Welcome to

JDBC Programming Workshop

UTD Computer Science Outreach

Get Code and Presentation

http://bit.ly/16XO2bt

Hibernate

Hibernate is a high-performance Object/Relational persistence and query service which is licensed under the open source GNU Lesser General Public License (LGPL) and is free to download. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities.

First problem, what if we need to modify the design of our database after having developed few pages or our application?

Mismatch problems in Loading and Sharing:

- Granularity- Sometimes you will have an object model which has more classes than the number of corresponding tables in the database.
- Inheritance - RDBMSs do not define anything similar to Inheritance
- Associations- Object-oriented languages represent associations using object references where as am RDBMS represents an association as a foreign key column.
- Navigation- The ways you access objects in Java and in a RDBMS are fundamentally different.

Why Object Relational Mapping (ORM)?

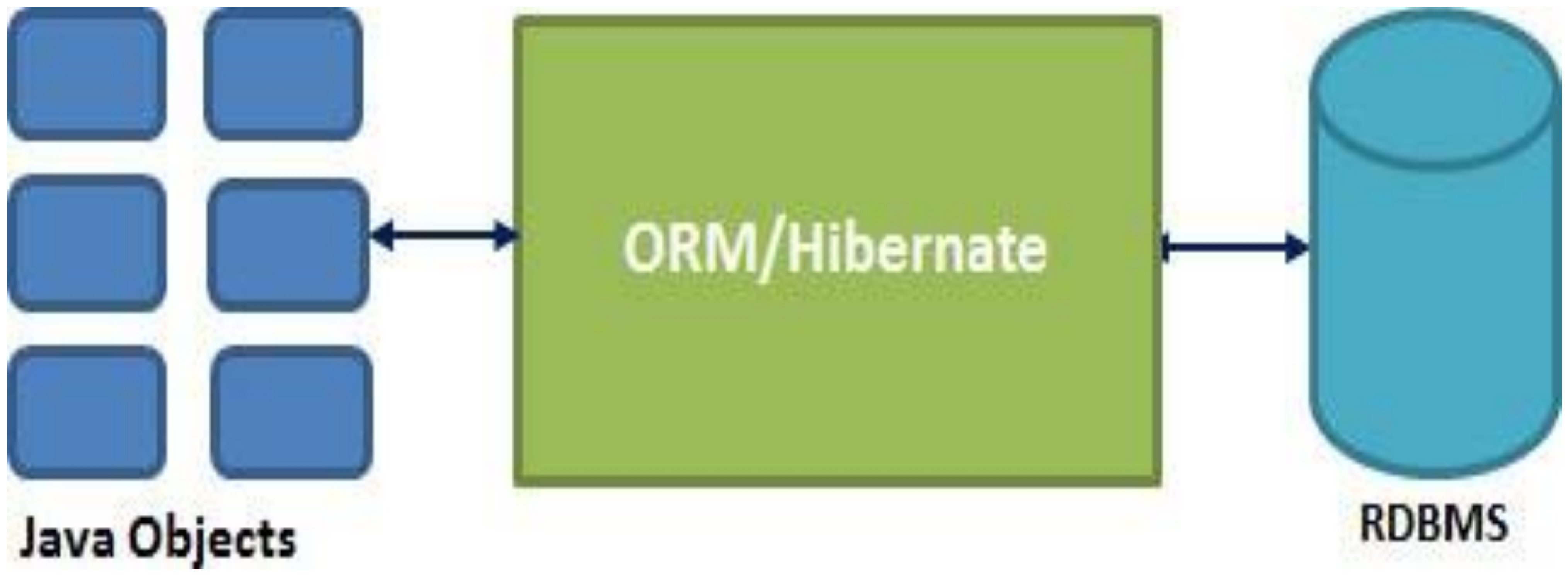
- ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc
- When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C#

Java ORM Frameworks:

- There are several persistent frameworks and ORM options in Java, Which is an ORM service that stores and retrieves objects into a relational database.
- Enterprise JavaBeans Entity Beans
- Java Data Objects
- Castor
- TopLink
- Spring DAO
- Hibernate

- Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

- Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.



Hibernate Configuration

Hibernate requires to know in advance where to find the mapping information that defines how your Java classes relate to the database tables. Hibernate also requires a set of configuration settings related to database and other related parameters. All such information is usually supplied as a standard Java properties file called **hibernate.properties**, or as an XML file named **hibernate.cfg.xml**.

Hibernate Mapper files

An Object/relational mappings are usually defined in an XML document.

This mapping file instructs Hibernate how to map the defined POJO class or classes to the database tables.

Hibernate Mapping Types

- When you prepare a Hibernate mapping document, we have seen that you map Java data types into RDBMS data types. The **types** declared and used in the mapping files are not Java data types; they are not SQL database types either. These types are called Hibernate mapping types, which can translate from Java to SQL data types and vice versa.
- integer >> Integer
- long >> Long
- string >> String
- date >> Date/ Timestamp

Hibernate Query Language

- HQL is an object-oriented query language, instead of operating on tables and columns. HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries

-
  ```
  String hql = "FROM Employee";
  Query query = session.createQuery(hql);
  List results = query.list();
  ```

-
  ```
  String hql = "SELECT E.firstName FROM Employee E WHERE E.id = 10";
  Query query = session.createQuery(hql);
  List results = query.list();
  ```

-
  ```
  String hql = "FROM Employee E WHERE E.id = :employee_id";
  Query query = session.createQuery(hql);
  query.setParameter("employee_id",10);
  List results = query.list();
  ```

CASE STUDY :: Employee Database

Java DAO

The Data Access Object design pattern provides a technique for separating object persistence and data access logic from any particular persistence mechanism or API. There are clear benefits to this approach from an architectural perspective.

The Java DAO approach provides flexibility to change application's persistence mechanism over time without the need to re-engineer application logic that interacts with the Data Access object tier

Thank You !!!

Questions ???

Feedback: bit.ly/cwsfeed