

Object-Relation Mapping

Hogeschool Rotterdam
Rotterdam, Netherlands

Lecture topics

- Object-Relation Mapping.
- Setting up Postgres.
- Setting up hibernate.

JDBC

- Provides a set of Java API for accessing the relational databases from Java program.
- It allows querying/updating database data
- JDBC represents statements using one of the following classes:

JDBC

- Provides a set of Java API for accessing the relational databases from Java program.
- It allows querying/updating database data
- JDBC represents statements using one of the following classes:
 - ① Statement – the statement is sent to the database server each and every time.

JDBC

- Provides a set of Java API for accessing the relational databases from Java program.
- It allows querying/updating database data
- JDBC represents statements using one of the following classes:
 - 1 Statement – the statement is sent to the database server each and every time.
 - 2 PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

JDBC

- Provides a set of Java API for accessing the relational databases from Java program.
- It allows querying/updating database data
- JDBC represents statements using one of the following classes:
 - 1 Statement – the statement is sent to the database server each and every time.
 - 2 PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
 - 3 CallableStatement – used for executing stored procedures on the database.

```
..  
try (Connection conn = DriverManager.getConnection(  
    "jdbc:somejdbcvndor:other_data_needed  
    by_some_jdbc_vendor",  
    "myLogin",  
    "myPassword" ) ) {  
  
    Statement stmt = conn.createStatement()  
    stmt.executeUpdate( "INSERT INTO Ships(  
        name) VALUE('destroyer_XYZ')" );  
}  
  
..
```

Disadvantages of JDBC

- Less maintainable code for large projects
- Queries are DBMS specific

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.
- Hides details of SQL queries from OO logic.

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.
- Hides details of SQL queries from OO logic.
- Based on JDBC 'under the hood'.

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.
- Hides details of SQL queries from OO logic.
- Based on JDBC 'under the hood'.
- Maps Java POJOs (plain old java objects) to relational databases

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.
- Hides details of SQL queries from OO logic.
- Based on JDBC 'under the hood'.
- Maps Java POJOs (plain old java objects) to relational databases
- Portability: DB independent

ORM

- It's a technique for converting data between relational databases and object-oriented programming languages.
- Hides details of SQL queries from OO logic.
- Based on JDBC 'under the hood'.
- Maps Java POJOs (plain old java objects) to relational databases
- Portability: DB independent
- Performance: Object and query caching mechanism

ORM and Data Persistence Pattern

- There are some pattern designed for persistence of objects in DBS
- ORM framework are based on design patterns and framework specific implementations
- Only two data persistence pattern will be discussed in this lecture

ORM and Data Persistence Pattern

- There are some pattern designed for persistence of objects in DBS
- ORM framework are based on design patterns and framework specific implementations
- Only two data persistence pattern will be discussed in this lecture
 - Active record pattern
 - Data mapper

Active Record Pattern (ARP)

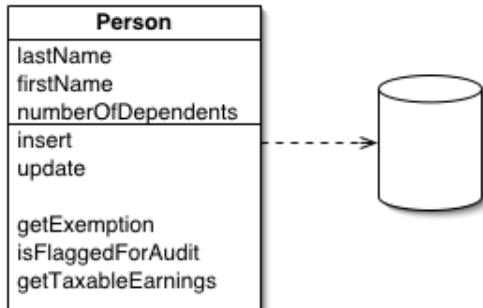
- In ARP there is an object for every table or view that wraps a row
- This object encapsulates the database access, and adds domain logic on that data

Active Record Pattern (ARP)

- In ARP there is an object for every table or view that wraps a row
- This object encapsulates the database access, and adds domain logic on that data
- So this object carries both data and behavior

Active Record Pattern

- sample structure of active record



Active Record Pattern (ARP)

- What are the limitations of this pattern regarding objects structure?

Active Record Pattern (ARP)

- What are the limitations of this pattern regarding objects structure?
- Think of collections and inheritance!

Data Mapper Pattern (DMP)

- In DMP there is a layer of mappers that moves data between objects and a database
- This layer keeps both in-memory objects and database independent from each others
- The reasons for this are:

Data Mapper Pattern (DMP)

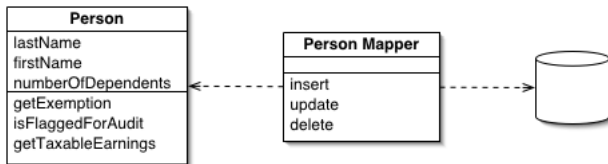
- In DMP there is a layer of mappers that moves data between objects and a database
- This layer keeps both in-memory objects and database independent from each others
- The reasons for this are:
 - Objects and relational databases have different mechanisms for structuring data

Data Mapper Pattern (DMP)

- In DMP there is a layer of mappers that moves data between objects and a database
- This layer keeps both in-memory objects and database independent from each others
- The reasons for this are:
 - Objects and relational databases have different mechanisms for structuring data
 - Many parts of an object, such as collections and inheritance, aren't present in relational databases
 - Object schema and database schema do not match in many cases

Data Mapper

- sample structure of data mapper



Data Mapper Framework: Hibernate

- Hibernate implements data mapper pattern
- Components of Hibernate :
 - Configuration file for the settings: hibernate.cfg.xml

Data Mapper Framework: Hibernate

- Hibernate implements data mapper pattern
- Components of Hibernate :
 - Configuration file for the settings: hibernate.cfg.xml
 - Mapping files for each entity: Entity.hbm.xml OR

Data Mapper Framework: Hibernate

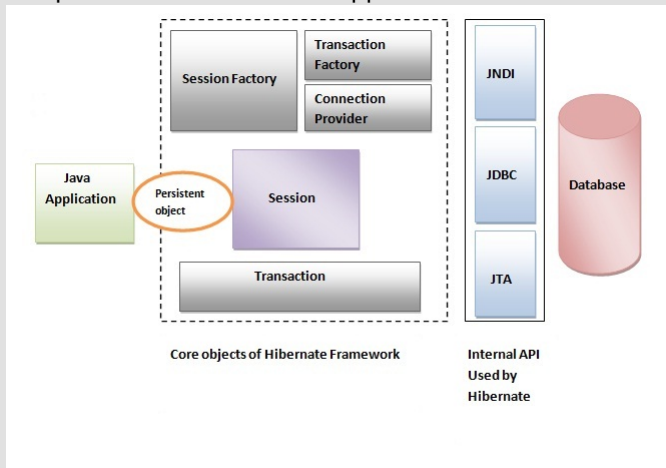
- Hibernate implements data mapper pattern
- Components of Hibernate :
 - Configuration file for the settings: hibernate.cfg.xml
 - Mapping files for each entity: Entity.hbm.xml OR
 - Annotation in the POJO-files (@keyword in java files)

Data Mapper Framework: Hibernate

- Hibernate implements data mapper pattern
- Components of Hibernate :
 - Configuration file for the settings: hibernate.cfg.xml
 - Mapping files for each entity: Entity.hbm.xml OR
 - Annotation in the POJO-files (@keyword in java files)
 - Hibernate Query Language (HQL) is an object-oriented query language

Hibernate

- sample structure of data mapper



Example of Hibernate configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-
configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">
      org.hibernate.dialect.PostgreSQLDialect
    </property>
    <property name="hibernate.connection.
      driver_class">
        com.postgres.jdbc.Driver
      </property>

    <!-- Assume test is the database name -->
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost/test
    </property>
```

POJO sample for Ships

```
public class Ships {  
    private int serial;  
    private String name;  
    private String armour;  
    ...  
    public Ships() {}  
    ...  
}
```


Example of a mapping file in Hibernate

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate_ Mapping_DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="Ships" table="Ships">
    <meta attribute="class-description">
      This class contains the employee detail.
    </meta>
    <id name="serial" type="int" column="serial">
      <generator class="native"/>
    </id>
    <property name="serial" column="serial" type="integer"/>
    <property name="name" column="name" type="string"/>
    <property name="armour" column="armour" type="string"/>
  </class>
</hibernate-mapping>
```

Example of using annotation instead of mapping files

```
import javax.persistence.*;
```

```
@Entity
```

```
@Table(name = "ships")
```

```
public class Ships {
```

```
    @Id @GeneratedValue
```

```
    @Column(name = "serial")
```

```
    private int serial;
```

```
    @Column(name = "name")
```

```
    private String name;
```

```
    @Column(name = "armour")
```

```
    private String armour;
```

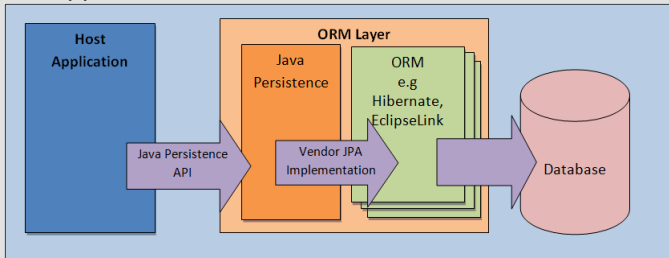
```
    public Employee() {}
```

Sample code for persisting a new ship in Hibernate

```
....  
Session session = factory.openSession();  
    Transaction tx = null;  
    Integer shipId = null;  
    try{  
        tx = session.beginTransaction();  
        Ships s = new Ships("walle","metal□armour"  
            );  
        shipId = (Integer) session.save(Ships);  
        tx.commit();  
    }catch (HibernateException e) {  
        if (tx!=null) tx.rollback();  
        e.printStackTrace();  
    }finally {  
        session.close();  
    }  
....
```

Java Persistence API

- Provides the standard specification for managing the relational data in applications
- JPA is a layer between third party ORM like Hibernate and the application



Java Persistence API

- Entities
- EntityManager
- Persistence Units
- JPA Query Language

Sample code for persisting a new ship in JPA with Hibernate

```
....  
EntityManagerFactory emf = Persistence.  
    createEntityManagerFactory("InfDev5PU");  
    //create a session in case of Hibernate  
EntityManager em = emf.createEntityManager();  
    //starts a transaction  
em.getTransaction().begin();  
Ships s = new Ships("walle", "metal_ armour");  
    //calls the save method of Hibernate  
em.persist(em);  
em.getTransaction().commit();  
em.close();  
....
```

Lab

- Check the tables mentioned in les 0
- Create these tables in Postgres
- Insert new data into the database using Hibernate