Лекция №8
по курсу
«Системное программирование»
тема: «LINQ запросы, SQL запросы»

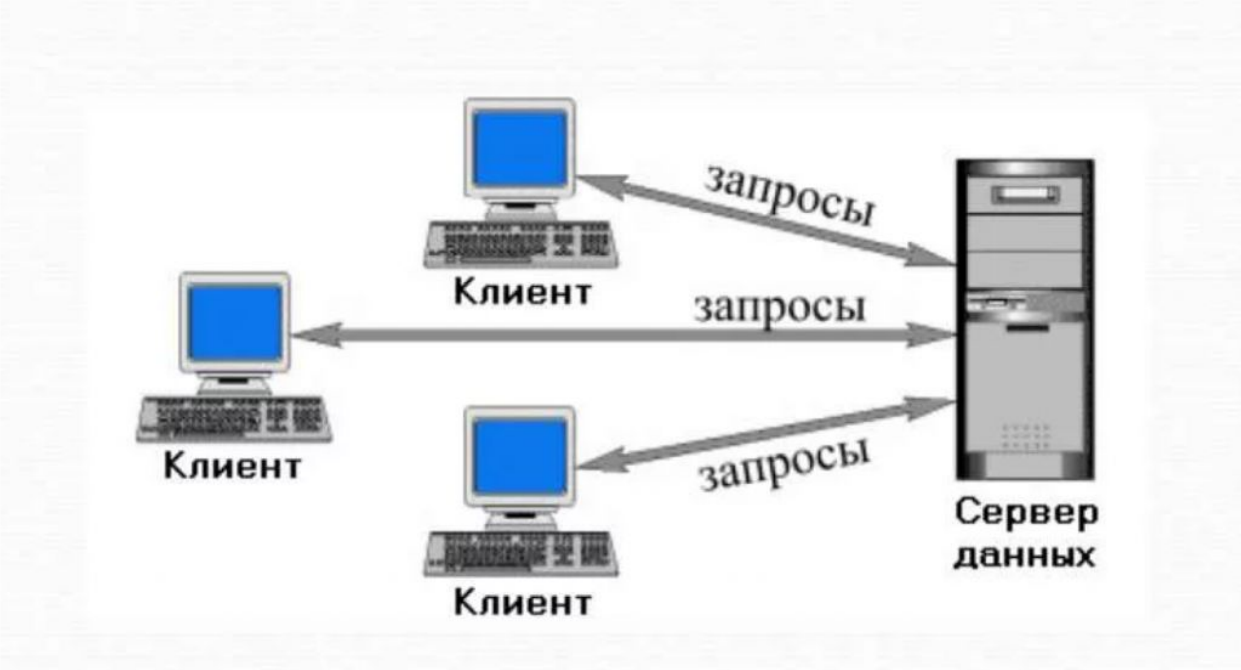
Лектор: д.т.н., Оцоков Шамиль Алиевич,

email: otsokovShA@mpei.ru

Москва, 2021

СУБД – системы управления база данных

MSSQL – сетевые субд
Oracle
MySQL (распространена в интернете)
MsAccess – локальная субд



SQL – Structured Query Language

- **SQL** – это структурированный язык запросов к реляционным базам данных (БД).
- SQL – декларативный язык, основанный на операциях реляционной алгебры.
- Стандарты SQL, определённые Американским национальным институтом стандартов (ANSI):
  - ✓ SQL-1 (SQL/89) – первый вариант стандарта.
  - ✓ **SQL-2 (SQL/92) – основной расширенный стандарт.**
  - ✓ SQL-3 (SQL/1999, SQL/2003) – относится к объектно-реляционной модели данных.
- Подмножества языка SQL:
  - ✓ **DDL** (Data Definition Language) – команды создания/изменения/удаления объектов базы данных (*create/alter/drop*);
  - ✓ **DML** (Data Manipulation Language) – команды добавления/модификации/удаления данных (*insert/update/delete*), а также команда извлечения данных *select;*
  - ✓ DCL (Data Control Language) – команды управления данными (установка/снятие ограничений целостности). Входит в подмножество DDL.

Команда SELECT – выборка данных

Общий синтаксис:

SELECT [{ ALL | DISTINCT }] { *список_вывода* | * }
FROM *имя_таблицы1* [*алиас1*] [, *имя_таблицы2* [*алиас2*].,..]
[WHERE *условие_отбора_записей*]
[GROUP BY { *имя_поля* | *выражение* }.,..]
[HAVING *условие_отбора_групп*]
[UNION [ALL] SELECT ...]
[ORDER BY *имя_поля1* | *целое* [ASC | DESC]
[, *имя_поля2* | *целое* [ASC | DESC].,..]];

Примеры:

select * from departs;
select name, post from emp;

Команды DDL

CREATE – создание объекта.
ALTER – изменения структуры объекта.
DROP – удаление объекта.
Общий вид синтаксиса команд DDL:

create
alter *тип_объекта имя_объекта*
[параметры];
drop

Создание таблиц

CREATE TABLE [*имя_схемы*.]*имя_таблицы*
(*имя_поля* *тип_данных* [(*размер*)] [NOT NULL]
[DEFAULT *выражение*]
[*ограничения_целостности_поля*...]
.,..
[, *ограничения_целостности_таблицы* .,..]
)
[*параметры*];

ограничения_целостности (ОЦ):
[CONSTRAINT *имя_ОЦ*] *название_ОЦ* [*параметры*]

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework

Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Connection, Command, DataReader, DataSet и DataAdapter

Чтобы использовать один и тот же набор объектов для разных источников данных, необходим соответствующий провайдер данных.

- Провайдер для OLE DB
- Провайдер для MS SQL Server
- Провайдер для Oracle

…

ADO.NET


ADO.NET
Провайдер данных
Подключенный уровень
Connection
Command
DataReader
DataAdapter
Отключенный уровень
DataSet
DataTable
Приложение
База данных

Работа с базами данных

Платформа .NET предоставляет пространство имен System.Data.dll, используя классы для взаимодействия с реляционными системы баз данных. Эти классы входят в состав ADO.NET, в котором есть три концептуальные части:

1. Подключенный слой
2. Отключенный слой
3. Entity Framework

Работа с базами данных

ExecuteNonQuery - это метод, выполняемый для объекта Command (SqlCommand), используемый для выполнения оператора, заданный объектом Command, и возвращает не набор (-ы) результатов, а количество строк, затронутых в база данных при выполнению запроса. Он в основном вызывается для объекта Command, для запросов insert, delete, и update.

Работа с базами данных

Listing 12-4. ExecuteNonQuery on Insert Command

```
string connectionString = "YOUR CONNECTION STRING HERE";
SqlConnection con = new SqlConnection(connectionString);
con.Open();

string command = "Insert into Student values(1,'Hamza Ali')";
SqlCommand cmd = new SqlCommand(command, con);
int result = cmd.ExecuteNonQuery();
con.Close();
if (result > 0)
  Console.WriteLine("Data is Inserted");
else
  Console.WriteLine("Error while inserting");
```

Работа с базами данных

ExecuteScalar

Метод ExecuteScalar также выполняется для объекта Command в том случае, если вы пишете запросы, которые вернуть одно значение. Это тот случай, когда вы используете агрегатные функции в своих запросах.

Работа с базами данных

```
string con = "YOUR CONNECTION STRING HERE";
string command = "select count(*) from Student";
SqlCommand cmd = new SqlCommand(command, con);
var noOfStudents = cmd.ExecuteScalar();
con.Close();
Console.WriteLine(noOfStudents);
```

Работа с базами данных

ExecuteReader

Метод ExecuteReader также вызывается для объекта Command, из которого вам нужно получить данные, т. Е. в случае запроса «выбрать». Этот метод возвращает объект SqlDataReader, который остается подключенным к база данных все время, пока читатель открыт. SqlDataReader - это набор результатов только для пересылки, что означает, что вы не может перейти к предыдущей записи и может читать по одной записи за раз. Вы можете прочитать конкретный столбец таблица по номеру индекса или имени столбца.

Работа с базами данных

```
string con = "YOUR CONNECTION STRING HERE";

string command = "select * from Student";
SqlCommand cmd = new SqlCommand(command, con);

SqlDataReader reader = cmd.ExecuteReader();

int StudentID = 0;
string StudentName = null;
if (reader.HasRows)
{
  while (reader.Read())
  {
  StudentID = int.Parse(reader[0].ToString());//0 index means first clm in the table which
  is StudentID
  StudentName = reader["StudentName"].ToString();//it will fetch the value of provided clm
  name
  }
}
reader.Close();
con.Close();

Console.WriteLine("ID is: " + StudentID);
Console.WriteLine("Name is: " + StudentName);
```

Работа с базами данных

Отключенный слой

На отключенном уровне вы обычно используете DataSets и DataTables, которые копируют структуру реляционного база данных в памяти. DataSet создается в результате выполнения запроса к подключенному
база данных. Им можно управлять в памяти, а изменения в базе данных происходят с помощью DataAdapter.

DataTable и DataSets - еще один способ извлечения результатов из базы данных.

Работа с базами данных

DataTable такой же, как DataReader, за исключением того, что DataTable может перемещаться вперед и назад. Он отключен из базы данных, и вы можете вносить изменения в данные в DataTable и фиксировать или обновлять базу данных с помощью

DataSet - это контейнер DataTables. Вы можете написать запрос, который возвращает несколько наборов результатов и может быть содержится в DataSet. Затем вы можете выполнять дальнейшие операции с полученным DataSet, такие как фильтрация или сортировка и т.д. Эти обновления размещаются в памяти.

Работа с базами данных

DataAdapter - важный объект при работе с отключенным слоем. Он действует как мост между данные в памяти и базе данных. DataAdapter заполняет DataTable или DataSets и повторно подключает данные в

память в базу данных. Вы можете выполнять запросы вставки, обновления, удаления или чтения, пока данные находятся в памяти.

а затем повторно подключитесь к базе данных, чтобы зафиксировать изменения.

На следующем слайде показано, как выполнять операции, ориентированные на базу данных, с использованием отключенного уровня:

Работа с базами данных

DataAdapter - важный объект при работе с отключенным слоем. Он действует как мост между данные в памяти и базе данных. DataAdapter заполняет DataTable или DataSets и повторно подключает данные в

память в базу данных. Вы можете выполнять запросы вставки, обновления, удаления или чтения, пока данные находятся в памяти.

а затем повторно подключитесь к базе данных, чтобы зафиксировать изменения.

На следующем слайде показано, как выполнять операции, ориентированные на базу данных, с использованием отключенного уровня:

Работа с базами данных

```
string con = "YOUR CONNECTION STRING HERE";

string command = "select * from Student";
SqlDataAdapter ad = new SqlDataAdapter(command, con);

DataTable tbl = new DataTable();
ad.Fill(tbl);//Now the data in DataTable (memory)
con.Close();//connection closed

foreach (DataRow item in tbl.Rows)
{
  Console.WriteLine("ID is: " + item[0]);
  Console.WriteLine("Name is: " + item[1]);
}
```

Работа с базами данных

Когда вызывается метод DataAdapter Fill, будет выполнен запрос, а метод Fill () заполнит DataTable . DataTable не нужно поддерживать соединение для заполнения данных, что не относится к DataReader (на подключенном уровне). Это отключенный уровень, и он имеет лучшую производительность, чем подключенный уровень, поскольку он имеет дело с данными, присутствующими в память, к которой имеется быстрый доступ.

Вы также можете использовать DataSet вместо DataTable, если ожидаете нескольких наборов результатов.

DataSet имеет свойство Table, с помощью которого вы можете перебирать конкретные данные таблицы.

Работа с базами данных

Listing 12-8. Insertion of data (disconnected layer)

```
string connectionString = "YOUR CONNECTION STRING HERE";
SqlConnection con = new SqlConnection(connectionString);
con.Open();

string command = "select * from Student";//Currently has One Row(for example)
SqlDataAdapter ad = new SqlDataAdapter(command, con);

DataTable tbl = new DataTable();
ad.Fill(tbl);//Now the data in DataTable (memory)

//Data in Memory (One Row)
foreach (DataRow item in tbl.Rows)
{
  Console.WriteLine("ID is: " + item[0]);
  Console.WriteLine("Name is: " + item[1]);
}

//New Record to add in DataTable
DataRow newRow = tbl.NewRow();
newRow["StudentID"] = 2;
newRow["StudentName"] = "Ali Asad";
tbl.Rows.Add(newRow);
```

Работа с базами данных

```
//Two Rows(As new row added to DataTable)
foreach (DataRow item in tbl.Rows)
{
  Console.WriteLine("ID is: " + item[0]);
  Console.WriteLine("Name is: " + item[1]);
}

//Now newRow has to add in Database(Pass newRow Parameters to this insert query)
string newCommand = @"Insert into Student(StudentID,StudentName)
                                Values(@StudentID,@StudentName)";

   SqlCommand insertCommand = new SqlCommand(newCommand, con);

   //Create the parameters
   insertCommand.Parameters.Add(new SqlParameter("@StudentID", SqlDbType.Int, Int32.
   MaxValue,"StudentID"));
   insertCommand.Parameters.Add(new SqlParameter("@StudentName",  SqlDbType.VarChar,
   40,"StudentName"));

   //Associate Insert Command to DataAdapter so that it could add into Database
   ad.InsertCommand = insertCommand;

   ad.Update(tbl);

   con.Close();
```

Using

Ключевое слово `using` имеет три основных применения:

- Инструкция using определяет область, по завершении которой объект удаляется.
- Директива using создает псевдоним для пространства имен или импортирует типы, определенные в других пространствах имен.
- Директива using static импортирует элементы из одного класса.

Using

Ключевое слово Using упрощает работу с объектами которые реализуют интерфейс IDisposable.

Интерфейс IDisposable содержит один метод .Dispose(), который используется для освобождения ресурсов, которые захватил объект. При использовании Using не обязательно явно вызывать .Dispose() для объекта.

```
using (SqlConnection conn = new SqlConnection()) {
// какая-нибудь SQL операция
}
```

При этом компилятор генерирует следующий код:

```
SqlConnection conn = new SqlConnection();
try {

} finally {
    // здесь для conn вызывается .Dispose()
}
```

Using блоки делают код более читабельным и компактным.

Using

```
using (var font1 = new Font("Arial", 10.0f))
{
    byte charset = font1.GdiCharSet;
}
```

```
{
  var font1 = new Font("Arial", 10.0f);
  try
  {
    byte charset = font1.GdiCharSet;
  }
  finally
  {
    if (font1 != null)
      ((IDisposable)font1).Dispose();
  }
}
```