Hibernate One To Many Mapping Example Annotation

APRIL 2, 2018 BY PANKAJ — 19 COMMENTS

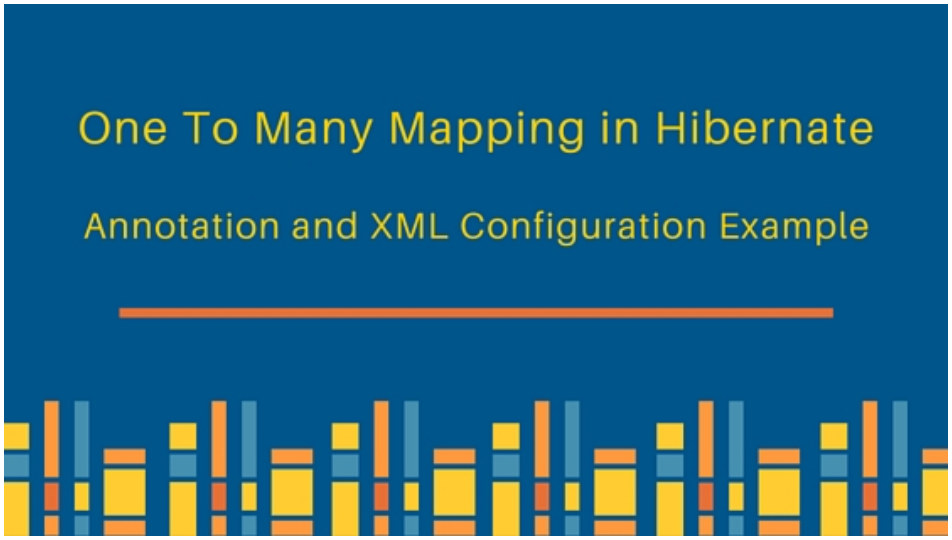
Today we will look into One To Many Mapping in Hibernate. We will look into Hibernate One To Many Mapping example using Annotation and XML configuration.

Table of Contents [hide]

1 One To Many Mapping in Hibernate
- 1.1 One To Many Mapping in Hibernate – Database Setup
- 1.2 Hibernate One To Many Mapping Project Structure
- 1.3 Hibernate Maven Dependencies
- 1.4 Hibernate One To Many Mapping Model Classes
- 1.5 Hibernate SessionFactory Utility Class
- 1.6 Hibernate Configuration XML File
- 1.7 Hibernate One To Many Mapping Example – XML Configuration
- 1.8 Hibernate One To Many Mapping Example – Test Program

2 Hibernate One To Many Mapping Annotation
- 2.1 Hibernate One To Many Mapping Example Annotation
- 2.2 Hibernate SessionFactory Utility Class
- 2.3 Hibernate One To Many Mapping Annotation Model Classes
- 2.4 Hibernate One To Many Mapping Annotation Example Test Program

One To Many Mapping in Hibernate



In simple terms, one to many mapping means that one row in a table can be mapped to multiple rows in another table. For example, think of a Cart system where we have another table for Items. A cart can have multiple items, so here we have one to many mapping. We will use Cart-Items scenario for our hibernate one to many mapping example.

One To Many Mapping in Hibernate – Database Setup

We can use **foreign key constraint** for one to many mapping. Below is our database script for `Cart` and `Items` table. I am using MySQL database for Hibernate one to many mapping example.

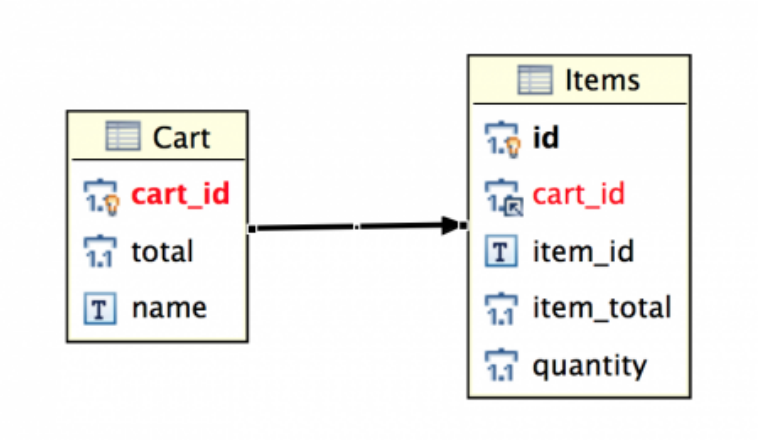
`setup.sql`

```
CREATE TABLE `Cart` (
  `cart_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `total` decimal(10,0) NOT NULL,
  `name` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`cart_id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

CREATE TABLE `Items` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `cart_id` int(11) unsigned NOT NULL,
  `item_id` varchar(10) NOT NULL,
  `item_total` decimal(10,0) NOT NULL,
  `quantity` int(3) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cart_id` (`cart_id`),
```

```
  CONSTRAINT `items_ibfk_1` FOREIGN KEY (`cart_id`) REFERENCES `Cart` (`cart_id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
```

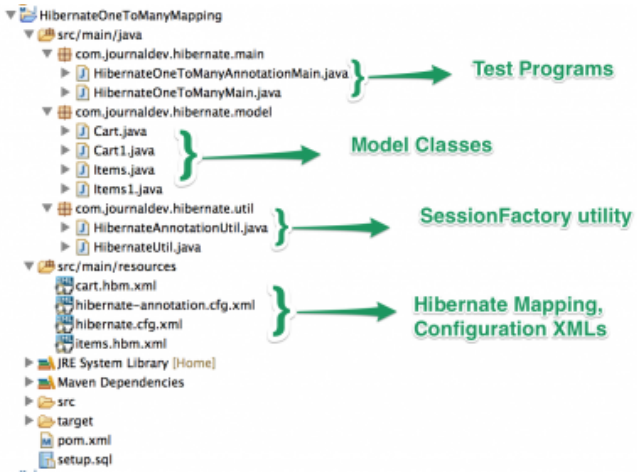
Below is the ER diagram of the Cart and Items table.



Our database setup is ready, let's move on to creating hibernate One to Many Mapping example project. First of all, we will use XML based configuration and then we will implement one to many mapping using Hibernate and JPA annotation.

Hibernate One To Many Mapping Project Structure

Create a simple Maven project in Eclipse or you favorite IDE, the final project structure will look like below image.



Hibernate Maven Dependencies

Our final pom.xml file contains dependencies for Hibernate and MySQL driver. Hibernate uses JBoss logging and it automatically gets added as transitive dependencies.

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.journaldev.hibernate</groupId>
    <artifactId>HibernateOneToManyMapping</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
          <dependency>
                  <groupId>org.hibernate</groupId>
                  <artifactId>hibernate-core</artifactId>
                  <version>4.3.5.Final</version>
          </dependency>
          <dependency>
                  <groupId>mysql</groupId>
                  <artifactId>mysql-connector-java</artifactId>
                  <version>5.0.5</version>
          </dependency>
    </dependencies>
```

Note that I am using latest Hibernate version **4.3.5.Final** and MySQL driver version based on my database installation.

Hibernate One To Many Mapping Model Classes

For our tables Cart and Items, we have model classes to reflect them.

Cart.java

```
  package com.journaldev.hibernate.model;

  import java.util.Set;

  public class Cart {

          private long id;
          private double total;
          private String name;
          private Set<Items> items;

          public long getId() {
                  return id;
          }
```

```
	public void setId(long id) {
		this.id = id;
	}
	public double getTotal() {
		return total;
	}
	public void setTotal(double total) {
		this.total = total;
```

I am using Set of Items, so that every record is unique. We can also use List or Array for one to many mapping in hibernate.

Items.java

```
package com.journaldev.hibernate.model;

public class Items {

	private long id;
	private String itemId;
	private double itemTotal;
	private int quantity;
	private Cart cart;

	//Hibernate requires no-args constructor
	public Items(){}

	public Items(String itemId, double total, int qty, Cart c){
		this.itemId=itemId;
		this.itemTotal=total;
		this.quantity=qty;
		this.cart=c;
	}
	public String getItemId() {
		return itemId;
	}
```

Items have many to one relationship to Cart, so we don't need to have Collection for Cart object.

Hibernate SessionFactory Utility Class

We have a utility class for creating Hibernate SessionFactory.

HibernateUtil.java

```
package com.journaldev.hibernate.util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {

        private static SessionFactory sessionFactory;

        private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
                Configuration configuration = new Configuration();
                configuration.configure("hibernate.cfg.xml");
                System.out.println("Hibernate Configuration loaded");

                ServiceRegistry serviceRegistry = new
StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

                System.out.println("Hibernate serviceRegistry created");
```

Hibernate Configuration XML File

Our hibernate configuration xml file contains database information and mapping resource details.

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password">pankaj123</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>
        <property name="hibernate.connection.username">pankaj</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
        <property name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.show_sql">true</property>

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

Hibernate One To Many Mapping Example – XML Configuration

This is the most important part of tutorial, let's see how we have to map both Cart and Items classes for one to many mapping in hibernate.

cart.hbm.xml

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

<hibernate-mapping package="com.journaldev.hibernate.model">
        <class name="Cart" table="CART" >
                <id name="id" type="long">
                        <column name="cart_id" />
                        <generator class="identity" />
                </id>
                <property name="total" type="double">
                        <column name="total" />
                </property>
                <property name="name" type="string">
                        <column name="name" />
                </property>
                <set name="items" table="ITEMS" fetch="select">
                        <key>
                                <column name="cart_id" not-null="true"></column>
                        </key>
                        <one-to-many class="Items"/>
                </set>
```

The important part is the `set` element and `one-to-many` element inside it. Notice that we are providing key to be used for one to many mapping i.e cart_id.

items.hbm.xml

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

<hibernate-mapping package="com.journaldev.hibernate.model">

	<class name="Items" table="ITEMS">
		<id name="id" type="long">
			<column name="id" />
			<generator class="identity" />
		</id>
		<property name="itemId" type="string">
			<column name="item_id"></column>
		</property>
		<property name="itemTotal" type="double">
			<column name="item_total"></column>
		</property>
		<property name="quantity" type="integer">
			<column name="quantity"></column>
		</property>

		<many-to-one name="cart" class="Cart">
```

Notice that from items to cart, it's many to one relationship. So we need to use `many-to-one` element for cart and we are providing column name that will be mapped with the key. So based on the Cart hibernate mapping configuration, it's key cart_id will be used for mapping.

Our project for Hibernate One To Many Mapping Example using XML mapping is ready, let's write a test program and check if it's working fine or not.

Hibernate One To Many Mapping Example – Test Program

`HibernateOneToManyMain.java`

```
package com.journaldev.hibernate.main;

import java.util.HashSet;
import java.util.Set;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
```

```
import com.journaldev.hibernate.model.Cart;
import com.journaldev.hibernate.model.Items;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateOneToManyMain {

        public static void main(String[] args) {

                Cart cart = new Cart();
                cart.setName("MyCart");

                Items item1 = new Items("I1", 10, 1, cart);
                Items item2 = new Items("I2", 20, 2, cart);
```

Notice that we need to save both Cart and Items objects one by one. Hibernate will take care of updating the foreign keys in Items table. When we execute above program, we get following output.

```
Hibernate Configuration loaded
Hibernate serviceRegistry created
Session created
Hibernate: insert into CART (total, name) values (?, ?)
Hibernate: insert into ITEMS (item_id, item_total, quantity, cart_id) values (?, ?, ?,
?)
Hibernate: insert into ITEMS (item_id, item_total, quantity, cart_id) values (?, ?, ?,
?)
Hibernate: update ITEMS set cart_id=? where id=?
Hibernate: update ITEMS set cart_id=? where id=?
Cart ID=6
Closing SessionFactory
```

Notice that Hibernate is using Update query to set the cart_id in ITEMS table.

Hibernate One To Many Mapping Annotation

Now that we have seen how to implement One To Many mapping in Hibernate using XML based configurations, let's see how we can do the same thing using JPA annotations.

Hibernate One To Many Mapping Example Annotation

Hibernate configuration file is almost same, except that mapping element changes because we are using Classes for hibernate one to many mapping using annotation.

hibernate-annotation.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password">pankaj123</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>
        <property name="hibernate.connection.username">pankaj</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <property name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.show_sql">true</property>

        <mapping class="com.journaldev.hibernate.model.Cart1"/>
        <mapping class="com.journaldev.hibernate.model.Items1"/>
    </session-factory>
</hibernate-configuration>
```

Hibernate SessionFactory Utility Class

SessionFactory utility class is almost same, we just need to use the new hibernate configuration file.

HibernateAnnotationUtil.java

```
package com.journaldev.hibernate.util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateAnnotationUtil {

        private static SessionFactory sessionFactory;

        private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate-annotation.cfg.xml
```

```
                Configuration configuration = new Configuration();
                configuration.configure("hibernate-annotation.cfg.xml");
                System.out.println("Hibernate Annotation Configuration loaded");

                ServiceRegistry serviceRegistry = new
StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();
```

Hibernate One To Many Mapping Annotation Model Classes

Since we don't have xml based mapping files, all the mapping related configurations will be done using JPA annotations in the model classes. If you understand the xml based mapping, it's very simple and similar.

Cart1.java

```
package com.journaldev.hibernate.model;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name="CART")
public class Cart1 {

        @Id
        @GeneratedValue(strategy=GenerationType.IDENTITY)
        @Column(name="cart_id")
        private long id;

        @Column(name="total")
```

Important point to note is the `OneToMany` annotation where `mappedBy` variable is used to define the property in `Items1` class that will be used for the mapping purpose. So we should have a property named "cart1" in Items1 class. Don't forget to include all the getter-setter methods.

Items1.java

```
package com.journaldev.hibernate.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name="ITEMS")
public class Items1 {

        @Id
        @GeneratedValue(strategy=GenerationType.IDENTITY)
        @Column(name="id")
        private long id;

        @Column(name="item_id")
        private String itemId;
```

Most important point in above class is the `ManyToOne` annotation on Cart1 class variable and `JoinColumn` annotation to provide the column name for mapping.

That's it for one to many mapping in hibernate using annotation in model classes. Compare it with XML based configurations, you will find them very similar.

Let's write a test program and execute it.

Hibernate One To Many Mapping Annotation Example Test Program

Our test program is just like xml based configuration, we are just using the new classes for getting Hibernate Session and saving the model objects into database.

`HibernateOneToManyAnnotationMain.java`

```
package com.journaldev.hibernate.main;

import java.util.HashSet;
import java.util.Set;
```

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

import com.journaldev.hibernate.model.Cart1;
import com.journaldev.hibernate.model.Items1;
import com.journaldev.hibernate.util.HibernateAnnotationUtil;

public class HibernateOneToManyAnnotationMain {

        public static void main(String[] args) {

                Cart1 cart = new Cart1();
                cart.setName("MyCart1");

                Items1 item1 = new Items1("I10", 10, 1, cart);
                Items1 item2 = new Items1("I20", 20, 2, cart);
```

When we execute above hibernate one to many mapping annotation example test program, we get following output.

```
Hibernate Annotation Configuration loaded
Hibernate Annotation serviceRegistry created
Session created
Hibernate: insert into CART (name, total) values (?, ?)
Hibernate: insert into ITEMS (cart_id, item_id, item_total, quantity) values (?, ?, ?,
?)
Hibernate: insert into ITEMS (cart_id, item_id, item_total, quantity) values (?, ?, ?,
?)
Cart1 ID=7
item1 ID=9, Foreign Key Cart ID=7
item2 ID=10, Foreign Key Cart ID=7
Closing SessionFactory
```

That's all for hibernate one to many mapping, download the sample project from below link and do some more experiments.

Download Hibernate OneToMany Mapping Project

« PREVIOUS
Hibernate One to One Mapping Example Annotation

NEXT »
Hibernate Many To Many Mapping – Join Tables

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: HIBERNATE

Comments

Amit Goyal says
DECEMBER 27, 2017 AT 11:18 PM

well written…surely helps

Reply

Sameer says

JUNE 13, 2017 AT 5:56 AM

Hibernate: update ITEMS set cart_id=? where id=?

Hibernate: update ITEMS set cart_id=? where id=?

Why, Other columns in ITEMS table not getting updated?

How can you make sure that, other COLUMNs as in if ITEMS table had column like "Updated'

How can we update that as well?

Reply

Rabi says

MAY 27, 2017 AT 9:22 PM

Good article.Really helpful for beginners.

Reply

neha kapoor says

SEPTEMBER 24, 2016 AT 10:07 PM

Hi Pankaj,

Why is the output different in case of Annotation vs. XML conf.?

The difference in output is the 2 update statements (to update cart_id in items' table) that are missing in case of annotation test program.

P.S. – I assume there is no difference, it's just that u didn't paste the full output stack.

Thanks,

Neha

Reply

Pankaj says

SEPTEMBER 25, 2016 AT 11:27 PM

These logs are getting printed by Hibernate API, I think the log statements are different in both the cases. Don't reply on these, best is to check in DB if values are getting updated or not.

Reply

Aj says
SEPTEMBER 23, 2016 AT 11:02 AM
I saw some posts on blogs saying no-arg constructor is not mandatory for hibernate entities but in hibernate docs its written to use no-arg constructor, can someone please help me here?
Reply

Pankaj says
SEPTEMBER 23, 2016 AT 10:40 PM
No-Args constructor is required for hibernate. You can write a simple test program and check it out.
Reply

Sayed Masihullah says
OCTOBER 12, 2017 AT 3:32 AM
If you dont specify no-args constructor in your entity class then definitely it will take defulat constructor (no args constructor) but if you write only parameterized constructor in entity class u will get exception …hope u will be cleared
Reply

Anant says
JUNE 26, 2016 AT 6:46 AM
Hi Pankaj,
Here you are saving items and cart object, ideally we should save one object only which have refernce of another object as id.
Thanks,
Reply

hamed says
APRIL 12, 2018 AT 6:20 AM
have u any idea to handle it?
Reply

suresh says
DECEMBER 22, 2015 AT 6:21 AM

HI,
Its very handy tutorial but my problem is i got stuck in the error
not-null property references a null or transient value:
Reply

Mou TK says
SEPTEMBER 28, 2015 AT 8:03 AM
such a nice & easy tutorial , well explained
thanks
Reply

Akhib says
SEPTEMBER 10, 2015 AT 3:51 AM
thanks for the example.
the example is very good.but can any one help me.how to retrive update the data which we are saving in the above example
thank you.
Reply

Akshay says
JULY 29, 2015 AT 11:34 PM
Hi Pankaj,
I have downloaded example and try to implement similar one-to-many and many-to-one mapping however netbeans is throwing errors as below.
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'sessionFactory' defined in ServletContext resource [/WEB-INF/dispatcher-servlet.xml]: Invocation of init method failed; nested exception is org.hibernate.AnnotationException: mappedBy reference an unknown target entity property: one.form.invoicedetails.Invoicedetails.INVOICEID in one.form.invoice.Invoice.invoicedetails
Following are my form.java classes.
Invoice.java
@Entity(name = "Oneinvoice")
@Table(name = "Oneinvoive")
public class Invoice implements Serializable{
@Column(name="INVOICEID")
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int invoiceid;

@Column(name = “ORDERDATE”)
private Timestamp orderdate = new Timestamp(new Date().getTime());
@Column (name = “DESCRIPTION”)
private String description;
@OneToMany(mappedBy=”INVOICEID”)
private Set invoicedetails;
//setters and getters
Invoicedetails.java
@Entity(name=”Oneinvoicedetails”)
@Table(name=”Oneinvoicedetails”)
public class Invoicedetails implements Serializable{
@Column(name = “INVOICEDETAILSID”)
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int invoicedetailsid;
@ManyToOne
@JoinColumn(name = “INVOICEID”, nullable = false)
private Invoice invoiceid;
@Column(name = “QUANTITY”)
private int quantity;
@Column(name = “NOTE”)
private String note;
//setters and getters
Please help me.
Reply

kavya says
MAY 27, 2015 AT 3:41 AM
good example to understand one to many mapping
Reply

Bikash Mohapatra says
APRIL 23, 2015 AT 8:17 AM
great explanation, helps in comparing both xml and annotation based configurations….
Reply

sr says
SEPTEMBER 22, 2014 AT 12:50 AM

Hi Pankaj,
I am trying to implement struts2 with hibernate one to many mapping using set but i am not able to get values from jsp to action class in the child object and i am using model driven interface in the struts2 action
here is my jsp page
Department Name:
First Name:
Last Name:
Date of Birth:
Cell Phone:
in the action class in the getModel() object i am not getting child objects value , here department is parent and Employee is child object
Please help
Reply

Eric says
MAY 23, 2014 AT 12:59 AM
I can't find dependancy for >>$ import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
Netbeans cant find the matching items. What do I need to do?
Reply

Pankaj says
MAY 23, 2014 AT 9:17 AM
It's part of Hibernate jar, if you are using Maven then it should show. Please check your hibernate version too.
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *
Comment

Name *

Email *

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

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

HIBERNATE FRAMEWORK

Hibernate Tutorial

- Hibernate Example
- Hibernate SessionFactory
- Hibernate Session get load
- Hibernate Session save
- HQL Example
- Hibernate Criteria
- Hibernate SQL
- Hibernate Named Query
- Hibernate Log4J
- Hibernate Validator
- Hibernate Tomcat DataSource

Hibernate Mapping

- Hibernate One to One Mapping
- Hibernate One to Many Mapping
- Hibernate Many to Many Join Tables

Hibernate Caching

- Hibernate Cache
- Hibernate EHCache

Hibernate Integrations

- Hibernate Spring
- Hibernate Spring MVC
- Hibernate Struts 2
- Hibernate Primefaces
- Hibernate Primefaces Spring
- Hibernate SpringRoo Primefaces
- Hibernate JSF Spring

Miscellaneous

- Hibernate Tools Eclipse Plugin
- Hibernate Configuration Offline
- [Solved] No identifier specified
- Hibernate Program Not Terminating
- Access to DialectResolutionInfo
- get is not valid
- No CurrentSessionContext configured
- Hibernate Interview Questions

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial