



**SPRING 05**

김규석 교수  
(스프링프레임워크)

## ● 학습목표

---

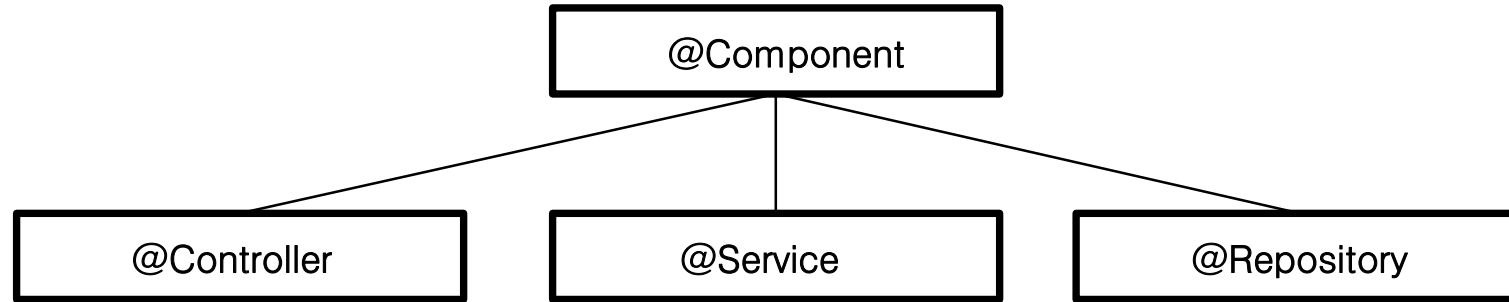
- ✓ Annotation의 이해
- ✓ Bean의 이해
- ✓ Hibernate의 이해

## ● Annotation

---

### ✓ Spring Annotation의 구조

- @Controller, @Service, @Repository 등으로 Bean을 등록하고 일반적으로 @Autowired로 주입받아 사용한다.
- DI를 하기 위한 방법

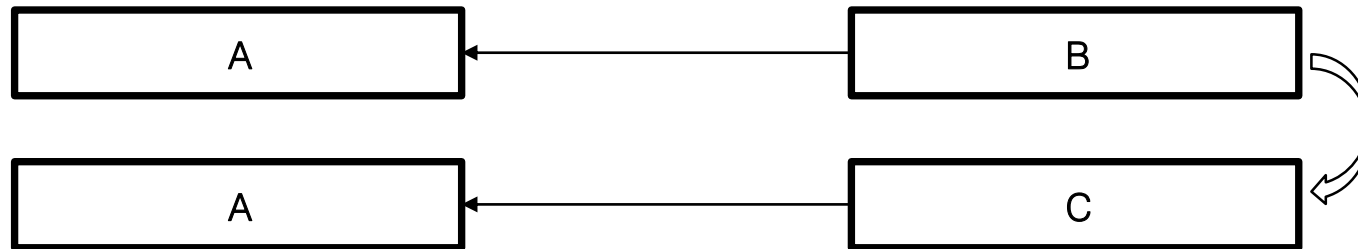


## ● Annotation

### ✓ Annotation을 활용한 객체 생성

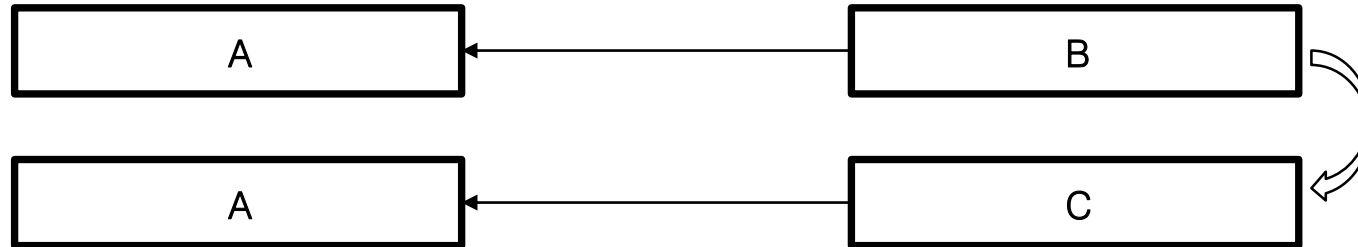
#### – Before

: 아래처럼 객체 B를 C로 변경하고자 하면, 객체를 새로 생성하고 설정을 해줘야 함



#### – After

: Spring Annotation을 활용하면 @Component로 자동 생성, 자동 설정(연결)이 됨



## ● Annotation

---

### ✓ 실습 #1

- 아래는 아주 간단한 예제이다.
- 다양한 Annotation이 존재하나 @value를 먼저 사용해 보기
- Annotation 관련해서는 Bean을 생성하면서 더 해보자.

```
@Value("30")  
public int num;  
  
@Value("temptemptemp")  
public String temp;  
  
@Value("true")  
public String flag1;
```

```
30  
temptemptemp  
true
```

## ● Bean

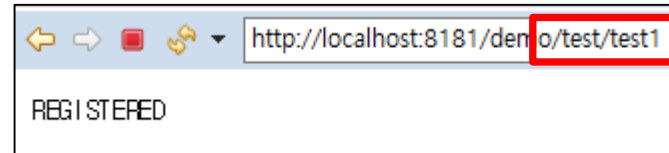
### ✓ 실습 #2

#### – URL mapping 하기

```
@Controller
public class HomeController {
    @Autowired
    public Join join;

    @RequestMapping("/test/test1")
    @ResponseBody
    public String doRegister() {
        return join.register();
    }
}

@Component
class Join {
    public String register() {
        return "REGISTERED";
    }
}
```



## ● Bean

---

### ✓ Bean

- 일반적으로 XML을 통해서 생성되는 Spring 에서의 객체임
- Java의 객체로 생각할 수 있음
- ApplicationContext.getBean()을 통해서 얻을 수 있는 ApplicationContext가

담고 있는 객체

- Bean을 생성하는 방법은 4가지가 있고, 실습을 통해 방법 몇 가지를 숙지해 보자

## ● Bean

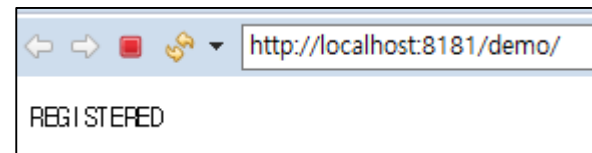
### ✓ 실습 #3

#### – 객체 생성하는 방법

```
@Controller
public class HomeController {
    @Autowired
    public Join join;

    @RequestMapping("/")
    @ResponseBody
    public String doRegister() {
        return join.register();
    }
}

@Component
class Join {
    public String register() {
        return "REGISTERED";
    }
}
```





## ● Bean

### ✓ 실습 #4

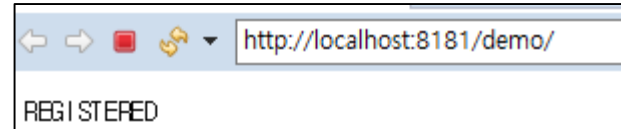
– 객체 생성하는 방법(Member interface를 활용한 클래스에 자동 연결해 줌)

```
@Controller
public class HomeController {
    @Autowired
    public Member join;

    @RequestMapping("/")
    @ResponseBody
    public String doRegister() {
        return join.register();
    }
}

interface Member {
    public String register();
}

@Component
class Join implements Member{
    public String register() {
        return "REGISTERED";
    }
}
```



## ✓ 실습 #5

– 객체 생성하는 방법(Member interface를 활용한 클래스가 여러 개 일 때)

```
@Controller
public class HomeController {
    @Autowired
    @Qualifier("test1")
    public Member join;

    @RequestMapping("/")
    @ResponseBody
    public String doRegister() {
        return join.register();
    }
}

interface Member {
    public String register();
}

@Component("test1")
class Join implements Member{
    public String register() {
        return "REGISTERED";
    }
}

@Component("test2")
class Join2 implements Member{
    public String register() {
        return "REGISTERED2";
    }
}
```

← → ⏏ ⏏ ⏏ http://localhost:8181/demo/  
REGISTERED

← → ⏏ ⏏ ⏏ http://localhost:8181/demo/  
REGISTERED2

```
@Controller
public class HomeController {
    @Autowired
    @Qualifier("test2")
    public Member join;

    @RequestMapping("/")
    @ResponseBody
    public String doRegister() {
        return join.register();
    }
}

interface Member {
    public String register();
}

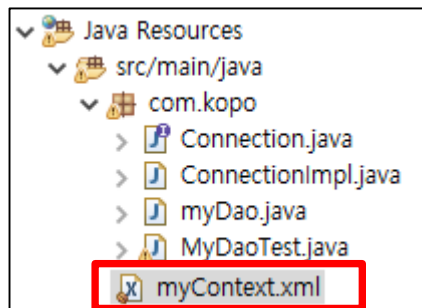
@Component("test1")
class Join implements Member{
    public String register() {
        return "REGISTERED";
    }
}

@Component("test2")
class Join2 implements Member{
    public String register() {
        return "REGISTERED2";
    }
}
```

## ● Bean

### ✓ 실습 #6(cont'd)

– XML에서 Bean을 만들어서 Java 코드에서 호출해 보기

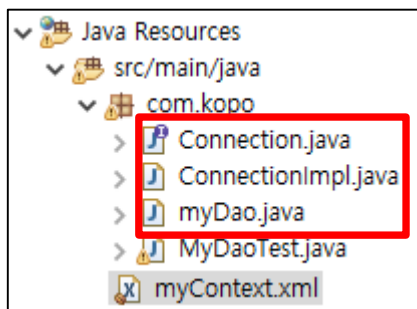


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:jdbc="http://www.springframework.org/schema/jdbc"
  xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd
    http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

  <bean id="connectionObj" class="com.kopo.ConnectionImpl"/>
  <bean id="myDao" class="com.kopo.myDao">
    <property name="connectionObj" ref="connectionObj"/>
  </bean>
</beans>
```

## ● Bean

### ✓ 실습 #6(cont'd)



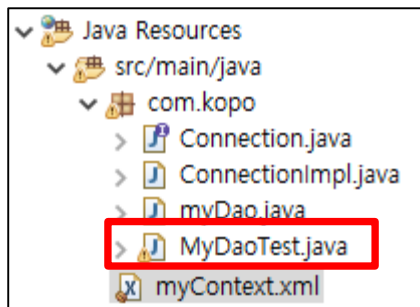
```
package com.kopo;  
  
public interface Connection {  
    public String checkConnection();  
}
```

```
package com.kopo;  
  
public class ConnectionImpl implements Connection {  
    @Override  
    public String checkConnection() {  
        return "connection OK";  
    }  
}
```

```
package com.kopo;  
  
public class myDao {  
    private Connection connection;  
  
    public void check(){  
        String connectionStatus = connection.checkConnection();  
        System.out.println("Connection Status : " + connectionStatus);  
    }  
  
    public Connection getConnectionObj() {  
        return connection;  
    }  
  
    public void setConnectionObj(Connection connection) {  
        this.connection = connection;  
    }  
}
```

## ● Bean

### ✓ 실습 #6



```
package com.kopo;
import org.springframework.context.ApplicationContext;

public class MyDaoTest {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("myContext.xml");
        myDao dao = context.getBean("myDao", myDao.class);
        dao.check();
        myDao dao2 = null;
        System.out.println("dao : " + dao);
        System.out.println("dao : " + dao2);
    }
}
```

## ● Bean

### ✓ 실습 #7

- 실습 #6의 코드에서 아래 코드를 수정한 후, 에러가 나는 원인을 파악하고 수정하시오

```
<bean id="connectionObj" class="com.kopo.ConnectionImpl"/>
<bean id="myDao" class="com.kopo.myDao">
    <property name="connectionObj" ref="connectionObj"/>
</bean>
<bean id="myDao2" class="com.kopo.myDao">
</bean>
```

```
public class MyDaoTest {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("myContext.xml");
        myDao dao = context.getBean("myDao", myDao.class);
        dao.check();
        myDao dao2 = context.getBean("myDao2", myDao.class);
        System.out.println("dao : " + dao);
        dao2.check();
        System.out.println("dao : " + dao2);
    }
}
```

```
Connection Status : connection OK
dao : com.kopo.myDao@1f554b06
Exception in thread "main" java.lang.NullPointerException
    at com.kopo.myDao.check(MyDao.java:7)
    at com.kopo.MyDaoTest.main(MyDaoTest.java:13)
```

## ● Hibernate

---

### ✓ 특징

1. MyBatis는 SQL 쿼리문을 사용하는 것임
2. Hibernate는 SessionFactory의 persist(), update(), delete() 등과 같은 함수를 사용
3. Hibernate는 DB table의 구조에 의존적이지 않음

“그 이유는 예제 코드를 실행해 보면서 이해하기”

즉, Mybatis는 SQL 쿼리문을 사용하기에 초보자도 이해하거나 적용을 하기에 쉬움

하지만, Table 구조가 바뀌는 등의 변경사항에 Dependency가 걸림

Hibernate는 구조를 정확히 알고 구현해야 한다는 점에서의 어려운 점은 있으나

Table 구조에 Independent 하여 대규모/대용량 프로젝트에 적합함

## ● Hibernate

### ✓ Mybatis와 Hibernate의 비교

```
memberMapper.xml
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org/DTD Config 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.example.mapper.memberMapper">
7
8     <select id="selectMember" resultType="memberVO">
9         SELECT ID, PW, NAME FROM TEST
10    </select>
11
12 </mapper>
```

```
ItemsDAO.java
1 package com.kopo.hibernateExample.dao;
2
3 import java.util.List;
4
11
12 @Repository
13 public class ItemsDAO {
14
15     @Autowired
16     private SessionFactory sessionFactory;
17
18     public void setSessionFactory(SessionFactory sf) {
19         this.sessionFactory = sf;
20     }
21
22     public List<Items> getAllItems() {
23         Session session = this.sessionFactory.getCurrentSession();
24         List<Items> itemList = session.createQuery("from Items").list();
25         return itemList;
26     }
27
28     public Items getItem(int id) {
29         Session session = this.sessionFactory.getCurrentSession();
30         Items item = (Items) session.get(Items.class, new Integer(id));
31         return item;
32     }
33
34     public Items addItem(Items item) {
35         Session session = this.sessionFactory.getCurrentSession();
36         session.persist(item);
37         return item;
38     }
39
40     public void updateItem(Items item) {
41         Session session = this.sessionFactory.getCurrentSession();
42         session.update(item);
43     }
44
45     public void deleteItem(int id) {
46         Session session = this.sessionFactory.getCurrentSession();
47         Items p = (Items) session.load(Items.class, new Integer(id));
48         if (null != p) {
49             session.delete(p);
50         }
51     }
52 }
```



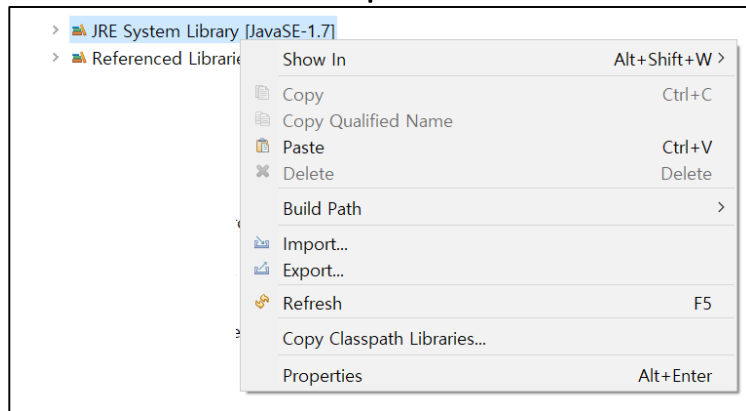
## ● Hibernate

### ✓ 실습 #8(cont'd)

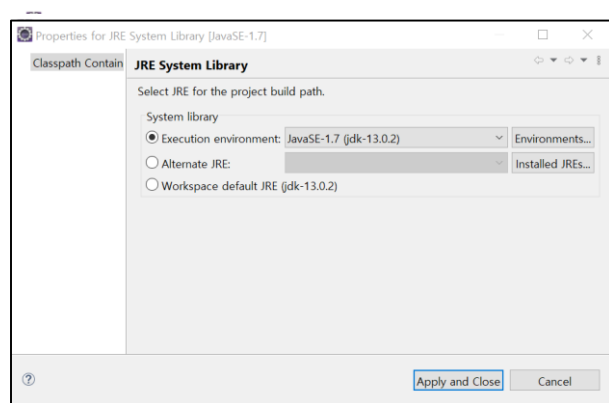
[환경 구성(JDK 추가)]

– JDK를 설치(NAS도 첨부되어 있음)

– 아래 그림에서 Properties로 진입



– Installed JREs 선택

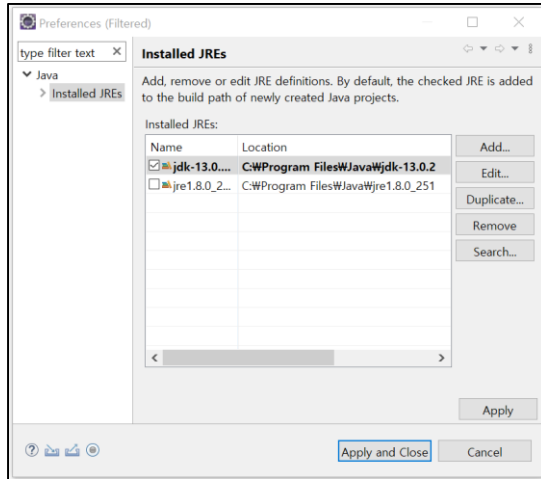


# ● Hibernate

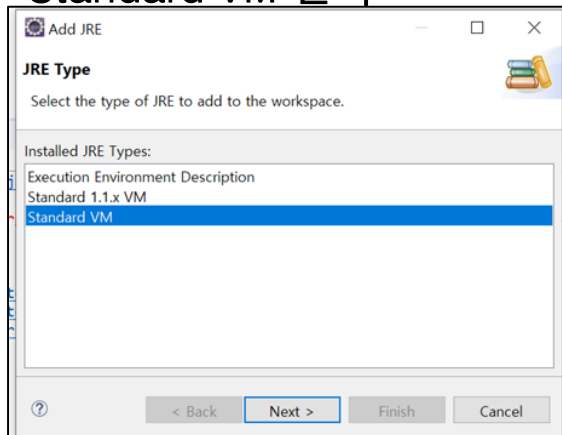
## ✓ 실습 #8(cont'd)

[환경 구성(JDK 추가)]

– Add 버튼을 누름



– Standard VM 선택

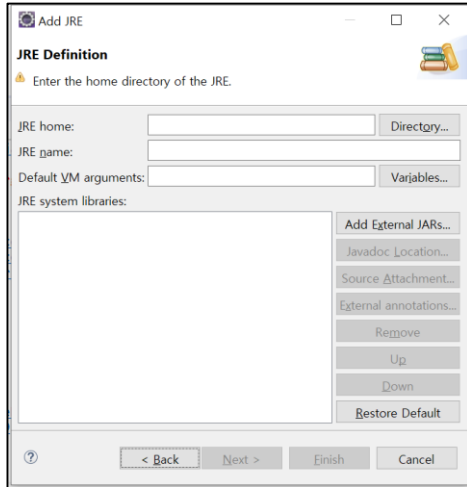


# ● Hibernate

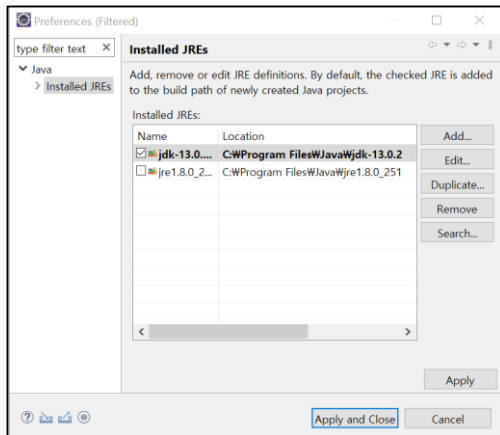
## ✓ 실습 #8(cont'd)

### [환경 구성(JDK 추가)]

– Add External JARs를 통해 JDK가 설치된 폴더 선택



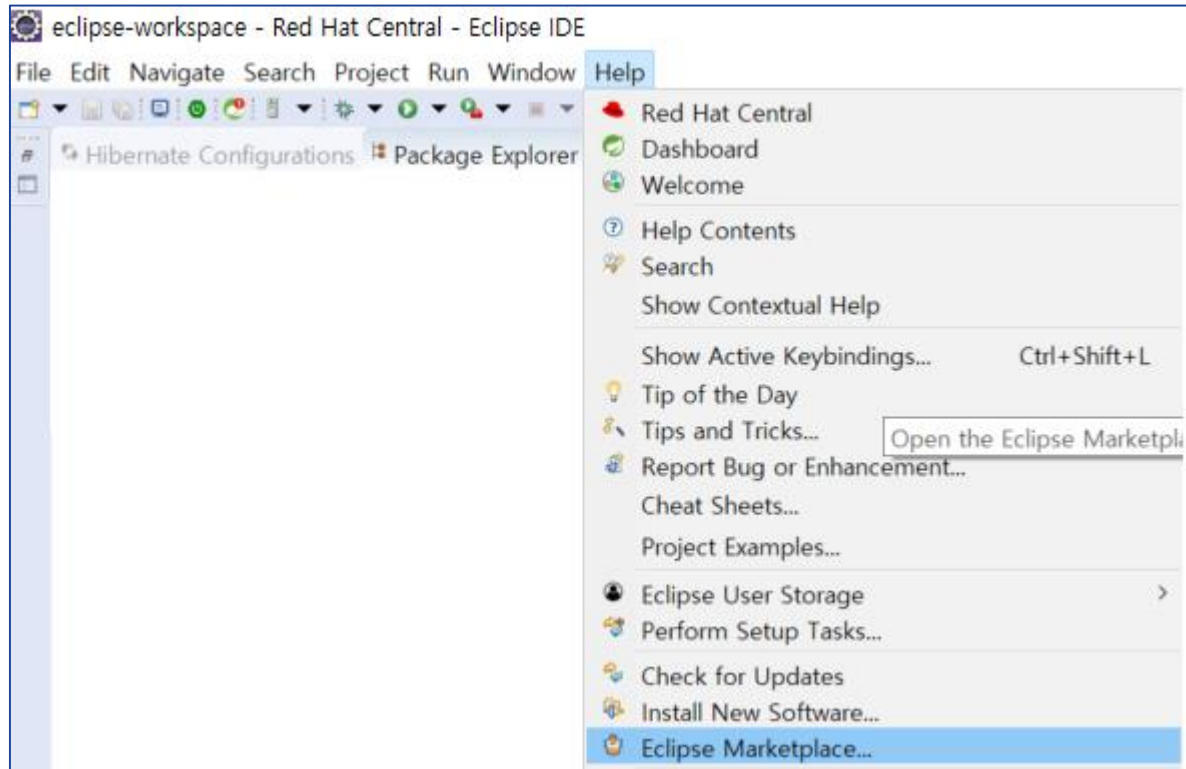
– JDK 선택 후 적용



## ● Hibernate

### ✓ 실습 #8(cont'd)

#### [Plugin 설치]



## ● Hibernate

### ✓ 실습 #8(cont'd)

#### [Plugin 설치]

##### Hibernate Search Plugin v2.0.0.Final



requires Jboss Hibernate Tools already installed The Eclipse plugin for Hibernate Search helps you work with lucene indexes, analyzers and other features in a... [more info](#)

by [bdshadow](#), EPL

★ 13



Installs: **12.5K** (223 last month)

Installed

##### JBoss Tools 4.15.0.Final



JBoss Tools is an umbrella project for a set of Eclipse plugins that includes support for JBoss and related technologies, such as Hibernate, JBoss AS / WildFly,... [more info](#)

by [Red Hat, Inc.](#), EPL

★ 1043



Installs: **1.09M** (10,676 last month)

Installed

## ● Hibernate

### ✓ 실습 #8(cont'd)

– 첨부된 예제를 실행하기 위해 아래와 같이 DB와 Table을 구성

```
선택 MySQL 8.0 Command Line Client
mysql> use testdata:
Database changed
mysql> desc items:
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| itemsName | varchar(100) | NO | | NULL    |
| price | int  | NO   |     | NULL    |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

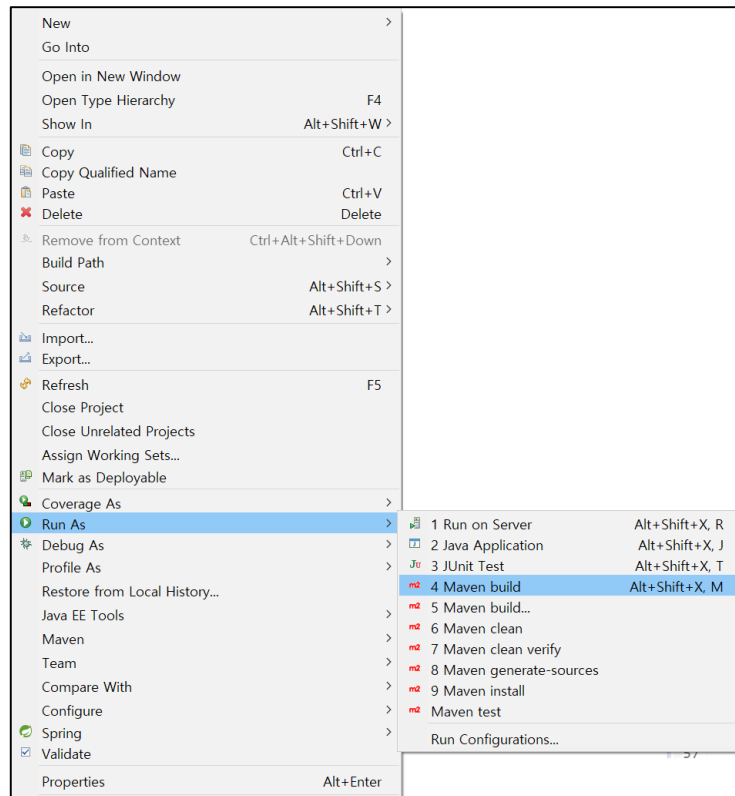
mysql> select * from items:
+-----+-----+-----+
| id | itemsName | price |
+-----+-----+-----+
| 1  | apple    | 1000  |
| 2  | apple2   | 1000  |
+-----+-----+-----+
2 rows in set (0.03 sec)

mysql> _
```

## ✓ 실습 #8(cont'd)

### [빌드 및 실행하기]

1. 첨부된 예제를 import 함
2. Maven build를 함
3. Run on Server를 통해 실행함



## ● Hibernate

### ✓ 실습 #8(cont'd)

#### – Controller 구성

```
package com.kopo.hibernateExample.controller;

import java.util.List;

@Controller
public class ItemsController {

    @Autowired
    ItemsService itemsService;

    @RequestMapping(value = "/items", method = RequestMethod.GET, headers = "Accept=application/json")
    public String getItems(Model model) {

        List<Items> listOfItems = itemsService.getAllItems();
        model.addAttribute("items", new Items());
        model.addAttribute("listOfItems", listOfItems);
        return "itemsDetails";
    }
}
```



## ✓ 실습 #8(cont'd)

### – DAO 구성

```
@Repository
public class ItemsDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sf) {
        this.sessionFactory = sf;
    }

    public List<Items> getAllItems() {
        Session session = this.sessionFactory.getCurrentSession();
        List<Items> itemsList = session.createQuery("from Items").list();
        return itemsList;
    }

    public Items getItem(int id) {
        Session session = this.sessionFactory.getCurrentSession();
        Items item = (Items) session.get(Items.class, new Integer(id));
        return item;
    }

    public Items addItem(Items item) {
        Session session = this.sessionFactory.getCurrentSession();
        session.persist(item);
        return item;
    }

    public void updateItem(Items item) {
        Session session = this.sessionFactory.getCurrentSession();
        session.update(item);
    }

    public void deleteItem(int id) {
        Session session = this.sessionFactory.getCurrentSession();
        Items p = (Items) session.load(Items.class, new Integer(id));
        if (null != p) {
            session.delete(p);
        }
    }
}
```

## ● Hibernate

### ✓ 실습 #8(cont'd)

#### – Items 구성

```
package com.kopo.hibernateExample.model;

import javax.persistence.Column;

@Entity
@Table(name="items")
public class Items{

    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    int id;

    @Column(name="itemsName")
    String itemsName;

    @Column(name="price")
    long price;

    public Items() {
        super();
    }

    public Items(int i, String itemsName, long price) {
        super();
        this.id = i;
        this.itemsName = itemsName;
        this.price = price;
    }

    public int getId() {
        return id;
    }

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

    public String getItemsName() {
        return itemsName;
    }

    public void setItemsName(String itemsName) {
        this.itemsName = itemsName;
    }

    public long getPrice() {
        return price;
    }

    public void setPrice(long price) {
        this.price = price;
    }
}
```

## ● Hibernate

### ✓ 실습 #8(cont'd)

#### – ItemsService 구성

```
1 package com.kopo.hibernateExample.server;
2
3 import java.util.List;
4
5 @Service("itemsService")
6 public class ItemsService {
7
8     @Autowired
9     ItemsDAO itemsDao;
10
11     @Transactional
12     public List<Items> getAllItems() {
13         return itemsDao.getAllItems();
14     }
15
16     @Transactional
17     public Items getItem(int id) {
18         return itemsDao.getItem(id);
19     }
20
21     @Transactional
22     public void addItem(Items item) {
23         itemsDao.addItem(item);
24     }
25
26     @Transactional
27     public void updateItem(Items item) {
28         itemsDao.updateItem(item);
29     }
30
31     @Transactional
32     public void deleteItem(int id) {
33         itemsDao.deleteItem(id);
34     }
35 }
```

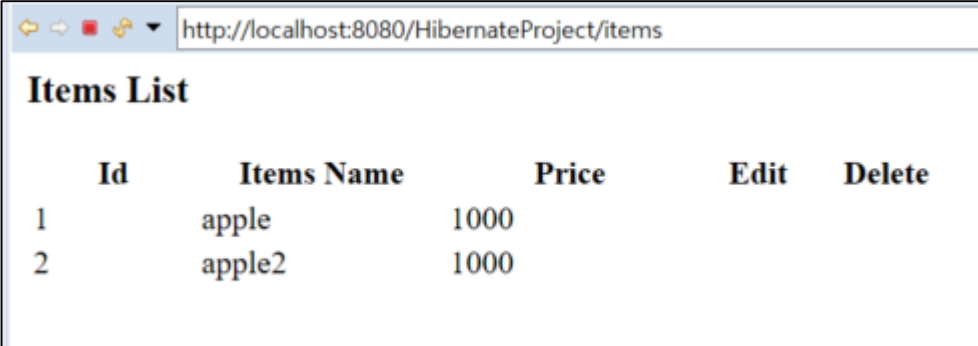
## ● Hibernate

### ✓ 실습 #8

– 예제 코드로 아래와 같이 실행이 됐다면,

1. 아래 화면의 URL을 items가 아닌 첫 실행 화면으로 하기

2. Table 이름을 items가 아닌 다른 이름으로 변경하기



Id	Items Name	Price	Edit	Delete
1	apple	1000		
2	apple2	1000		

## ● 과제 #1

---

- ✓ Spring-Mybatis를 기반으로 게시판 개발(+1일)

## ● 과제 #2

---

- ✓ Spring-Hibernate를 기반으로 게시판 개발(+2일)

## ● 과제 #3

---

- ✓ Spring-Hibernate를 기반으로 프로젝트 개발(+4일)