Hibernate One To One Mapping Tutorial (XML Mapping)

BY VIRAL PATEL · NOVEMBER 15, 2011

Let us understand how One-to-one mapping works in Hibernate. Following is a simply yet concept building example where we will understand One-to-one mapping in Hibernate framework using XML Mappings. We will use two tables "employee" and "employeedetail" which exhibits one-to-one relationship. Using Hibernate we will implement this relationship.

FOLLOW:

RECENT POSTS

Spring Boot Custom Favicon Example – How to set custom Favicon in Spring Boot

Spring Boot JSP Hello World Tutorial with Example

Hibernate Tutorial Series

- Introduction to Hibernate Framework
- Hibernate Maven MySQL Hello World example (XML Mapping)
- Hibernate Maven MySQL Hello World example (Annotation)
- Understanding Relationship Mapping
  - One To One Mapping example (XML Mapping)
  - One To One Mapping example (Annotation)
  - One To Many Mapping example (XML Mapping)
  - One To Many Mapping example (Annotation)
  - Many To Many Mapping example (XML Mapping)
  - Many To Many Mapping example (Annotation)
  - Self-Join One To Many Annotations Mapping example
  - Self-Join Many To Many Annotations Mapping example
- Inheritance in Hibernate
  - One Table Per Class Hierarchy (Annotation & XML mapping)
  - One Table Per Subclass (Annotation & XML mapping)
  - One Table Per Concrete Class (Annotation & XML mapping)

Tools and technologies used in this article:

1. Java JDK 1.5 above
2. MySQL 5 above
3. Eclipse 3.2 above

Spring 4 MVC REST Controller Example (JSON CRUD Tutorial)

Spring 4 MVC Tutorial Maven Example – Spring Java Configuration

Find Process ID of Process using a Port in Windows

4. Hibernate 3 above
5. Maven 3 above

1. Create Database

We will use MySQL in our example. Create two tables “employee” and “employeedetail” with One-to-one relationship. Following is the SQL script for same.

```
/* EMPLOYEE table */
CREATE TABLE `employee` (
    `employee_id` BIGINT(10) NOT NULL AUTO_INCREMENT,
    `firstname` VARCHAR(50) NULL DEFAULT NULL,
    `lastname` VARCHAR(50) NULL DEFAULT NULL,
    `birth_date` DATE NOT NULL,
    `cell_phone` VARCHAR(15) NOT NULL,
    PRIMARY KEY (`employee_id`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=216



/* EMPLOYEEDETAIL table */
CREATE TABLE `employeedetail` (
    `employee_id` BIGINT(20) NOT NULL AUTO_INCREMENT,
    `street` VARCHAR(50) NULL DEFAULT NULL,
    `city` VARCHAR(50) NULL DEFAULT NULL,
    `state` VARCHAR(50) NULL DEFAULT NULL,
    `country` VARCHAR(50) NULL DEFAULT NULL,
    PRIMARY KEY (`employee_id`),
    CONSTRAINT `FKEMPL` FOREIGN KEY (`employee_id`) REFERENCES `employee` (`employee
```

```
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=216
```

Note that a one-to-one relationships occurs when one entity is related to exactly one occurrence in another entity. Thus, there will be one primary key which will be mapped with both the entities.

2. Hibernate Maven dependency

Following dependencies we will use in Maven which will add Hibernate support.

File: /pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/ma
    <modelVersion>4.0.0</modelVersion>
    <groupId>net.viralpatel.hibernate</groupId>
    <artifactId>HibernateHelloWorldXML</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>HibernateHelloWorldXML</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
```

```
            <version>5.1.10</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate</artifactId>
            <version>3.2.6.ga</version>
        </dependency>
    </dependencies>
</project>
```

3. Hibernate Model class

For this example we will create two Model class: Employee.java and EmployeeDetail.java. Add following in your project.

File: /src/main/java/net/viralpatel/hibernate/Employee.java

```
package net.viralpatel.hibernate;

import java.sql.Date;

public class Employee {

    private Long employeeId;
    private String firstname;
    private String lastname;
    private Date birthDate;
    private String cellphone;
    private EmployeeDetail employeeDetail;

    public Employee() {
```

```
    }

    public Employee(String firstname, String lastname, Date birthdate,
            String phone) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthDate = birthdate;
        this.cellphone = phone;
    }

    // Getter and Setter methods
}
```

File: /src/main/java/net/viralpatel/hibernate/EmployeeDetail.java

```
package net.viralpatel.hibernate;

public class EmployeeDetail {

    private Long employeeId;
    private String street;
    private String city;
    private String state;
    private String country;

    private Employee employee;

    public EmployeeDetail() {

    }

    public EmployeeDetail(String street, String city, String state, String country)
```

```
        this.street = street;
        this.city = city;
        this.state = state;
        this.country = country;
    }

    // Getter and Setter methods

}
```

Note that in above model classes, employeeId is common. This is the primary key of Employee table that exhibits One-to-one relationship with EmployeeDetail table.

4. Hibernate XML Mapping (HBM)

Add following hibernate mapping (hbm) files `Employee.hbm.xml` and `EmployeeDetail.hbm.xml` for the model classes created in above steps.

File: /src/main/java/net/viralpatel/hibernate/Employee.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="net.viralpatel.hibernate">

    <class name="Employee" table="EMPLOYEE">
        <id name="employeeId" column="EMPLOYEE_ID">
            <generator class="native" />
        </id>
```

```
        <one-to-one name="employeeDetail" class="net.viralpatel.hibernate.EmployeeDe
            cascade="save-update"></one-to-one>

        <property name="firstname" />
        <property name="lastname" column="lastname" />
        <property name="birthDate" type="date" column="birth_date" />
        <property name="cellphone" column="cell_phone" />

    </class>
</hibernate-mapping>
```

File: /src/main/java/net/viralpatel/hibernate/EmployeeDetail.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="net.viralpatel.hibernate">

    <class name="EmployeeDetail" table="EMPLOYEEDETAIL">

        <id name="employeeId" type="java.lang.Long">
            <column name="EMPLOYEE_ID" />
            <generator class="foreign">
                <param name="property">employee</param>
            </generator>
        </id>
        <one-to-one name="employee" class="net.viralpatel.hibernate.Employee"
            constrained="true"></one-to-one>

        <property name="street" column="STREET"/>
        <property name="city" column="CITY"/>
```

```
        <property name="state" column="STATE"/>
        <property name="country" column="COUNTRY"/>
    </class>

</hibernate-mapping>
```

Note that in above hibernate mapping we are implementing One-to-one relationship. For both the model classes we are using a single primary key EmployeeId. In EmployeeDetail hbm file we have defined a foreign identifier generator so that primary it uses primary key of Employee table.

5. Hibernate Configuration

Following is the hibernate configuration hibernate.cfg.xml file. Add this in your project.

File: /src/main/resources/hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//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/tutorial</proper
        <property name="connection.username">root</property>
        <property name="connection.password"></property>

        <property name="connection.pool_size">1</property>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
        <property name="current_session_context_class">thread</property>
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</p
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">validate</property>

        <mapping resource="net/viralpatel/hibernate/EmployeeDetail.hbm.xml"/>
        <mapping resource="net/viralpatel/hibernate/Employee.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

6. Hibernate Utility class

File:/src/main/java/net/viralpatel/hibernate/HibernateUtil.java

```
package net.viralpatel.hibernate;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            return new Configuration()
                    .configure()
                    .buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
```

```
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

7. Main class to test One-to-one mapping

Add following Main.java class in your project to test One-to-one relationship mapping functionality.

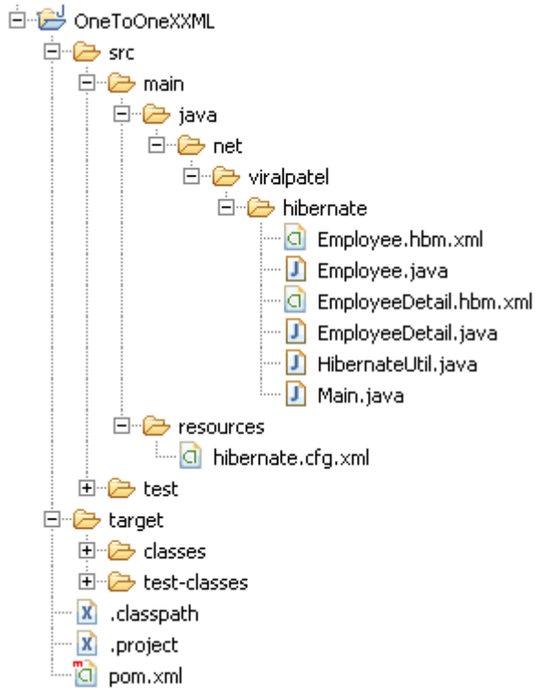
File:/src/main/java/net/viralpatel/hibernate/Main.java

```
package net.viralpatel.hibernate;

import java.sql.Date;
import java.util.List;

import org.hibernate.Session;
import org.hibernate.SessionFactory;

public class Main {

    public static void main(String[] args) {

        SessionFactory sf = HibernateUtil.getSessionFactory();
```

```
        Session session = sf.openSession();
        session.beginTransaction();

        EmployeeDetail employeeDetail = new EmployeeDetail("10th Street", "LA", "San

        Employee employee = new Employee("Nina", "Mayers", new Date(121212),
                "114-857-965");
        employee.setEmployeeDetail(employeeDetail);
        employeeDetail.setEmployee(employee);


        session.save(employee);


        List<Employee> employees = session.createQuery("from Employee").list();
        for (Employee employee1 : employees) {
            System.out.println(employee1.getFirstname() + " , "
                    + employee1.getLastname() + ", "
                    + employee1.getEmployeeDetail().getState());
        }

        session.getTransaction().commit();
        session.close();

    }
}
```

8. Review Project Structure

Below is the final project structure with all the source files.



8. Execute the example

Execute the Main class. Hibernate will insert a row in Employee and EmployeeDetail table.

Output:

```
Hibernate: insert into EMPLOYEE (firstname, lastname, birth_date, cell_phone) values
Hibernate: insert into EMPLOYEEDETAIL (STREET, CITY, STATE, COUNTRY, EMPLOYEE_ID) va
Hibernate: select employee0_.EMPLOYEE_ID as EMPLOYEE1_1_, employee0_.firstname as fi
Hibernate: select employeede0_.EMPLOYEE_ID as EMPLOYEE1_0_0_, employeede0_.STREET as
Nina , Mayers, San Francisco
```

That's All Folks

Today we saw how to write Hibernate program and implements One-to-one relationship mapping using XML mappings. We used Maven to generate Java project and added Hibernate dependencies into it.

Download Source

Hibernate-One-to-one-tutorial-XML-Mapping.zip (9 kb)

Related Articles

1. **Hibernate One To One Annotation Mapping Tutorial**
2. **Hibernate Many To Many XML Mapping Tutorial**
3. **Hibernate Many To Many Annotation Mapping Tutorial**
4. **Hibernate One To Many XML Mapping Tutorial**
5. **Hibernate Self Join Many To Many Annotations mapping example**
6. **Hibernate Self Join Annotations One To Many mapping example**
7. **Hibernate Maven MySQL Hello World example (XML Mapping)**

Tags: Hibernate hibernate-configuration maven MySQL

PREVIOUS STORY
Hibernate Hello World example using Annotation

NEXT STORY
Change spring-servlet.xml Filename (Spring Web Contenxt Configuration Filename)

YOU MAY ALSO LIKE...

How to: Reset MySQL root password

Hibernate One To Many Annotation tutorial

Hibernate Many To Many Annotation Mapping Tutorial

Problem with comparison of float or double column in MySQL

28 COMMENTS

Tapan 23 November, 2011, 6:52

Hi Viral,
Its a lovely post and really appriciated your efforts but i have one doubt only, that is, suppose if i want to create a JSP form which contains both the tables details/columns, how can i do that cause In controll class i will create an instance of Employe.java and will declear all the fields on JSP like but how will i declear 'EmployeeDetails' variable on my JSP if i try 'employeeDetails.street' against any then it doesn't identify and gives error.

so in nutshell my requirement is i have form which have all fields exists on employee and employeeDetails table, how will i declear subTable columns on my JSP page ?
Can you/anyone please guide me for this.
Thanks

Reply

Viral Patel 13 December, 2011, 23:46
Hi Tapan,
The EmployeeDetail is part of Employee object. You can simply pass Employee object to JSP and render all attributes as:

```
${employee.firstname}
${employee.lastname}
${employee.employeeDetail.street}
${employee.employeeDetail.city}
```

Hope this works.

Reply

ajay mittal 23 November, 2011, 14:03
Hi,

It's awesome man.

but can you suggest how to read some data from a text file and check whether these values are the same in a table through HIBERNATE XML and annotations too.
Thanks a lot

Reply

Viral Patel 13 December, 2011, 23:49
Hi Ajay,
Your requirement seems to be very specific. All you need to do is to
1) Read a text file in Java using File IO.

2) Iterate through the data in file
3) Use Hibernate to check if data is present by query the db.

Reply

sherry 5 July, 2012, 16:18

Hi , i want to know about
this
employee
What is this pram tag and its “property” attribute

Reply

sherry 5 July, 2012, 16:20

Hi , i want to know about

employee

What is this pram tag and its “property” attribute

Reply

sherry 5 July, 2012, 16:21

employee
What is this param tag and its “property” atribute

Reply

Anil 9 August, 2012, 15:23

Nice Article viral :)

Very helpful

Thanks,
Anil

Reply

Adarsh 3 September, 2012, 16:12

Thanks A lot man :D

Reply

Sam 20 October, 2012, 12:16

Thanx man. Nice tutorial

Reply

> **Kouhei** 18 November, 2012, 12:53
>
> Hi James! I'm with you because I just ululsay put my PC into hibernate mode. And although, it's not difficult to make some few clicks to hibernate a PC, but if there's an easier way I would appreciate it very much. It's just through this post that I learned to personalize the power button and I just did it! Thanks to you a lot!
>
> Reply

nguyendang 25 October, 2012, 14:05

Hi there,

i got an error when i run my test for the second times and it thow me the error: "Oct 25, 2012 3:21:52 PM org.hibernate.impl.SessionFactoryObjectFactory addInstance
INFO: Not binding factory to JNDI, no JNDI name configured
org.hibernate.SessionException: Session is closed!
Exception in thread "main" java.lang.NullPointerException
at geomancer.utils.testHibernateConnection.main(testHibernateConnection.java:32)
at org.hibernate.impl.AbstractSessionImpl.errorIfClosed(AbstractSessionImpl.java:49)
at org.hibernate.impl.SessionImpl.list(SessionImpl.java:1541)
at org.hibernate.impl.CriteriaImpl.list(CriteriaImpl.java:283)
at geomancer.utils.testHibernateConnection.main(testHibernateConnection.java:27)
Java Result: 1

i don't why, but the first times is ok

Reply

Amit Kumar Singh 17 March, 2013, 11:41

It's very very helpful tutorial……

Reply

Naresh 28 April, 2013, 17:31

Hi Viral,

I have two tables with composite id having one to one relationship.
Table One : PK Columns : column1, column2
Table Two : PK Columns : column3, column4.
Columns are having different names. I have to map column1 – column3 and column2-column4.
How to give one to one association in xml configuration.

Reply

sreenu 7 May, 2013, 18:45

mapping instance can not be created ..exception

Reply

Benito Camelo 19 May, 2013, 1:36

Thanks so much!! you saved my life :D

Reply

techbuzz 4 June, 2013, 12:38

I'm trying to insert only child details with parent already present .Can you please help with it.
Using above method but instead of creating new object of employee getting it from datasource.

```
EmployeeDetail employeeDetail = new EmployeeDetail("10th Street", "LA",

        Employee employee = getEmplyee(employeeid);
        employee.setEmployeeDetail(employeeDetail);
        employeeDetail.setEmployee(employee);


        session.save(employee);
```

Getting error “not-null property references a null or transient value
Check nullability

Reply

Arsh 2 July, 2013, 16:37

Hi Viral,

You have not do session.save(employeedetail)
so without this, is it okay?

Thanks for your reply.

Reply

Bharat 30 August, 2013, 12:20

Hi sherry,

emplyee is a object of Employee(ownerClass) class in EmployeeDetails(ownedClass). property emplyee specifiy, that owned class will take employee id as foreign key.

Reply

ved 22 December, 2013, 23:40

Pls explain it constrained=”true”

Reply

ved 22 December, 2013, 23:48

I got the explanation in hibernate forum
Hi, constrained=”true” is use in case of shared primary key. (in case of parent child relationship, when child’s primary key works as foreign key pointing to primary key of parent)..

Equivalent annotation is
@PrimaryKeyJoinColumn
https://forum.hibernate.org/viewtopic.php?f=1&t=975113
Thanx its very nice tutorial ……

Reply

heffrey 30 March, 2014, 23:15

I have an issue with composition where one class has another:

public class InventoryItem {
Ingredient ingredient;
}

I want to store the IngredientID in my Inventory table as a foreign key to the Ingredient table. Unfortunately, Ingredient is an abstract class with three concrete subclasses. I do not know how to map this properly and haven't seen any info anywhere. Any ideas?

Reply

Bogdan 15 April, 2014, 0:48

Thanks a lot!
It was really briefly and clearly!

Reply

Fawad Khaliq 16 April, 2014, 17:22

Very helpful thanks

Reply

Aliasger Burhanpurwala 10 October, 2014, 14:17

Hi Mr. Patel,

It was a very nice example to understand one-to-one mapping in hibernate and I really appreciate ur efforts to explain it in such a simple way. However I have a doubt. We r getting the employeeDetails data in the employee table. But is the inverse also possible i.e If i query employeeDetails data will that object

also contain the employee data. Actually I am trying to implement similar thing in my project but am facing dome issues. Can u please shed some light on this.

Thankx

Reply

Timi 10 December, 2014, 20:36

very nice tutorial. i got confusesd with the system.out.println as it was showing only some data. changed to:

```
System.out.println(employee1.getFirstname() + &quot; , &quot;
                        + employee1.getLastname() + &quot;, &quot;
                        + employee1.getBirthDate() + &quot;, &quot;
                        + employee1.getCellphone() + &quot;, &quot;
                        + employee1.getEmployeeDetail().getStreet() + &c
                        + employee1.getEmployeeDetail().getCountry() + &
                        + employee1.getEmployeeDetail().getCity() + &quc
                        + employee1.getEmployeeDetail().getState());
```

and now is the full data :P.

Thanks a lot.

Timi

Reply

bala 19 October, 2015, 0:51

Very good tutorial it works for me .thanks for sharing the article

Reply

Naveen Gupta 28 March, 2017, 23:12

Getting this error :
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.Environment).

log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
org.hibernate.HibernateException: hibernate.cfg.xml not found
Exception in thread “main” java.lang.NullPointerException
at com.DAO.Main.main(Main.java:16)

Reply

LEAVE A REPLY

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment