```
HOL : Spring MVC
-------------------------------
Task1. Spring MVC Demo
1. Package Explorer > right-click > New > Spring Legacy Project
2. Select Spring MVC Project
3. Project name : HelloWorldWeb
4. Next
5. Enter a topLevelPackage : com.example.biz
6. Finish


7. HelloWorldWeb project 수정하기
  1)HelloWorldWeb > right-click > Properties
  2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
  3)Apply click
  4)Build the project now? > Yes click
  5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
  > Finish
  6)Apply click
  7)Project Facets > Java > 13
  8)Runtimes tab > Check [Apache Tomcat v9.0]
  9)Apply and Close click
  10)Build the project now? > Yes click


8. pom.xml 수정하기
  <properties>
    <java-version>13</java-version>
    <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
    <org.aspectj-version>1.9.5</org.aspectj-version>
    <org.slf4j-version>1.7.30</org.slf4j-version>
  </properties>
    ...
  <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
  ...
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.3</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
  </dependency>


```

9. pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS

10. Servers View 추가하기
1)Window > Show View > Other > Server > Servers
2)Open click

11. Tomcat Server 추가하기
1)In Servers View, [No servers are available. Click this link to create a new server...] click
2)Apache > Tomcat v9.0 Server Select
3)Next
4)Name : Apache Tomcat v9.0
5)Tomcat installation directory : C:\Program Files\apache-tomcat-9.0.33
6)JRE : jdk-13.0.2
7)Finish
8)HelloWorldWeb project > right-click > Properties > Project Facets > Select Java > Change Version 13
9)Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close

12. HelloWorldWeb Project right-click > Run As > Run on Server > Finish

13. http://localhost:8080/biz/

Hello world!

The time on the server is April 19, 2020 at 6:53:00 PM KST.<--원래 한글 깨짐

14. 한글 깨짐을 수정하려면 src/main/webapp/WEB-INF/views/home.jsp에서
<%@ page session="false" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>로 수정

15. Context name 변경하기
1)Package Explorer에서 Servers/Tomcat v9.0 Server at localhost-config/server.xml에서 다음과 같이 수정한다.
-path="/biz" --> path="/demo"
<Context docBase="HelloWorldWeb" path="/demo" reloadable="true" source="org.eclipse.jst.jee.server:HelloWorldWeb"/>

2)수정 후 restart 하면 http://localhost:8080/biz --> http://localhost:8080/demo 로 변경됨

Task2. resources Folder 이용하기
1. Image 경로 알아내기
1)src/main/webapp/resources/에 images Folder를 STS Package Explorer에서 생성한다.
2)Internet에서 적당한 image를 src/main/webapp/resources/images/에 다운로드한다.
3)home.jsp에 아래 code를 추가한다.

4)HelloWorldWeb Project right-click > Run As > Run on Server > Finish
Image가 잘 나온다.

2. Image 경로 변경
1)apple.jpg image 경로를 src/main/webapp/images/로 이동.
2)하지만 이렇게 하면 image가 보이지 않는다.
3)왜냐하면, servlet-context.xml에서 resource의 경로는 <resources mapping="/resources/**" location="/resources/" />이기 때문.
4)즉, 기본적으로 resources folder 아래에서 resource를 찾는다.

3. <resources />추가
1)다시 resources Folder 하위로 images Folder를 이동
2)/src/main/webapp/하위에 images Folder를 생성하고 image를 넣고 home.jsp에 아래의 code를 추가한다.

3)실행하면 아래부분의 image는 보이지 않는다.
4)왜냐하면 새로 추가한 images Folder는 servlet-context.xml에서 설정하지 않았기 때문.
5)Image를 보이게 하기 위해 servlet-context.xml에 아래의 Code를 추가한다.
<resources mapping="/resources/**" location="/resources/" />
<resources mapping="/images/**" location="/images/" />
6)Project right-click > Run As > Run on Server > Restart >
-image 2개가 제대로 보인다.

Task3. Controller Class 제작하기
1. 제작순서
1)@Controller를 이용한 class 생성
2)@RequestMapping을 이용한 요청 경로 지정
3)요청 처리 method 구현
4)View 이름 return

5)UserController class 생성
-src/main/java/com.example.biz > right-click > New > Class
-Name : UserController
-Finish

```
package com.example.biz;

import org.springframework.stereotype.Controller;

@Controller
public class UserController {

}
```

2. 요청 처리 method 생성

```
package com.example.biz;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
```

```
168   public class UserController {
169     @RequestMapping("/view")
170     public String view(Model model){
171       model.addAttribute("currentDate", new java.util.Date());
172       return "view";
173     }
174
175     @RequestMapping("/user")
176     public String user(Model model){
177       model.addAttribute("username", "한지민");
178       model.addAttribute("userage", 24);
179       model.addAttribute("job", "Developer");
180       return "user";
181     }
182
183     @RequestMapping("/fruits")
184     public String fruits(Model model){
185       String [] array = {"Apple", "Mango", "Lemon", "Grape"};
186       model.addAttribute("fruits", array);
187       return "fruits";
188     }
189   }
190
191
192 3. View에 Data 전달
193   1)view.jsp 생성
194     -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
195     -Next
196     -File name : view.jsp
197     -Finish
198
199       <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
200       <!DOCTYPE html>
201       <html>
202         <head>
203           <meta charset="UTF-8">
204           <title>Insert title here</title>
205         </head>
206         <body>
207           <h1>view.jsp 입니다.</h1>
208           현재 날짜와 시간은 ${currentDate} 입니다.
209         </body>
210       </html>
211
212   2)fruits.jsp 생성
213     -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
214     -Next
215     -File name : fruits.jsp
216     -Finish
217
218       <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
219       <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
220       <!DOCTYPE html>
221       <html>
222       <head>
223       <meta charset="UTF-8">
```

```
224         <title>Insert title here</title>
225         </head>
226         <body>
227           <h2>fruits.jsp</h2>
228           <ul>과일 종류
229           <c:forEach items="${fruits}" var="fruit">
230             <li>${fruit}</li>
231           </c:forEach>
232           </ul>
233         </body>
234         </html>
235
236     3)user.jsp 생성
237       -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
238       -Next
239       -File name : user.jsp
240       -Finish
241
242         <%@ page language="java" contentType="text/html; charset=UTF-8"
243           pageEncoding="UTF-8"%>
244         <!DOCTYPE html>
245         <html>
246         <head>
247         <meta charset="UTF-8">
248         <title>Insert title here</title>
249         </head>
250         <body>
251           <h2>user.jsp</h2>
252           <ul>
253             <li>Name : ${username}</li>
254             <li>Age : ${userage}</li>
255             <li>Job : ${job}</li>
256           </ul>
257         </body>
258         </html>
259
260     4)http://localhost:8080/demo/view
261     5)http://localhost:8080/demo/fruits
262     6)http://localhost:8080/demo/user
263
264
265 4. View에 ModelAndView 객제로 data 전달
266     1)UserController.java에 아래의 코드 추가
267
268       @RequestMapping(value = "/demo", method = RequestMethod.GET)
269       public ModelAndView demo() {
270         /*
271         ModelAndView mav = new ModelAndView("view2");
272         mav.addObject("username", "한지민");
273         mav.addObject("currentDate", new java.util.Date());
274         return mav;
275         */
276         ModelAndView mav = new ModelAndView();
277         mav.addObject("userid", "example");
278         mav.addObject("passwd", "12345678");
279         mav.setViewName("/demo");
280         return mav;
281       }
```

2)src/main/webapp/WEB-INF/views/demo.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html">
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
  </head>
  <body>
    아이디 : ${userid} <br />
    패스워드 : ${passwd}
  </body>
</html>
```

3)http://localhost:8080/demo/demo
아이디 : example
패스워드 : 12345678

5. Controller class에 @RequestMapping 적용
1)src/main/java/com.example.biz.StudentController.java 생성

```
package com.example.biz;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/bbs")
public class StudentController {

  @RequestMapping(value="/get", method = RequestMethod.GET)
  public ModelAndView getStudent() {

    ModelAndView mav = new ModelAndView();
    mav.setViewName("/bbs/get");   //  /WEB-INF/views/bbs/get.jsp
    mav.addObject("name", "한지민");
    mav.addObject("age", 25);
    return mav;
  }
}
```

2)src/main/webapp/WEB-INF/views/bbs/get.jsp
-views > right-click > New > Folder
-Folder name : bbs
-Finish

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```
    <title>Insert title here</title>
    </head>
    <body>
      학생 이름 : ${name} <br />
      학생 나이 : ${age}
    </body>
    </html>
```

3)http://localhost:8080/demo/bbs/get
학생 이름 : 한지민
학생 나이 : 25

Task4. 다양한 GET Request 처리하기

1. Package Explorer > right-click > New > Spring Legacy Project
2. Select Spring MVC Project
3. Project name : MVCDemo
4. Next
5. Enter a topLevelPackage : com.example.biz
6. Finish

7. MVCDemo project 수정하기
   1)HelloWorldWeb > right-click > Properties
   2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
   3)Apply click
   4)Build the project now? > Yes click
   5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)] > Finish
   6)Apply click
   7)Project Facets > Java > 13
   8)Runtimes tab > Check [Apache Tomcat v9.0]
   9)Apply and Close click
   10)Build the project now? > Yes click

8. pom.xml 수정하기

```
  <properties>
    <java-version>13</java-version>
    <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
    <org.aspectj-version>1.9.5</org.aspectj-version>
    <org.slf4j-version>1.7.30</org.slf4j-version>
  </properties>
    ...
  <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
  ...
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
```

```
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.3</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
  </dependency>


9. pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS


10. src/main/java/com.example.biz/RequestController.java 생성

  package com.example.biz;

  import org.springframework.stereotype.Controller;

  @Controller
  public class RequestController {
  }


11. HttpServletRequest class 이용하기
  1)src/main/webapp/static folder 생성
    -src/main/webapp > right-click > New > Folder
    -Folder name : static
    -Finish

  2)static folder resource 등록하기
    -servlet-context.xml에 다음 코드 추가
      <resources mapping="/static/**" location="/static/" />

  3)src/main/webapp/static/register.html

    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>회원정보</title>
    </head>
    <body>
      <h1>회원정보</h1>
      <form action="/biz/confirm" method="GET">
      <ul>
        <li>User ID : <input type="text" name="userid"></li>
        <li>Password : <input type="password" name="passwd"></li>
        <li>Name : <input type="text" name="name"></li>
        <li>Age : <input type="number" name="age"></li>
        <li>Gender : <input type="radio" name="gender" value="남성">남성
          <input type="radio" name="gender" value="여성">여성</li>
        <li><input type="submit" value="전송하기"></li>
```

```
        </ul>
        </form>
    </body>
    </html>
```

4)RequestController.java

```
    @RequestMapping(value="/confirm", method=RequestMethod.GET)
    public String confirm(HttpServletRequest request, Model model) {
      String userid = request.getParameter("userid");
      String passwd = request.getParameter("passwd");
      String name = request.getParameter("name");
      int age = Integer.parseInt(request.getParameter("age"));
      String gender = request.getParameter("gender");

      model.addAttribute("userid", userid);
      model.addAttribute("passwd", passwd);
      model.addAttribute("name", name);
      model.addAttribute("age", age);
      model.addAttribute("gender", gender);
      return "confirm";   //  /WEB-INF/views/confirm.jsp
    }
```

5)src/main/webapp/WEB-INF/views/confirm.jsp

```
    <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Insert title here</title>
      </head>
      <body>
        아이디 : ${userid} <br />
        패스워드 : ${passwd} <br />
        사용자 이름 : ${name} <br />
        나이 : ${age} <br />
        성별 : ${gender} <br />
      </body>
    </html>
```

6)Project right-click > Run As > Run on Server > restart
7)http://localhost:8080/biz/static/register.html
8)or http://localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234

```
    아이디 : jimin
    패스워드 : 1234
    사용자 이름 : 한지민
    나이 : 25
    성별 : 여성
```

12. @RequestParam Annotation 이용하기

1)RequestController.java

```
    @RequestMapping(value="/confirm", method=RequestMethod.GET)
    public String confirm(@RequestParam("userid") String userid,
                          @RequestParam("passwd") String passwd,
```

```
                    @RequestParam("name") String name,
                    @RequestParam("age") int age,
                    @RequestParam("gender") String gender ,Model model) {

      model.addAttribute("userid", userid);
      model.addAttribute("passwd", passwd);
      model.addAttribute("name", name);
      model.addAttribute("age", age);
      model.addAttribute("gender", gender);
      return "confirm";
    }
```

2)src/main/webapp/WEB-INF/views/confirm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
  </head>
  <body>
    아이디 : ${userid} <br />
    패스워드 : ${passwd} <br />
    사용자 이름 : ${name} <br />
    나이 : ${age} <br />
    성별 : ${gender} <br />
  </body>
</html>
```

3)localhost:8080/biz/confirm?name=한지민&gender=여성
&age=25&userid=jimin&passwd=1234

```
아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성
```

13. Data Commander 객체 이용하기1

1)lombok library를 pom.xml에 추가하고 maven install할 것

2)src/main/java/com.example.vo.UserVO.java 생성

-src/main/java > right-click > New > Package
-Name : com.example.vo
-Finish
-com.example.vo > right-click > New > Click
-Name : UserVO

```
  package com.example.vo;

  import lombok.AllArgsConstructor;
  import lombok.Getter;
  import lombok.NoArgsConstructor;
  import lombok.Setter;
  import lombok.ToString;
```

```
564         @NoArgsConstructor
565         @AllArgsConstructor
566         @Getter
567         @Setter
568         @ToString
569         public class UserVO {
570           private String userid;
571           private String passwd;
572           private String name;
573           private int age;
574           private String gender;
575         }
576
577     3)RequestController.java
578
579       @RequestMapping(value="/confirm", method=RequestMethod.GET)
580       public String confirm(@RequestParam("userid") String userid,
581           @RequestParam("passwd") String passwd,
582           @RequestParam("name") String name,
583           @RequestParam("age") int age,
584           @RequestParam("gender") String gender  ,Model model) {
585
586         UserVO userVO = new UserVO();
587         userVO.setUserid(userid);
588         userVO.setPasswd(passwd);
589         userVO.setName(name);
590         userVO.setAge(age);
591         userVO.setGender(gender);
592
593         model.addAttribute("userVO", userVO);
594
595         return "confirm1";
596       }
597
598     4)src/main/webapp/WEB-INF/views/confirm1.jsp
599
600       <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
601       <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
602       <c:set var="user" value="${userVO}"/>
603       <!DOCTYPE html>
604       <html>
605       <head>
606       <meta charset="UTF-8">
607       <title>Insert title here</title>
608       </head>
609       <body>
610         <h1>confirm1.jsp</h1>
611         <h2>사용자 정보</h2>
612         아이디 : ${user.userid} <br />
613         패스워드 : ${user.passwd} <br />
614         이름 : ${user.name} <br />
615         나이 : ${user.age} <br />
616         성별 : ${user.gender}
617       </body>
618       </html>
619
620     5)Project right-click > Run As > Run on Server > restart
```

6)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
confirm1.jsp

사용자 정보

아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성

14. Data Commander 객체 이용하기2
1)RequestController.java

```
@RequestMapping(value="/confirm", method=RequestMethod.GET)
public String confirm(UserVO userVO) {
  return "confirm2";
}
```

2)src/main/webapp/WEB-INF/views/confirm2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="user" value="${userVO}"/>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>confirm2.jsp</h1>
  <h2>사용자 정보</h2>
  아이디 : ${user.userid} <br />
  패스워드 : ${user.passwd} <br />
  이름 : ${user.name} <br />
  나이 : ${user.age} <br />
  성별 : ${user.gender}
</body>
</html>
```

3)Project right-click > Run As > Run on Server > restart
4)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
confirm2.jsp

사용자 정보

아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성

15. @PathVariable 이용하기
1)RequestController.java

```
@RequestMapping(value="/confirm/{userid}/{passwd}/{name}/{age}/{gender}",
method=RequestMethod.GET)
public String confirm(@PathVariable String userid, @PathVariable String passwd,
                      @PathVariable String name, @PathVariable int age,
                  @PathVariable String gender,Model model) {
  model.addAttribute("userInfo", new UserVO(userid, passwd, name, age, gender));
  return "confirm3";
}
```

2)src/main/webapp/WEB-INF/views/confirm3.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="user" value="${userInfo}"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>confirm3.jsp</h1>
  <h2>사용자 정보</h2>
  아이디 : ${user.userid} <br />
  패스워드 : ${user.passwd} <br />
  이름 : ${user.name} <br />
  나이 : ${user.age} <br />
  성별 : ${user.gender}
</body>
</html>
```

3)Project right-click > Run As > Run on Server > restart
4)localhost:8080/biz/confirm/jimin/1234/한지민/25/여성
confirm3.jsp

사용자 정보

아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성

Task5. @RequestMapping Parameter 다루기
1. GET 방식과 POST 방식
1)src/main/java/com.example.biz/HomeController.java

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@RequestParam("userid") String userid,
                    @RequestParam("passwd") String passwd,
```

```
                    Model model) {

      model.addAttribute("userid", userid);
      model.addAttribute("passwd", passwd);
      return "login";
    }

2)src/main/webapp/resources/login.html
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>로그인 폼</title>
    </head>
    <body>
      <form method="GET" action="/biz/login">
        아이디 : <input type="text" name="userid" /><br />
        패스워드 : <input type="password" name="passwd" /><br />
        <input type="submit" value="로그인하기" />
      </form>
    </body>
    </html>

3)http://localhost:8080/biz/resources/login.html에서 submit 하면 405 error 발생
4)왜냐하면 서로의 method가 불일치하기 때문
5)해결방법
    -src/main/java/com.example.biz/HomeController.java 수정
    -즉 login method(요청 처리 method)의 이름은 같지만 parameter의 type과 return type이 틀리기 때문
    에 Method Overloading 됨.

      @RequestMapping(value="/login", method=RequestMethod.POST)
      public String login(@RequestParam("userid") String userid,
                    @RequestParam("passwd") String passwd,
                    Model model) {

        model.addAttribute("userid", userid);
        model.addAttribute("passwd", passwd);
        return "login";
      }
      @RequestMapping(value="/login", method=RequestMethod.GET)
      public ModelAndView login(@RequestParam("userid") String userid,
                    @RequestParam("passwd") String passwd) {

        ModelAndView mav = new ModelAndView();
        mav.addObject("userid", userid);
        mav.addObject("passwd", passwd);
        mav.setViewName("login");
        return mav;
      }

6)src/main/webapp/WEB-INF/views/login.jsp

    <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
```

```
787     <title>Insert title here</title>
788     </head>
789     <body>
790       아이디 : ${userid} <br />
791       패스워드 : ${passwd}
792     </body>
793     </html>
794
795   7)http://localhost:8080/biz/resources/login.html
796     아이디 : jimin
797     패스워드 : 1234
798
799
800 2. @ModelAttribute Annotation 이용하기
801   1)@ModelAttribute Annotation을 이용하면 Data Commander 객체의 이름을 변경할 수 있다.
802   2)src/main/webapp/resources/register.html
803
804     <!DOCTYPE html>
805     <html>
806     <head>
807     <meta charset="UTF-8">
808     <title>회원가입 폼</title>
809     </head>
810     <body>
811       <form method="POST" action="/biz/register">
812         아이디 : <input type="text" name="userid" /><br />
813         패스워드 : <input type="password" name="passwd" /><br />
814         이름 : <input type="text" name="name" /><br />
815         나이 : <input type="number" name="age" /><br />
816         성별 : <input type="radio" name="gender" value="남성" />남성   
817             <input type="radio" name="gender" value="여성" />여성<br />
818         <input type="submit" value="가입하기" />
819       </form>
820     </body>
821     </html>
822
823   3)src/main/java/com.example.biz/HomeController.java
824
825     @RequestMapping(value="/register", method=RequestMethod.POST)
826     public String register(@ModelAttribute("u") UserVO userVO) {    //userVO가 아니라 u로 변경
827
828       return "register";
829     }
830
831   4)src/main/webapp/WEB-INF/views/register.jsp
832
833     <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
834     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
835     <c:set var="user" value="${u}"/>
836     <!DOCTYPE html>
837     <html>
838     <head>
839     <meta charset="UTF-8">
840     <title>Insert title here</title>
841     </head>
842     <body>
843       <h1>사용자 정보</h1>
```

```
        <ul>
          <li>아이디 : ${user.userid}</li>
          <li>패스워드 : ${user.passwd}</li>
          <li>이름 : ${user.name}</li>
          <li>나이 : ${user.age}</li>
          <li>성별 : ${user.gender}</li>
        </ul>
      </body>
      </html>

   5)Spring에서 POST 방식으로 Data를 보낼 때 한글깨짐 현상 발생
   6)해결방법
   7)web.xml

      <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>

   8)http://localhost:8080/biz/resources/register.html -->
   9)http://localhost:8080/biz/register
      사용자 정보

      아이디 : jimin
      패스워드 : 1234
      사용자 이름 : 한지민
      나이 : 25
      성별 : 여성


3. redirect: 키워드 이용하기
   1)src/main/java/com.example.biz/HomeController.java

      @RequestMapping("/verify")
      public String verify(HttpServletRequest request, Model model) {
        String userid = request.getParameter("userid");
        if(userid.equals("admin")) {     //만일 userid가 admin 이면 /admin으로 리다이렉트
          return "redirect:admin";
        }
        return "redirect:user";          //만일 userid가 admin 이 아니면 /user로 리다이렉트
        //return "redirect:http://www.naver.com";    //절대 경로도 가능
      }

      @RequestMapping("/admin")
      public String verify1(Model model) {
        model.addAttribute("authority", "관리자권한");
        return "admin";
      }

      @RequestMapping("/user")
```

```
902       public String verify2(Model model) {
903         model.addAttribute("authority", "일반사용자");
904         return "user";
905       }
906
907     2)/src/main/webapp/WEB-INF/views/admin.jsp
908       <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
909       <!DOCTYPE html>
910       <html>
911       <head>
912       <meta charset=UTF-8">
913       <title>Insert title here</title>
914       </head>
915       <body>
916         <h1>관리자 페이지</h1>
917         권한 : ${authority}
918       </body>
919       </html>
920
921     3)/src/main/webapp/WEB-INF/views/user.jsp
922
923       <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
924       <!DOCTYPE html>
925       <html>
926       <head>
927       <meta charset=UTF-8">
928       <title>Insert title here</title>
929       </head>
930       <body>
931         <h1>일반 사용자 페이지</h1>
932         권한 : ${authority}
933       </body>
934       </html>
935
936     4)http://localhost:8080/biz/verify?userid=admin --> http://localhost:8080/biz/admin
937     5)http://localhost:8080/biz/verify?userid=user --> https://www.naver.com
938
939
940
941 ---------------------------------------------------------------------------
942 Task6. Java Annotation으로 Web Application project 생성하기
943 1. Package Explorer > right-click > New > Spring Legacy Project
944 2. Select Spring MVC Project
945 3. Project name : HelloWorldWeb1
946 4. Next
947 5. Enter a topLevelPackage : com.example.biz
948 6. Finish
949
950
951 7. HelloWorldWeb1 project 수정하기
952   1)HelloWorldWeb > right-click > Properties
953   2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
954   3)Apply click
955   4)Build the project now? > Yes click
956   5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
      > Finish
```

```
   6)Apply click
   7)Project Facets > Java > 13
   8)Runtimes tab > Check [Apache Tomcat v9.0]
   9)Apply and Close click
   10)Build the project now? > Yes click


8. pom.xml 수정하기
   <properties>
     <java-version>13</java-version>
     <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
     <org.aspectj-version>1.9.5</org.aspectj-version>
     <org.slf4j-version>1.7.30</org.slf4j-version>
   </properties>
     ...
   <dependency>
       <groupId>log4j</groupId>
       <artifactId>log4j</artifactId>
       <version>1.2.17</version>
   ...
   <dependency>
     <groupId>javax.servlet</groupId>
     <artifactId>javax.servlet-api</artifactId>
     <version>4.0.1</version>
     <scope>provided</scope>
   </dependency>
   <dependency>
     <groupId>javax.servlet.jsp</groupId>
     <artifactId>javax.servlet.jsp-api</artifactId>
     <version>2.3.3</version>
     <scope>provided</scope>
   </dependency>
   <dependency>
     <groupId>org.junit.jupiter</groupId>
     <artifactId>junit-jupiter-api</artifactId>
     <version>5.6.2</version>
     <scope>test</scope>
   </dependency>


9. pom.xml > right-click > Run As > Maven install
   [INFO] BUILD SUCCESS


10. HelloWorldWeb1 > right-click > Run As > Run on Server
   Hello world!
   The time on the server is April 19, 2020 at 10:16:40 PM KST.


11. web.xml 삭제
12. /WEB-INF/spring folder 삭제
13. pom.xml 수정
   1)web.xml을 삭제하면 pom.xml에서 error 발생
   2)pom.xml 하단에 <plugins> 설정

     <plugin>
       <groupId>org.apache.maven.plugins</groupId>
       <artifactId>maven-war-plugin</artifactId>
```

```
        <version>3.2.3</version>
        <configuration>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
      </plugin>


14. com.example.config folder 생성
  1)src/main/java > right-click > New > Folder
  2)Folder name : config
  3)Finish
  4)config > right-click > New > Class
  5)Name : RootConfig
  6)Finish

    package config;

    import org.springframework.context.annotation.Configuration;

    @Configuration
    public class RootConfig {

    }

  7)config > right-click > New > Class
  8)Name : ServletConfig
  9)Interfaces : org.springframework.web.servlet.config.annotation.WebMvcConfigurer
  10)Finish

    package config;

    import org.springframework.context.annotation.ComponentScan;
    import org.springframework.web.servlet.config.annotation.EnableWebMvc;
    import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
    import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
    import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
    import org.springframework.web.servlet.view.InternalResourceViewResolver;
    import org.springframework.web.servlet.view.JstlView;

    @EnableWebMvc
    @ComponentScan(basePackages = {"com.example"})
    public class ServletConfig implements WebMvcConfigurer {

      @Override
      public void configureViewResolvers(ViewResolverRegistry registry) {
        InternalResourceViewResolver bean = new InternalResourceViewResolver();
        bean.setViewClass(JstlView.class);
        bean.setPrefix("/WEB-INF/views/");
        bean.setSuffix(".jsp");
        registry.viewResolver(bean);
      }
      @Override
      public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
      }
    }

  11)config > right-click > New > Class
```

12)Name : WebConfig

```
package config;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitia
lizer;

public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer{

  @Override
  protected Class<?>[] getRootConfigClasses() {
    return new Class[] { RootConfig.class };
  }

  @Override
  protected Class<?>[] getServletConfigClasses() {
    return new Class[] { ServletConfig.class };
  }

  @Override
  protected String[] getServletMappings() {
    return new String[] { "/" };
  }
}
```

15. HelloWorldWeb1 > right-click > Run As > Run on Server > restart

```
Hello world!

The time on the server is April 19, 2020 at 10:39:26 PM KST.
```

16. ServletConfig.java 수정

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
  registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
  registry.addResourceHandler("/images/**").addResourceLocations("/images/");
  registry.addResourceHandler("/static/**").addResourceLocations("/static/");
}
```

17. webapp/static folder 생성

1)webapp/static/css folder 생성

- -bootstrap-theme.css
- -bootstrap.css

2)webapp/static/fonts folder 생성

- -glyphicons-halflings-regular.eot
- -glyphicons-halflings-regular.svg
- -glyphicons-halflings-regular.ttf
- -glyphicons-halflings-regular.woff
- -glyphicons-halflings-regular.woff2

3)webapp/static/js folder 생성

- -bootstrap.js
- -jquery-3.4.1.js

18. webapp/images folder 생성
  1)apple.jpg

19. views/main.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Welcome example.com</title>
   <link rel="stylesheet" href="static/css/bootstrap.css">
   <link rel="stylesheet" href="static/css/bootstrap-theme.css">
   <script src="static/js/jquery-3.4.1.js"></script>
</head>
<body>
   <div class="jumbotron">
      <h1>Welcome to www.example.com</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nemo accusantium,
      aspernatur quos porro commodi perspiciatis, assumenda consequuntur eius inventore
      eaque omnis, magni natus corrupti doloremque? Quia aliquid excepturi tempora
      praesentium modi necessitatibus sequi quisquam dolorem nihil deserunt magnam,
      distinctio sunt aspernatur nisi. Id odio quaerat amet quidem adipisci totam ad.</p>
      <p>
       <h2>Example heading <span class="label label-success">Success</span></h2>
       <h4>Current Server Time is <span class="label
       label-info">${serverTime}</span></h2>
      </p>
     </div>
     <div>
       <img src="images/apple.jpg"/>
     </div>
</body>
</html>
```

20. com.example.biz/HomeController.java 수정

```
package com.example.biz;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * Handles requests for the application home page.
 */
```

```
1181    @Controller
1182    public class HomeController {
1183      private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
1184      @RequestMapping(value = "/", method = RequestMethod.GET)
1185      public String home(Locale locale, Model model) {
1186        logger.info("Welcome home! The client locale is {}.", locale);
1187
1188        Date date = new Date();
1189        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
          DateFormat.LONG, locale);
1190        String formattedDate = dateFormat.format(date);
1191        model.addAttribute("serverTime", formattedDate );
1192        return "main";
1193      }
1194
1195    }
1196
1197
1198  21. HelloWorldWeb1 > right-click > Run As > Run on Server > restart
1199
1200
1201
1202  -------------------------------
1203  Task7. Database와 연동하기
1204  1. Package Explorer > right-click > New > Spring Legacy Project
1205  2. Select Spring MVC Project
1206  3. Project name : MVCDemo1
1207  4. Next
1208  5. Enter a topLevelPackage : com.example.biz
1209  6. Finish
1210
1211
1212  7. MVCDemo1 project 수정하기
1213    1)MVCDemo1 > right-click > Properties
1214    2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
1215    3)Apply click
1216    4)Build the project now? > Yes click
1217    5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
        > Finish
1218    6)Apply click
1219    7)Project Facets > Java > 13
1220    8)Runtimes tab > Check [Apache Tomcat v9.0]
1221    9)Apply and Close click
1222    10)Build the project now? > Yes click
1223
1224
1225  8. pom.xml 수정하기
1226    <properties>
1227      <java-version>13</java-version>
1228      <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
1229      <org.aspectj-version>1.9.5</org.aspectj-version>
1230      <org.slf4j-version>1.7.30</org.slf4j-version>
1231    </properties>
1232      ...
1233    <dependency>
1234        <groupId>log4j</groupId>
1235        <artifactId>log4j</artifactId>
1236        <version>1.2.17</version>
```

```
  ...
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.3</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
  </dependency>


9. pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS


10. Create Table in MariaDB
    CREATE TABLE Member
    (
      userid        VARCHAR(20),
      username        VARCHAR(20)   NOT NULL,
      userage        TINYINT   NOT NULL,
      gender      VARCHAR(10)   NOT NULL,
      city          VARCHAR(50),
      CONSTRAINT member_userid_pk  PRIMARY KEY(userid)
    );
    -반드시 [test] Database의 조합을 utf8_general_ci로 맞출 것
    -반드시 Member Table의 기본조합이 utf8_general_ci 임을 확인할 것


11. src/main/webapp/static folder 생성
  1)src/main/webapp/static/css folder
  2)src/main/webapp/static/images folder
  3)src/main/webapp/static/js folder
    -jquery-1.12.4.js
  4)src/main/webapp/static/register.html
    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <title>회원 가입</title>
    </head>
    <body>
      <h1>회원 가입 창</h1>
      <form action="/biz/create" method="post">
        <ul>
          <li>ID : <input type="text" name="userid" /></li>
          <li>이름 : <input type="text" name="username" /></li>
          <li>나이 : <input type="number" name="age" /></li>
```

```
            <li>성별 : <input type="radio" name="gender" value="남성"/>남성
                       <input type="radio" name="gender" value="여성"/>여성</li>
            <li>거주지 : <input type="text" name="city" /></li>
            <li><input type="submit" value="가입하기" /></li>
          </ul>
        </form>
      </body>
      </html>


10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
   <resources mapping="/static/**" location="/static/" /> 추가

   <context:component-scan base-package="com.example" /> 수정


11. src/main/resources/mariadb.properties
   db.driverClass=org.mariadb.jdbc.Driver
   db.url=jdbc:mariadb://localhost:3306/test
   db.username=root
   db.password=javamariadb


12. Spring JDBC 설치
   1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

      <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-jdbc</artifactId>
          <version>5.2.0.RELEASE</version>
      </dependency>

   2)pom.xml에 붙여 넣고 Maven Install 하기
      [INFO] BUILD SUCCESS


13. MariaDB Jdbc Driver library 검색 및 설치
   1)Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.

      <dependency>
          <groupId>org.mariadb.jdbc</groupId>
          <artifactId>mariadb-java-client</artifactId>
          <version>2.5.1</version>
      </dependency>

   2)pom.xml에 붙여 넣고 Maven Install 하기
      [INFO] BUILD SUCCESS


14. src/main/webapp/WEB-INF/spring/root-context.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"
      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-4.3.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->
    <context:property-placeholder location="classpath:mariadb.properties"/>
    <bean id="dataSource"
    class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
      <property name="driverClass" value="${db.driverClass}" />
      <property name="url" value="${db.url}" />
      <property name="username" value="${db.username}" />
      <property name="password" value="${db.password}" />
    </bean>

    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
      <property name="dataSource" ref="dataSource" />
    </bean>
  </beans>


15. src/test/java/com.example.biz/TestApp class 생성
  1)com.example.biz > right-click > New > JUnit Test Case
  2)Select [New JUnit 4 test]
  3)Name : TestApp
  4)Finish
    package com.example.biz;

    import org.junit.Before;
    import org.junit.Test;
    import org.springframework.context.ApplicationContext;
    import org.springframework.context.support.GenericXmlApplicationContext;
    import org.springframework.jdbc.core.JdbcTemplate;

    public class TestApp {
      private ApplicationContext ctx;

      @Before
      public void init() {
        this.ctx = new
        GenericXmlApplicationContext("file:src/main/webapp/WEB-INF/spring**/root-context
        .xml");
      }
      @Test
      public void test() {
        JdbcTemplate jdbcTemplate = this.ctx.getBean("jdbcTemplate", JdbcTemplate.class);
        System.out.println(jdbcTemplate);
      }
    }

  5)Run as > JUnit Test > Green bar


16. package 생성
  1)src/main/java/com.example.vo
  2)src/main/java/com.example.dao
  3)src/main/java/com.example.service


17. src/com.example.vo.MemberVO class 생성
```

```
1406  package com.example.vo;
1407
1408  public class MemberVO {
1409    private String userid;
1410    private String username;
1411    private int age;
1412    private String gender;
1413    private String city;
1414
1415    public MemberVO() {}
1416
1417    public MemberVO(String userid, String username, int age, String gender, String city) {
1418      this.userid = userid;
1419      this.username = username;
1420      this.age = age;
1421      this.gender = gender;
1422      this.city = city;
1423    }
1424
1425    public String getUserid() {
1426      return userid;
1427    }
1428
1429    public void setUserid(String userid) {
1430      this.userid = userid;
1431    }
1432
1433    public String getUsername() {
1434      return username;
1435    }
1436
1437    public void setUsername(String username) {
1438      this.username = username;
1439    }
1440
1441    public int getAge() {
1442      return age;
1443    }
1444
1445    public void setAge(int age) {
1446      this.age = age;
1447    }
1448
1449    public String getGender() {
1450      return gender;
1451    }
1452
1453    public void setGender(String gender) {
1454      this.gender = gender;
1455    }
1456
1457    public String getCity() {
1458      return city;
1459    }
1460
1461    public void setCity(String city) {
1462      this.city = city;
1463    }
```

```
1464
1465      @Override
1466      public String toString() {
1467        return "MemberVO [userid=" + userid + ", username=" + username + ", age=" + age
          + ", gender=" + gender
1468            + ", city=" + city + "]";
1469      }
1470    }
1471
1472  18. com/example.dao
1473    1)MemberDao interface
1474      package com.example.dao;
1475
1476      import java.util.List;
1477
1478      import com.example.vo.MemberVO;
1479
1480      public interface MemberDao {
1481        int create(MemberVO memberVo);
1482        MemberVO read(String userid);
1483        List<MemberVO> readAll();
1484        int update(MemberVO memberVo);
1485        int delete(String userid);
1486      }
1487
1488    2)MemberDaoImpl.java
1489      package com.example.dao;
1490
1491      import java.util.List;
1492
1493      import org.springframework.beans.factory.annotation.Autowired;
1494      import org.springframework.jdbc.core.JdbcTemplate;
1495      import org.springframework.stereotype.Repository;
1496
1497      import com.example.vo.MemberVO;
1498
1499      @Repository("memberDao")
1500      public class MemberDaoImpl implements MemberDao {
1501        @Autowired
1502        JdbcTemplate jdbcTemplate;
1503
1504      @Override
1505      public int create(MemberVO memberVo) {
1506        return 0;
1507      }
1508
1509      @Override
1510      public MemberVO read(String userid) {
1511        return null;
1512      }
1513
1514      @Override
1515      public List<MemberVO> readAll() {
1516        return null;
1517      }
1518
1519      @Override
1520      public int update(MemberVO memberVo) {
```

```
1521        return 0;
1522      }
1523
1524      @Override
1525      public int delete(String userid) {
1526        return 0;
1527      }
1528    }
1529
1530  19. com.example.service
1531    1)MemberService interface
1532      package com.example.service;
1533
1534      import java.util.List;
1535
1536      import com.example.vo.MemberVO;
1537
1538      public interface MemberService {
1539        int create(MemberVO memberVo);
1540        MemberVO read(String userid);
1541        List<MemberVO> readAll();
1542        int update(MemberVO memberVo);
1543        int delete(String userid);
1544      }
1545
1546    2)MemberServiceImpl.java
1547      package com.example.service;
1548
1549      import java.util.List;
1550
1551      import org.springframework.beans.factory.annotation.Autowired;
1552      import org.springframework.stereotype.Service;
1553
1554      import com.example.dao.MemberDao;
1555      import com.example.vo.MemberVO;
1556
1557      @Service("memberService")
1558      public class MemberServiceImpl implements MemberService {
1559        @Autowired
1560        MemberDao memberDao;
1561
1562        @Override
1563        public int create(MemberVO memberVo) {
1564          return 0;
1565        }
1566
1567        @Override
1568        public MemberVO read(String userid) {
1569          return null;
1570        }
1571
1572        @Override
1573        public List<MemberVO> readAll() {
1574          return null;
1575        }
1576
1577        @Override
1578        public int update(MemberVO memberVo) {
```

```
1579          return 0;
1580        }
1581
1582        @Override
1583        public int delete(String userid) {
1584          return 0;
1585        }
1586      }
1587
1588
1589  20. com.example.biz
1590    1)HomeController.java
1591
1592      package com.example.biz;
1593
1594      import org.springframework.beans.factory.annotation.Autowired;
1595      import org.springframework.stereotype.Controller;
1596
1597      import com.example.service.MemberService;
1598
1599      /**
1600       * Handles requests for the application home page.
1601       */
1602      @Controller
1603      public class HomeController {
1604        @Autowired
1605        MemberService memberService;
1606      }
1607
1608
1609  21. Data Insert
1610    1)/src/main/webapp/static/register.html
1611    2)com.example.biz/HomeController.java
1612
1613      @Controller
1614      public class HomeController {
1615        @Autowired
1616        MemberService memberService;
1617
1618        @RequestMapping(value = "/create", method = RequestMethod.POST)
1619        public String home(MemberVO memberVo, Model model) {
1620          int row = this.memberService.create(memberVo);
1621          if(row == 1) model.addAttribute("status", "Insert Success");
1622          else model.addAttribute("status", "Insert Failure");
1623          return "create";    //   /WEB-INF/views/create.jsp
1624        }
1625      }
1626
1627    3)com.example.service/MemberServiceImpl.java
1628
1629      @Service("memberService")
1630      public class MemberServiceImpl implements MemberService {
1631        @Autowired
1632        MemberDao memberDao;
1633
1634        @Override
1635        public int create(MemberVO memberVo) {
1636          return this.memberDao.create(memberVo);
```

```
    }
  }

4)com.example.dao.MemberDaoImpl.java

  @Repository("memberDao")
  public class MemberDaoImpl implements MemberDao {
    @Autowired
    JdbcTemplate jdbcTemplate;

    @Override
    public int create(MemberVO memberVo) {
      String sql = "INSERT INTO Member VALUES(?,?,?,?,?)";
      return this.jdbcTemplate.update(sql, memberVo.getUserid(),
          memberVo.getUsername(), memberVo.getAge(),
          memberVo.getGender(), memberVo.getCity());
    }
  }

5)views/create.jsp

  <%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
  <!DOCTYPE html>
  <html>
  <head>
  <meta charset="UTF-8">
  <title>Insert title here</title>
  </head>
  <body>
    <h1>${status}</h1>
  </body>
  </html>

22. POST 발송시 한글 깨짐 처리하기
1)web.xml

  <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>

23. Test
1)http://localhost:8080/biz/static/register.html
2)http://localhost:8080/biz/create
  Insert Success


24. Data Select
1)HomeController.java
```

```
1694
1695      @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1696      public String view(@PathVariable String userid, Model model) {
1697        MemberVO memberVo = this.memberService.read(userid);
1698        model.addAttribute("member", memberVo);
1699        return "view";
1700      }
1701
1702    2)MemberServiceImpl.java
1703
1704      @Override
1705      public MemberVO read(String userid) {
1706        return this.memberDao.read(userid);
1707      }
1708
1709    3)MemberDaoImpl.java
1710
1711      @Override
1712      public MemberVO read(String userid) {
1713        String sql = "SELECT * FROM Member WHERE userid = ?";
1714        return this.jdbcTemplate.queryForObject(sql, new Object[] {userid},
1715                    new MyRowMapper());
1716      }
1717
1718      class MyRowMapper implements RowMapper<MemberVO>{
1719        @Override
1720        public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1721          MemberVO memberVo = new MemberVO(rs.getString("userid"),
1722              rs.getString("username"), rs.getInt("userage"),
1723              rs.getString("gender"), rs.getString("city"));
1724          return memberVo;
1725        }
1726      }
1727
1728    4)views/view.jsp
1729
1730      <%@ page language="java" contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
1731      <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
1732      <c:set var="user" value="${member}" />
1733      <!DOCTYPE html>
1734      <html>
1735      <head>
1736        <meta charset="UTF-8">
1737        <title>Insert title here</title>
1738        <script src="/biz/static/js/jquery-1.12.4.js"></script>
1739        <script>
1740          $(function(){
1741            $("#btnList").bind("click", function(){
1742              location.href = "/biz/list";
1743            });
1744            $("#btnDelete").bind("click", function(){
1745              location.href = "/biz/delete/${user.userid}";
1746            });
1747          });
1748        </script>
1749      </head>
1750      <body>
```

```
1751        <h1>${user.username}의정보</h1>
1752        <form action="/biz/update" method="post">
1753          <input type="hidden" name="userid" value = "${user.userid}" />
1754          <ul>
1755            <li>아이디 : ${user.userid }</li>
1756            <li>나이 : <input type='number' name="age" value='${user.age}' /></li>
1757            <li>성별 : <c:if test='${user.gender eq "남성"}'>
1758                 <input type="radio" name="gender" value="남성" checked />남성  
1759                 <input type="radio" name="gender" value="여성" />여성
1760                 </c:if>
1761                 <c:if test='${user.gender eq "여성"}'>
1762                 <input type="radio" name="gender" value="남성" />남성  
1763               <input type="radio" name="gender" value="여성" checked />여성
1764            </c:if>
1765            </li>
1766            <li>거주지 : <input type="text" name="city" value="${user.city }" /></li>
1767            <li><input type='submit' value='수정하기' /></li>
1768            <li><input type='button' value='삭제하기'  id="btnDelete"/></li>
1769            <li><input type='button' value='목록으로'  id="btnList"/></li>
1770          </ul>
1771        </form>
1772      </body>
1773      </html>
1774
1775    5)Test
1776      http://localhost:8080/biz/view/jimin
1777
1778
1779  25. Data List
1780    1)HomeController.java
1781
1782      @RequestMapping(value = "/list", method = RequestMethod.GET)
1783      public String list(Model model) {
1784        List<MemberVO> list = this.memberService.readAll();
1785        model.addAttribute("userlist", list);
1786        return "list";    // /WEB-INF/views/list.jsp
1787      }
1788
1789    2)MemberServiceImpl.java
1790
1791      @Override
1792      public List<MemberVO> readAll() {
1793        return this.memberDao.readAll();
1794      }
1795
1796    3)MemberDaoImpl.java
1797
1798      @Override
1799      public List<MemberVO> readAll() {
1800        String sql = "SELECT * FROM Member ORDER BY userid DESC";
1801        return this.jdbcTemplate.query(sql, new MyRowMapper());
1802      }
1803
1804    4)iews/list.jsp
1805
1806      <%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
1807      <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>Member List</h1>
  <table border='1'>
    <thead>
      <tr>
        <th>아이디</th><th>이름</th><th>나이</th><th>성별</th><th>거주지</th>

      </tr>
    </thead>
    <tbody>
      <c:forEach items="${userlist}" var="user">
        <tr>
          <td><a
          href="/biz/view/${user.userid}">${user.userid}</a></td><td>${user.userna
          me}</td>
          <td>${user.age}</td><td>${user.gender }</td>
          <td>${user.city }</td>
        </tr>
      </c:forEach>
    </tbody>
  </table>
</body>
</html>
```

5)Test
http://localhost:8080/biz/list

26. Data Delete

1)HomeController.java

```
@RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
public String delete(@PathVariable String userid) {
  this.memberService.delete(userid);
  return "redirect:/list";
}
```

2)MemberServiceImpl.java

```
@Override
public int delete(String userid) {
  return this.memberDao.delete(userid);
}
```

3)MemberDaoImpl.java

```
@Override
public int delete(String userid) {
  String sql = "DELETE FROM Member WHERE userid = ?";
  return this.jdbcTemplate.update(sql, userid);
}
```

4)Test
http://localhost:8080/biz/delete/chulsu

27. Data Update
1)HomeController.java

```
@RequestMapping(value = "/update", method = RequestMethod.POST)
public String update(@RequestParam("userid") String userid,
      @RequestParam("age") int age,
      @RequestParam("gender") String gender,
      @RequestParam("city") String city) {
   this.memberService.update(
         new MemberVO(userid, "", age, gender, city));
   return "redirect:/list";
}
```

2)MemberServiceImpl.java

```
@Override
public int update(MemberVO memberVo) {
   return this.memberDao.update(memberVo);
}
```

3)MemberDaoImpl.java

```
@Override
public int update(MemberVO memberVo) {
   String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " +
            "WHERE userid = ?";
   return this.jdbcTemplate.update(sql, memberVo.getAge(),
         memberVo.getGender(), memberVo.getCity(), memberVo.getUserid());
}
```

4)Test
http://localhost:8080/biz/list에서
해당 ID Click
데이터 수정
[수정하기] button click

28. All Codes
1)HomeController.java

```
package com.example.biz;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.example.service.MemberService;
import com.example.vo.MemberVO;
```

```
1921
1922      /**
1923       * Handles requests for the application home page.
1924       */
1925      @Controller
1926      public class HomeController {
1927        @Autowired
1928        MemberService memberService;
1929
1930        @RequestMapping(value = "/create", method = RequestMethod.POST)
1931        public String home(MemberVO memberVo, Model model) {
1932          int row = this.memberService.create(memberVo);
1933          if(row == 1) model.addAttribute("status", "Insert Success");
1934          else model.addAttribute("status", "Insert Failure");
1935          return "create";    //   /WEB-INF/views/create.jsp
1936        }
1937
1938        @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1939        public String view(@PathVariable String userid, Model model) {
1940          MemberVO memberVo = this.memberService.read(userid);
1941          model.addAttribute("member", memberVo);
1942          return "view";
1943        }
1944
1945        @RequestMapping(value = "/list", method = RequestMethod.GET)
1946        public String list(Model model) {
1947          List<MemberVO> list = this.memberService.readAll();
1948          model.addAttribute("userlist", list);
1949          return "list";    // /WEB-INF/views/list.jsp
1950        }
1951
1952        @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1953        public String delete(@PathVariable String userid) {
1954          this.memberService.delete(userid);
1955          return "redirect:/list";
1956        }
1957
1958        @RequestMapping(value = "/update", method = RequestMethod.POST)
1959        public String update(@RequestParam("userid") String userid,
1960            @RequestParam("age") int age,
1961            @RequestParam("gender") String gender,
1962            @RequestParam("city") String city) {
1963          this.memberService.update(
1964              new MemberVO(userid, "", age, gender, city));
1965          return "redirect:/list";
1966        }
1967      }
1968
1969    2)MemberServiceImpl.java
1970
1971      package com.example.service;
1972
1973      import java.util.List;
1974
1975      import org.springframework.beans.factory.annotation.Autowired;
1976      import org.springframework.stereotype.Service;
1977
1978      import com.example.dao.MemberDao;
```

```
1979    import com.example.vo.MemberVO;
1980
1981    @Service("memberService")
1982    public class MemberServiceImpl implements MemberService {
1983      @Autowired
1984      MemberDao memberDao;
1985
1986      @Override
1987      public int create(MemberVO memberVo) {
1988        return this.memberDao.create(memberVo);
1989      }
1990
1991      @Override
1992      public MemberVO read(String userid) {
1993        return this.memberDao.read(userid);
1994      }
1995
1996      @Override
1997      public List<MemberVO> readAll() {
1998        return this.memberDao.readAll();
1999      }
2000
2001      @Override
2002      public int update(MemberVO memberVo) {
2003        return this.memberDao.update(memberVo);
2004      }
2005
2006      @Override
2007      public int delete(String userid) {
2008        return this.memberDao.delete(userid);
2009      }
2010    }
2011
2012  3)MemberDaoImpl.java
2013
2014    package com.example.dao;
2015
2016    import java.sql.ResultSet;
2017    import java.sql.SQLException;
2018    import java.util.List;
2019
2020    import org.springframework.beans.factory.annotation.Autowired;
2021    import org.springframework.jdbc.core.JdbcTemplate;
2022    import org.springframework.jdbc.core.RowMapper;
2023    import org.springframework.stereotype.Repository;
2024
2025    import com.example.vo.MemberVO;
2026
2027    @Repository("memberDao")
2028    public class MemberDaoImpl implements MemberDao {
2029      @Autowired
2030      JdbcTemplate jdbcTemplate;
2031
2032      @Override
2033      public int create(MemberVO memberVo) {
2034        String sql = "INSERT INTO Member VALUES(?,?,?,?,?)";
2035        return this.jdbcTemplate.update(sql, memberVo.getUserid(),
          memberVo.getUsername(), memberVo.getAge(),
```

```
2036                memberVo.getGender(), memberVo.getCity());
2037        }
2038
2039        class MyRowMapper implements RowMapper<MemberVO> {
2040          @Override
2041          public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2042            MemberVO memberVo = new MemberVO(rs.getString("userid"),
              rs.getString("username"), rs.getInt("userage"),
2043                  rs.getString("gender"), rs.getString("city"));
2044            return memberVo;
2045          }
2046        }
2047
2048        @Override
2049        public MemberVO read(String userid) {
2050          String sql = "SELECT * FROM Member WHERE userid = ?";
2051          return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new
              MyRowMapper());
2052        }
2053
2054        @Override
2055        public List<MemberVO> readAll() {
2056          String sql = "SELECT * FROM Member ORDER BY userid DESC";
2057          return this.jdbcTemplate.query(sql, new MyRowMapper());
2058        }
2059
2060        @Override
2061        public int update(MemberVO memberVo) {
2062          String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " + "WHERE
              userid = ?";
2063          return this.jdbcTemplate.update(sql, memberVo.getAge(), memberVo.getGender(),
              memberVo.getCity(),
2064                memberVo.getUserid());
2065        }
2066
2067        @Override
2068        public int delete(String userid) {
2069          String sql = "DELETE FROM Member WHERE userid = ?";
2070          return this.jdbcTemplate.update(sql, userid);
2071        }
2072      }
2073
2074
2075  ------------------------------------------------------------
2076  Task7. Form Data Validation
2077  1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
2078  2. Select Spring MVC Project
2079  3. Project name : FormValidationDemo > Next
2080  4. Enter a topLevelPackage : com.example.biz > Finish
2081  5. pom.xml 수정하기
2082    <properties>
2083      <java-version>1.8</java-version>
2084      <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
2085      <org.aspectj-version>1.9.4</org.aspectj-version>
2086      <org.slf4j-version>1.7.28</org.slf4j-version>
2087    </properties>
2088      ...
2089    <dependency>
```

```
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>javax.servlet.jsp-api</artifactId>
      <version>2.3.3</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>

6. pom.xml > right-click > Run As > Maven install
   [INFO] BUILD SUCCESS

7. FormValidationDemo Project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
   Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close

8. UserVO 객체 생성
   1)src/main/java/com.example.vo package 생성
   2)src/main/java/com.example.vo.UserVO class

      package com.example.vo;

      public class UserVO {
        private String name;
        private int age;
        private String userid;
        public String getName() {
          return name;
        }
        public void setName(String name) {
          this.name = name;
        }
        public int getAge() {
          return age;
        }
        public void setAge(int age) {
          this.age = age;
        }
        public String getUserid() {
          return userid;
        }
        public void setUserid(String userid) {
          this.userid = userid;
        }
        @Override
        public String toString() {
          return "UserVO [name=" + name + ", age=" + age + ", userid=" + userid + "]";
        }
      }
```

9. Validator를 이용한 검증
1)Data Command 객체에서 유효성 검사를 할 수 있다.
2)UserValidator 객체 생성
3)src/main/java/com.example.biz.UserValidator class

```
package com.example.biz;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import com.example.vo.UserVO;

public class UserValidator implements Validator {

  @Override
  public boolean supports(Class<?> arg0) {
    //검증할 객체의 class 타입 정보를 반환
    return UserVO.class.isAssignableFrom(arg0);
  }

  @Override
  public void validate(Object obj, Errors errors) {
    System.out.println("검증시작");
    UserVO userVO = (UserVO)obj;

    String username = userVO.getName();
    if(username == null || username.trim().isEmpty()) {
      System.out.println("이름의 값이 빠졌습니다.");
      errors.rejectValue("name", "No Value");
    }

    int userage = userVO.getAge();
    if(userage == 0) {
      System.out.println("나이의 값이 빠졌습니다.");
      errors.rejectValue("age", "No Value");
    }

    String userid = userVO.getUserid();
    if(userid == null || userid.trim().isEmpty()) {
      System.out.println("아이디의 값이 빠졌습니다.");
      errors.rejectValue("userid", "No Value");
    }
  }
}
```

4)src/main/java/com.example.biz/HomeController.java

```
@RequestMapping(value = "/register", method=RequestMethod.GET)
public String register() {
  return "register";
}

@RequestMapping(value = "/register", method=RequestMethod.POST)
public String register(@ModelAttribute("userVO") UserVO userVO, BindingResult result) {
  String page = "register_ok";
  UserValidator validator = new UserValidator();
```

```
        validator.validate(userVO, result);
        if(result.hasErrors()) {
          page = "register";
        }
        return page;
      }
```

5)src/main/webapp/WEB-INF/views/register.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입 폼</title>
</head>
<body>
  <form action="/biz/register" method="post">
    Name : <input type="text" name="name" /><br />
    Age : <input type="number" name="age" /><br />
    ID : <input type="text" name="userid" /><br />
    <input type="submit" value="가입하기" />
  </form>
</body>
</html>
```

6)src/main/webapp/WEB-INF/views/register_ok.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="user" value="${userVO}" />
<!DOCTYPE html">
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입 결과 창</title>
</head>
<body>
  <ul>
    <li>이름 : ${user.name}</li>
    <li>나이 : ${user.age}</li>
    <li>아이디 : ${user.userid}</li>
  </ul>
</body>
</html>
```

7)Test

http://localhost:8080/biz/register에서
이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
한 개라도 입력하지 않으면 다시 입력창으로 간다.

10. ValidataionUtils class를 이용한 검증

1)ValidatationUtils class는 validate() method를 좀 더 편리하게 사용할 수 있게 해줌.
2)UserValidator.java 수정

```
        /*String username = userVO.getName();
        if(username == null || username.trim().isEmpty()) {
          System.out.println("이름의 값이 빠졌습니다.");
```

```
2262          errors.rejectValue("name", "No Value");
2263        }*/
2264
2265        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "No Value");
2266
2267  11. @Valid와 @InitBinder 이용하기
2268    1)Spring Framework이 대신 검증해 줌
2269    2)mvnrepository에서 'hibernate validator'로 검색
2270
2271      <dependency>
2272          <groupId>org.hibernate.validator</groupId>
2273          <artifactId>hibernate-validator</artifactId>
2274          <version>6.0.18.Final</version>
2275      </dependency>
2276
2277    3)pom.xml에 넣고 Maven Clean > Maven Install
2278    4)HomeController.java 수정
2279
2280      @RequestMapping(value = "/register", method=RequestMethod.POST)
2281      public String register(@ModelAttribute("userVO") @Valid UserVO userVO, BindingResult
          result) {
2282        String page = "register_ok";
2283        //UserValidator validator = new UserValidator();
2284        //validator.validate(userVO, result);
2285        if(result.hasErrors()) {
2286          page = "register";
2287        }
2288
2289        return page;
2290      }
2291
2292      @InitBinder
2293      protected void initBinder(WebDataBinder binder) {
2294        binder.setValidator(new UserValidator());
2295      }
2296
2297  12. Test
2298    http://localhost:8080/biz/register에서
2299    이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
2300    한 개라도 입력하지 않으면 다시 입력창으로 간다.
2301
2302
2303  ------------------------------------------------------------------------------
2304  Task8. Convert J2EE to Spring MVC
2305  1. In J2EE Perspective
2306  2. Project Explorer > right-click > New > Dynamic Web Project
2307  3. Project name : SpringWebDemo > Next > Check [Generate web.xml deployment
      descriptor] > Finish
2308  4. Convert to Maven Project
2309    1)project right-click > Configure > Convert to Maven Project > Finish
2310    2)Project : /SpringWebDemo
2311    3)Group Id : SpringWebDemo
2312    4)Artifact Id : SpringWebDemo
2313    5)version : 0.0.1-SNAPSHOT
2314    6)Packaging : war
2315    7)Finish
2316
2317  5. Add Spring Project Nature
```

-project right-click > Spring Tools > Add Spring Project Nature

6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
</dependencies>
```

7. Spring mvc library 검색 및 설치

1)http://mvnrepository.com에서 'spring mvc'로 검색

2)pom.xml에 추가

```
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.2.0.RELEASE</version>
  </dependency>
```

3)Maven Clean > Maven Install

8. Build path에 config foler 추가

1)project right-click > Build Path > Configure Build Path > Select [Source] tab

2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]

3)Folder name : config > Finish > OK > Apply and Close

4)Java Resources > config 폴더 확인

9. config folder에 beans.xml file 생성

1)Spring Perspective로 전환

2)config > right-click > New > Other > Spring > Spring Bean Configuration File > beans.xml

3)생성시 beans,context, mvc 체크

```
<?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"
  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-3.2.xsd">

</beans>
```

10. ContextLoaderListener class 설정

1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -

ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입

```
  <!-- needed for ContextLoaderListener -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>location</param-value>
  </context-param>

  <!-- Bootstraps the root web application context before servlet initialization -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
```

2)아래 code로 변환

```
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:beans.xml</param-value>
  </context-param>
```

11. DispatcherServlet Class 추가

1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet declaration] 선택하면 아래의 code가 자동 추가된다.

```
  <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
  <servlet>
    <servlet-name>springDispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>location</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <!-- Map all requests to the DispatcherServlet for handling -->
  <servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>url</url-pattern>
  </servlet-mapping>
```

2)아래의 code로 변환

```
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:beans*.xml</param-value>
  </init-param>

  <servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
```

12. mvnrepository에서 'jstl'로 검색 후 설치

1)목록에서 2번째 : 1.2버전

```
  <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
  <dependency>
```

```
2427          <groupId>javax.servlet</groupId>
2428          <artifactId>jstl</artifactId>
2429          <version>1.2</version>
2430      </dependency>
2431
2432    2)pom.xml에 붙여넣고 Maven Clean > Maven Install
2433
2434
2435  13. Hello Controller 작성
2436    1)src/com.example.vo package 생성
2437    2)src/com.example.vo.HelloVO class 생성
2438
2439      package com.example.vo;
2440
2441      public class HelloVO {
2442        private String name;
2443
2444        public void setName(String name) {
2445          this.name = name;
2446        }
2447
2448        public String sayHello() {
2449          return "Hello " + name;
2450        }
2451      }
2452
2453    3)src/com.example.controller package 생성
2454    4)com.example.controller.HelloController class 생성
2455
2456      package com.example.controller;
2457
2458      import org.springframework.beans.factory.annotation.Autowired;
2459      import org.springframework.stereotype.Controller;
2460      import org.springframework.ui.Model;
2461      import org.springframework.web.bind.annotation.RequestMapping;
2462
2463      import com.example.vo.HelloVO;
2464
2465      @Controller
2466      public class HelloController {
2467        @Autowired
2468        private HelloVO helloBean;
2469
2470        @RequestMapping("/hello.do")
2471        public String hello(Model model) {
2472          String msg = helloBean.sayHello();
2473          model.addAttribute("greet", msg);
2474          return "hello.jsp";
2475        }
2476      }
2477
2478
2479  14. beans.xml 수정
2480      <context:component-scan base-package="com.example" />
2481
2482      <bean id="helloVO" class="com.example.vo.HelloVO">
2483        <property name="name" value="한지민" />
2484      </bean>
```

15. WebContent/hello.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
  </head>
  <body>
    ${greet}
  </body>
</html>
```

16. project > right-click > Run As > Run on Server > Finish

17. http://localhost:8080/SpringWebDemo/hello.do
    Hello 한지민

Task9. Convert J2EE to Spring MVC
1. In J2EE Perspective
2. Project Explorer > right-click > New > Dynamic Web Project
3. Project name : SpringWebDemo1 > Next > Check [Generate web.xml deployment descriptor] > Finish
4. Convert to Maven Project
  -project right-click > Configure > Convert to Maven Project > Finish
5. Add Spring Project Nature
  -project right-click > Spring Tools > Add Spring Project Nature

6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>com.oracle</groupId>
```

```
        <artifactId>ojdbc6</artifactId>
        <version>11.2</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.2.0.RELEASE</version>
    </dependency>
  </dependencies>
```

2)Maven Clean > Maven Install

7. Build path에 config foler 추가
1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
3)Folder name : config > Finish > OK > Apply and Close
4)Java Resources > config 폴더 확인

8. config folder에 beans.xml file 생성
1)Spring Perspective로 전환
2)config Folder > right-click > New > Spring Bean Configuration File
3)File name : beans.xml
4)생성시 beans,context, mvc 체크

```
    <?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"
      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-3.2.xsd">


    </beans>
```

9. ContextLoaderListener class 설정
1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener - ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입

```
    <!-- needed for ContextLoaderListener -->
    <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>location</param-value>
    </context-param>

    <!-- Bootstraps the root web application context before servlet initialization -->
    <listener>
      <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
```

2)아래 코드로 변환

```
    <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:beans.xml</param-value>
```

```
    </context-param>


10. DispatcherServlet Class 추가
  1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.

    <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
    <servlet>
      <servlet-name>springDispatcherServlet</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>location</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
    </servlet>

    <!-- Map all requests to the DispatcherServlet for handling -->
    <servlet-mapping>
      <servlet-name>springDispatcherServlet</servlet-name>
      <url-pattern>url</url-pattern>
    </servlet-mapping>

  2)아래의 코드로 변환
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:beans*.xml</param-value>
    </init-param>

    <servlet-mapping>
      <servlet-name>springDispatcherServlet</servlet-name>
      <url-pattern>*.do</url-pattern>
    </servlet-mapping>


11. UserVO class 생성
  1)src/com.example.vo package 생성
  2)src/com.example.vo.UserVO class 생성

    package com.example.vo;

    public class UserVO {
      private String userId;
      private String name;
      private String gender;
      private String city;
      public UserVO() {}
      public UserVO(String userId, String name, String gender, String city) {
        this.userId = userId;
        this.name = name;
        this.gender = gender;
        this.city = city;
      }
      public String getUserId() {
        return userId;
      }
```

```
2651        public void setUserId(String userId) {
2652          this.userId = userId;
2653        }
2654        public String getName() {
2655          return name;
2656        }
2657        public void setName(String name) {
2658          this.name = name;
2659        }
2660        public String getGender() {
2661          return gender;
2662        }
2663        public void setGender(String gender) {
2664          this.gender = gender;
2665        }
2666        public String getCity() {
2667          return city;
2668        }
2669        public void setCity(String city) {
2670          this.city = city;
2671        }
2672        @Override
2673        public String toString() {
2674          return "UserVO [userId=" + userId + ", name=" + name + ", gender=" + gender +
           ", city=" + city + "]";
2675        }
2676      }
2677
2678
2679  12. UserDao 객체 생성
2680    1)src/com.example.dao package 생성
2681    2)src/com.example.dao.UserDao interface
2682
2683      package com.example.dao;
2684
2685      import java.util.List;
2686
2687      import com.example.vo.UserVO;
2688
2689      public interface UserDao {
2690        void insert(UserVO user);
2691
2692        List<UserVO> readAll();
2693
2694        void update(UserVO user);
2695
2696        void delete(String id);
2697
2698        UserVO read(String id);
2699      }
2700
2701      -src/com.example.dao.UserDaoImplJDBC.java 생성
2702
2703        package com.example.dao;
2704
2705        import java.sql.ResultSet;
2706        import java.sql.SQLException;
2707        import java.util.List;
```

```
2708
2709        import javax.sql.DataSource;
2710
2711        import org.springframework.beans.factory.annotation.Autowired;
2712        import org.springframework.dao.EmptyResultDataAccessException;
2713        import org.springframework.jdbc.core.JdbcTemplate;
2714        import org.springframework.jdbc.core.RowMapper;
2715        import org.springframework.stereotype.Repository;
2716
2717        import com.example.vo.UserVO;
2718
2719        @Repository("userDao")
2720        public class UserDaoImplJDBC implements UserDao {
2721
2722          private JdbcTemplate jdbcTemplate;
2723
2724          @Autowired
2725          public void setDataSource(DataSource dataSource) {
2726            this.jdbcTemplate = new JdbcTemplate(dataSource);
2727          }
2728
2729          class UserMapper implements RowMapper<UserVO> {
2730            public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2731              UserVO user = new UserVO();
2732              user.setUserId(rs.getString("userId"));
2733              user.setName(rs.getString("name"));
2734              user.setGender(rs.getString("gender"));
2735              user.setCity(rs.getString("city"));
2736              return user;
2737            }
2738          }
2739
2740          @Override
2741          public void insert(UserVO user) {
2742            String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
2743            jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
              user.getCity());
2744
2745            System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
              user.getName());
2746          }
2747
2748          @Override
2749          public List<UserVO> readAll() {
2750            String SQL = "SELECT * FROM users";
2751            List<UserVO> userList = jdbcTemplate.query(SQL, new UserMapper());
2752            return userList;
2753          }
2754
2755          @Override
2756          public void update(UserVO user) {
2757            String SQL = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid =
              ?";
2758            jdbcTemplate.update(SQL, user.getName(), user.getGender(), user.getCity(),
              user.getUserId());
2759            System.out.println("갱신된 Record with ID = " + user.getUserId());
2760          }
2761
```

```
2762          @Override
2763          public void delete(String id) {
2764            String SQL = "DELETE FROM users WHERE userid = ?";
2765            jdbcTemplate.update(SQL, id);
2766            System.out.println("삭제된 Record with ID = " + id);
2767          }
2768
2769          @Override
2770          public UserVO read(String id) {
2771            String SQL = "SELECT * FROM users WHERE userid = ?";
2772            try {
2773              UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new
                  UserMapper());
2774              return user;
2775            } catch (EmptyResultDataAccessException e) {
2776              return null;
2777            }
2778          }
2779        }
2780
2781
2782  13. UserService 객체 생성
2783    1)src/com.example.service package 생성
2784    2)src/com.example.service.UserService interface
2785
2786      package com.example.service;
2787
2788      import java.util.List;
2789
2790      import com.example.vo.UserVO;
2791
2792      public interface UserService {
2793        void insertUser(UserVO user);
2794
2795        List<UserVO> getUserList();
2796
2797        void deleteUser(String id);
2798
2799        UserVO getUser(String id);
2800
2801        void updateUser(UserVO user);
2802      }
2803
2804    3)src/com.example.service.UserServiceImpl.java
2805
2806      package com.example.service;
2807
2808      import java.util.List;
2809
2810      import org.springframework.beans.factory.annotation.Autowired;
2811      import org.springframework.stereotype.Service;
2812
2813      import com.example.dao.UserDao;
2814      import com.example.vo.UserVO;
2815
2816      @Service("userService")
2817      public class UserServiceImpl implements UserService {
2818        @Autowired
```

```
2819        UserDao userDao;
2820
2821        @Override
2822        public void insertUser(UserVO user) {
2823          this.userDao.insert(user);
2824        }
2825
2826        @Override
2827        public List<UserVO> getUserList() {
2828          return this.userDao.readAll();
2829        }
2830
2831        @Override
2832        public void deleteUser(String id) {
2833          this.userDao.delete(id);
2834        }
2835
2836        @Override
2837        public UserVO getUser(String id) {
2838          return this.userDao.read(id);
2839        }
2840
2841        @Override
2842        public void updateUser(UserVO user) {
2843          this.userDao.update(user);
2844        }
2845      }
2846
2847
2848  14. UserController 객체 생성
2849    1)src/com.example.controller package 생성
2850    2)com.example.controller.UserController class 생성
2851
2852      package com.example.controller;
2853
2854      import org.springframework.beans.factory.annotation.Autowired;
2855      import org.springframework.stereotype.Controller;
2856
2857      import com.example.service.UserService;
2858
2859      @Controller
2860      public class UserController {
2861        @Autowired
2862        private UserService userService;
2863
2864        @RequestMapping("/userInfo.do")
2865        public String getUserList(@RequestParam("userId") String userId, Model model) {
2866          UserVO user = userService.getUser(userId);
2867          //System.out.println(user);
2868          model.addAttribute("user", user);
2869          return "userInfo.jsp";
2870        }
2871      }
2872
2873
2874  15. config/dbinfo.properties file 생성
2875
2876    db.driverClass=oracle.jdbc.driver.OracleDriver
```

```
db.url=jdbc:oracle:thin:@localhost:1521:XE
db.username=hr
db.password=hr


16. beans.xml 수정

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

  <context:property-placeholder location="classpath:dbinfo.properties" />
  <bean id="dataSource"
  class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
    <property name="driverClass" value="${db.driverClass}" />
    <property name="url" value="${db.url}" />
    <property name="username" value="${db.username}" />
    <property name="password" value="${db.password}" />
  </bean>


17. WebContent/index.jsp 생성

  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  <c:redirect url="userInfo.do" />


18. WebContent/userInfo.jsp 생성

  <%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <c:set var="user" value="${user}"/>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
      <h1>userInfo.jsp</h1>
      <h2>사용자 정보</h2>
      아이디 : ${user.userId} <br />
      이름 : ${user.name} <br />
      성별 : ${user.gender} <br />
      도시 : ${user.city} <br />
    </body>
    </html>


19. project > right-click > Run As > Run on Server > Finish

20. http://localhost:8080/SpringWebDemo/userinfo.do?userId=scott


------------------------------------------------------------
Task10. File Upload with Spring MVC
1. In J2EE Perspective
2. Project Explorer > right-click > New > Dynamic Web Project
3. Project name : FileUploadDemo > Next > Check [Generate web.xml deployment
```

descriptor] > Finish
4. Convert to Maven Project
1)project right-click > Configure > Convert to Maven Project > Finish

5. Add Spring Project Nature
1)project right-click > Spring Tools > Add Spring Project Nature

6. 새로 생선된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
     <groupId>org.springframework</groupId>
     <artifactId>spring-webmvc</artifactId>
     <version>5.2.0.RELEASE</version>
  </dependency>
</dependencies>
```

7. ContextLoaderListener class 설정
1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener - ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입

```
  <!-- needed for ContextLoaderListener -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>location</param-value>
  </context-param>

  <!-- Bootstraps the root web application context before servlet initialization -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
```

2)아래 코드로 변환

```
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>
```

8. DispatcherServlet Class 추가
1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.

```
  <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
  <servlet>
    <servlet-name>springDispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>location</param-value>
    </init-param>
```

```
        <load-on-startup>1</load-on-startup>
      </servlet>

      <!-- Map all requests to the DispatcherServlet for handling -->
      <servlet-mapping>
        <servlet-name>springDispatcherServlet</servlet-name>
        <url-pattern>url</url-pattern>
      </servlet-mapping>

    2)아래의 코드로 변환
      <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:beans.xml</param-value>
      </init-param>

      <servlet-mapping>
        <servlet-name>springDispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
      </servlet-mapping>


9. FileUpload library 추가
  1)Apache에서 제공하는 Common FileUpload library를 사용하여 file upload를 처리하기 위한 library
  2)mvnrepository에서 'common fileupload'라고 검색하여 library 추가
      <dependency>
          <groupId>commons-fileupload</groupId>
          <artifactId>commons-fileupload</artifactId>
          <version>1.4</version>
      </dependency>

  3)mvnrepository에서 'commons io'라고 검색하여 library 추가
      <dependency>
          <groupId>commons-io</groupId>
          <artifactId>commons-io</artifactId>
          <version>2.6</version>
      </dependency>

  4)Maven Clean > Maven Install


10. Thumbnail Image Library 추가
  1)mvnrepository에서 'imgscalr-lib'라고 검색하여 libary 추가
      <dependency>
          <groupId>org.imgscalr</groupId>
          <artifactId>imgscalr-lib</artifactId>
          <version>4.2</version>
      </dependency>

  2)Maven Clean > Maven Install


11. Build path에 config Foler 추가
  1)project right-click > Build Path > Configure Build Path > Select [Source] tab
  2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
  3)Folder name : config > Finish > OK > Apply and Close
  4)Java Resources > config 폴더 확인


```

12. config Folder에 applicationContext.xml file 생성
  1)config > right-click > New > Other > Spring > Spring Bean Configuration File
  2)Name : applicationContext.xml
  3)Namespace Tab에서 context, mvc check 할 것

```
<?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:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc
  http://www.springframework.org/schema/mvc/spring-mvc-4.3.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">

<context:component-scan
    base-package="com.example" />
  <mvc:annotation-driven />
</beans>
```

13. config Folder에 beans.xml file 생성
  1)config > right-click > New > Other > Spring > Spring Bean Configuration File
  2)Name : beans.xml
  3)beans.xml에 Spring mulipartResolver 추가

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
  <property name="maxUploadSize" value="10240000" />
  <property name="defaultEncoding" value="utf-8" />
</bean>
```

14. web.xml에 한글 File Encoding 처리하기
  1)한글 File이 Upload될 때 File 명이 깨지는 것을 해결하기 위해 web.xml에 아래 내용을 추가한다.

```
<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>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

15. View 작성
  1)WebContent/form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>file 업로드 예제</h1>
  <form method="post" action="upload" enctype="multipart/form-data">
    <label>email:</label> <input type="text" name="email"> <br>
    <br> <label>file:</label> <input type="file" name="file1">
    <br>
    <br> <input type="submit" value="upload">
  </form>
</body>
</html>
```

16. Service 작성
  1)src/com.example.service package
  2)src/com.example.service.FileUploadService.java

```
package com.example.service;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;

import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

  @Service
  public class FileUploadService {
    // 리눅스 기준으로 file 경로를 작성 ( 루트 경로인 /으로 시작한다. )
    // 윈도우라면 workspace의 드라이브를 파악하여 JVM이 알아서 처리해준다.
    // 따라서 workspace가 C드라이브에 있다면 C드라이브에 upload 폴더를 생성해 놓아야 한다.
    private static final String SAVE_PATH = "/upload";
    private static final String PREFIX_URL = "/upload/";

    public String restore(MultipartFile multipartFile) {
      String uri = null;

      try {
        // file 정보
        String originFilename = multipartFile.getOriginalFilename();
        String extName = originFilename.substring(originFilename.lastIndexOf("."),
        originFilename.length());
        Long size = multipartFile.getSize();

        // 서버에서 저장 할 file 이름
        String saveFileName = genSaveFileName(extName);

        System.out.println("originFilename : " + originFilename);
        System.out.println("extensionName : " + extName);
        System.out.println("size : " + size);
```

```
3154            System.out.println("saveFileName : " + saveFileName);
3155
3156            writeFile(multipartFile, saveFileName);
3157            uri = PREFIX_URL + saveFileName;
3158           }
3159           catch (IOException e) {
3160            // 원래라면 RuntimeException 을 상속받은 예외가 처리되어야 하지만
3161            // 편의상 RuntimeException을 던진다.
3162            // throw new FileUploadException();
3163            throw new RuntimeException(e);
3164           }
3165           return uri;
3166          }
3167
3168
3169          // 현재 시간을 기준으로 file 이름 생성
3170          private String genSaveFileName(String extName) {
3171           String fileName = "";
3172
3173           Calendar calendar = Calendar.getInstance();
3174           fileName += calendar.get(Calendar.YEAR);
3175           fileName += calendar.get(Calendar.MONTH);
3176           fileName += calendar.get(Calendar.DATE);
3177           fileName += calendar.get(Calendar.HOUR);
3178           fileName += calendar.get(Calendar.MINUTE);
3179           fileName += calendar.get(Calendar.SECOND);
3180           fileName += calendar.get(Calendar.MILLISECOND);
3181           fileName += extName;
3182
3183           return fileName;
3184          }
3185
3186
3187          // file을 실제로 write 하는 메서드
3188          private boolean writeFile(MultipartFile multipartFile, String saveFileName)
3189                    throws IOException{
3190           boolean result = false;
3191
3192           byte[] data = multipartFile.getBytes();
3193           FileOutputStream fos = new FileOutputStream(SAVE_PATH + "/" + saveFileName);
3194           fos.write(data);
3195           fos.close();
3196
3197           return result;
3198          }
3199         }
3200
3201    3)SAVE_PATH는 file을 저장할 위치를 가리킨다.
3202      -일반적으로 server는 Linux 기반이므로 Linux 경로명을 사용하는 것이 좋다.
3203      -즉 file을 root 경로인 / 아래의 upload folder에 저장하겠다는 의미인데, Windows에서는 JVM이 알아서
          workspace가 존재하는 drive의 위치를 찾아서 drive를 root 경로로 하여 upload folder에 저장한다.
3204      -예를들어 Eclipse workspace가 C drive에 있다면 C drive의 upload folder에 file이 저장될 것이다.
3205    4)PREFIX_URL은 저장된 file을 JSP에서 불러오기 위한 경로를 의미한다.
3206    5)MultipartFile 객체는 file의 정보를 담고 있다.
3207    6)uri을 반환하는 이유는 view page에서 바로 image file을 보기 위함이다.
3208      -만약 DB에서 image 경로를 저장 해야 한다면, 이와 같이 uri을 반환하면 좋을 것이다.
3209    7)현재 시간을 기준으로 file 이름을 바꾼다.
3210      -이렇게 하는 이유는, 여러 사용자가 올린 file의 이름이 같을 경우 덮어 씌어지는 문제가 발생하기 때문이다.
```

-따라서 file 이름이 중복될 수 있는 문제를 해결하기 위해 ms단위의 시스템 시간을 이용하여 file 이름을 변경한다.
8)FileOutputStream 객체를 이용하여 file을 저장한다.

17. Controller 작성
1)com.example.controller package
2)com.example.controller.FileUploadController.java

```
package com.example.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import com.example.service.FileUploadService;

@Controller
public class FileUploadController {
  @Autowired
  FileUploadService fileUploadService;

  @RequestMapping("/form")
  public String form() {
    return "form.jsp";
  }

  @RequestMapping(value = "/upload", method = RequestMethod.POST)
  public String upload(@RequestParam("email") String email, @RequestParam("file1")
  MultipartFile file, Model model) {
    String uri = fileUploadService.restore(file);
    model.addAttribute("uri", uri);
    return "result.jsp";
  }
}
```

18. WebContent/result.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>Upload completed</h1>
  <div class="result-images">
    <img src="${pageContext.request.contextPath }${uri }" style="width: 150px">
  </div>
  <p>
    <a href='/FileUploadDemo/form'> 다시 업로드 하기 </a>
  </p>
</body>
```

```
</html>
```

19. C:/(현재 workspace가 C:라면)upload Folder 생성할 것

20. Project > right-click > Run As > Run on Server
http://localhost:8080/FileUploadDemo/form

21. 문제점 및 해결
1)Upload Folder(C:/upload)를 보면 File이 Upload된 것을 확인할 수 있지만, 결과 화면을 보면 Image가 제대로 출력 되지 않을 것이다.
2)Image File을 right-click하여 경로를 보면 아마 다음과 같을 것이다.
3)http://localhost:8080/FileUploadDemo/upload/업로드한 파일
4)File을 저장할 때 [upload]라는 Folder에 저장을 했는데, File을 저장할 때의 Upload는 C Drive 내의 [upload] Folder이고,
5)위 URL에서 [upload]는 Application 상 경로에 있는 upload이므로 WEB-INF 폴더의 하위 folder로서의 upload를 의미한다.
6)즉 실제 File이 저장된 Server 상의 위치(물리 주소)와 Application에서 보여주고자 하는 File 경로(가상 주소)가 일치하지 않은 것이다.
7)따라서 실제 File이 저장되어 있는 위치와 Application 상의 위치를 일치시키는 작업이 필요하다.
8)beans.xml에 물리 주소와 가상 주소를 mapping 해주는 code를 추가하도록 해야한다.

```
<!-- resource mapping -->
<!-- location : 물리적 주소 / mapping : 가상 주소 -->
<mvc:resources location="file:///C:/upload/" mapping="/upload/*"/>
```

9)이제 정상적으로 result.jsp에서 image가 출력될 것이다.

22. Multiple File Upload
1)이번에는 여러 개의 File을 Upload 할 수 있는 Multiple Upload를 알아보자.
2)수정할 부분은 <input> tag와 Controller에서 MultipartFile 객체를 받는 Parameter 부분 두 곳인데, 필요한 부분만 보자.
3)form.jsp

```
<input type="file" name="files" multiple>
```

4)<input> 태그에서는 multiple 속성만 추가하면 된다.
5)"File선택"을 클릭하면 ctrl 키를 눌러서 여러 개의 File을 선택할 수 있다.

6)FileUploadController

```
@RequestMapping( "/upload" )
public String upload(@RequestParam String email,
   @RequestParam(required=false) List<MultipartFile> files, Model model) {

      ...

}
```

7)Controller에서는 여러 개의 File을 받기 때문에 MultipartFile을 List로 받아야 한다.