Spring Data JPA One To Many Foreign Key Example

June 20, 2017 by **javainterviewpoint** — **Leave a Comment**

In this **Spring Data JPA One To Many** article, we will learn how to achieve **One To Many Mapping** using Spring Data JPA. In this approach, we will re-use the same two tables which we used in our previous **Spring Data JPA One To One Example**.

Creating table

query in the query editor to get the table created.

```
CREATE TABLE
"EMPLOYEE"
 ( "EMP_ID"
NUMBER(10,0) NOT
NULL ENABLE,
 "NAME"
VARCHAR2(255 CHAR),
 PRIMARY KEY
("EMP_ID")
 );

 CREATE TABLE
"EMPLOYEE_ADDRESS"
 (
 "ADDR_ID"
NUMBER(10,0) NOT
NULL ENABLE,
 "EMP_ID"
NUMBER(10,0) NOT
NULL ENABLE,
 "STREET"
VARCHAR2(255 CHAR),
 "CITY"
VARCHAR2(255 CHAR),
 "STATE"
VARCHAR2(255 CHAR),
 "COUNTRY"
VARCHAR2(255 CHAR),
```

```
("EMP_ID")
REFERENCES EMPLOYEE
("EMP_ID")
 );
```

Folder Structure:

1. Create a simple *Maven Project* **"SpringDataJPA"** and create a package for our source files "***com.javainterviewpoint***" under **src/main/java**
2. Now add the following dependency in the **POM.xml**

```
<project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
che.org/maven-
v4_0_0.xsd">
 <groupId>com.ja
vainterviewpoint
</groupId>
 <artifactId>Spr
ingJPA</artifact
Id>
 <packaging>jar<
/packaging>
 <version>0.0.1-
SNAPSHOT</versio
n>
 <name>SpringJPA
Maven
Webapp</name>
 <url>http://mav
en.apache.org</u
rl>
 <properties>

<hibernate.versi
on>4.2.0.Final</
hibernate.versio
n>

<spring.version>
4.3.5
RELEASE</spring.
version>
 </properties>

 <dependencies>
    <!-- DB
```

```
<groupId>org.hib
ernate.common</g
roupId>

<artifactId>hibe
rnate-commons-
annotations</art
ifactId>

<version>4.0.5.F
inal</version>

</dependency>
    <dependency>

<groupId>org.hib
ernate</groupId>

<artifactId>hibe
rnate-
entitymanager</a
rtifactId>

<version>4.1.9.F
inal</version>

</dependency>
    <dependency>

<groupId>commons
-dbcp</groupId>

<artifactId>comm
```

```
rsion>

</dependency>
    <dependency>

<groupId>javassi
st</groupId>

<artifactId>java
ssist</artifactI
d>

<version>3.12.1.
GA</version>

</dependency>
    <dependency>

<groupId>org.hib
ernate.javax.per
sistence</groupI
d>

<artifactId>hibe
rnate-jpa-2.0-
api</artifactId>

<version>1.0.1.F
inal</version>

</dependency>
    <dependency>

<groupId>org.spr
```

```
jpa</artifactId>

<version>1.11.3.
RELEASE</version
>

</dependency>
    <dependency>

<groupId>com.ora
cle</groupId>

<artifactId>ojdb
c14</artifactId>

<version>11.2.0<
/version>

</dependency>
    <dependency>

<groupId>org.hib
ernate</groupId>

<artifactId>hibe
rnate-
core</artifactId
>

<version>4.1.9.F
inal</version>

</dependency>
```

```
ingframework</gr
oupId>

<artifactId>spri
ng-
tx</artifactId>

<version>4.3.5.R
ELEASE</version>

</dependency>
    <dependency>

<groupId>org.spr
ingframework</gr
oupId>

<artifactId>spri
ng-
context</artifac
tId>

<version>4.3.5.R
ELEASE</version>

</dependency>
    <dependency>

<groupId>org.spr
ingframework</gr
oupId>

<artifactId>spri
ng-
```

```
</dependency>
    <dependency>

<groupId>org.spr
ingframework</gr
oupId>

<artifactId>spri
ng-
core</artifactId
>

<version>4.3.5.R
ELEASE</version>

</dependency>
    <dependency>

<groupId>org.spr
ingframework</gr
oupId>

<artifactId>spri
ng-
tx</artifactId>

<version>4.3.5.R
ELEASE</version>

</dependency>
    <dependency>

<groupId>org.spr
```

```
jdbc</artifactId
>

<version>4.3.5.R
ELEASE</version>

</dependency>
    <dependency>

<groupId>org.spr
ingframework</gr
oupId>

<artifactId>spri
ng-
orm</artifactId>

<version>3.2.5.R
ELEASE</version>

</dependency>

    <!-- CGLIB
is required to
process
@Configuration
classes -->
    <dependency>

<groupId>cglib</
groupId>

<artifactId>cgli
b</artifactId>
```

```
    <!-- Servlet
API and JSTL -->
    <dependency>

<groupId>javax.s
ervlet</groupId>

<artifactId>java
x.servlet-
api</artifactId>

<version>3.0.1</
version>

<scope>provided<
/scope>

</dependency>
    <dependency>

<groupId>jstl</g
roupId>

<artifactId>jstl
</artifactId>

<version>1.2</ve
rsion>

</dependency>

    <!-- Test --
>
```

```
<artifactId>junit</artifactId>

<version>4.7</version>

<scope>test</scope>

</dependency>
    <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>4.3.5.RELEASE</version>

<scope>test</scope>

</dependency>
    <dependency>

<groupId>org.springframework</groupId>
```

```
1</version>

<scope>test</sco
pe>

</dependency>

</dependencies>
  <build>

<finalName>Sprin
gJPA</finalName>
 </build>
</project>
```

3. Create the Java classes ***Employee.java, Employee_Address.java, EmployeeRepository.java, SaveLogic.java*** and ***RetrieveLogic.java*** under ***com.javainterviewpoint*** folder.
4. Place the ***SpringConfig.xml*** under the

interesting articles which you may like ...

- Spring Data JPA Many To Many Foreign Key Example
- Spring PropertyPlaceholderConfigurer Example
- Spring JdbcTemplate Example + JdbcDaoSupport
- Spring CRUD Example with JdbcTemplate + Maven + Oracle
- @Autowired, @Resource, @Qualifier, @Inject Annotation
- Spring Bean Life Cycle – Bean Initialization and Destruction

- How to create Spring Beans Using Spring FactoryBean
- How to specify Spring Bean Reference and Spring Inner Bean
- Spring Dependency Checking and Spring @Required Annotation

Spring Data JPA One To Many Foreign Key Example

Employee.java

Create a new Java file **Employee.java** under the package **com.javainterviewpoint** a nd add the following code

```
.
java.util.Set;

import
javax.persistence.C
ascadeType;
import
javax.persistence.C
olumn;
import
javax.persistence.E
ntity;
import
javax.persistence.F
etchType;
import
javax.persistence.G
eneratedValue;
import
javax.persistence.I
d;
import
javax.persistence.O
neToMany;
import
javax.persistence.T
able;

@Entity
@Table(name="EMPLOY
EE")
public class
Employee
{
```

```
    private int
empId;


@Column(name="NAME"
)
    private String
empName;


@OneToMany(fetch =
FetchType.EAGER,map
pedBy="employee",ca
scade =
CascadeType.ALL)
    private
Set<Employee_Addres
s> employeeAddress;

    public
Employee()
    {
        super();
    }

    public
Employee(int empId,
String empName,
Set<Employee_Addres
s> employeeAddress)
    {
        super();
        this.empId
= empId;
```

```
s =
employeeAddress;
    }

    public int
getEmpId()
    {
        return
empId;
    }

    public void
setEmpId(int empId)
    {
        this.empId
= empId;
    }

    public String
getEmpName()
    {
        return
empName;
    }

    public void
setEmpName(String
empName)
    {

this.empName =
empName;
    }
```

```
    {
        return
employeeAddress;
    }

    public void
setEmployeeAddress(
Set<Employee_Addres
s> employeeAddress)
    {

this.employeeAddres
s =
employeeAddress;
    }

    @Override
    public String
toString()
    {
        return
"Employee [empId="
+ empId + ",
empName=" + empName
+ "]";
    }
}
```

Our **Employee** class is a simple **POJO** class consisting of the **getters** and **setters** for the

used the below **JPA Annotations**.

1. **@Entity** – This annotation will mark our **Employee** class as an **Entity Bean**.
2. **@Table** – **@Table** annotation will map our class to the corresponding database table. You can also specify other attributes such as **indexes, catalog, schema, uniqueConstraints**. The **@Table** annotation is an optional annotation if this annotation is not provided then the class name will be used as the table name.
3. **@Id** – The **@Id** annotation marks the particular field as the

used to specify how the primary key should be generated. Here **SEQUENCE** Strategy will be used as this the default strategy for Oracle

5. **@OneToMany** – We have used the **mappedBy** attribute – This denotes the property which will be used for mapping purpose, here we have an attribute **"employee"** so in our **Employee_Address** class. This is a mandatory annotation.
6. **@Column** – This annotation maps the corresponding fields to their respective columns in the database table.

under the package **com.javainterviewpoint** and add the following code

```
package com.javainterviewpoint;

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

```
public class
Employee_Address
{
    @Id
    @Column(name =
"ADDR_ID")
    @GeneratedValue
    private int
addrId;


@Column(name="STREE
T")
    private String
street;

@Column(name="CITY"
)
    private String
city;

@Column(name="STATE
")
    private String
state;

@Column(name="COUNT
RY")
    private String
country;


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

```
    public
Employee_Address()
    {
        super();
    }

    public
Employee_Address(in
t addrId, String
street, String
city, String state,
String country,
Employee employee)
    {
        super();
        this.addrId
= addrId;
        this.street
= street;
        this.city =
city;
        this.state
= state;

this.country =
country;

this.employee =
employee;
    }

    public int
getAddrId()
```

```
    public void
setAddrId(int
addrId)
    {
        this.addrId
= addrId;
    }

    public String
getStreet()
    {
        return
street;
    }

    public void
setStreet(String
street)
    {
        this.street
= street;
    }

    public String
getCity()
    {
        return
city;
    }

    public void
setCity(String
city)
    {
```

```
getState()
    {
        return
state;
    }

    public void
setState(String
state)
    {
        this.state
= state;
    }

    public String
getCountry()
    {
        return
country;
    }

    public void
setCountry(String
country)
    {

this.country =
country;
    }

    public Employee
getEmployee()
    {
        return
```

```
e employee)
    {

this.employee =
employee;
    }
}
```

@ManyToOne annotation defines the relationship many to one (One **Employee** can have many **Employee_Address**)

@JoinColumn annotation indicates that this entity will act as the owner of the relationship (This table has a column with a **foreign key** to the referenced table)

SpringConfig.xml

Place the **SpringConfig.xml** file also under the ***src/main/resources*** folder

```
xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
        xmlns:context="http://www.springframework.org/schema/context"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:jpa="http://www.springframework.org/schema/data/jpa"
        xsi:schemaLocation="http://www.springframework.org/schema/jdbc
http://www.spri
```

```
http://www.spri
ngframework.org
/schema/beans
http://www.spri
ngframework.org
/schema/beans/s
pring-beans.xsd

http://www.spri
ngframework.org
/schema/tx/
http://www.spri
ngframework.org
/schema/tx/spri
ng-tx-3.2.xsd

http://www.spri
ngframework.org
/schema/context
http://www.spri
ngframework.org
/schema/context
/spring-
context.xsd

http://www.spri
ngframework.org
/schema/mvc
http://www.spri
ngframework.org
/schema/mvc/spr
ing-mvc.xsd

http://www.spri
```

```
/schema/data/jpa/spring-jpa-1.2.xsd">


 <context:component-scan base-package="com.javainterviewpoint">
</context:component-scan>
 <jpa:repositories base-package="com.javainterviewpoint"
 entity-manager-factory-ref="entityManagerFactoryBean">
</jpa:repositories>

 <bean id="saveLogic" class="com.javainterviewpoint.SaveLogic" />
 <bean id="retrieveLogic"
```

```
 <!--
EntityManagerFa
ctory -->
 <bean
id="entityManag
erFactoryBean"
 class="org.spr
ingframework.or
m.jpa.LocalCont
ainerEntityMana
gerFactoryBean"
>
   <property
name="dataSourc
e"
ref="dataSource
" />
   <!-- Now
/META-
INF/persistence
.xml is no
longer needed -
->
   <property
name="packagesT
oScan"
value="com.java
interviewpoint"
/>
   <property
name="jpaVendor
Adapter">
      <bean
class="org.spri
```

```
   <property
name="jpaProper
ties">
     <props>
       <prop
key="hibernate.
hbm2ddl.auto">u
pdate</prop>
       <prop
key="hibernate.
dialect">org.hi
bernate.dialect
.Oracle10gDiale
ct</prop>
     </props>
   </property>
 </bean>

 <bean
id="dataSource"
 class="org.spr
ingframework.jd
bc.datasource.D
riverManagerDat
aSource">
   <property
name="driverCla
ssName"
value="oracle.j
dbc.driver.Orac
leDriver" />
   <property
name="url"
value="jdbc:ora
```

```
   <property
name="password"
value="root" />
 </bean>

 <bean
id="transaction
Manager"
class="org.spri
ngframework.orm
.jpa.JpaTransac
tionManager">
    <property
name="entityMan
agerFactory"
ref="entityMana
gerFactoryBean"
/>
 </bean>
</beans>
```

We have defined the below beans in our **SpringConfig** file.

- **dataSource** : This bean holds all the database related configurations such as driverClassName, url, username, password.

will be passing the **datasource** reference and set values to the properties **jpaVendorAdapter**, **jpaProperties**

- **transactionManager** : We are using the JpaTransactionManager for managing the transactions for our application, we will be passing the **entityManagerFactoryBean** reference to it.

EmployeeRepository.java

Our **EmployeeRepository** interface extends the **JpaRepository** interface. The **JpaRepository** interface contains the basic methods for performing **CRUD** Operations over an entity.

```
com.javainterviewpo
int;

import
org.springframework
.data.jpa.repositor
y.JpaRepository;
import
org.springframework
.stereotype.Compone
nt;

public interface
EmployeeRepository
extends
JpaRepository<Emplo
yee,Integer>
{

}
```

SaveLogic.java

```
package
com.javainterviewpo
int;

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

import
org.springframework
```

```
assPathXmlApplicati
onContext;
import
org.springframework
.stereotype.Compone
nt;

@Component
public class
SaveLogic
{
    private static
SaveLogic
saveLogic;

    @Autowired
    private
EmployeeRepository
employeeRepository;

    public static
void main( String[]
args )
    {
        //Reading
the Configuration
file

ClassPathXmlApplica
tionContext context
= new
ClassPathXmlApplica
tionContext("spring
Config.xml");
```

```
getBean("saveLogic"
);


saveLogic.saveEmplo
yee();


context.close();
    }

    public void
saveEmployee()
    {
        Employee
employee = new
Employee();

employee.setEmpName
("JIP");


Employee_Address
employeeAddress1 =
new
Employee_Address();

employeeAddress1.se
tStreet("Street
1");

employeeAddress1.se
tCity("City 1");
```

```
tState("State 1");

employeeAddress1.se
tEmployee(employee)
;



Employee_Address
employeeAddress2 =
new
Employee_Address();

employeeAddress2.se
tStreet("Street
2");

employeeAddress2.se
tCity("City 2");

employeeAddress2.se
tCountry("Country
2");

employeeAddress2.se
tState("State 2");

employeeAddress2.se
tEmployee(employee)
;



Set<Employee_Addres
s>
```

```
1);

employeeAddressSet.
add(employeeAddress
2);


employee.setEmploye
eAddress(employeeAd
dressSet);


employeeRepository.
save(employee);

System.out.println(
"Employee and
Employee Address
saved
successfully!!");
    }
}
```

- In our **SaveLogic** class, we have read the Configuration file**(SpringConfig.xml)** and get all the bean definition through **ClassPathX mlApplicationConte xt**

```
("springConfig.
xml");
```

- Get the **SaveLogic** Class instance by calling the **getBean()** method over the context created.

```
saveLogic =
(SaveLogic)cont
ext.getBean("sa
veLogic");
```

- Call the **saveEmployee()** method

```
saveLogic.saveE
mployee();
```

- Set the values for the Properties of **Employee** and **Employee_Address** class, and call the **save()** method over the

method is already implemented by **JpaRepository**]

Console:

```
INFO: HHH000261:
Table found:
EMPLOYEE
Jun 16, 2017
4:28:34 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000037:
Columns: [name,
emp_id]
Jun 16, 2017
4:28:34 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000108:
Foreign keys: []
Jun 16, 2017
4:28:34 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000126:
Indexes:
[sys_c0015848]
Jun 16, 2017
```

```
Table found:
EMPLOYEE_ADDRESS
Jun 16, 2017
4:28:38 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000037:
Columns: [street,
emp_id, state,
addr_id, country,
city]
Jun 16, 2017
4:28:38 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000108:
Foreign keys:
[fk_emp]
Jun 16, 2017
4:28:38 PM
org.hibernate.tool.
hbm2ddl.TableMetada
ta <init>
INFO: HHH000126:
Indexes:
[sys_c0015851]
Jun 16, 2017
4:28:38 PM
org.hibernate.tool.
hbm2ddl.SchemaUpdat
e execute
INFO: HHH000232:
```

```
successfully!!
```

RetrieveLogic.java

```
package
com.javainterviewpo
int;

import
java.util.List;
import
java.util.Set;

import
org.springframework
.beans.factory.anno
tation.Autowired;
import
org.springframework
.context.support.Cl
assPathXmlApplicati
onContext;
import
org.springframework
.stereotype.Compone
nt;

@Component
public class
RetrieveLogic
{
    private static
```

```
EmployeeRepository
employeeRepository;

    public static
void main(String[]
args)
    {
        // Reading
the Configuration
file

ClassPathXmlApplica
tionContext context
= new
ClassPathXmlApplica
tionContext("spring
Config.xml");

        // Get the
RetrieveLogic bean

retrieveLogic =
(RetrieveLogic)
context.getBean("re
trieveLogic");


retrieveLogic.retri
eveEmployee();


context.close();
    }
    public void
```

```
List<Employee>
employeeList =
employeeRepository.
findAll();

        //
Displaying the
Employee details
        for
(Employee employee
: employeeList)
        {

System.out.println(
"*** Employee
Details ***");


System.out.println(
"Employee Id   : "
+
employee.getEmpId()
);

System.out.println(
"Employee Name : "
+
employee.getEmpName
());


System.out.println(
"*** Employee
```

```
employee.getEmploye
eAddress();
            for
(Employee_Address
employeeAddress :
empAddressSet)
            {

System.out.println(
"Street  : " +
employeeAddress.get
Street());

System.out.println(
"City    : " +
employeeAddress.get
City());

System.out.println(
"State   : " +
employeeAddress.get
State());

System.out.println(
"Country : " +
employeeAddress.get
Country());
            }
        }
    }
}
```

) and get all the bean definition through **ClassPathXmlApplicationContext**

```
ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("springConfig.xml");
```

- Get the **SaveLogic** Class instance by calling the **getBean()** method over the context created.

```
retrieveLogic=(retrieveLogic)context.getBean("saveLogic");
```

- Call the **retrieveEmployee()** method

method over the **employeeRepository** instance [**findAll()** method is already implemented by **JpaRepository**]

Output:

```
*** Employee Details ***
Employee Id : 84
Employee Name : JIP
*** Employee Address Details ***
Street : Street 2
City : City 2
State : State 2
Country : Country 2
Street : Street 1
City : City 1
State : State 1
Country : Country 1
```

Download Source Code

Leave a Reply

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

Comment

Name *

Email *

Website

Java Basics

JVM Architecture

Object in Java

Class in Java

How to Set Classpath for Java in Windows

Components of JDK

Decompiling a class file

Use of Class.forName in java

Use Class.forName in SQL JDBC

Oops Concepts

Inheritance in Java

Types of Inheritance in Java

Java Keywords

Miscellaneous

Serialization and Deserialization in Java with Example

Generate SerialVersionUID in Java

How to make a class Immutable in Java

Differences betwen HashMap and Hashtable

Difference between Enumeration and Iterator ?

Difference between fail-fast and fail-safe Iterator

Difference Between Interface and Abstract Class in Java

Difference between equals() and ==

Sort Objects in a ArrayList using Java Comparable Interface

USEFUL LINKS

Spring 4.1.x Documentation

Spring 3.2.x Documentation

Spring 2.5.x Documentation

Java 6 API

Java 7 API

Java 8 API

Java EE 5 Tutorial

Java EE 6 Tutorial

Java EE 7 Tutorial

Maven Repository

Hibernate ORM

javainterviewpoint.com is a tech blog dedicated to all Java/J2EE developers and Web Developers. We publish useful tutorials on Java, J2EE and all latest frameworks. All examples and tutorials posted here are very well tested in our development environment. Connect with us on facebook Privacy Policy | Contact Us