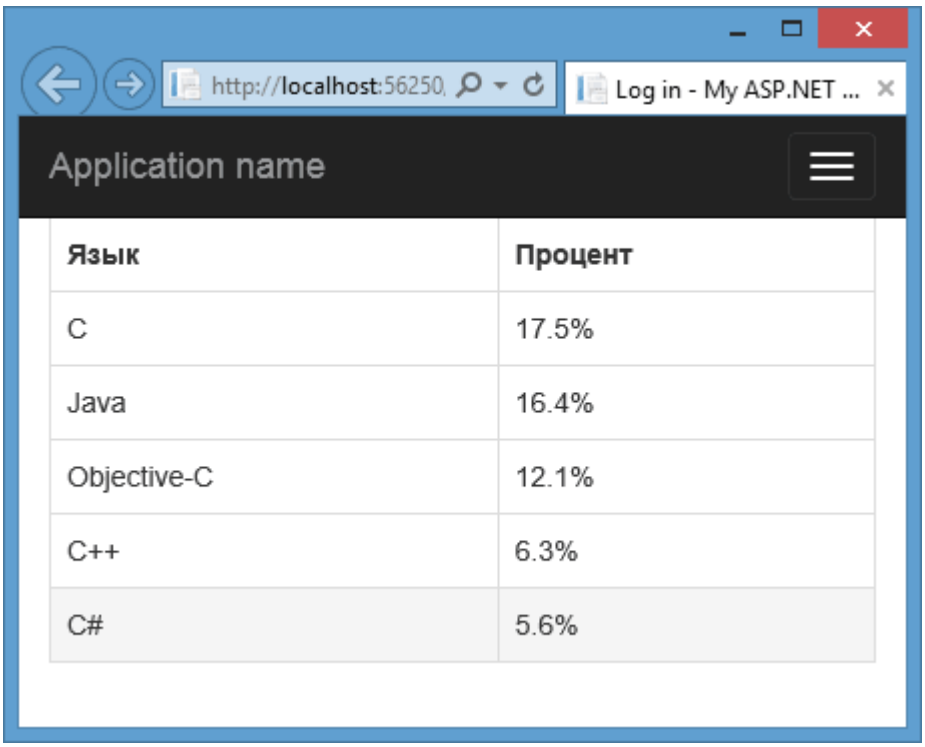

Рисунок 22 – Пример применения классatable-bordered

Итак, мы рассмотрели только небольшую часть возможностей, колторые предлагает Bootstrap. Но вне сомнения то, что использование данного фреймворка позволяет быстрее создавать адаптивные веб-приложения и расширяет арсенал веб-разработчика.

Модальные окна

Фреймворк Bootstrap предлагает прекрасные возможности по созданию модальных окон. По своей функциональности они напоминают модальные окна в jQuery UI.

Для использования модальных окон создадим новый проект. Наш проект будет использовать модель, описывающую модель компьютера:

```
publicclassComputer
{
    publicintId { get; set; }
    publicstringName { get; set; }
    publicstringCompany { get; set; }
    publicintYear { get; set; }
}
```

Изменим контроллер HomeController следующим образом:

```
publicclassHomeController : Controller
{
    staticList<Computer> comps=newList<Computer>();
    staticHomeController()
    {
        comps.Add(newComputer { Id = 1, Name = "Apple II", Company = "Apple",
Year = 1977 });
        comps.Add(newComputer { Id = 2, Name = "Macintosh", Company =
"Apple", Year = 1983 });
        comps.Add(newComputer { Id = 3, Name = "IBM PC", Company = "IBM",
Year = 1981 });
        comps.Add(newComputer { Id = 4, Name = "Altair", Company = "MITS",
Year = 1975 });
    }
    publicActionResult Index()
    {
        returnView(comps);
    }
    publicActionResult Details(intid)
    {
        Computer c = comps.FirstOrDefault(com =>com.Id == id);
        if(c!=null)
            returnPartialView(c);
        returnHttpNotFound();
    }
}
```

Есть обычный метод Index, который возвращает стандартное представление со списком, и есть метод Details, который генерирует частичное представление с данными о выбранной модели. Это частичное представление и будет использоваться модальным окном.

Теперь создадим представления. Изменим представление Index.chtml следующим образом:

```
@model IEnumerable<ModalBootstrap.Models.Computer>
@{
    ViewBag.Title = "Компьютеры";
}
<h3>Список моделей</h3>
<divid="modDialog"class="modal fade">
    <divid="dialogContent"class="modal-dialog"></div>
</div>
<ul>
    @foreach (var c in Model)
    {
        <li>@Html.ActionLink(c.Name, "Details",
                new { id = c.Id }, new { @class = "compItem" })</li>
    }
</ul>
@section scripts
{
    <scripttype="text/javascript">

        $(function () {
            $.ajaxSetup({ cache: false });
           $(".compItem").click(function (e) {

                e.preventDefault();
                $.get(this.href, function (data) {
                    $('#dialogContent').html(data);
                    $('#modDialog').modal('show');
                });
            });
        })
    </script>
}
```

Нажатие на любой элемент списока приведет к тому, что сработает обработчик, определенный в коде javascript. Этот обработчик будет загружать содержимое частичное представления, которое мы чуть позже определим. Данные загружаются в элемент с id="dialogContent". И чтобы этот элемент расценивался как модальное диалоговое окно, у него устанавливается класс modal-dialog.

Этот блок определен внутри блока modDialog, для которого установлены классы modalfade.

Теперь создадим частичное представление Details.cshtml, которое будет использоваться модальным окном:

```
@model ModalBootstrap.Models.Computer

<divclass="modal-content">
    <divclass="modal-header">
        <buttonclass="close"data-dismiss="modal"area-hidden="true">X</button>
        <h4>Модель компьютера</h4>
    </div>
    <divclass="modal-body">
        <dlclass="dl-horizontal">
            <dt>Название модели:</dt>
            <dd>@Html.DisplayFor(model =>model.Name)</dd>

            <dt>Производитель</dt>
            <dd>@Html.DisplayFor(model =>model.Company)</dd>

            <dt>Годвыпуска</dt>
            <dd>@Html.DisplayFor(model =>model.Year)</dd>
        </dl>
    </div>
</div>
```

Здесь опять же используются классы bottstrap для формирования содержимого модального окна.

Все содержимое помещается в блок div с классом modal-content. Для создания заголовочной части модального окна используется блок с классом modal-header. В частности кроме собственно заголовка мы можем определить кнопку закрытия окна, присвоив ей класс close. И также здесь указываем с помощью атрибута data-dismiss элемент, который должен "закрываться" - то есть блок div с классом modal.

Непосредственно содержимое помещается в блок modal-body.

Теперь мы можем протестировать приложение. При нажатии на любую ссылку моделей должно появиться модальное окно с подробной информацией:

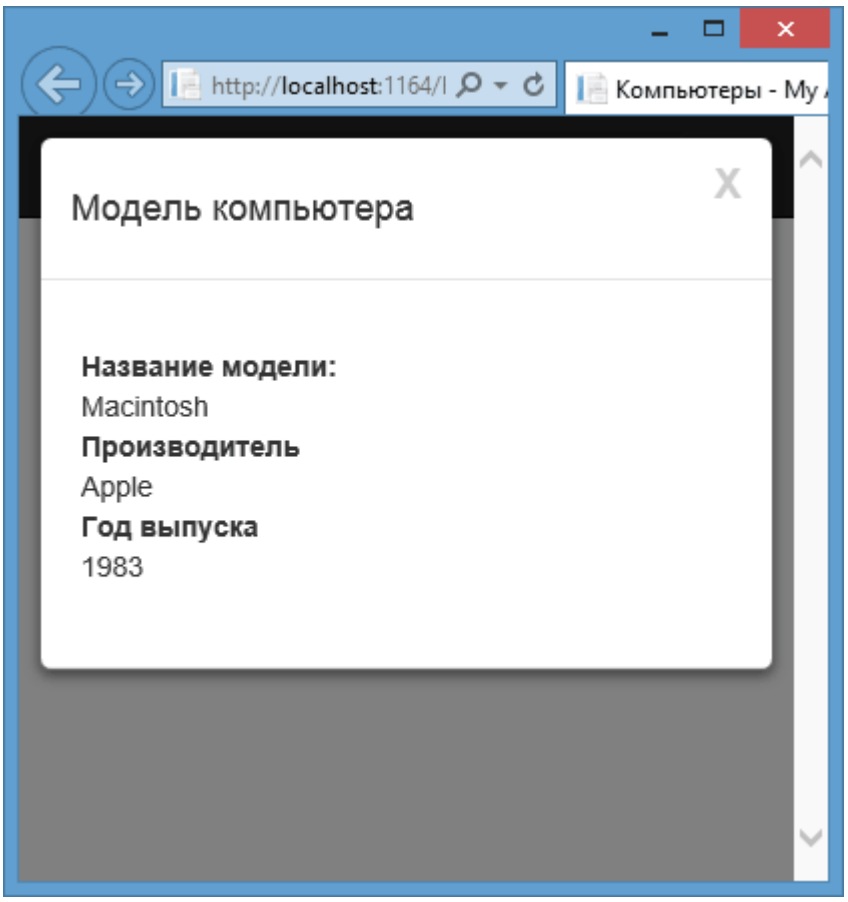

Рисунок 23 – Пример реализованного модального окна

И при нажатии на крестик, модальное окно закроется. Таким образом, мы можем использовать bootstrap для создания модальных окон.