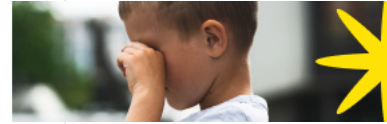


Who lost one or both parents during the Russian war in Ukraine



[Home](#) » [Enterprise Java](#) » [spring](#) » [JPA/ORM](#) » [Spring Hibernate Integration Example – Mysql and Maven Showcase](#)

ABOUT THEODORA FRAGKOULI



Theodora has graduated from Computer Engineering and Informatics Department in the University of Patras. She also holds a Master degree in Economics from the National and Technical University of Athens. During her studies she has been involved with a large number of projects ranging from programming and software engineering to telecommunications, hardware design and analysis. She works as a junior Software Engineer in the telecommunications sector where she is mainly involved with projects based on Java and Big Data technologies.

Spring Hibernate Integration Example – Mysql and Maven Showcase

Posted by: Theodora Fragkouli in JPA/ORM October 10th, 2013 2 Comments 4097 Views



In this tutorial we shall show you how to create a simple Spring Hibernate MySQL example. The Spring Framework supports integration with Hibernate for resource management, data access object (DAO) implementations and transaction strategies. Here we are using a simple Entity class that is mapped to a database table and we implement the basic CRUD (create- retrieve- update- delete) functionality to the database.

Our preferred development environment is Eclipse. We are using Eclipse Juno (4.2) version, along with Maven Integration plugin version 3.1.0. You can download Eclipse from [here](#) and Maven Plugin for Eclipse from [here](#). The installation of Maven plugin for Eclipse is out of the scope of this tutorial and will not be discussed. We are also using Spring version 3.2.3 and the JDK 7_u_21. The Hibernate version is 4.1.9, and the database used in the example is MySQL Database Server 5.6.

Let's begin.

Want to master Spring Framework ?

Subscribe to our newsletter and download the **Spring Framework Cookbook** right now!

In order to help you master the leading and innovative Java framework, we have compiled a kick-ass guide with all its major features and use cases! Besides studying them online you may download the eBook in PDF format!



Download NOW!

1. Create a new Maven project

Go to File -> Project ->Maven -> Maven Project.

JOIN US



With **1,240,600** unique visitors
500 authors w
placed among t
related sites ar
Constantly bein
lookout for part
encourage you
So If you have
unique and inte

content then you should check out our [Java Partners](#) program. You can also be a [guest author](#) for Java Code Geeks and hone your writing skills.

AD

ADVERTISEMENT

NEWSLETTER

117,635 insiders are already enjoying weekly updates and complimentary whitepapers!

Join them now to gain exclusive access to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Your email address

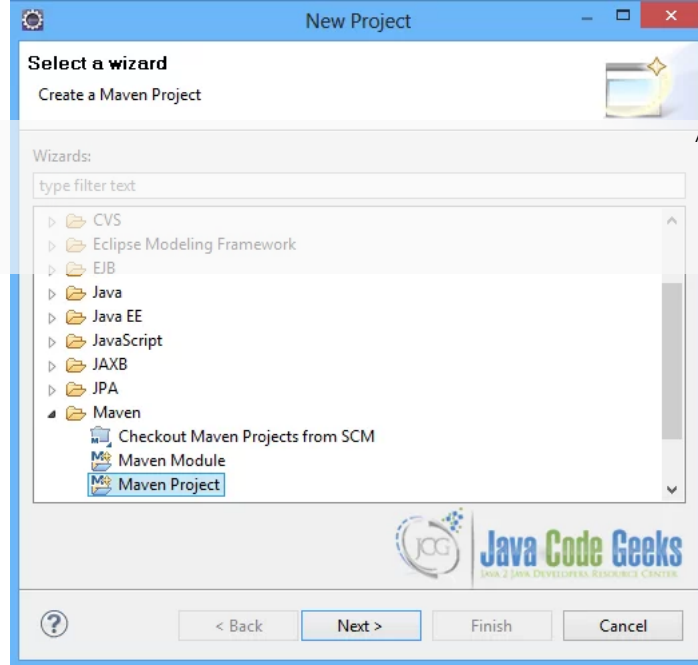
☒ Receive Java & Developer job alerts in your Area

☐ I have read and agree to the terms and conditions

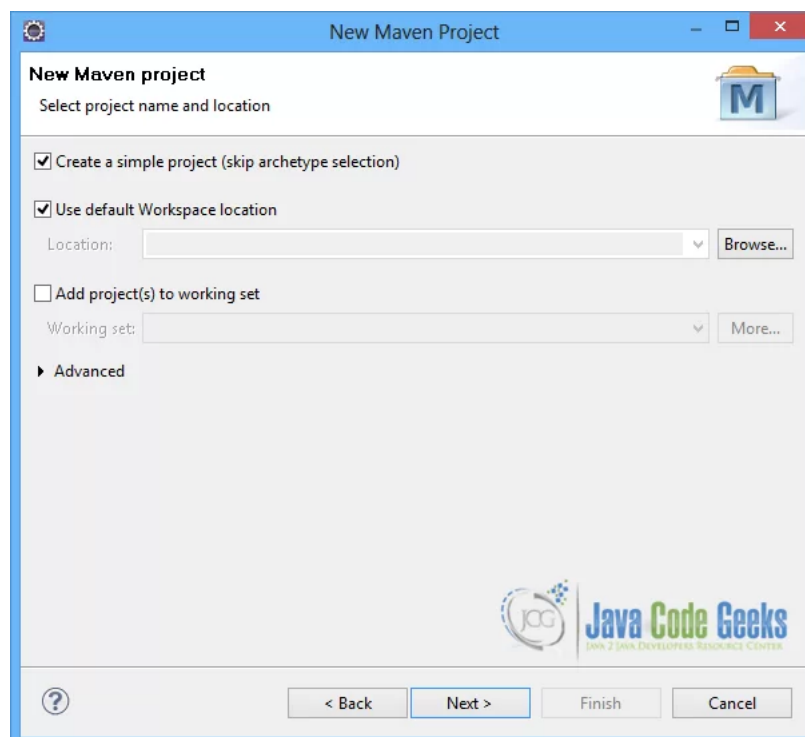
[Sign up](#)

AD

ADVERTISEMENT



In the "Select project name and location" page of the wizard, make sure that "Create a simple project (skip archetype selection)" option is **checked**, hit "Next" to continue with default values.



In the "Enter an artifact id" page of the wizard, you can define the name and main package of your project. We will set the "Group Id" variable to

```
"com.javacodegeeks.snippets.enterprise"
```

and the "Artifact Id" variable to

```
"springexample"
```

. The aforementioned selections compose the main project package as

```
"com.javacodegeeks.snippets.enterprise.springexample"
```

and the project name as

```
"springexample"
```

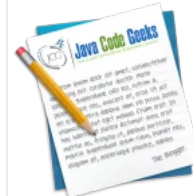
. Hit "Finish" to exit the wizard and to create your project.

ADVERTISEMENT

AD

ADVERTISEMENT

JOIN US



With **1,240,600** unique visitors and **500** authors who have placed among the top related sites around the world, we are constantly being looked out for partnerships. So if you have a unique and interesting content then you should check out our Java Code Geeks partners program. You can also be a guest author for Java Code Geeks and hone your writing skills.

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoying our weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Receive Java & Developer job alerts directly in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD



New Maven Project

Configure project

Artifact

Group Id: com.javacodegeeks.snippets.enterprise

Artifact Id: springexample

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

Browse...

Clear

Advanced

Java Code Geeks

?

< Back

Next >

Finish

Cancel

The Maven project structure is shown below:

It consists of the following folders:

- /src/main/java folder, that contains source files for the dynamic content of the application,
- /src/test/java folder contains all source files for unit tests,
- /src/main/resources folder contains configurations files,
- /target folder contains the compiled and packaged deliverables,
- the pom.xml is the project object model (POM) file. The single file that contains all project related configuration.

2. Add Spring 3.2.3 dependency

- Locate the "Properties" section at the "Overview" page of the POM editor and perform the following changes:
Create a new property with name **org.springframework.version** and value **3.2.3.RELEASE**.
- Navigate to the "Dependencies" page of the POM editor and create the following dependencies (you should fill the "GroupId", "Artifact Id" and "Version" fields of the "Dependency Details" section at that page):
Group Id : **org.springframework** Artifact Id : **spring-web** Version : **\${org.springframework.version}**

Alternatively, you can add the Spring dependencies in Maven's

pom.xml

file, by directly editing it at the "Pom.xml" page of the POM editor, as shown below:


pom.xml:

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"; xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
03     <modelVersion>4.0.0</modelVersion>
04     <groupId>com.javacodegeeks.snippets.enterprise</groupId>
05     <artifactId>springexample</artifactId>
06     <version>0.0.1-SNAPSHOT</version>
07
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-core</artifactId>
12             <version>${spring.version}</version>
```

AD

ADVERTISEMENT

JOIN US



With **1,240,600** unique visitors and **500** authors worldwide, we are placed among the top related sites around the globe. Constantly being a lookout for partnerships, we encourage you to join our unique and interesting content then you should check out our Java Code Geeks partners program. You can also be a guest author for Java Code Geeks and hone your writing skills.

AD

ADVERTISEMENT

NEWSLETTER

117,635 insiders are already enjoying weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies!

Email address:

Receive Java & Developer job alerts in your Area

I have read and agree to the terms and conditions

AD

ADVERTISEMENT

```

14     </dependency>
15     <dependency>
16         <groupId>org.springframework</groupId>
17         <artifactId>spring-context</artifactId>
18         <version>${spring.version}</version>
19     </dependency>
20 </dependencies>
21
22 <properties>
23     <spring.version>3.2.3.RELEASE</spring.version>
24 </properties>
25 </project>

```

ADVERTISEMENT

As you can see Maven manages library dependencies declaratively. A local repository is created (by default under {user_home}\.m2 folder) and all required libraries are downloaded and placed there from public repositories. Furthermore intra – library dependencies are automatically resolved and manipulated.

3. Add Hibernate and MySql dependencies

The Hibernate and MySql-connector dependencies are added, along with the org.apache.commons.dbcp package, that provides database Connection Pool API. We also need the spring-orm package and the javax.persistence api.

pom.xml

```

01 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
03     <modelVersion>4.0.0</modelVersion>
04     <groupId>com.javacodegeeks.snippets.enterprise</groupId>
05     <artifactId>springexample</artifactId>
06     <version>0.0.1-SNAPSHOT</version>
07
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-core</artifactId>
12             <version>${spring.version}</version>
13         </dependency>
14
15         <dependency>
16             <groupId>org.springframework</groupId>
17             <artifactId>spring-context</artifactId>
18             <version>${spring.version}</version>
19         </dependency>
20
21         <dependency>
22             <groupId>org.springframework</groupId>
23             <artifactId>spring-orm</artifactId>
24             <version>${spring.version}</version>
25         </dependency>
26
27         <dependency>
28             <groupId>commons-dbcp</groupId>
29             <artifactId>commons-dbcp</artifactId>
30             <version>1.2.2</version>
31         </dependency>
32
33         <dependency>
34             <groupId>javax.persistence</groupId>
35             <artifactId>persistence-api</artifactId>
36             <version>1.0</version>
37         </dependency>
38
39         <dependency>
40             <groupId>org.hibernate</groupId>
41             <artifactId>hibernate-core</artifactId>
42             <version>4.1.9.Final</version>
43         </dependency>
44
45         <dependency>
46             <groupId>mysql</groupId>
47             <artifactId>mysql-connector-java</artifactId>
48             <version>5.1.6</version>
49         </dependency>
50     </dependencies>
51     <properties>
52         <spring.version>3.2.3.RELEASE</spring.version>
53     </properties>
54 </project>

```

4. The entity class

Employee.java

class is a class with three properties. It uses the

javax.persistence

annotations to be mapped to a table,

EMPLOYEE

in the database. In particular, the

@Entity

annotation specifies that the class is an entity. The

@Table

annotation specifies the primary table for the annotated entity. The

@Column

ADVERTISEMENT

AD

JOIN US



With **1,240,600** unique visitors and **500** authors worldwide, we are placed among the top related sites around the globe. Constantly being ranked by Google, we encourage you to look out for our unique and interesting content then you should check out our Java Code Geeks partners program. You can also be a guest author for Java Code Geeks and hone your writing skills.

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoying our weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Receive Java & Developer job alerts directly in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD

annotation is used to specify a mapped column for the persistent field, whereas the

```
@Id
```

annotation specifies the primary key field of the entity.

[Employee.java](#)

ADVERTISEMENT

```
01 package com.javacodegeeks.snippets.enterprise.model;
02
03 import javax.persistence.Column;
04 import javax.persistence.Entity;
05 import javax.persistence.Id;
06 import javax.persistence.Table;
07
08 @Entity
09 @Table(name = "EMPLOYEE")
10 public class Employee {
11
12     @Id
13     @Column(name = "ID", nullable = false)
14     private String id;
15
16     @Column(name = "NAME", nullable = false)
17     private String name;
18
19     @Column(name = "AGE", nullable = false)
20     private long age;
21
22     public Employee() {
23     }
24
25     public String getId() {
26         return id;
27     }
28
29     public void setId(String id) {
30         this.id = id;
31     }
32
33     public String getName() {
34         return name;
35     }
36
37     public void setName(String name) {
38         this.name = name;
39     }
40
41     public long getAge() {
42         return age;
43     }
44
45     public void setAge(long age) {
46         this.age = age;
47     }
48
49 }
```

5. The DAO class

The Data Access Object implemented to interact with the database uses Hibernate data access technology. It is the

```
EmployeeDAOImpl.java
```

class. It uses the

```
@Repository
```

annotation, to guarantee that the Data Access Object (DAO) provides exception translation. When using Hibernate, we must decide how to handle the native exception classes. The DAO throws a subclass of a

```
HibernateException
```

, that is a run-time exception and does not have to be declared or caught. We may also deal with

```
IllegalArgumentException
```

and

```
IllegalStateException
```

. This means that callers can only treat exceptions as generally fatal, unless they want to depend on Hibernate's own exception structure. Spring enables exception translation to be applied transparently through the

```
@Repository
```

annotation.

The DAO uses the Hibernate

```
SessionFactory
```

that provides Sessions to access the Database. It gets it as bean reference from the Spring IoC container. All the methods implemented in the DAO get

```
Session
```

instances by using the

```
getCurrentSession()
```

method of

```
SessionFactory
```

AD

ADVERTISEMENT

JOIN US



With **1,240,600** unique visitors and **500** authors worldwide, we are placed among the top related sites around the globe. Constantly being a lookout for partnerships, we encourage you to join our unique and interesting

content then you should check out our Java Code Geeks partners program. You can also be a guest writer for Java Code Geeks and hone your writing skills.

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoying our weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Receive Java & Developer job alerts directly in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD



The

SessionFactory

is injected using the

@Autowired

ADVERTISEMENT

annotation.

The basic CRUD methods implemented here use the

persist(Object object)

,

get(Class clazz, Serializable id)

,

update(Object object)

and

delete(Object object)

API methods of

Session

to create, retrieve, update and delete an object from the database.

EmployeeDAOImpl.java

```
01 package com.javacodegeeks.snippets.enterprise.dao;
02
03 import org.hibernate.SessionFactory;
04 import org.springframework.beans.factory.annotation.Autowired;
05 import org.springframework.stereotype.Repository;
06
07 import com.javacodegeeks.snippets.enterprise.model.Employee;
08
09 @Repository("employeeDAO")
10 public class EmployeeDAOImpl implements EmployeeDAO {
11
12     @Autowired
13     private SessionFactory sessionFactory;
14
15     @Override
16     public void persistEmployee(Employee employee) {
17         sessionFactory.getCurrentSession().persist(employee);
18     }
19
20     @Override
21     public Employee findEmployeeById(String id) {
22         return (Employee) sessionFactory.getCurrentSession().get(Employee.class, id);
23     }
24
25     @Override
26     public void updateEmployee(Employee employee) {
27         sessionFactory.getCurrentSession().update(employee);
28     }
29
30     @Override
31     public void deleteEmployee(Employee employee) {
32         sessionFactory.getCurrentSession().delete(employee);
33     }
34
35 }
36 }
```

The interface of

EmployeeDAOImpl.java

is shown below:

EmployeeDAO.java

```
01 package com.javacodegeeks.snippets.enterprise.dao;
02
03 import com.javacodegeeks.snippets.enterprise.model.Employee;
04
05 public interface EmployeeDAO {
06
07     void persistEmployee(Employee employee);
08
09     Employee findEmployeeById(String id);
10
11     void updateEmployee(Employee employee);
12
13     void deleteEmployee(Employee employee);
14
15 }
```

6. The Service class

The

EmployeeDAOImpl.java


class is injected in the

EmployeeServiceImpl.java

ADVERTISEMENT

AD

JOIN US



With **1,240,600** unique visitors and **500** authors worldwide, we are placed among the top related sites around the globe. Constantly being ranked by Google, we are a lookout for partners to encourage you to join. So if you have a unique and interesting content then you should check out our Java Code Geeks partners program. You can also be a guest author for Java Code Geeks and hone your writing skills.

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoying weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Receive Java & Developer job alerts directly in your Area

☐ I have read and agree to the terms and conditions

ADVERTISEMENT

AD

AD

class. Thus, in the basic implementation here, the DAO methods are invoked to perform the basic interaction with the database. The

EmployeeServiceImpl.java

class is annotated with the

@Service

ADVERTISEMENT

annotation, dictating that it is a Spring Bean and thus allowing Spring to auto-detect it.

The

@Transactional

annotation is placed before the methods, to denote that a transaction is created when each method is invoked. The transaction will be configured in Spring configuration file.

EmployeeServiceImpl.java

```
01 package com.javacodegeeks.snippets.enterprise.service;
02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.stereotype.Service;
05 import org.springframework.transaction.annotation.Transactional;
06
07 import com.javacodegeeks.snippets.enterprise.dao.EmployeeDAO;
08 import com.javacodegeeks.snippets.enterprise.model.Employee;
09
10 @Service("employeeService")
11 public class EmployeeServiceImpl implements EmployeeService{
12
13     @Autowired
14     EmployeeDAO employeeDAO;
15
16     @Override
17     @Transactional
18     public void persistEmployee(Employee employee) {
19         employeeDAO.persistEmployee(employee);
20     }
21
22
23     @Override
24     @Transactional
25     public void updateEmployee(Employee employee) {
26         employeeDAO.updateEmployee(employee);
27     }
28
29     @Override
30     @Transactional
31     public Employee findEmployeeById(String id) {
32         return employeeDAO.findEmployeeById(id);
33     }
34
35     @Override
36     @Transactional
37     public void deleteEmployee(Employee employee) {
38         employeeDAO.deleteEmployee(employee);
39     }
40
41 }
42 }
```

The interface of

EmployeeService.java

class is shown below:

EmployeeService.java

```
01 package com.javacodegeeks.snippets.enterprise.service;
02
03 import com.javacodegeeks.snippets.enterprise.model.Employee;
04
05 public interface EmployeeService {
06
07     void persistEmployee(Employee employee);
08
09     Employee findEmployeeById(String id);
10
11     void updateEmployee(Employee employee);
12
13     void deleteEmployee(Employee employee);
14 }
```

7. Configure Spring Beans

The

applicationContext.xml

file shown below defines and configures all the beans needed for the interaction with the database.

First of all, since we are using Spring beans, we must use the

<context:component-scan

> element to define where the beans are, so that the IOC container will detect them.

We also use the

<tx:annotation-driven/>

element, so that Spring is @Transactional-aware and can detect the

AD

ADVERTISEMENT



JOIN US



With **1,240,600** unique visitors a **500** authors w placed among t related sites ar Constantly bein lookout for part encourage you So If you have a unique and inter content then you should check out our J partners program. You can also be a gu for Java Code Geeks and hone your writ

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoy weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java as well as insights about Android, Scala, Groovy and other related technologies!

Email address:

Receive Java & Developer job alerts in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD



@Transactional

annotations to configure the appropriate beans with transactional behavior.

In the `datasource`

bean the `DataSource`

is defined. Spring obtains a connection to the database through a `DataSource`

. The properties to be configured here are the `driverClassName`

, the `url`

to the database and the `username`

and `password`

for the connection to the database.

In the `sessionFactory`

bean we must define the `SessionFactory`

class. The `SessionFactory`

class is a thread-safe object that is instantiated once to serve the entire application. The `SessionFactory`

is used to create Sessions. A Session is used to get a physical connection with a database. The Session object is instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The class that implements the `sessionFactory`

is the `org.springframework.orm.hibernate4.LocalSessionFactoryBean`

class. We can configure the properties this class provides in the bean definition. In the `datasource`

property, that is a reference to the `DataSource`

we set the `DataSource` to be used by the `SessionFactory`. In the `annotatedClasses`

property we must specify annotated entity classes to register with this Hibernate `SessionFactory`. The `Employee`

class is the value of this property. The `org.springframework.orm.hibernate4.LocalSessionFactoryBean`

class also provides a `hibernateProperties`

property to configure. Here we can configure all properties provided by Hibernate. For example, JDBC Properties, Hibernate Configuration Properties, Cache and Transaction Properties and SQL dialects. Here we have set two properties. The `hibernate.dialect`

property is set to `MySQL`, and the `hibernate.show_sql`

is set to `true` so that the queries implemented are printed.

Last but not least, the `transactionManager`

AD

ADVERTISEMENT



JOIN US



With **1,240,600** unique visitors and **500** authors who have been placed among the top 100 related sites around the world. Constantly being looked out for partnerships, encourage you to join. So If you have a unique and interesting content then you should check out our **Java Code Geeks** partners program. You can also be a **guest author** for Java Code Geeks and hone your writing skills.

AD

ADVERTISEMENT

NEWSLETTER

117,635 insiders are already enjoying our weekly updates and complimentary whitepapers! **Join them now** to gain **exclusive access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

Receive Java & Developer job alerts in your Area

I have read and agree to the terms and conditions

AD

ADVERTISEMENT



bean is defined. The class to implement the transaction is the

```
org.springframework.orm.hibernate4.HibernateTransactionManager
```

. The bean has a property named

```
sessionFactory
```

ADVERTISEMENT

, whose value is a reference to the

```
sessionFactory
```

bean.

[applicationContext.xml](#)

```
01 <beans xmlns="http://www.springframework.org/schema/beans"
02     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/sche
03     xmlns:aop="http://www.springframework.org/schema/aop" xmlns:context="http://www.springframework.or
04     xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springframework.org/sch
05     xmlns:task="http://www.springframework.org/schema/task"
06     xsi:schemaLocation="http://www.springframework.org/schema/aop http://www.springframework.org/sche
07
08     <context:component-scan base-package="com.javacodegeeks.snippets.enterprise.*" />
09
10     <tx:annotation-driven/>
11
12     <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
13         <property name="driverClassName" value="com.mysql.jdbc.Driver" />
14         <property name="url" value="jdbc:mysql://localhost:3306/test" />
15         <property name="username" value="root" />
16         <property name="password" value="root" />
17     </bean>
18
19     <bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
20         <property name="dataSource" ref="dataSource"></property>
21         <property name="annotatedClasses">
22             <list>
23                 <value>com.javacodegeeks.snippets.enterprise.model.Employee</value>
24             </list>
25         </property>
26         <property name="hibernateProperties">
27             <props>
28                 <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
29                 <prop key="hibernate.show_sql">true</prop>
30             </props>
31         </property>
32     </bean>
33
34     <bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager"
35         p:sessionFactory-ref="sessionFactory">
36     </bean>
37 </beans>
```

8. Run the Application

In

```
App.java
```

class we load the

```
applicationContext.xml
```

file. We create an

```
Employee
```

object and use the CRUD methods to interact with the database.

[App.java](#)

```
01 package com.javacodegeeks.snippets.enterprise;
02
03 import org.springframework.context.ConfigurableApplicationContext;
04 import org.springframework.context.support.ClassPathXmlApplicationContext;
05
06 import com.javacodegeeks.snippets.enterprise.model.Employee;
07 import com.javacodegeeks.snippets.enterprise.service.EmployeeService;
08
09 public class App {
10
11     public static void main(String[] args) {
12         System.out.println("load context");
13         ConfigurableApplicationContext context = new ClassPathXmlApplicationContext("applicationContex
14         Employee em = new Employee();
15         em.setId("123");
16         em.setName("John");
17         em.setAge(35);
18         EmployeeService emService = (EmployeeService) context.getBean("employeeService");
19         emService.persistEmployee(em);
20         System.out.println("Updated age : " + emService.findEmployeeById("123").getAge());
21         em.setAge(32);
22         emService.updateEmployee(em);
23         System.out.println("Updated age : " + emService.findEmployeeById("123").getAge());
24         emService.deleteEmployee(em);
25         context.close();
26     }
27 }
28 }
```

When you run the application, you will see the sql queries in the output. You will also see the the age of the first employee and the age of the updated employee.

AD

ADVERTISEMENT

JOIN US



content then you should check out our J partners program. You can also be a gu for Java Code Geeks and hone your writ

With **1,240,600** unique visitors a **500** authors w placed among t related sites ar Constantly bein lookout for part encourage you So If you have a unique and inter

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoy weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java as well as insights about Android, Scala, Groovy and other related technologies!

Email address:

Receive Java & Developer job alerts in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD

```

Hibernate: insert into EMPLOYEE (AGE, NAME, ID) values (?, ?, ?)
Hibernate: select employee0_.ID as ID0_0_, employee0_.AGE as AGE0_0_, employee0_.NAME as NAME0_0_ from EMPLOYEE employee0_
Persisted age :35
ADVERTISMENT
Hibernate: update EMPLOYEE set AGE=?, NAME=? where ID=?
Hibernate: select employee0_.ID as ID0_0_, employee0_.AGE as AGE0_0_, employee0_.NAME as NAME0_0_ from EMPLOYEE employee0_
Updated age :32
Hibernate: select employee_.ID, employee_.AGE as AGE0_, employee_.NAME as NAME0_ from EMPLOYEE employee_ where employee_.I
Hibernate: delete from EMPLOYEE where ID=?

```

This was an example of Spring Hibernate and Mysql integration.

Download the Eclipse project of this tutorial : SpringHibernateMysqlMavenExample.zip

Tagged with: HIBERNATE



Do you want to know how to develop your skillset to become a **Java Rockstar?**



Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

Email address:

- ☒ Receive Java & Developer job alerts in your Area
☐ I have read and agree to the terms & conditions

Sign up

LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS

✉ Subscribe ▼



Join the discussion

B I U S

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

2 COMMENTS



Oldest ▼



Mishu ⌚ 4 years ago

AD

JOIN US



With **1,240,600** unique visitors and **500** authors worldwide, we are placed among the top related sites around the globe. Constantly being a lookout for partnerships, we encourage you to encourage you to join our unique and interesting

content then you should check out our Java Code Geeks partners program. You can also be a guest author for Java Code Geeks and hone your writing skills.

ADVERTISEMENT

AD

NEWSLETTER

117,635 insiders are already enjoying our weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

Email address:

☒ Receive Java & Developer job alerts in your Area
☐ I have read and agree to the terms & conditions

ADVERTISEMENT

AD



while executing above application it says: “No bean named ‘employeeService’ is defined”.

Kindly help !

+ 0 – ➔ Reply



Sid ⌚ 3 years ago

| ➔ Reply to [Mishu](#)

You can add a @Component or @Bean to the employeeService class, to wire it as a bean.

+ 0 – ➔ Reply

ADVERTISEMENT

AD

ADVERTISEMENT

JOIN US

KNOWLEDGE BASE

Courses

Minibooks

News

Resources

Tutorials

THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

HALL OF FAME

Android Alert Dialog Example

Android OnClickListener Example

How to convert Character to String and a String to Character Array in Java

Java Inheritance example

Java write to File Example

java.io.FileNotFoundException – How to solve File Not Found Exception

java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception

java.lang.NoClassDefFoundError – How to solve No Class Def Found Error

JSON Example With Jersey + Jackson

Spring JdbcTemplate Example

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

Examples Java Code Geeks and all content copyright © 2010-2023, Exelixis Media P.C. | [Terms of Use](#) | [Privacy Policy](#) | [Contact](#)



117,635 insiders are already enjoy weekly updates and complimentary whitepapers!

Join them now to gain **exclusive access** to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies!

Email address:

Receive Java & Developer job alerts in your Area

I have read and agree to the terms and conditions

ADVERTISEMENT

AD

