Hibernate-1-Introduction

Presented by Sagar
Java Consultant

Hibernate – What?

- Object – Relational Mapping tool for Java
- Maps Java classes to database tables
- Powerful , high performance , open source OR persistence and query service
- Avoids JDBC API for Result set handling

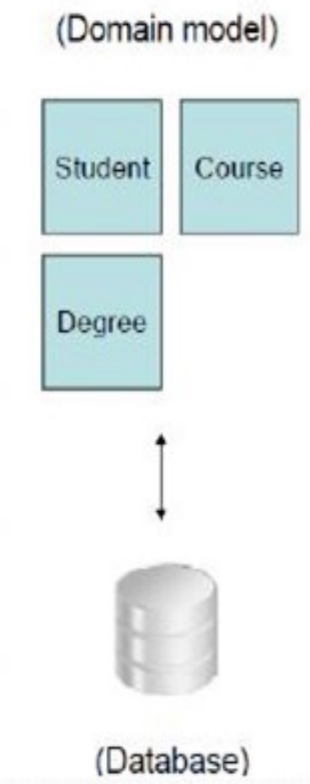
RDBMS – OOP

•**RDBMS :**

- Flexible and robust approach to data management
- De-facto standard in software development

•**OOP :**

- Business logic can be implemented in Java (opposed to stored procedures)
- Allows for use of design patterns and concepts like polymorphism
- Improves code reuse and maintainability



•**Hence demand for mapping interaction**

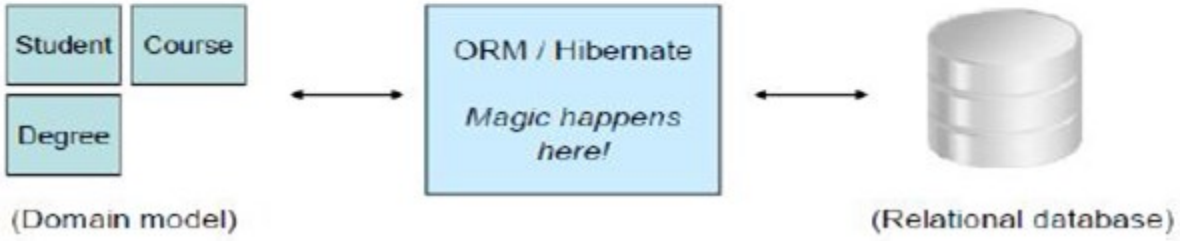
ORM

The Solution - Object – Relational Mapping

Provides a simple API for storing and retrieving Java objects directly to and from the database

No need to follow specific rules or design patterns

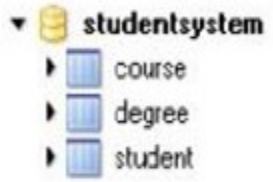
Transparent: Direct data manipulation in RDBMS using an OOP Language



Hibernate - Why?

- Object model and relational model mismatch
- Mapping is an issue

```
public class Student
{
    private String name;
    private String address;
    private Set<Course> courses;
    private Set<Degree> degrees;
}
```



Java object with properties and associations

Relational database with tables and columns

Hibernate – Why?

- Database independent application
- Avoid writing queries
- Avoid JDBC API completely
- Hibernate uses connection pooling technique
- Automatic Key Generation
- Develop the Application in short Period of time

Hibernate – Why?

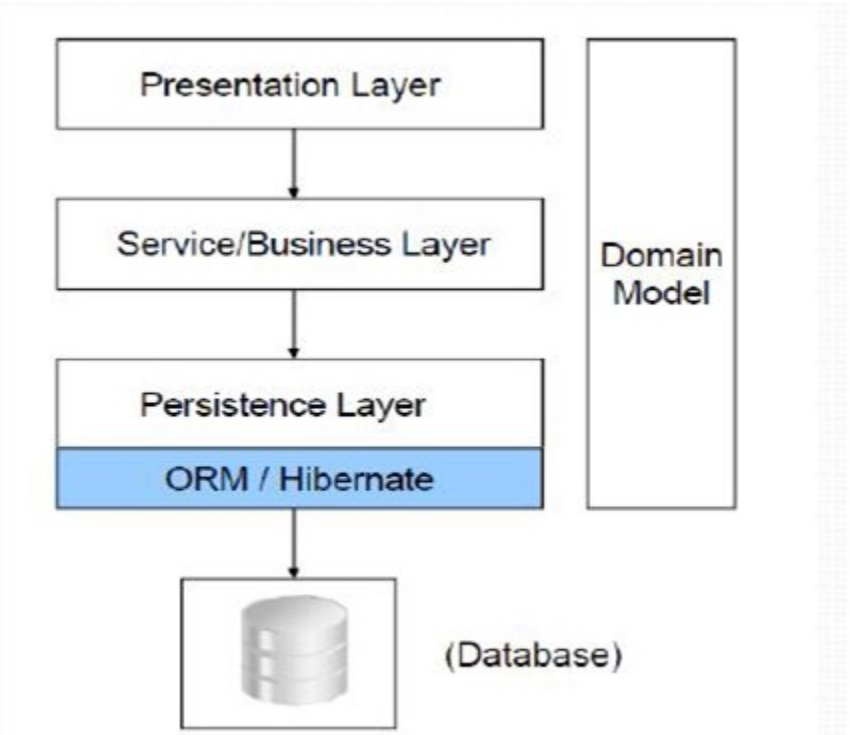
Impedance mismatch
- Object-oriented vs. relational

Java developers are not database developers
- Reduce the need for developers to know and fully understand database design, SQL, performance tuning
- Increase portability across database vendors

Increase performance by deferring to experts
- Potential decrease in database calls
- More efficient SQL statements

ORM Architecture

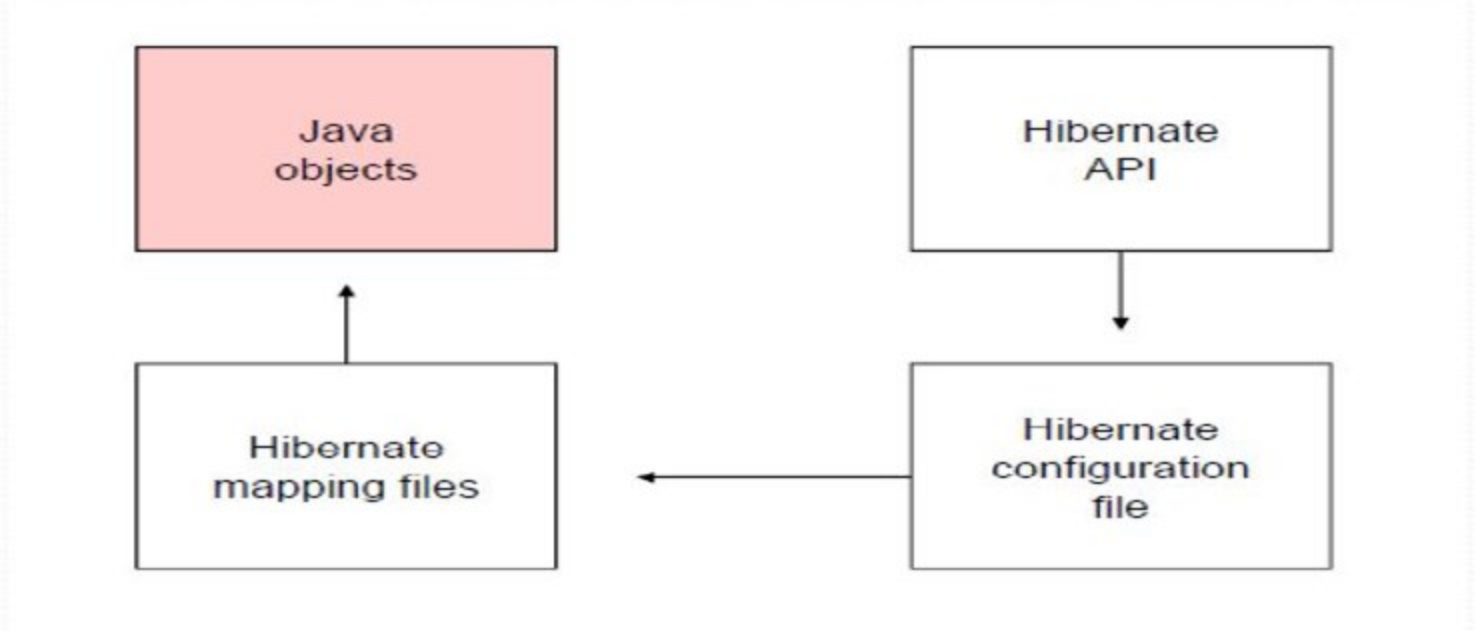
- Middleware that manages persistence
- Provides an abstraction layer between the domain model and the database



ORM Features

- Provides full-featured query facilities
- Support for query in native SQL Dialect
- Takes less Development time
- Supports Automatic Key generation
- XML Binding
- High performance object caching.

Hibernate Components



The POJO

```
public class Contact {

  private long id;
  private String firstName;
  private String lastName;
  private String email;

  public String getEmail() {
    return email;          }

  public String getFirstName() {
    return firstName;          }

  public String getLastName() {
    return lastName;          }

  public void setEmail(String string) {
    email = string;                }

  public void setFirstName(String string) {
    firstName = string;          }

  public void setLastName(String string) {
    lastName = string;       }

  public long getId() {
    return id;       }

  public void setId(long l) {
    id = l;       }
}
```

Java objects

Mapping file

- The file **contact.hbm.xml** is used to map Contact Object to the Contact table in the database.

```
<hibernate-mapping>
    <class name="Example.Contact" table="CONTACT">
        <id name="id" type="long" column="ID" >
            <generator class="assigned"/>
        </id>

        <property name="firstName">
            <column name="FIRSTNAME" />
        </property>
        <property name="lastName">
            <column name="LASTNAME"/>
        </property>
        <property name="email">
            <column name="EMAIL"/>
        </property>
    </class>
</hibernate-mapping>
```

Hibernate mapping files

Configuration File

- Hibernate uses the **hibernate.cfg.xml** to create the connection pool and setup required environment.

Hibernate configuration file

```
<hibernate-configuration>
    <session-factory>
      <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver
       </property>
      <property name="hibernate.connection.url">jdbc:mysql://localhost/hibernate
    </property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>
        <property name="hibernate.connection.pool_size">10</property>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <!-- Mapping files -->
        <mapping resource="contact.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Hibernate Object life cycle

```
Session session = factory.openSession();
Student student = new Student();
student.setName("chandrashekhar");
Transaction transaction = session.beginTransaction();
```

Student Object is in Transient State

```
session.save(student);
transaction.commit();
session.close();
```

Student Object is in Persistent State

student Detached state

Thank You