Informatics II for Engineering Sciences (MSE)

Chapter IV – Databases

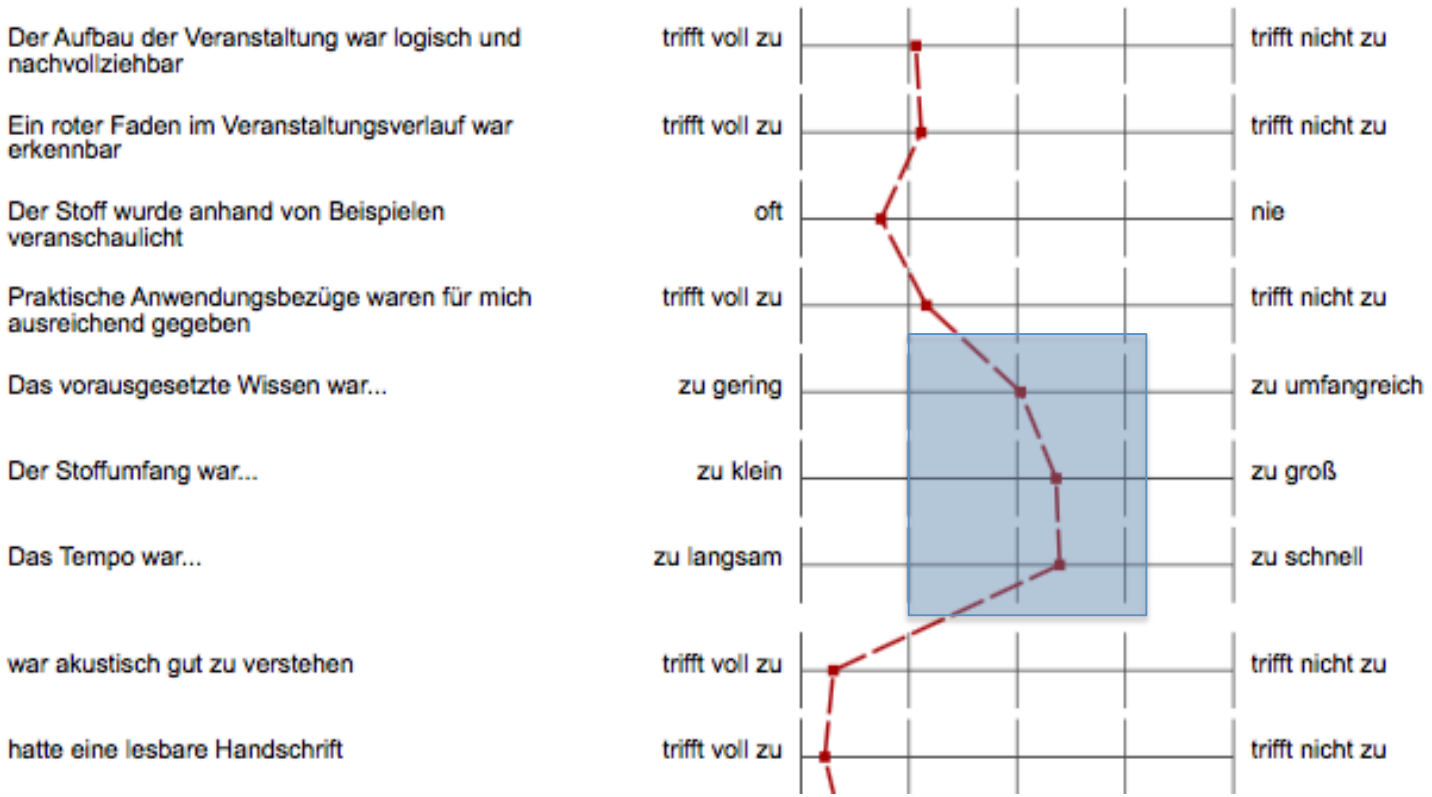
Bernd Brügge Ph.D. , Jan Knobloch, Emitzá Guzmán

Your Feedback

Categories – open text Feedback	Votes	Proposal
practical use	2	***we are going to test it today***
to many slides	4	***we are going to test it today***
more programmming / examples	9	***we are going to test it today***
no solutions for homeworks	3	to be discussed with collegues
speaker is to fast / lecture to fast	1	stop me - when im too fast
to much expected knowledge	2	...
repetition of last lecture	1	***we are going to test it today***
no separate tutorial	3	What do you think?

Who would prefer a separate tutorial over a mixed lecture with exercises ?

Your Feedback



Last week - Content

Relational Databases:

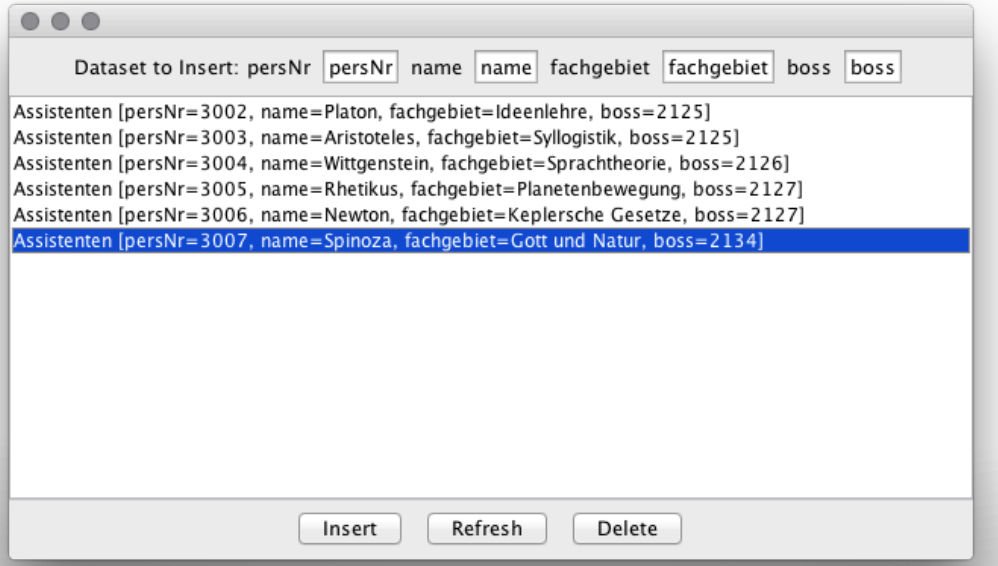
How to use select statements to retrieve certain datasets from an relational database.

Iteration using ResultSet.next() to iterate over existing Results.

Some calls to frameworks / drivers in Java force you to use the “try-catch” paradigm to catch possible occurring exceptions.

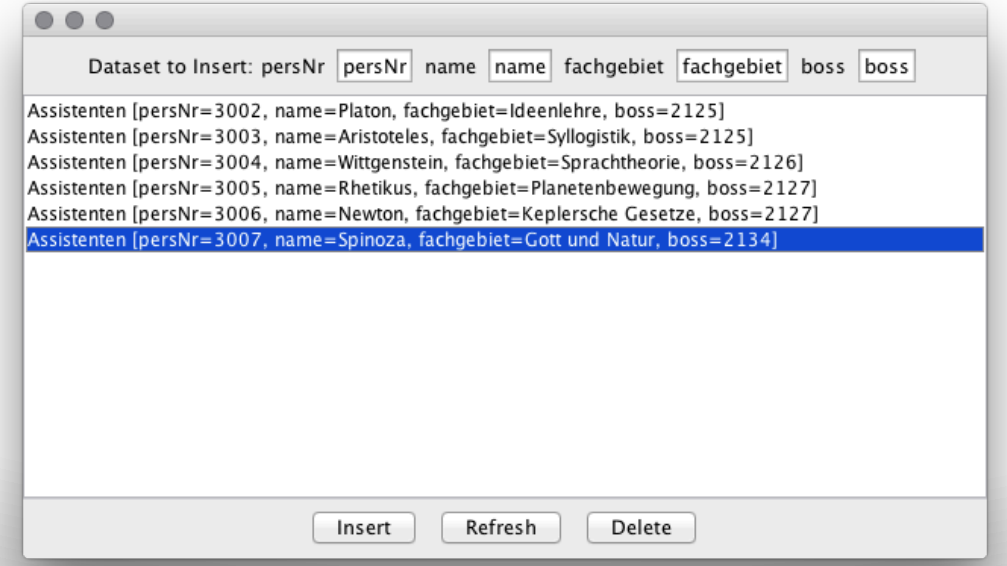
Our goal for today

- Using a given template project to connect a database to a graphical frontend.
  - Using the JDBC MySQL Driver
  - Mapping Objects to Relational Data (Assistenten)
  - Use an Object-Relational Mapper (Hibernate)



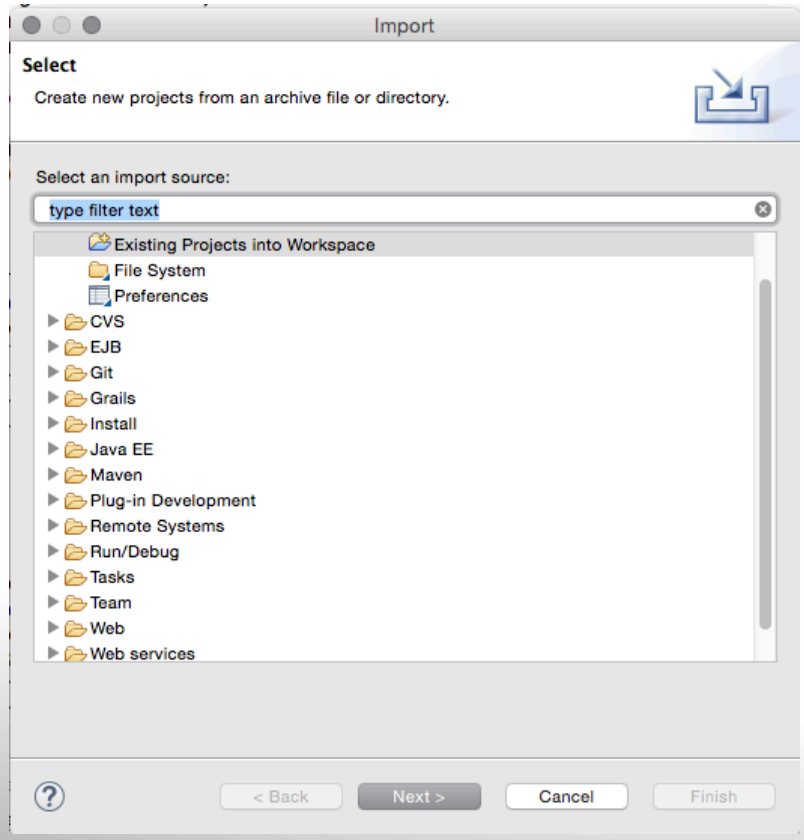
Our goal for today

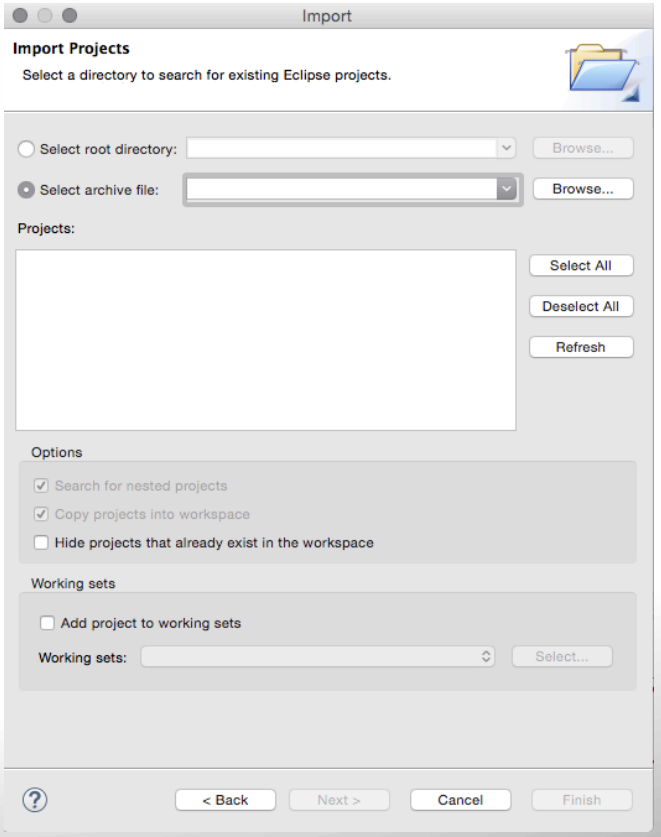
- Optional:
  - Take a closer look into the Model View Controller Paradigm implemented in the sample Project
  - Using a NoSQL Database instead of a MySQL database to implement the same functionality



Getting Started #1

- Import the given sample Project into Eclipse.
  - File – Import – Existing Projects into Workspace – Select Archive File - Finish





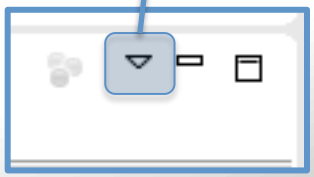
Getting Started #2

- Setup the Task bar to see what you have to accomplish
  - Window – Show View – Tasks
  - Down arrow to filter for Tasks dependent to this project

Problems | Javadoc | Declaration | Console | Tasks

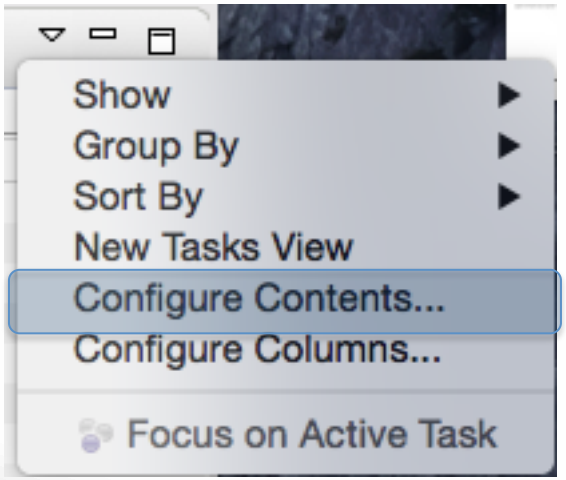
19 items

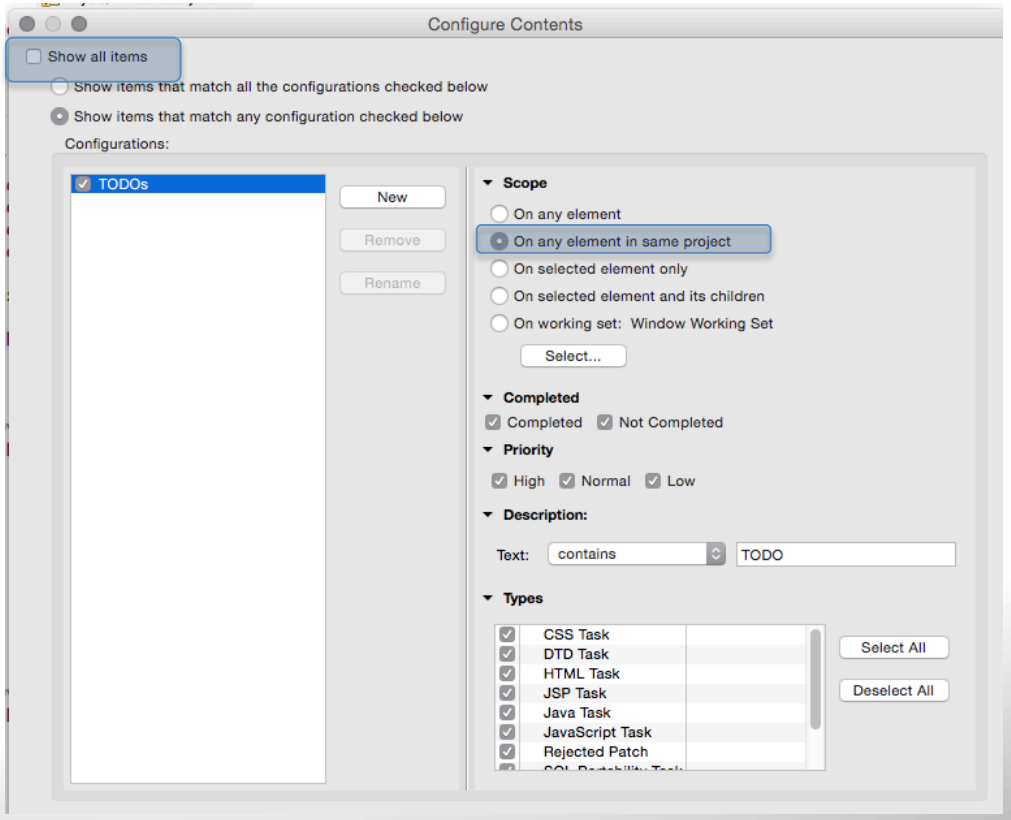
✓	!	Description	Resource	Path	Location	Type
		TODO: OPTIONAL NoSQL: deleteDataset - use assistenten collection and BasicDBObject	NoSQL.java	/MSE_DB_MVC...	line 55	Java Task
		TODO: OPTIONAL NoSQL: getAllDatasets - use assistenten collection	NoSQL.java	/MSE_DB_MVC...	line 64	Java Task
		TODO: OPTIONAL NoSQL: insertDataset - use assistenten collection and BasicDBObject	NoSQL.java	/MSE_DB_MVC...	line 48	Java Task
		TODO: OPTIONAL NoSQLMorphia: deleteDataset	NoSQLMorph...	/MSE_DB_MVC...	line 38	Java Task
		TODO: OPTIONAL NoSQLMorphia: getAllDatasets	NoSQLMorph...	/MSE_DB_MVC...	line 46	Java Task
		TODO: OPTIONAL NoSQLMorphia: insertDataset	NoSQLMorph...	/MSE_DB_MVC...	line 30	Java Task
		TODO: TASK1: Implement the method getAllDatasets() - use connection and statement	MySQL.java	/MSE_DB_MVC...	line 34	Java Task
		TODO: TASK10: Remove the MVC comment below and start your application.	Main.java	/MSE_DB_MVC...	line 77	Java Task
		TODO: TASK11: Implement the method insertDataset()	StudentContr...	/MSE_DB_MVC...	line 20	Java Task
		TODO: TASK12: Implement the method refreshDataset()	StudentContr...	/MSE_DB_MVC...	line 35	Java Task
		TODO: TASK13: Implement the method deleteDataset()	StudentContr...	/MSE_DB_MVC...	line 27	Java Task
		TODO: TASK2: Implement the method insertDataset() - use connection and statement	MySQL.java	/MSE_DB_MVC...	line 42	Java Task
		TODO: TASK3: Implement the method deleteDataset() - use connection and statement	MySQL.java	/MSE_DB_MVC...	line 49	Java Task
		TODO: TASK4: Test your implementation in his main class!	Main.java	/MSE_DB_MVC...	line 54	Java Task
		TODO: TASK5: Annotate the Assistent Class with the minimum Annotations @Entity for the Class and @Id for your p...	Assistent.java	/MSE_DB_MVC...	line 12	Java Task
		TODO: TASK6: Implement the method getAllDatasets() - use session	MySQLHiber...	/MSE_DB_MVC...	line 58	Java Task
		TODO: TASK7: Implement the method insertDataset() - use session	MySQLHiber...	/MSE_DB_MVC...	line 34	Java Task
		TODO: TASK8: Implement the method deleteDataset() - use session	MySQLHiber...	/MSE_DB_MVC...	line 46	Java Task
		TODO: TASK9: Test your implementation in his main class!	Main.java	/MSE_DB_MVC...	line 55	Java Task



Getting Started #2

- Filter tasks to just select TODOs from our actual project
  - Select Configure Contents…
  - Deselect “Show all items” and select “on any element in the same project”





Exercise Tasks #1-13

- Navigate to your Main-Method:
  - Path: “src/students/Main.java”

```
* 1.) Setup your MySQL Connection the "manual" way.
* 1.1) TASK1: Implement the method getAllDatasets()
* 1.2) TASK2: Implement the method insertDataset()
* 1.3) TASK3: Implement the method deleteDataset()
* 1.4) TASK4: Test your implementation in his main class!
*
* 2.) Setup your MySQLHibernate Connection the "easy" way.
* 2.1) TASK5: Annotate the Assistent Class with the minimum Annotations @Entity and @Id for your persNr Id field
* 2.2) TASK6: Implement the method getAllDatasets()
* 2.3) TASK7: Implement the method insertDataset()
* 2.4) TASK8: Implement the method deleteDataset()
* 2.5) TASK9: Test your implementation in his main class!
*
*
* 3.) TASK10: Remove the MVC comment below and start your application.
* 3.1) After pressing the refresh, insert, and delete buttons you will see outputs
* that certain functionalities haven't been implemented yet.
* 3.2) TASK11: Implement the method insertDataset() in your StudentController - use MySQLHibernate or MySQL
* 3.3) TASK12: Implement the method refreshDataset() in your StudentController - use MySQLHibernate or MySQL
* 3.4) TASK13: Implement the method deleteDataset() in your StudentController  -use MySQLHibernate or MySQL
*/
```

Exercise Tasks #1-13

- You can click on your Tasks to jump directly to the classes/functions you need to edit.

Problems | Javadoc | Declaration | Console | Tasks

19 items

Description	Resource	Path	Location	Type
TODO: OPTIONAL NoSQL: deleteDataset - use assistenten collection and BasicDBObject	NoSQL.java	/MSE_DB_MVC...	line 55	Java Task
TODO: OPTIONAL NoSQL: getAllDatasets - use assistenten collection	NoSQL.java	/MSE_DB_MVC...	line 64	Java Task
TODO: OPTIONAL NoSQL: insertDataset - use assistenten collection and BasicDBObject	NoSQL.java	/MSE_DB_MVC...	line 48	Java Task
TODO: OPTIONAL NoSQLMorphia: deleteDataset	NoSQLMorph...	/MSE_DB_MVC...	line 38	Java Task
TODO: OPTIONAL NoSQLMorphia: getAllDatasets	NoSQLMorph...	/MSE_DB_MVC...	line 46	Java Task
TODO: OPTIONAL NoSQLMorphia: insertDataset	NoSQLMorph...	/MSE_DB_MVC...	line 30	Java Task
TODO: TASK1: Implement the method getAllDatasets() - use connection and statement	MySQL.java	/MSE_DB_MVC...	line 34	Java Task
TODO: TASK10: Remove the MVC comment below and start your application.	Main.java	/MSE_DB_MVC...	line 77	Java Task
TODO: TASK11: Implement the method insertDataset()	StudentContr...	/MSE_DB_MVC...	line 20	Java Task

```java
@Override
public List<Assistent> getAllDatasets() {
    System.out.println("MySQL: getAllDatasets");
    //TODO: TASK1: Implement the method getAllDatasets() - use connection and statement
    System.out.println("MySQL: Not implemented yet!");
    return null;
}
```

Exercise Task #1

- TASK1: Implement the method getAllDatasets()

- Hints:
  - Use the connection variable
  - Create a statement to retrieve a Resultset of Assistenten.
  - Parse the ResultSet by using resultset.next()
  - Parse the Contents of the Resultset by using getInt(), getString() from the ResultSet

  - An “Assistent” table has the following columns:
  - PersNr (1), Name (2), Fachgebiet (3), Boss (4)

Exercise Task #1 - Solution

- TASK1: Implement the method getAllDatasets()

```
@Override
public List<Assistent> getAllDatasets() {
    System.out.println("MySQL: getAllDatasets");
    // TODO: TASK1: Implement the method getAllDatasets() - use connection
    // and statement
    try {
        Statement s = connection.createStatement();
        s.execute("SELECT * from Assistenten");
        ResultSet rs = s.getResultSet();
        LinkedList<Assistent> list = new LinkedList<Assistent>();
        while (rs.next()) {
            Assistent a = new Assistent();
            a.setPersNr(rs.getInt(1));
            a.setName(rs.getString(2));
            a.setFachgebiet(rs.getString(3));
            a.setBoss(rs.getInt(4));
            list.add(a);
        }
        return list;
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return null;
}
```

Exercise Task #2

- TASK2: Implement the method insertDataset()
- Hints:
  - Use the connection variable
  - Create a statement to insert a new dataset to Assistenten
  - Execute the statement. (SQL INSERT syntax)

Exercise Task #2 - Solution

- TASK2: Implement the method insertDataset()

```
@Override
public void insertDataset(Assistent a) {
    System.out.println("MySQL: insertDataset");
    // TODO: TASK2: Implement the method insertDataset() - use connection
    // and statement
    Statement s;
    try {
        s = connection.createStatement();
        s.execute("INSERT INTO Assistenten(PersNr, Name, Fachgebiet, Boss) VALUES ('"
                + a.getPersNr() + "' , '"
                + a.getName() + "' , '"
                + a.getFachgebiet() + "' , '"
                + a.getBoss() + "');");
    } catch (SQLException e) {
        e.printStackTrace();
    }

}
```

Exercise Task #3

- TASK3: Implement the method deleteDataset()
- Hints:
  - Use the connection variable
  - Create a statement to delete a dataset in the “Assistenten” Table
  - Execute the statement. (SQL DELETE syntax)

Exercise Task #3 - Solution

- TASK3: Implement the method deleteDataset()

```
@Override
public void deleteDataset(Assistent a) {
    System.out.println("MySQL: deleteDataset");
    // TODO: TASK3: Implement the method deleteDataset() - use connection
    // and statement
    Statement s;
    try {
        s = connection.createStatement();
        s.execute("DELETE FROM Assistenten WHERE PersNr = '" + a.getPersNr() + "';");
    } catch (SQLException e) {

        e.printStackTrace();
    }

}
```

Exercise Task #4

- TASK4: Test your implementation in this main class!
- Hints:
  - Watch for SQL Constrains e.g. the Boss ID has to be existent and the PersNr is not allowed to be duplicated.

Exercise Task #4 - Solution

- TASK4: Test your implementation in this main class!

```
//TODO: TASK4: Test your implementation in this main class!
mysql.insertDataset(new Assistent(5050, "jan", "MSE", 2125));
mysql.deleteDataset(new Assistent(5050, "jan", "MSE", 2125));
List<Assistent> assis = mysql.getAllDatasets();
for (Assistent a : assis)
{
    System.out.println(a);
}
```

End of Exam relevant material

- The following slides are just for completeness and are not relevant for the exam.

Object Relational Mappings

- What did we do?
  - We mapped manually by iterating over different elements of our Entity and saving them into an actual Java object. These simple Java Objects are also called “POJOs” – Plain Old Java Objects

- What do we want to do?
  - We don’t want to care about mapping issues due to e.g. wrong column specifications etc.
  - We want a simple way to map Java objects to our database even including relations like one-to-one, many-to-one and many-to-many.

Object Relational Mappings

Different Frameworks can do this mapping:

Hibernate – for SQL

Morphia – for NoSQL

Frameworks can map objects to database entities by using:

- Java Annotations
- XML configurations

Object Relational Mappings

Minimal Mappings using Annotations:

On class Level – tell the framework which class belongs to which table **@Entity, @Table(name = “”)**

On property Level – tell the framework which Java properties correspond to which columns in the table – at minimum specify the unique ID of the table using **@Id**

Exercise Task #5

- TASK5: Annotate the Assistent Class with the minimum Annotations:
- Hints:
  - Annotate the class with **@Entity and @Table(name = "Assistenten")**
  - Annotate the Getter-function of persNr with **@Id**

Exercise Task #5 - Solution

- TASK5: Annotate the Assistent Class with the minimum Annotations @Entity and @Table(name = "Assistenten") for the Class and @Id for your persNr Id field

```
@Entity
@Table(name = "Assistenten")
public class Assistent implements Serializable{

    int persNr;
    String name;
    String fachgebiet;
    int boss;


    public Assistent() {
        super();
    }
    @Id
    public int getPersNr() {
        return persNr;
    }
```

Exercise Task #6

- TASK6: Implement the method getAllDatasets()
- Hints:
  - Use the session and create an criteria using Assistent.class
  - Use the list() function on the result of the criteria

Exercise Task #6 - Solution

- TASK6: Implement the method getAllDatasets()

```
@Override
public List<Assistent> getAllDatasets() {
    Session s = factory.openSession();
    s.beginTransaction();
    // TODO: TASK6: Implement the method getAllDatasets() - use session
    List<Assistent> list = s.createCriteria(Assistent.class).list();
    System.out.println("MySQLHibernate: insertDataset");
    s.getTransaction().commit();
    s.close();

    return list;
}
```

Exercise Task #7

- TASK7: Implement the method insertDataset()
- Hints:
  - Use the session and call persist()

Exercise Task #7 - Solution

- TASK7: Implement the method insertDataset()

```
@Override
public void insertDataset(Assistent a) {
    Session s = factory.openSession();
    s.beginTransaction();
    // TODO: TASK7: Implement the method insertDataset() - use session
    System.out.println("MySQLHibernate: insertDataset");
    s.persist(a);
    s.getTransaction().commit();
    s.close();
}
```

Exercise Task #8

- TASK8: Implement the method deleteDataset()
- Hints:
  - Use the session and call delete()

Exercise Task #8 - Solution

- TASK8: Implement the method deleteDataset()

```
@Override
public void deleteDataset(Assistent a) {
    Session s = factory.openSession();
    s.beginTransaction();
    // TODO: TASK8: Implement the method deleteDataset() - use session
    System.out.println("MySQLHibernate: deleteDataset");
    s.delete(a);
    s.getTransaction().commit();
    s.close();

}
```

Exercise Task #9

- TASK9: Test your implementation in this main class!

Exercise Task #9 - Solution

- TASK9: Test your implementation in his main class!

```
mysqlhibernate.insertDataset(new Assistent(5050, "jan", "MSE", 2125));
mysqlhibernate.deleteDataset(new Assistent(5050, "jan", "MSE", 2125));

mysqlhibernate.getAllDatasets();
List<Assistent> assis = mysqlhibernate.getAllDatasets();
for (Assistent a : assis)
{
    System.out.println(a);
}
```

Exercise Task #10

- TASK10: Remove the MVC comment below and start your application.

Exercise Task #10 - Solution

- TASK10: Remove the MVC comment below and start your application.

```
//TODO: TASK10: Remove the MVC comment below and start your application.
MVC mvc = new MVC(new StudentController());
```

Exercise Task #11

- TASK11: Implement the method insertDataset()
- Hints:
  - Call your MySQLHibernate class – insertDataset() method
  - Call addElement() on the provided graphical list

Exercise Task #11 - Solution

- TASK11: Implement the method insertDataset()

```
public void insertDataset(Assistent a, DefaultListModel<Assistent> listModel) {
    System.out.println("StudentController: insertDataset");
    //TODO: TASK11: Implement the method insertDataset()
    // Write Assistant to database;
    mysqlhibernate.insertDataset(a);
    // Load all assistants from database and refresh/update the list
    listModel.addElement(a);

}
```

Exercise Task #12

- TASK12: Implement the method refreshDataset()
- Hints:
  - Call your MySQLHibernate class – getAllDatasets() method
  - Remove all elements from the list and add the new elements.

Exercise Task #12 - Solution

- TASK12: Implement the method refreshDataset()

```
public void refreshDataset(DefaultListModel<Assistent> listModel) {
    System.out.println("StudentController: refreshDataset");
    //TODO: TASK12: Implement the method refreshDataset()
    listModel.removeAllElements();
    // Load all assistants from database and refresh/update the list
    List<Assistent> assis = mysqlhibernate.getAllDatasets();
    for (Assistent a : assis) {
        listModel.addElement(a);
    }
}
```

Exercise Task #13

- TASK13: Implement the method deleteDataset()
- Hints:
  - Call your MySQLHibernate class – deleteDataset() method
  - Remove the single element from the list.

Exercise Task #13 - Solution

- TASK13: Implement the method deleteDataset()

```
public void deleteDataset(Assistent a, DefaultListModel<Assistent> listModel) {
    System.out.println("StudentController: deleteDataset");
    //TODO: TASK13: Implement the method deleteDataset()
    // Delete Assistant from database;
    mysqlhibernate.deleteDataset(a);
    // Load all assistants from database and refresh/update the list
    listModel.removeElement(a);

}
```