Връзки към източници на данни с ADO.NET

Еволюция на приложенията

- Трислойни приложения
  - Различните типове функционалност са в различни слоеве
  - Предимства
    - Отделяне на функционалността между потребителски интерфейс, бизнес правила и съхранение / достъп до данните
  - Недостатъци
    - По-трудна поддръжка
    - Повече усилия за осигуряване на сигурността
  - Пример
    - ASP.NET Web-приложение ↔ ASP.NET Web услуга ↔ MS SQL Server

Трислойно приложение

Presentation tier

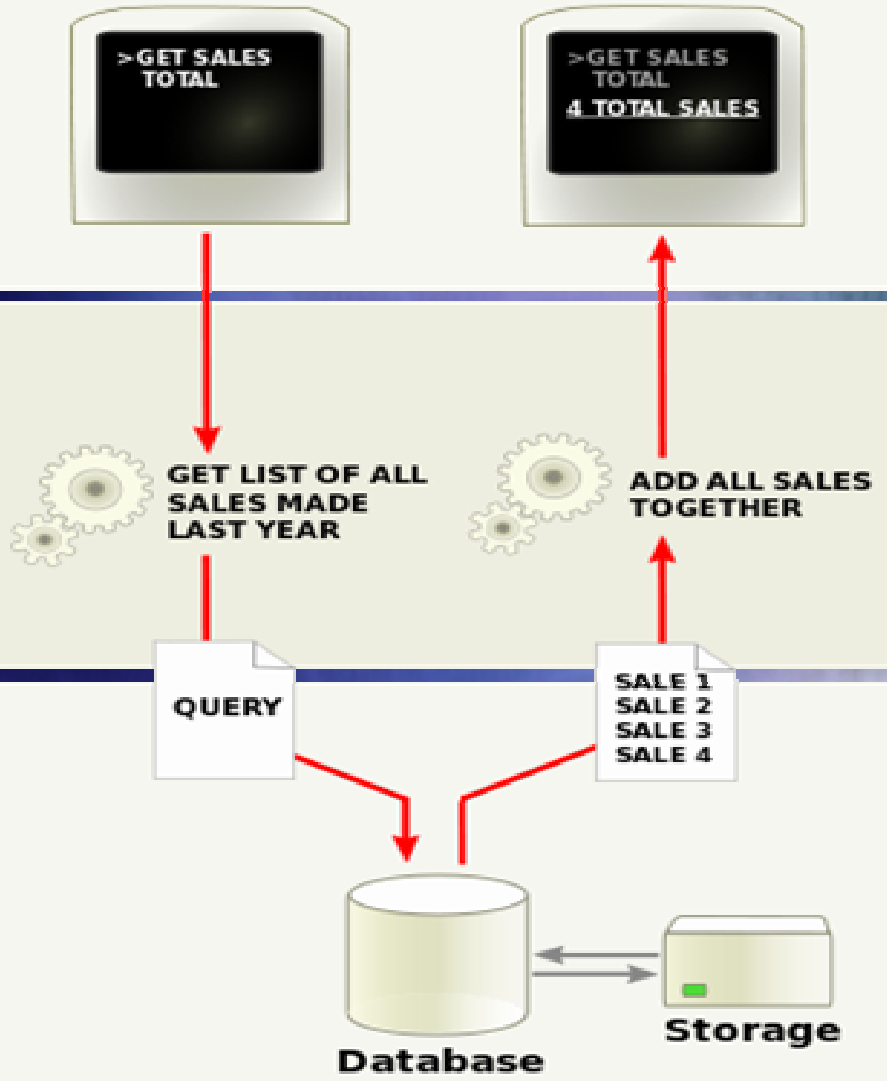
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



Шаблон Repository

- В много приложения бизнес логиката достъпва данните пряко от хранилища на данни (бази данни, списъци, уеб услуги и т.н.). Прекият достъп до данните може да доведе до следното:
  - Дублициран код (duplicated code);
  - Повишена възможност за програмни грешки;
  - Затруднения при централизирането на политиките, свързани с обработката на данни (например кеширането);
  - Слаба типизираност (weak typing) на бизнес данните;
  - Невъзможност за лесно тестване на бизнес логиката отделно от външни зависимости

Прилагане на Repository

- Чрез използването на Repository може да се постигнат една или повече от следните цели:
  - Да се увеличи количеството програмен код, който може да се тества автоматизирано.
  - Да се достъпва източника на данни от много места и да се приложат централизирани правила за достъп;
  - Да се имплементира и централизира стратегия за кеширане на данните от източника;

Прилагане на Repository

- Да се подобрят читаемостта и “ремонтопригодността” на кода чрез пълно разделяне на бизнес логиката от логиката за работа с данни или услуги;
- Да се използват обособени части от бизнес логиката (business entities), които са силно типизирани и така да се установят някои проблеми още при компилиране, а не чак при стартиране;
- Може да се асоциира определен механизъм на поведение към свързаните данни (например да се определят бизнес правила между отделните елементи на данните в едно entity)

Microsoft .net

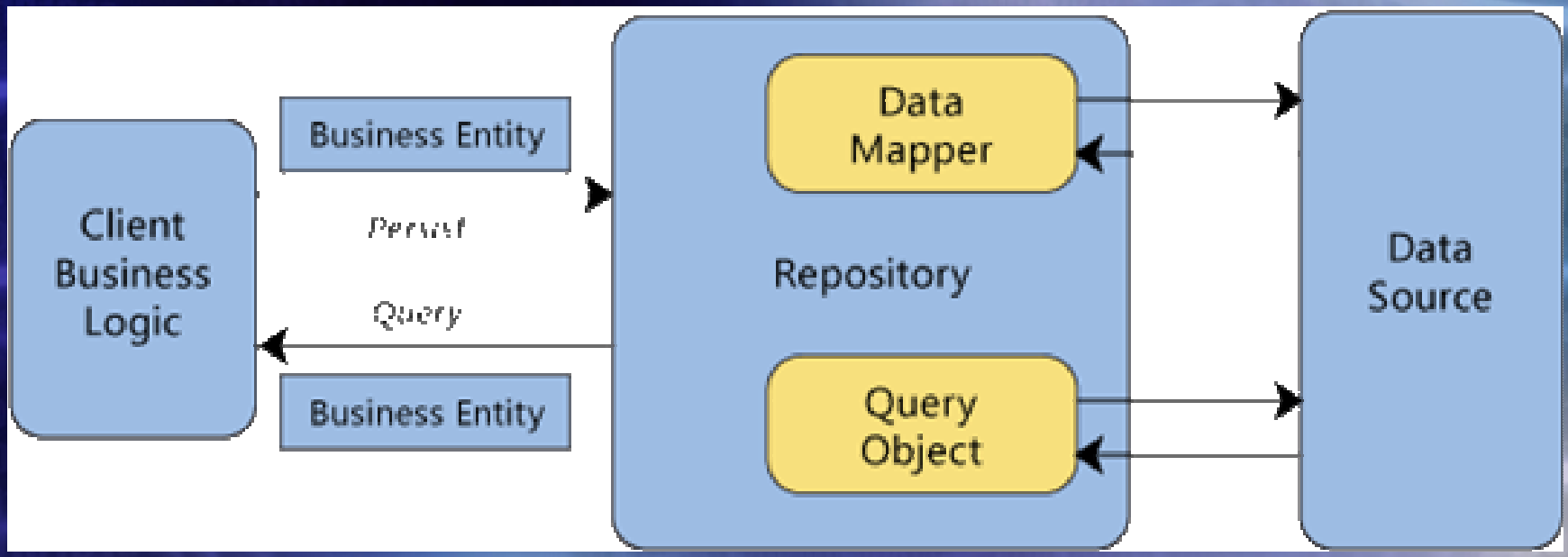
Прилагане на Repository

- Repository се използва, за да отдели логиката, която взаимодейства с данните и ги пренася към entity модела от бизнес логиката, която действа върху модела;
- Шаблонът Repository посредничи между слоя за работа с данни и бизнес слоя/слоевете на приложението. Той изпраща заявки за данни, пренася получените данни към съответното бизнес entity и препраща промените в това entity обратно към източника на данни. На практика, този шаблон отделя бизнес логиката от взаимодействието с данните или уеб услугата.

Цели на Repository

- Разделението между слоя за работа с данни и слоя на бизнес логиката има следните три цели:
  - Централизиране на логиката за работа с данни или уеб услуги.
  - Осигуряване на точка на заменяне за компонентните тестове (unit tests).
  - Осигуряване на гъвкава архитектура, която може да се адаптира като цялостен дизайн в развитието на приложението.

Действие на Repository



Microsoft .net

Data Providers в ADO.NET

- Data Providers са съвкупности от класове, които осигуряват връзка с различни бази от данни
  - За различните RDBMS системи се използват различни Data Providers
    - Различните производители използват различни протоколи за връзка със сървърите за данни
  - Дефинират се от 4 основни обекта:
    - `Connection` – за връзка с базата
    - `Command` – за изпълнение на SQL
    - `DataReader` – за извличане на данни
    - `DataAdapter` – за връзка с `DataSet`

Data Providers в ADO.NET

- В ADO.NET има няколко стандартни Data Providers
  - `SqlClient` – за връзка със SQL Server
  - `OleDB` – за връзка със стандарта OleDB
  - `Odbc` – за връзка със стандарта ODBC
  - `Oracle` – за връзка с Oracle
- Трети доставчици предлагат Data Providers за връзки с други RDBMS:
  - IBM DB2
  - MySQL
  - PostgreSQL
  - Borland Interbase / Firebird

OLE DB Data Provider

- `OleDbConnection` – осъществява връзка с OLE DB източник на данни

```
OleDbConnection dbConn = new OleDbConnection(
    @"Provider=Microsoft.Jet.OLEDB.4.0;Data
    Source=C:\MyDB.mdb;Persist Security Info=False");
```

- `OleDbCommand` – изпълнява SQL команди върху OLE DB връзка към база данни
- `OleDbParameter` – параметър на команда
- `OleDbDataReader` – за извличане на данни от команда, изпълнена през OLE DB
- `OleDbDataAdapter` – обменя данни между `DataSet` обекти и OLE DB база данни

Клас OleDbConnection

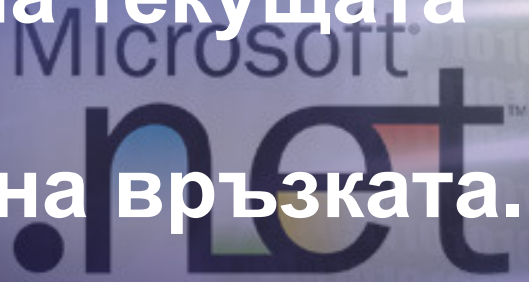
- **Представя единична отворена връзка към източник на данни.**
- С клиент/сървър база данни, това е еквивалентно на мрежова връзка към сървъра;
- В зависмост от функционалността, поддържана от OLE DB доставчика, някои методи или свойства на OleDbConnection обекта може да бъдат недостъпни.

Клас OleDbConnection

- Когато се създава инстанция на OleDbConnection, всички свойства се инициализират с техните начални стойности (списък от тези стойности може да се види в конструктора на класа).
- Може да се отвори повече от един DataReader към една инстанция на OleDbConnection.
- Ако OleDbConnection обект излезе от обхват, връзката не се затваря автоматично. Затова тя трябва да се затвори експлицитно чрез извикване на методи **Close()** или **Dispose()**, или чрез използване на обекта в **Using** състояние.

По-важни методи и свойства на OleDbConnection

- OleDbConnection() – създава нова инстанция на класа;
- OleDbConnection(ConnectionString) - създава нова инстанция на класа чрез специален стринг за връзка;
- ConnectionString – инициализира стринг от параметри за връзка с базата;
- DataBase – връща името на текущата база данни;
- State – текущо състояние на връзката.

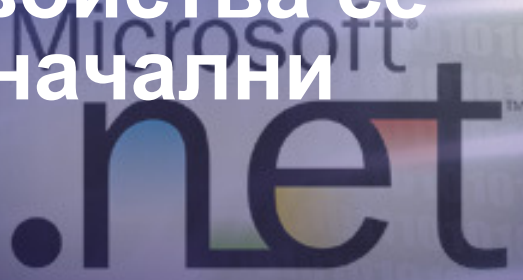


По-важни методи и свойства на OleDbConnection

- Open() – отваря връзка към БД, чрез параметри, зададени с ConnectionString;
- Close() – затваря връзката към БД;
- CreateCommand() – връща OleDbCommand обект, свързан с OleDbConnection обекта;
- ChangeDatabase(String) – променя текущата БД за OleDbConnection обект с отворена връзка.

Клас OleDbCommand

- **Представя SQL команда или съхранена процедура, за изпълнение от източник на данни;**
- **Наследява абстрактния клас DbCommand;**
- **Имплементира интерфейсите** ICloneable, IDbCommand, IDisposable
- **Когато се създава инстанция на OleDbCommand, всички свойства се инициализират с техните начални стойности**



По-важни методи и свойства на OleDbCommand

- OleDbCommand() – инициализира нова инстанция на класа;
- OleDbCommand(String) – инициализира нова инстанция на класа с текст на заявката;
- OleDbCommand(String, OleDbConnection) – инициализира нова инстанция на класа с текст на заявката и връзка към БД;
- OleDbCommand(String, OleDbConnection, Transaction) – инициализира нова инстанция на класа с текст на заявката, връзка към БД и транзакция, в която командата ще се изпълнява.

По-важни методи и свойства на OleDbCommand

- CommandText – SQL заявка или съхранена процедура за изпълнение от източника на данни;
- CommandTimeout – време за изчакване преди да се прекратят опитите за изпълнение;
- Connection – връзката, използвана от тази инстанция на OleDbCommand;
- Transaction – OleDbTransaction обект, в който командата се изпълнява.

По-важни методи и свойства на OleDbCommand

- Clone() – създава копие на текущата инстанция;
- CreateParameter() – нова инстанция на OleDbParameter;
- ExecuteNonQuery() – изпълнява SQL заявка към БД и връща броя на получените записи;
- ExecuteReader() – изпраща CommandText към БД и създава OleDbDataReader обект;
- Cancel() – опитва да прекрати изпълнението на командата.

Клас OleDbParameter

- **Представя параметър на OleDbCommand обект;**
- **Наследява абстрактния клас DbParameter;**
- **Имплементира интерфейсите** ICloneable, IDbDataParameter, IDataParameter;
- **Класът OleDbParameter не може да се наследява.**

Конструктори на OleDbParameter

- OleDbParameter() – създава нова инстанция на класа;
- OleDbParameter(String, Object) – създава нова инстанция на класа чрез име на параметъра и стойност;
- OleDbParameter(String, OleDbType) – създава нова инстанция чрез име на параметъра и тип на данните;
- OleDbParameter(String, OleDbType, Int32) – създава нова инстанция чрез име на параметъра, тип на данните и дължина.

По-важни свойства на OleDbParameter

- DbType – тип на параметъра;
- Direction – стойност, която определя дали параметърът е само за вход, само за изход, двупосочен или върнат резултат от съхранена процедура;
- ParameterName – име на параметъра;
- Size – дължина на параметъра;
- Value – стойност на параметъра.

Клас OleDbDataReader

- Осигурява начин за четене на еднопосочен поток от записи от източника на данни;
- Наследява класа DbDataReader;
- Не имплементира интерфейси;
- Класът OleDbDataReader не може да се наследява;
- Инстанция се създава чрез обект от клас OleDbCommand.

По-важни методи на OleDbDataReader

- GetData(Int32) – връща OleDbDataReader обект за заявеният номер на колона;
- GetEnumerator() – връща итератор, който може да обхожда редовете на четеца;
- Read() – чете следващ запис;
- NextResult() – прочита следващия резултат при четене на резултатите от SQL заявка;
- Close() – затваря четеца.

Примерно използване

```
public static void ReadData(string connString, string
queryString)
{
  using (OleDbConnection conn = new OleDbConnection(connString))
  {
    OleDbCommand command = new OleDbCommand(queryString, conn);
    conn.Open();
    OleDbDataReader reader = command.ExecuteReader();
    while (reader.Read()) {
      Console.WriteLine(reader[0].ToString());
    }
    reader.Close();
  }
}
```

Клас OleDbTransaction

- Представя SQL транзакция към източника на данни;
- Наследява класа DbTransaction;
- Не имплементира интерфейси;
- Класът OleDbTransaction не може да се наследява;
- Инстанция се създава чрез обект от клас OleDbConnection (метод BeginTransaction()).

По-важни свойства и методи на OleDbTransaction

- Connection – OleDbConnection обект, свързан с транзакцията;
- IsolationLevel – определя как ще се ‘заключва’ транзакцията по време на връзката;
- Commit() – прехвърля данни към базата;
- Rollback() – връща от състояние на изчакване (нов опит за прехвърляне на данните)

Връзки с ADO.NET

Въпроси?