Из цикла лекций **«Технологии разработки Internet-приложений»** для студентов 4-го курса кафедры Компьютерных технологий физического факультета Донецкого национального университета

Технологии разработки **Internet-**приложений

ASP.NET приложения – работа с базами данных посредством ADO.NET

проф. В.К.Толстых, www.tolstykh.com

Класс SqlConnection

Для того, чтобы получить возможность взаимодействия с источником БД посредством ADO.NET, приложение должно первым делом установить подключение к этому источнику данных. Классом представляющим физическое подключение к **SQL Server**, является **SqlConnection** из пространства имен System.Data.SqlClient. Это скрытый (не наследуемый) клонируемый класс, реализующий интерфейс *IDbConnection*.

У класса **SqlConnection** есть два конструктора. Один из них используется по умолчанию и не имеет параметров, а другой имеет один параметр-строку подключения:

- `public SqlConnection();`
- `public SqlConnection(string);`

Следующий код демонстрирует типичный способ конфигурирования и установки подключения к SQL Server:

```
string connString = "SERVER=...;DATABASE=...;LOGIN=...;PWD=...";
SqlConnection conn = new SqlConnection(connString);
conn.Open();
...
conn.Close();
```

Свойства класса SqlConnection

Свойство	Описание
`ConnectionString`	Возвращает и позволяет задать строку, используемую для открытия подключения к базе данных .
`ConnectionTimeOut`	Возвращает длительность тайм-аута в секундах, пор истечении которого попытка подключения считается неудачной.
`DataBase`	Возвращает имя используемой БД
`DataSourse`	Возвращает имя экземпляра SQL Server, к которому производится подключение. Соответствует атрибуту Server строки подключения.
`PacketSize`	Возвращает размер сетевых пакетов, передаваемых при взаимодействии с SQL Server, в байтах. Значение по умолчанию – 8192, но может иметь любое значение в диапазоне от 512 до 32676.
`ServerVersion`	Возвращает строку, содержащую номер версии текущего экземпляра SQL Server в форме *старший.младший.выпуск*
`State`	Возвращает информацию о текущем состоянии подключения: открыто оно или закрыто. По умолчанию подключение закрыто.
`StatisticEnabled`	Разрешает извлечение статической информации через текущее подключение.
`WorkStationId`	Возвращает сетевое имя клиента.

Методы класса SqlConnection

Метод	Описание
`BeginTransaction`	Начинает транзакцию БД. Позволяет задать имя и уровень изоляции и уровень изоляции транзакции.
`ChangeDataBase`	Позволяет сменить текущую базу данных подключения. Его параметром является имя базы данных.
`Close`	Закрывает подключение к базе данных. Этот метод используется для закрытия открытого ранее подключения
`CreateCommand`	Создает и возвращает объект SqlCommand, связанный с подключением.
`Dispose`	Вызывает метод Close.
`EnlistTransaction`	Связывает подключение с заданной локальной или распределенной транзакцией.
`GetSchema`	Извлекает информацию схемы для заданного уровня(то есть таблиц или баз данных).
`ResetStatistic`	Выполняет сброс службы статистики.
`RetrieveStatistic`	Возвращает хеш-таблицу с информацией о подключении, такой как сведения о пересланных данных, пользователях и транзакциях.
`Open`	Открывает подключение к базе данных.

Примечание!

Следует заметить, что когда объект подключения выходит из области видимости приложения, он автоматически закрывается. Позднее, но не обязательно сразу, сборщик мусора удалит этот объект, однако подключение останется открытым, поскольку сборщик не знает особенностей каждого объекта и не может выполнять действия, которые определяются их семантикой.

Так что, не забывайте явно закрывать подключение, вызывая метод `Close` или `Dispose` до того, как объект выйдет за пределы видимости.

Выполнение команд

После установки между клиентом и базой данных физического канала связи можно начинать подготовку и выполнение команд. Объектная модель **ADO.NET** определяет два типа командных объектов – традиционный одноразовый командный объект и ***адаптер данных***. Первый выполняет команду **SQL** или хранимую процедуру и возвращает курсор, с помощью которого можно осуществить проход по записям и прочитать данные. Пока курсор используется, подключение активно и открыто.

Адаптер данных – более мощный объект, сам он использует одноразовый командный объект и курсор, а программисту предоставляет интерфейс, позволяющий работать с данными после отключения от их источника. Для этого адаптер извлекает данные и загружает их в контейнерный класс **DataSet** или **DataTable**.

Класс SqlCommand

Данный класс представляет оператор **SQL Server** или хранимую процедуру. Объект **Command** исполняет запрос **SQL**, который может быть в форме встроенного текста, процедуры сервера или прямого доступа к таблице. Рассмотрим набор конструкторов класса ***SqlCommand***:

- `Public SqlCommand();`
- `Public SqlCommand(string);`
- `Public SqlCommand(string, SqlConnection);`
- `Public SqlCommand(string, SqlConnection, SqlTransaction);`

Аргумент ***string*** содержит текст подлежащий выполнению команды или имя хранимой процедуры, аргумент `SqlConnection` – объект предназначенного для этой цели подключения, а аргумент `SqlTransaction`, если он создан, представляет транзакционный контекст, в котором будет выполняться команда. Командные объекты ADO.NET никогда сами не открывают подключение. Программист должен явно связать его объект с объектом команды, а также явно открыть и закрыть подключение.

Класс SqlConnection (свойства)

Свойство	Описание
`CommandText`	Возвращает и позволяет задать подлежащий выполнению оператор или имя хранимой процедуры .
`CommandTimeOut`	Возвращает и позволяет задать максимальную длительность выполнения команды в секундах.
`CommandType`	Указывает и позволяет задать способ интерпретации содержимого свойства CommandText
`Connection`	Возвращает и позволяет задать объект подключения, который будет использоваться для выполнения команды. По умолчанию имеет значение `null`.
`NotificationAutoEnList`	Указывает будет ли команда автоматически связана с сервисом уведомлений SQL Server 2005.
`Parameters`	Возвращает коллекцию параметров команды
`Transaction`	Возвращает и позволяет задать транзакцию, в рамках которой будет выполнятся команда. Транзакция д.б. связана с тем же подключением, что и команда.
`UpdateRowSourse`	Указывает и позволяет определить способ применения результатов запроса к обновляемой строке.

Класс SqlConnection (методы)

Метод	Описание
`BeginExecuteNonQuery`	Выполняет команду не являющуюся запросом на выборку, без блокирования записей.
`BeginExecuteReader`	Выполняет запрос на выборку без блокирования записей.
`Cancel`	Осуществляет попытку отменить выполнение команды.
`CreateParameter`	Создает новый экземпляр объекта `SQLParameter.`
`EndExecuteNonQuery`	Завершает асинхронное выполнение команды, не являющееся запросом на выборку.
`EndExecuteReader`	Завершает выполнение асинхронного запроса на выборку.
`ExecuteNonQuery`	Выполняет команду, не являющуюся запросом на выборку, и возвращает количество обработанных строк.
`ExecuteScalar`	Выполняет запрос на выборку и возвращает значение первого столбца первой строки результирующего набора данных
`ExecuteReader`	Выполняет запрос на выборку и возвращает курсор, доступный только для чтения.
`ResetCommandTimeout`	Сбрасывает значение тайм-аута команды, устанавливая тайм-аут по умолчанию.

Способы выполнения команд

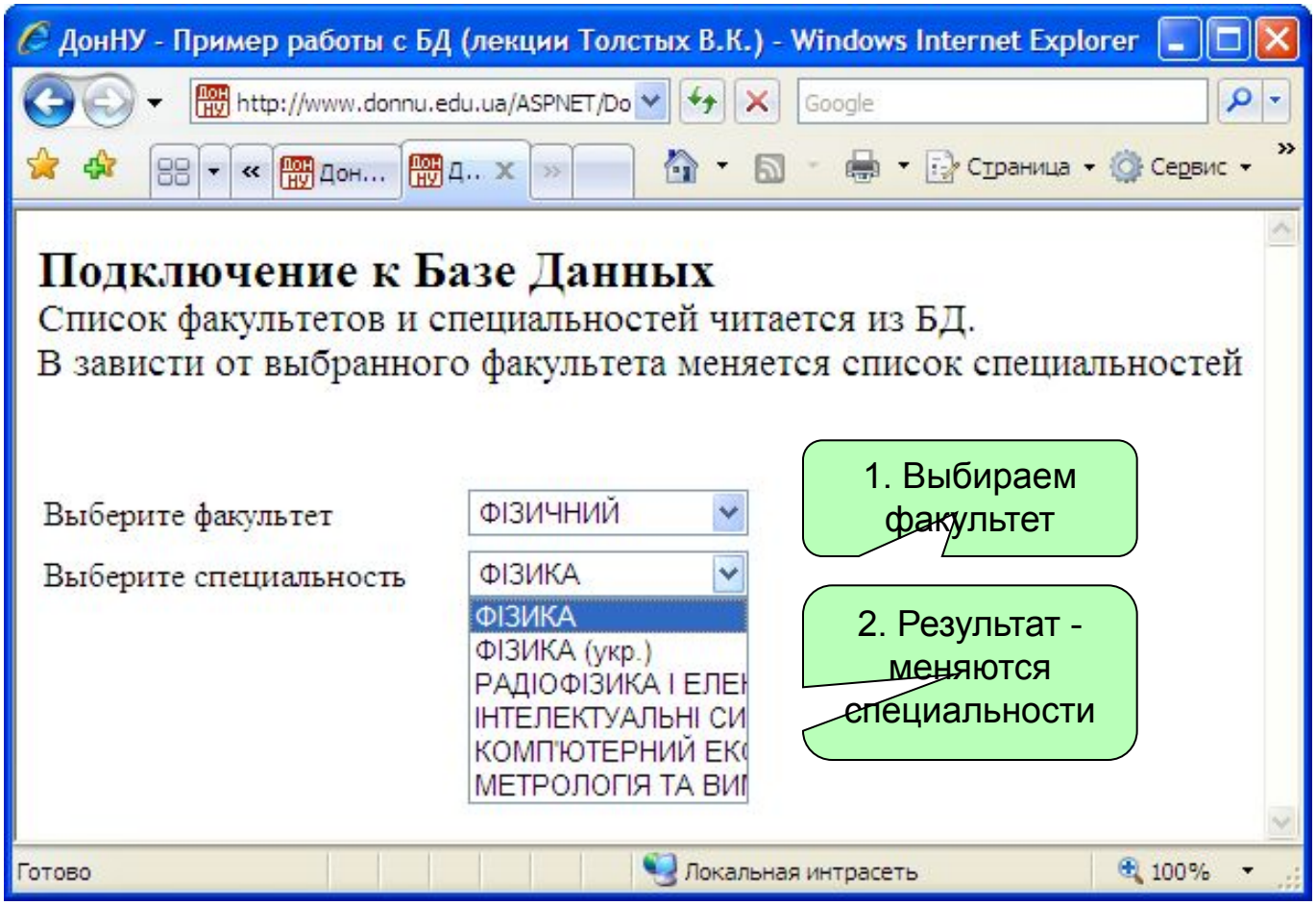
Для выполнения команд используются следующие методы, **ExecuteNonQuery, ExecuteReader, ExecuteScalar.** Они действуют практически одинаково, но возвращают разные значения. Обычно для выполнения операций обновления, таких как UPDATE, INSERT и DELETE, используется метод **ExecuteNonQuery,** возвращающий количество задействованных в операции строк. Для операций других типов, таких как SET или CREATE, он возвращает значение -1.

Метод **ExecuteReader** предназначен для выполнения запросов на выборку. Он возвращает объект чтения данных – экземпляр класса **SqlDatareader**. Если попытаться выполнить посредством метода **ExecuteReader** запрос на обновление, то обновление данных будет произведено успешно, но вы не получите обработанных строк.

Когда требуется получить единственное значение, удобнее всего воспользоваться методом **ExecuteScalar.** Он прекрасно работает с операторами SELECT COUNT или командами, возвращающими агрегированные данные. Этот метод можно вызывать и для обычных запросов – в таком случае вы получите значение первого столбца первой строки результирующего набора, а все остальные будут проигнорированы.

Пример подключения к БД

Пример на сервере ДонНУ



Код C# примера

```
13      protected void Page_Load(object sender, EventArgs e)
14      {
15          if (!IsPostBack)
16          {
17              Spec(Fac());
18          }
19      }
20      protected string Fac()
21      {   // смена факультетов
22          DropDownListFac.Items.Clear();
23          dnuClassSelectFak SelectFac  = new dnuClassSelectFak(1);
24          for (int i = 0; i < SelectFac.CountFak(); i++)
25          {
26              DropDownListFac.Items.Add(SelectFac.GetFak(i).ToUpper());
27              DropDownListFac.Items[i].Value = SelectFac.GetKod(i);
28          }
29          return DropDownListFac.Items[DropDownListFac.SelectedIndex].Value;
30      }
31      // ЗАПОЛНЕНИЕ СПЕЦИАЛЬНОСТЕЙ
32      protected string Spec(string num)
33      {   // смена специальностей
34          DropDownListSpec.Items.Clear();
35          dnuClassSelectSpec SelectSpec = new dnuClassSelectSpec(num);
36          for (int i = 0; i < SelectSpec.CountSpec(); i++)
37          {
38              DropDownListSpec.Items.Add(SelectSpec.GetSpec(i));
39              DropDownListSpec.Items[i].Value = SelectSpec.GetKod(i);
40          }
41          return DropDownListSpec.Items[DropDownListSpec.SelectedIndex].Value;
42      }
43      protected void DropDownListFac_SelectedIndexChanged(object sender, EventArgs e)
44      {   // при смене индекса произвести смену списка специльностей
45          string IndexFac = DropDownListFac.Items[DropDownListFac.SelectedIndex].Value;
46          Spec(IndexFac); //Spec("1");
47      }
48  }
```

Заполнение факультетов элемента DropDownList данными из БД

очистка предыдущих значений

Заполнение специальностей элемента DropDownList данными из БД

Код из класса dnuClassSelectFuc

```
private List ListFak = new List();
   private List ListFakIndex = new List();

public Select_fuc(string SqlStringConn)
{
      string Zapros = "Select * From n_fak Order By nfak";
      string SqlStringConnAb = SqlStringConn;

      using (SqlConnection conn = new SqlConnection(SqlStringConnAb))
      {
         SqlCommand cmd = new SqlCommand(Zapros, conn);
         cmd.Connection.Open();

         SqlDataReader reader = cmd.ExecuteReader();
         while (reader.Read())
         {
            ListFak.Items.Add( reader["nfak"].ToString() );
            ListFakIndex.Items.Add( reader["fak"].ToString() );
         }
         reader.Close();
      }}

   //возвращает название факультета по заданному индексу
   public string GetFac(int num)
   {
      string s = "Факультет не найден!";
      if ((num < 0) || (num > 14))
         return s;
      else
         return ListFak.Items[num].ToString();
   }
```

Строим **SQL**-запрос

Подключаемся к базе

Метод класса **GetFac** - возвращает название факультета по заданному индексу

Трёхуровневая модель работы с данными