8) Элементы управления и синтаксис страниц ASP.NET Web Forms // jQuery, AngularJS

Элементы управления и синтаксис страниц ASP.NET Web Forms

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ HTML

ASP.NET содержит набор элементов управления, близко соответствующий обычным элементам HTML. В коде разметки страницы *серверные элементы управления HTML* выглядят как обычные HTML-теги, но с атрибутом `runat="server"`. Серверные элементы HTML определены для большинства тэгов, таких как `<form>`, `<input>`, `<select>`, `<table>`, `<img>`, `<a>`. Все они наследуются от одного класса – `HtmlControl` (пространство имен `System.Web.UI.HtmlControls`), производного от класса `Control`.

Свойства класса `HtmlControl`

Свойство	Описание
`Attributes`	Коллекция всех атрибутов элемента управления и их значений
`Disabled`	Булево значение, указывающее, отключен ли элемент управления
`Style`	Коллекция, представляющая все CSS-свойства элемента управления
`TagName`	Возвращает имя HTML-тэга элемента управления

Коллекцию `Attributes` можно использовать для установки таких HTML-атрибутов, которым не соответствуют специализированные свойства элемента управления. Содержимое коллекции заносится в HTML-вывод как набор пар «`атрибут=значение`». Например, в следующем коде устанавливается значение атрибута `onload` HTML-тега `<body>`.

```
<script>
    function Init() { alert("Hello"); }
</script>

<script runat="server" language="C#">
    protected void Page_Load(object sender, EventArgs e)
    {
        theBody.Attributes["onload"] = "Init()";
    }
</script>

<html>
   <body ID ="theBody" runat="server" />
</html>
```

Большинство элементов управления HTML можно разделить на две категории: *контейнерные* и *предназначенные для ввода данных*.

Контейнерные элементы управления HTML

Класс	Что представляет
HtmlAnchor	Якорь HTML, тэг <a>
HtmlButton	Тэг <button>, который определен в спецификации HTML 4.0
HtmlForm	Тэг <form>. Может использоваться только в качестве контейнера интерактивных серверных элементов управления, но не пригоден для создания HTML-форм, программируемых на сервере
HtmlGenericControl	HTML-тэг, для которого в ASP.NET не определен специальный класс. Примерами являются тэги <font>, <hr>, <iframe>. Их программируют с использованием коллекции Attributes
HtmlHead	Тэг <head>. Позволяет программно управлять метатэгами, таблицами стилей и заголовком страницы ★
HtmlSelect	Тэг <select>, то есть группу вариантов выбора
HtmlTable	Таблицу HTML - тэг <table>
HtmlTableCell	Тэг <td>, то есть ячейку таблицы
HtmlTableRow	Тэг <tr>, то есть строку таблицы
HtmlTextArea	Многострочное текстовое поле - тэг <textarea>

Базовым классом контейнерных элементов управления HTML является класс HtmlContainerControl. Он представляет все элементы, которые должны иметь закрывающийся тэг: формы, блоки выбора, таблицы, а также якоря и текстовые области. По сравнению с классом HtmlControl контейнерный элемент управления имеет два дополнительных строковых свойства - InnerHtml и InnerText. Оба свойства управляют чтением и записью литерального контента, расположенного между открывающим и закрывающим тэгами элемента.

Для страницы, содержащей тэг <head> с атрибутом runat="server", автоматически создается элемент управления HtmlHead, определяющий *заголовок страницы*. Заголовок страницы представлен новым свойством Header класса Page. Если тэг <head> отсутствует или не имеет атрибута runat="server", данное свойство содержит null. Элемент управления HtmlHead имеет свойство Title, посредством которого можно извлекать и задавать заголовок страницы.

Якорный элемент управления HtmlAnchor и элемент управления HtmlButton могут использоваться не только для перехода к другой странице (что является их основным назначением), но и для осуществления возврата формы. Ниже продемонстрировано объявление якоря, в котором событию-щелчку назначены и клиентский, и серверный обработчики. Атрибутом onclick определяется клиентский обработчик, написанный на JavaScript, а атрибутом onserverclick - серверный обработчик, код которого будет выполнен после возврата формы.

```
<a runat="server" onclick="Run()" onserverclick="DoSome">Click</a>
```

Элемент управления HtmlSelect представляет список опций, из которых можно выбрать одну или несколько. Этот элемент поддерживает *связывание с источником данных*. Его поведением управляют свойства Size (количество опций) и Multiple (разрешен ли выбор нескольких опций). Сами элементы-опции хранятся в коллекции Items, состоящей из объектов ListItem. Чтобы задать текст элементов, можно либо установить свойство Text каждого из

объектов ListItem, либо разместить между открывающимся и закрывающимся тэгами <select> группу тэгов <option>.

В ASP.NET простейшую таблицу HTML можно вывести с помощью элемента управления HtmlTable. Но серверные таблицы не столь мощны, как обычные таблицы HTML, создаваемые с использованием тэга <table>. Главное их ограничение: HtmlTable не поддерживает тэги <caption>, <col>, <colgroup>, <tbody>, <thead> и <tfoot>. Дочерними элементами HtmlTable по определению могут быть только объекты класса HtmlTableRow.

Элемент управления HtmlTextArea, соответствующий тэгу <textarea>, позволяет программно создавать и конфигурировать многострочные текстовые поля. У класса HtmlTextArea имеются свойства Rows и Cols, с помощью которых задаётся количество строк и столбцов поля, а свойство Value может использоваться для определения выводимого в нем текста. При возврате формы класс HtmlTextArea генерирует событие ServerChange, которое позволяет проверить на сервере данные, содержащиеся в элементе управления.

В языке HTML элемент <input> имеет несколько разновидностей и может использоваться для вывода кнопки типа submit, флажка или текстового поля. Каждой из таких разновидностей элемента <input> соответствует свой класс ASP.NET. Все эти классы являются производными от HtmlInputControl - абстрактного класса, определяющего их общий программный интерфейс. Данный класс наследуется от HtmlControl, добавляя к нему свойства Name, Type и Value. Свойство Name возвращает имя, присвоенное элементу управления. Свойство Type, соответствующее атрибуту type HTML-элемента, доступно только для чтения. Что касается свойства Value, представляющего содержимое поля ввода, то его значение можно и считывать, и записывать.

Элементы управления HTML, предназначенные для ввода данных

Класс	Что представляет
HtmlInputButton	Различные виды кнопок, поддерживаемые HTML. Допустимыми значениями атрибута Type являются button, submit и reset
HtmlInputCheckBox	Флажок HTML, то есть тэг <input> типа checkbox
HtmlInputFile	Загрузчик файлов - тэг <input> типа file
HtmlInputHidden	Скрытый буфер текстовых данных - тэг <input> типа hidden
HtmlInputImage	Графическая кнопка - тэг <input> типа image
HtmlInputPassword	Защищённое текстовое поле - тэг <input> типа password ★
HtmlInputRadioButton	Переключатель - тэг <input> типа radio
HtmlInputReset	Командная кнопка типа reset ★
HtmlInputSubmit	Командная кнопка типа submit ★
HtmlInputText	Текстовое поле - тэг <input> типа password или text

Событие ServerChange, генерируемое при изменении в состоянии элемента управления между возвратами формы, поддерживается input-элементами HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden и HtmlInputText.

Элемент управления HtmlInputButton генерирует событие ServerClick, позволяя задать код, который будет выполнен на сервере после щелчка кнопки.

Элемент управления HtmlInputImage имеет несколько дополнительных свойств, связанных с выводом изображения. В частности, он позволяет задать альтернативный текст (выводимый, когда изображение недоступно), рамку и способ выравнивания изображения по отношению к остальной части страницы. Обработчик события ServerClick этого элемента управления получает структуру данных ImageClickEventArgs, которая хранит в свойствах X и Y координаты указателя мыши в момент щелчка.

Элемент управления HtmlInputFile является HTML-средством загрузки файлов из браузера на веб-сервер. На сервере файл упаковывается в объект типа HttpPostedFile и остается там до тех пор, пока не будет явно обработан, например, сохранен на диске или в базе данных. Объект HttpPostedFile имеет свойства и методы, с помощью которых можно получить информацию о файле, извлечь его и сохранить. Кроме этого, элемент управления HttpInputFile позволяет ограничить перечень типов файлов, которые разрешено загружать на сервер. ASP.NET позволяет в некоторой степени контролировать количество загружаемых данных. Максимально допустимый размер файла (по умолчанию - 4 Мбайт) можно задать в атрибуте maxRequestLength раздела <httpRuntime> конфигурационного файла веб-приложения.

5.9. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ WEB

Элементы управления Web, определенные в пространстве имен System.Web.UI.WebControls, являются серверными компонентами, создаваемыми при наличии атрибута runat="server". В теле aspx-страницы Web-элементы можно отличить по префиксу пространства имен asp. Некоторые из этих элементов управления подобны серверным элементам HTML, но интерфейс программирования при этом имеют разный. Web-элементы обладают более согласованным и абстрактным API и более богатой функциональностью.

Все элементы управления Web наследуются от базового класса WebControl. Он является производным от класса Control и определяет ряд собственных свойств и методов. Большинство членов класса WebControl связаны с внешним видом и поведением элементов управления (это шрифты, стиль, цвета, CSS).

Свойства класса WebControl перечислены в табл. 19.

Таблица 19

Свойства класса WebControl

Свойство	Описание
AccessKey	Позволяет задать клавишу, которая совместно с клавишей Alt будет использоваться для быстрого перехода к элементу управления. Поддерживается в IE версии 4.0 и выше
Attributes	Коллекция атрибутов, не имеющих соответствия среди свойств элемента управления. Атрибуты, задаваемые через эту коллекцию, выводятся в составе страницы как атрибуты HTML
BackColor	Цвет фона элемента управления
BorderColor	Цвет рамки элемента управления
BorderStyle	Стиль рамки элемента управления
BorderWidth	Ширина рамки элемента управления
CssClass	Класс CSS, связанный с элементом управления
Enabled	Указывает, активен ли элемент управления
Font	Свойства шрифта элемента управления

ForeColor	Цвет фона элемента управления; используется при выводе текста
Height	Высота элемента управления. Задается как значение типа Unit
Style	Возвращает коллекцию CssStyleCollection, составленную из атрибутов, которые присвоены выводимому тэгу элемента
Tablndex	Индекс перехода по клавише Tab для элемента управления
ToolTip	Текст всплывающей подсказки, которая выводится при наведении на элемент управления указателя мыши
Width	Ширина элемента управления. Задается как значение типа Unit

Класс WebControl определяет несколько дополнительных методов, отсутствующих у базового класса Control. Все они перечислены в табл. 20.

Таблица 20

Методы класса WebControl

Метод	Описание
ApplyStyle()	Копирует в элемент управления непустые элементы заданного стилевого объекта. Существующие стилевые свойства переопределяются
CopyBaseAttributes()	Импортирует из заданного элемента управления Web свойства AccessKey, Enabled, ToolTip, TabIndex и Attributes. Иными словами, копирует все свойства, не инкапсулированные объектом Style
MergeStyle()	Подобно методу ApplyStyle() копирует в элемент управления непустые элементы заданного стиля, но существующие стилевые свойства не переопределяются
RenderBeginTag()	Осуществляет рендеринг открывающегося HTML-тэга элемента управления в заданный объект записи текста. Вызывается перед методом RenderControl()
RenderEndTag()	Осуществляет рендеринг закрывающегося HTML-тэга элемента управления. Вызывается сразу после метода RenderControl()

Перечислим наиболее популярные и важные элементы управления Web в табл. 21, а затем рассмотрим некоторые из них более подробно.

Таблица 21

Ключевые элементы управления Web

Элемент управления	Что представляет
Button	Кнопку, реализованную в виде тэга <input>
Checkbox	Флажок, реализованный в виде тэга <input>
FileUpload	Элемент интерфейса, дающий возможность пользователю выбрать файл для загрузки на сервер ★
HiddenField	Скрытое поле ★
HyperLink	Якорный тэг <a>; позволяет указать либо адрес для перехода, либо сценарий для выполнения
Image	Изображение, реализованное в виде тэга <img>
ImageButton	Изображение, отвечающее на щелчки мыши подобно настоящей кнопке
ImageMap	Изображение с необязательной областью в нём, которую можно щелкать мышью ★

Label	Обыкновенный статический текст, не реагирующий на щелчки. Реализован в виде тэга
LinkButton	Якорный тэг, обеспечивающий возврат формы с использованием соответствующего механизма ASP.NET. Это гиперссылка особого рода, для которой программист не может задать целевой URL
MultiView	Элемент управления, действующий как контейнер группы дочерних элементов типа View ★
Panel	HTML-контейнер, реализованный с использованием блочного элемента . В ASP.NET 2.0 этот контейнер поддерживает скроллинг
RadioButton	Одну кнопку переключателя, реализованную в виде тэга <input>
Table	Внешний табличный контейнер; эквивалентен HTML-элементу <table>
TableCell	Ячейку таблицы; эквивалентен HTML-элементу <td>
TableRow	Строку таблицы; эквивалентен HTML-элементу <tr>
TextBox	Текстовое поле, реализованное в виде тэга <input> или <textarea>, что зависит от запрошенного типа текста. Может работать в одно- или многострочном режиме либо в режиме ввода пароля
View	Контейнер группы элементов управления. Этот элемент управления всегда должен содержаться в элементе управления MultiView ★

Кнопочные элементы управления

В ASP.NET элементы управления Web, генерирующие кнопки, реализуют интерфейс IButtonControl. Его реализуют элементы Button, ImageButton и LinkButton, а в общем случае - любой специализированный элемент управления, который должен действовать как кнопка. В табл. 22 перечислены все члены интерфейса IButtonControl.

Таблица 22

Интерфейс IButtonControl

Элемент	Описание
CausesValidation	Значение булева типа, указывающее, должна ли по щелчку элемента управления выполняться валидация формы
CommandArgument	Возвращает и позволяет задать значение необязательного параметра, передаваемого обработчику события Command кнопки вместе со связанным с этой кнопкой значением CommandName
CommandName	Возвращает и позволяет задать имя связанной с кнопкой команды, передаваемое обработчику события Command
PostBackUrl	Определяет URL страницы, которая будет обрабатывать возврат формы, вызванный щелчком кнопки. Данная функция, специфическая для ASP.NET, называется *межстраничным возвратом формы*
Text	Надпись на кнопке
ValidationGroup	Имя проверочной группы, в состав которой входит кнопка
Visible	Указывает, видим ли элемент управления

В дополнение к свойствам интерфейса IButtonControl, класс Button имеет свойства OnClientClick и UseSubmitBehavior. Свойство OnClientClick позволяет установить имя функции JavaScript, которая будет выполняться на клиенте в ответ на событие onclick (свойство OnClientClick имеется также у элементов LinkButton и ImageButton).

Гиперссылки

Элемент управления `HyperLink` создает ссылку на другую страницу и обычно выводится в виде текста, задаваемого в свойстве `Text`. В качестве альтернативы гиперссылка может быть представлена изображением, и тогда URL этого изображения задается в свойстве `ImageUrl`. Когда установлены оба свойства, преимущество имеет `ImageUrl`, а содержимое свойства `Text` выводится в виде всплывающей подсказки. Свойство `NavigateUrl` определяет URL, на который указывает гиперссылка. А в свойстве `Target` задается имя окна или фрейма, где будет выводиться контент, расположенный по целевому URL.

Статические изображения и графические кнопки

Элемент управления `Image` выводит на веб-странице обычное статическое изображение, путь к которому задается в свойстве `ImageUrl`. Адреса изображений могут быть абсолютными или относительными. При желании в свойстве `AlternateText` можно задать альтернативный текст, который будет выводиться в случае, когда изображение недоступно или когда браузер не показывает изображения. Способ выравнивания изображения относительно других элементов страницы указывается в свойстве `ImageAlign`.

Если нужно перехватывать щелчки изображения, воспользуйтесь вместо элемента управления `Image` элементом `ImageButton`. Класс `ImageButton` расширяет класс `Image` событиями `Click` и `Command`, генерируемыми в ответ на щелчок. Обработчик событиям `Click`, получает структуру данных `ImageClickEventArgs`. Эта структура содержит информацию о координатах точки элемента управления, которую щелкнул пользователь.

Нововведением ASP.NET 2.0 стал элемент управления `ImageMap`. В своей простейшей форме этот элемент выводит на странице изображение. Однако когда для него определена так называемая *горячая область*, элемент управления инициирует возврат формы или переход по заданному URL. Горячей областью называется часть изображения, щелчок которой вызывает определенное действие. Она реализуется в виде класса, наследующего класс `HotSpot`. Существует три предопределенных типа горячих областей: многоугольники, круги и прямоугольники.

Панели с прокруткой

Элемент управления `Panel` служит для группировки других элементов управления с использованием тэга `<div>`. В ASP.NET панели могут иметь вертикальные и горизонтальные полосы прокрутки, реализованные с использованием CSS-стиля `overflow`. Следующий пример показывает, как создается прокручиваемая панель:

```
<asp:Panel ID="MainPanel" runat="server" Height="60px"
           Width="420px" ScrollBars="Auto" BorderStyle="Solid">
    <h2>Choose</h2>
    <asp:CheckBox ID="cbx1" runat="server" /><br />
    <asp:CheckBox ID="cbx2" runat="server" /><br />
    <asp:CheckBox ID="cbx3" runat="server" /><br />
    <asp:CheckBox ID="cbx4" runat="server" /><br />
</asp:Panel>
```

Загрузка файлов на сервер

Элемент управления `FileUpload` обладает той же функциональностью, что и рассмотренный ранее элемент управления `HtmlInputFile`. Однако его программный интерфейс несколько иной, пожалуй, более интуитивный. Свойство `HasFile` и метод `SaveAs()` скрывают ссылку на объект, представляющий загруженный на сервер файл, а свойство `FileName` содержит имя этого файла.

Элемент управления `Xml`

Элемент управления `Xml` используется для вывода на странице ASP.NET XML-документа. Содержимое XML-файла может выводиться в исходном виде или с применением к нему XSL-трансформации (XSLT). Данный элемент управления является декларативным аналогом класса `XslTransform` и может использовать этот класс для своих целей.

С помощью элемента управления `Xml` удобно создавать блоки используемых клиентом XML-данных, задавая документы и применяемые к ним трансформации. XML-документ можно задать разными способами: используя объектную модель XML, в виде строки или указав имя файла. Трансформация XSLT определяется как заранее сконфигурированный экземпляр класса `XslTransform` или путем указания имени файла:

```
<asp:xml ID="xmlElem" runat="server" DocumentSource="document.xml"
                      TransformSource="transform.xsl" />
```

Если вы намерены применять к XML-данным ту или иную трансформацию, её можно задать между открывающим и закрывающим тэгами элемента управления. Также элемент управления `Xml` позволяет указывать требуемую трансформацию программными средствами.

Элемент управления `PlaceHolder`

Элемент управления `PlaceHolder` наследуется непосредственно от класса `Control` и используется только как контейнер для других элементов управления страницы. `PlaceHolder` не генерирует собственного вывода, и его функции ограничены отображением дочерних элементов управления, динамически добавляемых в его коллекцию `Controls`. Такой элемент управления не сообщает странице новой функциональности, а просто помогает сгруппировать связанные между собой элементы управления и облегчает их идентификацию.

Элементы управления View и MultiView

В ASP.NET 2.0 введены два новых элемента управления, предназначенных для создания группы сменяющих одна другую панелей дочерних элементов. Элемент управления `MultiView` определяет группу представлений, создаваемых экземплярами класса `View`. В каждый конкретный момент времени только одно из них активно и выводится для клиента. Элемент управления `View` предназначен для использования в составе элемента управления `MultiView`, а сам по себе он использоваться не может. Вот как определяется элемент управления `MultiView`:

```
<asp:MultiView ID="Tables" runat="server">
    <asp:View ID="Employees" runat="server">
        (здесь какие-то элементы управления)
    </asp:View>
    <asp:View ID="Products" runat="server">
        (здесь другие элементы управления)
    </asp:View>
</asp:MultiView>
```

Выбрать активное представление можно, используя событие возврата формы. Чтобы указать, какое представление будет следующим, можно либо установить свойство `ActiveViewIndex`, либо передать объект-представление методу `SetActiveView()`.

Общие сведения о синтаксисе веб-страниц ASP.NET

Веб-страницы ASP.NET создаются аналогично статическим веб-страницам HTML (страницам, которые не включают серверную обработку), но они содержат дополнительные элементы, которые ASP.NET распознает и обрабатывает при запуске страницы. Характеристики, отличающие веб-страницы ASP.NET от статических HTML-страниц (или других страниц):

- Файл страницы имеет расширение ASPX, а не HTM, HTML или какое-либо другое расширение. Расширение ASPX приводит к обработке страниц средой ASP.NET.
- Дополнительная директива @ Page или другая директива, соответствующая типу создаваемой страницы.
- Элемент form, который настраивается правильным образом для ASP.NET. Элемент form необходим только в том случае, когда страница содержит элементы управления, значения которых требуется использовать во время обработки страницы.
- Серверные веб-элементы управления.
- Код сервера, если на страницу добавляется собственный код.

Можно переименовать любую HTML-страницу, изменив ее расширение на ASPX, и она будет выполняться в виде веб-страницы ASP.NET. Однако если страница не включает серверную обработку, то нет необходимости добавлять к ней расширение ASPX, поскольку это увеличивает затраты при обработке страницы.

Пример веб-страницы ASP.NET

В следующем примере показана страница, содержащая основные элементы, которые составляют веб-страницу ASP.NET. Эта страница содержит статический текст, который может иметься и в HTML-странице, а также элементы, которые имеют отношение только к ASP.NET. Элементы, относящиеся только к ASP.NET, выделены.

```
<%@ Page Language="C#" %>
<html>
<script runat="server">Void Button1_Click(object sender, System.EventArgs
e) {    Label1.Text = ("Welcome, " + TextBox1.Text);}</script>
<head runat="server">
  <title>Basic ASP.NET Web Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <h1>Welcome to ASP.NET</h1>
    <p>Type your name and click the button.</p>
    <p>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server"          Text="Click"
OnClick="Button1_Click" />
    </p>
    <p>
      <asp:Label ID="Label1" runat="server"></asp:Label>
    </p>
  </form>
</body>
</html>
```

Директивы @

Страницы ASP.NET обычно содержат директивы, которые позволяют указать свойства и конфигурацию для страницы. Директивы используются ASP.NET в качестве инструкций, определяющих способ обработки страницы, но они не отображаются как часть разметки, которая отправляется в веб-обозреватель.

Наиболее часто используется директива @ Page, с помощью которой можно задавать множество параметров конфигурации для страницы, включая следующие параметры:

- Серверный язык программирования для кода на странице.
- Содержится ли код сервера непосредственно в странице, которая тогда называется однофайловой страницей, или код содержится в отдельном файле класса, и тогда страница называется страницей с выделенным кодом. Страница из предыдущего примера является однофайловой страницей; код находится непосредственно в странице, и директива @ Page не включает сведения о связанных файлах класса. Дополнительные сведения см. далее в разделе "Код сервера", а также в разделе Модель кода веб-страниц ASP.NET.
- Параметры отладки и трассировки.
- Имеет ли страница связанную главную страницу, и, следовательно, должна ли она рассматриваться как страница содержимого.

Если директива @ Page не включена в страницу, или если эта директива не включает определенные настройки, параметры наследуются из файла конфигурации для веб-приложения (файл Web.config) или из файла конфигурации узла (файл Machine.config).

Кроме директивы @ Page, можно включить другие директивы, которые поддерживают дополнительные параметры, специфичные для страницы. Наиболее часто используются следующие директивы:

- @ Import. Эта директива позволяет указывать пространства имен, на которые требуется ссылаться в коде.
- @ OutputCache. Эта директива позволяет указать, что страница должна кэшироваться, равно как и параметры, указывающие, когда и как долго кэшируются страницы.
- @ Implements. Эта директива позволяет указать, что страница реализует интерфейс .NET.
- @ Register. Эта директива позволяет регистрировать дополнительные элементы управления для использования на странице. Директива @ Register объявляет префикс тега элемента управления и расположение сборки элемента управления. Эту директиву необходимо использовать в случаях, когда требуется добавить на страницу пользовательские элементы управления или пользовательские элементы управления ASP.NET.

Некоторые типы файлов ASP.NET используют директиву, отличную от @ Page. Например, главная страница ASP.NET использует директиву @ Master, а пользовательские элементы управления ASP.NET используют директиву @ Control. Каждая директива позволяет задать различные параметры, соответствующие файлу.

Элементы формы

Если страница содержит элементы управления, дающие пользователям возможность взаимодействовать со страницей и отправлять ее, страница должна включать элемент form. При этом используется стандартный HTML-элемент form, но согласно определенным правилам. Ниже приведены правила использования элемента form:

- Страница может содержать только один элемент form.
- Элемент form должен содержать атрибут runat, который имеет значение server. Этот атрибут позволяет ссылаться на форму и на элементы управления страницы программным способом в коде сервера.
- Серверные элементы управления, которые могут выполнять обратную передачу, должны находиться внутри элемента form.
- Открывающий тег не должен содержать атрибут action. ASP.NET устанавливает эти атрибуты динамически при обработке страницы, переопределяя любые параметры, которые, возможно, были установлены ранее.

Серверные веб-элементы управления

В большинстве случаев на страницы ASP.NET добавляются элементы управления, дающие пользователю возможность взаимодействовать со страницей, включая кнопки, текстовые поля, списки и т. д. Эти серверные веб-элементы управления похожи на кнопки HTML и элементы input. Однако они обрабатываются на сервере, что позволяет

использовать код сервера для задания их свойств. Эти элементы управления также вызывают события, которые могут обрабатываться в коде сервера.

Серверные элементы управления используют специальный синтаксис, который ASP.NET распознает при запуске страницы. В следующем примере кода показаны некоторые типичные серверные веб-элементы управления.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server"
    Text="Click" OnClick="Button1_Click" />
```

Имена тегов для серверных элементов управления ASP.NET начинаются с префикса; в данном случае это префикс `asp:`. Префикс может отличаться, если элемент управления не является частью платформы .NET Framework. Серверные элементы управления ASP.NET также включают атрибут runat="server" и, при необходимости, идентификатор, который можно использовать для ссылки на элемент управления в коде сервера.

Во время выполнения страница определяет серверные элементы управления и выполняет код, который связан с этими элементами управления. Многие элементы управления отображают некоторую разметку HTML или другую разметку на странице. Например, элемент управления asp:textbox отображает на странице элемент input с атрибутом type="text". Однако однозначное сопоставление между серверным веб-элементом управления и HTML-элементом не обязательно. Например, элемент управления asp:calendar отображает HTML-таблицу. Некоторые элементы управления не отображают в веб-обозревателе ничего; они только обрабатываются на сервере и предоставляют сведения для других элементов управления.

Элементы HTML в качестве серверных элементов управления

Вместо серверных элементов управления ASP.NET или в дополнение к ним можно использовать обычные элементы HTML как серверные элементы управления. Можно добавить атрибут runat="server" и атрибут ID к любому HTML-элементу на странице. При запуске страницы ASP.NET идентифицирует такой элемент как серверный элемент управления и делает его доступным для кода сервера. Например, можно добавить необходимые элементы в HTML-элемент body, как показано в следующем примере:

```
<body runat="server" id="body">
```

Затем можно ссылаться на элемент body в коде сервера — например, для задания цвета фона в ответ на ввод данных пользователем или получение информации из базы данных во время выполнения.

Код сервера

Большинство страниц ASP.NET имеет код, который выполняется на сервере при обработке страницы. ASP.NET поддерживает множество языков, включая C#, Visual Basic, J#, JScript и другие.

ASP.NET поддерживает две модели написания кода сервера для веб-страницы. В однофайловой модели код для страницы находится в элементе script, где открывающийся тег содержит атрибут runat="server". В примере, приведенном ранее в этом разделе, показана однофайловая модель.

Кроме того, можно создать код для страницы в отдельном файле класса; такая модель называется моделью с выделенным кодом. В этом случае на веб-странице ASP.NET обычно отсутствует код сервера. Вместо этого директива @ Page включает сведения, которые связывают страницу ASPX с ее файлом с выделенным кодом. В следующем примере показана типичная директива @ Page для страницы с файлом с выделенным кодом:

```
<%@ Page Language="C#" CodeFile="Default.aspx.cs" Inherits="Default" %>
```

Атрибут CodeFile задает имя отдельного файла класса, а атрибут Inherits задает имя класса в файле с выделенным кодом, соответствующем странице.

jQuery, AngularJS

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML.

Библиотека jQuery предоставляет удобный API (Интерфейс программирования приложений)для работы с AJAX.

Возможности

- Движок кроссбраузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;
- События;
- Визуальные эффекты;
- AJAX-дополнения;
- JavaScript-плагины.

Query, как правило, включается в веб-страницу как один внешний JavaScript-файл:

```
<head>
<script src="jquery-2.1.1.min.js">
</script>
</head>
```

Вся работа с jQuery ведётся с помощью функции $.

```
jQuery(function($) {
  // здесь код скрипта, где в $ будет находиться объект,
предоставляющий доступ к функциям jQuery})
```

Работу с jQuery можно разделить на 2 типа:

- Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.
- Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Пример добавления к элементу обработчика события click с помощью jQuery:

```
$("a").click(function() {
  alert("Hello world!");
});
```

В данном случае при нажатии на элемент <A> происходит вызов alert("Hello world!").

Библиотека jQuery предназначена прежде всего для удобного поиска и манипулирования элементами на веб-странице. При нахождении определенного элемента с помощью jQuery можно повесить на него обработчики событий, анимировать, скрыть или, наоборот, отобразить, создать для элемента интерактивное взаимодействие с пользователем, изменить его стили и т.д. И даже если вы не работали раньше с jQuery, ее освоение не составит особых трудностей. Для более подробного ознакомления с данной библиотекой можно обратиться к руководству Изучаем jQuery. Здесь же я представлю краткий обзор возможностей библиотеки.

Функция jQuery

Функция `jQuery` позволяет использовать всей мощь библиотеки jQuery. Данная функция (`jQuery`) в качестве псевдонима имеет знак `$` (так как символ `$` легко набрать на клавиатуре, и он представляет действительное имя функции, которое можно употреблять в JavaScript). Поэтому следующие записи функции `jQuery` будут идентичны:

```
jQuery(document).ready(function(){
  // здесь код анонимной функции
});
```

```
$(function(){
  // здесь код анонимной функции
});
```

```
$(document).ready(function(){
  // здесь код анонимной функции
});
```

Все три вышеописанных случая идентичны и срабатывают сразу после загрузки веб-страницы. Весь остальной функционал помещается внутрь функции `jQuery`. И поскольку данная функция осуществляет выборку и модификацию элементов, то для нее иногда необходимо, чтобы вся веб-страница была уже загружена. Поэтому скрипт данной функции или ссылку на файл скрипта помещают обычно в самый низ веб-страницы.

Селекторы jQuery

Для выборки из структуры страницы нужных элементов используются селекторы.

Основные селекторы jQuery

Шаблон селектора	Значение	Пример
$("Element")	Выбирает все элементы с данным именем тега	`$("p")` выбирает все теги `p`. `$("ul")` выбирает все элементы `ul`

$("#id")	Выбирает элемент с определенным значением id	Так, в следующем коде:
		`<div id="box1"></div>`
		`<div id="box2"></div>`
		селектор `$("#box1")` выбирает элемент, помеченный жирным

$(".className")	Выбирает все элементы с определенным значением атрибута class	Допустим, у нас следующий код:
		`<div class="apple"></div>`
		`<div class="apple"></div>`
		`<div class="orange"></div>`
		`<div class="banana"></div>`
		то селектор `$(".apple")` выбирает все элементы, помеченные жирным

$("selector1,selector2,selectorN")	Выбирает элементы, которые соответствуют данным селекторам	Если у нас следующий код:
		`<div class="apple"></div>`
		`<div class="apple"></div>`
		`<div class="orange"></div>`
		`<div class="banana"></div>`
		то селектор `$(".apple, .orange")` выберет элементы, выделенные жирным

Например:

```
$(function () {
    $(".results").css("top", "20px");
    });
});
```

Сначала мы получаем элемент, у которого class имеет значение results (`< div class="results"></div>`), а потом с помощью функции `css` устанавливаем определенное значение для его свойства `top`. Причем если у нас на странице несколько элементов, у которых `class="results"`, то селектор вернет весь набор из этих элементов. И к каждому из элементов данного набора будет применено преобразование.

Выше в таблице показан лишь небольшой базовый список селекторов. Полный же список селекторов вы можете найти на сайте http://www.w3.org/TR/css3-selectors/

Фильтры jQuery

В дополнение к селекторам применяются фильтры. Можно выделить следующий набор базовых фильтров:

Фильтр	Значение
:eq(n)	Выбирает n-й элемент выборки (нумерация начинается с нуля)
:even	Выбирает элементы с четными номерами
:odd	Выбирает элементы с нечетными номерами
:first	Выбирает первый элемент выборки
:last	Выбирает последний элемент выборки
:gt(n)	Выбирает все элементы с номером, большим n
:lt(n)	Выбирает все элементы с номером, меньшим n
:header	Выбирает все заголовки (h1, h2, h3)
:not(селектор)	Выбирает все элементы, которые не соответствуют селектору, указанному в скобках

Например, если у нас на странице несколько элементов, у которых `class="results"`, а нам надо выбрать только первый, то мы можем применить следующие выражения: `$(".results:first")` или `$(".results:eq(0)")`

Специальный род фильтров - фильтры контента обеспечивают доступ к элементам, имеющим определенное содержимое:

Фильтр	Значение
:contains('content')	Получает все элементы, которые содержат `content`
:has('селектор')	Получает все элементы, которые содержат хотя бы один дочерний элемент, соответствующий селектору
:empty	Получает все элементы, которые не имеют дочерних элементов

:first-child	Получает все элементы, которые являются первыми дочерними элементами в своих родителях
:last-child	Получает все элементы, которые являются последними дочерними элементами в своих родителях
:nth-child(n)	Получает все элементы, которые являются n-ными элементами в своих родителях (нумерация идет с единицы)
:only-child	Получает все элементы, которые являются единственными дочерними элементами в своих родителях
:parent	Получает все элементы, которые имеют, как минимум, один дочерний элемент

Например, если мы хотим получить все элементы, содержащие текст `asp.net mvc`, мы можем применить следующее выражение: `$(':contains("asp.net mvc")')`

И завершая обзор фильтров, следует упомянуть о фильтрах форм, которые позволяют получить определенные элементы html-форм:

Фильтр	Значение
button	Получает все элементы `button` и элементы `input` с типом `button`
:checkbox	Получает все элементы `checkbox`
:checked	Получает все отмеченные элементы `checkbox` и `radio`
:disabled	Получает все элементы, которые отключены
:enabled	Получает все элементы, которые включены
:input	Получает все элементы `input`
:password	Получает все элементы `password`
:radio	Получает все элементы `radio`
:reset	Получает все элементы `reset`
:selected	Получает все отмеченные элементы `option`
:submit	Получает все элементы `input` с типом `submit`
:text	Получает все элементы `input` с типом `text`

Мы можем комбинировать в одном выражении несколько селекторов и фильтров: `$('.results:odd:has('img')')`. В данном случае мы выбираем все нечетные элементы с `class="results"`, которые содержат элементы `img`, то есть изображения.

События jQuery

jQuery предоставляет специальные методы для распространенных событий, как например, `click` или `submit`. Мы можем повесить свои обработчики для событий `mouseover` (наведение мыши) или `keydown`(нажатие клавиатуры) на любой элемент веб-страницы.

Например, обработчик нажатия мыши на элемент с `id="bg"` мог бы выглядеть следующим образом.

```
$("#bg").mousedown (function(e) {});
```

Или для примера обработаем нажатие клавиши:

```
$(document).keydown(function(e){
    // если нажата клавиша вверх
     if (e.which==38)
     {
        // поднимаем некоторый элемент на 5 пикселей вверх
        $("#paddleB").css("top",top-5);
    }
});
```

Методы jQuery

Как говорилось выше, jQuery выполняет две основные задачи - поиск элементов и их изменение. Если для поиска предназначены селекторы и фильтры, то для манипуляции над элементами используются методы. Эти методы позволяют изменять внешний вид элемента, анимировать его, перемещать в структуре элементов DOM. Это методы довольно многочисленны, поэтому рассмотрим лишь вкратце:

Метод	Описание
addClass('someClass')	Добавляет для выбранного элемента класс `someClass`
removeClass('someClass')	Удаляет для выбранного элемента класс `someClass`
toggleClass('someClass')	Переключает для выбранного элемента класс `someClass` - если его нет, он добавляется, а если он есть - то удаляется
css('свойство', 'значение')	Устанавливает для указанного свойства выбранного элемента указанное значение (`$("#paddleB").css("top",25);`)
append('новый элемент')	Вставляет внутрь выбранного элемента новый элемент в качестве последнего дочернего (`$("#results").append('<li>Новый элемент списка</li>');`)
prepend('новый элемент')	Вставляет внутрь выбраного элемента новый элемент в качестве первого дочернего
empty()	Удаляет все дочерние элементы у выбранного элемента
remove()	Удаляет элемент из структуры элементов DOM
attr('атрибут','значение')	Устанавливает для атрибута новое значение
removeAttr('атрибут')	Удаляет атрибут у выбранных элементов
children()	Получает все дочерние элементы у выбранных элементов
parent()	Получает все родительские элементы у выбранных элементов
parent()	Получает все родительские элементы у выбранных элементов
hide()	Скрывает выбранные элементы

show()	Отображает выбранные элементы
toggle()	Скрывает видимые элементы и отображает невидимые
animate()	Анимирует элемент

Например, стандартный прием, когда по наведению курсора мыши на изображение, оно увеличивается в размерах, а после отвода курсора - уменьшается:

```
$(function () {
    $("img").mouseover(function () {
        $(this).animate({ height: '+=20', width: '+=20' });
    });
    $("img").mouseout(function () {
        $(this).animate({ height: '-=20', width: '-=20' });
    });
});
```

Сначала при помощи селектора мы выбираем все элементы `img`, затем вешаем на них обработчик наведения курсора `mouseover`. Обработчик события наведения мыши в качестве аргумента принимает анонимную функцию, которая срабатывает при наведении курсора.

В этой функции с помощью ключевого слова `this` мы получаем элемент, на который мы наводим курсор, а затем с помощью функции `animate` мы устанавливаем свойства, которые будут анимироваться при наведении курсора.

Подобным образом работаем второй обработчик `mouseout`. В итоге при наведении курсора изображение увеличится, а при потере изображением фокуса курсора, оно вернется в первоначальные размеры.

8.2.2 AngularJS

AngularJS представляет собой JavaScript-фреймворк, использующий шаблон MVC. Собственно использование MVC является его одной из отличительных особенностей.

Для описания интерфейса используется декларативное программирование, а бизнес-логика отделена от кода интерфейса, что позволяет улучшить тестируемость и расширяемость приложений.

Еще одной отличительной чертой фреймворка является двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом. Таким образом, AngularJS синхронизирует модель и представление.

Цели разработки[

- Отношение к тестированию как к важной части разработки. Сложность тестирования напрямую зависит от структурированности кода.
- Разделение клиентской и серверной стороны, что позволяет вести разработку параллельно.
- Проведение разработчика через весь путь создания приложения: от проектирования пользовательского интерфейса, через написание бизнес-логики, к тестированию

Популярные AngularJS-директивы

С помощью директив AngularJS можно создавать пользовательские HTML-теги и атрибуты, чтобы добавить поведение некоторым элементам.

ng-app

Объявляет элемент корневым для приложения.

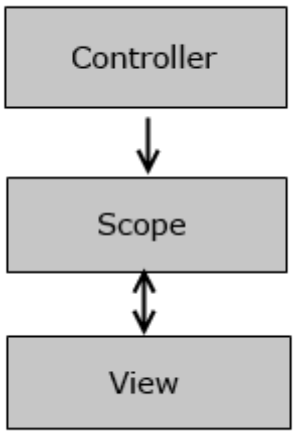
ng-bind

Автоматически заменяет текст HTML-элемента на значение переданного выражения.

ng-model

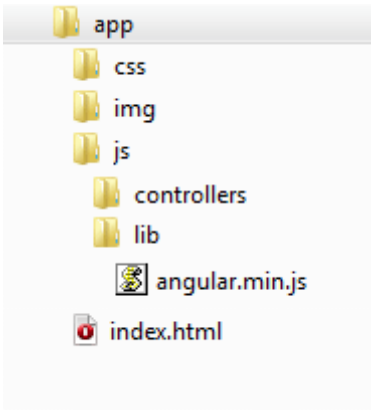
То же что и ng-bind, только обеспечивает двустороннее связывание данных. Изменится содержимое элемента, ангуляр изменит и значение модели. Изменится значение модели, ангуляр изменит текст внутри элемента.

Центральным звеном паттерна MVC, в том числе его реализации во фреймворке AngularJS, является контроллер. Контроллер принимает ввод от пользователя, обрабатывает полученные данные, а также может передавать обратно в представление некоторые данные через объект scope. Схематично подобное взаимодействие можно представить так:



Благодаря привязке мы можем в представлении изменить переданные из контроллера данные модели.

Создадим простое приложение AngularJS. Во-первых, сразу обозначим общую структуру приложения. Создадим в корневом каталоге веб-сервера (в качестве веб-сервера можно выбрать, например, NodeJS или Apache) папку *app*, в которой будут следующие подкаталоги:



Папка *css* будет содержать используемые стили, *img* - изображения, *js* - скрипты. В папке *js* создадим два каталога: controllers - для контроллеров и lib, в который поместим собственно библиотеку angular.min.js и в который в дальнейшем будем класть и другие сопроводительные скрипты.

Также в папке app у нас будет находиться веб-страничка index.html - само представление.

Создадим контроллер. Для этого в ранее созданную папку controllers добавим следующий файл *StateController.js*:

```
function StateController($scope){

    $scope.state = {

        name: 'Russia',
        area: '17 098 246',
        population: '143',
        capital: {
            name: 'Moscow',
            population: '12'
        }
    }
}
```

Контроллер по сути представляет обычную функцию javascript с передаваемым параметром $scope. В этот объект затем в функции передается сложный объект state.

Теперь перейдем к веб-странице index.html. Изменим ее код следующим образом:

```
1
2  <!doctype html>
   <html ng-app>
3  <head>
   <meta charset="utf-8">
4  <link rel="stylesheet" href="css/mystyles.css" />
5  </head>
6  <body>
7  <div ng-controller="StateController">
8  <p>Название страны: {{state.name}}</p>
9  <p>Территория: {{state.area}} кв. км</p>
10 <p>Столица: {{state.capital.name}}</p>
11 </div>
   <script src="js/lib/angular.min.js"></script>
12 <script src="js/controllers/StateController.js"></script>
13 </body>
14 </html>
15
16
```

Поскольку в контроллере мы передали через объект $scope некоторый объект state, то мы можем использовать этот объект в представлении. Также мы можем обращаться к его свойствам, как в данном случае.

С помощью пары двойных фигурных скобок мы создаем выражения. Выражения в AngularJS представляют вкрапления кода javascript в разметку html. В выражении не обязательно обращаться к свойствам переданного через $scope объекта. Например, мы можем передать математическое выражение: `<p>2+3={{2+3}}</p>`. И вместо данного выражения браузер выведет нам его результат. Или, к примеру, в нашей модели есть свойство population для хранения численности населения всего государства и такое же свойство есть у вложенного объекта capital для хранения населения столицы. Мы можем подсчитать численность населения страны вне столицы с помощью следующего выражения: `{{state.population - state.capital.population}}`

Вывод массивов

$scope может передавать не только один одиночный объект, но и набор однотипных объектов одной модели. Например, изменим контроллер StateController следующим образом:

```
1  function StateController($scope){
2
```

```
    $scope.states=[{
        name: 'Россия',
        area: '17 098 246',
        population: '143',
        capital: {
            name: 'Москва',
            population: '12'
        }
    },{
        name: 'Германия',
        area: '357 021',
        population: '81',
        capital: {
            name: 'Берлин',
            population: '3,5'
        }
    },{
        name: 'Франция',
        area: '674 685',
        population: '66',
        capital: {
            name: 'Париж',
            population: '2,2'
        }
    }]
}
```

Здесь у нас фактически массив из трех объектов state. Было бы неудобно в представлении обращаться к каждому из трех (если не больше) объектов и получать значения его свойств. И в данном случае нам может помочь директива **ng-repeat**. Изменим код веб-странички следующим образом:

```
<!doctype html>
<html ng-app>
```

```
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="css/mystyles.css" />
</head>
<body>
<div ng-controller="StateController">
<li ng-repeat="state in states">
<b>{{state.name}}</b>
<p>Население: {{state.population}} млн. чел.</p>
<p>Столица: {{state.capital.name}}</p>
</li>
</ul>
</div>
<script src="js/lib/angular.min.js"></script>
<script src="js/controllers/StateController.js"></script>
</body>
</html>
```

В данном случае с помощью выражения `<li ng-repeat="state in states">` у нас будет создаваться для каждого объекта в наборе states элемент `li`, который будет содержать определенную информацию.