1. 개발 단계 유스케이스

가. 메이븐(maven) 프로젝트 만들기

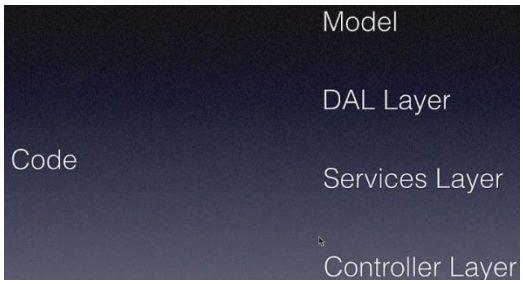
나. DispatcherServleIt 설정, POST Form 필드의 한글처리를 위한 필터 지정: web.xml

다. 스프링 설정: dispatcher-servlet.xml

- HibernateTemplate
- SessionFactory
- DataSource
- ViewResolver

라. 코드(Code)

- 개요



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

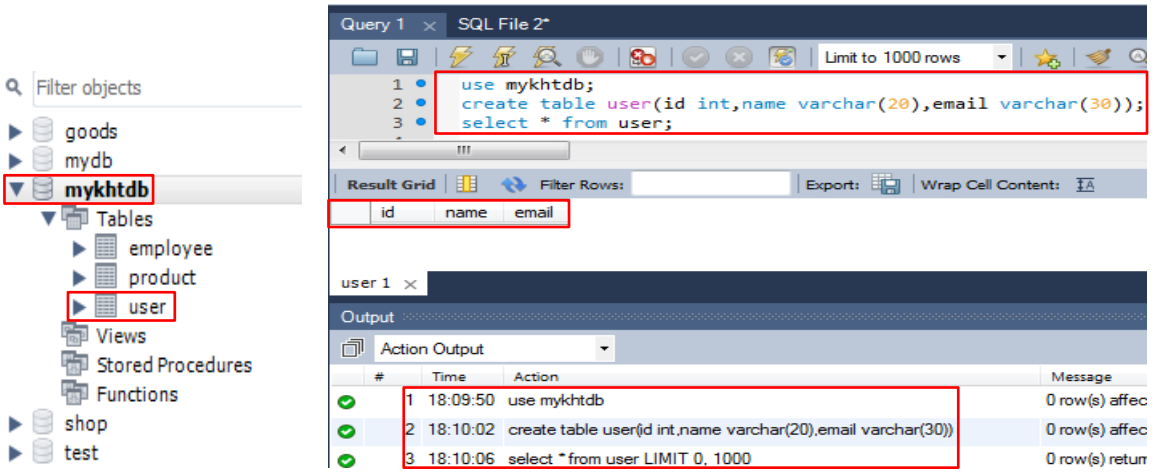
2. 데이터 베이스에 유저(User) 테이블

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

use mydb;

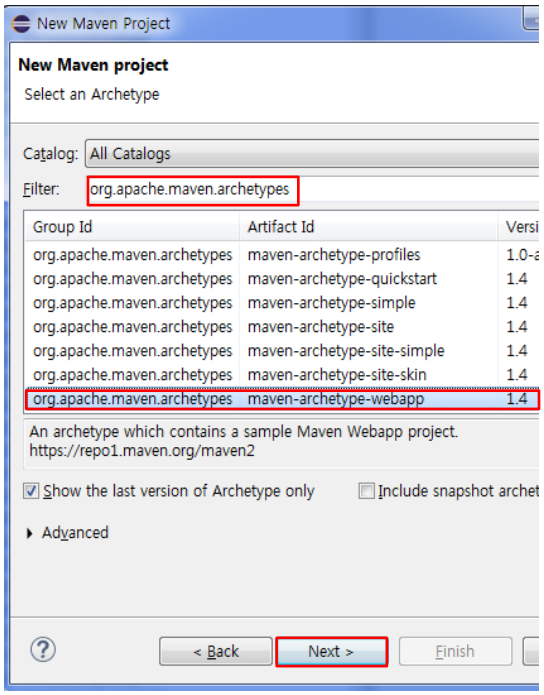
create table user(id int,name varchar(20),email varchar(30));

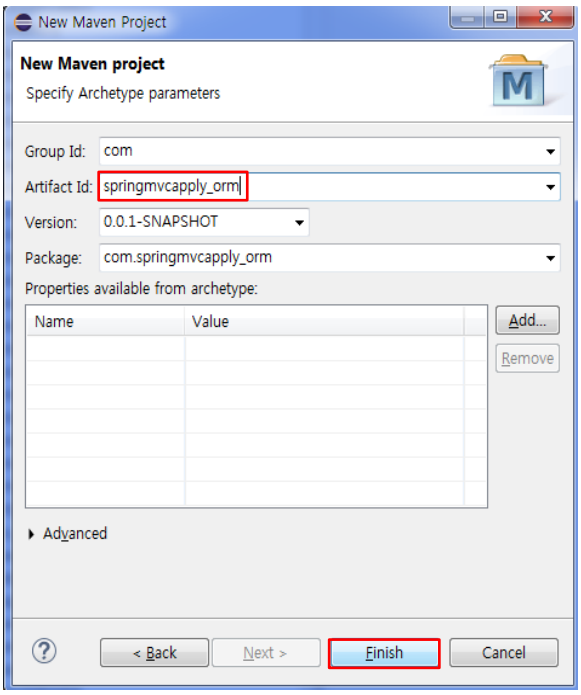
select * from user;



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

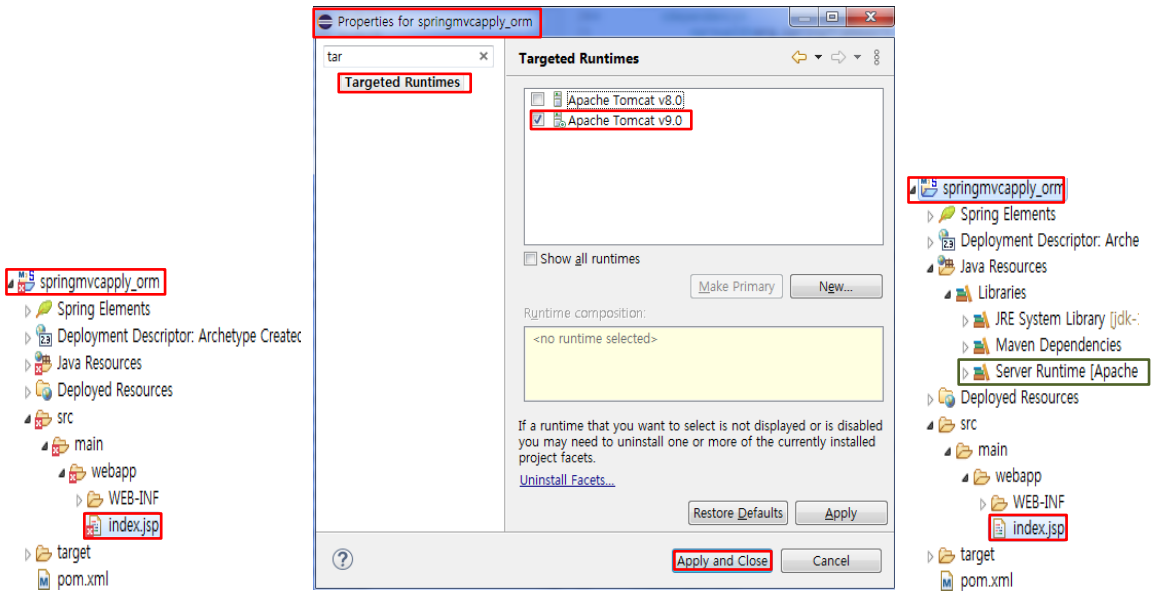
JSP 태그 라이브러리 모듈

스프링ORM 연동모듈

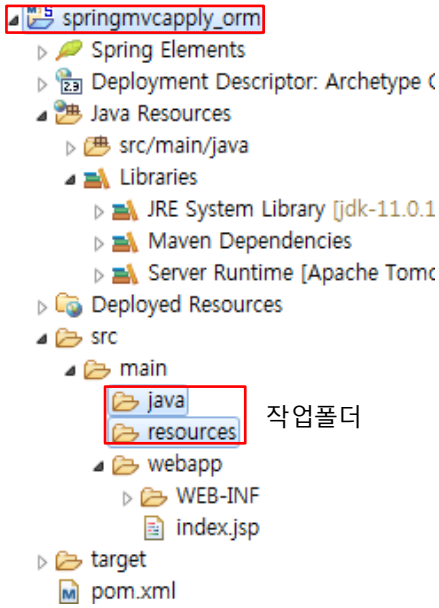
스프링ORM 구현체 모듈

MySQL 접속드라이버

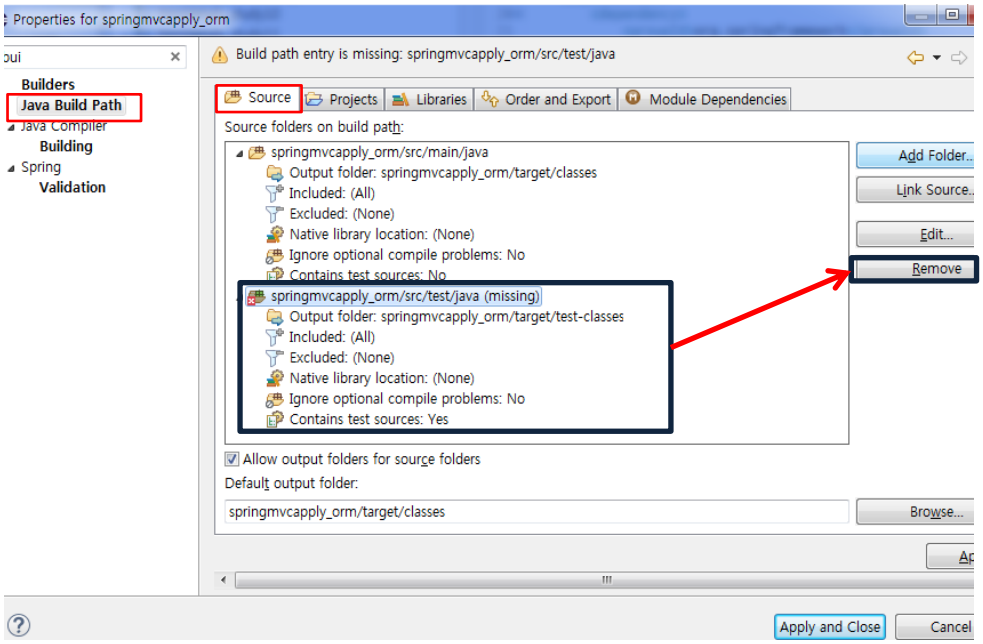
다. JSP 파일 에러: 서블릿 런타임 추가

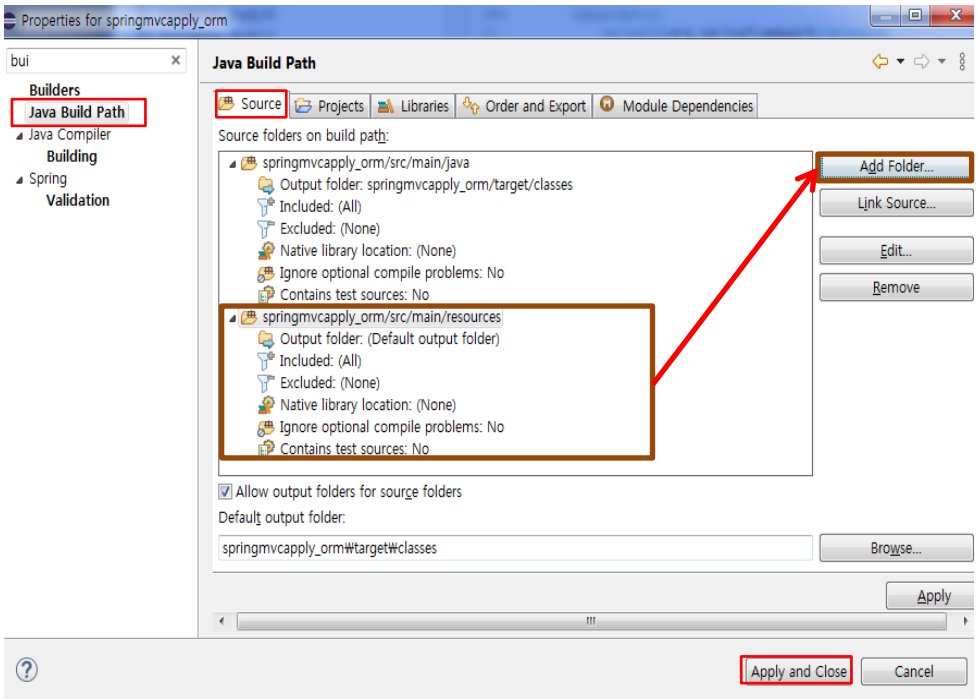


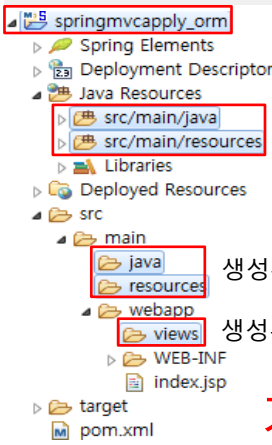
라. 작업 디렉토리 생성: java, resources



마. 작업 디렉토리 빌드 패스 등록







빌드패스에 등록된 작업폴더

생성된 작업 폴더

생성된 뷰 폴더

기본 메이븐 프로젝트 생성 완료

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

web.xml DispatcherServlet

springmvcapply_orm/pom.xml | web.xml

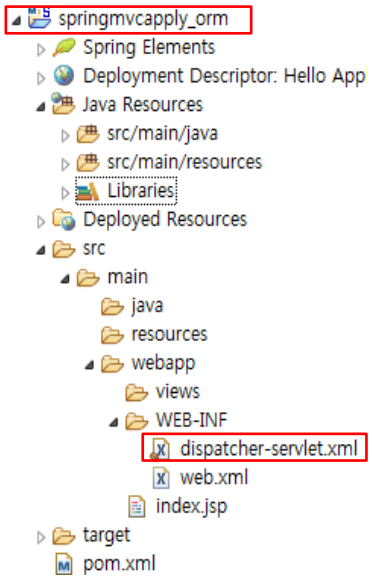
```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6      https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
7
8  <web-app>
9    <display-name>Hello Applied Spring MVC For ORM</display-name>
10
11     <servlet>
12         <servlet-name>dispatcher</servlet-name>
13         <servlet-class>
14             org.springframework.web.servlet.DispatcherServlet
15         </servlet-class>
16     </servlet>
17
18     <servlet-mapping>
19         <servlet-name>dispatcher</servlet-name>
20         <url-pattern>/</url-pattern>
21     </servlet-mapping>
22
23     <filter>
24         <filter-name>encodingFilter</filter-name>
25         <filter-class>
26             org.springframework.web.filter.CharacterEncodingFilter
27         </filter-class>
28         <init-param>
29             <param-name>encoding</param-name>
30             <param-value>UTF-8</param-value>
31         </init-param>
32     </filter>
33
34     <filter-mapping>
35         <filter-name>encodingFilter</filter-name>
36         <url-pattern>/*</url-pattern>
37     </filter-mapping>
38
39 </web-app>
```

DispatcherServlet 설정

POST Form 필드 한글처리 설정

- springmvcapply_orm
  - Spring Elements
  - Deployment Descriptor: Hello Appl
  - Java Resources
    - src/main/java
    - src/main/resources
    - Libraries
  - Deployed Resources
  - src
    - main
      - java
      - resources
      - webapp
        - views
        - WEB-INF
          - web.xml
        - index.jsp
  - target
  - pom.xml

5. 스프링 설정 만들기: dispatcher-servlet.xml



dispatcher-servlet.xml

```
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>
```

dispatcher-servlet.xml

```
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>                                          SQL 자동생성 클래스
25                 <prop key="hibernate.dialet">org.hibernate.dialect.MySQLDialect</prop>
26                 <prop key="hibernate.show_sql">true</prop>
27             </props>        생성된 SQL 출력여부 지정
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>
```

6. 뷰 리졸버 설정

dispatcher-servlet.xml

```xml
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <tx:annotation-driven />
        나중에 코드 작성이 끝나면 다시 context:component-scan을 재작성 할 예정
    <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource"
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root" p:password="root

    <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">org.hibernate.dialect.MySQLDialect</
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value></value>
            </list>
        </property>
    </bean>

    <bean class="org.springframework.orm.hibernate5.HibernateTemplate"
        name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />

    <bean class="org.springframework.orm.hibernate5.HibernateTransactionManager"
        name="transactionManager" p:sessionFactory-ref="sessionFactory" />

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        name="viewResolver">
        <property name="prefix">
            <value>/WEB-INF/jsps/</value>
        </property>

        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

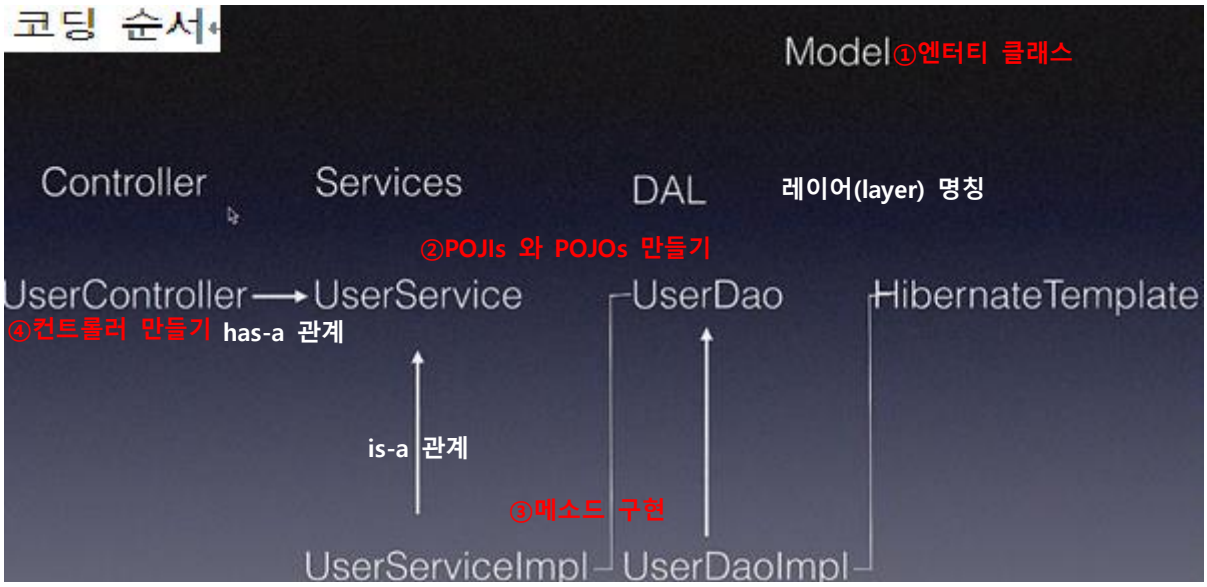
아래 애노테이션을 찾는 위치 지정

@Controller

@Service

@Repository

7. 코딩 순서



8. 모델 만들기

User.java

```java
package com.springmvcapply_orm.user.entity;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "user")
public class User {

    @Id
    private int id;
    private String name;
    private String email;

    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;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", email=" + email + "]";
    }
}
```

클래스명은 **User**, 테이블명은 **user**라 @Table 애노테이션 사용

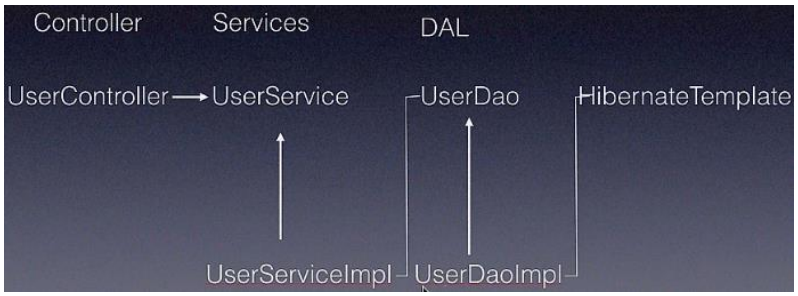
기본키 지정

- springmvcapply_orm
  - Spring Elements
  - Deployment Descriptor: <web app>
  - Java Resources
    - src/main/java
      - com.springmvcapply_orm.user.entity
        - User.java
    - src/main/resources

User.java | dispatcher-servlet.xml

```xml
    <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">org.hibernate.dialect.MySQLDialec
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.springmvcapply_orm.user.entity.User</value>
            </list>
        </property>
    </bean>
```

9. POJIs와 POJOs 만들기



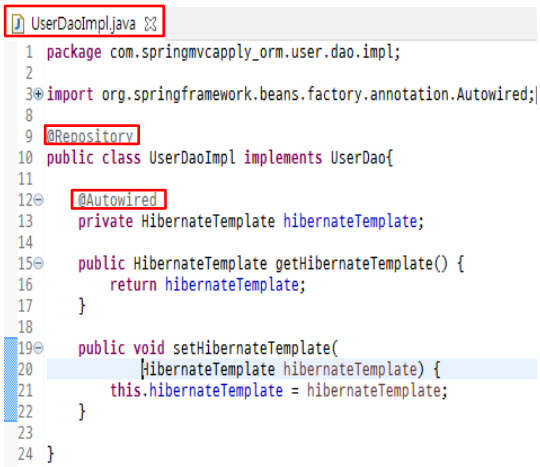
가. 궁금증

@Component로 다 쓰지 왜 굳이 @Repository, @Service, @Controller 등을 사용하냐면 예를 들어 @Repository는 DAO의 메서드에서 발생할 수 있는 unchecked exception들을 스프링의 DataAccessException으로 처리할 수 있기 때문이다.

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

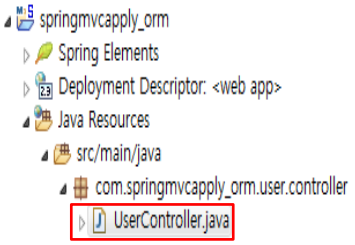
그리고 그렇게 작성하면 자동으로 등록되는 빈의 이름은 클래스의 첫 문자가 소문자로 바뀐 이름이 자동 적용된다. (HomeController -> homeController)

나. DAL 레이어 준비



다. 서비스 레이어 준비

라. 컨트롤러 준비



UserController.java

```java
package com.springmvcapply_orm.user.controller;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
public class UserController {

    @Autowired
    private UserService service;

    public UserService getService() {
        return service;
    }

    public void setService(UserService service) {
        this.service = service;
    }
}
```

10. DAO와 Service 메소드 구현

가. DAO

UserDao.java UserDaoImpl.java

```java
package com.springmvcapply_orm.user.dao;

import com.springmvcapply_orm.user.entity.User;

public interface UserDao {

    int create(User user);

}
```

UserDao.java UserDaoImpl.java

```java
package com.springmvcapply_orm.user.dao.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.hibernate5.HibernateTemplate;
import org.springframework.stereotype.Repository;

import com.springmvcapply_orm.user.dao.UserDao;
import com.springmvcapply_orm.user.entity.User;

@Repository
public class UserDaoImpl implements UserDao{

    @Autowired
    private HibernateTemplate hibernateTemplate;

    public HibernateTemplate getHibernateTemplate() {
        return hibernateTemplate;
    }

    public void setHibernateTemplate(
            HibernateTemplate hibernateTemplate) {
        this.hibernateTemplate = hibernateTemplate;
    }

    @Override
    public int create(User user) {
        int result = (int) hibernateTemplate.save(user);
        return result;
    }
}
```

나. Service

UserService.java UserServiceImpl.java

```java
package com.springmvcapply_orm.user.service;

import com.springmvcapply_orm.user.entity.User;

public interface UserService {

    int save(User user);

}
```

UserService.java UserServiceImpl.java

```java
package com.springmvcapply_orm.user.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserDao dao;

    public UserDao getDao() {
        return dao;
    }

    public void setDao(UserDao dao) {
        this.dao = dao;
    }

    @Override
    @Transactional
    public int save(User user) {
        // 비지니스 로직
        int result = dao.create(user);
        return result;
    }
}
```

11. 컨트롤러(Controller) 메소드 구현

UserController.java

```java
package com.springmvcapply_orm.user.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.springmvcapply_orm.user.entity.User;
import com.springmvcapply_orm.user.service.UserService;

@Controller
public class UserController {

    @Autowired
    private UserService service;

    @RequestMapping("registrationPage")
    public String showRegisterationPage()
    {
        return "userReg";

    }

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

        return "userReg";

    }

    public UserService getService() {
        return service;
    }

    public void setService(UserService service) {
        this.service = service;
    }
}
```

12. 결과를 만들고 보여주기

userReg.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>User Registration</title>
    </head>
    <body>
        <form action="registerUser" method="post">
            <pre>
                Id: <input type="text" name="id" />
                Name: <input type="text" name="name" />
                Email: <input type="text" name="email" />
                <input type="submit" name="register" />
            </pre>
        </form>

        <br />${result}

    </body>
</html>
```

13. 스프링 설정에서 애노테이션 활성화

dispatcher-servlet.xml | UserController.java

```xml
<?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:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="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.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <context:component-scan base-package="com.springmvcapply_orm.user" />

    <tx:annotation-driven />
```

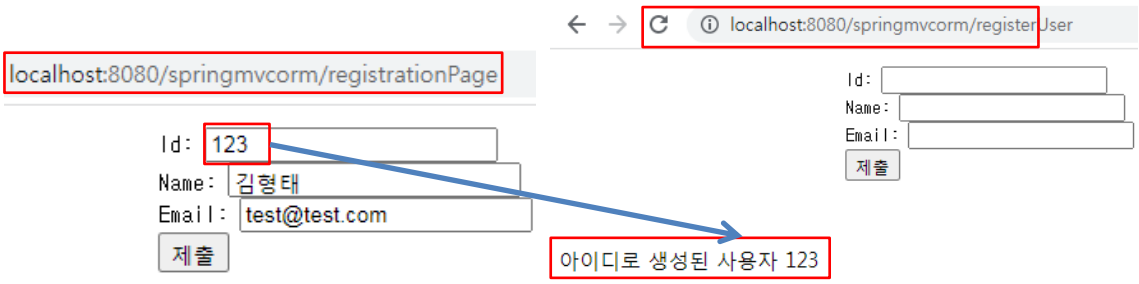
14. 스프링 설정에서 엔티티 네임 업데이트

dispatcher-servlet.xml

```xml
    <bean class="org.springframework.jdbc.datasource.DriverManagerDataSo
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root" p:passw

    <bean class="org.springframework.orm.hibernate5.LocalSessionFactoryB
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">org.hibernate.dialect.MySQL
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.springmvcapply_orm.user.entity.User</value>
            </list>
        </property>
    </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: '₩xEA₩xB9₩x80₩xED₩x98₩x95...' for column 'name' at row

다. 컨트롤러 소스 변경

UserController.java

```java
import java.io.UnsupportedEncodingException;

@Controller
public class UserController {

    @Autowired
    private UserService service;

    @RequestMapping("registrationPage")
    public String showRegisterationPage()
    {
        return "userReg";

    }

    @RequestMapping(value = "registerUser", method = RequestMethod.POST)
    public String registerUser(@ModelAttribute("user") User user,
            ModelMap model) throws UnsupportedEncodingException
    {
        System.out.println(user);

        user.setName(URLDecoder.decode(user.getName(), "utf-8"));

        int result = service.save(user);
        model.addAttribute("result", "아이디로 생성된 사용자 " + result);

        return "userReg";

    }

    public UserService getService() {
        return service;
    }

    public void setService(UserService service) {
        this.service = service;
    }
}
```

localhost:8080/springmvcorm/registrationPage

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

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: '₩xEA₩xB9₩x80₩xED₩x98₩x95...' for column 'name' at row 1

```java
UserController.java
import java.io.UnsupportedEncodingException;

@Controller
public class UserController {

    @Autowired
    private UserService service;

    @RequestMapping("registrationPage")
    public String showRegisterationPage()
    {
        return "userReg";

    }

    @RequestMapping(value = "registerUser", method = RequestMethod.POST)
    public String registerUser(@ModelAttribute("user") User user,
            ModelMap model) throws UnsupportedEncodingException
    {
        System.out.println(user);

        String kht1 = URLDecoder.decode(user.getName(), "utf-8");
        String kht2 = URLEncoder.encode(kht1, "utf-8");
        user.setName(kht2);

        int result = service.save(user);
        model.addAttribute("result", "아이디로 생성된 사용자 " + result);

        return "userReg";

    }

    public UserService getService() {
        return service;
    }

    public void setService(UserService service) {
        this.service = service;
    }
}
```

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

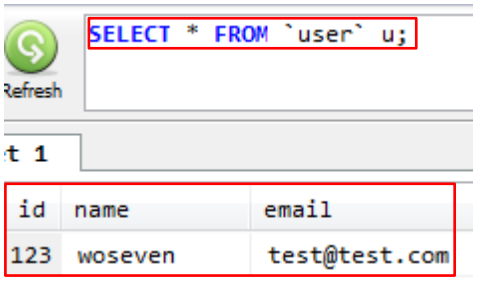
ERROR: Data truncation: Data too long for column 'name' at row 1

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

```java
@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";

}
```



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

UserController.java | web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
    <display-name>Hello Spring MVC</display-name>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherSer
    </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>
```

localhost:8080/springmvcorm/registrationPage

Id: 234
Name: 김형태
Email: test1@test.com
제출

localhost:8080/springmvcorm/registerUser

Id:
Name:
Email:
제출

아이디로 생성된 사용자 234

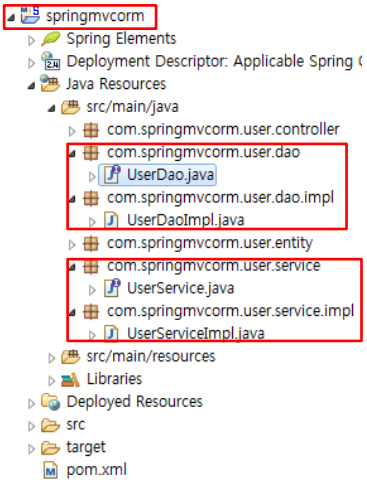
```
SELECT * FROM `user` u;
```

efresh

t 1

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

16. 모든 유저 불러오는 DAO와 서비스 구현



UserDao.java | UserDaoImpl.java

```java
package com.springmvcorm.user.dao;

import java.util.List;

public interface UserDao {

    int create(User user);

    List<User> findUsers();

}
```

UserDao.java | UserDaoImpl.java

```java
package com.springmvcorm.user.dao.impl;

import java.util.List;

@Repository
public class UserDaoImpl implements UserDao{

    @Autowired
    private HibernateTemplate hibernateTemplate;

    public HibernateTemplate getHibernateTemplate() {
        return hibernateTemplate;
    }

    public void setHibernateTemplate(
            HibernateTemplate hibernateTemplate) {
        this.hibernateTemplate = hibernateTemplate;
    }

    @Override
    public int create(User user) {
        int result = (int) hibernateTemplate.save(user);
        return result;
    }

    @Override
    public List<User> findUsers() {
        List<User> users = hibernateTemplate.loadAll(User.class);
        return users;
    }
}
```

UserService.java | UserServiceImpl.java

```java
package com.springmvcorm.user.service;

import java.util.List;

public interface UserService {

    int save(User user);

    List<User> getUsers();

}
```

UserServiceImpl.java

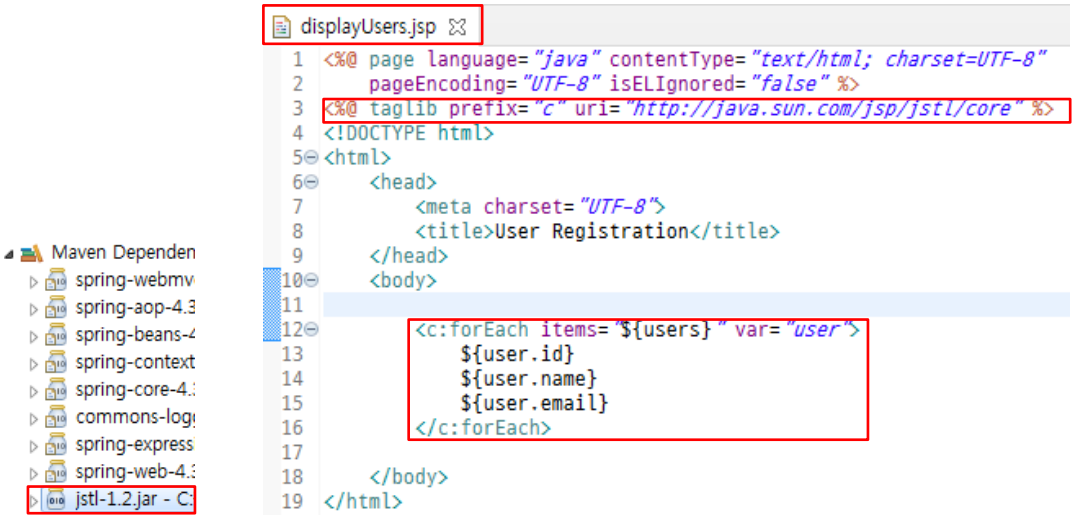
```java
package com.springmvcorm.user.service.impl;

import java.util.List;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserDao dao;

    public UserDao getDao() {
        return dao;
    }

    public void setDao(UserDao dao) {
        this.dao = dao;
    }

    @Override
    @Transactional
    public int save(User user) {
        // 비지니스 로직
        int result = dao.create(user);
        return result;
    }

    @Override
    public List<User> getUsers() {
        List<User> users = dao.findUsers();
        return users;
    }
}
```

17. 모든 유저 불러오기 컨트롤러 구현

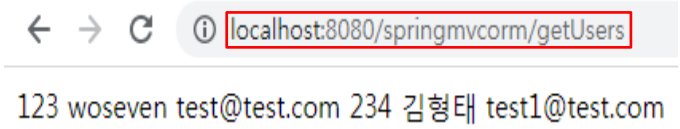
UserServiceImpl.java | UserController.java

```java
    {
        return "userReg";

    }

    @RequestMapping(value = "registerUser", me
    public String registerUser(@ModelAttribute
            ModelMap model) throws Unsupported
    {
        int result = service.save(user);
        model.addAttribute("result", "아이디로

        return "userReg";

    }

    @RequestMapping("getUsers")
    public String getUsers(ModelMap model)
    {
        List<User> users = service.getUsers();
        model.addAttribute("users", users);

        return "displayUsers";

    }
```

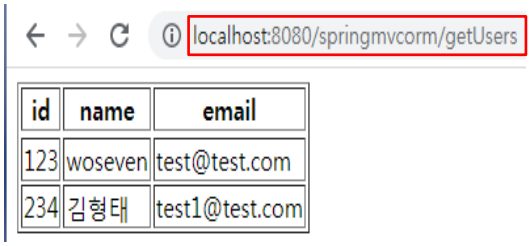
18. JSTL를 이용한 뷰 페이지 만들기



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



123 woseven test@test.com 234 김형태 test1@test.com



20. ID로 정려

displayUsers.jsp | UserDao.java | User.java

```java
package com.springmvcorm.user.entity;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "user")
public class User implements Comparable<User>{

    @Id
    private Integer id;
    private String name;
    private String email;

    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;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", email=" + email + "]";
    }

    @Override
    public int compareTo(User user) {
        return this.id.compareTo(user.id);
    }
}
```

localhost:8080/springmvcorm/registrationPage

Id: 33

Name:

Email: www@test.com

제출

Refresh

```
SELECT * FROM `user` u;
```

ltset 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

```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         // 비지니스 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

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

Refresh

```
SELECT * FROM `user` u;
```

tset 1

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