ERASMUS+
Enriching lives, opening minds

Институт Бизнеса

Белорусского Государственного Университета

WEB-технологии

IESED

SCHOOL OF BUSINESS OF BSU

Введение

Технологии World-Wide Web первоначально спроектированные в CERN (European Organization for Nuclear Research) как средство доставки документов из одного научного заведения в другое, сейчас используются как платформа для сложных интерактивных приложений, которые медленно, но верно вытесняют инсталлируемые приложения (installable applications).

Этому способствуют преимущества, которыми обладают WEB-приложения:

- постоянный доступ с различных устройств,
- автоматическое обновление,
- возможность интеграции различных приложений, написанных для различных платформ

Однако, создание WEB-приложений требует несколько иных подходов, нежели создание традиционных инсталлируемых приложений и на сегодняшний день невозможно без интеграции самых различных подходов и технологий.

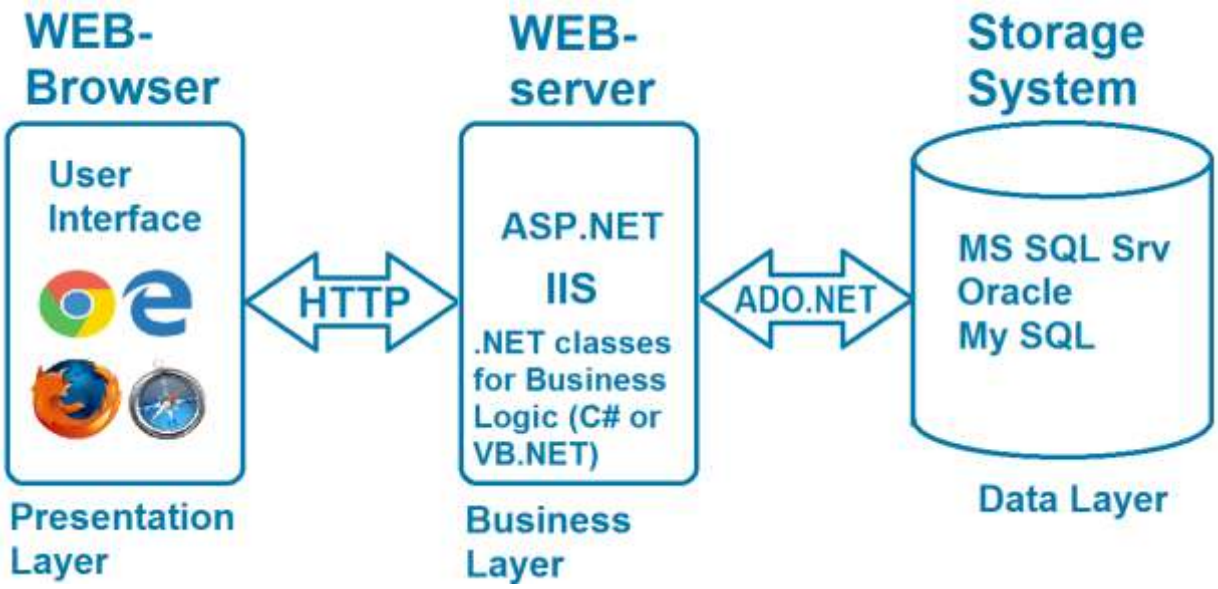
Данный курс знакомит этими различными WEB-технологиями и дает навыки создания WEB-приложений. Студенты изучают теоретически и получают практические навыки с языками разметки, скриптовыми языками, сетевыми протоколами, графикой, программированием на основе событий, объектно-ориентированным программированием, базами данных.

Т.е. те технологии, которые позволяют строить современные WEB-приложения.

Есть трудность с выбором технологий WEB-программирования - т.к. в этой области изменения происходят наиболее быстро. Билл Гейтс в своей книге “Business @ the Speed of Thought” говорил, что Micosoft практически полностью обновляет эти технологии раз в 3 года. И курс приходится обновлять очень часто - те технологии, которые являются мэйнфреймовыми сегодня, через год могут потерять свои позиции.

Однако кое-что остается относительно устойчивым – концепции программирования и архитектура приложений.

Мы исходим из позиции, что понимание концепций и архитектуры важнее знаний конкретных методов кодирования, которые претерпевают изменение может быть быстрее, чем хотелось бы.



Архитектура WEB-приложения обычно включают следующие составные части:

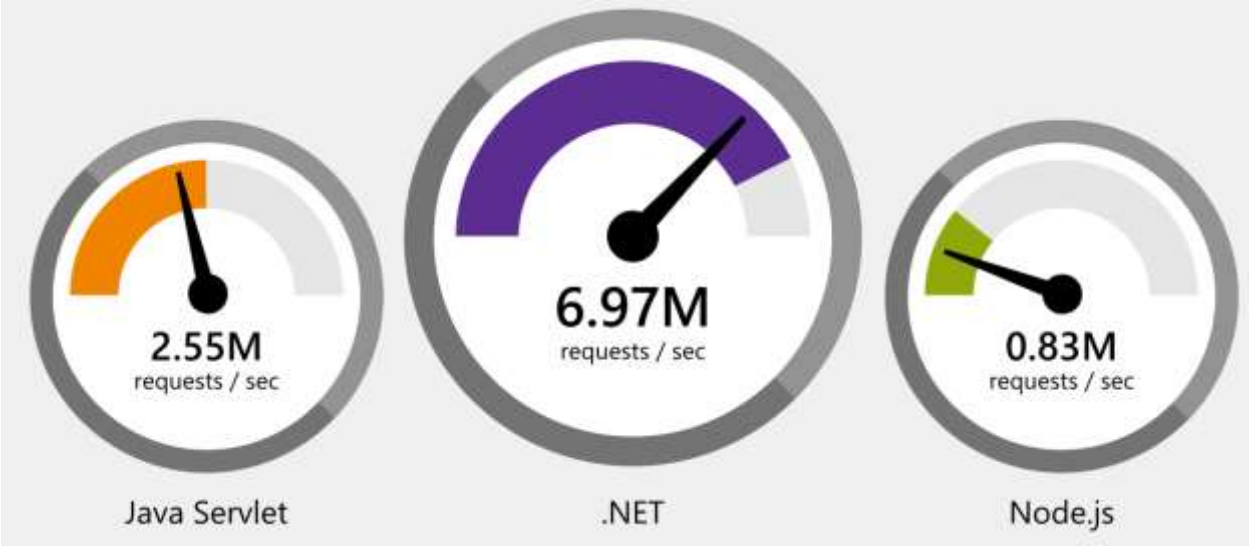
- Интерфейс приложения в Web Browser'e (Front End)
- Серверная часть (Back End)
- Система хранения данных (Data Storage)

На уровне интерфейса (Presentation Layer) изучаются

HTML/CSS/JavaScript и Document object Model (DOM) - Document structure Browser software а также концепции Model View Controller, Single page applications (фреймворки Angular JS и Vue.JS).

На уровне сервера (Business Layer) изучается технология ASP.NET на базе C# и VB.NET.

На уровне данных (Data Layer) изучается технология взаимодействия с БД ADO.NET. Такой выбор обусловлен тем, что на сегодняшний день платформа ASP.NET работает быстрее, чем любой популярный веб-фреймворк (ASP.NET performs faster than any popular web framework)



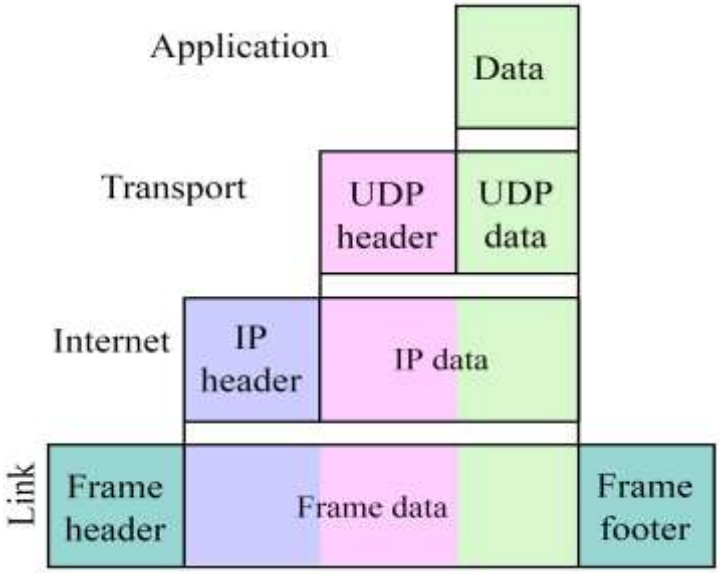
Если через год или 2 ситуация изменится и Node.js превзойдет ASP.NET по скорости или популярности, то на уровне сервера можно будет изучать Node.js.

А пока мы представляем то, что нам представляется лучшим и наиболее полезным для студентов на сегодняшний день.

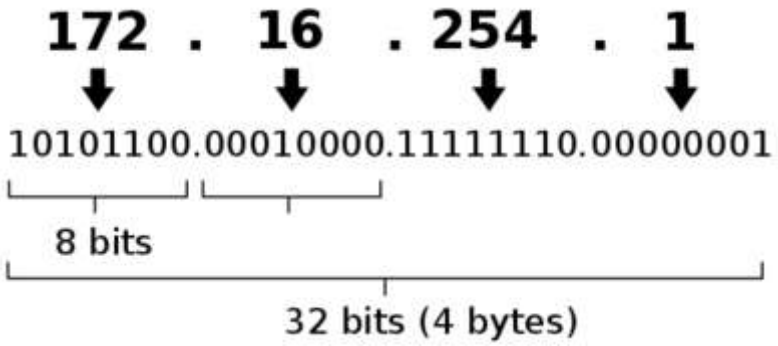
Базовые концепции WEB-технологий

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической настройки узла) - сетевой протокол, позволяющий сетевым устройствам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP

TCP/IP — совокупность стандартов и правил, определяющих, как данные передаются по сети.



IP адрес - Internet Protocol Address — уникальный идентификатор (адрес) устройства,



подключённого к локальной сети или интернету. IP-адрес представляет собой 32-битовое (по версии IPv4).

Доменное имя – более понятно, чем IP-адрес. Доменные имена дают возможность адресации интернет узлов в более удобной для человека форме.

Компьютерная система для получения информации о IP-узлах по имени хоста называется Domain Name System (DNS).

С помощью утилиты **ping** можно узнать IP-адрес по доменному имени.

Утилита tracert (traceroute в Apple) — это служебная компьютерная программа, предназначенная для определения маршрутов следования данных в сетях TCP/IP.

Порт (англ. port) — натуральное число, записываемое в заголовках протоколов транспортного уровня. С помощью порта определяется получатель пакета в пределах одного хоста. Порт указывается через двоеточие после адреса устройства

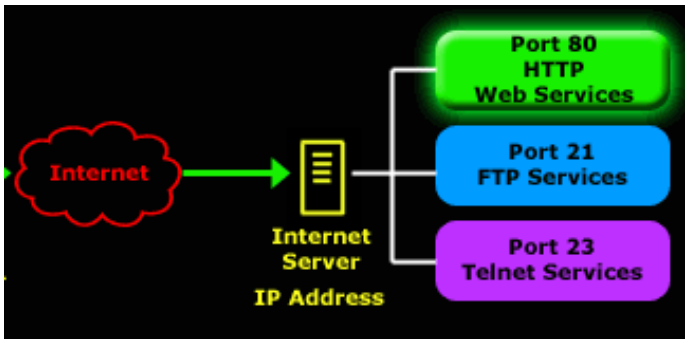
HTTP порт 80,

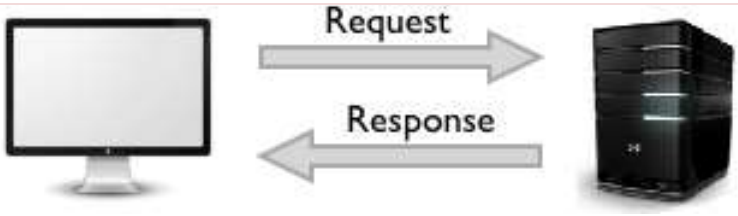
HTTPS — порт **443**.

FTP— порт 21.

SMTP — порт **25**

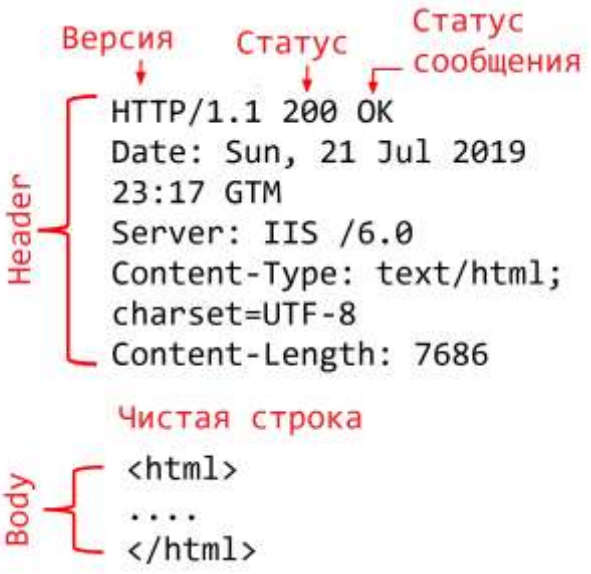
SMTP — порт **23**





Структура HTTP-запроса

Структура HTTP-ответа



Коды состояния

1xx: Информационные сообщения

100 continue, что означает, что клиент ещё отправляет оставшуюся часть запроса.

2xx: Сообщения об успехе

200 OK

Если клиент получил код из серии 2xx, то запрос ушёл успешно.

3xx: Перенаправление

301 Moved Permanently: ресурс теперь можно найти по другому URL адресу.

4xx: Клиентские ошибки

404 Not Found. Ресурс не найден на сервере.

401 Unauthorized: для совершения запроса нужна аутентификация. Информация передаётся через заголовок Authorization.

403 Forbidden: сервер не открыл доступ к ресурсу.

5xx: Ошибки сервера

500 Internal Server Error.

503 Service Unavailable: это может случиться, если на сервере произошла ошибка или он перегружен.

Браузеры

Предназначение WEB-браузера - чтение документов HTML и компоновка их в визуальные веб-страницы (с возможностью проигрывания музыки и отображения видеофайлов). Браузер не отображает HTML тегов, но использует теги, чтобы интерпретировать содержание страницы и форматировать текст нужным образом.

На сегодняшний день существуют нижеследующие веб-браузеры (в которых необходимо проверять Ваш WEB-дизайн, чтобы Ваш сайт отражался в них корректно):

- Internet Explorer (от Microsoft)
- Firefox (от Mozilla)
- Chrome (от Google)
- Safari (от Apple)
- Opera (Opera из Норвегии)

HTML

HTML представляет собой язык гипертекстовой разметки. Он является основным строительным материалом WEB-страниц.

HTML состоит из текста и тегов, заключенных в скобки, которые указывают как отображать текст и графику.

HTML теги обычно используют парами <html></html>.

Первый тег называется открывающим тегом, например <html> <script>

Второй – закрывающим </html> </script> - обозначающий конец зоны действия тэга. В промежутке между этими тегами веб-дизайнеры могут добавить текст, таблицы, изображения и т.д.

Пример HTML-страницы:

```
<html>
 <body>
  <h1>Заголовок</h1>
   <p>Параграф</p>
 </body>
</html>
```

0.html

Более подробно смотрите: http://asp.net.by/html/

CSS

CSS - Cascading Style Sheets, что означает дословно "каскадные таблицы стилей". С помощью CSS описывается стиль всего документа или сайта в целом, либо его отдельных элементов.

Стили описываются в теге

<style>

</style>

Их можно записать в отдельный файл и загрузить его в html страницу следующим образом:

<link REL="STYLESHEET" TYPE="text/css" HREF="ss.css">

Более подробно смотрите: http://asp.net.by/css/

Адаптивная разметка

Адаптивная разметка - это такая разметка, которую можно настроить под различные размеры экранов. Осуществляется создание адаптивной разметки с помощью медиа запросов, которые появились в спецификации CSS3 и в настоящий момент поддерживаются всеми основными браузерами.

Необходимость адаптивной разметки определяется тем, что в настоящий момент число пользователей, использующих мобильный трафик достигло 60%. Мобильный трафик становится более значимым и владельцы сайтов должны считаться с этой статистикой. В первую очередь сайт должен быть оптимизирован под мобильные устройства – Mobile First.

Принципами Mobile First являются:

1) Самое важное содержание должно быть показано в первую очередь

2) Веб сайт должен быть легким и грузиться быстро

3).Дополнительная информация должна грузиться по требованию пользователя.

Более подробно смотрите: http://asp.net.by/mobile/

CSS FlexBox

Считаются лучшим средством для создания адаптированного сайта для небольшого экрана, даже когда размер экрана неизвестен и (или) динамичен.

Более подробно смотрите: http://asp.net.by/mobile/

JavaScript

JavaScript представляет собой объектно-ориентированный язык, созданный для создания динамических WEB-страниц.

Что позволяет создавать сложные динамические сайты.

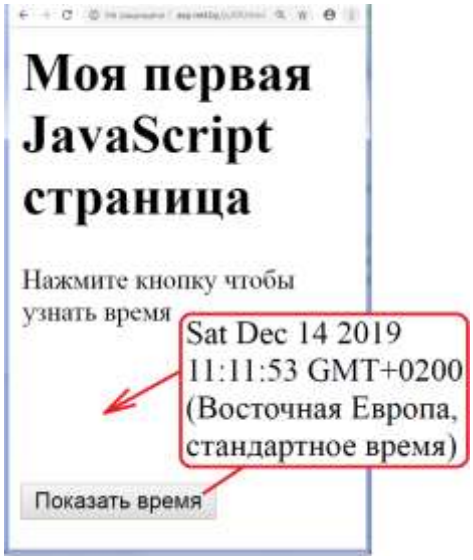
Вот пример простой HTML страницы с JavaScript кодом:

```
<html>
<body>
```

http://asp.net.by/js/00.html

```
<h1>Моя первая JavaScript страница</h1>
<p>Нажмите кнопку чтобы узнать время</p>
<p id="date_area"></p>
<button type="button"
 onclick="cmdDate()">
Показать  время
</button>
<script>
  function cmdDate()
  {
document.getElementById("date_area")
                    .innerHTML = Date();
  }
 </script>
 </body>
</html>
```

Показать время



ECMAScript был создан для стандартизации JavaScript.

Позволяет использовать Java подобный синтаксис при определении классов:

```
<script>
class ACat {
  constructor(n) {
    this.name = n;        //свойство
  }
}

mycat = new ACat("Барсик");
```

http://asp.net.by/ES6/07.htm

document.write("Моего кота зовут " + mycat.**name**);

</script>

Более подробно смотрите: http://asp.net.by/es6/

JSON

Механизм, который используется для пересылки данных от сервера к клиенту.

Это формат для обмена данными.

Правила синтаксиса JSON

- Данные в парах имя / значение ("lastName":"**Musk**")
- Данные разделяются запятыми ("firstName":"**Elon**", "lastName":"**Musk**")
- Фигурные скобки определяют предметы
  { "firstName":"**Elon**", "lastName":"**Musk**" }
- Квадратные скобки содержат массивы {"managers":[.....]}

Пример:

XML		**JSON**
< managers >		{"managers":[
<manager> <firstName>**Bill**</firstName> <lastName>**Gates**</lastName> </manager>		{ "firstName":"**Bill**", "lastName":"**Gates**" },
<manager> <firstName>**Mike**</firstName> <lastName>**Dell**</lastName> </manager>		{ "firstName":"**Mike**", "lastName":"**Dell**" },

<manager>
<firstName>**Elon**</firstName>
<lastName>**Musk**</lastName>
</manager>
		{
		"firstName":"**Elon**",
		"lastName":"**Musk**"
		}

</managers>		]}

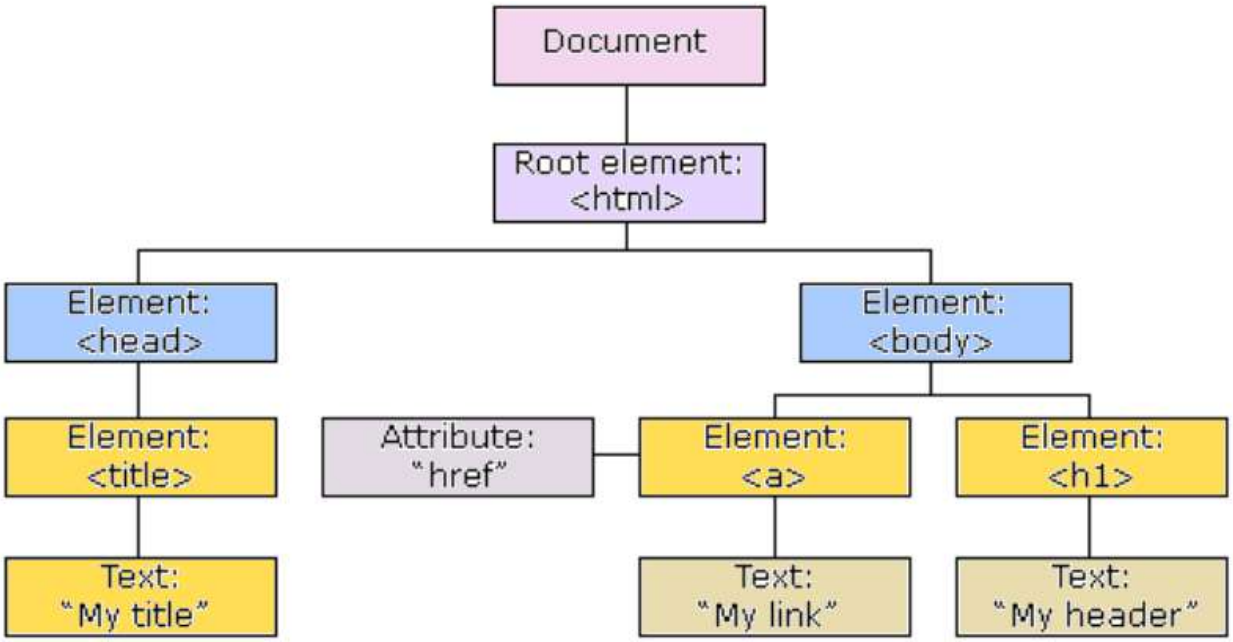
Формат JSON переводится в формат JavaScript object с помощью оператора:
var myObj = JSON.parse(usManagers);

myObj.firstName myObj.lastName

Более подробно о формате JSON см.: http://asp.net.by/JSON/

DOM

DOM - кросс-платформенный и независимый интерфейс, который представляет XML или HTML документ в виде древовидной структуры, в которой каждый узел представляет собой объект, представляющие собой часть документа.



JavaScript может запрашивать или изменять HTML документ

```
<html>
<body>
<form name="fr" action="">
Name:  <input type="text" name="fname" value="Ann" size="2">
<input type="button" value="Go"
onClick='javascript:document.getElementsByName("fname")[0].value="Alex"'>  см.1
onClick="javascript:document.fr.fname.value='Alex'">  см.2
</form>
</body>
</html>
```

Name: Ann Go

Name: Alex Go

Более подробно о DOM см.: http://asp.net.by/**DOM**/

AJAX

AJAX, Ajax (Asynchronous Javascript and XML — «асинхронный JavaScript и XML»). Подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.

В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее

http://asp.net.by/ajax/00.html

Более подробно о DOM см.: http://asp.net.by/**DOM**/

jQuery

Набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML.

Библиотека jQuery помогает:

- легко получать доступ к любому элементу DOM *($("h1").show();)*
- обращаться к атрибутам и содержимому элементов DOM *(p[0].value="Alex";)*
- манипулировать ими.

Библиотека jQuery предоставляет удобный API для работы с AJAX.

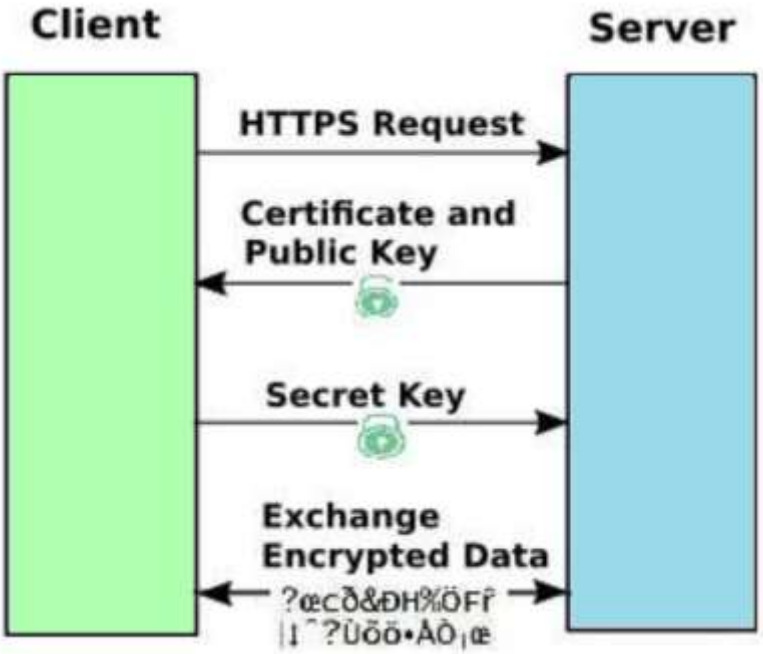
Пример - http://asp.net.by/jQuery/01.html

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
</script>
<script>
jQuery (document).ready(function(){
    $("#hide").click(function(){
        $("h1").hide();
    });
    $("#show").click(function(){
        $("h1").show();
     });
 });
 </script>
 </head>

 <body>
   <h1>Hello</ h1>
   < h1>jQuery</ h1>

 <button id="hide">Hide</button>
 <button id="show">Show</button>

</body>
</html>
```

Более подробно о jQuery см.: http://asp.net.by/**jQuery/**

SSL

SSL (англ. Secure Sockets Layer — уровень защищённых сокетов) — криптографический протокол, который подразумевает более безопасную связь.



Он использует:

- асимметричную криптографию для аутентификации ключей обмена,
- симметричное шифрование для сохранения конфиденциальности

Более подробно о SSL см.: http://asp.net.by/**SSL**/

Bootstrap

Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и вебприложений [getbootstrap.com]. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения

Более подробно о Bootstrap см.: http://asp.net.by/**Bootstrap**/

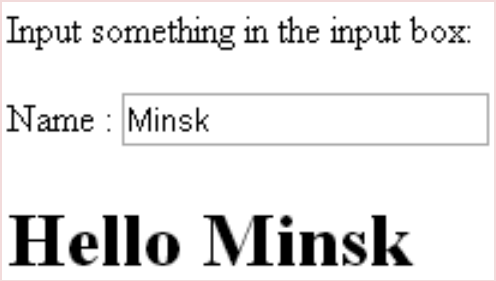
Angular JS

AngularJS – это a **JavaScript framework**. Он может быть добавлен на HTML страницу в <script> тэг.

AngularJS расширяет HTML аттрибуты с помощью директив, и связывает HTML с **выражениями**.

Пример http://asp.net.by/Angular/01.html

```
<!DOCTYPE html>
<html lang="en-US">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name :

<input type="text" ng-model="name">
</p>  <h1>Hello {{name}}</h1>
</div>
</body>

</html>
```



Более подробно о Angular.JS см.: http://asp.net.by/Angular/

React.JS

React (иногда React.js или ReactJS) — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций

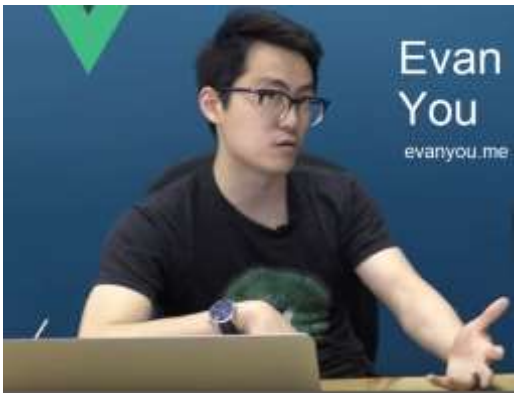
Пример см. http://asp.net.by/React/04.htm


```
<div id="root"></div>

<script type="text/babel">

function tick() {

const element=(<i>{new Date().toLocaleTimeString()}</i>);

ReactDOM.render(element,  document.getElementById('root')); }

setInterval(tick, 3000);

</script>
```

Более подробно о React.JS см.: http://asp.net.by/React/

Vue.JS

Vue (произносится /vjuː/, примерно как view) — это фреймворк для создания пользовательских интерфейсов.



Создателем Vue.js является Evan You, бывший сотрудник Google и Meteor Dev Group. Начал он разрабатывать фреймворк в 2013-м, а в феврале 2014-го состоялся первый публичный релиз.

Vue широко используется среди китайских компаний, например: Alibaba, Baidu, Xiaomi, Sina Weibo и др.Он входит в ядро Laravel и PageKit.

Недавно свободная система управления репозиториями GitLab тоже перешла на Vue.js.

Более подробно о Vue.JS см.: http://asp.net.by/Vue/

BackEnd

Основными элементами классической схемы работы интернета являются WEB-браузер и www-сервер.

Под BackEnd'ом понимают код, выполняемый сервере (или «серверная часть»). Сервер часто физически удален от пользователя.

Браузер и сервер взаимодействуют нижеследующим образом:

1. Браузер формирует запрос к серверу, используя протокол HTTP.

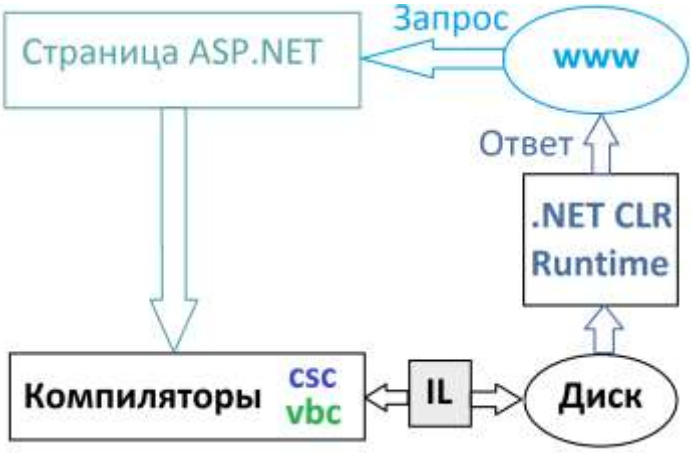
Как правило, браузер запрашивает HTML-страницу, то есть текстовый файл, содержащий HTML-код.

2. Сервер анализирует запрос браузера и извлекает из локального хранилища требуемый файл.

3. Сервер формирует HTTP-ответ, включающий требуемую информацию, и отсылает его браузеру по протоколу HTTP.

4. Браузер преобразует HTML код в изображение.

Для создания клиент-серверных приложений фирмой Microsoft была разработана технология ASP.NET. ASP.NET построена на Common Language Runtime (CLR), что позволяет программистам писать код ASP.NET с помощью любого поддерживаемый язык .NET, такие как C # и VB.NET.



ASP.NET веб-страница или веб-страница, известная официально как Web Forms, является основным строительным блоком для разработки приложений.

Веб-формы содержатся в файлах с расширением «.aspx».

Более подробно о BackEnd см.: http://asp.net.by/BackEnd/

Web Forms

В ASP.NET Web Forms серверный код включается непосредственно непосредственно в синтаксис HTML, аналогично PHP.

Страницы имеют расширение aspx.

```
<script runat="server">
    protected void ButtonLOG_Click(object sender, EventArgs e)
    {
        var X = int.Parse(TextBox_X.Text);
        LabelResult.Text = System.Math.Log10(X).ToString();
    }
</script>

<html>
<body>
   <form id="form1" runat="server">
   <div>

lg(<asp:TextBox ID="TextBox_X" runat="server" Width="32px">1000</asp:TextBox>)
<asp:Button ID="ButtonLOG" runat="server" Text="=" OnClick="ButtonLOG_Click" />
<asp:Label ID="LabelResult" runat="server" Text="Label"></asp:Label>
 </div>
</form>
</body>
</html>
```

lg(1000) = 3

Есть возможность размещения кода в отдельном файле *.aspx.cs

01.aspx

```
<%@ Page Language="C#"
AutoEventWireup="true" CodeFile="01.aspx.cs"
Inherits="_01" %>

<!DOCTYPE html>

<html>
<body>
   <form id="form1" runat="server">
   <div>

lg(<asp:TextBox ID="TextBox_X" runat="server" Width="32px">1000</asp:TextBox>)
<asp:Button ID="ButtonLOG" runat="server" Text="=" OnClick="ButtonLOG_Click" />
<asp:Label ID="LabelResult" runat="server" Text="Label"></asp:Label>
 </div>
</form>
</body>
</html>
```

01.aspx.cs

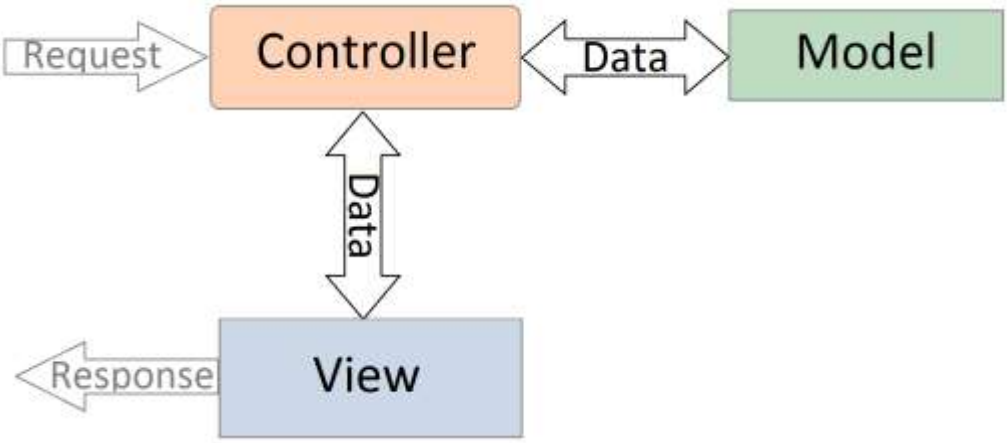
```
protected void
   ButtonLOG_Click(object
    sender, EventArgs e)
    {
        var X =
int.Parse(TextBox_X.Text);
        LabelResult.Text =
              Math.Log10(X).
                 ToString();
    }
```

Более подробно о Web Forms см.: http://asp.net.by/Web Forms/

MVC

В 2007 г. в Microsoft анонсировали новую платформу веб-разработки - MVC, построенную на основе ASP.NET.

Работа пользователя с приложением MVC осуществляется следующим образом: пользователь предпринимает действие, в ответ на которое MVC-приложение изменяет свою модель данных и доставляет обновленное представление пользователю. Затем цикл повторяется. Это хорошо укладывается в схему веб-приложений, предоставляемых в виде последовательностей запросов и ответов HTTP.



Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Более подробно о MVC см.: http://asp.net.by/MVC/

VB.NET

VB.NET или Visual Basic.NET в настоящее время является одним из наиболее популярных языков программирования.

Хотя он уступает по популярности таким языкам, как C++, C#, Java в силу различных причин, однако по возможностям и своему потенциалу выше перечисленным языкам мало в чем уступает.

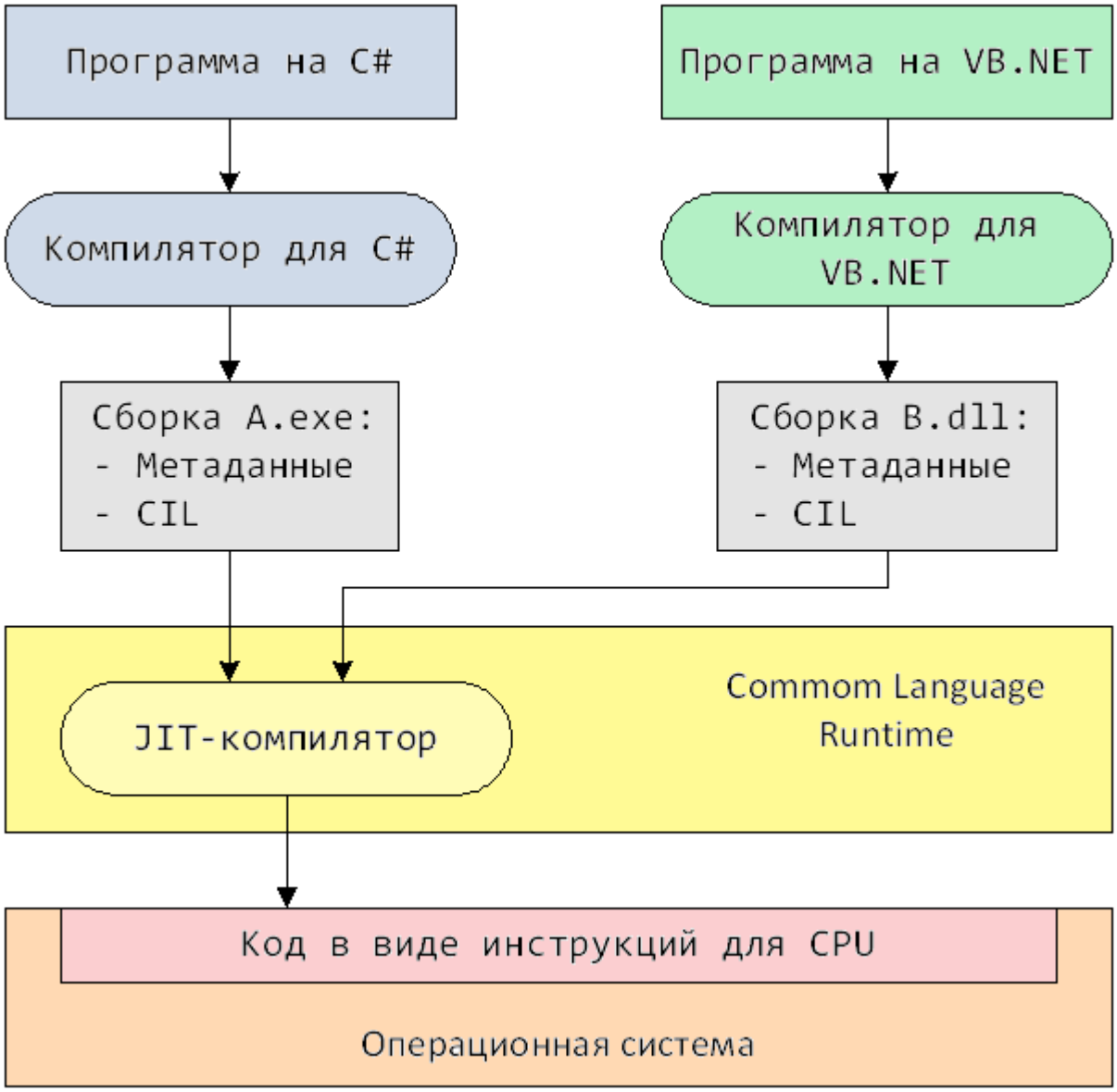
Одной из основных особенностей VB.NET является его объектно-ориентированность. VB.NET - полноценный объектно-ориентированный язык. Он поддерживает полиморфизм, наследование, статическую типизацию, перегрузку операторов.

Одной из основных особенностей VB.NET является его объектно-ориентированность. VB.NET - полноценный объектно-ориентированный язык. Он поддерживает полиморфизм, наследование, статическую типизацию, перегрузку операторов.

Более подробно о VB.NET см.: http://asp.net.by/VB.NET/

C#

C# (произносится си шарп) — объектно-ориентированный язык программирования, разработанный компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как основной язык разработки приложений для платформы Microsoft .NET Framework.



Компилятор C# компилирует программу, написанную на C#, в промежуточный код – Common Intermedia Language (IL) или Intermedia Language (IL) – который представляет собой высокоуровневый объектно-ориентированный ассемблер.

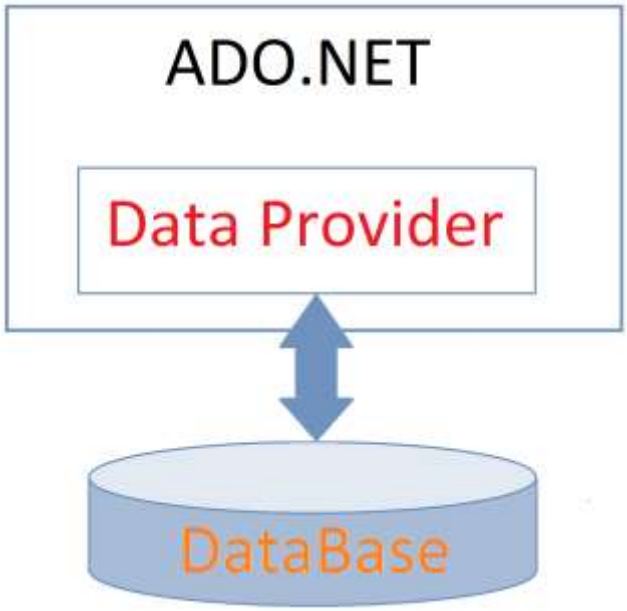
Этот код интерпретируется JIT – компилятором (Just-in-Time Compilator) в инструкции CPU.

Подобный подход позволяет разрабатывать различные части проекта на разных языках.

Более подробно о C# см.: http://asp.net.by/cs/

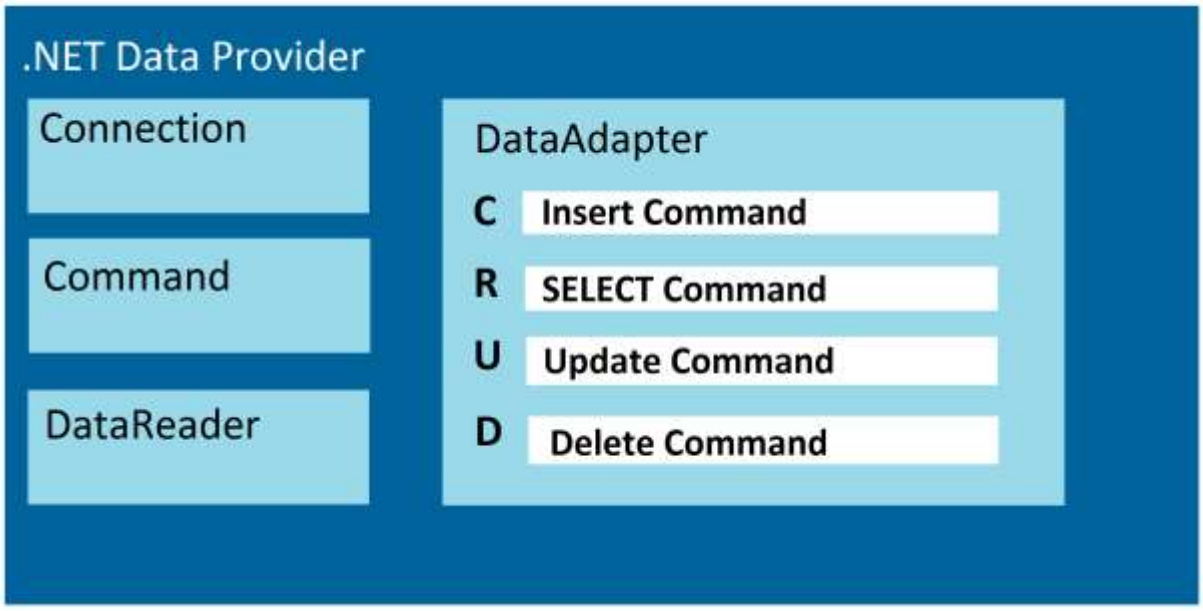
ADO.NET

ADO.NET – набор класоов, представляющих собой базовую функциональность для работы с базами данных и дугими источниками, хранящими информацию (Microsoft SQL Server, Microsoft Access, Microsoft Excel, Microsoft Outlook, Microsoft Exchange, Oracle, OLE DB, ODBC, XML, текстовые файлы).



Имеется возможность автономной работы с помощью объектов DataSet, которые представляют собой копии таблиц со связями.

Объект DataSet позволяет работать с базой данных в отключенном состоянии, и отправлять обратно полученные данные с помощью адаптера данных.



Доступ к ADO.NET возможен из C#, VB.NET и с любого языка .NET.

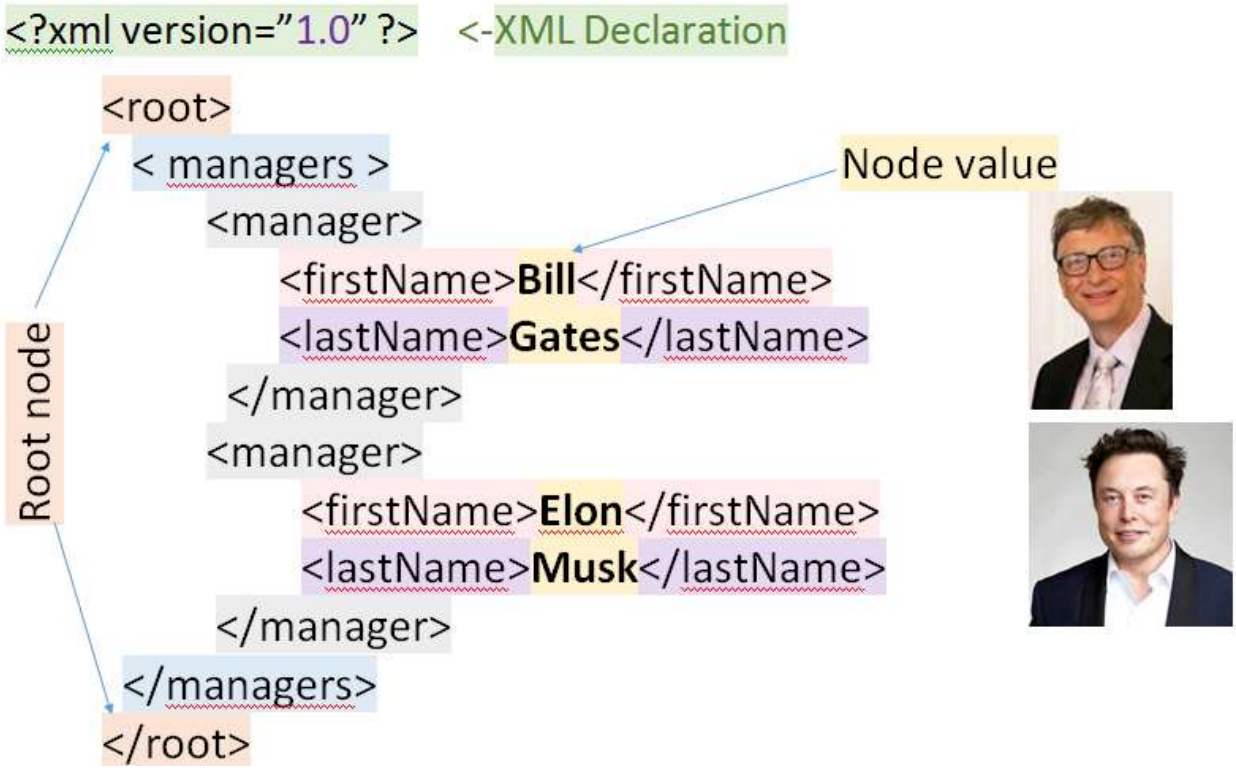
Классы ADO.NET содержатся в файле System.Data.dll

Более подробно о ADO.NET см.: http://asp.net.by/ADO.NET/

XML

XML (eXtensible Markup Language, расширяемый язык разметки) – это способ описания структурированных данных. Структурированными данными называются такие данные, которые обладают заданным набором семантических атрибутов и допускают иерархическое описание. XML-данные содержатся в документе, в роли которого может выступать файл, поток или другое хранилище информации, способное поддерживать текстовый формат.

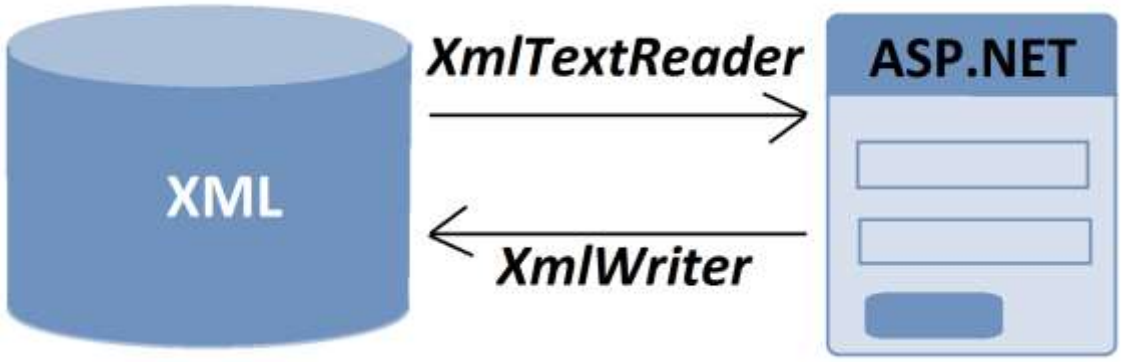
Любой XML-документ строится по определённым правилам.



Пространство имен System.Xml обеспечивает базовую функциональность работы с XML-файлами в ASP.NET.

Класс **XmlWriter** - это базовый класс, который обеспечивает процесс пересылки только для чтения для генерации потока XML. Он предоставляет методы для записи данных в документ XML.

XmlTextReader - это класс для чтения данных из XML-документа.



Конфигурационные файлы WEB.config используют XML.

Карты сайта для Google также пишутся в XML.

Пример фрагмента карты сайта SiteMap.XML:

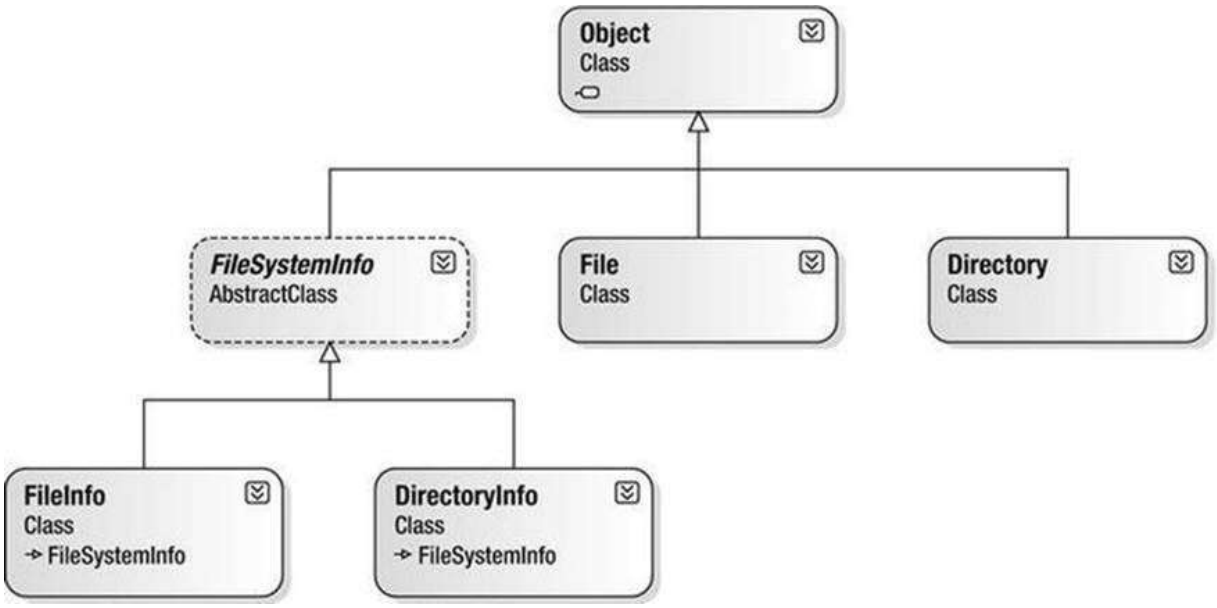
```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
<loc>http://sunny.gear.host/web-technologies/</loc>
<priority>0.5</priority>
<lastmod>2019-11-10T20:20:33+00:00</lastmod>
<changefreq>daily</changefreq>
</url>
……
</urlset>
```

Более подробно о работе с XML в ASP.NET см.: http://asp.net.by/XML/

Работа с файлами в ASP.NET

Для выполнения операций с файлами существует несколько классов, расположенных в пространстве имён "System.IO".

2 класса **File** и **FileInfo**, которые предназначены для работы с файлом, как с частью файловой системы. Есть несколько методов, которые позволяют работать с содержимым файла целиком.



Классы Directory, File, DirectoryInfo и FileInfo предназначены для работы с каталогами и файлами. Первые два класса выполняют операции при помощи статических методов, вторые два – при помощи экземплярных методов.

Пример работы с файлами в ASP.NET.

Пример.

http://asp.net.by/Files/Copy/


```
protected void btnFileCopy_Click(object sender, EventArgs e)
{
    string siteLocation = Request.PhysicalApplicationPath +"\\Files\\Copy";
    string fileToCopy = Path.Combine(siteLocation, "Default.aspx");
    string folderLocation = Path.Combine(siteLocation, "copiedFiles");
    string newLocation = Path.Combine(folderLocation, "Default.aspx");
```

```
if (Directory.Exists(folderLocation))
    {
        if (File.Exists(fileToCopy))
        {
            File.Copy(fileToCopy, newLocation, true);
            Label1.Text ="<font color='green'>File copied to folder</font>";
        }
        else
        {
            Label1.Text = "<font color='red'>No such file</font>";
        }
    }
    else
    {
        Label1.Text = " <font color='red'>No such directory</font>";
    }
}
```

Классы Directory, File, DirectoryInfo и FileInfo предназначены для работы с каталогами и файлами. Первые два класса выполняют операции при помощи статических методов, вторые два – при помощи экземплярных методов.

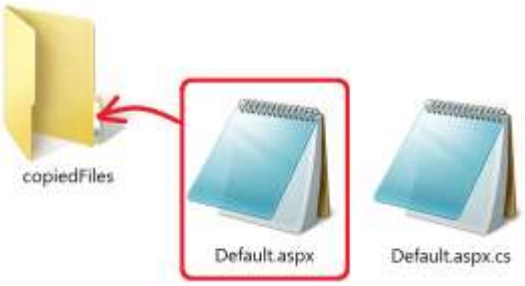
Вышеприведенный код проверяет, существует ли каталог

if (Directory.Exists(folderLocation))

Если это так, то мы проверяем, существует ли файл, который мы собираемся копировать

```
if (File.Exists(fileToCopy))
```

Если все в порядке, тогда мы идем дальше и копируем файл




```
File.Copy(fileToCopy, newLocation, true);
```

Если копируемый файл уже находится в папке, установка значения переопределения на **true** заменит его. Значение по умолчанию - **false**, и вы получите ошибку, если копируемый файл уже существует.

Чтобы переместить файл в новое место, вы должны использовать метод Move класса File:

```
File.Move(fileToMove, newLocation);
```

Более подробно о работе с файлами см.: http://asp.net.by/Files/

Конфигурация и безопасность

Аутентификация

Аутентификацией (authentication) называется процесс идентификации пользователей приложений.

Авторизацией (authorization) называется процесс предоставления доступа пользователям на основе их идентификационных данных.

Авторизация

ASP.NET совместно с веб-сервером обеспечивает несколько возможных типов аутентификации:

- Windows (по умолчанию),
- Forms
- Passport
- None

Выбор типа определяет механизм хранения маркеров аутентифицированного пользователя. В случае типа Windows маркер помещается в контекст потока рабочего процесса ASP.NET. Если используется тип Forms, маркер передается от клиента к серверу и обратно в cookie-файле.

Задать требования аутентификации клиентов веб-приложения можно путем добавления соответствующих элементов в конфигурационный файл приложения web.config.

Тип аутентификации клиентов задается с помощью элемента <authentication>, атрибут mode которого может принимать одно из четырех значений: Windows,

Forms, Passport и None. Сконфигурировав тип аутентификации, нужно продумать цели аутентификации. Аутентификация выполняется для ограничения доступа клиентов ко всему приложению или к его частям с помощью элемента <authorization>. В этот элемент добавляются подэлементы <allow> и <deny>, задающие предоставление или отказ в доступе отдельным пользователям и ролям.

Метасимвол * используется для представления всех пользователей, а ? – для представления анонимных пользователей.

Следующий пример конфигурационного файла отказывает анонимным пользователям в праве доступа к сайту:

```
<configuration>
<system.web>
  <authorization>
    <deny users="?" />
  </authorization>
</system.web>
</configuration>
```

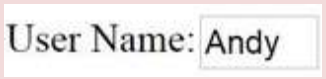
Элементы <allow> и <deny> поддерживают три атрибута: users, roles и verbs. Значениями этих атрибутов могут быть разделенные запятыми списки пользователей, ролей и команд.

В корне WEB-config должен выглядеть так:

```
<configuration>
  <system.web>
    <authentication mode="Forms">
      <forms loginUrl="LoginPage.aspx">
        <credentials passwordFormat="Clear">
          <user name="Andrey" password="651652"/>
          <user name="Andy" password="651"/>
        </credentials>
      </forms>
    </authentication>
    <compilation debug="true" targetFramework="4.0"/>
  </system.web>
</configuration>
```

Файл **LoginPage.aspx** выглядеть так:

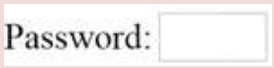

```
<html>
 <body>
 <form runat="server">
 <asp:Button Text="Log In" OnClick="OnLogIn"  RunAt="server" />
 User Name: <asp:TextBox ID="UserName" runAt="server" />
```




```
<font color="red">
              <asp:Label ID="l_Login" RunAt="server" />
             </asp:Label>
</font>
```


```
          <font color="red">
            <asp:Label ID="l_or" runat="server" Text="">
            </asp:Label> </font>
```


```
         Password:
         <asp:TextBox ID="Password" TextMode="password"
           RunAt="server" />
         <asp:Label ID="l_Password" runat="server"
                    Text="" ForeColor="Red"></asp:Label>
   </form>
```




```
<script language="C#" runat="server">

 void OnLogIn (Object sender, EventArgs e)

 {   if (FormsAuthentication.Authenticate(UserName.Text, Password.Text))

        FormsAuthentication.RedirectFromLoginPage(UserName.Text,  false);

     else

        {  l_Login.Text = "<= Invalid login";

           l_or.Text="or";

           l_Password.Text="<= Invalid password";

        }
 }

</script>
```

Более подробно о конфигурации и безопасности см: http://asp.net.by/Config/

Ярошевич Андрей Олегович

Институт Бизнеса

Белорусского Государственного Университета

e-mail: ao@asp.net.by

yaroshevich.com asp.net.by