Hibernate Many To Many Mapping – Join Tables

APRIL 2, 2018 BY PANKAJ — 8 COMMENTS

Today we will look into **Hibernate Many to Many Mapping** using XML and annotation configurations. Earlier we looked how to implement One To One and One To Many mapping in Hibernate.

Table of Contents [hide]

Hibernate Many to Many

Many-to-Many mapping is usually implemented in database using a **Join Table**. For example we can have `Cart` and `Item` table and `Cart_Items` table for many-to-many mapping. Every cart can have multiple items and every item can be part of multiple carts, so we have a many to many mapping here.

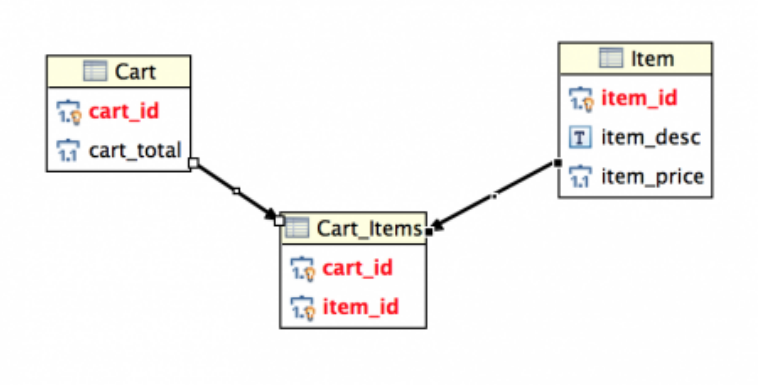
Hibernate Many to Many Mapping Database Setup

Below script can be used to create our many-to-many example database tables, these scripts are for MySQL database. If you are using any other database, you might need to make small changes to get it working.

```
DROP TABLE IF EXISTS `Cart_Items`;
DROP TABLE IF EXISTS `Cart`;
DROP TABLE IF EXISTS `Item`;

CREATE TABLE `Cart` (
  `cart_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `cart_total` decimal(10,0) NOT NULL,
  PRIMARY KEY (`cart_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `Item` (
  `item_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `item_desc` varchar(20) NOT NULL,
  `item_price` decimal(10,0) NOT NULL,
  PRIMARY KEY (`item_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Cart_Items` (
  `cart_id` int(11) unsigned NOT NULL,
  `item_id` int(11) unsigned NOT NULL,
  PRIMARY KEY (`cart_id`,`item_id`),
  CONSTRAINT `fk_cart` FOREIGN KEY (`cart_id`) REFERENCES `Cart` (`cart_id`),
```

Notice that Cart_Items table doesn't have any extra columns, actually it doesn't make much sense to have extra columns in many-to-many mapping table. But if you have extra columns, the implementation changes little bit and we will look into that in another post.

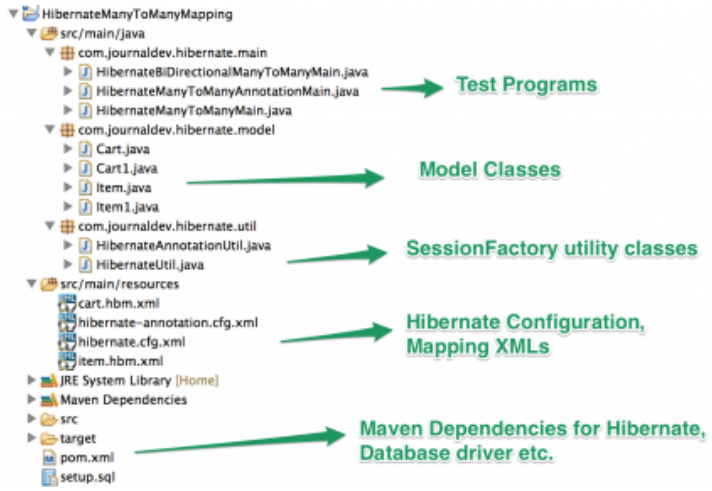
Below diagram shows the Entity relationship between these tables.



Our Database setup is ready now, let's move on to create the hibernate many-to-many mapping project.

Hibernate Many To Many Mapping Project Structure

Create a maven project in Eclipse or your favorite IDE, below image shows the structure and different components in the application.



We will first look into the XML based mapping implementations and then move over to using JPA annotations.

Hibernate Maven Dependencies

Our final pom.xml contains Hibernate dependencies with latest version **4.3.5.Final** and mysql driver dependencies.

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/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
```

```
	<groupId>com.journaldev.hibernate</groupId>
	<artifactId>HibernateManyToManyMapping</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>
```

Hibernate Many to Many XML Configuration Model Classes

Cart.java

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

import java.util.Set;

public class Cart {

	private long id;
	private double total;

	private Set<Item> items;

	public double getTotal() {
		return total;
	}

	public void setTotal(double total) {
		this.total = total;
	}

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

Item.java

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

import java.util.Set;

public class Item {

	private long id;
	private double price;
	private String description;

	private Set<Cart> carts;

	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public double getPrice() {
		return price;
```

Notice that Cart has set of Item and Item has set of Cart, this way we are implementing Bi-Directional associations. It means that we can configure it to save Item when we save Cart and vice versa.

For one-directional mapping, usually we have set in one of the model class. We will use annotations for one-directional mapping.

Hibernate Many To Many Mapping XML Configuration

Let's create hibernate many to many mapping xml configuration files for Cart and Item. We will implement bi-directional many-to-many mapping.

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="cart_total" />

                <set name="items" table="CART_ITEMS" fetch="select" cascade="all">
                        <key column="cart_id" />
                        <many-to-many class="Item" column="item_id" />
                </set>
        </class>

</hibernate-mapping>
```

Notice that set of items is mapped to CART_ITEMS table. Since Cart is the primary object, cart_id is the key and `many-to-many` mapping is using Item class item_id column.

`item.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="Item" table="ITEM">
                <id name="id" type="long">
                        <column name="item_id" />
                        <generator class="identity" />
                </id>
                <property name="description" type="string" column="item_desc" />

                <property name="price" type="double" column="item_price" />

                <set name="carts" table="CART_ITEMS" fetch="select" cascade="all">
                        <key column="item_id" />
                        <many-to-many class="Cart" column="cart_id" />
                </set>

        </class>
```

As you can see from above, the mapping is very similar to Cart mapping configurations.

Hibernate Configuration for XML Based Many to Many Mapping

Our hibernate configuration file looks like below.

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="item.hbm.xml" />
        </session-factory>
```

Hibernate SessionFactory Utility Class for XML Based Mapping

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()
```

It's a simple utility class that works as a factory for `SessionFactory`.

Hibernate Many To Many Mapping XML Configuration Test Program

Our hibernate many to many mapping setup is ready, let's test it out. We will write two program, one is to save Cart and see that Item and Cart_Items information is also getting saved. Another one to save item data and check that corresponding Cart and Cart_Items are saved.

`HibernateManyToManyMain.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.Item;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateManyToManyMain {

        //Saving many-to-many where Cart is primary
        public static void main(String[] args) {

                Item iphone = new Item();
                iphone.setPrice(100); iphone.setDescription("iPhone");
```

```
                Item ipod = new Item();
```

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

```
Hibernate Configuration loaded
Hibernate serviceRegistry created
Hibernate: insert into CART (cart_total) values (?)
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Hibernate: insert into CART (cart_total) values (?)
Before committing transaction
Hibernate: insert into CART_ITEMS (cart_id, item_id) values (?, ?)
Hibernate: insert into CART_ITEMS (cart_id, item_id) values (?, ?)
Hibernate: insert into CART_ITEMS (cart_id, item_id) values (?, ?)
Cart ID=1
Cart1 ID=2
Item1 ID=1
Item2 ID=2
```

Note that once the item data is saved through first cart, the item_id is generated and while saving second cart, it's not saved again.

Another important point to note is that many-to-many join table data is getting saved when we are committing the transaction. It's done for better performance incase we choose to rollback the transaction.

HibernateBiDirectionalManyToManyMain.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.Item;
import com.journaldev.hibernate.util.HibernateUtil;

public class HibernateBiDirectionalManyToManyMain {
```

```
        //Saving many-to-many where Item is primary
        public static void main(String[] args) {

                Item iphone = new Item();
                iphone.setPrice(100); iphone.setDescription("iPhone");
```

Output of above program is:

```
Hibernate Configuration loaded
Hibernate serviceRegistry created
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Hibernate: insert into CART (cart_total) values (?)
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Hibernate: insert into CART (cart_total) values (?)
Hibernate: insert into CART_ITEMS (item_id, cart_id) values (?, ?)
Hibernate: insert into CART_ITEMS (item_id, cart_id) values (?, ?)
Hibernate: insert into CART_ITEMS (item_id, cart_id) values (?, ?)
Cart ID=3
Cart1 ID=4
Item1 ID=3
Item2 ID=4
```

You can easily relate it to the earlier test program, since we have configured bi-directional mapping we can save Item or Cart and mapped data will get saved automatically.

Hibernate Many To Many Mapping Annotation

Now that we have seen how to configure **many-to-many** mapping using hibernate xml configurations, let's see an example of implementing it through annotations. We will implement one-directional many-to-many mapping using JPA annotations.

Hibernate Configuration XML File

Our annotations based hibernate configuration file looks like below.

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.Item1" />
        </session-factory>
```

Hibernate SessionFactory utility class

Our utility class to create SessionFactory looks like below.

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 Many to Many Mapping Annotation Model Classes

This is the most important part for annotation based mapping, let's first look at the Item table model class and then we will look into Cart table model class.

Item1.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.Table;

@Entity
@Table(name="ITEM")
public class Item1 {

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

	@Column(name="item_price")
	private double price;

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

Item1 class looks simple, there is no relational mapping here.

Cart1.java

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

import java.util.Set;

import javax.persistence.CascadeType;
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.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

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

	@Id
	@Column(name = "cart_id")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
```

Most important part here is the use of `ManyToMany` annotation and `JoinTable` annotation where we provide table name and columns to be used for many-to-many mapping.

Hibernate Many To Many Annotation Mapping Test Program

Here is a simple test program for our hibernate many to many mapping annotation based configuration.

`HibernateManyToManyAnnotationMain.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.Item1;
import com.journaldev.hibernate.util.HibernateAnnotationUtil;

public class HibernateManyToManyAnnotationMain {

	public static void main(String[] args) {
		Item1 item1 = new Item1();
		item1.setDescription("samsung"); item1.setPrice(300);
```

```
Item1 item2 = new Item1();
item2.setDescription("nokia"); item2.setPrice(200);
Cart1 cart = new Cart1();
cart.setTotal(500);
```

When we execute above program, it produces following output.

```
Hibernate Annotation Configuration loaded
Hibernate Annotation serviceRegistry created
Hibernate: insert into CART (cart_total) values (?)
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Hibernate: insert into ITEM (item_desc, item_price) values (?, ?)
Before committing transaction
Hibernate: insert into CART_ITEMS (cart_id, item_id) values (?, ?)
Hibernate: insert into CART_ITEMS (cart_id, item_id) values (?, ?)
Cart ID=5
Item1 ID=6
Item2 ID=5
```

It's clear that saving cart is also saving data into Item and Cart_Items table. If you will save only item information, you will notice that Cart and Cart_Items data is not getting saved.

That's all for Hibernate Many-To-Many mapping example tutorial, you can download the sample project from below link and play around with it to learn more.

Download Hibernate ManyToMany Mapping Project

« PREVIOUS
Hibernate One To Many Mapping Example Annotation

NEXT »
Hibernate Tools Eclipse Plugin

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

Shani says
NOVEMBER 16, 2016 AT 12:16 AM

what about update record and delete record in Manytomany

Reply

Steven says

AUGUST 3, 2016 AT 6:00 PM

Concise and to the point. Very useful for me as a refresher. Thanks!

Reply

roberto says

JANUARY 31, 2016 AT 11:28 PM

HOW TO SAVE WITH EXISTS CAR? PLEASE HELP ME

Reply

Pranish says

DECEMBER 8, 2015 AT 8:57 AM

How would I know where to declare the many-to-many annotation in which POJO class using annotations?

Reply

Ty Davis says

JULY 11, 2015 AT 1:57 PM

It is completely ridiculous how java over complicates the idea of writing/updating records to a DB.
EVERY java programmer should take a course in the IBM System i (AS/400) for a TRUE db management course.
Java has taken mundane, routine tasks and turned them into a doctoral thesis.

Reply

Sudheer says

AUGUST 4, 2014 AT 10:33 AM

Its very nice….I am one of the followers of journeldev.com…Thank u Pankaj..

Reply

Dener Miranda says
JULY 8, 2014 AT 8:55 AM
Hi Pankaj,
Why dont you use a SchemaUpdate to automatize the creation of SQL?
Reply

Pankaj says
JULY 8, 2014 AT 11:09 AM
Because I don't like that. I think DB creation should not be done through Hibernate, even in example.
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...

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