МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Физико-технический факультет**

**Кафедра теоретической физики и компьютерных технологий**

**Реферат на тему**

**Разработка веб-приложений на платформе ASP.NET**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лохмытко Дмитрий Олегович

Курс 3

Направление 09.03.02 Информационные системы и технологии

Преподаватель

канд. биологических наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Н.Куликова

Краснодар 2017

*Web-приложения* представляют собой особый тип программ, построенных по архитектуре "клиент-сервер". Особенность их заключается в том, что само Web-приложение находится и выполняется на сервере - клиент при этом получает только результаты работы. Работа приложения основывается на получении запросов от пользователя (клиента), их обработке и выдачи результата. Передача запросов и результатов их обработки происходит через Интернет ( рис.1.1).



**Рис. 1.1.**  Архитектура Web-приложения

Отображением результатов запросов, а также приемом данных от клиента и их передачей на сервер обычно занимается специальное приложение — браузер (Internet Expolrer, Mozilla, Opera и т. д.). Как известно, одной из функций браузера является отображение данных, полученных из Интернета, в виде страницы, описанной на языке HTML, следовательно, результат, передаваемый сервером клиенту, должен быть представлен на этом языке.

На стороне сервера Web-приложение выполняется специальным программным обеспечением (Web-сервером), который и принимает запросы клиентов, обрабатывает их, формирует ответ в виде страницы, описанной на языке HTML, и передает его клиенту. Одним из таких Web-серверов является Internet Information Services (IIS) компании Microsoft. Это единственный Web-сервер, который способен выполнять Web-приложения, созданные с использованием технологии ASP.NET.

В процессе обработки запроса пользователя Web-приложение компонует ответ на основе исполнения программного кода, работающего на стороне сервера, Web-формы, страницы HTML, другого содержимого, включая графические файлы. В результате, как уже было сказано, формируется HTML-страница, которая и отправляется клиенту. Получается, что результат работы Web-приложения идентичен результату запроса к традиционному Web-сайту, однако, в отличие от него, Web-приложение генерирует HTML-код в зависимости от запроса пользователя, а не просто передает его клиенту в том виде, в котором этот код хранится в файле на стороне сервера. То есть Web-приложение динамически формирует ответ с помощью исполняемого кода — так называемой исполняемой части.

За счет наличия исполняемой части, Web-приложения способны выполнять практически те же операции, что и обычные Windows-приложения, с тем лишь ограничением, что код исполняется на сервере, в качестве интерфейса системы выступает браузер, а в качестве среды, посредством которой происходит обмен данными, — Интернет. К наиболее типичным операциям, выполняемым Web-приложениями, относятся:

* прием данных от пользователя и сохранение их на сервере;
* выполнение различных действий по запросу пользователя: извлечение данных из базы данных (БД), добавление, удаление, изменение данных в БД, проведение сложных вычислений;
* аутентифицирование пользователя и отображение интерфейса системы, соответствующего данному пользователю;
* отображение постоянно изменяющейся оперативной информации и т. д.

Краткое описание архитектуры ASP.NET и .NET Framework

ASP.NET — это платформа для создания Web-приложений и Web-сервисов, работающих под управлением IIS. Сегодня существуют другие технологии, позволяющие создавать Web-приложения. К ним относятся прежде всего, очень популярные сегодня языки PHP и PERL, более старая и менее популярная технология CGI и т. д. Однако ASP.NET отличается от них высокой степенью интеграции с серверными продуктами, а также с инструментами Microsoft для разработки доступа к данным и обеспечения безопасности. Кроме того, ASP.NET позволяет разрабатывать Web- и Windows-приложения, используя очень похожие технологические цепочки, одинаковые языки программирования, технологии доступа к данным и т. д. Более того, базовые языки программирования, с помощью которых сегодня возможна разработка Web-приложений, являются полностью объектно-ориентированными, что делает разработку исполнимой части, а также ее модификацию, обслуживание, отладку и повторное использование гораздо более простым занятием, чем в других технологиях. Существует достаточно большой перечень сильных сторон использования ASP.NET для создания сложных Web-приложений. Целью данного курса не является описание всех сильных и слабых сторон этой платформы.

Заметим лишь, что ASP.NET функционирует исключительно на серверах Windows, так как требует наличия IIS. Для создания Web-приложений, не требующих IIS, а использующих, например, Web-сервер Apache и работающих на серверах под управлением операционных систем, отличных от Windows, применяются другие технологии.

Важным моментом в понимании архитектуры ASP.NET является тот факт, что она является частью инфраструктуры .NET Framework.

*ASP.NET — это технология создания динамических веб-приложений. Она является частью платформы .NET Framework. Приложения ASP.NET можно писать на любом языке, совместимом с технологиями .NET. В их число входят Visual Basic, C# и J#. Страницы ASP.NET (веб-формы) проходят компиляцию, что обеспечивает лучшую производительность по сравнению с приложениями на основе сценариев.*

ASP.NET значительно упрощает создание веб-приложений, имеющих практическое применение. Серверные элементы управления ASP.NET позволяют применять декларативную модель программирования, схожую с HTML, в которой для создания динамической страницы требуется гораздо меньше кода, нежели при использовании классической технологии ASP. Отображение данных, проверка ввода и загрузка файлов происходят гораздо проще. Главное же преимущество страниц ASP.NET состоит в том, что они работают в любом обозревателе, включая обозреватели Netscape, Opera, AOL и Internet Explorer.

ASP.NET позволяет вам развить собственные навыки программирования. В отличие от классического ASP, где поддерживались только интерпретируемые VBScript и JScript, технология ASP.NET поддерживает более 25 языков .NET languages (поддержка Visual Basic, C# и JScript.NET является встроенной, дополнительных средств не потребуется), что дает вам полную свободу в выборе языка программирования.

Работать со всем многообразием функций ASP.NET можно в любом текстовом редакторе, даже в блокноте. Впрочем, [Visual Studio](http://msdn2.microsoft.com/vstudio/default.aspx) может дать дополнительные возможности веб-разработки. Теперь веб-формы ASP.NET можно создавать с использованием графической модели: вы просто щелкаете и перетаскиваете элементы. Вам в помощь имеется полная поддержка кодирования, включающая завершение операторов и цветовую разметку. Visual Studio также обеспечивает разностороннюю поддержку отладки и развертывания веб-приложений ASP.NET.

**Архитектура веб-приложения в ASP.NET. Типы приложений**

В ASP.NET можно создавать различные типы проектов ASP.NET, в том числе веб-сайты, веб-приложения, веб-службы и серверные элементы управления Ajax.

Между проектами *веб-сайтов* и проектами *веб-приложений* существует разница. Некоторые функции, такие как MVC и некоторые средства автоматизации веб-развертывания, работают только с проектами веб-приложений. Другие функции, такие как платформы динамических данных, работают как с проектами веб-сайтов, так и проектами веб-приложений.

Модель проекта веб-сайта по умолчанию использует структуру каталога файлов для определения содержимого проекта. В этой модели отсутствует файл проекта, а все файлы в каталоге являются частью проекта.

В проекте веб-приложения частью проекта являются только файлы, на которые имеются явные ссылки в файле проекта решения. Эти файлы отображаются в Обозревателе решений и являются единственными файлами, компилируемыми при построении.

Файл проекта веб-приложения упрощает реализацию некоторых скриптов. Например, можно разделить одно приложение ASP.NET на несколько проектов Visual Studio, создав ссылки на файлы в различных файлах проектов. Также можно легко исключить файлы из проекта.

Проекты веб-приложений следует использовать при необходимости выполнения одного из перечисленных ниже действий.

* Перенос крупных приложений из Visual Studio .NET 2003 в Visual Studio 2010.
* Управление именами выходных сборок.
* Использование изолированных классов для создания ссылок на классы страниц и классы пользовательских элементов управления.
* Построение веб-приложений с использованием нескольких веб-проектов.
* Добавление действий, выполняемых до и после построения, во время компиляции.
* Добавление действий, выполняемых до и после построения, во время компиляции.

Каждое веб-приложение (и веб-сайт), разрабатываемое на ASP.NET состоит из информационной части, программного кода и сведений о конфигурации.
**Информационная часть** содержит статические и динамические элементы html страницы и реализуется в виде Web-форм. Статические элементы представляют собой типичные элементы языка html, динамические же компонуются программным кодом приложения во время его выполнения (например, запросы к базе данных).
**Программный код** реализует логику, определенную в процедурах обработки данных, которые определяют реакцию приложения на запросы пользователя. Программный код исполняется сервером и взаимодействует с динамическими элементами информационной части для формирования отклика приложения.

**Сведения о конфигурации** представляют собой файлы, содержащие параметры, определяющие способ исполнения приложения на сервере, параметры безопасности, реакцию приложения на возникающие ошибки.

Основным элементом Web-приложения является Web-форма (или Web-страница), которая, с одной стороны, похожа на Windows-форму, т. к. позволяет размещать внутри себя различные элементы управления, способные отображать данные и реагировать на действия пользователя, а с другой — представляет собой HTML-страницу, т. к. содержит все ее атрибуты. Описания элементов управления, упомянутых ранее, представляются в коде HTML-страницы в виде специальных тегов.
При разработке Web-приложений на основе ASP.NET возможны два варианта организации Web-форм.

*В первом случае* весь код информационной части и программная часть хранятся в одном файле с расширением .aspx. Программный код при этом помещается в так называемые блоки сценариев. При этом сохраняется возможность использования всех принципов современного программирования, таких как реакция на события элементов управления, подпрограммы и т.д. Эту модель целесообразно использовать при создании простых Web-приложений, поскольку в этом случае все хранится в одном пакете.

*Во втором случае* каждая Web-страница разделяется на две части: Web-форму и файл, содержащий программный код. При этом форма, как и в первом случае, сохраняется в файле с расширением .aspx, а программный код — в файле с расширением .cs. Такая модель обеспечивает лучшую организацию элементов Web-приложения за счет отделения пользовательского интерфейса от программной логики.

На рис. 3.2. показан типовой сценарий взаимодействия элементов Web-приложения друг с другом и с клиентом, осуществляющим запрос формы этого приложения.



**Рис. 3.2.**  Типовой сценарий взаимодействия элементов Web-приложения с клиентом

Как видно из рисунка, при обращении клиента к Web-приложению последнее запускается на сервере IIS. Запущенное приложение формирует отклик. Для этого на сервере создается экземпляр запрошенной Web-формы, она генерирует HTML-текст отклика, который и передается браузеру клиента. Сразу после этого экземпляр Web-формы уничтожается. Пользователь, получив HTML-страницу, сгенерированную приложением, имеет возможность заполнять различные поля формы (тестовые поля, переключатели и т. п.). После заполнения всех необходимых полей формы пользователь инициирует отправку данных, введенных им в страницу, обратно на сервер. Это происходит за счет использования технологии обратной отсылки, которая вызывается при выполнении определенных действий (например, нажатия на кнопку). Получив данные от пользователя, сервер создает новый экземпляр Web-формы, заполняет его полученными данными и обрабатывает все необходимые события. По окончании обработки сервер формирует HTML-код ответа и отправляет его клиенту, а затем уничтожает экземпляр Web-формы.

В момент окончания работы с Web-приложением пользователь либо закрывает браузер, либо переходит на другую интернет-страницу. В этот момент завершается сеанс работы пользователя с данным приложением, однако само приложение может быть завершено сервером не сразу после окончания последнего сеанса работы пользователя. Это связано с управлением распределением памяти платформой .NET Framework, которая основана на периодической проверке ссылок объектов. Если в результате такой проверки обнаружится, что объект больше не используется, сервер уничтожает его, освобождая таким образом занимаемую им память. Поэтому нельзя точно сказать, когда именно наступит событие Application\_End для данного Web-приложения.

Такой принцип организации выполнения Web-приложений хорошо подходит для масштабируемых приложений с интенсивным сетевым обменом. Однако у него есть и недостатки. В частности, оказывается невозможным сохранять данные, принадлежащие форме, даже в течение работы пользователя с приложением. Т. е. если мы захотим создать некую переменную, хранящую, например идентификатор заказа, с которым мы в данный момент работаем, сделать это будет невозможно, т. к. форма после отправки клиенту сразу же уничтожается. Чтобы обойти этот недостаток, ASP.NET использует специальный механизм для сохранения данных, введенных в элементы управления Web-формы. Согласно этому принципу, в рамках каждого запроса на сервер отправляются все данные, которые были введены в элементы управления. При этом на сервере возникает событие Page\_Init, целью которого является создание Web-формы и ее инициализация. В процессе инициализации в элементы управления созданной формы записываются переданные от клиента данные. Теперь эти данные становятся доступны приложению посредством обработки события Page\_Load, возникающего при каждом обращении к странице.