**Содержание**

Введение

1. Исследовательская часть

1.1 Описание предметной области

1.2 Особенности создания страниц на языке APS.NET

1.3 Особенности создания и обращения к базам данных

2. Конструкторская часть

2.1 Структура базы данных

2.2 Принцип работы первой страницы сайта

2.3 Принцип работы второй страницы сайта

2.4 Принцип работы третьей страницы сайта

2.5 Принцип работы четвертой страницы сайта

2.6 Принцип работы пятой страницы сайта

2.7 Принцип работы шестой страницы сайта

2.8 Принцип работы седьмой страницы сайта

2.9 Принцип работы восьмой страницы сайта

2.10 Принцип работы девятой страницы сайта

2.11 Принцип работы десятой страницы сайта

3. Технологическая часть

3.1 Настройка приложения

3.2 Работа с сайтом

3.3 Системные требования

Заключение

Список использованной литературы

**Введение**

Целью данной курсовой научно-исследовательской работы студента является создание сайта на языке ASP.NET 2005, который позволяет просматривать и вносить изменения в базу данных студентов. ASP.NET – это часть технологии.NET, используемая для написания мощных клиент – серверных интернет приложений. Она позволяет создавать динамические страницы HTML. Динамическими принято называть web – страницы, которые перед отправкой клиенту проходят цикл обработки на сервере. ASP.NET возникла в результате объединения более старой технологии ASP и.NET Framework. Данная работа состоит из четырех динамических страниц, часть из которых взаимодействует с базой данных. Для работы с базой данных используется язык структурированных запросов – SQL (Structured Query Language).

**1. Исследовательская часть**

* 1. **Описание предметной области**

Данная работа использует язык APS.NET 2005, который позволяет выполнять обработку событий, вследствие чего старицы являются динамическими. При обработке некоторых событий, происходит соединение с базой данных и выполнение различных запросов и операций. Следствием таким запросов может быть изменения базы данных, добавление новых данных в базу, вывод данных из базы на дисплей. Для взаимодействия с базой данных используется пространство имен *System.Data.SqlClient.*

**1.2 Особенности создания страниц на языке APS.NET**

Любая страница состоит из двух частей. Первая часть – Design, представляет собой пустое пространство, на котором можно разместить элементы управления. Вторая часть – Source, представляет собой код языка ASP.NET. Добавление нового кода, происходит при добавлении нового объекта в окне Design, либо при ручном заполнении. Для того чтобы добавить новый элемент в окно Design, его необходимо выбрать из вкладки Toolbox, и перетащить на форму. По аналогии с C#, имеется окно Properties, в котором можно задать те или иные параметры для элементов управления, или задать те или иные события.

Каждая страница начинается со следующего кода: « <% Page Language=”текущий язык программирования” %> ». Где тег “<%” всегда предназначен для интерпретации ASP – кода. Атрибут Language – это указание, что в скриптах данной страницы будет использоваться определенный язык программирования. В данной работе – это язык C#. Затем идет код вида: « <form runat=”server”> ». Этот тег дает указание компилятору обрабатывать элементы управления страницы. Данный тег имеет свойство runat, для которого установлено значение “server” (других значений не бывает). При использовании этого свойства элемент управления обрабатывается компилятором, а не передается браузеру «как есть».

При двойном щелчке на объекте в окне Design откроется файл, с названием аналогичным названию страницы, но отличающийся от нее тем, что добавляется выражение типа «.cs» в конце расширения файла. Также появится код, который соответствует событию по умолчанию для данного элемента. В данном файле ведется разработка кода уже не на языке ASP.NET, а на том языке, который вы выбрали, для данной работы – это язык C#. Задание и обращение к свойствам, методам ничем не отличается от обращения к ним же при написании любой другой программы на конкретно выбранном языке.

**Метод Response.Write()**

Данный метод отправляет в браузер HTML код. Этот метод является полезным при выведении коротких сообщений. К примеру, он может использоваться при проверке введенных значений для вывода результата.

* 1. **Особенности создания и обращения к базам данных**

Базы данных прошли долгий путь развития. Вначале все данные хранились в простых файлах. По мере увеличения объемов данных встал вопрос о том, как получить быстрый доступ к нужной информации. Для этого данные стали индексироваться. Другой вопрос – как избежать дублирования, когда одни и те же данные хранятся в разных местах. Сегодня мощная промышленная СУБД немыслима без систем защиты информации, журналирования, транзакции и хранимых процедур.

Данные в СУБД хранятся в таблицах. Таблица состоит из полей и записей. Запись – единица хранения данных, строка таблицы. Для работы с базами данных используется язык структурированных запросов – SQL (Structured Query Language). Команды этого языка называются запросами. Запросы служат для получения данных, для создания и изменения структуры таблиц, добавления, удаления и обновления записей и многого другого. Последовательность команд может храниться прямо на сервере СУБД в виде хранимой процедуры. Либо задаваться в виде набора команд, которые всегда проходят стадию компиляции.

Для создания базы данных в окне Sever Explorer нужно нажать на пункт Data Connections правой кнопкой мыши, и в контекстном меню выбрать Create New Sql Database.

Рис. 1. Создание базы данных

На (рис. 1) показан пример создания базы данных. Для того чтобы указать компилятору, что база находится на локальном компьютере, в поле «Server name» нужно вписать «.\SQLExpress» или «(local)». Имя базы данных указывается в графе «New database name». После нажатия кнопки «ОК», база данных сохранится под указанным именем с расширением «.mdf».

После создания базы данных, в ней необходимо создать таблицы для хранения данных. Таблицу можно сделать и программно, и через окно «Server Properties». При выполнении такой последовательности действий появится узел созданной базы. Его нужно раскрыть, и щелкнуть мышью на пункт Tables, в контекстном меню нужно выбрать «Add New Table». Появится диалоговое окно, в котором нужно указать название столбца, тип данных, которые будут храниться в нем, и разрешение на хранение в ячейке нулевых значений. Бывают такие ситуации, что при создании новых строк, значения в каком-то определенном столбце автоматически увеличивались, этого можно добиться путем выбора пункта «Set Primary Key».

**Объект Connection**

Объект Connection для соединения с базой данных нуждается в строке соединения для указания пути к СУБД. Свойства класса Connection указаны в таблице:

|  |  |
| --- | --- |
| Свойство | Описание |
| DataSource | Имя экземпляра базы сервера при использовании SqlConnection |
| Database | Возвращает имя базы данных, используемой в объекте Connection после открытия |
| AttachDbFilename | Указывает путь к файлу, который является локальной базой данных |
| ConnectionString | Строка соединения с СУБД |

**Метод Open()**

Данный метод вызывается после создания объекта SqlConnection, с заданной строкой соединения - ConnectionString. Метод Open() пытается установить соединения с базой данных. Если соединения установить не удается, то появляется соответствующее окно с сообщением об ошибке.

**Использование объекта SqlCommand**

Объект SqlCommand исполняет запрос SQL, который может быть в форме встроенного текста, процедуры сервера или прямого доступа к таблице. Если это запрос на выборку данных SELECT, то данные обычно помещают в DataSet или в DataReader, но также можно производить и ручные манипуляции. Запросом на добавление данных является INSERT, на изменение UPDATE, на удаление DELETE.

Процедуры сервера нуждаются в параметрах. Они хранятся в коллекции Parametrs и имеют тип SqlParametr. Текстовые команды также могут получать параметры, перед которыми ставится префикс @.

"SELECT id FROM [sTudenT] WHERE SecondName = @DropDownList1.Text"

Или же можно использовать форматированный вывод:

string.Format("SELECT id FROM [sTudenT] WHERE SecondName = '{0}'", DropDownList1.Text);

**Метод ExecuteNonQuery()**

Данный метод применим к объектам типа SqlCommand. С помощью данного метода можно выполнять любые операции с базами данных, которые не связаны с запросом или получением данных, например, обновление, удаление записей, создание и изменение таблиц, создание процедур сервера. Она возвращает количество измененных записей в том случае, если выполняются команды Select, Update, Delete.

**Метод ExecuteScalar()**

Данный метод применим к объектам типа SqlCommand. Этот метод возвращает результат запроса в случае, если это единственное значение.

Рассмотрим пример:

string.Format("SELECT id FROM [sTudenT] WHERE SecondName = '{0}'", DropDownList1.Text);

В результате будет возвращено значение поля “id” базы даных, с параметром SecondName равным значению, выбранному из списка. Данный метод работает гораздо быстрее метода ExecuteReader().

**Метод ExecuteReader()**

Данный метод выполняется, если нужно получить табличные данные. Результат выполнения – курсор, в котором можно двигаться только от начала до конца.

В результате выполнения метода ExecuteReader() объекта Command создается объект DataReader. После использования данного метода необходимо закрывать соединения с базой, иначе такой процесс останется активным и будет занимать ресурсы. Закрыть соединение можно двумя способами. Первый – вызвать перегруженный метод ExecuteReader(), который принимает параметр типа CommandBehavior сл значением CommandBehavior.CloseConnection. В таком случае необходимо перелистать полученную выборку от начала до конца, и соединение закроется, когда будет достигнут конец.

В Visual Studio 2005 придусмотрен интерфейс автоматического вывода определнного столбца базы данных в такие элементы управления, к примеру, как DropDownList. На следующий рисунках изображен пример такого интерфеса:

Рис. 2. Выбор источника данных

Как видно на (рис. 2), сперва, выбирается источник данных. Этот источник создается автоматически.

Рис. 3. Указание пути к базе данных

На (рис. 3) выбирается строка присоединения к базе данных.

Рис. 4. Выбор столбца

На (рис. 4) выбирается столбец, который будет отображен в конкретном элементе управления. При компиляции страницы, в выбранном элементе управления, будет находится содержания выбранного столбца. Такой метод значительно упрощает разработку приложений.

**2. Конструкторская часть**

**2.1 Структура базы данных**

База данных состоит из трех таблиц. Таблица sTudenT служит для хранения фамилии, имени и отчества студента. Имеет следующую структуру:

|  |  |  |  |
| --- | --- | --- | --- |
| id | SecondName | FirstName | oT4ecTBo |

Соответственно в столбец SecondName добавляется фамилия студента, в столбец FirstName добавляется имя студента, в столбец oT4ecTBo добавляется отчество студента. Столбец id задан как Primary Key, сведовательно значения этого столбца формируются автоматически.

Таблица subjects состоит из двух столбцов id и Name. Соответственно id формируется автоматически, а в столбец Name заносятся названия предметов.

|  |  |
| --- | --- |
| id | Name |

Таблица markbook имеет более сложную структуру, она состоит из 4 столбцов.

|  |  |  |  |
| --- | --- | --- | --- |
| id | idst | idsubj | mark |

Столбец id формируется автоматически. Столбец idst служит для занесения туда идентификационного номера студента, который берется из таблицы sTudenT. Столбец idsubj служит для занесения в него идентификационного номера предмета, который берется из таблицы subjects. Столбец mark служит для занесения оценки по определенному предмету. Все столбцы имеют формат int.

**2.2 Принцип работы первой страницы сайта**

Первая страница является своего рода навигатором по сайту. Она состоит из трех объектов типа HyperLink (ссылка), которые размещены на элементе управления Panel. Цель каждой ссылки – это перенаправление на другую страницу. Для того, чтобы при нажатии на ссылку она перенаправляла на нужную страницу необходимо задать в окне Properties свойство NavigateUrl.

**2.3 Принцип работы второй страницы сайта**

Вторая страница служит для заполнения таблицы markbook. В окне Design распложены два элемента DropDownList, один TextBox и кнопка Button. Элементы DropDownList берут информацию из таблиц sTudenT и subjects посредством интерфейса автоматического вывода столбца. Первый DropDownList отображает фамилии студентов, второй названия предметов.

При загрузке страницы происходит соединение с базой данных посредством метода Open(). При нажатии на кнопку происходит проверка правильности заполнения данных, то есть оценка должна находиться в интервале от 1 до 5, оценку нельзя ставить по одному и тому же предмету два раза. Затем, если данные введены правильно, создаются две строки запроса, первая выбирает id студента из таблицы sTudenT, вторая id предмета из таблицы subjects. Далее осуществляется сам запрос и при помощи метода ExecuteScalar() значения id присваиваются соответствующим переменным. Далее выполняется запрос на добавления данных, введенных на странице, в таблицу markbook, при помощи метода ExecuteNonQuery().

**2.4 Принцип работы третьей страницы сайта**

Третья страница служит для создания новых студентов и новых предметов. В окне Design расположены четыре элемента TextBox и два Button. Первая кнопка служит для занесения нового студента в базу данных, вторая для занесения нового предмета. При загрузке страницы происходит соединение с базой данных посредством метода Open(). При нажатии на одну из кнопок происходит проверка введенных данных, то есть поля нельзя оставлять пустыми. Далее создается запрос на добавление данных.

string.Format("INSERT INTO [subjects] ([Name]) VALUES ('{0}')", subj);

А затем при помощи метода ExecuteNonQuery(), данные заносятся в соответствующую таблицу. При успешном добавлении выдается соответствующее сообщение.

**2.5 Принцип работы четвертой страницы сайта**

Данная страница служит для просмотра успеваемости студента. В окне Design расположены элементы: DropDownList, Button и Table. Элемент DropDownList использует интерфейса автоматического вывода столбца и выводит список фамилий студентов, которые он берет из таблицы sTudenT. При загрузке страницы происходит соединение с базой данных посредством метода Open(). Далее выбирается id студента и id предметов из таблиц sTudenT и subjects. Причем для любого студента соответствует несколько id предметов, соответственно, для того чтобы их вывести создается объект SqlDataReader который считывает данные построчно при помощи метода Read(). Для подсчета количества строк используется цикл, в котором значение переменной при прочитывании строки увеличивается на 1. Это нужно для того, чтобы в дальнейшем знать длину массива, в котором будут храниться id строк таблицы markbook. При завершении цикла соединение закрывается. Далее соединение снова открывается, и цикл повторяется, в котором в массив id заносятся id строк, для которых прошла выборка, то есть каждому студенту соответствует несколько строк из таблицы markbook. В этом же цикле создаются запросы на выборку id предметов и оценок по этим предметам для строк, прошедших выборку. Затем соединение опять закрывается, следовательно, его нужно заново открыть. Далее запускается цикл, цель которого заменить id предметов на реальные названия предметов и сохранить их в массив. После всех выполненных операций необходимо заполнить таблицу. Таблица имеет формат:

|  |  |  |
| --- | --- | --- |
| № | Предмет | Оценка |

Заполнение таблицы происходит следующим образом: создается объект типа TableRow (строка), в ней создаются ячейки TableCell. В которые можно записывать данные через свойство Cell.Text. Все операции делаются в цикле, так как количество строк одна или больше.

**2.6 Принцип работы пятой страницы сайта**

Пятая страница служит для удаления студента из базы данных. На этой странице расположено два элемента: DropDownList, в котором происходит выбор фамилия студента, и Button, при нажатии на который происходит само удаление данных. При удалении, данные удаляются из двух таблиц: sTudenT и markbook. Для удаления данных используется запрос с ключевым словом «DELETE». Из таблицы sTudenT данные удаляются легко, а для удаления данных из таблицы markbook выполняются следующие операции: из таблицы sTudenT выбирается id студента, затем делается запрос на отбор строк в таблице markbook, соответствующих этому id (по столбцу idst), после этого выбираются id строк из таблицы markbook и только потом производится запрос на удаление. Причем количество строк может быть больше чем одна, следовательно для формирования списка конкретных строк применяется метод ExecuteReader(). Данные удаляются при помощи метода ExecuteNonQuery(). После выполнения всех операций выводится сообщение об удачном удаления студента из таблицы.

**2.7 Принцип работы шестой страницы сайта**

Седьмая страницы служит для поиска студента по базе данных. В окне Design находятся три элемента: TextBox, Button, Table. В элемент TextBox вводится фамилия студента для поиска. Элемент Table используется для вывода данных при успешном нахождении студента. Процедура поиска заключается в нахождении id студента по введенным данным в поле TextBox. Соответственно делается следующий запрос:

string.Format("SELECT id FROM [sTudenT] WHERE SecondName = '{0}'", SecondName);

Если id студента был найден, соответственно по величине он будет больше нуля, если нет, то он будет равен -1. Далее, если студент был найден, то по найденному id осуществляются запросы на выборку имени и отчества из той же таблицы sTudenT. Затем выдается сообщение об успешном нахождении студента. Для вывода данных в таблицу используется последовательность действий аналогичная пункту 2.5.

**2.8 Принцип работы седьмой страницы сайта**

Седьмая страница сайта используется для изменения фамилии имени и отчества студента. Все операции выполняются с таблицей sTudenT. Для изменения данных используется запрос с ключевым словом «UPDATE». В окне Design приложения расположены три элемента типа TextBox, один DropDownList и один элемент типа Button. В элементе DropDownList располагаются фамилии всех студентов из столбца SecondName. Три элемента типа TextBox служат для введения новых данных. При нажатии на кнопку происходит проверка введенных данных, то есть нельзя вводить студента с фамилией, которая уже используется в базе, нельзя вводить пустые поля. Для изменения данных используется запрос вида:

string.Format("UPDATE [sTudenT] SET SecondName = '{0}', FirstName = '{1}', oT4ecTBo = '{2}' WHERE id = '{3}'", new\_sec\_name, new\_fir\_name, new\_oT4ecTBo, id);

Как видно одним из параметров является id студента. Данный параметр находится путем запроса:

string.Format("SELECT id FROM [sTudenT] WHERE SecondName = '{0}'", DropDownList1.Text);

При успешном изменении данных выводится соответствующее сообщение.

**2.9 Принцип работы восьмой страницы сайта**

Данная страница используется для изменения оценок студента в таблице markbook. Для того чтобы изменить оценку необходимо получить id предмета и id студента. Эти id берутся из таблиц sTudenT и subject. Соответственно делаются два запроса. Параметром для первого запроса, который выбирает id студента, является текущее значение элемента DropDownList1, а для второго текущее значение элемента DropDownList2. Затем при помощи ключевого слова «UPDATE» изменяется таблица. Запрос на изменение выглядит следующим образом:

string.Format("UPDATE [markbook] SET mark = '{0}' WHERE idst = '{1}' AND idsubj = '{2}'", TextBox1.Text, idst, idsubj);

При успешном изменении данных, выводится соответствующее сообщение.

**2.10 Принцип работы девятой страницы сайта**

Данная страница служит для удаления предметов из базы данных. В окне Design расположены только два элемента: DropDownList и Button. Соответственно в элемент DropDownList заносятся данные из таблицы subjects, а именно столбец Name. Принцип работы данной страницы заключается в следующем: сначала по значению предмета, выбранного в элементе DropDownList, выполняется запрос на получение id предмета из таблицы subjects, затем по полученному id выбираются все подходящие записи в таблице markbook. Выборка данных производится при помощи метода Read() принадлежащего классу SqlDataReader. Полученные данные заносятся в массив. Затем выполняется сам запрос на удаление подходящих строк их таблицы markbook. После выполнения всех выше описанных операций выполняется операция на удаление строки из таблицы subjects.

**2.11 Принцип работы десятой страницы сайта**

Последняя страница сайта служит для изменения названия предмета. В окне Design расположены два элемента DropDownList и Button. Соответственно в элемент DropDownList заносятся данные из таблицы subjects, а именно столбец Name. Затем при помощи запроса с ключевым словом «SELECT» выполняется запрос на поиск id предмета в таблице subjects. После этого выполняется проверка правильности ввода, а именно нет ли совпадающих предметов, не пустое ли поле и т.д. Если все проверки были пройдены успешно, то выполняется запрос на обновление таблицы subjects, для этого используется запрос с ключевым словом «UPDATE». Если все было выполнено успешно, то выдается соответствующее сообщение, если нет, то выдается соответствующее сообщение об ошибке.

**3. Технологическая часть**

**3.1 Настройка приложения**

Для настройки приложения необходимо изменить путь к базе данных. Данный путь указывается на каждой странице сайта и выглядит следующим образом:

@"Data Source=.\SQLEXPRESS;AttachDbFilename='C:\путь к базе данных\ Database.mdf';Integrated Security=True;User Instance=True";

После изменения пути, приложение будет работать. Для запуска приложения необходимо разместить сайт в домене, поддерживающем язык ASP.NET, либо скомпилировать приложение. Соответственно при компиляции будет запущен виртуальный сервер, на котором будет запущено данное приложение.

**3.2 Работа с сайтом**

При запуске приложение выглядит следующим образом:

Рис. 5. Стартовая страница

Если в базе нет никакой информации, то ее нужно добавить, для этого нужно щелкнуть на ссылку «Добавить новые данные». Появится окно, в котором можно добавить нового студента, или новый предмет. Диалоговое окно изображено на рисунке:

Рис. 6. Добавление нового студента

Для добавления нового студента необходимо заполнить все поля и нажать на кнопку подтвердить, после нажатия появится соответствующее сообщение. Для добавления нового предмета необходимо проделать аналогичные действия, то есть ввести название предмета и также нажать на кнопку подтвердить.

После того как предметы и студенты были добавлены, необходимо заполнить поля успеваемости студента, то есть нужно ввести оценки по каждому из предметов. Это можно сделать на следующей странице:

Рис. 7. Заполнение успеваемости

Для заполнения успеваемости студента, необходимо выбрать определенного студента из раскрывающего списка, затем необходимо выбрать соответствующий предмет, вести оценку и нажать кнопку подтвердить. После заполнения успеваемости можно посмотреть результат на следующей странице.

Рис. 8. Успеваемость студента

Соответственно чтобы посмотреть информацию по каждому из студентов, его необходимо выбрать из списка, и нажать на кнопку. Появится таблица, в которой наглядно видно название предмета и оценку по этому предмету.

Бывают такие ситуации, что название предмета или инициалы студента были введены неверно, для данных целей предусмотрены страницы «изменить ФИО студента» и «изменить название предмета». Страница для изменения «ФИО студента» выглядит следующим образом:

Рис. 9. Изменение ФИО студента

Для того чтобы изменить ФИО, необходимо выбрать студента из списка, затем в поля «Новая фамилия», «Новое имя», «Новое отчество», необходимо ввести данные. После ввода необходимо нажать на кнопку изменить, если вы хотите изменить информацию, или нажать на ссылку «вернуться на главную», если не хотите вносить изменений. После внесения изменений появится сообщение об ошибке, если произошли ошибки ввода данных, либо сообщение об успешном выполнении данной операции. Для изменения названия предмета предусмотрена следующая страница:

Рис. 10. Изменение названия предмета

Принцип работы ее идентичен странице служащей для изменения ФИО студента. Также нужно выбрать предмет, затем ввести новое значение и нажать на кнопку изменить.

Бывают такие ситуации, что необходимо удалить того или иного студента из базы данных, или удалить какой-либо предмет. Для этих целей используются страницы «Удалить студента», «Удалить предмет». На рис. 11 показан внешний вид станицы «Удалить студента»:

Рис. 11. Удаление студента

Для того чтобы удалить студента необходимо его выбрать из списка и нажать на кнопку удалить. После нажатия будет выведено соответствующее сообщение.

Страница «Удалить предмет» практически идентична странице «Удалить студента». Ее внешний вид представлен ниже:

Рис. 12. Удаление предмета

Для удаления предмета, выбирается соответствующий предмет, и после нажатия на кнопку удалить он будет удален.

Данный сайт позволяет производить поиск студентов по базе. Интерфейс изображен на рис. 13:

Рис. 13. Поиск студента

Для того чтобы найти студента, необходимо ввести его фамилию. Если студент был найден, то выводится соответствующее сообщение об успешном нахождении студента. Также будет отображена успеваемость студента в таблице.

**3.3 Системные требования**

Для нормального функционирования клиентского приложения требуется: операционная система Windows® 98/2000/XP, процессор Intel® Pentium 800 МГц, 128 Мб ОЗУ, 32 Мб Direct X – совместимая 3D видеокарта, 1 Мб свободного места на жестком диске, клавиатура, мышь, Microsoft.NET Framework 2.0.

**Заключение**

Данная курсовая научная работа показывает возможности языка ASP.NET 2005. Данный язык на данный момент является самым современным и самым функциональным по сравнению с другими языками. Некоторые моменты автоматизированы, что облегчает разработку приложений. Создание баз данных автоматизировано. Просмотр баз данных можно вести прямо из среды разработки, что является довольно удобным. Запросы к базе данных являются простыми для понимания, следовательно, их легко писать. Данную работу можно усовершенствовать, к примеру, добавив в базу данных такие столбы как «семестр», «преподаватель», «год сдачи экзамена» и т.д. Также можно усовершенствовать дизайн приложения.

**Список использованной литературы**

1. Гаряка А.А. Основы ASP.NET 2.0: Учебное пособие / А.А. Гаряка. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. – 296 с.: ил.-(Основы информационных технологий)