Universidade do Minho
Departamento de Informática

DSS 08/09

Camada de Dados - JDBC

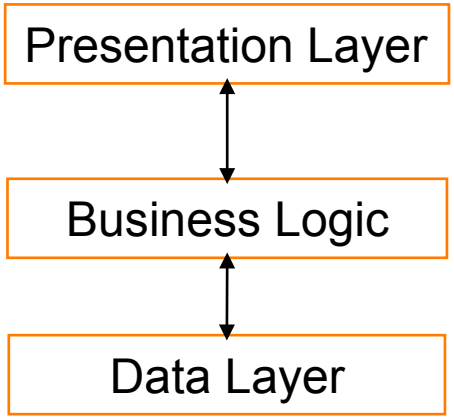
Aula 1

António Nestor Ribeiro /António Ramires Fernandes/
José Creissac Campos

{anr,arf,jfc}@di.uminho.pt

Camada de Dados

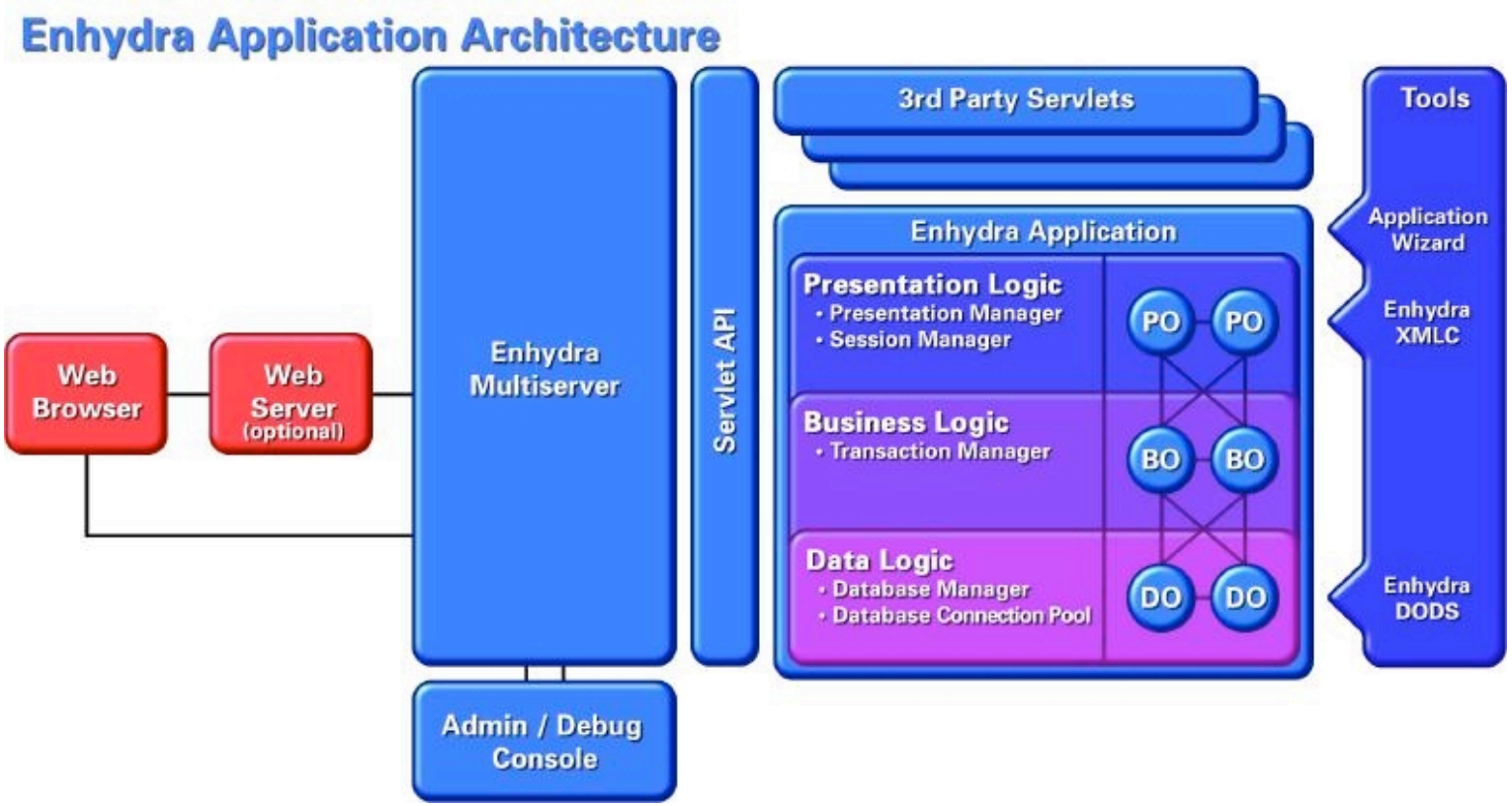
- A camada de dados, *data layer*, permite isolar o acesso aos dados, por forma a que o resto da aplicação não esteja dependente da origem ou estrutura sob a qual os dados estão armazenados.



- A combinação que aqui será explorada consiste na utilização de bases de dados acedidas na camada de dados através de JDBC.

Aplicações Multi-Camada

- Diversos tipos de objectos para efectuarem diferentes operações
  - Presentation Objects
  - Business Objects
  - Data Objects



JDBC

- JDBC = **J**ava **D**ata**B**ase **C**onnectivity
- Pacotes: java.sql

```
import java.sql.*;
```

- Passos:

1. inicializar o driver
2. estabelecer uma ligação
3. ...
4. fechar a ligação

JDBC - Driver

- Passo I: carregar a classe do driver

```
static Class forName(String nome);
```

  - excepção: `ClassNotFoundException`
  - Driver ODBC: `sun.jdbc.odbc.JdbcOdbcDriver`

```
try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch (ClassNotFoundException e) {

        // Driver não disponível
}
```

Drivers JDBC

- Existem várias implementações de drivers JDBC para os diversos motores de BD.
  - **Type 1**: drivers que implementam a API JDBC como um mapeamento para uma outra API. A portabilidade está dependente da existência do driver destino. A ligação a ODBC é um exemplo destes drivers;
  - **Type 2**: drivers que são escritos parcialmente em Java e numa outra linguagem. É utilizado um cliente específico para o acesso à base de dados pretendida;
  - **Type 3**: drivers escritos em Java e que comunicam com um servidor de middleware que faz o interface com as fontes de dados;
  - **Type 4**: drivers em que o cliente liga directamente à base de dados utilizando exclusivamente Java.

JDBC - Ligação

- Passo II: Estabelecer uma ligação
- Classe **DriverManager** disponibiliza os seguintes métodos de classe:

```
Connection getConnection( String url);
Connection getConnection( String url, String login, String pass);
Connection getConnection( String url, java.util.Properties.info);
```

  - excepção: `SQLException`

- **Connection** é uma interface que define um conjunto de métodos para operar com a base de dados.

JDBC - Ligação

- Passo II: Estabelecer uma ligação (2)
- Parâmetro url => protocolo:subprotocolo:identificador
  - protocolo: é constante e representado pela string "jdbc"
  - subprotocolo: é função do motor da base de dados e da forma de acesso (directa ou indirecta, por exemplo através de ODBC). No caso de acesso através de ODBC o subprotocolo é "odbc".
  - identificador da base de dados: Indica qual a base de dados a utilizar. No caso da ligação ODBC-JDBC é o identificador definido em *Data Source Name*.

JDBC - Ligação

- Passo II: Estabelecer uma ligação (3)

```
Connection con;

...

try {
        con = DriverManager.getConnection("jdbc:odbc:bolsa");
}
catch (SQLException e) {

        // Erro ao estabelecer a ligação
}
```

JDBC - Libertar Recursos

- Passo IV: terminar a ligação
- Interface **Connection**

```
void close();
boolean isClosed();
```

  - excepção: `SQLException`

```
try {
        con.close();
}
catch (SQLException e) {

        // Erro ao terminar a ligação
}
```

JDBC - Comandos SQL

- Tipos de comandos SQL
  - DDL: Data Definition Language

```
CREATE TABLE idFunc (
      cod VARCHAR(10) NOT NULL,
      nome VARCHAR(50),
      primary key(cod))
```

  - Selecção

```
SELECT * FROM cotações
```

  - Actualização

```
UPDATE clientes
          SET numerário = 100000
          WHERE nome = "João"
```

JDBC - Comandos SQL

- Primeiro passo: criar um *statement*
- Interface **Connection**

```
Statement createStatement();
```

  - excepção: `SQLException`

- Interface **Statement**

```
ResultSet executeQuery(String sql);
int executeUpdate(String sql);
```

  - excepção: `SQLException`

JDBC - SELECT

- Exemplo para comandos de selecção

```
Connection con;
... // iniciar a ligação

Statement st;
ResultSet res;
String sql;

sql = "SELECT nome FROM clientes WHERE numerário > 100000";

try {

        st = con.createStatement();
        res = st.executeQuery(sql);

} catch (SQLException e) {
        // lidar com as excepções
}
```

JDBC - Actualização e DDL

- `int executeUpdate(String sql);`
  - Se for um comando de actualização (UPDATE, INSERT ou DELETE)
    - devolve o número de registos afectados
  - Comandos DDL (ex: CREATE TABLE)
    - devolve 0

```
Statement st;

try {

        st = con.createStatement();
        int count = st.executeUpdate("UPDATE...");

} catch (SQLException e) {
        // lidar com as excepções
}
```

JDBC - Result Set

- Acesso aos resultados armazenados num ResultSet
- Interface **ResultSet**
  - funciona como iterador sobre os registos devolvidos
  - `boolean next();`
  - dentro de um registo fornece um conjunto de métodos para aceder aos campos, por exemplo:
    - `getString(int índiceDoCampo);`
    - `getString(String nomeDoCampo);`
  - Um ResultSet está disponível até ser fechado ou até o statement ser reutilizado ou fechado

JDBC - Result Set

- exemplo de acesso a um ResultSet

```
Statement st; ResultSet rs;

try {

        rs = st.executeQuery("SELECT saldo FROM contas");
        int total = 0;
        while (rs.next())
                total += rs.getInt ("saldo");

        // nunca fazer isto em casa!!!
        System.out.println("Soma :" + total");

} catch (SQLException e) {
        // lidar com as excepções
}
```

JDBC - Exercício

- Crie uma base de dados em Access com uma tabela "clientes" com os seguintes campos: cod (chave), saldo e nome. Preencha a tabela com alguns valores;
- Registe a base de dados no ODBC;
- Escreva um programa em java para inserir um registo na tabela e em seguida calcular o total dos saldos (iterando o ResultSet).

Apêndice - Dummy's Guide to JDBC-ODBC e Access

- Para registar uma base de dados no ODBC realizar os seguintes passos:

1. Abrir "Data Sources (ODBC)". ControlPanel -> Administrative Tools;
2. Seleccionar "Add" e escolher "Driver do Microsoft Access";
3. Preencher "Data Source Name" (este vai ser o nome pela qual a base de dados é conhecida numa aplicação java com JDBC-ODBC;
4. Pressionar "Select..." para indicar o nome do ficheiro da base de dados;
5. "OK"!