Relationships in Hibernate

# One To One

## Maven (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>one-to-one1</groupId>

<artifactId>one-to-one1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>one-to-one1</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

**<dependency>**

**<groupId>com.mchange</groupId>**

**<artifactId>c3p0</artifactId>**

**<version>0.9.5</version>**

**</dependency>**

**<dependency>**

**<groupId>org.hibernate</groupId>**

**<artifactId>hibernate-core</artifactId>**

**<version>4.3.7.Final</version>**

**</dependency>**

</dependencies>

</project>

## SQL

/\* Concept : Customer has an address \*/

/\*Drop the table\*/

**drop** **table** IF **EXISTS** customer;

**drop** **table** IF **EXISTS** address;

/\*Create the required tables\*/

**CREATE** **TABLE** IF **NOT** **EXISTS** address (

addressId **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

city **varchar**(255) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (addressId)

) ;

**CREATE** **TABLE** IF **NOT** **EXISTS** customer (

customerId **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

firstName **varchar**(255) **DEFAULT** **NULL**,

lastName **varchar**(255) **DEFAULT** **NULL**,

addressId **int**(11) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (customerId),

**FOREIGN** **KEY** (addressId) **REFERENCES** address (addressId)

) ;

/\*Select the created table\*/

**select** \* **from** customer;

**select** \* **from** address;

## Spring-Hibernate-Configuration

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />

**<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"***

**destroy-method=*"close"*>**

**<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />**

**<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />**

**<property name=*"user"* value=*"deba"* />**

**<property name=*"password"* value=*"deba"* />**

**<!-- these are C3P0 properties -->**

**<property name=*"acquireIncrement"* value=*"5"* />**

**<property name=*"initialPoolSize"* value=*"5"* />**

**<property name=*"minPoolSize"* value=*"5"* />**

**<property name=*"maxPoolSize"* value=*"20"* />**

<!-- <property name="maxIdleTime" value="${c3p0.maxIdleTime}" />

<property name="checkoutTimeout" value="${c3p0.checkoutTimeout}" />

<property name="preferredTestQuery" value="${c3p0.preferredTestQuery}" />

<property name="idleConnectionTestPeriod" value="${c3p0.idleConnectionTestPeriod}" />

<property name="maxIdleTimeExcessConnections" value="${c3p0.maxIdleTimeExcessConnections}" />

<property name="numHelperThreads" value="${c3p0.numHelperThreads}" />

<property name="unreturnedConnectionTimeout" value="${c3p0.unreturnedConnectionTimeout}" />

<property name="debugUnreturnedConnectionStackTraces" value="${c3p0.debugUnreturnedConnectionStackTraces}" />

<property name="testConnectionOnCheckin" value="${c3p0.testConnectionOnCheckin}" />

<property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}" />

<property name="acquireRetryDelay" value="${c3p0.acquireRetryDelay}" />

<property name="breakAfterAcquireFailure" value="${c3p0.breakAfterAcquireFailure}" /> -->

</bean>

**<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->**

**<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>**

**<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>**

**</bean>**

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

**<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>**

**<property name=*"dataSource"* ref=*"dataSource"* />**

**<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true**

**</value> </property> -->**

**<property name=*"hibernateProperties"*>**

**<value>**

**hibernate.id.new\_generator\_mappings=true,**

**hibernate.show\_sql=true**

**<!-- hibernate.hbm2ddl.auto=update -->**

**<!-- hibernate.current\_session\_context\_class=thread -->**

**</value>**

**</property>**

**<property name=*"annotatedClasses"*>**

**<list>**

**<value>com.ddlab.rnd.orm.entity.Customer</value>**

**<value>com.ddlab.rnd.orm.entity.Address</value>**

**</list>**

**</property>**

**</bean>**

</beans>

## Java Source Code

### Address.java

**package** com.ddlab.rnd.orm.entity;

**import** java.io.Serializable;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToOne;

**import** javax.persistence.Table;

**@Entity**

**@Table(name="address")**

**public** **class** Address **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -6494554948746566564L;

**@Id**

**@Column(name="addressId")**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private** **int** addressId;

@Column(name="city")

**private** String city;

**@OneToOne(mappedBy="address")**

private Customer customer;

**public** Address() {

}

**public** Address(String city) {

**this**.city = city;

}

**public** **int** getAddressId() {

**return** addressId;

}

**public** **void** setAddressId(**int** addressId) {

**this**.addressId = addressId;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** Customer getCustomer() {

**return** customer;

}

**public** **void** setCustomer(Customer customer) {

**this**.customer = customer;

}

}

### Customer.java

**package** com.ddlab.rnd.orm.entity;

**import** java.io.Serializable;

**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.OneToOne;

**import** javax.persistence.Table;

**@Entity**

**@Table(name="customer")**

**public** **class** Customer **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -282016336815027846L;

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**@Column(name="customerId")**

**private** **int** customerId;

@Column(name="firstName")

**private** String firstName;

@Column(name="lastName")

**private** String lastName;

**@OneToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name="addressId")**

**private Address address;**

**public** **int** getCustomerId() {

**return** customerId;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

### App.java

**package** com.ddlab.rnd.orm.entity;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App

{

**public** **static** **void** main( String[] args )

{

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory)context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Customer cust = **new** Customer();

cust.setFirstName("Deb");

cust.setLastName("Mishra");

Address address1 = **new** Address("Odisha");

cust.setAddress(address1);

session.save(cust);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**) transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

System.***out***.println("successfully saved into database");

}

}

# One To Many

## Maven (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>one-to-many</groupId>

<artifactId>one-to-many</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-hibernate</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

</project>

## SQL (ddl.sql)

**drop** **table** cart;

**CREATE** **TABLE** Cart(

cart\_id **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

total **decimal**(10,0) **NOT** **NULL**,

name **varchar**(10),

**PRIMARY** **KEY** (cart\_id)

);

**CREATE** **TABLE** Items (

id **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

cart\_id **int**(11) **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),

**FOREIGN** **KEY** (cart\_id) **REFERENCES** Cart (cart\_id)

);

## Spring-hibernate-Configuration (app-context.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />

**<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"***

**destroy-method=*"close"*>**

**<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />**

**<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />**

**<property name=*"user"* value=*"deba"* />**

**<property name=*"password"* value=*"deba"* />**

**<!-- these are C3P0 properties -->**

**<property name=*"acquireIncrement"* value=*"5"* />**

**<property name=*"initialPoolSize"* value=*"5"* />**

**<property name=*"minPoolSize"* value=*"5"* />**

**<property name=*"maxPoolSize"* value=*"20"* />**

<!-- <property name="maxIdleTime" value="${c3p0.maxIdleTime}" />

<property name="checkoutTimeout" value="${c3p0.checkoutTimeout}" />

<property name="preferredTestQuery" value="${c3p0.preferredTestQuery}" />

<property name="idleConnectionTestPeriod" value="${c3p0.idleConnectionTestPeriod}" />

<property name="maxIdleTimeExcessConnections" value="${c3p0.maxIdleTimeExcessConnections}" />

<property name="numHelperThreads" value="${c3p0.numHelperThreads}" />

<property name="unreturnedConnectionTimeout" value="${c3p0.unreturnedConnectionTimeout}" />

<property name="debugUnreturnedConnectionStackTraces" value="${c3p0.debugUnreturnedConnectionStackTraces}" />

<property name="testConnectionOnCheckin" value="${c3p0.testConnectionOnCheckin}" />

<property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}" />

<property name="acquireRetryDelay" value="${c3p0.acquireRetryDelay}" />

<property name="breakAfterAcquireFailure" value="${c3p0.breakAfterAcquireFailure}" /> -->

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

**<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>**

**<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>**

**</bean>**

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

**<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>**

**<property name=*"dataSource"* ref=*"dataSource"* />**

**<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true**

**</value> </property> -->**

**<property name=*"hibernateProperties"*>**

**<value>**

**hibernate.id.new\_generator\_mappings=true**

**<!-- hibernate.current\_session\_context\_class=thread -->**

**</value>**

**</property>**

**<property name=*"annotatedClasses"*>**

**<list>**

**<value>com.ddlab.spring.hibernate.Cart1</value>**

**<value>com.ddlab.spring.hibernate.Items1</value>**

**</list>**

**</property>**

**</bean>**

</beans>

## Java source code

### Cart.java

**package** com.ddlab.spring.hibernate;

**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")

**private** **double** total;

@Column(name="name")

**private** String name;

**@OneToMany(mappedBy="cart1")**

**private Set<Items1> items1;**

**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;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Set<Items1> getItems1() {

**return** items1;

}

**public** **void** setItems1(Set<Items1> items1) {

**this**.items1 = items1;

}

}

### Items.java

**package** com.ddlab.spring.hibernate;

**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;

@Column(name="item\_total")

**private** **double** itemTotal;

@Column(name="quantity")

**private** **int** quantity;

**@ManyToOne**

**@JoinColumn(name="cart\_id")**

**private Cart1 cart1;**

//Hibernate requires no-args constructor

**public** Items1(){}

**public** Items1(String itemId, **double** total, **int** qty, Cart1 c){

**this**.itemId=itemId;

**this**.itemTotal=total;

**this**.quantity=qty;

**this**.cart1=c;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getItemId() {

**return** itemId;

}

**public** **void** setItemId(String itemId) {

**this**.itemId = itemId;

}

**public** **double** getItemTotal() {

**return** itemTotal;

}

**public** **void** setItemTotal(**double** itemTotal) {

**this**.itemTotal = itemTotal;

}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

**public** Cart1 getCart1() {

**return** cart1;

}

**public** **void** setCart1(Cart1 cart1) {

**this**.cart1 = cart1;

}

}

### App.java

**package** com.ddlab.spring.hibernate;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

/\*\*

\* One to many

\*

\*/

**public** **class** App

{

**public** **static** **void** main( String[] args )

{

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory)context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

System.***out***.println(session);

Cart1 cart = **new** Cart1();

cart.setName("MyCart1");

Items1 item1 = **new** Items1("I10", 10, 1, cart);

Items1 item2 = **new** Items1("I20", 20, 2, cart);

Set<Items1> itemsSet = **new** HashSet<Items1>();

itemsSet.add(item1); itemsSet.add(item2);

cart.setItems1(itemsSet);

cart.setTotal(10\*1 + 20\*2);

Transaction tx = **null**;

**try**{

System.***out***.println("Session created");

//start transaction

tx = session.beginTransaction();

//Save the Model object

session.save(cart);

session.save(item1);

session.save(item2);

//Commit transaction

tx.commit();

System.***out***.println("Cart1 ID="+cart.getId());

System.***out***.println("item1 ID="+item1.getId()+", Foreign Key Cart ID="+item1.getCart1().getId());

System.***out***.println("item2 ID="+item2.getId()+", Foreign Key Cart ID="+item1.getCart1().getId());

}**catch**(Exception e){

System.***out***.println("Exception occured. "+e.getMessage());

e.printStackTrace();

}**finally**{

**if**(!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

# Many To One

## Maven (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>many-to-one1</groupId>

<artifactId>many-to-one1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>many-to-one1</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

</project>

## SQL (ddl.sql)

**drop** **table** if **exists** student;

**drop** **table** if **exists** address1;

**CREATE** **TABLE** IF **NOT** **EXISTS** address1 (

ADDRESS\_ID bigint(20) **NOT** **NULL** AUTO\_INCREMENT,

ADDRESS\_CITY **varchar**(255) **DEFAULT** **NULL**,

ADDRESS\_STATE **varchar**(50) **NOT** **NULL**,

ADDRESS\_STREET **varchar**(250) **NOT** **NULL**,

ADDRESS\_ZIPCODE **varchar**(10) **NOT** **NULL**,

**PRIMARY** **KEY** (ADDRESS\_ID)

);

**CREATE** **TABLE** IF **NOT** **EXISTS** student (

STUDENT\_ID bigint(20) **NOT** **NULL** AUTO\_INCREMENT,

STUDENT\_NAME **varchar**(100) **NOT** **NULL**,

ADDRESS\_ID bigint(20) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (STUDENT\_ID),

**FOREIGN** **KEY** (ADDRESS\_ID) **REFERENCES** address1 (ADDRESS\_ID)

);

**select** \* **from** ADDRESS1;

**select** \* **from** STUDENT;

## Spring-Hibernate-Consiguration (App-Context.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

**<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />**

**<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"***

**destroy-method=*"close"*>**

**<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />**

**<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />**

**<property name=*"user"* value=*"deba"* />**

**<property name=*"password"* value=*"deba"* />**

**<!-- these are C3P0 properties -->**

**<property name=*"acquireIncrement"* value=*"5"* />**

**<property name=*"initialPoolSize"* value=*"5"* />**

**<property name=*"minPoolSize"* value=*"5"* />**

**<property name=*"maxPoolSize"* value=*"20"* />**

<!-- <property name="maxIdleTime" value="${c3p0.maxIdleTime}" />

<property name="checkoutTimeout" value="${c3p0.checkoutTimeout}" />

<property name="preferredTestQuery" value="${c3p0.preferredTestQuery}" />

<property name="idleConnectionTestPeriod" value="${c3p0.idleConnectionTestPeriod}" />

<property name="maxIdleTimeExcessConnections" value="${c3p0.maxIdleTimeExcessConnections}" />

<property name="numHelperThreads" value="${c3p0.numHelperThreads}" />

<property name="unreturnedConnectionTimeout" value="${c3p0.unreturnedConnectionTimeout}" />

<property name="debugUnreturnedConnectionStackTraces" value="${c3p0.debugUnreturnedConnectionStackTraces}" />

<property name="testConnectionOnCheckin" value="${c3p0.testConnectionOnCheckin}" />

<property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}" />

<property name="acquireRetryDelay" value="${c3p0.acquireRetryDelay}" />

<property name="breakAfterAcquireFailure" value="${c3p0.breakAfterAcquireFailure}" /> -->

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>

<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>

</bean>

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

**<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>**

**<property name=*"dataSource"* ref=*"dataSource"* />**

**<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true**

**</value> </property> -->**

**<property name=*"hibernateProperties"*>**

**<value>**

**hibernate.id.new\_generator\_mappings=true**

**<!-- hibernate.hbm2ddl.auto=update -->**

**hibernate.show\_sql=true**

**hibernate.format\_sql**

**<!-- hibernate.current\_session\_context\_class=thread -->**

**</value>**

**</property>**

**<property name=*"annotatedClasses"*>**

**<list>**

**<value>com.ddlab.rnd.hibernate.Student</value>**

**<value>com.ddlab.rnd.hibernate.Address</value>**

**</list>**

**</property>**

**</bean>**

</beans>

## Java Source Code

### Address.java

**package** com.ddlab.rnd.hibernate;

**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 = "ADDRESS1")**

**public class Address {**

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "ADDRESS\_ID")**

**private long addressId;**

@Column(name = "ADDRESS\_STREET", nullable = **false**, length=250)

**private** String street;

@Column(name = "ADDRESS\_CITY")

**private** String city;

@Column(name = "ADDRESS\_STATE", nullable = **false**, length=50)

**private** String state;

@Column(name = "ADDRESS\_ZIPCODE", nullable = **false**, length=10)

**private** String zipcode;

**public** Address() {

}

**public** Address(String street, String city, String state, String zipcode) {

**this**.street = street;

**this**.city = city;

**this**.state = state;

**this**.zipcode = zipcode;

}

**public** **long** getAddressId() {

**return** **this**.addressId;

}

**public** **void** setAddressId(**long** addressId) {

**this**.addressId = addressId;

}

**public** String getStreet() {

**return** **this**.street;

}

**public** **void** setStreet(String street) {

**this**.street = street;

}

**public** String getCity() {

**return** **this**.city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getState() {

**return** **this**.state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** String getZipcode() {

**return** **this**.zipcode;

}

**public** **void** setZipcode(String zipcode) {

**this**.zipcode = zipcode;

}

}

### Student.java

**package** com.ddlab.rnd.hibernate;

**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.ManyToOne;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "STUDENT")**

**public class Student {**

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "STUDENT\_ID")**

**private long studentId;**

@Column(name = "STUDENT\_NAME", nullable = **false**, length = 100)

**private** String studentName;

**@ManyToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name="ADDRESS\_ID")**

**private Address studentAddress;**

**public** Student() {

}

**public** Student(String studentName, Address studentAddress) {

**this**.studentName = studentName;

**this**.studentAddress = studentAddress;

}

**public** **long** getStudentId() {

**return** **this**.studentId;

}

**public** **void** setStudentId(**long** studentId) {

**this**.studentId = studentId;

}

**public** String getStudentName() {

**return** **this**.studentName;

}

**public** **void** setStudentName(String studentName) {

**this**.studentName = studentName;

}

**public** Address getStudentAddress() {

**return** **this**.studentAddress;

}

**public** **void** setStudentAddress(Address studentAddress) {

**this**.studentAddress = studentAddress;

}

}

### App.java

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

/\*\*

\* Many to One

\* According to the relationship many students can have the same address.

\*/

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Address address = **new** Address("Balangir", "Balangir", "Odisha",

"767123");

// By using cascade=all option the address need not be saved

// explicitly when the student object is persisted the address will

// be automatically saved.

// session.save(address);

Student student1 = **new** Student("Deb", address);

Student student2 = **new** Student("John", address);

session.save(student1);

session.save(student2);

transaction.commit();

} **catch** (HibernateException e) {

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

}

# Many To Many

Authors publish Books, and Books have Authors. Any one Author can publish many Books, and any one Book can be published by many Authors, so this is why it is a many to many relationship.

Other examples of the many to many relationship are Students to Courses and Employees to Projects.

Hibernate @ManyToMany Bidirectional

## Maven (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>many-to-many</groupId>

<artifactId>many-to-many</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>many-to-many</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

</project>

## SQL (ddl.sql)

**drop** **table** IF **EXISTS** author\_book;

**drop** **table** IF **EXISTS** book;

**drop** **table** IF **EXISTS** author;

**CREATE** **TABLE** IF **NOT** **EXISTS** author (

AUTHOR\_ID bigint(20) **NOT** **NULL** AUTO\_INCREMENT,

author\_name **varchar**(255) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (AUTHOR\_ID)

);

**CREATE** **TABLE** IF **NOT** **EXISTS** book (

BOOK\_ID bigint(20) **NOT** **NULL** AUTO\_INCREMENT,

book\_name **varchar**(255) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (BOOK\_ID)

);

**CREATE** **TABLE** IF **NOT** **EXISTS** author\_book (

book\_id bigint(20) **NOT** **NULL**,

author\_id bigint(20) **NOT** **NULL**,

**PRIMARY** **KEY** (book\_id,author\_id),

**FOREIGN** **KEY** (book\_id) **REFERENCES** book (BOOK\_ID),

**FOREIGN** **KEY** (author\_id) **REFERENCES** author (AUTHOR\_ID)

);

**select** \* **from** book;

**select** \* **from** author;

**select** \* **from** author\_book;

## Spring-Hibernate-Configuration (app-Context.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

**<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />**

**<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"***

**destroy-method=*"close"*>**

**<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />**

**<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />**

**<property name=*"user"* value=*"deba"* />**

**<property name=*"password"* value=*"deba"* />**

**<!-- these are C3P0 properties -->**

**<property name=*"acquireIncrement"* value=*"5"* />**

**<property name=*"initialPoolSize"* value=*"5"* />**

**<property name=*"minPoolSize"* value=*"5"* />**

**<property name=*"maxPoolSize"* value=*"20"* />**

<!-- <property name="maxIdleTime" value="${c3p0.maxIdleTime}" />

<property name="checkoutTimeout" value="${c3p0.checkoutTimeout}" />

<property name="preferredTestQuery" value="${c3p0.preferredTestQuery}" />

<property name="idleConnectionTestPeriod" value="${c3p0.idleConnectionTestPeriod}" />

<property name="maxIdleTimeExcessConnections" value="${c3p0.maxIdleTimeExcessConnections}" />

<property name="numHelperThreads" value="${c3p0.numHelperThreads}" />

<property name="unreturnedConnectionTimeout" value="${c3p0.unreturnedConnectionTimeout}" />

<property name="debugUnreturnedConnectionStackTraces" value="${c3p0.debugUnreturnedConnectionStackTraces}" />

<property name="testConnectionOnCheckin" value="${c3p0.testConnectionOnCheckin}" />

<property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}" />

<property name="acquireRetryDelay" value="${c3p0.acquireRetryDelay}" />

<property name="breakAfterAcquireFailure" value="${c3p0.breakAfterAcquireFailure}" /> -->

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

**<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>**

**<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>**

**</bean>**

**<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->**

**<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>**

**<property name=*"dataSource"* ref=*"dataSource"* />**

**<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true**

**</value> </property> -->**

**<property name=*"hibernateProperties"*>**

**<value>**

**hibernate.id.new\_generator\_mappings=true,**

**hibernate.show\_sql=true**

**<!-- hibernate.hbm2ddl.auto=update -->**

**<!-- hibernate.current\_session\_context\_class=thread -->**

**</value>**

**</property>**

**<property name=*"annotatedClasses"*>**

**<list>**

**<value>com.ddlab.rnd.hibernate.Author</value>**

**<value>com.ddlab.rnd.hibernate.Book</value>**

**</list>**

**</property>**

**</bean>**

</beans>

## Java Source File

### Author.java

**package** com.ddlab.rnd.hibernate;

**import** java.util.HashSet;

**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.ManyToMany;

**import** javax.persistence.Table;

**@Entity**

**@Table(name="author")**

**public class Author**

{

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name="author\_id")**

**private Long authorId;**

@Column(name="author\_name")

**private** String authorName;

**@ManyToMany(cascade=CascadeType.*ALL*, mappedBy="authors")**

**private Set<Book> books = new HashSet<Book>();**

**public** Long getAuthorId() {

**return** authorId;

}

**public** **void** setAuthorId(Long authorId) {

**this**.authorId = authorId;

}

**public** String getAuthorName() {

**return** authorName;

}

**public** **void** setAuthorName(String authorName) {

**this**.authorName = authorName;

}

**public** Set<Book> getBooks() {

**return** books;

}

**public** **void** setBooks(Set<Book> books) {

**this**.books = books;

}

}

### Book.java

**package** com.ddlab.rnd.hibernate;

**import** java.util.HashSet;

**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="book")**

**public** **class** Book

{

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name="book\_id")**

**private Long bookId;**

@Column(name="book\_name")

**private** String bookName;

**@ManyToMany(cascade=CascadeType.*ALL*)**

**@JoinTable(name="author\_book", joinColumns=@JoinColumn(name="book\_id"), inverseJoinColumns=@JoinColumn(name="author\_id"))**

**private Set<Author> authors = new HashSet<Author>();**

**public** Long getBookId() {

**return** bookId;

}

**public** **void** setBookId(Long bookId) {

**this**.bookId = bookId;

}

**public** String getBookName() {

**return** bookName;

}

**public** **void** setBookName(String bookName) {

**this**.bookName = bookName;

}

**public** Set<Author> getAuthors() {

**return** authors;

}

**public** **void** setAuthors(Set<Author> authors) {

**this**.authors = authors;

}

}

### Test1.java

**package** com.ddlab.rnd.hibernate;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

/\*

\* Many to One

\*/

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Set<Book> booksSet = **new** HashSet<Book>();

Book book1 = **new** Book();

book1.setBookName("Horror");

booksSet.add(book1);

Book book2 = **new** Book();

book2.setBookName("Novel");

booksSet.add(book2);

Author author1 = **new** Author();

author1.setAuthorName("Deb");

author1.setBooks(booksSet);

book1.getAuthors().add(author1);

// session.save(author1);

// OR

session.save(book1);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**)

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

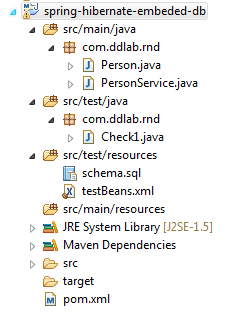
System.***out***.println("successfully saved into database");

}

}

Hibernate with Spring Embedded Database (H2 or HSQL)

# Project Structure



# Maven (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>spring-hibernate-embeded-db</groupId>

<artifactId>spring-hibernate-embeded-db</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-hibernate-embeded-db</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>3.0.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>3.0.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>3.0.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>3.6.9.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-annotations</artifactId>

<version>3.5.6-Final</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.5.4</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.16.1-GA</version>

<scope>runtime</scope>

</dependency>

<!-- without a slf4j bridge, you get no logging if anything goes wrong. -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.4</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.10</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>3.0.6.RELEASE</version>

<scope>test</scope>

</dependency>

<dependency> <!-- needed to get AOPs around the test cases -->

<groupId>cglib</groupId>

<artifactId>cglib-nodep</artifactId>

<version>2.2.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hsqldb</groupId>

<artifactId>hsqldb</artifactId>

<version>2.2.8</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

# Schema.sql (src/test/resources)

**drop** **table** Person if **exists**;

**create** **table** Person (id **INTEGER** IDENTITY, name **varchar**(50) **not** **null**);

# Spring-Hibernate Configuration(src/test/testBeans.xml)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config/>

<tx:annotation-driven />

<jdbc:embedded-database id=*"dataSource"*>

<jdbc:script location=*"classpath:schema.sql"* />

</jdbc:embedded-database>

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"* />

<property name=*"packagesToScan"* value=*"com.ddlab.rnd"* />

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.HSQLDialect</prop>

<!-- <prop key="hibernate.hbm2ddl.auto">create</prop> -->

<prop key=*"hibernate.show\_sql"*>true</prop>

</props>

</property>

</bean>

<bean id=*"transactionManager"*

class=*"org.springframework.orm.hibernate3.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"* />

</bean>

<bean id=*"personService"* class=*"com.ddlab.rnd.PersonService"*/>

</beans>

# Java Source files

## Person.java

**package** com.ddlab.rnd;

**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="Person")**

**public** **class** Person {

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**@Column(name="id")**

**private int id;**

@Column(name="name")

**private** String name;

**public** Person() {

}

**public** Person(String name) {

**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;

}

}

## PersonService.java

**package** com.ddlab.rnd;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**public** **class** PersonService {

**@Autowired**

**private SessionFactory sessionFactory;**

**public** **void** store(Person person) {

Session session = sessionFactory.openSession();

**try** {

session.beginTransaction();

session.save(person);

session.getTransaction().commit();

} **catch** (Exception e) {

e.printStackTrace();

session.getTransaction().rollback();

}

**finally** {

session.close();

}

}

**public** **int** getPersonList() {

**int** personList = 0;

Session session = sessionFactory.openSession();

**try** {

session.beginTransaction();

personList = session.createQuery("from Person").list().size();

} **catch** (Exception e) {

e.printStackTrace();

}

**finally** {

session.close();

}

**return** personList;

}

}

## Check1.java (src/test/java)

**package** com.ddlab.rnd;

**import** **static** org.junit.Assert.\*;

**import** org.junit.BeforeClass;

**import** org.junit.Test;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Check1 {

**private** **static** ApplicationContext *context* = **null**;

**private** **static** PersonService *service* = **null**;

@BeforeClass

**public** **static** **void** init() {

**new** Check1().doSetup();

}

**public** **void** doSetup() {

*context* = **new** ClassPathXmlApplicationContext("testBeans.xml");

*service* = (PersonService) *context*.getBean("personService");

System.***out***.println("Service :::"+*service*);

}

@Test

**public** **void** test1() {

Person p = **new** Person("Deb");

*service*.store(p);

*assertEquals*(1,*service*.getPersonList());

}

}