

Hibernate Step by Step Guide

“Code with Passion!”



What we are going to build

- A simple Hibernate application persisting *Person* objects
- The database table, *person*, has the following columns
 - *int id*
 - *string cname*
- We will add data to and retrieve data from the table through Hibernate
- We will use the following database servers
 - Java DB (Derby)
 - MySQL as database servers
- We will use the two schemes representing domain-class/table mapping
 - XML
 - Annotations



Steps for Building Simple Hibernate App

Steps to follow for Writing files

1. Write domain classes (as POJO classes)

➤ *Person.java*

2. Write Hibernate mapping files for the domain classes (per each domain class) if we choose to use XML for domain class to table mapping

➤ *Person.hbm.xml*

3. Write Hibernate configuration file (per application)

➤ *hibernate.cfg.xml*

Step 1: Write Domain Classes (Person.java)

```
public class Person implements Serializable {  
  
    private int id;  
    private String name;  
  
    protected Person() {  
    }  
  
    public Person(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
}
```

Step 2: Write Mapping File for Each Domain Class (Person.hbm.xml)

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd"
<hibernate-mapping>
    <class name="Person" table="person">
        <id name="id" type="int">
            <generator class="increment"/>
        </id>

        <property name="name" column="cname" type="string"/>
    </class>
</hibernate-mapping>
```

Step 3: Write Hibernate configuration file (hibernate.cfg.xml) – Java DB

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property
name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property
name="connection.url">jdbc:derby://localhost:1527/mydatabase</property>
        <property name="connection.username">app</property>
        <property name="connection.password">app</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.DerbyDialect</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">>false</property>

        <!-- Mapping files -->
        <mapping resource="Person.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Lab:

Exercise 0-2: Build “Helloworld” App
3513_hibernate_stepbystep.zip





Using MySQL

Modify Hibernate configuration file (hibernate.cfg.xml) – MySQL

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property
    name="connection.url">jdbc:mysql://localhost:3306/mydatabase</property
>
<property name="connection.username">root</property>
<property name="connection.password"></property>

<property name="show_sql">>true</property>
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="myeclipse.connection.profile">mysql</property>

<mapping resource="Person.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

Lab:

**Exercise 3: Use MySQL as
Database server**
3513_hibernate_stepbystep.zip





Using Annotation for Domain Class

Step 2: Use Annotation for Domain classes (instead of XML file)

@Entity

@Table(name = "person")

public class Person implements Serializable{

@Id

@GeneratedValue(strategy=GenerationType.*IDENTITY*)

@Column(name = "id")

private int id;

@Column(name = "CNAME")

private String name;

...

Lab:

**Exercise 4: Use Annotation for
Domain classes**
3513_hibernate_stepbystep.zip



Code with Passion!

