

Web 3

Lesson 2: JDBC

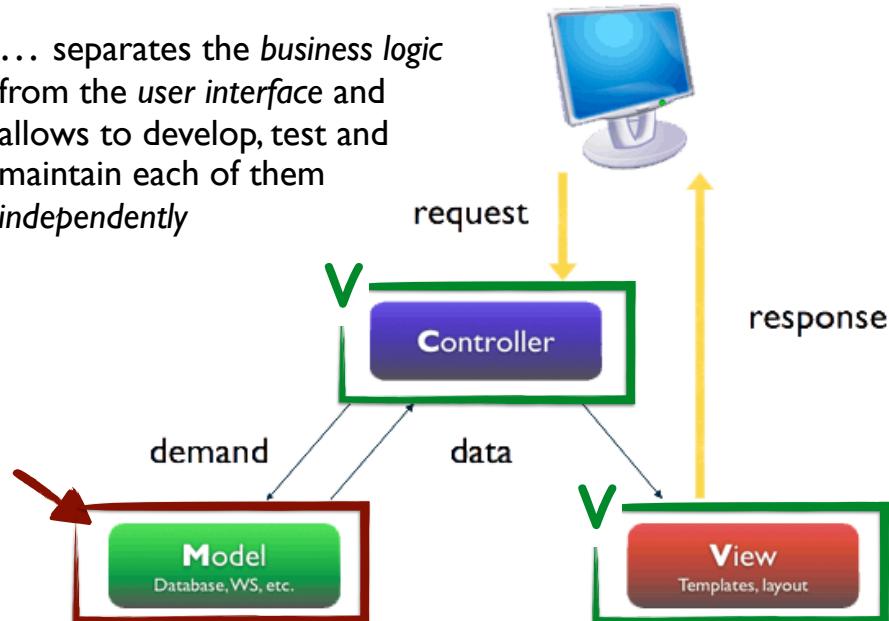
AGENDA

- JDBC
- Exception handling



MVC

... separates the *business logic* from the *user interface* and allows to develop, test and maintain each of them *independently*



WARNING



- a lot of technical issues
- important for the test

EXAM QUESTIONS...



- What does JDBC stand for?
- What is the advantage of JDBC?
- Given some code from a database class: explain what these lines mean
- ...

KEEP DATA IN LIST

```
public class CountryService {  
    private List<Country> countries = new ArrayList<Country>();  
  
    public void addCountry(Country country){  
        countries.add(country);  
    }  
  
    public List<Country> getCountryes(){  
        return new ArrayList<Country>(countries);  
    }  
  
    ...  
}
```



- order of addition matters
- we want doubles
- we want to use an identifier to lookup elements

BETTER: KEEP DATA IN MAP

```
public class CountryService {  
    private Map<String, Country> countries =  
        new HashMap<String, Country>();  
  
    public void addCountry(Country country){  
        countries.put(country.getName(), country);  
    }  
  
    public List<Country> getCountries(){  
        return new ArrayList<Country>(countries.values());  
    }  
    ...  
      
      
    choose carefully!  


- order of addition does not matter
- we want no doubles
- we want to use an identifier to lookup elements

```

Identifier: do not choose a modifiable field as id. For instance: if you choose email as unique identifier (primary key) you should not be able to modify the value afterwards. Sometimes it is safer to choose a technical field without any special meaning (increasing number, ...)

EVEN BETTER:
KEEP DATA IN DATABASE



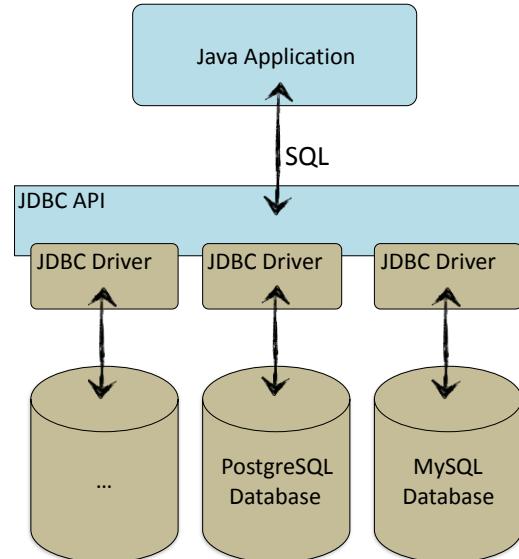
JDBC

- Java DataBase Connectivity
- Java API → Set of classes you can use
- Allows Java program to communicate
 - with **database**
 - via **SQL**

WHY?

- "Write Once, Run Anywhere"
- Communication with...
 - relational databases: *PostgreSQL, Java DB, MySQL, ...*
 - spreadsheets
 - ...

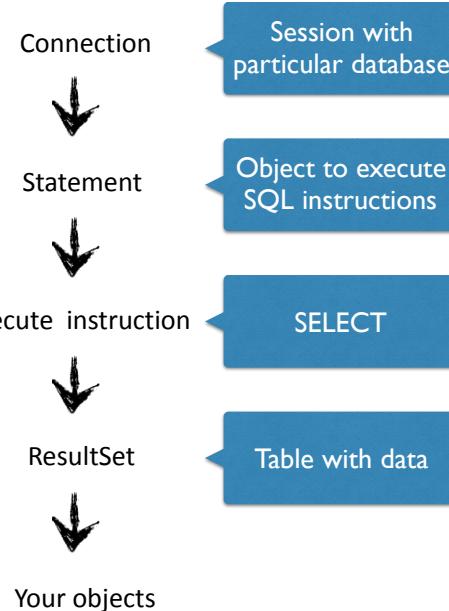
OVERVIEW



ACTION - QUERY INSTRUCTIONS

Action	Query
INSERT, UPDATE, DELETE, ...	SELECT
statement.execute("query") returns: nothing	statement.executeQuery("query") returns: part of the data (ResultSet)
statement.executeUpdate("query") returns: number of affected rows	

How - QUERY



HOW - QUERY

Connection



```
Connection connection = DriverManager.getConnection(  
    "url",  
    properties );
```

Statement



```
Statement statement = connection.createStatement();
```

Execute instruction



```
ResultSet result =  
    statement.executeQuery( "SELECT * FROM MyTable" );
```

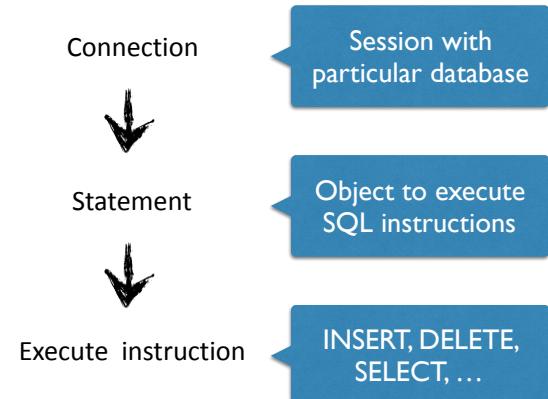
ResultSet



Your objects

```
String name = result.getString( "Name" );  
MyObject object = new MyObject( name );
```

How - ACTION



How - ACTION

Connection



```
Connection connection = DriverManager.getConnection(  
    "url",  
    properties );
```

Statement



```
Statement statement = connection.createStatement();
```

Execute instruction

```
statement.executeUpdate("DELETE FROM MyTable WHERE  
name='Greetje'");
```

POSTGRESQL DRIVER

- <http://jdbc.postgresql.org/download.html>
 - JDBC41 Postgresql Driver, Version 42.1.4
 - for Java 1.8
- Eclipse: add jar in folder WEB-INF/lib

CONNECTION DATA

- URL

- driver: <vendor>://<server>:<port>/<dbname>
- `jdbc:postgresql://gegevensbanken.khleuven.be:51718/2TX?currentSchema=<name of your schema>`

- Properties

- user: <LDAP userid>
- password: <password>
- ssl: `true`
- sslfactory:
`org.postgresql.ssl.NonValidatingFactory`

```

public class CountryDbDemo {
    public static void main(String[] args) throws SQLException {
        Properties properties = new Properties();
        String url = "jdbc:postgresql://gegevensbanken.khleuven.be:51314/webontwerp?";
        properties.setProperty("currentSchema", <name of your schema>);
        properties.setProperty("user", <userid>);
        properties.setProperty("password", <password>);
        properties.setProperty("ssl", "true");
        properties.setProperty("sslfactory", "org.postgresql.ssl.NonValidatingFactory");
        Class.forName("org.postgresql.Driver");
        Connection connection = DriverManager.getConnection(url, properties);
        Statement statement = connection.createStatement();
        ResultSet result = statement.executeQuery( "SELECT * FROM country" );
        while(result.next()){
            String name = result.getString("name");
            String capital = result.getString("capital");
            int numberOfVotes = Integer.parseInt(result.getString("votes"));
            int numberOfInhabitants = Integer.parseInt(result.getString("inhabitants"));

            Country country= new Country(name, numberOfInhabitants,capital, numberOfVotes);
            System.out.println(country);
        }
        statement.close();
        connection.close();
    }
}

```

JDBC API



for web-application

EXAMPLE

Class.forName("org.postgresql.Driver") is actually not needed in a main method, it is only needed in a web application

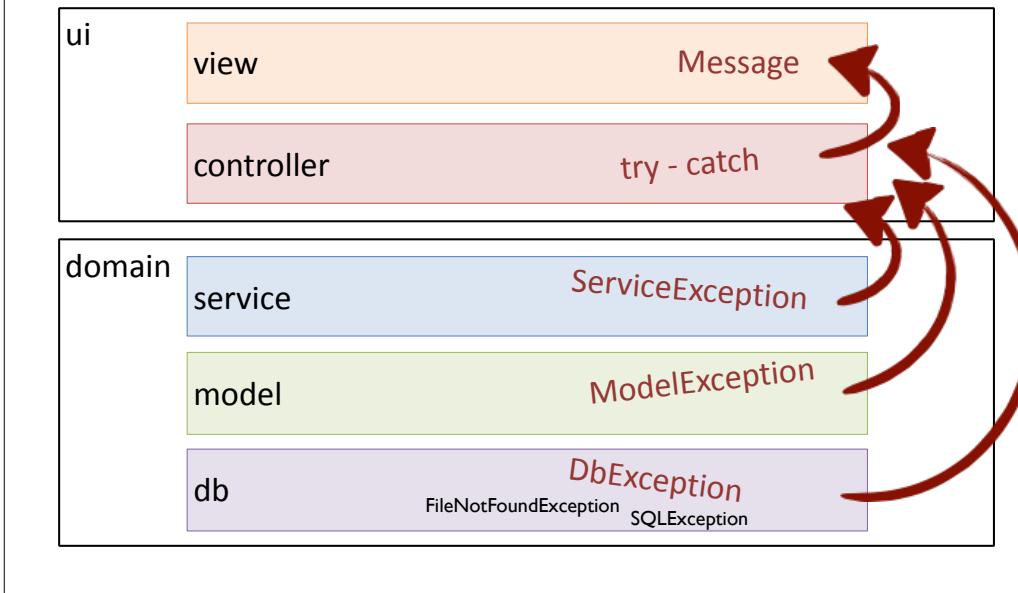
AGENDA

JDBC

Exception handling



EXCEPTION HANDLING



CUSTOM EXCEPTIONS

Tells me where it went wrong

```
package db
```

Unchecked

```
public class DbException extends RuntimeException {  
  
    public DbException() {  
        super();  
    }  
  
    public DbException(String message, Throwable exception) {  
        super(message, exception);  
    }  
  
    public DbException(String message) {  
        super(message);  
    }  
  
    public DbException(Throwable exception) {  
        super(exception);  
    }  
}
```

you can pass the original exception

EXAMPLE

```
try {  
    Connection connection = DriverManager.getConnection(url,properties);  
    Statement statement = connection.createStatement();  
    ResultSet result = statement.executeQuery( "SELECT * FROM country" );  
  
    statement.execute(sql);  
} catch (SQLException e) {  
    throw new DbException(e.getMessage, e);  
}
```



pass the original exception

AGENDA

☒ JDBC

☒ Exception handling



WARNING

- For the test we need access to your table, etc...
- Grant access to u0082726, u0015529 and u0034562
 - to your **schema**
 - to each **table**
 - to each **sequence**
- Or we cannot give you any points !

REFERENCES

- JDBC:
 - <https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html>
 - <http://docs.oracle.com/javase/tutorial/jdbc/index.html>
 - <http://www.oracle.com/technetwork/java/overview-141217.html>

REFERENCES

- PostgreSQL:
 - <http://jdbc.postgresql.org>
 - <http://jdbc.postgresql.org/documentation/head/index.html>