ORM Service

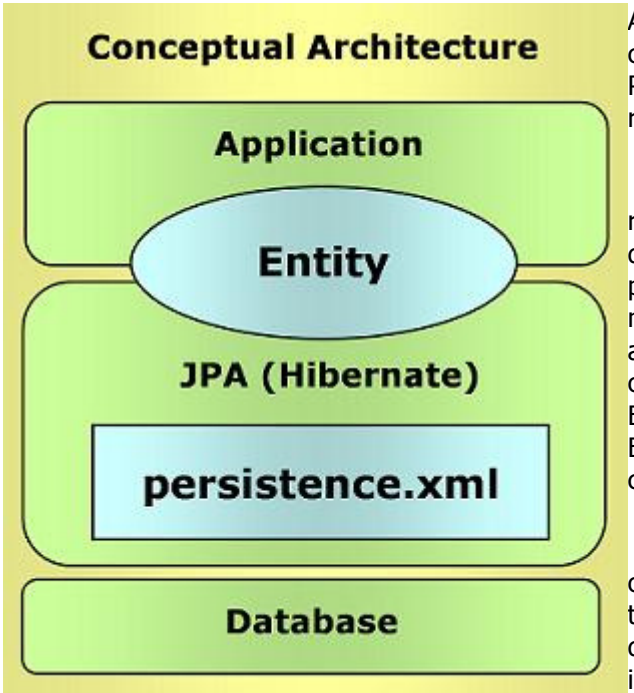
Summary

As an OR Mapping service that resolves the inconsistency between Object Oriented Modeling and Relational Data Modeling, it presents JPA standard service, the Java standard as implement and uses Hibernate, known to have the highest performance in JPA implements. Features of service are as follows:

- Since it is not affected by specific DBMS, it can operate with change of setting information, without change on data access processing code, even if DBMS changes.
- Reduce the time of creating SQL and of creating the code to change to the object to deliver from the SQL execution result. However, SQL taskis also possible if required.
- Adopt the Lazy Loading strategy of approaching to DBMS at the required time only, and pursue performance improvement of application by reducing the number of approach for DBMS through Cache utilization.
- Enable easy work by defining it as the minimum annotation in Entity Class without managing mapping in separate XML file.
- As the Entity Class is defined as a general class, it can be used as the persistence object while applying inheritance, diversity or encapsulation as they area.
- Support the JPA tool(Dali) that enables easy development and supports the implements in many vendors since it is Java standard.
- Learning is required for the developer who is familiar with the processing method using SQL. Obstacle exists due to this.

Description

Main Concept



As shown in the figure to the left, the required main components for DBMS based application are Entity, Persistence.xml and each of them performs the following roles.

- Entity: It deals with data that physically exists regardless of application execution. In general, if developing the application that uses DBMS data, it processes the data of application through SQL that matches specific DBMS in the business layer of application. In JPA-based application, however, interlock of DBMS and data of application is possible, focusing on Entity. Since mapping related items can be defined in Entity class based on annotation, relationship with table can be expressed without separate file.

- Persistence.xml: a file to create essentially if operating the application using JPA as a configuration file that can designate the property per implement or designate the declarative or target entity class for implement.

- JPA(Hibernate): Elements of Hibernates as JPA implement are Hibernate Core , Hibernate Annotations and Hibernate EntityManager and include implementation class such as Entity Manager required for JPA configuration.

- JPA Tool: JPA support tool includes Dali JPA Tools as a sub project in Eclipse Web Tools Platform. Using this tool, tasks requiring hard works such as creation of Entity class from the table created at DB can be automatically processed. See Dali Homepagefor more details.

Get Started

Explain what is required to start ORM service simply before detailed description on ORM service.

Step1. Preparation

Required Library

Following are the Library list and description required for utilization of this service.

Library	Description	Associated Library
antlr-2.7.7.jar	Parser library	
commons-collections-3.2.jar	Library for collection processing	
commons-dbcp-1.2.2.jar	DataSource library	
commons-logging-1.1.1.jar	Library for Logging processing	Referred in hibernate-annotations-3.4.0.GA.jar
log4j-1.3alpha-8.jar	Library for Logging processing	
slf4j-api-1.5.3.jar	Library for Logging processing	
slf4j-log4j12-1.5.3.jar	Library for Logging processing	
commons-pool-1.3.jar	Library for pooling processing	Referred in commons-dbcp-1.2.2.jar
dom4j-1.6.1.jar	XML parsing library	Referred in hibernate-3.2.4.ga.jar
ejb3-persistence-1.0.2.GA.jar	JPA Interface class library	
hibernate-annotations-3.4.0.GA.jar	Hibernate Annotation	
hibernate-entitymanager-3.4.0.GA.jar	Hibernate Entity Manager implement library	
hibernate-commons-annotations-3.1.0.GA.jar	Hibernate common annotation library	Referred in hibernate-entitymanager-3.4.0.GA.jar
hibernate-core-3.3.0.SP1.jar	Hiberante Core library	Referred in hibernate-entitymanager-3.4.0.GA.jar
javassist-3.4.GA.jar	Java bytecode operation library	Referred in hibernate-entitymanager-3.4.0.GA.jar
jta-1.1.jar	JTA interface library	Referred in hibernate-entitymanager-3.4.0.GA.jar
hsqldb-1.8.0.10.jar	HSQL JDBC driver	
mysql-connector-java-5.1.6.jar	MYSQL JDBC driver	
ojdbc-14.jar	ORACLE JDBC driver	
junit-4.4.jar	Test support library	

Step2. Creating Entity Class

Create Entity class of simple forms. It consists of 4 Attributes, which consists of respective getter,settermethods.

Entity Class

```
@Entity
public class Department implements Serializable {

private static final long serialVersionUID = 1L;

    @Id
privateStringdeptId;

privateStringdeptName;

privateDatecreateDate;
```

```
privateBigDecimalempCount;

publicStringgetDeptId() {
returndeptId;
    }

public void setDeptId(StringdeptId) {
this.deptId = deptId;
    }
    ...
}
```

- @Entity: Defining that Department is Entity class
- @Id: Designating the Primary Key information

Step3. Creating persistence.xml

This skill provides implement as a property file to perform JPA with Entity class defined above, and include class information, entity class information, DB access information, logging information and table auto-creation information.

```
<persistence-unit name="PersistUnit" transaction-type="RESOURCE_LOCAL">

<provider>org.hibernate.ejb.HibernatePersistence</provider>

<class>egovframework.Department</class>
<exclude-unlisted-classes/>

<properties>
<property name="hibernate.connection.driver_class" value="org.hsqldb.jdbcDriver"/>
<property name="hibernate.connection.url" value="jdbc:hsqldb:mem:testdb"/>
<property name="hibernate.connection.username" value="sa"/>
<property name="hibernate.dialect" value="org.hibernate.dialect.HSQLDialect"/>

<property name="hibernate.connection.autocommit" value="false"/>
<property name="hibernate.show_sql" value="true"/>
<property name="hibernate.format_sql" value="true"/>
<property name="hibernate.hbm2ddl.auto" value="create"/>
</properties>

</persistence-unit>
```

- provider: designating the class of implements
- class: defining entity class name
- exclude-unlisted-classes: the class not designated above is excluded even if it is defined as entity
- hibernate.connection.*: being able to change according to respective environment with DB connection information
- hibernate.dialect: class for use of Hibernate per DBMS(defined separately per DBMS)
- hibernate.connection.autocommit: requiring declarative commit since whether to set auto commit is false.
- hibernate.show_sql: including SQL in the log
- hibernate.format_sql: printing SQL according to format
- hibernate.hbm2ddl.auto: creating DDL automatically, create the table for the class defined as Entity

Step4. Creating test class

We have form the task of entering, modifying, viewing and deleting processing in the form of JUNIT, using the department defined above.

```
@Test
public void testDepartment() throws Exception {

Stringmodifyname = "Marketing Department";
Stringdeptid = "DEPT-0001";
    Department department = makeDepartment(deptid);

    // Create Entity Manager
emf = Persistence.createEntityManagerFactory("PersistUnit");
em = emf.createEntityManager();

    // Input
em.getTransaction().begin();
em.persist(department);
em.getTransaction().commit();

em.getTransaction().begin();
    Department departmentAfterInsert = em.find(Department.class, deptid );
    // Confirm input
assertEquals("Department Name
Compare!",department.getDeptName(),departmentAfterInsert.getDeptName());

    // Modify
departmentAfterInsert.setDeptName(modifyname);
em.merge(departmentAfterInsert);
em.getTransaction().commit();

em.getTransaction().begin();
    Department departmentAfterUpdate = em.find(Department.class, deptid );
    // Confirm Modification
assertEquals("Department Modify Name
Compare!",modifyname,departmentAfterUpdate.getDeptName());

    // Delete
em.remove(departmentAfterUpdate);
em.getTransaction().commit();

    // Confirm Deletion
    Department departmentAfterDelete = em.find(Department.class, deptid );
assertNull("Department is Deleted!",departmentAfterDelete);

em.close();

}
```

- Persistence.createEntityManagerFactory: Create Entity Manager Factory
- emf.createEntityManager: Create Entity Manager
- em.getTransaction().begin(): Start Transaction
- em.getTransaction().commit(): Commit
- em.persist: Insert
- em.find: SELECT
- em.merge: UPDATE
- em.remove: DELETE
- assertEquals: Compare whether the values are same(JUnitmethod)
- assertNull: check whether it is NULL or not(JUnitmethod)

Step5. Execute

1. Download the ormsimpleguide.zip file and unzip.
2. Select the folder unzipped at Eclipse and import the project.
3. Check whether there is META-INF/persistence.xml, log4j.xml in Department.java, DepartmentTest.java, resources folder in the src folder of project.

4. Check whether there is library file in lib.
5. Select DepartmentTest.java, right-click to run Run As >JUnit Test.
6. Check whether it is normally performed in JUnit result window.

※In case of ORACLE or MySQL, if you set and execute in reference to annotation of persistence.xml, you can check whether it works properly.

Detailed Description

1. Entities
2. Entity Operation
3. Association Mapping
4. Query Language
5. Native SQL
6. Concurrency
7. Cache Handling
8. Fetch Strategy
9. Spring Integration
10. JPA Configuration

Reference