

1. 개발 단계 유스케이스

가. 메이븐(maven) 프로젝트 만들기

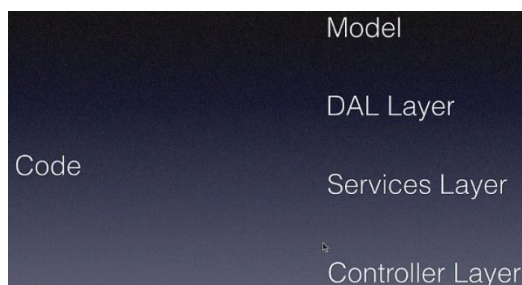
나. DispatcherServlet 설정, POST Form 필드의 한글처리를 위한 필터 지정: web.xml

다. 스프링 설정: dispatcher-servlet.xml

- HibernateTemplate
- SessionFactory
- DataSource
- ViewResolver

라. 코드(Code)

- 개요



- 모델(Model)
- 데이터 액세스 레이어(DAL Layer)
- 서비스 레이어(Services Layer): 인터페이스(interface)와 클래스(classes)의 집합
- 컨트롤러 레이어(Controller Layer)

2. 데이터 베이스에 유저(User) 테이블

```
use mydb;
create table user(id int,name varchar(20),email varchar(30));
select * from user;
```

use mydb;

create table user(id int,name varchar(20),email varchar(30));

select * from user;

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'Filter objects' pane shows a tree view of the database structure. The 'mykhtdb' database is selected, and the 'user' table is highlighted. The main pane shows the 'Query 1' window with the following SQL code:

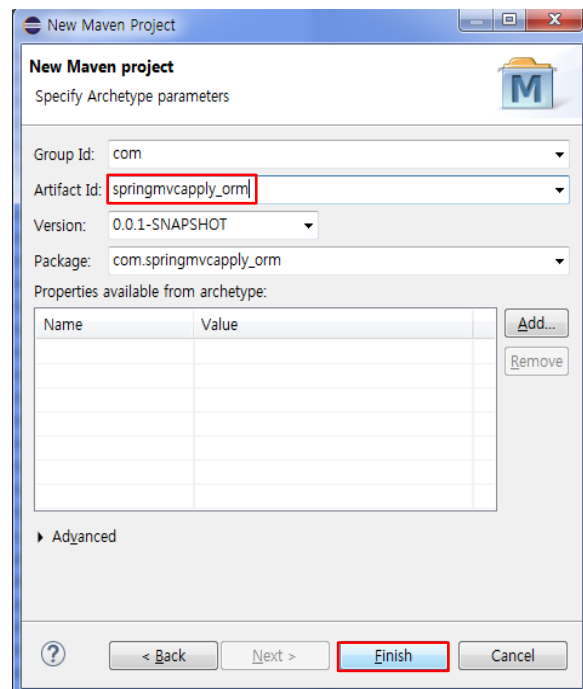
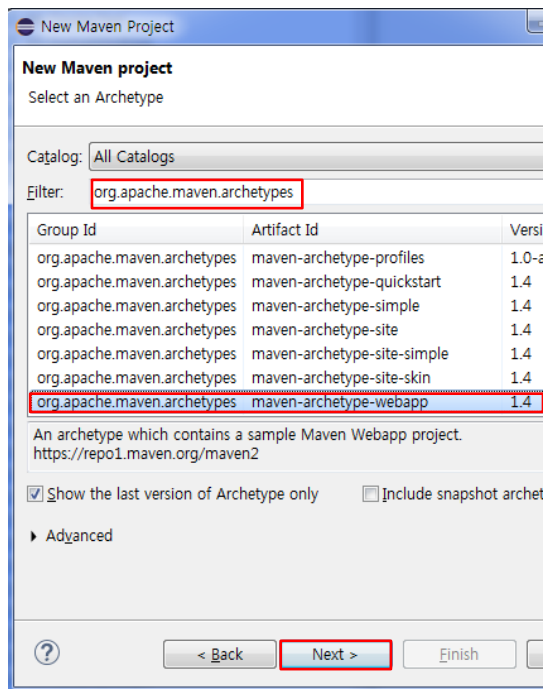
```
use mykhtdb;
create table user(id int,name varchar(20),email varchar(30));
select * from user;
```

The 'Result Grid' shows the columns 'id', 'name', and 'email'. The 'Output' pane shows the 'Action Output' table with the following rows:

#	Time	Action	Message
1	18:09:50	use mykhtdb	0 row(s) affected
2	18:10:02	create table user(id int,name varchar(20),email varchar(30))	0 row(s) affected
3	18:10:06	select * from user LIMIT 0, 1000	0 row(s) returned

3. 메이븐 프로젝트 만들기

가. 메이븐 프로젝트 생성



나. pom.xml: 스프링MVC, 스프링ORM

springmvcapply_orm/pom.xml

```
19 <dependencies>
20 <dependency>
21 <groupId>org.springframework</groupId>
22 <artifactId>spring-webmvc</artifactId>
23 <version>${springframework.version}</version>
24 </dependency>
25
26 <dependency>
27 <groupId>jstl</groupId>
28 <artifactId>jstl</artifactId>
29 <version>1.2</version>
30 </dependency>
31
32 <dependency>
33 <groupId>org.springframework</groupId>
34 <artifactId>spring-orm</artifactId>
35 <version>${springframework.version}</version>
36 </dependency>
37
38 <dependency>
39 <groupId>org.hibernate</groupId>
40 <artifactId>hibernate-core</artifactId>
41 <version>5.4.33.Final</version>
42 </dependency>
43
44
45 <dependency>
46 <groupId>mysql</groupId>
47 <artifactId>mysql-connector-java</artifactId>
48 <version>5.1.6</version>
49 </dependency>
50 <dependency>
51 <groupId>javax.annotation</groupId>
52 <artifactId>javax.annotation-api</artifactId>
53 <version>1.3.2</version>
54 </dependency>
55
56 </dependencies>
57 <build>
58 <pluginManagement>
59 <plugins>
60 <plugin>
61 <groupId>org.apache.maven.plugins</groupId>
62 <artifactId>maven-compiler-plugin</artifactId>
63 <version>3.8.0</version>
64 <configuration>
65 <source>11</source>
66 <target>11</target>
```

스프링MVC 모듈

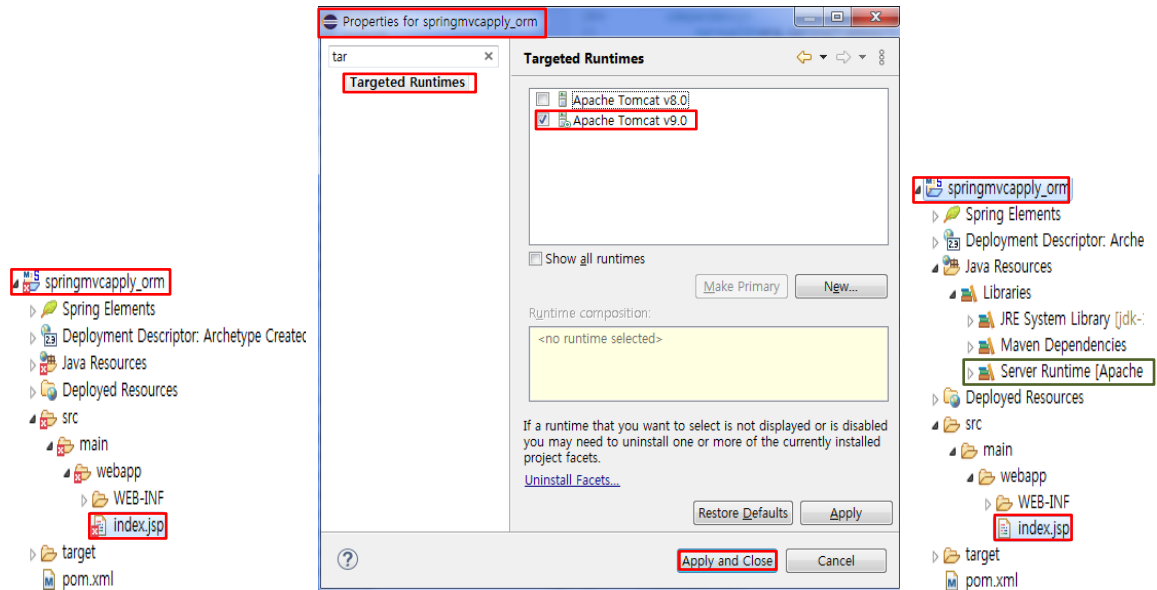
JSP 태그 라이브러리 모듈

스프링ORM 연동모듈

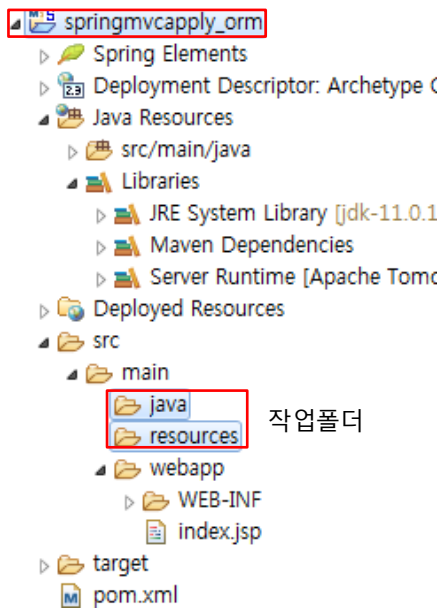
스프링ORM 구현체 모듈

MySQL 접속드라이버

다. JSP 파일 에러: 서블릿 런타임 추가

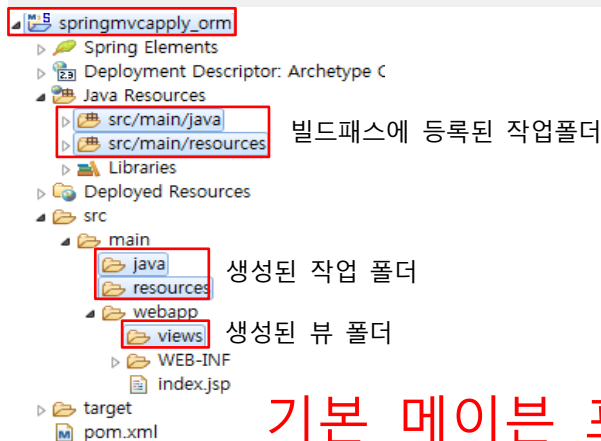
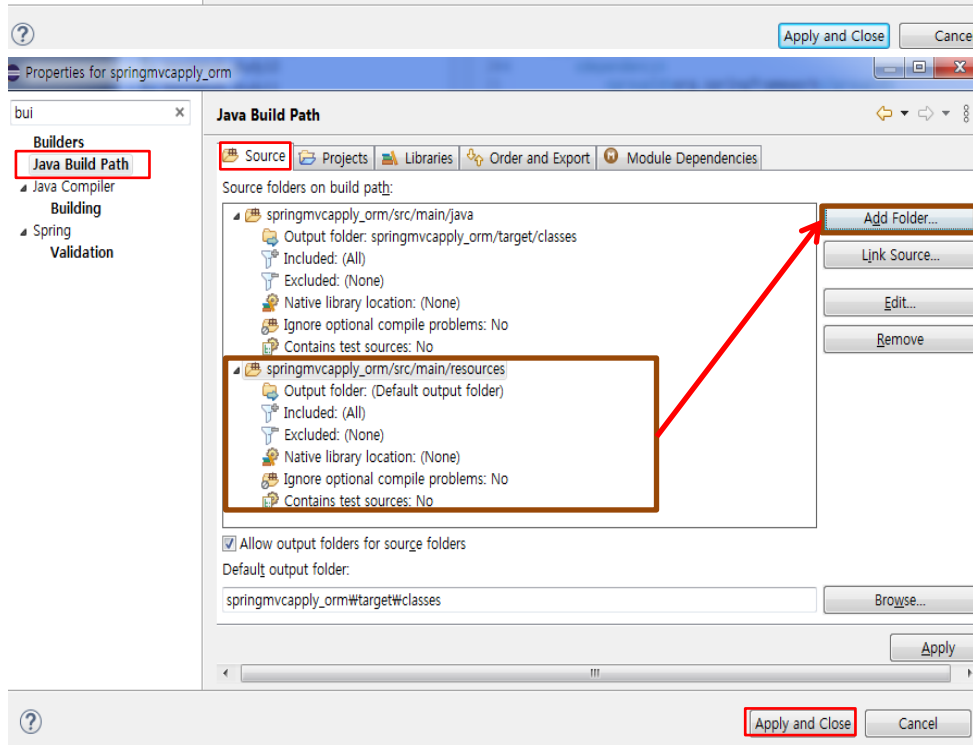
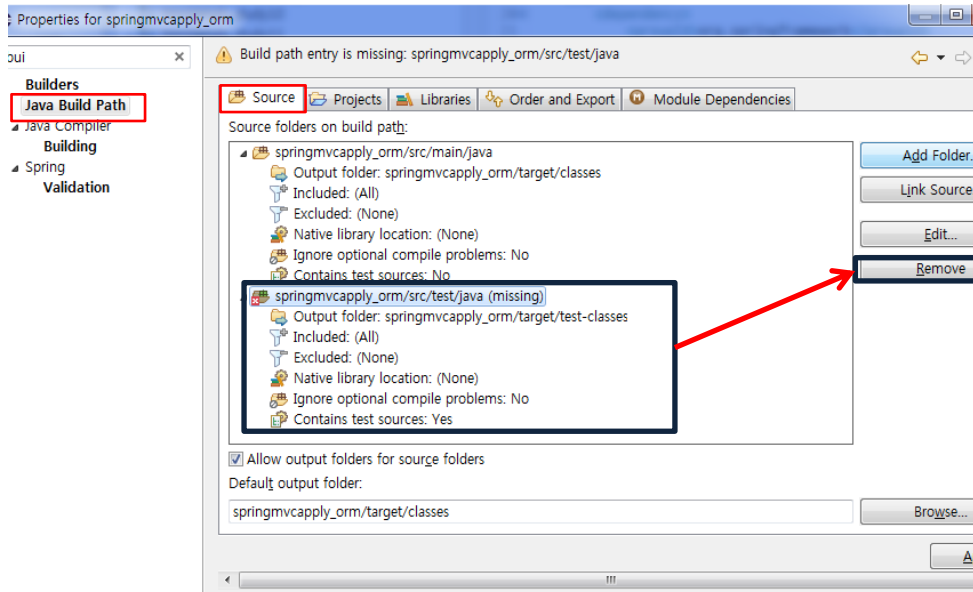


라. 작업 디렉토리 생성: java, resources



작업폴더

마. 작업 디렉토리 빌드 패스 등록



기본 메이븐 프로젝트 생성 완료

4. 프런트 컨트롤러 설정: DispatcherServlet 지정: web.xml

web.xml

DispatcherServlet

The screenshot shows an IDE with the 'springmvcapply_orm' project selected in the left sidebar. The 'web.xml' file is open in the editor. The XML content is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <web-app>
    <display-name>Hello Applied Spring MVC For ORM</display-name>
    <servlet>
      <servlet-name>dispatcher</servlet-name>
      <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
      </servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>dispatcher</servlet-name>
      <url-pattern>/</url-pattern>
    </servlet-mapping>
    <filter>
      <filter-name>encodingFilter</filter-name>
      <filter-class>
        org.springframework.web.filter.CharacterEncodingFilter
      </filter-class>
      <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
      </init-param>
    </filter>
    <filter-mapping>
      <filter-name>encodingFilter</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>
  </web-app>
```

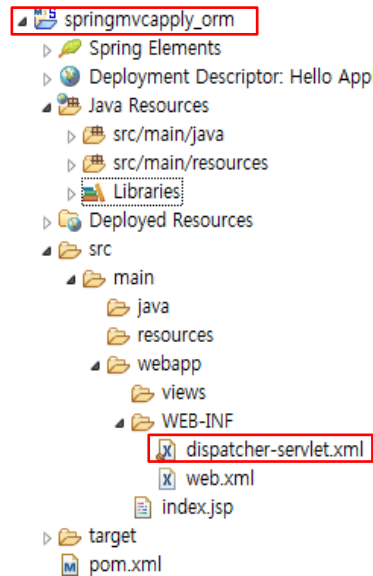
Annotations in the image:

- A red box highlights the 'web.xml' file in the project sidebar.
- A red box highlights the 'web.xml' file in the IDE tab bar.
- A red box highlights the `<servlet>` and `<servlet-mapping>` blocks, with the text 'DispatcherServlet 설정' (DispatcherServlet configuration) next to it.
- A brown box highlights the `<filter>` and `<filter-mapping>` blocks, with the text 'POST Form 필드 한글처리 설정' (POST Form field Korean processing setting) next to it.

5. 스프링 설정 만들기: dispatcher-servlet.xml

The diagram shows a dark background with the text 'dispatcher-servlet.xml' on the left. On the right, several Spring components are listed, each with a small icon representing a bean or configuration element:

- HibernateTemplate
- SessionFactory
- DataSource
- ViewResolver



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context='
4       xmlns:p="http://www.springframework.org/schema/p" xmlns:c="http://www
5       xmlns:tx="http://www.springframework.org/schema/tx"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7       http://www.springframework.org/schema/beans/spring-beans.xsd
8       http://www.springframework.org/schema/context
9       http://www.springframework.org/schema/context/spring-context.xsd
10      http://www.springframework.org/schema/tx
11      http://www.springframework.org/schema/tx/spring-tx.xsd">
12
13
14
15
16
17
18
19
20 </beans>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xmlns:tx="http://www.springframework.org/schema/tx"
8       xsi:schemaLocation="http://www.springframework.org/schema/beans
9       http://www.springframework.org/schema/beans/spring-beans.xsd
10      http://www.springframework.org/schema/context
11      http://www.springframework.org/schema/context/spring-context.xsd
12      http://www.springframework.org/schema/tx
13      http://www.springframework.org/schema/tx/spring-tx.xsd">
14
15     <tx:annotation-driven /> @Transactional 사용
16
17     <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource"
18         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
19         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root" p:password="root" />
20
21     <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
22         name="sessionFactory" p:dataSource-ref="dataSource">
23         <property name="hibernateProperties">
24             <props>
25                 <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
26                 <prop key="hibernate.show_sql">true</prop>
27             </props>
28         </property>
29         <property name="annotatedClasses"> 데이터베이스 테이블과 매핑될 클래스 지정
30             <list>
31                 <value></value>
32             </list>
33         </property>
34     </bean>
35
36     <bean class="org.springframework.orm.hibernate5.HibernateTemplate"
37         name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />
38
39     <bean class="org.springframework.orm.hibernate5.HibernateTransactionManager"
40         name="transactionManager" p:sessionFactory-ref="sessionFactory" />
41
42 </beans>
```

SQL 자동생성 클래스

생성된 SQL 출력여부 지정

6. 뷰 리졸버 설정

```

9      http://www.springframework.org/schema/beans/spring-beans.xsd
10     http://www.springframework.org/schema/context
11     http://www.springframework.org/schema/context/spring-context.xsd
12     http://www.springframework.org/schema/tx
13     http://www.springframework.org/schema/tx/spring-tx.xsd">
14
15     <tx:annotation-driven />
16     <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource"
17         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
18         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root" p:password="root"
19         @Controller
20         @Service
21         @Repository
22         <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
23             name="sessionFactory" p:dataSource-ref="dataSource">
24             <property name="hibernateProperties">
25                 <props>
26                     <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
27                     <prop key="hibernate.show_sql">true</prop>
28                 </props>
29             </property>
30             <property name="annotatedClasses">
31                 <list>
32                     <value></value>
33                 </list>
34             </property>
35         </bean>
36
37         <bean class="org.springframework.orm.hibernate5.HibernateTemplate"
38             name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />
39
40         <bean class="org.springframework.orm.hibernate5.HibernateTransactionManager"
41             name="transactionManager" p:sessionFactory-ref="sessionFactory" />
42
43         <bean
44             class="org.springframework.web.servlet.view.InternalResourceViewResolver"
45             name="viewResolver">
46             <property name="prefix">
47                 <value>/WEB-INF/jsp</value>
48             </property>
49
50             <property name="suffix">
51                 <value>.jsp</value>
52             </property>
53         </bean>
54     </beans>

```

아래 애노테이션을 찾는 위치 지정

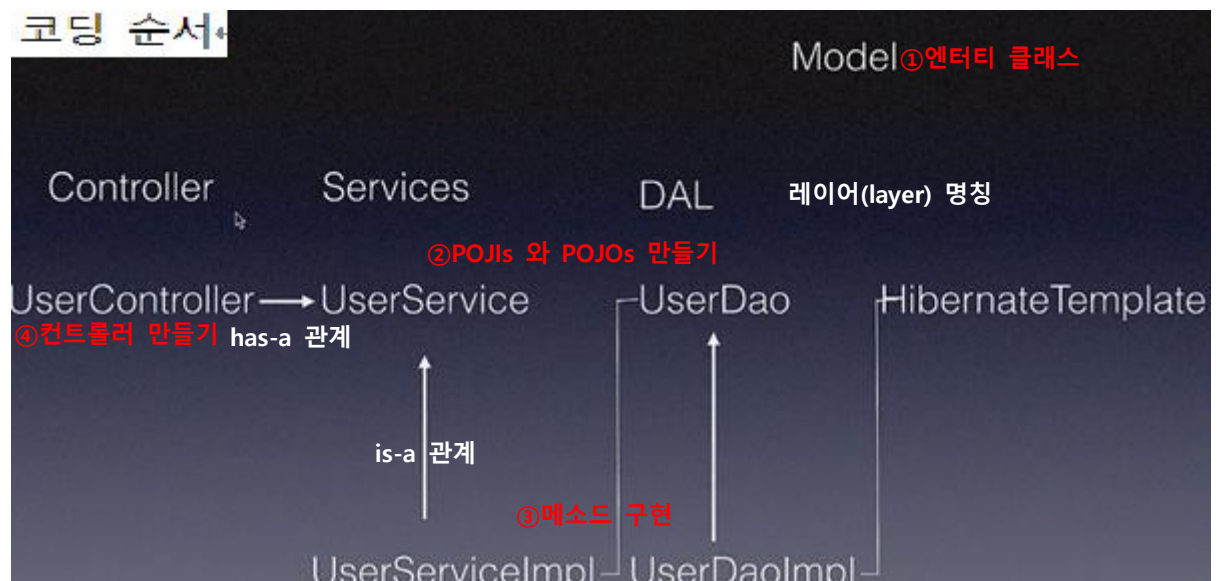
@Controller

@Service

@Repository

7. 코딩 순서

코딩 순서



8. 모델 만들기

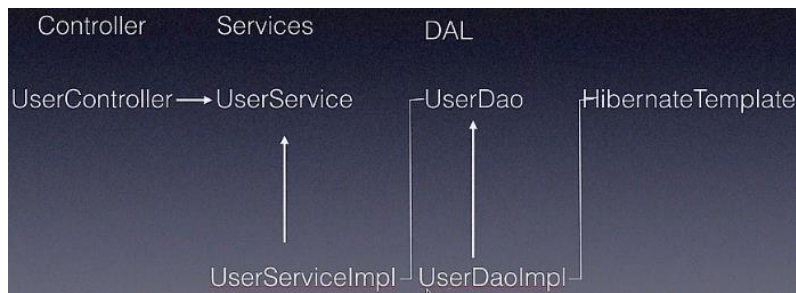
User.java

```
1 package com.springmvcapply_orm.user.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Id;
5 import javax.persistence.Table;
6
7 @Entity
8 @Table(name = "user") 클래스명은 User, 테이블명은 user라 @Table 애노테이션 사용
9 public class User {
10
11     @Id 기본키 지정
12     private int id;
13     private String name;
14     private String email;
15
16     public int getId() {
17         return id;
18     }
19
20     public void setId(int id) {
21         this.id = id;
22     }
23
24     public String getName() {
25         return name;
26     }
27
28     public void setName(String name) {
29         this.name = name;
30     }
31
32     public String getEmail() {
33         return email;
34     }
35
36     public void setEmail(String email) {
37         this.email = email;
38     }
39
40
41     @Override
42     public String toString() {
43         return "User [id=" + id + ", name=" + name + ", email=" + email + "];"
44     }
45 }
```

dispatcher-servlet.xml

```
21 <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
22       name="sessionFactory" p:dataSource-ref="dataSource">
23     <property name="hibernateProperties">
24         <props>
25             <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
26             <prop key="hibernate.show_sql">true</prop>
27         </props>
28     </property>
29     <property name="annotatedClasses">
30         <list>
31             <value>com.springmvcapply_orm.user.entity.User</value>
32         </list>
33     </property>
34 </bean>
```


9. POJIs와 POJOs 만들기



가. 궁금증

`@Component`로 다 쓰지 왜 굳이 `@Repository`, `@Service`, `@Controller` 등을 사용하냐면 예를 들어 `@Repository`는 DAO의 메서드에서 발생할 수 있는 unchecked exception들을 스프링의 `DataAccessException`으로 처리할 수 있기 때문이다.

또한 가독성에서도 해당 애노테이션을 갖는 클래스가 무엇을 하는지 단 번에 알 수 있다.

그리고 그렇게 작성하면 자동으로 등록되는 빈의 이름은 클래스의 첫 문자가 소문자로 바뀐 이름이 자동 적용된다. (`HomeController` -> `homeController`)

나. DAL 레이어 준비

```

1 package com.springmvcapply_orm.user.dao;
2
3 public interface UserDao {
4
5
6 }
  
```

```

1 package com.springmvcapply_orm.user.dao.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Repository
6 public class UserDaoImpl implements UserDao{
7
8     @Autowired
9     private HibernateTemplate hibernateTemplate;
10
11     public HibernateTemplate getHibernateTemplate() {
12         return hibernateTemplate;
13     }
14
15     public void setHibernateTemplate(
16         HibernateTemplate hibernateTemplate) {
17         this.hibernateTemplate = hibernateTemplate;
18     }
19 }
  
```

다. 서비스 레이어 준비

```

1 package com.springmvcapply_orm.user.service;
2
3 public interface UserService {
4
5
6 }
  
```

```

1 package com.springmvcapply_orm.user.service.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Service
6 public class UserServiceImpl implements UserService {
7
8     @Autowired
9     private UserDao dao;
10
11     public UserDao getDao() {
12         return dao;
13     }
14
15     public void setDao(UserDao dao) {
16         this.dao = dao;
17     }
18 }
  
```

라. 컨트롤러 준비

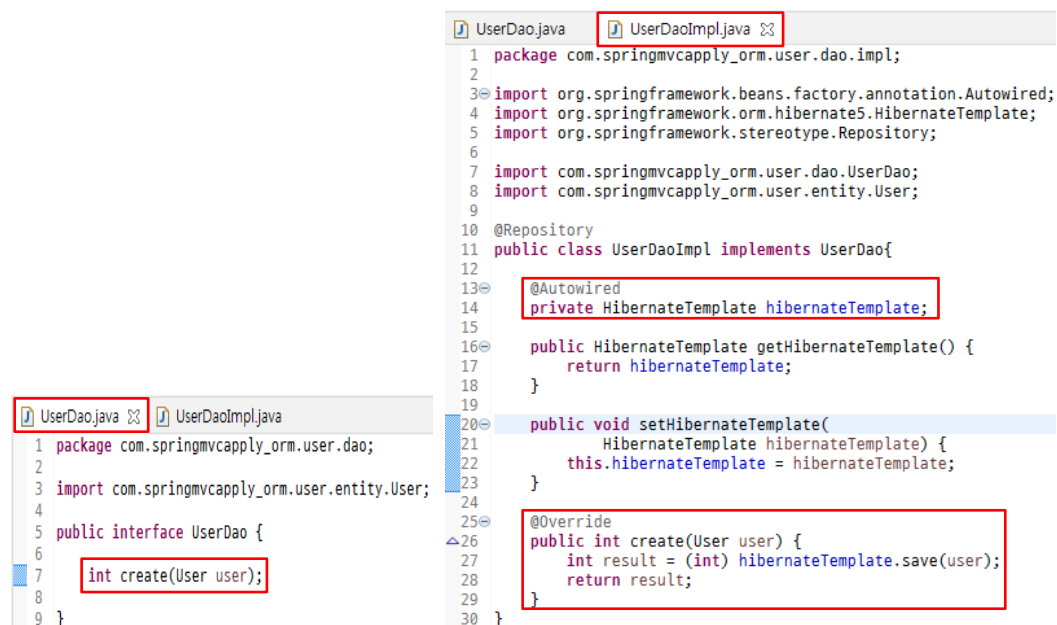


The screenshot shows the project structure on the left and the code for `UserController.java` on the right. The project structure includes `springmvcapply_orm`, `Spring Elements`, `Deployment Descriptor: <web app>`, `Java Resources`, `src/main/java`, `com.springmvcapply_orm.user.controller`, and `UserController.java`. The code for `UserController.java` is as follows:

```
1 package com.springmvcapply_orm.user.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Controller
6 public class UserController {
7
8     @Autowired
9     private UserService service;
10
11     public UserService getService() {
12         return service;
13     }
14
15     public void setService(UserService service) {
16         this.service = service;
17     }
18 }
19
20
21
22 }
```

10. DAO와 Service 메소드 구현

가. DAO



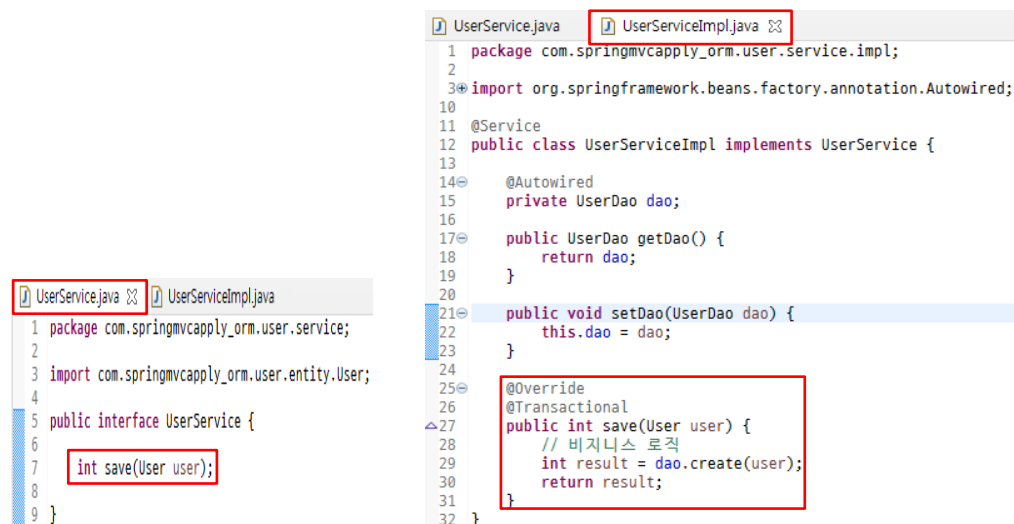
The screenshot shows the code for `UserDao.java` and `UserDaoImpl.java`. The code for `UserDao.java` is as follows:

```
1 package com.springmvcapply_orm.user.dao;
2
3 import com.springmvcapply_orm.user.entity.User;
4
5 public interface UserDao {
6
7     int create(User user);
8
9 }
```

The code for `UserDaoImpl.java` is as follows:

```
1 package com.springmvcapply_orm.user.dao.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.orm.hibernate5.HibernateTemplate;
5 import org.springframework.stereotype.Repository;
6
7 import com.springmvcapply_orm.user.dao.UserDao;
8 import com.springmvcapply_orm.user.entity.User;
9
10 @Repository
11 public class UserDaoImpl implements UserDao {
12
13     @Autowired
14     private HibernateTemplate hibernateTemplate;
15
16     public HibernateTemplate getHibernateTemplate() {
17         return hibernateTemplate;
18     }
19
20     public void setHibernateTemplate(
21         HibernateTemplate hibernateTemplate) {
22         this.hibernateTemplate = hibernateTemplate;
23     }
24
25     @Override
26     public int create(User user) {
27         int result = (int) hibernateTemplate.save(user);
28         return result;
29     }
30 }
```

나. Service



The screenshot shows the code for `UserService.java` and `ServiceImpl.java`. The code for `UserService.java` is as follows:

```
1 package com.springmvcapply_orm.user.service;
2
3 import com.springmvcapply_orm.user.entity.User;
4
5 public interface UserService {
6
7     int save(User user);
8
9 }
```

The code for `ServiceImpl.java` is as follows:

```
1 package com.springmvcapply_orm.user.service.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Service
6 public class UserServiceImpl implements UserService {
7
8     @Autowired
9     private UserDao dao;
10
11     public UserDao getDao() {
12         return dao;
13     }
14
15     public void setDao(UserDao dao) {
16         this.dao = dao;
17     }
18
19     @Override
20     @Transactional
21     public int save(User user) {
22         // 비즈니스 로직
23         int result = dao.create(user);
24         return result;
25     }
26 }
```

11. 컨트롤러(Controller) 메소드 구현

```
UserController.java
1 package com.springmvcapply_orm.user.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.ModelMap;
6 import org.springframework.web.bind.annotation.ModelAttribute;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RequestMethod;
9
10 import com.springmvcapply_orm.user.entity.User;
11 import com.springmvcapply_orm.user.service.UserService;
12
13 @Controller
14 public class UserController {
15
16     @Autowired
17     private UserService service;
18
19     @RequestMapping("registrationPage")
20     public String showRegistrationPage()
21     {
22         return "userReg";
23     }
24
25     @RequestMapping(value = "registerUser", method = RequestMethod.POST)
26     public String registerUser(@ModelAttribute("user") User user,
27                               ModelMap model)
28     {
29         int result = service.save(user);
30         model.addAttribute("result", "아이디로 생성된 사용자 " + result);
31         return "userReg";
32     }
33
34     public UserService getService() {
35         return service;
36     }
37
38     public void setService(UserService service) {
39         this.service = service;
40     }
41
42 }
43
44 }
```

12. 결과를 만들고 보여주기

```
userReg.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8" isELIgnored="false" %>
3 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>User Registration</title>
9 </head>
10 <body>
11   <form action="registerUser" method="post">
12     <pre>
13       Id: <input type="text" name="id" />
14       Name: <input type="text" name="name" />
15       Email: <input type="text" name="email" />
16       <input type="submit" name="register" />
17     </pre>
18   </form>
19
20   <br />${result}
21
22 </body>
23 </html>
```

13. 스프링 설정에서 애노테이션 활성화

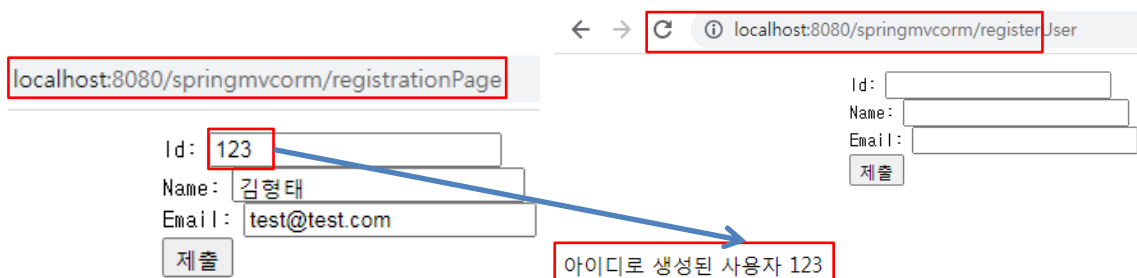
```
dispatcher-servlet.xml | UserController.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xmlns:tx="http://www.springframework.org/schema/tx"
8       xsi:schemaLocation="http://www.springframework.org/schema/beans
9                           http://www.springframework.org/schema/beans/spring-beans.xsd
10                          http://www.springframework.org/schema/context
11                          http://www.springframework.org/schema/context/spring-context.xsd
12                          http://www.springframework.org/schema/tx
13                          http://www.springframework.org/schema/tx/spring-tx.xsd">
14
15     <context:component-scan base-package="com.springmvcapply_orm.user" />
16
17     <tx:annotation-driven />
```

14. 스프링 설정에서 엔티티 네임 업데이트

```
dispatcher-servlet.xml
18
19 <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource"
20       name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
21       p:url="jdbc:mysql://localhost/mykhtdb" p:username="root" p:password="" />
22
23 <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
24       name="sessionFactory" p:dataSource-ref="dataSource">
25     <property name="hibernateProperties">
26       <props>
27         <prop key="hibernate.dialect">org.hibernate.dialect.MySQL
28         <prop key="hibernate.show_sql">true</prop>
29       </props>
30     </property>
31     <property name="annotatedClasses">
32       <list>
33         <value>com.springmvcapply_orm.user.entity.User</value>
34       </list>
35     </property>
36 </bean>
```

15. 테스트

가. 웹페이지



나. 데이터 베이스

User [id=123, name=김형태, email=test@test.com]

Hibernate: insert into user (email, name, id) values (?, ?, ?)

4월 21, 2023 4:37:09 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

WARN: SQL Error: 1366, SQLState: HY000

4월 21, 2023 4:37:09 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

ERROR: Incorrect string value: 'WxEAwx9B9Wx80WxEDWx98Wx95...' for column 'name' at row

다. 컨트롤러 소스 변경

```
UserController.java
15
16 @Controller
17 public class UserController {
18
19     @Autowired
20     private UserService service;
21
22     @RequestMapping("registrationPage")
23     public String showRegistrationPage()
24     {
25         return "userReg";
26     }
27
28
29     @RequestMapping(value = "registerUser", method = RequestMethod.POST)
30     public String registerUser(@ModelAttribute("user") User user,
31                               @ModelAttribute("result") ModelMap model) throws UnsupportedEncodingException
32     {
33         System.out.println(user);
34
35         user.setName(URLDecoder.decode(user.getName(), "utf-8"));
36
37         int result = service.save(user);
38         model.addAttribute("result", "아이디로 생성된 사용자 " + result);
39
40         return "userReg";
41     }
42
43
44     public UserService getService() {
45         return service;
46     }
47
48     public void setService(UserService service) {
49         this.service = service;
50     }
51 }
```

localhost:8080/springmvcorm/registrationPage

Id:

Name:

Email:

User [id=123, name=김형태, email=test@test.com]

김형태

Hibernate: insert into user (email, name, id) values (?, ?, ?)

4월 21, 2023 5:00:26 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

WARN: SQL Error: 1366, SQLState: HY000

4월 21, 2023 5:00:26 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

ERROR: Incorrect string value: 'WxEA WB9Wx80WxEDWx98Wx95...' for column 'name' at row 1

```

UserController.java
30 import java.io.UnsupportedEncodingException;
16
17 @Controller
18 public class UserController {
19
20     @Autowired
21     private UserService service;
22
23     @RequestMapping("registrationPage")
24     public String showRegistrationPage()
25     {
26         return "userReg";
27     }
28
29
30     @RequestMapping(value = "registerUser", method = RequestMethod.POST)
31     public String registerUser(@ModelAttribute("user") User user,
32                               ModelMap model) throws UnsupportedEncodingException
33     {
34         System.out.println(user);
35
36         String kht1 = URLDecoder.decode(user.getName(), "utf-8");
37         String kht2 = URLEncoder.encode(kht1, "utf-8");
38         user.setName(kht2);
39
40         int result = service.save(user);
41         model.addAttribute("result", "아이디로 생성된 사용자 " + result);
42
43         return "userReg";
44     }
45
46     public UserService getService() {
47         return service;
48     }
49
50     public void setService(UserService service) {
51         this.service = service;
52     }
53
54 }

```

User [id=123, name=김형태, email=test@test.com]

Hibernate: insert into user (email, name, id) values (?, ?, ?)

4월 21, 2023 5:06:03 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

WARN: SQL Error: 0, SQLState: 22001

4월 21, 2023 5:06:03 오전 org.hibernate.engine.jdbc.spi.SqlExceptionHelper logExceptions

ERROR: Data truncation: Data too long for column 'name' at row 1

4월 21, 2023 5:06:03 오전 org.apache.catalina.core.StandardWrapperValve invoke

SEVERE: 경로 [/springmvcorm]의 컨텍스트 내의 서블릿 [dispatcher]을(를) 위한 Servlet.service() 호출이, 근본

원인(root cause)과 함께, 예외 [Request processing failed; nested exception is

[org.springframework.dao.DataIntegrityViolationException](#): could not execute statement; SQL [n/a]; nested exception is

[org.hibernate.exception.DataException](#): could not execute statement]을(를) 발생시켰습니다.

com.mysql.jdbc.MySQLDataTruncation: Data truncation: Data too long for column 'name' at row 1

```

@RequestMapping(value = "registerUser", method = RequestMethod.POST)
public String registerUser(@ModelAttribute("user") User user,
                           ModelMap model) throws UnsupportedEncodingException
{
    int result = service.save(user);
    model.addAttribute("result", "아이디로 생성된 사용자 " + result);

    return "userReg";
}

```

localhost:8080/springmvcorm/registrationPage

Id: 123
 Name: woseven
 Email: test@test.com
 제출

SELECT * FROM `user` u;		
Refresh		
t 1		
id	name	email
123	woseven	test@test.com

라. POST 방식으로 입력 박스에 한글 입력할 때 깨지는 것을 방지

```
UserController.java web.xml
1 <!DOCTYPE web-app PUBLIC
2 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
3 "http://java.sun.com/dtd/web-app_2_3.dtd" >
4
5 <web-app>
6   <display-name>Hello Spring MVC</display-name>
7
8   <servlet>
9     <servlet-name>dispatcher</servlet-name>
10    <servlet-class>org.springframework.web.servlet.DispatcherSer
11  </servlet>
12
13  <servlet-mapping>
14    <servlet-name>dispatcher</servlet-name>
15    <url-pattern>/</url-pattern>
16  </servlet-mapping>
17
18  <filter>
19    <filter-name>encodingFilter</filter-name>
20    <filter-class>
21      org.springframework.web.filter.CharacterEncodingFilter
22    </filter-class>
23    <init-param>
24      <param-name>encoding</param-name>
25      <param-value>UTF-8</param-value>
26    </init-param>
27  </filter>
28
29  <filter-mapping>
30    <filter-name>encodingFilter</filter-name>
31    <url-pattern>/</url-pattern>
32  </filter-mapping>
33 </web-app>
```

← → ↻ ⓘ localhost:8080/springmvcorm/registerUser

localhost:8080/springmvcorm/registrationPage

Id:

Name:

Email:

Id:

Name:

Email:

아이디로 생성된 사용자 234

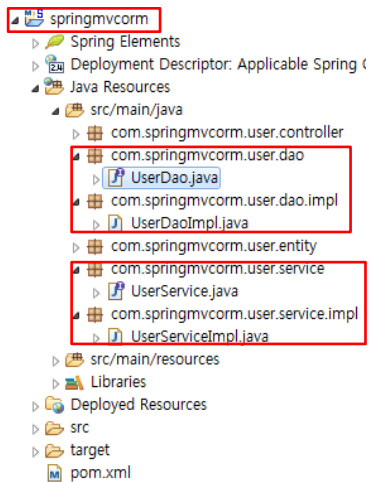
refresh

SELECT * FROM `user` u;

1

id	name	email
123	woseven	test@test.com
234	김형태	test1@test.com

16. 모든 유저 불러오는 DAO와 서비스 구현



```
1 package com.springmvcorm.user.dao;
2
3 import java.util.List;
4
5
6 public interface UserDao {
7
8     int create(User user);
9
10    List<User> findUsers();
11
12 }
13
```

```
1 package com.springmvcorm.user.dao.impl;
2
3 import java.util.List;
4
5 @Repository
6 public class UserDaoImpl implements UserDao {
7
8     @Autowired
9     private HibernateTemplate hibernateTemplate;
10
11     public HibernateTemplate getHibernateTemplate() {
12         return hibernateTemplate;
13     }
14
15     public void setHibernateTemplate(
16         HibernateTemplate hibernateTemplate) {
17         this.hibernateTemplate = hibernateTemplate;
18     }
19
20     @Override
21     public int create(User user) {
22         int result = (int) hibernateTemplate.save(user);
23         return result;
24     }
25
26     @Override
27     public List<User> findUsers() {
28         List<User> users = hibernateTemplate.loadAll(User.class);
29         return users;
30     }
31 }
32
```

UserService.java

```

1 package com.springmvcorm.user.service;
2
3 import java.util.List;
4
5
6 public interface UserService {
7
8     int save(User user);
9
10    List<User> getUsers();
11
12 }

```

UserServiceImpl.java

```

1 package com.springmvcorm.user.service.impl;
2
3 import java.util.List;
4
5
6 @Service
7 public class UserServiceImpl implements UserService {
8
9     @Autowired
10    private UserDao dao;
11
12    public UserDao getDao() {
13        return dao;
14    }
15
16    public void setDao(UserDao dao) {
17        this.dao = dao;
18    }
19
20    @Override
21    @Transactional
22    public int save(User user) {
23        // 비즈니스 로직
24        int result = dao.create(user);
25        return result;
26    }
27
28    @Override
29    public List<User> getUsers() {
30        List<User> users = dao.findUsers();
31        return users;
32    }
33 }

```

17. 모든 유저 불러오기 컨트롤러 구현

UserController.java

```

24 {
25     return "userReg";
26 }
27
28
29 @RequestMapping(value = "registerUser", method = RequestMethod.POST)
30 public String registerUser(@ModelAttribute("user") User user,
31                             ModelMap model) throws UnsupportedEncodingException {
32     int result = service.save(user);
33     model.addAttribute("result", "아이디로");
34     return "userReg";
35 }
36
37
38
39
40 @RequestMapping("getUsers")
41 public String getUsers(ModelMap model) {
42     List<User> users = service.getUsers();
43     model.addAttribute("users", users);
44
45     return "displayUsers";
46 }
47
48 }

```

18. JSTL을 이용한 뷰 페이지 만들기

displayUsers.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8" isELIgnored="false" %>
3 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="UTF-8">
8     <title>User Registration</title>
9   </head>
10  <body>
11
12    <c:forEach items="${users}" var="user">
13      ${user.id}
14      ${user.name}
15      ${user.email}
16    </c:forEach>
17
18  </body>
19 </html>

```

Maven Depend

spring-webmv

spring-aop-4.3

spring-beans-4

spring-context

spring-core-4.1

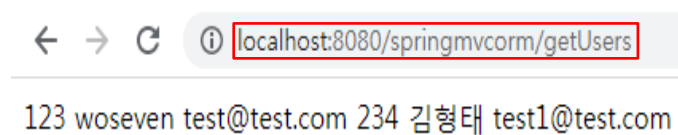
commons-log

spring-express

spring-web-4.1

jstl-1.2.jar - C

19. 테스트: 문제가 발생하면, Servlet Runtime을 체크해제고 적용, 다시 체크하고 적용



displayUsers.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8" isELIgnored="false" %>
3 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4 <!DOCTYPE html>
5 <html>
6   <head>
7     <meta charset="UTF-8">
8     <title>User Registration</title>
9   </head>
10  <body>
11    <table border="1">
12      <tr>
13        <th>id</th>
14        <th>name</th>
15        <th>email</th>
16      </tr>
17      <c:forEach items="${users}" var="user">
18        <tr>
19          <td>${user.id}</td>
20          <td>${user.name}</td>
21          <td>${user.email}</td>
22        </tr>
23      </c:forEach>
24    </table>
25  </body>
26 </html>

```

← → ↻ ⓘ localhost:8080/springmvcorm/getUsers

id	name	email
123	woseven	test@test.com
234	김형태	test1@test.com

20. ID로 정렬

```

1 package com.springmvcorm.user.entity;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Id;
5 import javax.persistence.Table;
6
7 @Entity
8 @Table(name = "user")
9 public class User implements Comparable<User>{
10
11     @Id
12     private Integer id;
13     private String name;
14     private String email;
15
16     public int getId() {
17         return id;
18     }
19
20
21     public void setId(int id) {
22         this.id = id;
23     }
24
25     public String getName() {
26         return name;
27     }
28
29     public void setName(String name) {
30         this.name = name;
31     }
32
33     public String getEmail() {
34         return email;
35     }
36
37     public void setEmail(String email) {
38         this.email = email;
39     }
40
41     @Override
42     public String toString() {
43         return "User [id=" + id + ", name=" + name + ", email=" + email + "];"
44     }
45
46     @Override
47     public int compareTo(User user) {
48         return this.id.compareTo(user.id);
49     }
50 }

```

localhost:8080/springmvcorm/registrationPage

Id: 33
 Name:
 Email: www@test.com
 제출

Refresh

SELECT * FROM `user` u;

ResultSet 1

id	name	email
123	woseven	test@test.com
234	김형태	test1@test.com
33		www@test.com

localhost:8080/springmvcorm/getUsers

id	name	email
123	woseven	test@test.com
234	김형태	test1@test.com
33		www@test.com

바뀌지 않고 그대로네???

```

UserController.java  UserServiceImpl.java
1 package com.springmvcorm.user.service.impl;
2
3 import java.util.Collections;
13
14 @Service
15 public class UserServiceImpl implements UserService {
16
17     @Autowired
18     private UserDao dao;
19
20     public UserDao getDao() {
21         return dao;
22     }
23
24     public void setDao(UserDao dao) {
25         this.dao = dao;
26     }
27
28     @Override
29     @Transactional
30     public int save(User user) {
31         // 비즈니스 로직
32         int result = dao.create(user);
33         return result;
34     }
35
36     @Override
37     public List<User> getUsers() {
38         List<User> users = dao.findUsers();
39         Collections.sort(users);
40         return users;
41     }
42 }

```

데이터 베이스 테이블에는 123, 234, 33순으로 저장되어
 결과를 보여주는 displayUsers.jsp 파일은
 33, 123, 234 순으로 출력

localhost:8080/springmvcorm/getUsers

SELECT * FROM `user` u;

Refresh

id	name	email
33		www@test.com
123	woseven	test@test.com
234	김형태	test1@test.com

ResultSet 1

id	name	email
123	woseven	test@test.com
234	김형태	test1@test.com
33		www@test.com