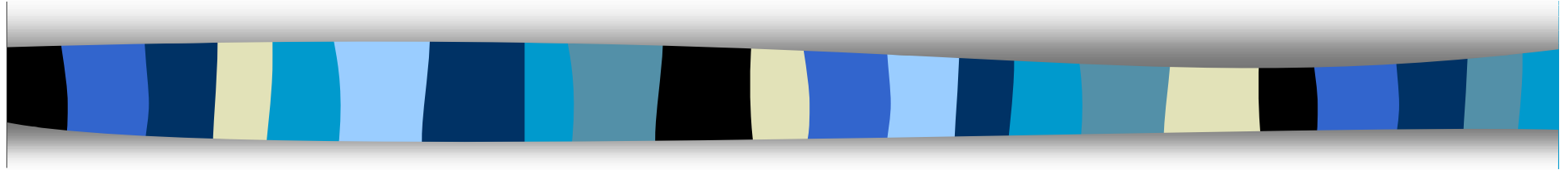


JDBC - How to...



Philippe Roose (LIUPPA/ IUT Bayonne)

<http://www.iutbayonne.univ-pau.fr/~roose/>

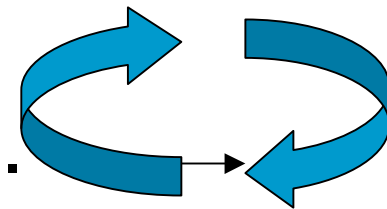


JDBC : What is it ?

- API allowing to connect to most of DBMS (SQL) of the market
- Specific drivers according to the DBMS
- Generic drivers with ODBC
- Distant/local access plus interconnections of DBMS.

The five steps

- 1/ Check and load the corresponding driver
- 2/ Connection with eventually (preferable !) authentication.
- 3/ Creation of a request
- 4/ Getting results if needed.
- 5/ Close the connection





Step 1

- Check if drivers are into the PATH
 - modify the CLASSPATH if needed to point on :
 - jdbc/lib/classes12.zip
 - with Oracle, for example :
 - /opt/oracle/jdbc/lib/classes12.zip



Step 1

```
try {  
    Class.forName("oracle.jdbc.driver.OracleDriver");  
}  
catch (ClassNotFoundException e) {  
    System.out.println ("Error loading driver" +  
        e.getMessage());  
}
```



Step 1

■ Driver loading

```
try{ // driver loading
    DriverManager.registerDriver(new
    oracle.jdbc.driver.OracleDriver());
}
catch (SQLException e) {
    System.out.println (e.getMessage());
}
```



Step 2

- We need to connect to the DB, to select the base and eventually to authenticate
- Same *try as drivers load*.



Step 2

```
String name= "login";
```

```
String passwd = "mp";
```

```
String url = "jdbc:oracle:thin:@localhost:1521:";
```

```
String base = "base";
```

```
Connection conn; // no new, c is an abstract class
```

```
conn =
```

```
    DriverManager.getConnection(url+base,name,passwd);
```



Step 3

- We can now create the request
- 2 possibilities
 - Directly as parameter,
 - Using a **String** containing the request.



Step 3

```
// Creation of the request
```

```
Statement req = conn.createStatement();
```

```
ResultSet res = req.executeQuery  
                ("SELECT id FROM table");
```

- The `executeQuery` method is available for selection operations. When we need to modify the structure/content, we will use the method **`executeUpdate`**
- Instance of class `ResultSet` is a needed object to contain results of requests.



Step 4

- The result of a request is often a set of tuples

```
// Gettings results
```

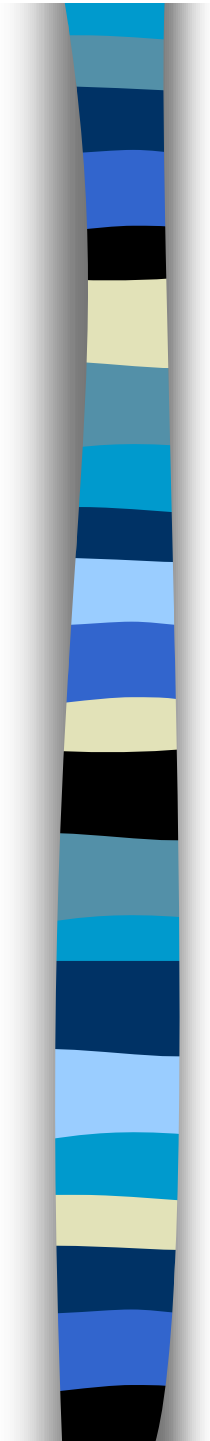
```
while (res.next()) // next -> next tuple/row
```

```
    System.out.println("-->" + res.getString(« ID »));
```



Step 4

- When we want to get a field of a row (tuple), we will use methods : **getXXX**
 - getString(« Name of field » ou number of field)
 - getFloat(« Name of field » ou number of field)
 - getInt(« Name of field » ou number of field)
 - ... cf. java.sun.com/docs/books/tutorial/jdbc/basics/
 - getXXX does not function with the following types : **binary, varbinary, longvarbinary, date, time, timestamp.**



	T	S	M	I	B	F	D	N	C	V	L	V	V	L	D	T	T
	I	I	I	I	I	O	O	O	B	A	A	A	A	A	A	A	A
	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
	Y	L	E	G	R	L	F	D	O	C	V	B	B	B	D	T	T
	N	E	N	E	A	A	A	A	A	A	A	A	A	A	A	A	A
	T	T	R	T	L	T	T	T	T	T	R	R	R	R	T	M	P
getBytes	x	x	x	x	x	x	x	x	x	x	x						
getDate										x	x	x				X	x
getTime										x	x	x				X	x
getTimestamp										x	x	x			x	x	X
getAsciiStream										x	x	X	x	x	x		
getUnicodeStream										x	x	X	x	x	x		
getBinaryStream												x	x	X			
getObject	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
getByte	X	x	x	x	x	X	x	X	x	X	x	x					
getShort	x	X	x	x	x	x	x	x	x	x	x	x	x				
getInt	x	x	X	x	x	x	x	x	x	x	x	x	x				
getLong	x	x	x	X	x	x	x	x	x	x	x	x	x				
getFloat	x	x	x	x	X	x	x	x	x	x	x	x	x				
getDouble	x	x	x	x	x	X	X	x	x	x	x	x	x				
getBigDecimal	x	x	x	x	x	x	x	X	X	x	x	x	x				
getBoolean	x	x	x	x	x	x	x	x	x	X	x	x	x				
getString	x	x	x	x	x	x	x	x	x	x	X	X	x	x	x	x	x



Step 5

- As with file use, we have to close the id request and the access to the DB.

```
req.close();  
conn.close();
```