**Программирование в интернет**

**Введение:**

CLI (Common Language Infrastructure) – спецификация общеязыковой инфраструктуры. Определяет архитектуру исполнительной системы и набор представляемых сервисов. Текущая версия CLI 3.5.

CLR (Common Language Runtime) – общеязыковая среда исполнения, виртуальная машина на которой исполняются все приложения, работающие в среде .NET.

MSIL (Microsoft IL) – «высокоуровневый ассемблер» виртуальной машины .NET. Промежуточный язык, разработанный фирмой «Microsoft» для платформы .NET Framework. JIT-компилятор CIL является частью CLR — общей среды выполнения программ, написанных на языках .NET.

FCL (Framework Class Library) – стандартная библиотека классов платформы «.NET Framework». Можно рассматривать, как API CRL.

.NET Framework – программная технология программирования. Это совокупность: CLR ( +JIT).

ASP.NET – технология создания web-приложений и web-сервисов компании Microsoft. Технология является составной частью Microsoft.NET. Является развитием более старой технологии Microsoft ASP. Концептуально является новой.

* ASP.NET компилируется также, как и любое приложение .NET.
* ASP.NET поддерживает три языка C#, VB.NET и MSIL.
* ASP.NET приложение исполняется CLR.
* Web-форма: часть web-приложения
* Web-форма: интерпретируется сервером - результат интерпретации: html и js-код.
* Web-приложения: выполняются на сервере
* Web-приложения: не поддерживают состояние.
* ASP.NET: модель управляемая событиями.

1. HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи произвольных данных. Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

* Потребителей (клиентов), которые инициируют соединение и посылают запрос;
* Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (в частности, для этого используется HTTP-заголовок). Именно благодаря возможности указания способа кодирования сообщения, клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

1. Стартовая строка (англ. Starting line) — определяет тип сообщения;
2. Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. Тело сообщения (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Для версии протокола 1.1 сообщение запроса обязательно должно содержать заголовок Host.

Строка запроса: МЕТОД URI HTTP/версия

Строка ответа: HTTP/версия КОД ПОЯСНЕНИЕ

Статус коды:

* 1хх - инфо
* 2хх - успешно
* 3хх - переадресация
* 4хх - ошибка клиента
* 5хх - ошибка сервера

Методы:

* GET - используется для запроса содержимого указанного ресурса.
* POST - применяется для передачи пользовательских данных заданному ресурсу (не кэш).
* PUT – используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу (не кэш).
* DELETE - удаляет указанный ресурс.
* CONNECT - преобразует соединение запроса в прозрачный TCP/IP-туннель.
* OPTIONS - используется для определения возможностей веб-сервера (не кэш).
* TRACE - возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавили или изменили в запросе.
* PATCH - аналогично PUT, но применяется только к фрагменту ресурса.
* HEAD – применяется для извлечения метаданных, проверки наличия ресурса и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки:

* General Headers (основные заголовки) — должны включаться в любое сообщение клиента и сервера.
* Request Headers (заголовки запроса) — используются только в запросах клиента.
* Response Headers (заголовки ответа) — только для ответов от сервера.
* Entity Headers (заголовки сущности) — сопровождают каждую сущность сообщения.

Параметры:

* Название параметра должно состоять минимум из одного печатного символа (ASCII-коды от 33 до 126). Регистр символов в названиях не имеет значения. Заголовки с неизвестными именами должны игнорироваться. После названия сразу должен следовать символ двоеточия.

Stateless-протокол:

* В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

HTTPS:

* HTTPS (аббр. от англ. HyperText Transfer Protocol Secure) - расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL или TLS. В отличие от HTTP с TCP-портом 80, для HTTPS по умолчанию используется TCP-порт 443.

1. HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

HTML — теговый язык разметки документов. Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначается специальными пометками — тегами. Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных. В этом случае обычно не указывается закрывающий тег (например, тег переноса строки <br> - одиночный и закрывать его не нужно) . Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, атрибут href=”” у ссылки). Атрибуты указываются в открывающем теге.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML <!DOCTYPE…>.

CSS (Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

DOM (от англ. Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями «родительский-дочерний».

1. JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

JavaScript — это язык, который отличается следующими особенностями:

* Динамически типизированный. Типы переменных при JS-разработке задавать необязательно. В одну и ту же переменную можно, например, сначала записать строку, а потом — целое число.
* Слабо типизированный. В отличие от языков с сильной типизацией, языки со слабой типизацией не принуждают программиста, например, использовать в неких ситуациях объекты определённых типов, выполняя, при необходимости, неявные преобразования типов.
* Интерпретируемый. Широко распространено мнение, в соответствии с которым JavaScript является интерпретируемым языком программирования, что означает, что программы, написанные на нём, не нуждаются в компиляции перед выполнением. JS в этом плане противопоставляют таким языкам, как C, Java, Go. На практике же браузеры, для повышения производительности программ, выполняют компиляцию JS-кода перед его выполнением. Этот шаг, однако, прозрачен для программиста, он не требует от него дополнительных усилий.
* Мультипарадигменный. Писать JS-программы можно, используя объектно-ориентированную парадигму, в частности — применяя прототипы и появившиеся в стандарте ES6 классы. Программы на JS можно писать и в функциональном стиле, благодаря тому, что функции здесь являются объектами первого класса. JavaScript допускает и работу в императивном стиле, используемом в C.

## Стандарты:

ECMAScript, или ES, это название стандарта, которым руководствуются разработчики JavaScript-движков, то есть — тех сред, где выполняются JS-программы. Различные стандарты вводят в язык новые возможности, говоря о которых нередко упоминают наименование стандартов в сокращённой форме, например — ES6. ES6 — это то же самое, что и ES2015, только в первом случае число означает номер версии стандарта (6), а во втором — год принятия стандарта (2015).  
  
Сложилось так, что в мире веб-программирования очень долго был актуален стандарт ES3, принятый в 1999 году. Четвёртой версии стандарта не существует (в неё попытались добавить слишком много новых возможностей и так и не приняли). В 2009 году был принят стандарт ES5, который представлял собой прямо-таки огромное обновление языка, первое за 10 лет. После него, в 2011 году, был принят стандарт ES5.1, в нём тоже было немало нового. Весьма значительным, в плане новшеств, стал и стандарт ES6, принятый в 2015 году. Начиная с 2015 года, новые версии стандарта принимают каждый год.  
  
Самой свежей версией стандарта на момент публикации этого материала является ES9, принятая в июне 2018 года.

## Типы данных:

JavaScript иногда называют «нетипизированным» языком, но это не соответствует реальному положению дел. В переменные, и правда, можно записывать значения разных типов, но типы данных в JavaScript, всё-таки, есть. В частности, речь идёт о примитивных и об объектных типах данных.  
  
Для того чтобы определить тип данных некоего значения, можно воспользоваться оператором typeof. Он возвращает строку, указывающую тип операнда.

### Примитивные типы данных:

* number (число)
* string (строка)
* boolean (логическое значение)
* null (специальное значение null)
* undefined (специальное значение undefined)
* symbol (символ, используется в особых случаях, появился в ES6)

Dynamic HTML или DHTML — это способ (подход) создания интерактивного веб-сайта, использующий сочетание статичного языка разметки HTML, встраиваемого (и выполняемого на стороне клиента) скриптового языка JavaScript, CSS (каскадных таблиц стилей) и DOM(объектной модели документа).

Он может быть использован для создания приложения в веб-браузере: например для более простой навигации или для придания интерактивности форм. DHTML может быть использован для динамического перетаскивания элементов по экрану. Также он может служить как инструмент для создания основанных на браузере видеоигр.

1. Ajax – методология построения интерактивного пользовательского интерфейса web-приложения.

Методология Ajax описывает способы разработки страниц сайта, которые могут динамически изменяться на основе данных поступающих с сервера, но без полной перезагрузки страницы.

В основе методологии Ajax лежат следующие технологии: язык HTML, язык JavaScript, язык XML, модель DOM, протокол HTTP, протокол JSON, объект XMLHttpRequest.

HTML– гипертекстовый язык разметки. Интерпретируется браузером. В Ajax динамически изменяется содержимое html-документа.

JavaScript– скриптовый язык, предназначенный для создания сценариев поведения браузера. Интерпретируется браузером. В Ajax html-документ динамически изменяется на стороне клиента с помощью сценариев написанных на языке JavaScript.

DOM – объектная модель, позволяющая сценариям JavaScript получить доступ (читать и изменять содержимое) к элементам html-документа (к атрибутам и содержимому тегов). В Ajax ответ сервера ―встраивается с помощью JavaScript-сценария в загруженную ранее браузером страницу. При этом доступ к элементам html-документа осуществляется в соответствии с моделью DOM.

XMLHttpRequest– специальный API (предопределенный объект), используемый в языке JavaScript для обмена данными между сценарием JavaScript и серверным приложением по протоколу HTTP. В Ajax методы объекта XMLHttpRequest используется для отправки и получения данных между JavaScript-сценарием и серверным приложением. Данные могут получены в виде XML-документа и виде обыкновенного текста (в частном случае могут быть представлены в формате JSON).

1. Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

1. ASP.NET – технология создания web-приложений и web-сервисов компании Microsoft. Технология является составной частью Microsoft.NET. Является развитием более старой технологии Microsoft ASP. Концептуально является новой.

* ASP.NET компилируется также, как и любое приложение .NET.
* ASP.NET поддерживает три языка C#, VB.NET и MSIL.
* ASP.NET приложение исполняется CLR.

Модель событий ASP.NET:

В модели событий ASP.NET разработчик сначала добавляет в веб-форму элементы управления, а затем решает, на какие события он хочет реагировать. Каждый обработчик событий представляет собой отдельный метод, благодаря чему код страниц выглядит аккуратно и организованно.

Эта модель не является новой, но до появления ASP.NET она применялась исключительно в области программирования оконных пользовательских интерфейсов для многофункциональных клиентских приложений. Каким же образом работают события в ASP.NET? На удивление очень просто. Вкратце это выглядит так:

При первом запуске страницы ASP.NET создает объекты этой страницы и ее элементов управления. Далее выполняется код инициализации, после чего страница преобразуется в HTML и возвращается клиенту, а созданные объекты удаляются из памяти сервера.

На каком-то этапе пользователь выполняет действие, инициирующее обратную отправку данных, например, щелкает на кнопке. Тогда страница отправляется серверу вместе со всеми данными формы.

ASP.NET перехватывает эту возвращаемую страницу и снова воссоздает ее объекты, возвращая их в то состояние, в котором они находились тогда, когда эта страница в последний раз отправлялась клиенту.

Далее ASP.NET проверяет, какая именно операция привела к обратной отправке данных, и генерирует соответствующие события (например, Button.Click), на которые разработчик может предусмотреть в своем коде определенную реакцию.

Как правило, разработчики предпочитают выполнять на этом этапе какую-то серверную операцию (вроде обновления базы данных или чтения данных из файла), а затем изменять объекты элементов управления так, чтобы они отображали уже новую информацию.

Измененная страница преобразуется в HTML и возвращается клиенту. Объекты страницы удаляются из памяти.

Другими словами, ASP.NET использует данные формы не только для конфигурирования объектов имеющихся на странице элементов управления, но также и для принятия решения о том, какие события запускать.

Например, заметив, что с момента последней обратной отправки данных текст, отображаемый в текстовом поле, изменился, ASP.NET запустит событие, необходимое для того, чтобы уведомить об этом страницу. Реагировать на это событие или нет — уже выбирать самому разработчику.

Конечно, в системе событий, которая пока что была описана, имеется один пробел. Разработчики Windows-приложений уже давно привыкли иметь дело с полнофункциональной моделью событий, позволяющей коду реагировать на перемещения мыши, нажатия клавиш клавиатуры и мгновенные взаимодействия элементов управления. Но в ASP.NET действия клиента происходят на стороне клиента, а серверная обработка осуществляется на веб-сервере. Это означает, что ответ на событие всегда влечет за собой определенные накладные расходы. Поэтому быстро генерируемые события (вроде событий перемещения курсора мыши) в мире ASP.NET являются совершенно непрактичными.

Свойство AutoPostBack.

1. Web Forms.

* Web-форма: часть web-приложения;
* Web-форма: интерпретируется сервером - результат интерпретации: html и js-код;
* Web-приложения: выполняются на сервере;
* Web-приложения: не поддерживают состояние;
* ASP.NET: модель управляемая событиями.

Жизненный цикл формы:

- page request

- start

- page init

- load

- validation

- postback event handling

- rendering

- unload.

Основные события формы:

- page\_init

- page\_load

- события элементов управления

- page\_prerender

- page\_unload

- page\_disposed.

Viewstate:

Свойство Control.ViewState предлагает словарь для хранения значений между вызовами одной и той же страницы. Это способ, с помощью которого страница сохраняет состояние элементов управления между запросами. Во время обработки страницы текущее состояние страницы и её элементов управления хэшируется в строчку, значение которой хранится в странице как скрытое поле. Когда страница отправляется назад на сервер, происходит разбор этой строки во время инициализации страницы и восстановление состояния страницы и элементов управления, расположенных на ней.

Преимущества использования viewstate:

- Не используются ресурсы сервера.

- Viewstate содержится в структуре самой страницы.

- Простая реализация

- Автоматическое сохранение состояния страницы и элементов управления.

Недостатки использования viewstate: быстродействие. Поскольку view state хранится в самой странице, хранение больших значений приводит к замедлению отображения страницы и отправки её на сервер.

1. IIS (Internet Information Services, до версии 5.1 — Internet Information Server) — проприетарный набор серверов для нескольких служб Интернета от компании Microsoft. IIS распространяется с Windows NT.

Основным компонентом IIS является веб-сервер, который позволяет размещать в Интернете сайты. IIS поддерживает протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP.

Если веб-узел содержит файлы, расположенные на диске, отличном от диска домашнего каталога, или на других компьютерах, то для включения этих файлов в веб-узел следует создать виртуальные каталоги.

1. ASP.NET: серверные элементы управления, html-элементы управления.



Обработка серверных событий элементов управления HTML:

|  |  |
| --- | --- |
| ServerClick | HtmlAnchor, HtmlButton,HtmlInputButton,  HtmlInputSubmit, HtmlInputReset, HtmlInputImage |
| ServerChange | HtmlInputText,HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlInputSelect, HtmlInputTextArea |

1. ASP.NET: серверные элементы управления, полнофункциональные элементы управления.

Calendar

AdRotator

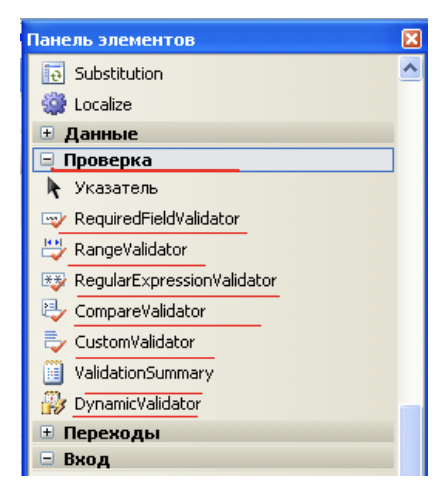
Wizard

MultiView

TreeView

Menu

1. ASP.NET: серверные элементы управления, элементы управления проверкой достоверности.



1. ASP.NET: серверные элементы управления, пользовательские элементы управления.

User Controls: часть aspx-страницы, которая может содержать статический HTML и серверные web-элементы.

Custom Server Control: скомпилированные классы, которые программно генерируют свой html-код. Custom Server Control всегда предварительно компилируются в DLL. Custom Server Control может унаследовать внешний вид у другого серверного элемента или воспроизводить свой собственный.

Основное назначение: создание повторно используемого кода.

User Controls:

ascx: расширение файлов User Controls.

ascx – файлы не могут быть запрошены браузером. Могут только внедряться в другие страницы.

Все User Controls являются наследниками System.Web.UI.UserControl.

Классы System.Web.UI.UserControl и System.Web.UI.UserControl.Page наследники класса TemplateControl.

Создание User Controls: Add New Item/Web User Control.

1. Кэширование

* Кэширование: техника хранения в памяти копий некоторой информации для повторного применения.
* Кэширование: повышение производительности и масштабируемости приложения.
* Кэширование: два вида: кэширование вывода и кэширование данных.
* Кэширование вывода: можно сохранить копию сгенерированной страницы отправленной клиенту; следующий запрос на такую же страницу не потребует ее генерации – используется страница из кэша. Возможно кэширование фрагментов страницы.
* Кэширование данных: программист может сохранить в кэше данные (например, в виде DataSet); страницы могут проверять наличие данных в кэше и использовать ее; возможно кэширование источников данных (DataSource, ObjectDataSource и XMLDataSource). В этом случае управление кэшем осуществляется автоматически.
* Кэширование IIS: не ASP статические ресурсы (например, графика), используемые ASP.NET-приложением, кэшируется IIS и не требует дополнительных усилий от программиста.
* Кэширование части страницы: фрагментарное кэширование (используется для пользовательских элементов управления), послекэшевая постановка (не может быть использована для пользовательских элементов управления.

Для кэширования данных применяется объект Cache.

* Cache безопасный для потоков, если сами объекты являются безопасными (не требует блокирования);
* Элементы Cache удаляются автоматически;
* Элементы Cache поддерживают зависимости;
* Cache привязан к домену приложения (после перезапуска домена – новый кэш);
* Cache не разделяется в серверном кластере.
* Session тоже сохраняется в Cache со скользящей политикой устаревания равной timeout.

Политики устаревания данных в кэше:

- политика скользящего устаревания (задается время неиспользования элемента хранения);

- политика абсолютного устаревания.

1. Темы (skin)

* Единообразие элементов на сайте. Стиль сайта.
* CSS, назначение и применение для стандартизации элементов HTML-страниц. Свойство CssClass.
* CSS работает на стороне клиента и используется браузером.
* Тема: поименованная коллекция свойств элементов управления.
* Тема: может включать css-файлы, графические элементы.
* Темы основаны на элементах управления, а не HTML.
* Темы применяются на стороне сервера.
* Тема могут применяться посредством конфигурационных файлов.
* Skin – файлы оформлений.
* ASP.NET поддерживает глобальные темы
* Если у локальной темы и глобальной совпадают имена – действует локальная тема.

1. Мастер-страницы

MasterPageFile=””

1. Основные системные объекты

**Объект Application** служит для предоставления совместного доступа к информации всем пользователям данного приложения.

**Объект Request** предназначен для получения информации, переданной в HTTP-запросе. Сюда входят параметры, переданные из HTML-форм с помощью метода POST или GET, модули настройки клиента и клиентские сертификаты. Объект **Request** также предоставляет доступ к двоичным данным, отправленным серверу, например, к загруженным на сервер файлам.

**Объект Response** используется для управления отправляемой пользователю информацией. Сюда входит непосредственная отправка данных обозревателю, перенаправление обозревателя по другому URL-адресу или установка значений в модуле настройки клиента.

**Объект Session** служит для хранения сведений о сеансе конкретного пользователя. Переменные, хранящиеся в объекте **Session**, не уничтожаются при переходе пользователя с одной страницы приложения на другую; значения этих переменных сохраняются на протяжении всего времени работы пользователя с приложением. Методы объекта **Session** можно также использовать для принудительного завершения сеанса и для настройки периода ожидания простаивающего сеанса.

HTTP-обработчики (HTTP Handlers) используются для генерации содержимого ответа на HTTP-запрос. Мы можем использовать HTTP-обработчики для настройки обработчик запросов. При обработке одного запроса мы можем задействовать несколько различных модулей, но только один http-обработчик может быть сопоставлен с запросом.

HTTP-обработчик представляет класс, реализующий интерфейс System.Web.IHttpHandler. Данный интерфейс определяет один метод и одно свойство:

ProcessRequest(context): данный метод в качестве параметра принимает объект контекста запроса HttpContext и генерирует ответ клиенту

IsReusable: это свойство указывает, будет ли данный обработчик использоваться другими запросами

Стандартные обработчики:

HttpForbiddenHandler (403)

HttpNotFoundHandler (404)

HttpMetodNotAllowedHandler (405)

HttpNotImplementationHandler (405)

1. ASP.NET: ASMX-сервисы, WSDL, SOAP, прокси, порядок разработки, принципы применения.

Web-сервис – это приложение предоставляющее открытый интерфейс, пригодный для использования другими приложениями в Web.

С самого начала корпорация Microsoft была одним из основных разработчиков стандарта SOAP. В 2002 году в составе самой первой версии ASP.NET 1.0 она представила технологию ASMX (Active Server Method Extended), которая позволила разработчикам в новейшей Visual Studio 2002 легко создавать и потреблять SOAP веб-сервисы.

Плюсы ASMX:

* Легкость в разработке
* Легкость в изучении
* Нет «ада» конфигурирования

Плюсы WCF:

* Очень разнообразные и гибкие возможности транспорта
* Актуальная и развивающаяся технология
* Различные варианты хостинга
* Возможность реализации большого множества стандартов WS-\*

SOAP (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Сейчас протокол используется для обмена произвольными сообщениями в формате XML

Структура SOAP сообщения:

* Envelope - обязательный элемент, который определяет начало и конец сообщения.
* Header - опциональный элемент который может включать в себя любый опциональные (не обязательные) аттрибуты сообщения.
* Body - обязательный элемент, который влючает в себя данные XML, которые определяют сообещние, которые должно быть отправлено.
* Fault - опцилнальный элемент, который предоставляет информацию об ошибках, которые возникают в процессе обработки сообещения.

WSDL (Web Services Description Language) – язык описание Web-сервисов.Последняя версия: WSDL Version 2.0, 26.06.2007

WSDL - это реализация XML.

Каждый WSDL-документ можно разбить на три логические части: 1) определение типов данных, отправляемых и получаемых сервисом xml-сообщений; 2) абстрактные операции – список операций, которые могут выполнены с сообщениями; 3) связывание сервисов – способ, которым сообщение будет доставлено.

Каждый документ WSDL 1.1 можно разбить на следующие логические части:

* определение типов данных (types) — определение вида отправляемых и получаемых сервисом XML-сообщений
* элементы данных (message) — сообщения, используемые web-сервисом
* абстрактные операции (portType) — список операций, которые могут быть выполнены с сообщениями
* связывание сервисов (binding) — способ, которым сообщение будет доставлено

Прокси-сервер (от англ. proxy — «представитель», «уполномоченный»), сервер-посредник — промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером (при этом о посредничестве могут как знать, так и не знать обе стороны), позволяющий клиентам как выполнять косвенные запросы (принимая и передавая их через прокси-сервер) к другим сетевым службам, так и получать ответы. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, e-mail), расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Прокси-сервер позволяет защищать компьютер клиента от некоторых сетевых атак и помогает сохранять анонимность клиента, но также может использоваться мошенниками для скрытия адреса сайта, уличённого в мошенничестве, изменения содержимого целевого сайта (подмена), а также перехвата запросов самого пользователя.

1. HTTP-обработчики

HTTP-обработчики (HTTP Handlers) используются для генерации содержимого ответа на HTTP-запрос. Мы можем использовать HTTP-обработчики для настройки обработчик запросов. При обработке одного запроса мы можем задействовать несколько различных модулей, но только один http-обработчик может быть сопоставлен с запросом.

HTTP-обработчик представляет класс, реализующий интерфейс System.Web.IHttpHandler. Данный интерфейс определяет один метод и одно свойство:

ProcessRequest(context): данный метод в качестве параметра принимает объект контекста запроса HttpContext и генерирует ответ клиенту

IsReusable: это свойство указывает, будет ли данный обработчик использоваться другими запросами

HTTP-обработчик выбирается системой после возникновения события MapRequestHandler. И сразу после выбора обработчика срабатывает событие PostMapRequestHandler.

Непосредственная генерация ответа обработчиком происходит после события PreRequestHandlerExecute - после этого события происходит вызов метода ProcessRequest и генерация ответа. А сразу после генерации ответа возникает событие PostRequestHandlerExecute

Также нужно зарегестрировать обработчик: webconfig или RouteConfig (создать класс и подключить его через add).

1. Ajax – методология построения интерактивного пользовательского интерфейса web-приложения.

Методология Ajax описывает способы разработки страниц сайта, которые могут динамически изменяться на основе данных поступающих с сервера, но без полной перезагрузки страницы.

В основе методологии Ajax лежат следующие технологии: язык HTML, язык JavaScript, язык XML, модель DOM, протокол HTTP, протокол JSON, объект XMLHttpRequest.

HTML– гипертекстовый язык разметки. Интерпретируется браузером. В Ajax динамически изменяется содержимое html-документа.

JavaScript– скриптовый язык, предназначенный для создания сценариев поведения браузера. Интерпретируется браузером. В Ajax html-документ динамически изменяется на стороне клиента с помощью сценариев написанных на языке JavaScript.

DOM – объектная модель, позволяющая сценариям JavaScript получить доступ (читать и изменять содержимое) к элементам html-документа (к атрибутам и содержимому тегов). В Ajax ответ сервера ―встраивается с помощью JavaScript-сценария в загруженную ранее браузером страницу. При этом доступ к элементам html-документа осуществляется в соответствии с моделью DOM.

XMLHttpRequest– специальный API (предопределенный объект), используемый в языке JavaScript для обмена данными между сценарием JavaScript и серверным приложением по протоколу HTTP. В Ajax методы объекта XMLHttpRequest используется для отправки и получения данных между JavaScript-сценарием и серверным приложением. Данные могут получены в виде XML-документа и виде обыкновенного текста (в частном случае могут быть представлены в формате JSON).

**ScriptManager** - серверный элемент управления, расположенный на веб-форме и обеспечивающую базовую функциональность ASP.NET AJAX. Его основная задача - управление остальными элементами ASP.NET AJAX в веб-форме и добавление нужных библиотек сценариев в веб-браузер для поддержки клиентской части ASP.NET AJAX. ScriptManager часто используется для регистрации прочих элементов управления, веб-сервисов и клиентских сценариев.

ScriptManager (как серверный элемент управления) реагирует на события в жизненном цикле страницы ASP.NET и использует эти события для координации работы всех элементов управления, параметров и кода ASP.NET AJAX. ScriptManager подключается к конкретному событию, получает уведомление, когда происходит это событие, и настраивает ряд параметров в зависимости от среды; этот процесс повторяется несколько раз на протяжении цикла рендеринга страницы ASP.NET.

**UpdatePanel** - удобный элемент управления, который позволяет взять обычную страницу с серверной логикой и обеспечить ее обновление в лишенном мерцания стиле Ajax.

Основная идея состоит в том, что веб-страница делится на одну или более отдельных областей, каждая из которых помещается в невидимый элемент управления UpdatePanel. Когда в Updatepanel происходит событие, которое обычно инициирует обратную отправку, элемент управления UpdatePanel перехватывает это событие и выполняет асинхронную обратную отправку. Ниже описан пример того, как это происходит:

Пользователь щелкает на кнопке внутри UpdatePanel.

Определенный клиентский код JavaScript (который был сгенерирован ASP.NET AJAX) перехватывает клиентское событие щелчка и выполняет асинхронный обратный вызов к серверу.

На сервере запускается нормальный жизненный цикл страницы, со всеми обычными событиями. И, наконец, страница визуализируется в виде HTML-кода и возвращается браузеру.

Клиентский код JavaScript принимает полный HTML-код и обновляет каждый элемент управления UpdatePanel на странице, заменяя его текущий HTML-код новым содержимым (если изменение было внесено в содержимое, которое расположено вне элемента управления UpdatePanel, оно игнорируется).

**Timer (автоматическая отправка с заданным периодом)**

**Триггер:**

При использовании условного режима обновления существует еще несколько вариантов инициирования обновления. Один из них - применение триггеров, указывающих элементу UpdatePanel о необходимости его визуализации, когда определенное событие происходит в определенном элементе управления на странице.

Формально UpdatePanel всегда использует триггеры. Все элементы управления в UpdatePanel автоматически становятся триггерами для него. В текущем примере было показано, как это работает с вложенными кнопками - при возникновении события Button.Click выполняется асинхронная обратная отправка. Однако это работает также со стандартным событием любого веб-элемента управления (в соответствии с атрибутом DefaultEvent в коде этого элемента управления), при условии, что событие инициирует обратную отправку страницы.

Например, если поместить элемент TextBox внутрь UpdatePanel и установить свойство TextBox.AutoPostBack в true, событие TextBox.TextChanged инициирует асинхронную обратную отправку, и элемент управления UpdatePanel будет обновлен.

В UpdatePanel создается триггер, в котором указывается идентификатор этого элемента управления (допустим, кнопка, которая не лежит в UpdatePanel) и событие, инициирующее обновление.

1. MVC

MVC: Model-View-Controller – патерн проектирования, в котором приложение состоит из трех взаимодействующих компонентов: модель, представление, контроллер. В правильно разработанном MVC-приложений компоненты относительно независимы. Каждый компонент имеет свою зону ответственности: модель – данные и бизнес-логика; представление – динамическое формирование разметки для отправки клиенту; контроллер – обработка запроса, формирование экземпляра модели, вызов Razor Engine.

Маршрутизация:

App\_Start – RouteConfig– метод RegisterRoutes – routes.MapRoute – прописать метод в Global.asax.

При получении запроса механизм маршрутизации парсит строку URL и помещает значения маршрута в словарь - в объект RouteValueDictionary, доступный через контекст приложения RequestContext.

Контроллер только готовит данные и выбирает, какое представление надо возвратить в качестве объекта ViewResult. Затем уже объект ViewResult обращается к движку представления для рендеринга представления в выходной ответ.

По умолчанию в ASP.NET MVC Core используется один движок представлений - Razor.

Все сущности в приложении принято выделять в отдельные модели. В зависимости от поставленной задачи и сложности приложения можно выделить различное количество моделей.

1. Web API

Класс родитель – ApiController.

В контроллере ValuesContoller нет стандартных методов действий, которые возвращают ActionResult, как в обычных контроллерах. Определенные в контроллере Web API методы сопоставляются с одноименными методами HTTP.

Маршрутизация в Web API

Другой особенностью проекта Web API является наличие файла WebApiConfig.cs (в папке App\_Start), который содержит определения маршрутов Web API.

1. WCF

Web-сервис – это приложение предоставляющее открытый интерфейс, пригодный для использования другими приложениями в Web.

С самого начала корпорация Microsoft была одним из основных разработчиков стандарта SOAP. В 2002 году в составе самой первой версии ASP.NET 1.0 она представила технологию ASMX (Active Server Method Extended), которая позволила разработчикам в новейшей Visual Studio 2002 легко создавать и потреблять SOAP веб-сервисы.

Плюсы ASMX:

* Легкость в разработке
* Легкость в изучении
* Нет «ада» конфигурирования

Плюсы WCF:

* Очень разнообразные и гибкие возможности транспорта
* Актуальная и развивающаяся технология
* Различные варианты хостинга
* Возможность реализации большого множества стандартов WS-\*

Windows Communication Foundation (WCF) — это платформа для построения сервисноориентированных приложений. С помощью WCF, можно отправлять данные как асинхронные сообщения из одной конечной точки службы в другую. Конечная точка службы может входить в постоянно доступную службу, размещаемую в IIS, или представлять службу, размещаемую в приложении. Конечная точка может быть клиентом службы, которая запрашивает данные от конечной точки службы. Сообщения могут представлять одиночный символ или одно слово, отправляемое в формате XML, или иметь вид сложного потока двоичных данных. Далее представлено несколько образцов сценариев.

Защищенная служба для обработки бизнес-транзакций.

Служба, передающая другим объектам текущие данные, такие как отчет о трафике, или другая служба наблюдения.

Служба бесед, которая позволяет двум пользователям общаться и обмениваться данными в реальном времени.

Приложение панели мониторинга, которая опрашивает одну или несколько служб и дает логическое представление полученных данных.

Предоставление доступа к рабочему процессу, реализованному с помощью Windows Workflow Foundation, в виде службы WCF.

Приложение Silverlight для запроса последних каналов данных в службе.

При создании таких приложений до появления WCF, WCF делает разработку конечных точек проще, чем когда-либо. Таким образом WCF позволяет предлагает удобный подход к созданию веб-служб и клиентов веб-служб.

SOAP (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Сейчас протокол используется для обмена произвольными сообщениями в формате XML

Структура SOAP сообщения:

* Envelope - обязательный элемент, который определяет начало и конец сообщения.
* Header - опциональный элемент который может включать в себя любый опциональные (не обязательные) аттрибуты сообщения.
* Body - обязательный элемент, который влючает в себя данные XML, которые определяют сообещние, которые должно быть отправлено.
* Fault - опцилнальный элемент, который предоставляет информацию об ошибках, которые возникают в процессе обработки сообещения.

WSDL (Web Services Description Language) – язык описание Web-сервисов.Последняя версия: WSDL Version 2.0, 26.06.2007

WSDL - это реализация XML.

Каждый WSDL-документ можно разбить на три логические части: 1) определение типов данных, отправляемых и получаемых сервисом xml-сообщений; 2) абстрактные операции – список операций, которые могут выполнены с сообщениями; 3) связывание сервисов – способ, которым сообщение будет доставлено.

Каждый документ WSDL 1.1 можно разбить на следующие логические части:

* определение типов данных (types) — определение вида отправляемых и получаемых сервисом XML-сообщений
* элементы данных (message) — сообщения, используемые web-сервисом
* абстрактные операции (portType) — список операций, которые могут быть выполнены с сообщениями
* связывание сервисов (binding) — способ, которым сообщение будет доставлено

Прокси-сервер (от англ. proxy — «представитель», «уполномоченный»), сервер-посредник — промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером (при этом о посредничестве могут как знать, так и не знать обе стороны), позволяющий клиентам как выполнять косвенные запросы (принимая и передавая их через прокси-сервер) к другим сетевым службам, так и получать ответы. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, e-mail), расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Прокси-сервер позволяет защищать компьютер клиента от некоторых сетевых атак и помогает сохранять анонимность клиента, но также может использоваться мошенниками для скрытия адреса сайта, уличённого в мошенничестве, изменения содержимого целевого сайта (подмена), а также перехвата запросов самого пользователя.

1. ASP.NET CORE

* ASP.NET Core: программная платформа, разработанная Microsoft и предназначается для разработки web-приложений. Является развитием технологии OWIN (The Open Web Interface for.NET). Katana – OWIN-совместимый хост, разработанный Microsoft.
* ASP.NET Core: может работать над .NET Core и над полной .NET.
* ASP.NET Core: если над .NET Core, кроссплатформенность, Windows, Linux, Mac OS.
* ASP.NET Core: последняя версия ASP.NET Core 2.0, планируется 2.1 в 2018
* ASP.NET Core: приложения могут работать под IIS (Windows) или под web-сервером Kestrel (кроссплатформенный вариант).
* ASP.NET Core: благодаря модульности, все отдельные компоненты загружаются через Nuget, нет необходимости применять библиотеку System.Web.dll (основная библиотека для приложений ASP.NET).
* ASP.NET Core: типы приложений MVC UI, Web API, Web Pages. Web Forms не поддерживается.
* ASP.NET Core: Startup.cs – класс Startup, содержащий логику обработки входящих запросов

Статические файлы:

Wwwroot

Startup – Configure – UseStaticFiles

Стартовые страницы:

App.UseDefaultFiles

Заголовки:

Request.Headers

Файлы для скачивания:

App.UseDirectoryBrowser – FileProvider – путь к файлам (должны быть в проекте)

Вывод в журнал

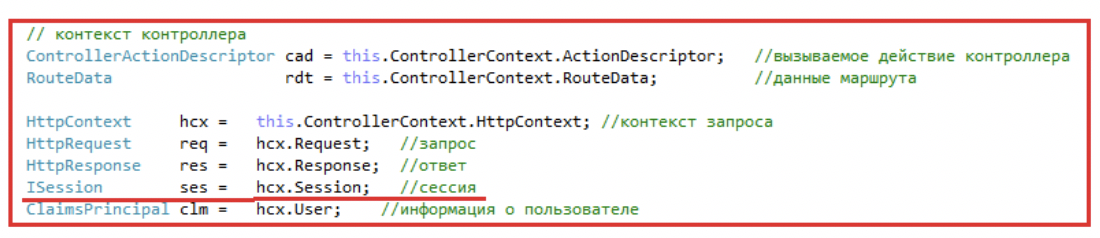
Configure (… , ILoggerFactory loggerFactory)

Фреймворк MVC по умолчанию подключает в приложение маршрутизацию на основе атрибутов. Такой тип маршрутизации еще называетсяAttribute-Based Routing. Атрибуты предоставляют более гибкий способ определения маршрутов. Маршруты, определенные с помощью атрибутов, имеют приоритет по сравнению с маршрутами, определенными в классе Startup.

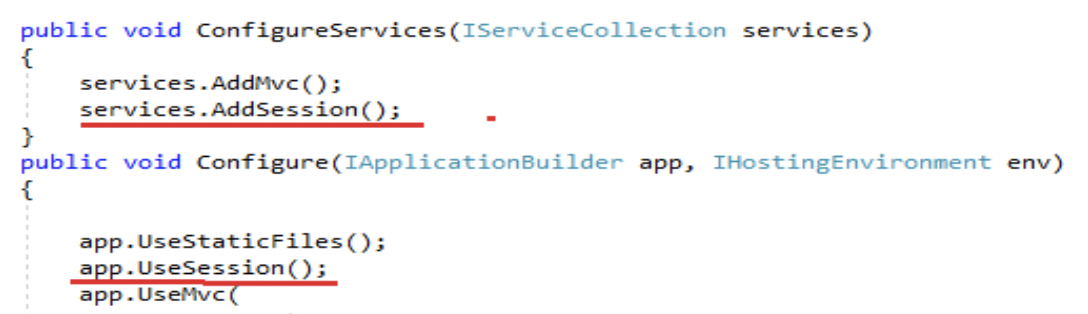
Для определения маршрута, необходимо использовать атрибут [Route].

1. Controller

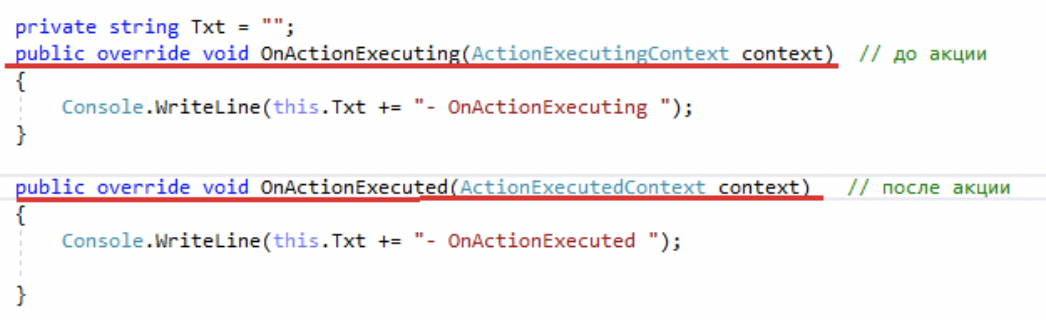
Контекст



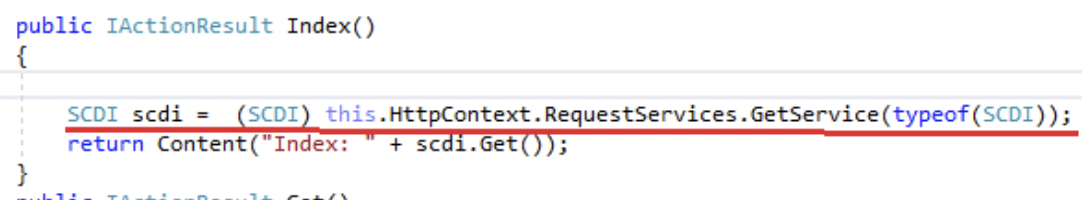
Поддержка сессий:



OnAction

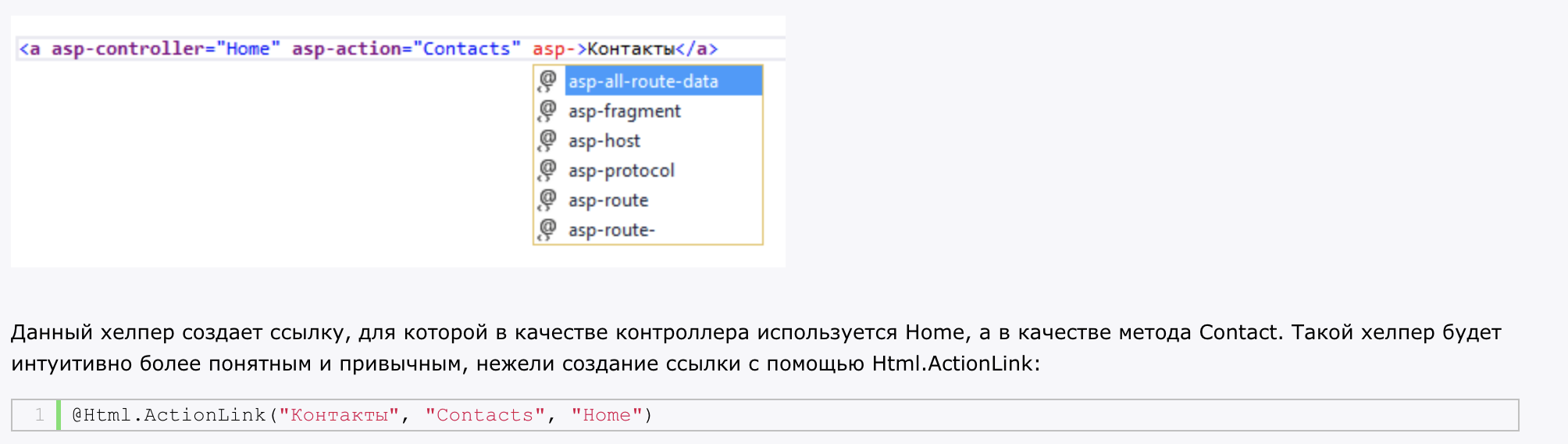


Динамическое добавление зависимостей:



1. View

View: \_ViewImports.cshtml; здесь размещаются общие директивы @using и директивы, связанные с TagHelpers, размещается в родном для представлений директории или в SHARED.



1. Model

ModelState: объект, для хранения состояния представления. Заполнение ModelState при заполнении параметров action.

View Model: объект, применяемый для передачи данных в представлении.

