

Разработка .NET приложений

Лекция 16.
ADO.NET

Базы данных

- Рассматриваем только реляционные базы данных
- Сервера баз данных: MS SQL Server, Oracle, MySQL, DB2, Foxpro, FireBird, PostgreSQL, SQLite...
- Хранят данные в виде таблиц
- Хранят отношение между таблицами
- Обеспечивают целостность данных, отказоустойчивость. ACID
 - Поддерживают механизм транзакций
- Обеспечивают доступ к данным посредством языка SQL

Язык SQL

- Стандарты SQL 86, SQL 89, ... SQL 2016
- Содержит команды в текстовом виде
- Запросы компилируются самим сервером
- Не все сервера баз данных поддерживают стандарты в полном объеме или имеют некоторые модификации команд описанных в стандартах
- Модификации языка
 - T-SQL (transact SQL, Microsoft)
 - PL-SQL (Oracle), PL/pgSQL (PostgreSQL)
 - MySQL

ADO.NET

- Предоставляет простой доступ к локальным и удаленным реляционным базам данных
- Предоставляет единую модель доступа к различным базам данных
- Доступ к базам данных осуществляют провайдеры данных характерные (“заточенные”) для данного сервера баз данных
 - Повышает производительность
 - Позволяет использовать специфические особенности сервера баз данных

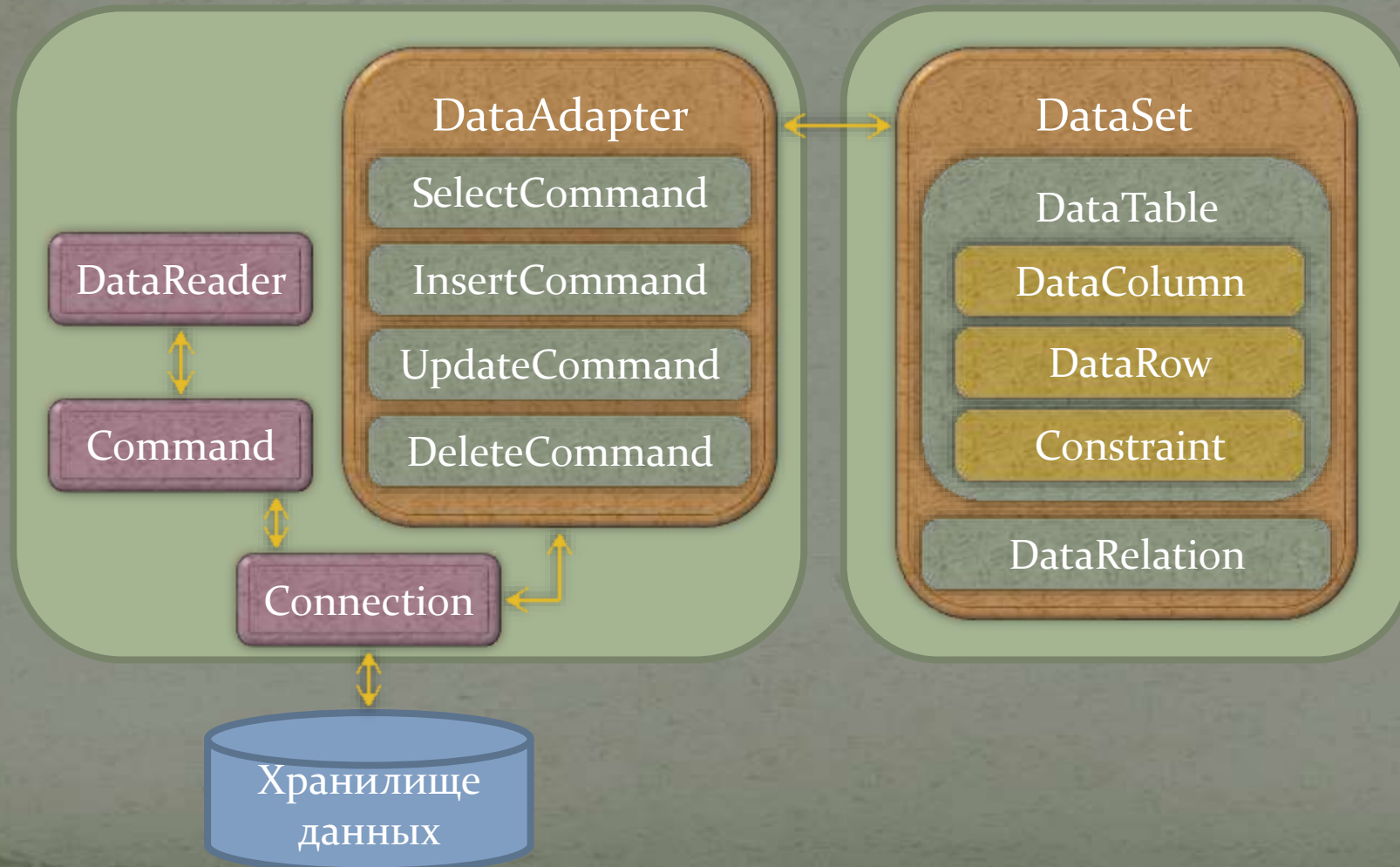
Модели доступа

- Присоединенная
 - Открывается соединение с БД
 - Соединение остается на весь период работы
- Отсоединенная
 - Данные загружаются из БД. Сохраняется локальная копия данных. Соединение закрывается
 - Данные модифицируются пользователем (при отсутствии соединения с БД)
 - Данные сохраняются в БД (кратковременно открывая соединение)
- Entity Framework – ORM (Object-relational mapping)
 - Предоставляет связь между объектами в коде и записями в реляционной базе данных.

ADO.NET

Присоединенный уровень

Отсоединённый уровень



ADO .NET

- Единая модель доступа к реляционным данным
 - Разные базы – схожие модели
- Провайдеры данных
 - Обеспечивает набор классов для доступа к конкретной БД
 - Провайдеры в .NET:
 - SqlConnection – MS SQL Server, начиная с 2020
 - Odbc - ODBC
 - OleDb – OLE DB
 - OracleClient – Oracle
 - Также доступны провайдеры для
 - FireBird, DB2, MySQL, PostgreSQL, ...
- Пространства имен (одноименные Nuget Package в .NET 5+)
 - System.Data – основное
 - System.Data.OleDb – Компоненты OLE DB провайдера
 - System.Data.SqlClient – Компоненты MS SQL Server провайдера
 - System.Data.SqlClientCe – Компоненты MS SQL Server Mobile провайдера
 - System.Data.Odbc – Компоненты ODBC провайдера
 - System.Data.OracleClient – Компоненты Oracle провайдера

Классы

- Компоненты модели
 - XXXConnection – соединение с БД
 - XXXCommand - команда
 - XXXParameter – параметр команды
 - XXXDataReader – чтение результата запроса
 - XXXTransaction – транзакция
 - XXXTableAdapter – адаптер данных
- Где XXX название провайдера
- Например:
 - SqlConnection – соединение с SQL Server
 - OdbcCommand – команда для ODBC провайдера
 - OracleParameter – параметр команды для Oracle
 - OleDbDataReader – reader для OLE DB Провайдера
 - SqlTransaction – транзакция для SQL Server
 - SqlTableAdapter – адаптер данных для SQL Server

Класс SqlConnection

- Класс соединения с БД MS SQL Server
- Содержит параметры соединения (строку соединения)
- Open() – открывает соединение с базой данных
- Close() – закрывает соединение
- Необходимо явное закрытие соединения
 - Необходимо вызывать Close()
 - Реализует IDisposable. При этом закрывает соединение (Close())
- Пример:

```
string cs =  
    "Data Source=(local);" +  
    "Initial Catalog=AdventureWorks;" +  
    "Integrated Security=SSPI;";  
using(SqlConnection sc =  
    new SqlConnection(cs)) {  
    sc.Open();  
    // do some work  
} // end of using()
```

Строка соединения

- Основные параметры:
 - Data Source – сетевое имя сервера
 - Encrypt – шифрованное соединение
 - Initial Catalog – имя БД на сервере
 - Integrated Security – способ аутентификации
 - True – текущий пользователь Windows
 - False – SQL Server-аутентификация
 - UserID, Password
- Класс SqlConnectionStringBuilder
 - Программное построение строки

SqlCommand

- Команда для работы с БД
- Основные свойства
 - `CommandText` – SQL-код, составляющий тело команды
 - `CommandType` – команда, таблица, хранимая процедура
 - `Parameters` – параметры команды

SqlCommand - методы

- Выполнение команды на SQL сервере
 - ExecuteReader() – создает курсор на чтение из БД (SqlDataReader)
 - ExecuteScalar() – позволяет получить скалярное значение
 - ExecuteNonQuery() – команды, не требующие чтения данных
- Prepare() – оптимизация команды (занимает время)
- Асинхронные версии команд

SqlDataReader

- Последовательное чтение результатов запроса
- `bool Read()` – следующая строка
- `object this[int i]`, `object this[string name]` – получение значения поля
- `int GetOrdinal(string name)` – номер столбца по имени
- `GetXXX(int i)` – получение типизированного значения столбца (осторожно с null). XXX – название простого типа
 - `int reader.GetInt32(int order)`
- `Close()`, `Dispose()` – закрытие DataReader

Пример

```
SqlConnection connection = new SqlConnection("Data Source=(local);Initial Catalog=Northwind;Integrated Security=True");
connection.Open();
SqlCommand command = new SqlCommand(@"SELECT OrderID, OrderDate, Freight, ShipAddress FROM Orders,
connection);
    List<Order> orderList = new List<Order>();
    using (SqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            Order order = new Order();
            order.Id = reader.GetInt32(0);
            order.OrderDate = (reader["OrderDate"] is DBNull) ? null : (DateTime?)reader["OrderDate"];
            order.Freight = (reader["Freight"] is DBNull) ? null : (double?)Convert.ToDouble(reader["Freight"]);
            order.ShipAddress = Convert.ToString(reader["ShipAddress"]);
            orderList.Add(order);
        }
    }
connection.Close();
```

Получение данных из БД

Демонстрация

Параметры

- Задаёт параметр команды
- `SqlCommand.Parameters` – коллекция параметров команды
 - `SqlCommand.Parameters[name]` – получение параметра по имени
- Имя параметра в команде начинается с @ (в MS SQL Server):
 - `SELECT name, author FROM books WHERE author=@author`

SqlParameter

- Представляет именованный параметр
- Конструкторы
 - `SqlParameter(string, object)` – со значением
 - `SqlParameter(string, SqlDbType)` – с заданным типом
- Свойства
 - `object SqlParameter.Value` – значение параметра
- Пример:
 - `SqlParameter author = new SqlParameter("@author", "Pushkin");`
 - `SqlCommand.Parameters.Add(author);`

Использование параметров

Демонстрация

Параметры vs. Текст

- Динамическое формирование:
 - `cmd.CommandText = "SELECT * FROM books" + "WHERE author=\\" + author + "\\";`
- Недостатки
 - Только для простых типов параметров – не работает с `image`, `text`, `binary`
 - Меньше возможностей оптимизации
 - Подверженность атакам свнедрением SQL-кода (SQL Injection)

SQL Injection

Демонстрация

Транзакции

- Обеспечиваю поддержку транзакций. ACID
 - Атомарность
 - Если в операцию вовлечены несколько порций данных, то фиксируются либо все, либо ни одна
 - Согласованность (Целостность)
 - Данные после завершения транзакции находятся в согласованном состоянии
 - Изоляция
 - Транзакция оперирует изолированными данными. Пока транзакция не завершена частичные данные никому не видны
 - Длительность
 - После завершения транзакции данные сохраняются. Даже в случае возникновения сбоев, сохраненные данные восстановятся

Транзакции

- Пространство имен System.Transactions
- Создание транзакции
 - using (TransactionScope scope = new TransactionScope()) { ...}
- Если внутри скопа произошло исключение, транзакция будет откатана
- Для коммита транзакции необходимо вызвать метод Complete()
- При обращении к нескольким серверам транзакция автоматически будет изменена на распределенную транзакцию и передана на управления в координатор распределенных транзакций (DTC)

```
using (TransactionScope scope = new TransactionScope())
{
    SqlConnection connection1 = new SqlConnection(connectionString1);
    ....
    SqlConnection connection2 = new SqlConnection(connectionString2);
    ...
    scope.Complete();
}
```

Транзакция

Демонстрация

Строка подключения в конфигурационном файле

- XML файл настроек приложения
- Позволяет менять настройки без перекомпиляции приложения
- В Visual Studio – файл App.config
- При компиляции переименовывается в файл виде ИмяПрилржения.exe.config

- Пример:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="sqlProvider" connectionString="Data Source=192.168.1.1;Initial
Catalog=Northwind;Integrated Security=True"/>
  </connectionStrings>
</configuration>
```

- Доступ к строке соединения из кода (нужен Nuget package System.Configuration.ConfigurationManager):
ConnectionStringSettings connectionString =
ConfigurationManager.ConnectionStrings["sqlProvider"];
SqlConnection connection = new SqlConnection(connectionString.ConnectionString);

Строка подключения

Демонстрация

Отсоединенная модель доступа к данным



Отсоединенный уровень

- Моделирует в памяти реляционные данные из БД
- Работа с данными, как будто есть постоянное подключение к БД
- Этапы работы:
 - Соединение с БД, получение данных
 - Работа с данными: редактирование, удаление, добавление без необходимости подключения к БД
 - Возможность отмены изменений
 - Эффективное применение изменений данных в БД (открытие соединения, внесение только изменений в данных и закрытие соединения)
- Вся логика работы с БД вынесена в отдельный тип – адаптер (DbDataAdapter)



- Основные классы могут использоваться и без БД
 - таблицы, столбцы, строки, отношения между таблицами

Основные типы

- **DataSet** – представляет реляционные данные в памяти
- **DataTable** – представляет таблицу с данными
- **DataColumn** – представляет схему таблицы
- **DataRow** – представляет строку в таблице. Содержит данные.
- **DataRelation** – отношение таблиц
- **DbDataAdapter** – адаптер данных. Поставляет данные из/в БД

DataSet

- Представляет реляционные данные в памяти
- Свойство **Tables** содержит коллекцию таблиц DataTable
- Свойство **Relations** содержит коллекцию отношений родительский/дочерний DataRelations между своими таблицами.
- Можно использовать навигацию между таблицами по отношениям.
- Свойство ExtendedProperties – представляет доп. информацию в виде пар имя/значение

DataSet. Свойства

- **DataSetName** – имя экземпляра DataSet
- **CaseSensitive** – чувствительность к регистру при сравнении (false)
- **EnforceConstraints** – применяются ли правила ограничений (true)
- **HasErrors** – имеются ли ошибки в любой строке любой таблицы в этом DataSet
- **RemotingFormat** – как DataSet должен сериализоваться (XML)

DataSet. Методы

- **RejectChanges()** – отменяет все изменения в DataSet (сделанные после загрузки или после вызова **AcceptChanges()**)
- **AcceptChanges()** – применяет все изменения в DataSet
- **HasChanges()** – имеются ли изменения в DataSet
- **GetChanges()** – возвращает копию DataSet, содержащую только изменения
- **Clear()** – очищает DataSet от данных (структура сохраняется)
- **Clone()** / **Copy()** – копируют структуру / структуру и данные в новый DataSet
- **ReadXml()** / **WriteXml()** – чтение / запись DataSet в поток (только данные или структуру и данные)

DataTable

- Представляет таблицу с данными
- Обладает многими свойствами и методами аналогичными DataSet
- Свойство **Columns** – содержит коллекцию колонок DataColumn
- Свойство **Rows** – содержит коллекцию строк DataRow

DataTable

- **TableName** – имя таблицы
- **ChildRelations / ParentRelations** – дочерние / родительские отношения для таблицы
- **Constraints** – коллекция ограничений в таблице
- **PrimaryKey** – массив столбцов – первичный ключ
- **DataSet** – ссылка на DataSet, которому принадлежит данная таблица (если есть)
- **CaseSensitive, Copy, RemotingFormat** и др. – аналогично DataSet

DataColumn

- Представляет один столбец в DataTable
- Множество всех DataColumn в таблице представляют схему таблицы
- Определяет тип данных в колонке
- Может содержать набор ограничений:
 - Первичный ключ
 - Значение по умолчанию
 - Уникальность
 - Допустимость DBNull
 - Разрешение только на чтение и др.

DataColumn

- ColumnName – имя колонки
- Caption – отображаемый заголовок колонки
- DataType – тип данных в данном столбце
- AutoIncrement, AutoIncrementSeed, AutoIncrementStep – задание автоинкремента для колонки
- AllowDBNull – задает, могут ли содержаться пустые значения
- DefaultValue – задает значение по умолчанию
- Unique – указывает, что данные в столбец должен быть уникальным
- ReadOnly – указывает, что данные только для чтения
- Table – получает DataTable, содержащий данную колонку
- Ordinal – числовое положение колонки в коллекции колонок Columns

DataRow

- Представляет конкретные **данные** в таблице
- Нельзя создать без таблицы. Нет конструктора

Создание с помощью

```
dataRow = dataTable.NewRow();
```

- Доступ к данным, содержащимся в строке с помощью индекса по номеру или имени колонки
- Имеет состояния

Демонстрация

Простой DataSet

DataRow. Свойства

- **RejectChanges()** – отменяет все изменения в строке (сделанные после загрузки или после вызова **AcceptChanges()**)
- **AcceptChanges()** – применяет все изменения в строке
- **Table** – таблица, содержащая строку
- **BeginEdit()**, **EndEdit()**, **CancelEdit()** – начало, окончание и отмена редактирования
- **Delete()** – помечает строку для удаления
- **HasErrors**, **GetColumnsInError**, **GetColumnError**, **ClearErrors**, **RowError** – позволяют определить ошибки в данных
- **RowState** – состояние текущей строки

RowState

- Added - добавлена
 - Deleted - удалена
 - Detached – не присоединена ни к одной таблице
 - Modified - изменена
 - Unchanged – неизменна
-
- По этим состояниям определяются изменения в таблицах и DataSet

Демонстрация

Состояния строки

Адаптеры данных

- Классы `DbDataAdapter`
 - `SqlDataAdapter`, `OracleDataAdapter` и т.д.
- Содержит команды:
 - `SelectCommand`
 - `InsertCommand`
 - `UpdateCommand`
 - `DeleteCommand`
- Метод `Fill()` – загружает данные в `DataSet` или таблицу
- Метод `Update` – обновляет данные в БД
- `SqlCommandBuilder` – помогает строить остальные команды по `SelectCommand`

Демонстрации

Редактирование таблицы БД

Типизированный DataSet

- Конкретное приложение работает с конкретной БД
 - Известны типы столбцов
 - Простой DataSet не типизирован
- Типизированный DataSet
 - Автоматическая генерация по БД
 - Типизация всех таблиц, полей и т.д.
 - Удобные методы поиска
 - Контроль типов на этапе компиляции

Типизированный DataSet

- Состоит из:
 - XML-схемы (описывает формат)
 - С#-кода (используется в коде)
 - Дополнительных адаптеров

Демонстрации

Типизированный DataSet