Лабораторная работа № 1

Создание приложения ASP.NET

# 1. Цель работы

Целью работы является рассмотреть особенности создания Web-сайтов с использованием Visual Studio 2008,а также отработать методику создания совместных классов.

# 2. Состав рабочего места

2.1. Оборудование: IBM-совместимый персональный компьютер (ПК).

2.2. Программное обеспечение: операционная система Windows, среда разработки Microsoft Visual Studio 2008.

# 3. Краткие теоретические сведения

**Основы работы в Visial Studio .NET 2008**

Почти все крупномасштабные Web-сайты ASP.NET разрабатываются с использованием Visual Studio - предлагаемой компанией Microsoft полнофункциональной среды разработки Web-приложений, гибкого и универсального инструмента проектирования и создания законченных приложений для платформы Windows. Работа над ней шла многие годы, и в нее включены многие функции, специфические для .NET Framework. Как и любая другая профессиональная среда разработки, Visual Studio включает в себя средства управления проектами, редактор исходного кода, конструкторы пользовательского интерфейса, мастера, компиляторы, компоновщики, инструменты, утилиты, документацию и отладчики. Она позволяет создавать приложения для 32- и 64-разрядных Windows-платформ, а также новой платформы .NET Framework. Одно из важнейших усовершенствований — возможность работы с разными языками и приложениями различных типов в единой среде разработки.

К другим основным преимуществам Visual Studio можно отнести следующие.

* Встроенный редактор WISWYG (What You See Is What You Get) ("Что видишь, то и получаешь"). С его помощью Visual Studio позволяет настраивать статическое HTML-содержимое, в том числе шрифты и стили.
* Меньше кода для написания благодаря автоматизации базовых задач по созданию стереотипного кода. Например, при добавлении на Web-страницу нового элемента управления необходимо также определить переменную, позволяющую манипулировать этим элементом управления в коде, что и выполняет Visual Studio. Похожая автоматизация применяется и при подключении к Web-службам.
* Интуитивный стиль программирования — форматирование кода, выполняемое Visual Studio во время его набора в виде автоматической расстановки отступов и использования цветового кодирования; значительно улучшает удобочитаемость кода и снижает вероятность допущения ошибок в коде.
* Встроенный Web-сервер, позволяющий запускать Web-сайт прямо из среды проектирования. Помимо удобства для пользователя, таким образом увеличивается степень безопасности, т. к. отсутствует возможность получения доступа с внешнего компьютера к разрабатываемому Web-сайту.
* Многоязыковая разработка — Visual Studio позволяет кодировать на любых языках с использованием одного и того же интерфейса (IDE). Более того, эта среда разработки предоставляет возможность помещения Web-страниц, реализованных на C#, в тот же проект, в котором уже имеются Web-страницы, написанные на Visual Basic. Единственное ограничение: на одной странице можно применять только один язык.
* Увеличение скорости разработки, для чего Visual Studio обладает множеством возможностей. Удобства вроде мощной функции поиска и замены, а также средства автоматического добавления и удаления комментариев, которые могут временно скрыть блок кода, позволяют работать быстро и эффективно.
* Отладка, для удобства осуществления которой Visual Studio располагает некоторым инструментарием. Так, возможно выполнять код по строке за раз, устанавливать интеллектуальные точки прерывания, которые можно сохранить для дальнейшего использования и в любое время отображать текущую информацию из памяти.

Это далеко не полный перечень всех возможностей данной среды разработки, предоставляющей разработчикам поистине огромный инструментарий.

**Редактирование Web-документов**

Web-формы и HTML-документы можно редактировать в графическом режиме с использованием "Drag & Drop", как обычные Windows-формы, либо в текстовом виде. Естественно, не все задачи можно решить в графическом режиме. Поэтому приходится редактировать Web-документы в режиме исходного кода. Если хорошо знать разметку элементов ASP.NET и HTML, то этот способ может оказаться даже удобнее использования графических инструментов. Поддерживаемая Visual Studio технология IntelliSense помогает завершать элементы ASP.NET и HTML, создаваемые в режиме исходного кода.

Важным моментом при размещении элементов внутри Web-страницы является то, что форматирование самой страницы подчиняется принципам форматирования HTML-страниц. Это логично, т. к. костяк файла .aspx составляет именно HTML-код страницы. В частности, из-за этого невозможно размещать элементы управления в произвольном месте Web-страницы, так, как к этому привыкли разработчики Windows-приложений. Для форматирования страниц Web-приложений очень часто используются таблицы, позволяющие размещать элементы внутри ячеек, позиционируя таким образом элементы управления в нужном месте страницы. Однако, в соответствие стандартам W3C, разметку рекомендуется производить с использованием блочных элементов и стилевого оформления. При этом важным является использование спецификации XHTML1.0 и выше.

**Редактирование кода**

При создании новой Web-формы автоматически генерируется связанный с ней файл, содержащий программный код этой страницы. Имя этого файла составляется из имени файла Web-формы и расширения .vb или .cs (в зависимости от используемого языка программирования: cs соответствует языку C#, vb — Visual Basic).

В каждом файле кода Visual Studio генерирует определение соответствующего класса, включающего в том числе и обработчик события Page\_Load. Редактор кода также поддерживает функцию Autocomplete, автоматически распознающую и предлагающую для завершения ключевые слова и члены классов, реализуемую при помощи технологии IntelliSense. Окно подсказки появляется всякий раз, когда происходит ввод ключевого слова, представляющего собой команду, директиву, тип данных либо зарезервированное слово. При этом пользователю достаточно выбрать из списка нужный вариант и нажать нецифросимвольную клавишу(например, <Enter>, <Tab>, <.>и т.д.). В случае, если окно подсказки закрылось, а команда так и не была выбрана, пользователь может вызвать его с помощью нажатия комбинации клавиш CTRL+пробел. Если написать исходный код страницы на языке Visual Basic, то после завершения ввода каждой строки функция Autocomplete будет делать первые буквы ключевых слов и имен членов заглавными. Однако в программе, написанной на Visual C#, ключевые слова и члены классов не распознаются, если при их вводе использован неверный регистр, так как C#, в отличие от Visual Basic.NET, чувствителен к регистру символов. Кроме того, после ввода каждой строки редактор кода выделяет синтаксические ошибки и необъявленные переменные, подчеркивая их волнистой линией. При наведении указателя мыши на выделенную ошибку отображается ее описание. Большинство автоматизированных возможностей редактора кода можно включать и отключать, изменяя параметры в диалоговом окне Options. Кроме того, в нем предусмотрена корректировка автоматических абзацных отступов, завершения блоков кода и других параметров, определяющих работу с языком.

### Установка точек прерывания и просмотр значений переменных

Существует возможность остановить исполнение проекта перед некоторой строкой кода, установив в этой строке точку прерывания. Для этого следует щелкнуть на сером поле слева от строки, перед которой нужно остановить исполнение программы, либо, выделив эту строку, нажать F9. При этом строка выделяется, а слева от нее появляется точка, свидетельствующая о наличии точки прерывания программы.

Встретив при исполнении проекта строку с точкой прерывания, отладчик Visual Studio остановит приложение перед выполнением кода этой строки, выделит ее и откроет в окне редактора кода.

После остановки выполнения приложения перед точкой прерывания стоит просмотреть значения активных переменных, наводя на них указатель мыши. Для просмотра значений переменных сложного типа, таких как переменные-объекты или массивы, следует добавить эти переменные в окно Watch путем выбора и перетаскивания. Просмотреть значения вложенных элементов (например, элементов массива или свойств объекта) можно, щелкнув по знаку "+" справа от сложного элемента в окне Watch.

### Пошаговое исполнение

Для возобновления исполнения приложения, остановленного перед точкой прерывания, следует щелкнуть по кнопке Continue на панели инструментов либо нажать F5. Можно также исполнять приложение построчно, нажимая F10 (команда Debug http://www.intuit.ru/img/symbols/rArr.gifStep Over) или F11 (команда Debug http://www.intuit.ru/img/symbols/rArr.gif Step Into).

Разница между этими командами заключается в следующем. F10 заставляет при вызове процедуры исполнить ее как одну команду. Другими словами, процедура исполняется без захода в нее (step over), после чего исполнение останавливается на следующей после вызова этой процедуры строке. Нажатие клавиши F11 вызывает исполнение процедуры с заходом в нее (step into), при этом исполнение останавливается на первой строке вызванной процедуры.

**Запуск приложения и организация виртуальных каталогов приложений**

Как известно, для функционирования ASP.NET приложения необходим Web-сервер, способный взаимодействовать с данной средой исполнения. Размещение Web-приложений на сервере обычно происходит в виртуальных каталогах, представляющих собой обычные каталоги, которые доступны через Web-сервер.

Виртуальные каталоги имеют большое значение, так как влияют на выполнение ASP.NET программного кода. Так, в процессе исполнения Web-приложения, Web-страницы и Web-службы, находящиеся в одном виртуальном каталоге, выполняются в рамках одного домена приложения. В случае же, если они расположены в разных виртуальных каталогах, их выполнение происходит в разных доменах.

Домен приложения представляет собой эквивалент процесса, реализованный в .NET. Он гарантирует невозможность для одного приложения оказывать влияние на другое, то есть использовать пересекающиеся области памяти.

Для создания доменов ASP.NET использует механизм отложенной инициализации, при котором домен приложения создается в момент получения первого запроса на страницу или Web-службу в этом приложении. Домен приложения может быть закрыт в силу различных причин, включая произошедшую в приложении ошибку. ASP.NET способен повторно использовать домены приложений. Это означает, что в случае возникновения ошибки произойдет перезапуск домена в момент очередного обращения пользователя к Web-странице или службе. Кроме того, при изменении приложения также происходит создание нового домена, который будет обслуживать все вновь поступающие запросы к приложению. Старый домен при этом будет сохранен до тех пор, пока не будут обработаны все оставшиеся запросы (включая запросы, находящиеся в очереди).

Следствием этой особенности является то, что обновление Web-приложения возможно производить в любой момент, не перезапуская Web-сервер и не беспокоясь о причинении вреда клиентам, работающим с ним в данный момент.

**Структура каталога приложения**

Виртуальный каталог может содержать большое количество ресурсов, используемых Web-приложением. К ним относятся таблицы стилей, изображения, XML-файлы и т. д. В то же время в виртуальном каталоге могут содержаться подкаталоги, которые, в зависимости от своего названия, могут выполнять специальные функции. Ниже перечислены специальные каталоги ASP.NET.

|  |  |
| --- | --- |
| Bin | Содержит все предварительно скомпилированные сборки .NET, которые обычно представляют собой DLL-библиотеки. Эти библиотеки используются Web-приложением и могут включать предварительно скомпилированные классы Web-страниц и служб, а также другие сборки, на которые ссылаются данные классы. |
| App\_Code | Содержит классы исходного кода, динамически скомпилированные для использования в рамках приложения. Обычно эти файлы кода представляют собой отдельные компоненты. |
| App\_GlobalRecources | Хранит глобальные ресурсы, доступные каждой странице Web-приложения. |
| App\_LocalRecources | Хранит локальные ресурсы, доступные только специальной странице. |
| App\_WebReferences | Хранит ссылки на Web-службы, используемые приложением. |
| App\_Data | Хранит файлы данных, включая XML-файлы и файлы SQL Express. |
| App\_Browsers | Содержит определения браузера, записанные в формате XML. Эти файлы определяют характеристики браузеров на стороне клиентов и влияют на визуализацию страницы. |
| App\_Themes | Хранит темы, используемые Web-приложением. |

Для упрощения отладки рассматриваемых примеров рекомендуется воспользоваться возможностями встроенного в Visual Studio Web-сервера, обеспечивающего ту же функциональность, что и IIS, за исключением того, что к нему возможны подключения только с локальной машины (той, на которой расположено само Web-приложение).

### Окно Solution Explorer. Структура ASP.NET приложения

Visual Studio упорядочивает приложения при помощи проектов и решений. Проект (project) — это набор файлов, из которых в итоге компонуется исполняемый файл. Решение (solution) — это группа проектов, образующих функциональную единицу. Файлы, составляющие решение, можно просматривать в окне Solution Explorer.

ASP.NET может придерживаться принципа разделения программного кода и разметки страниц интерфейса системы. Интерфейс системы описывается в файлах с расширением .aspx, а исходный код размещается в файлах с расширением .cs в случае, если в качестве языка программирования используется C# и .vb - если Visual Basic.

Еще одним важным файлом проекта ASP.NET приложения является файл web.config. При начальном создании проекта Visual Studio не создает этот файл. Однако при первом запуске приложения (с помощью нажатия клавиши F5) пользователю предлагается создать этот файл и установить в нем опцию, разрешающую отладку данного приложения.

Файл web.config является XML-образным конфигурационным файлом . Он содержит большое количество параметров настройки приложения, которые конфигурируют все его общие настройки, начиная с безопасности и заканчивая отладкой, подключением к источникам данных и управлением состояния.

# 4. Порядок выполнения работы

## Упражнение 1

## Создание Web сайта

В этом упражнении вы создадите новый Web сайт в Visual Studio 2008. Вы создадите домашнюю страницу для корпоративного сайта компании Adventure Works.

Выполнение

1. Откройте файл Default.aspx.
2. В среде **Visual Studio**, в меню **File**, нажмите **Open Web Site**.
3. В открывшемся диалоговом окне выберите ~\LabFiles\Starter\*language*\AdWorksWebSite , и нажмите **Open**.

1. **Создание домашней страницы**
2. Измените название заголовка страницы на предложенное Adventure Works.
   * 1. Ваш код должен выглядеть следующим образом:

**C#**

<title>Adventure Works Intranet Portal</title>

1. Добавьте таблицу для заголовка в верху тэга <form> без тэга **<div**>.
2. Добавьте следующие 7 навигационных гиперссылок в таблицу:

**Home**

**HR**

**Accounting**

**Legal**

**Sales**

**Training**

**Expenses**

Ваш код должен иметь вид:

<asp:Table ID="Table1" Runat="server">

<asp:TableRow ID="TR1" Runat="server">

<asp:TableCell ID="TD1" Runat="server">

<asp:HyperLink ID="Hyperlink1" Runat="server" NavigateURL="~/default.aspx">Home</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink2" Runat="server" NavigateURL="~/hr.aspx">HR</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink3" Runat="server" NavigateURL="~/accounting.aspx">Accounting</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink4" Runat="server" NavigateURL="~/legal.aspx">Legal</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink5" Runat="server" NavigateURL="~/sales.aspx">Sales</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink6" Runat="server" NavigateURL="~/training.aspx">Training</asp:Hyperlink> |

<asp:HyperLink ID="Hyperlink7" Runat="server" NavigateURL="~/expenses.aspx">Expenses</asp:Hyperlink>

</asp:TableCell>

</asp:TableRow>

</asp:Table>

1. Добавьте поле заголовка компании рядом с таблицей.
   * 1. Ваш код должен иметь вид:

**C#**

<asp:Label ID="Label1" Runat="server" Text="Adventure Works"></asp:Label><br />

<asp:Label ID="Label2" Runat="server" Text="Home"></asp:Label><br />

1. Добавьте таблицу для контента под заголовком страницы и напишите приветствие:
   * 1. Ваш код должен иметь вид:

**C#**

<asp:Table ID="Table2" Runat="server">

<asp:TableRow ID="TR2" Runat="server">

<asp:TableCell ID="TD2" Runat="server">

Welcome<br />

This is the Adventure Works Home page.

</asp:TableCell>

</asp:TableRow>

</asp:Table>

1. Чтобы сохранить файл **Default.aspx**, в меню **File** выберите **Save Default.aspx.**
2. **Просмотр страницы**
3. В любом месте странички Default.aspx кликните правой кнопкой мышки и выберерите View in Browser.
4. Закройте браузер.
5. Закройте все файлы и проекты, но оставьте Visual Studio открытой.

## Упражнение 2

## Создание простого совместного класса

В этом упражнении вы создадите общий класс для Интернет сайта который позволит посылать любые ресурсы с сайта посылать в административных e-mail сообщениях.

1. Откройте файл email.*cs*.
2. В среде Visual Studio, в меню File, нажмите Open Web Site.
3. В открывшемся диалоговом окне выберите **~\LabFiles\Starter\*language*\AdWorksWebSite ,** и нажмите **Open.**
4. В окне **Solution Explorer сделайте** двойной щелчок на папке **App\_Code,** а затем двойной щелчок на файле **email.cs.**
5. Добавьте ваши ресурсы в **System.Net.Mail**:
6. Используйте пространство имен **System.Net.Mail.**

**C#**

using System.Net.Mail;

1. Добавьте вашу логику в публичный класс **email.**
2. В классе **email** создайте приватные переменные:

**Address**

**Subject**

**Body**

**Attachment**

* + 1. Ваш код должен выглядеть следующим образом:

**C#**

private string \_toAddress;

private string \_subject;

private string \_body;

private string \_attachment;

1. После объявления закрытых полей класса определите соответствующие им открытые свойства класса:

**C#**

public string ToAddress

{

get { return \_toAddress; }

set { \_toAddress = value; }

}

public string Subject

{

get { return \_subject; }

set { \_subject = value; }

}

public string Body

{

get { return \_body; }

set { \_body = value; }

}

public string Attachment

{

get { return \_attachment; }

set { \_attachment = value; }

}

1. За описанием свойства следует открытый метод **SendMail.** Используйте класс **MailMessage** для создания объекта письма mail.

**C#**

public void SendMail()

{

MailMessage message = new MailMessage(“[admin@adventure-works.com](mailto:admin@adventure-works.com)”, ToAddress);

}

* + 1. Установите следующие свойства объекта mail в теле метода **SendMail.**

|  |  |
| --- | --- |
| Свойство объекта mail | Свойство класса, в котором хранится значение |
| Subject | Subject |
| Priority | High |
| Body | Body |

**C#**

message.Subject = Subject;

message.Priority = MailPriority.High;

message.Body = Body;

* + 1. Используйте конструкцию if для проверки наличия прикрепленных к письму файлов.

**C#**

if (Attachment != null)

{

Attachment messageAttachment = new Attachment(Attachment);

message.Attachments.Add(messageAttachment);

}

* + 1. Используйте класс **SmtpClient** для установления локального SMTP сервера, и отправки письма:

**C#**

SmtpClient emailClient = new SmtpClient("localhost");

emailClient.Send(message);

Сохраните класс.

# 5. Содержание отчета

В отчете должны быть представлены:

* Исходный код.
* Результат выполненной работы в виде скриншотов.