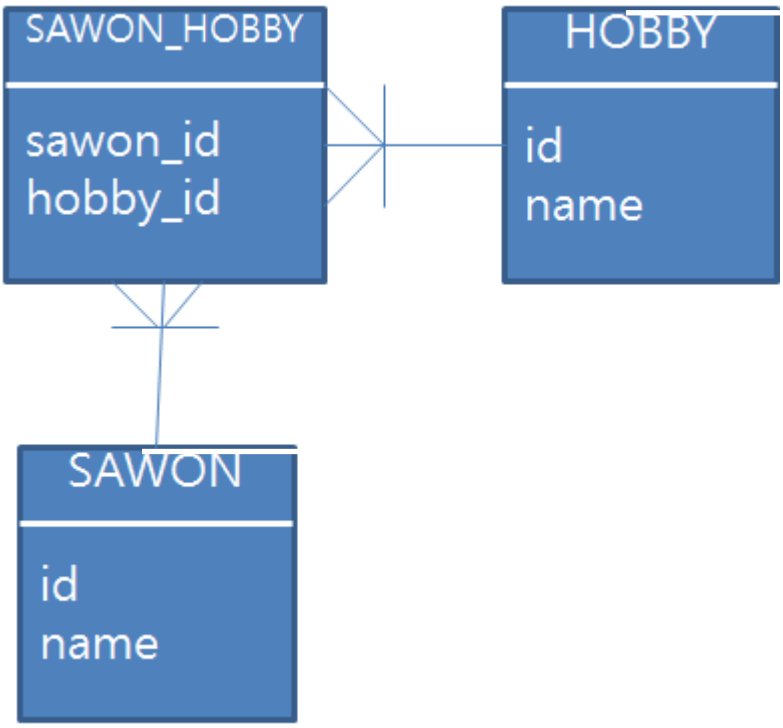
스프링부트,JPA,MAVEN,ORACLE,OneToMany 예제

간단히 Spring Boot, Maven, JPA(Hibernate ORM)를 이용하여 ManyToMany 형태의 예제를 작성해 보자.

Spring Boot 는 간단히 스프링 프로젝트를 구축해주는 것이며 JPA 는 JPA 는 EJB 3.0 스펙 작업에서 기존 EJB 의 ORM 인 엔티티빈(Entity Bean)을 JPA 로 변경하여 자바에서 영속성 관리를 위한 표준 API 이다. JPA 는 Hibernate, OpenJPA, EclipseLink, TopLink Essentials 과 같은 구현체가 있고 표준 인터페이스가 바로 JPA 인 것이다.



[예제 테이블 구조 – 예제 Hibernate 에서 자동으로 생성]

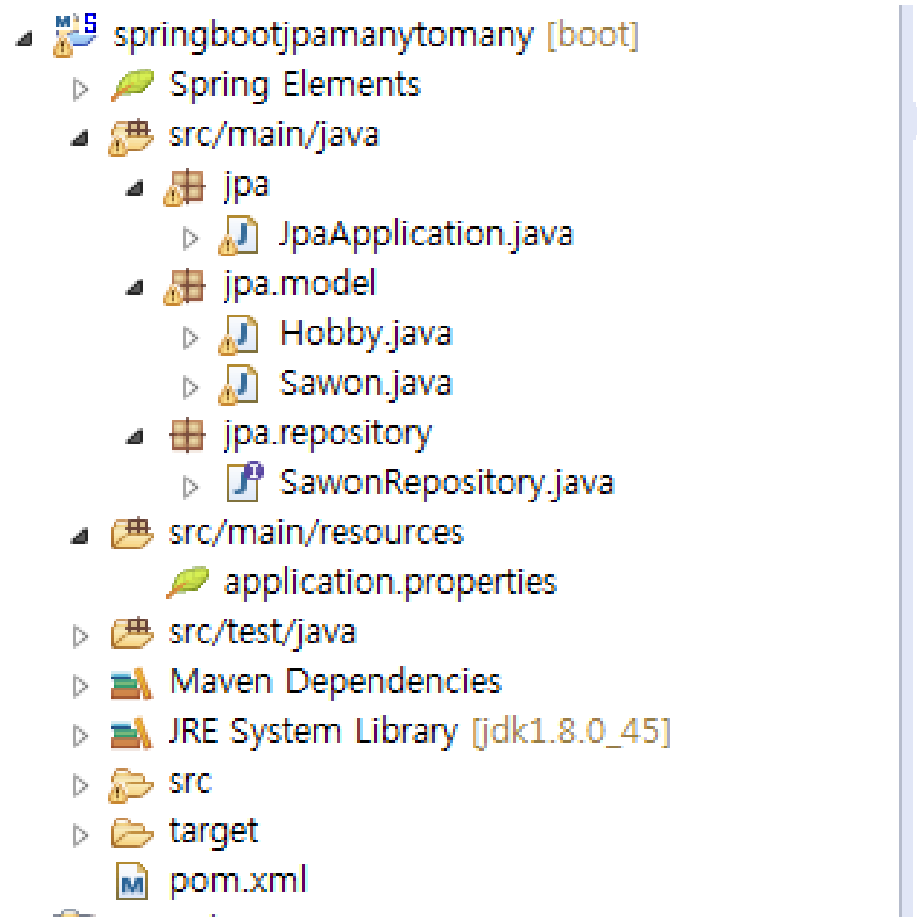
```
drop table sawon cascade constraints;
create table sawon(
   id number primary key,
   name varchar2(20)
);
```

```
drop table hobby cascade constraints;
create table hobby(
   id number primary key,
   name varchar2(20)
);

drop table sawon_hobby cascade constraints;
create table sawon_hobby(
   sawon_id number references sawon(id),
   hobby_id number references hobby(id)
);
```

1. Eclipse/STS 에서 New -> Project -> Spring -> Spring Maven Project 프로젝트를 생성하고 프로젝트명을 springbootjpamanytomany 라고 주자.

전체 프로젝트 구조는 다음과 같다.



2. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>ojc</groupId>
      <artifactId>jpa</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>jar</packaging>

      <name>JPA One-To-Many Example</name>
      <description>JPA One-To-Many Example</description>

      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>1.2.6.RELEASE</version>
            <relativePath /> <!-- lookup parent from repository -->
      </parent>

      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <java.version>1.8</java.version>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-data-
jpa</artifactId>
            </dependency>

            <dependency>
                  <groupId>mysql</groupId>
                  <artifactId>mysql-connector-java</artifactId>
                  <scope>runtime</scope>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
            </dependency>
```

```
		<!-- ORACLE -->
		<dependency>
			<groupId>com.oracle</groupId>
			<artifactId>ojdbc6</artifactId>
			<version>11.1.0.7.0</version>
		</dependency>
	</dependencies>
	<repositories>
		<repository>
			<id>oracle</id>
			<name>ORACLE JDBCRepository</name>
			<url>http://maven.jahia.org/maven2</url>
		</repository>
	</repositories>
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-
plugin</artifactId>
			</plugin>
		</plugins>
	</build>
</project>
```

3. Model(Hobby.java)

@Entity 어노테이션으로 JPA Entities 를 표현한다.

```
package jpa.model;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class Hobby {
	private int id;
	private String name;
	private Set<Sawon> sawons;
```

```
	public Hobby() { }
	public Hobby(int id, String name) {
		this.id = id;
		this.name = name;
	}

	public Hobby(int id, String name, Set<Sawon> sawons) {
		this.id = id;
		this.name = name;
		this.sawons = sawons;
	}

	@Id
	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;
	}
	/*
	 * mappedBy : ~에의해매핑되어있다는 의미
	 *		Sawon 테이블의 hobbies 필드를 매핑
	 *		관계가 양방향 일때 mappedBy 속성에 Relation 의
Owner 가
	 *		되는쪽의 어떤속성(필드)으로 접근을 하는지 설정
	 */
	@ManyToMany(mappedBy = "hobbies")
	public Set<Sawon> getSawons() {
		return sawons;
	}
	public void setSawons(Set<Sawon> sawons) {
		this.sawons = sawons;
	}
}
```

4. Model(Sawon.java) – JPA Entity

@Entity 어노테이션으로 JPA Entities 를 표현한다.

```
package jpa.model;

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

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

/*
 * @Table : 테이블과 연관하여 Entity 를 매핑, 안쓰면 Entity 클래스이름
 * @Id : Entity 의 식별자(Table PK 칼럼을 의미)
 * @Column : 테이블의 칼럼과 Enruty 의 필드를 매핑,안쓰면 필드명이 칼럼명
 * @ManyToMany : 두 Entity 의 M 대 N 관계 매핑
 * @JoinColumn : Entity 가 조인관계의 Owner 임을 선언,
 *                  아래에서 sawon_id 칼럼은 외래키로 Sawon 의 id 를 참조
 */
@Entity
public class Sawon {
        private int id;
        private String name;
        private Set<Hobby> hobbies;

        public Sawon() { }
        public Sawon(int id, String name, Set<Hobby> hobbies) {
                this.id = id;
                this.name = name;
                this.hobbies = hobbies;
        }
        @Id
        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;
        }
        /*
         * cascade 속성 : 관계를 맺고 있는 타겟 쪽에 어떠한 persistence operation 들을
         *                  연쇄적으로 적용하고 싶을 때 설정.
                        ALL, MERGE, PERSIST, REFRESH, REMOVE
                            참조무결성을 위한 설정
         */
        @ManyToMany(cascade=CascadeType.ALL)
        @JoinTable(name = "sawon_hobby",
                        joinColumns = @JoinColumn(name = "sawon_id", referencedColumnName="id"),
                        inverseJoinColumns = @JoinColumn(name = "habby_id", referencedColumnName = "id"))
        public Set<Hobby> getHobbies() {
                return hobbies;
        }
        public void setHobbies(Set<Hobby> hobbiefs) {
                this.hobbies = hobbies;
        }
        @Override
    public String toString() {
         String result = String.format(
                    "Sawon [id=%d, name='%s']%n",
                    id, name);
         if (hobbies != null) {
              for(Hobby hobby : hobbies) {
                   result += String.format(
                              "Hobby[id=%d, name='%s']%n",
                              hobby.getId(), hobby.getName());
              }
         }

         return result;
    }
```

```
}
```

5. src/main/resources 아래에 application.properties 파일을 만들자.

```
spring.datasource.url=jdbc:oracle:thin:@192.168.0.27:1521:onj
spring.datasource.driverClassName=oracle.jdbc.driver.OracleDriver
spring.datasource.username=test
spring.datasource.password=test

spring.jpa.hibernate.ddl-auto=create
spring.jpa.database=oracle
spring.jpa.show-sql=true
```

6. Repository(SawonRepository.java)

```
package jpa.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import jpa.model.Sawon;

public interface SawonRepository extends JpaRepository<Sawon, Integer> {

}
```

7. Client(JpaApplication.java)

```
package jpa;

import java.util.HashSet;
import javax.transaction.Transactional;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import jpa.model.Hobby;
```

```
import jpa.model.Sawon;
import jpa.repository.SawonRepository;

@SpringBootApplication
public class JpaApplication implements CommandLineRunner {

	private static final Logger logger =
LoggerFactory.getLogger(JpaApplication.class);

	@Autowired
	private SawonRepository sawonRepository;

	public static void main(String[] args) {
		SpringApplication.run(JpaApplication.class, args);
	}

	@Override
	@Transactional
	public void run(String... strings) throws Exception {
		Hobby h1 = new Hobby(1, "등산");
		Hobby h2 = new Hobby(2, "낚시");
		Hobby h3 = new Hobby(3, "골프");

		sawonRepository.save(new HashSet<Sawon>() {
			{
				add(new Sawon(1, "1 길동", new
HashSet<Hobby>() {
					{
						add(h1);
						add(h3);
					}
				}));

				add(new Sawon(2, "2 길동", new
HashSet<Hobby>() {
					{
						add(h1);
						add(h2);
					}
				}));

				add(new Sawon(3, "3 길동", new
HashSet<Hobby>() {
					{
						add(h2);
```

```
                                    add(h3);
                              }
                        }));

                  }
            });

            for(Sawon s : sawonRepository.findAll()) {
                  logger.info(s.toString());
            }
      }
}
```

8. 실행결과

이클립스에서 JpaApplication.java 를 실행

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.2.6.RELEASE)

2015-12-17 17:31:40.012  INFO 3076 --- [          main]
jpa.JpaApplication                      : Starting JpaApplication on
ooo-PC with PID 3076
(D:\java\project3\springbootjpamanytomany\target\classes started by
ooo in D:\java\project3\springbootjpamanytomany)
2015-12-17 17:31:40.074  INFO 3076 --- [          main]
s.c.a.AnnotationConfigApplicationContext : Refreshing
org.springframework.context.annotation.AnnotationConfigApplicationCo
ntext@768b970c: startup date [Thu Dec 17 17:31:40 KST 2015]; root of
context hierarchy
2015-12-17 17:31:41.650  INFO 3076 --- [          main]
j.LocalContainerEntityManagerFactoryBean : Building JPA container
EntityManagerFactory for persistence unit 'default'
2015-12-17 17:31:41.683  INFO 3076 --- [          main]
o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing
PersistenceUnitInfo [
      name: default
      ...]
2015-12-17 17:31:41.761  INFO 3076 --- [          main]
org.hibernate.Version                   : HHH000412: Hibernate Core
{4.3.11.Final}
```

```
2015-12-17 17:31:41.776  INFO 3076 --- [          main]
org.hibernate.cfg.Environment           : HHH000206:
hibernate.properties not found
2015-12-17 17:31:41.776  INFO 3076 --- [          main]
org.hibernate.cfg.Environment           : HHH000021: Bytecode
provider name : javassist
2015-12-17 17:31:42.088  INFO 3076 --- [          main]
o.hibernate.annotations.common.Version   : HCANN000001: Hibernate
Commons Annotations {4.0.5.Final}
2015-12-17 17:31:42.900  INFO 3076 --- [          main]
org.hibernate.dialect.Dialect           : HHH000400: Using dialect:
org.hibernate.dialect.Oracle9iDialect
2015-12-17 17:31:43.057  INFO 3076 --- [          main]
o.h.h.i.ast.ASTQueryTranslatorFactory    : HHH000397: Using
ASTQueryTranslatorFactory
2015-12-17 17:31:43.415  INFO 3076 --- [          main]
org.hibernate.tool.hbm2ddl.SchemaExport  : HHH000227: Running
hbm2ddl schema export
Hibernate: drop table hobby cascade constraints
Hibernate: drop table sawon cascade constraints
Hibernate: drop table sawon_hobby cascade constraints
Hibernate: create table hobby (id number(10,0) not null, name
varchar2(255 char), primary key (id))
Hibernate: create table sawon (id number(10,0) not null, name
varchar2(255 char), primary key (id))
Hibernate: create table sawon_hobby (sawon_id number(10,0) not null,
habby_id number(10,0) not null, primary key (sawon_id, habby_id))
Hibernate: alter table sawon_hobby add constraint
FK_bnccs61dd54lgcpu3jifwd4dc foreign key (habby_id) references hobby
Hibernate: alter table sawon_hobby add constraint
FK_qqk3c8ippcim47nrhxyislbtl foreign key (sawon_id) references sawon
2015-12-17 17:31:43.690  INFO 3076 --- [          main]
org.hibernate.tool.hbm2ddl.SchemaExport  : HHH000230: Schema export
complete
2015-12-17 17:31:44.220  INFO 3076 --- [          main]
o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX
exposure on startup
Hibernate: select sawon0_.id as id1_1_1_, sawon0_.name as name2_1_1_,
hobbies1_.sawon_id as sawon_id1_1_3_, hobby2_.id as habby_id2_2_3_,
hobby2_.id as id1_0_0_, hobby2_.name as name2_0_0_ from sawon
sawon0_, sawon_hobby hobbies1_, hobby hobby2_ where
sawon0_.id=hobbies1_.sawon_id(+) and hobbies1_.habby_id=hobby2_.id(+)
and sawon0_.id=?
Hibernate: select hobby0_.id as id1_0_0_, hobby0_.name as name2_0_0_
from hobby hobby0_ where hobby0_.id=?
Hibernate: select hobby0_.id as id1_0_0_, hobby0_.name as name2_0_0_
from hobby hobby0_ where hobby0_.id=?
Hibernate: select sawon0_.id as id1_1_1_, sawon0_.name as name2_1_1_,
```

```
hobbies1_.sawon_id as sawon_id1_1_3_, hobby2_.id as habby_id2_2_3_,
hobby2_.id as id1_0_0_, hobby2_.name as name2_0_0_ from sawon
sawon0_, sawon_hobby hobbies1_, hobby hobby2_ where
sawon0_.id=hobbies1_.sawon_id(+) and hobbies1_.habby_id=hobby2_.id(+)
and sawon0_.id=?
Hibernate: select hobby0_.id as id1_0_0_, hobby0_.name as name2_0_0_
from hobby hobby0_ where hobby0_.id=?
Hibernate: select sawon0_.id as id1_1_1_, sawon0_.name as name2_1_1_,
hobbies1_.sawon_id as sawon_id1_1_3_, hobby2_.id as habby_id2_2_3_,
hobby2_.id as id1_0_0_, hobby2_.name as name2_0_0_ from sawon
sawon0_, sawon_hobby hobbies1_, hobby hobby2_ where
sawon0_.id=hobbies1_.sawon_id(+) and hobbies1_.habby_id=hobby2_.id(+)
and sawon0_.id=?
Hibernate: insert into sawon (name, id) values (?, ?)
Hibernate: insert into hobby (name, id) values (?, ?)
Hibernate: insert into hobby (name, id) values (?, ?)
Hibernate: insert into sawon (name, id) values (?, ?)
Hibernate: insert into hobby (name, id) values (?, ?)
Hibernate: insert into sawon (name, id) values (?, ?)
Hibernate: select sawon0_.id as id1_1_, sawon0_.name as name2_1_
from sawon sawon0_
2015-12-17 17:31:44.564  INFO 3076 --- [          main]
jpa.JpaApplication                     : Sawon [id=2, name='2길동']

2015-12-17 17:31:44.564  INFO 3076 --- [          main]
jpa.JpaApplication                     : Sawon [id=1, name='1길동']

2015-12-17 17:31:44.564  INFO 3076 --- [          main]
jpa.JpaApplication                     : Sawon [id=3, name='3길동']

2015-12-17 17:31:44.564  INFO 3076 --- [          main]
jpa.JpaApplication                     : Started JpaApplication in
4.959 seconds (JVM running for 5.752)
2015-12-17 17:31:44.564  INFO 3076 --- [      Thread-1]
s.c.a.AnnotationConfigApplicationContext : Closing
org.springframework.context.annotation.AnnotationConfigApplicationCo
ntext@768b970c: startup date [Thu Dec 17 17:31:40 KST 2015]; root of
context hierarchy
2015-12-17 17:31:44.580  INFO 3076 --- [      Thread-1]
o.s.j.e.a.AnnotationMBeanExporter       : Unregistering JMX-exposed
beans on shutdown
2015-12-17 17:31:44.580  INFO 3076 --- [      Thread-1]
j.LocalContainerEntityManagerFactoryBean : Closing JPA
EntityManagerFactory for persistence unit 'default'
```

[실행 후 테이블 확인]
SQL> select * from sawon;

```
        ID NAME
---------- ----------
         2 2 길동
         1 1 길동
         3 3 길동
```

SQL> select * from hobby;

```
        ID NAME
---------- ----------
         1 등산
         2 낚시
         3 골프
```