```
1. Open API?
 1
 2
      1)개방형 API
 3
      2)프로그래밍에서 사용할 수 있는 개방되어 있는 상태의 interface를 말한다.
 4
      3)Portal이나 통계청, 기상청 등과 같은 관공서에서 가지고 있는 데이터를 외부 응용 프로그램에서 사용할
      수 있도록 Open API를 제공하고 있다.
 5
      4)Open API와 함께 사용하는 기술 중 REST가 있고, 대부분의 Open API는 REST 방식으로 지원되고
      있다.
 6
 7
 8
   2. RESTful 웹 서비스 개요
 9
      1)REST(REpresentational Safe Trasfer)
        -HTTP URI + HTTP Method
10
11
        -HTTP URI를 통해 제어할 자원(Resource)을 명시하고, HTTP Method(GET, POST, PUT,
        DELETE)를 통해 해당 Resource를 제어하는 명령을 내리는 방식의 아키텍처
12
        -HTTP protocol에 정의된 4개의 method들이 Resource에 대한 CRUD Operation을 정의
13
          --POST : Create(Insert)
14
          --GET : Read(Select)
15
          --PUT: Update or Create
16
          --DELETE : Delete
17
18
      2) RESTful API?
        -HTTP와 URI 기반으로 자원에 접근할 수 있도록 제공하는 Application 개발 interface.
19
20
        -즉, REST의 원리를 따르는 System을 가리키는 용어로 사용
21
        -기존의 웹 접근 방식과 RESTful API 방식과의 차이점(예:게시판)
22
                   기존 게시판
                                                      RESTful API를 지원하는 게시판
23
          --글읽기: GET /list.do?no=4&name=Spring
                                                    GET /board/Spring/4
24
          --글등록: POST /insert.do
                                                      POST /board/Spring/4
25
          --글삭제: GET /delete.do?no=4&name=Spring
                                                    DELETE /board/Spring/4
26
                                                    PUT /board/Spring/4
          --글수정: POST /update.do
27
28
        -기존의 게시판은 GET과 POST만으로 자원에 대한 CRUD를 처리하며, URI는 Action을 나타낸다.
29
        -RESTful 게시판은 4가지 메소드를 모두 사용하여 CRUD를 처리하며, URI는 제어하려는 자원을
        나타낸다.
30
31
32
   3. JSON과 XML
33
      1)RESTful 웹 서비스와 JSON XML.png 그림 참조
34
35
      2)JSON(JavaScript Object Notation)?
36
        -http://www.json.org
37
        -경량의 Data 교환 포맷
38
        -JavaScript에서 객체를 만들 때 사용하는 표현식을 의미
39
        -JSON 표현식은 사람과 기계 모두 이해하기 쉬우며 용량이 작아서, 최근에는 XML을 대체해서 데이터
        전송등에 많이 사용된다.
40
        -특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서 JSON 포맷의 데이터를 핸들링할 수 있는
        library를 제공하고 있다.
41
        -name : value 형식의 pair
42
          {
43
             "name": "조용필",
44
             "gender": "남성",
45
             "age": 50,
             "citv": "Seoul".
46
             "hobby": ["등산", "낚시", "게임"]
47
48
          }
49
50
      3)JSON library - Jackson
51
        -http://jackson.codehous.org
52
        -High-Performance JSON Processor!
        -Jackson은 JSON 형태를 Java 객체로, Java 객체를 JSON 형태로 변환해 주는 Java용 JSON
53
        library이다.
```

```
54
          -가장 많이 사용하는 JSON library이다.
 55
          JSON(Browser) <---> Java Object(Back-end) <---> RDBMS(Storage)
 56
 57
                         Jackson
                                                    Mybatis
 58
 59
60
       4)XML?
61
          -eXtensible Markup Language
62
          -Data를 저장하고 전달/교환하기 위한 언어
63
          -인간/기계 모두에게 읽기 편한 언어
64
          -데이터의 구조와 의미를 설명
65
          -HTML이 Data의 표현에 중점을 두었다면 XML은 Data를 전달하는 것에 중점을 맞춘 언어
66
          -HTML은 미리 정의된 Tag만 사용 가능하지만, XML은 사용자가 Tag를 정의할 수 있다.
67
          <?xml version="1.0" encoding="UTF-8"?>
68
          cproducts>
 69
            cproduct>
 70
               <name>Ballpen</name>
 71
               <price 단위="원">150</price>
 72
               <maker>모나미</maker>
 73
               <color>black</color>
 74
            </product>
 75
          </products>
 76
 77
 78
       5)Jackson version 1 library 설치
 79
          -http://mvnrepository.com에서 'jackson mapper'로 검색
          -'Data Mapper For Jackson' 1.9.13 버전을 pom.xml에 추가
80
81
          <!-- https://mvnrepository.com/artifact/org.codehaus.jackson/jackson-mapper-lqpl -->
82
            <dependency>
83
            <groupId>org.codehaus.jackson
84
            <artifactId>jackson-mapper-lgpl</artifactId>
85
            <version>1.9.13</version>
86
          </dependency>
87
88
       6)Jackson2 API 설치
89
          -http://mvnrepository.com에서 'jackson databind'로 검색
          -'Jackson Databind' 2.9.5 버전을 pom.xml에 추가
90
91
          -'Jackson Core' 2.9.5 버전을 pom.xml에 추가
92
          -'Jackson Annotations' 2.9.5 버전을 pom.xml에 추가
 93
94
            <!--
            https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
95
            <dependency>
96
               <groupId>com.fasterxml.jackson.core</groupId>
97
               <artifactId>jackson-databind</artifactId>
               <version>2.9.5</version>
98
99
            </dependency>
100
            <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core
            -->
101
            <dependency>
               <groupId>com.fasterxml.jackson.core</groupId>
102
               <artifactId>jackson-core</artifactId>
103
104
               <version>2.9.5</version>
105
            </dependency>
            <!--
106
            https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations
             -->
            <dependency>
107
108
               <groupId>com.fasterxml.jackson.core</groupId>
109
               <artifactId>jackson-annotations</artifactId>
```

```
110
              <version>2.9.5</version>
111
           </dependency>
112
113
       7)web.xml의 DispatcherServlet url-pattern 변경
         --기존--
114
115
         <servlet-mapping>
116
           <servlet-name>springDispatcherServlet</servlet-name>
           <url-pattern>*.do</url-pattern>
117
118
         </servlet-mapping>
119
120
         --변경--
121
         <servlet-mapping>
122
           <servlet-name>springDispatcherServlet</servlet-name>
123
            <url-pattern>/</url-pattern>
124
         </servlet-mapping>
125
126
       8)Spring Bean Configuration File(beans.xml) 설정
127
         -Spring MVC에 필요한 Bean들을 자동으로 등록해주는 Tag
128
           <mvc:annotation-driven />
129
130
       9)Spring MVC기반 RESTful 웹 서비스 구현 절차
         -RESTful 웹서비스를 처리할 RestfulController 클래스 작성 및 Spring Bean으로 등록
131
         -요청을 처리할 메소드에 @RequestMapping @RequestBody와 @ResponseBody annotation
132
133
         -REST Client Tool(Postman)을 사용하여 각각의 메소드 테스트
134
         -Ajax 통신을 하여 RESTful 웹서비스를 호출하는 HTML 페이지 작성
135
136
       10)사용자 관리 RESTful 웹서비스 URI와 Method
         Action Resource URI HTTP Method
137
138
         -사용자 목록 /users GET
139
         -사용자 보기 /users/{id} GET
         -사용자 등록 /users POST
140
141
         -사용자 수정 /users PUT
142
         -사용자 삭제 /user/{id} DELETE
143
144
       11)RESTful Controller를 위한 핵심 Annotation
145
         -Spring MVC에서는 Client에서 전송한 XML이나 JSON 데이터를 Controller에서 Java 객체로
         변환해서 받을 수 있는 기능(수신)을 제공하고 있다.
146
         -Java객체를 XML이나 JSON으로 변환해서 전송할 수 있는 기능(송신)을 제공하고 있다.
147
148
         -Annotation 설명
149
           --@RequestBody: HTTP Request Body(요청 몸체)를 Java객체로 전달받을 수 있다.
           --@ResponseBody: Java객체를 HTTP Response Body(응답 몸체)로 전송할 수 있다.
150
151
152
    4. Google Postman 설치
153
154
       https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop1
       30
155
       2) [앱실행]버튼 클릭
156
       3)Log in
157
158
159
    5. 데이터 변환 - JSON으로 변환
160
       1)시스템이 복잡해지면서 다른 시스템과 정보를 주고받을 일이 발생하는데, 이 때 데이터 교환 포맷으로
       JSON을 사용할 수 있다.
161
       2)검색결과를 JSON 데이터로 변환하려면 가장 먼저 jackson2 라이브러리를 다운받아야 한다.
162
       3)Jackson2는 자바 객체를 JSON으로 변환하거나 JSON을 자바 객체로 변환해주는 라이브러리다.
163
       4)
       https://www.concretepage.com/spring-4/spring-4-rest-xml-response-example-with-jackso
```

```
<u>n-2</u> 참조
164
       5)https://www.mkyong.com/java/jackson-2-convert-java-object-to-from-json/ 참조
165
       6)pom.xml에 다음과 같이 dependency를 추가한다.
166
          <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind
          -->
167
          <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
168
169
            <artifactId>jackson-databind</artifactId>
            <version>2.9.5</version>
170
171
          </dependency>
          <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
172
173
          <dependency>
174
            <groupId>com.fasterxml.jackson.core</groupId>
175
            <artifactId>jackson-core</artifactId>
176
            <version>2.9.5</version>
177
          </dependency>
178
          <!--
          https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations
          -->
179
          <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
180
            <artifactId>jackson-annotations</artifactId>
181
182
            <version>2.9.5</version>
183
          </dependency>
184
185
       7)Maven clean > Maven Install하면 Maven Dependencies에 아래와 같은 jar파일이 추가된다.
          -jackson-databind-2.9.5.jar
186
187
          -jackson-annotations-2.9.5.jar
          -jackson-core-2.9.5.jar
188
189
190
       8)보통 User가 Servlet이나 JSP를 요청하면 서버는 요청한 파일을 찾아서 실행한다.
191
       9)그 실행결과는 HTTP Response package의 body에 저장하여 Browser에 전송한다.
192
       10)그런데, 이 응답결과를 HTML이 아니라 JSON이나 XML로 변환하여 body에 저장하려면 Spring에서
       제공하는 변환기(Converter)를 사용해야 한다.
       11)Spring은 HttpMessageConverter를 구현한 다양한 변환기를 제공한다.
193
194
       12)이 변환기를 이용하면 Java 객체를 다양한 타입으로 변환하여 HTTP Response body에 설정할 수 있다.
195
       13)HttpMessageConverter를 구현한 클래스는 여러가지가 있으며, 이 중에서 Java 객체를 JSON
       responsebody로 변환할 때는 MappingJackson2HttpMessageConverter를 사용한다.
196
       14)따라서 MappingJackson2HttpMessageConverter를 Spring 설정 파일에 등록하면 되는데, 혹시
       이후에 XML 변환도 처리할 예정이라면 다음처럼 설정한다.
197
          <mvc:annotation-driven />
198
       15)Spring Bean Configuration File에 위와 같이 설정하면 HttpMessageConverter를 구현한 모든
       변환기가 생성된다.
199
       16)src/com.javasoft.controller.UserController.java에 다음과 같이 수정한다.
          package com.javasoft.controller;
200
201
202
          import org.springframework.beans.factory.annotation.Autowired;
203
          import org.springframework.stereotype.Controller;
204
          import org.springframework.web.bind.annotation.RequestMapping;
          import org.springframework.web.bind.annotation.RequestParam;
205
206
          import org.springframework.web.bind.annotation.ResponseBody;
207
208
          import com.javasoft.service.UserService;
209
          import com.javasoft.vo.UserVO;
210
211
          @Controller
212
          public class UserController {
213
            @Autowired
214
            private UserService userService;
215
```

```
/*@RequestMapping("/userinfo.do")
216
            public String getUserList(@RequestParam("userId") String userId, Model model) {
217
218
               UserVO user = userService.getUser(userId);
               model.addAttribute("user", user);
219
220
               return "userinfo.jsp";
221
            }*/
222
223
            @RequestMapping("/userinfo.do")
            @ResponseBody
224
225
            public UserVO userinfo(@RequestParam("userId") String userId) {
226
               return userService.getUser(userId);
227
            }
          }
228
229
230
       17)이전 메소드와 달리 @ResponseBody라는 annotation을 추가했는데, Java 객체를 Http
       Response 프로토콜의 body로 변환하기 위해 사용된다.
       18)이미 Spring Configuration File에 <mvc:annotation-driven>을 추가했기 때문에
231
       @ResponseBody가 적용된 메소드의 실행 결과는 JSON으로 변환되어 HTTP Response Body에
       다음과 같이 설정된다.
232
          {"userId":"jimin","name":"한지민","gender":"여","city":"서울"}
233
234
235
       19)만일 이때, Java 객체를 JSON으로 변환할 때, 특정 변수를 제외시키려면 @JsonIgnore
       annotation을 해당 변수의 getter에 설정하면 된다.
236
237
          package com.javasoft.vo;
238
          import com.fasterxml.jackson.annotation.JsonIgnore;
239
          public class UserVO {
240
241
            @JsonIgnore
242
            public String getGender() {
243
               return gender;
244
            }
245
246
       20)이렇게 하면 아래와 같이 성별이 포함되지 않는다는 것을 알 수 있다.
247
          {"userId":"jimin","name":"한지민","city":"서울"}
248
249
250
       21)Postman test
251
          GET http://localhost:8080/SpringWebDemo/userinfo.do/jimin Send
252
253
          Body JSON
254
255
256
            "userId": "jimin",
            "name": "한지민",
257
            "gender": "여",
258
            "city": "서울"
259
260
          }
261
262
263
     6. Lab
264
       1)In J2EE Perspective
265
       2)Project Explorer > right-click > New > Dynamic Web Project
       3)Project name: RestfulDemo > Next > Check [Generate web.xml deployment
266
       descriptor] > Finish
       4)Convert to Maven Project
267
268
          -project right-click > Configure > Convert to Maven Project > Finish
269
270
       5)Add Spring Project Nature
```

```
271
          -project right-click > Spring Tools > Add Spring Project Nature
272
       6)새로 생성된 pom.xml파일에 필요한 library 추가 > Maven Clean > Maven Install
273
274
          <dependencies>
275
             <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
276
             <dependency>
277
               <groupId>org.springframework</groupId>
278
               <artifactId>spring-context</artifactId>
               <version>4.3.17.RELEASE
279
280
             </dependency>
281
             <!-- https://mvnrepository.com/artifact/junit/junit -->
             <dependency>
282
               <groupId>junit</groupId>
283
284
               <artifactId>junit</artifactId>
               <version>4.12</version>
285
286
               <scope>test</scope>
287
             </dependency>
288
             <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
289
             <dependency>
290
               <groupId>org.springframework</groupId>
               <artifactId>spring-jdbc</artifactId>
291
292
               <version>4.3.17.RELEASE
293
             </dependency>
294
             <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
             <dependency>
295
296
               <groupId>org.springframework</groupId>
               <artifactId>spring-webmvc</artifactId>
297
               <version>4.3.17.RELEASE
298
299
             </dependency>
300
             <dependency>
301
               <groupId>javax.servlet</groupId>
302
               <artifactId>jstl</artifactId>
               <version>1.2</version>
303
             </dependency>
304
             <dependency>
305
306
               <groupId>com.oracle</groupId>
               <artifactId>oidbc8</artifactId>
307
308
               <version>12.2</version>
309
             </dependency>
310
             <!--
             https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
311
             <dependency>
               <groupId>com.fasterxml.jackson.core</groupId>
312
313
               <artifactId>jackson-databind</artifactId>
314
               <version>2.9.0</version>
             </dependency>
315
316
             <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core
             -->
317
             <dependency>
               <groupId>com.fasterxml.jackson.core</groupId>
318
319
               <artifactId>jackson-core</artifactId>
320
               <version>2.9.0</version>
321
             </dependency>
322
             <!--
            https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations
             -->
323
             <dependency>
324
               <groupId>com.fasterxml.jackson.core</groupId>
325
               <artifactId>jackson-annotations</artifactId>
326
               <version>2.9.0</version>
```

```
</dependency>
327
328
            <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
329
            <dependency>
               <groupId>org.mybatis</groupId>
330
               <artifactId>mybatis-spring</artifactId>
331
332
               <version>1.3.2</version>
333
            </dependency>
334
            <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
335
            <dependency>
336
               <groupId>org.mybatis</groupId>
               <artifactId>mybatis</artifactId>
337
               <version>3.4.6</version>
338
339
            </dependency>
340
          </dependencies>
341
342
       7)Build path에 config foler 추가
          -project right-click > Build Path > Configure Build Path > Select [Source] tab
343
344
          -Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
          -Folder name : config > Finish > OK > Apply and Close
345
346
          -Java Resources > config 폴더 확인
347
348
       8)config folder에 applicationContext.xml 파일 생성
349
          -Spring Perspective로 전환
350
          -config right-click > New > Spring Bean Configuration File
351
          -File name: applicationContext.xm
352
          -생성시 beans, context, mvc 체크
          <?xml version="1.0" encoding="UTF-8"?>
353
354
          <beans xmlns="http://www.springframework.org/schema/beans"</pre>
355
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
356
            xmlns:context="http://www.springframework.org/schema/context"
357
            xmlns:mvc="http://www.springframework.org/schema/mvc"
358
            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
359
               http://www.springframework.org/schema/beans/spring-beans.xsd
               http://www.springframework.org/schema/context
360
               http://www.springframework.org/schema/context/spring-context-4.3.xsd">
361
362
          </beans>
363
364
       9)ContextLoaderListener class 설정
365
          -비즈니스 로직용의 스프링 설정 파일 (ex:applicationContext.xml)을 작성했기 때문에 listener로
          ContextLoaderListener 클래스를 정의해야 한다.
366
          -ContextLoaderListener 클래스는 스프링 설정 파일(디폴트에서 파일명
          applicationContext.xml)을 로드하면 ServletContextListener 인터페이스를 구현하고 있기
          때문에 ServletContext 인스턴스 생성시(톰켓으로 어플리케이션이 로드된 때)에 호출된다.
          -즉, ContextLoaderListener 클래스는 DispatcherServlet 클래스의 로드보다 먼저 동작하여
367
          비즈니스 로직층을 정의한 스프링 설정 파일을 로드한다.
368
          -web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener
          -ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입
            <!-- needed for ContextLoaderListener -->
369
370
            <context-param>
371
               <param-name>contextConfigLocation</param-name>
372
               <param-value>location</param-value>
373
            </context-param>
374
375
            <!-- Bootstraps the root web application context before servlet initialization -->
376
            stener>
377
               listener-class>org.springframework.web.context.ContextLoaderListener/listene
```

```
r-class>
378
             </listener>
379
380
          -아래 코드로 변환
381
             <context-param>
382
               <param-name>contextConfigLocation</param-name>
               <param-value>classpath:applicationContext.xml</param-value>
383
384
             </context-param>
385
386
        10)DispatcherServlet Class 추가
387
          -web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet
          -DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.
388
          <!-- The front controller of this Spring Web application, responsible for handling all
389
          application requests -->
390
          <servlet>
391
             <servlet-name>springDispatcherServlet</servlet-name>
392
             <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
393
             <init-param>
394
               <param-name>contextConfigLocation</param-name>
395
               <param-value>location
             </init-param>
396
397
             <load-on-startup>1</load-on-startup>
398
          </servlet>
399
400
          <!-- Map all requests to the DispatcherServlet for handling -->
401
          <servlet-mapping>
402
             <servlet-name>springDispatcherServlet</servlet-name>
             <url-pattern>url</url-pattern>
403
404
          </servlet-mapping>
405
          -아래의 코드로 변환
406
407
          <servlet>
408
             <servlet-name>springDispatcherServlet</servlet-name>
             <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
409
410
             <init-param>
411
               <param-name>contextConfigLocation</param-name>
               <param-value>classpath:applicationContext*.xml</param-value>
412
413
             </init-param>
             <load-on-startup>1</load-on-startup>
414
          </servlet>
415
416
417
          <servlet-mapping>
418
             <servlet-name>springDispatcherServlet</servlet-name>
419
             <url-pattern>/</url-pattern>
420
          </servlet-mapping>
421
        11)MemberVO class 생성
422
423
          -src/com.javasoft.vo package 생성
424
          -src/com.javasoft.vo.MemberVO class 생성
425
426
             package com.javasoft.vo;
427
428
             public class MemberVO {
429
               private String name, userid, gender, city;
430
               public MemberVO() {}
               public MemberVO(String name, String userid, String gender, String city) {
431
                  this.name = name;
432
433
                  this.userid = userid;
434
                  this.gender = gender;
```

```
435
                   this.city = city;
436
                }
437
                public String getName() {
438
                   return name;
439
                }
440
                public void setName(String name) {
                   this.name = name;
441
442
                }
443
                public String getUserid() {
                   return userid;
444
445
446
                public void setUserid(String userid) {
447
                   this.userid = userid;
448
449
                public String getGender() {
450
                   return gender;
451
452
                public void setGender(String gender) {
453
                   this.gender = gender;
454
455
                public String getCity() {
456
                   return city;
457
                }
458
                public void setCity(String city) {
459
                   this.city = city;
460
461
                @Override
462
                public String toString() {
                   return "MemberVO [name=" + name + ", userid=" + userid + ", gender=" +
463
                   gender + ", city=" + city + "]";
464
                }
             }
465
466
467
        12)MemberDao 객체 생성
           -src/com.javasoft.dao package 생성
468
469
           -src/com.javasoft.dao.MemberDao interface
470
             package com.javasoft.dao;
471
472
473
             import java.util.List;
474
475
             import com.javasoft.vo.MemberVO;
476
477
             public interface MemberDao {
478
                void create(MemberVO member);
479
                List<MemberVO> readAll();
480
                MemberVO read(String userid);
                void update(MemberVO member);
481
482
                void delete(String userid);
             }
483
484
485
           -src/com.javasoft.dao.MemberDaoImpl.java 생성
486
487
             package com.javasoft.dao;
488
             import java.util.List;
489
490
491
             import org.springframework.beans.factory.annotation.Autowired;
             import org.springframework.stereotype.Repository;
492
493
             import org.apache.ibatis.session.SqlSession;
```

```
494
495
             import com.javasoft.vo.MemberVO;
496
497
             @Repository("memberDao")
498
             public class MemberDaoImpl implements MemberDao {
499
                @Autowired
500
                private SqlSession sqlSession;
501
502
                @Override
503
                public void create(MemberVO member) {
504
                  this.sqlSession.insert("Member.insert", member);
505
                }
506
507
                @Override
508
                public List<MemberVO> readAll() {
509
                  return this.sqlSession.selectList("Member.select");
510
                }
511
512
                @Override
513
                public MemberVO read(String userid) {
                  return this.sqlSession.selectOne("Member.selectMember", userid);
514
515
                }
516
517
                @Override
518
                public void update(MemberVO member) {
519
                  this.sqlSession.update("Member.update", member);
520
                }
521
522
                @Override
523
                public void delete(String userid) {
524
                  this.sqlSession.delete("Member.delete", userid);
525
                }
526
             }
527
528
529
        13)MemberService 객체 생성
530
          -src/com.javasoft.service package 생성
531
          -src/com.javasoft.service.MemberService interface
532
533
             package com.javasoft.service;
534
535
             import java.util.List;
536
537
             import com.javasoft.vo.MemberVO;
538
539
             public interface MemberService {
540
                void insertMember(MemberVO member);
                List<MemberVO> select();
541
542
                MemberVO selectMember(String userid);
543
                void updateMember(MemberVO member);
                void deleteMember(String userid);
544
545
             }
546
547
          -src/com.javasoft.service.MemberServiceImpl.java
548
549
             package com.javasoft.service;
550
551
             import java.util.List;
552
553
             import org.springframework.beans.factory.annotation.Autowired;
```

```
554
             import org.springframework.stereotype.Service;
555
556
             import com.javasoft.dao.MemberDao;
557
             import com.javasoft.vo.MemberVO;
558
559
             @Service("memberService")
560
             public class MemberServiceImpl implements MemberService {
561
               @Autowired
562
               private MemberDao memberDao;
563
564
               @Override
565
               public void insertMember(MemberVO member) {
566
                  this.memberDao.create(member);
567
568
569
               @Override
               public List<MemberVO> select() {
570
571
                  return this.memberDao.readAll();
572
               }
573
574
               @Override
575
               public MemberVO selectMember(String userid) {
576
                  return this.memberDao.read(userid);
577
               }
578
579
               @Override
               public void updateMember(MemberVO member) {
580
                  this.memberDao.update(member);
581
582
               }
583
584
               @Override
               public void deleteMember(String userid) {
585
586
                  this.memberDao.delete(userid);
587
               }
588
             }
589
        14)HomeController 객체 생성
590
591
          -src/com.javasoft.controller package 생성
592
          -com.javasoft.controller.HomeController class 생성
593
594
             package com.javasoft.controller;
595
596
             import org.springframework.beans.factory.annotation.Autowired;
597
             import org.springframework.stereotype.Controller;
             import org.springframework.ui.Model;
598
             import org.springframework.web.bind.annotation.RequestMapping;
599
600
             import org.springframework.web.bind.annotation.RequestParam;
601
602
             import com.javasoft.service.MemberService;
             import com.javasoft.vo.MemberVO;
603
604
             @Controller
605
606
             public class HomeController {
607
               @Autowired
608
               private MemberService service;
609
610
             }
611
        15)config/dbinfo.properties 파일 생성
612
613
```

```
614
              db.driver=oracle.jdbc.driver.OracleDriver
615
              db.url=jdbc:oracle:thin:@192.168.56.3:1521:ORCL
616
              db.username=scott
617
              db.password=tiger
618
619
          16)applicationContext.xml 수정
620
621
              <context:component-scan base-package="com.javasoft" />
                  <context:property-placeholder location="classpath:dbinfo.properties"/>
622
623
                  <bean id="dataSource"</pre>
                 class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
                     cproperty name="driverClass" value="${db.driver}"/>
624
                     cproperty name="url" value="${db.url}"/>
625
                     contentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentconte
626
627
                     property name="password" value="${db.password}"/>
628
                  </bean>
629
630
                  <!-- <bean id="viewResolver"
                 class="org.springframework.web.servlet.view.InternalResourceViewResolver">
631
632
                     cproperty name="prefix" value="/WEB-INF/views/" />
                     cproperty name="suffix" value=".jsp" />
633
                  </bean> -->
634
635
636
                  <mvc:annotation-driven />
637
                  <mvc:default-servlet-handler/>
638
639
          17)config/mybatis-config.xml
640
641
              <?xml version="1.0" encoding="UTF-8" ?>
642
              <!DOCTYPE configuration
643
                 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
644
                     "http://mybatis.org/dtd/mybatis-3-config.dtd">
645
646
              <configuration>
647
                  <typeAliases>
648
                     <typeAlias type="com.javasoft.vo.MemberVO" alias="memberVO"/>
649
                  </typeAliases>
650
              </configuration>
651
652
          18)config/member-mapper.xml
653
              <?xml version="1.0" encoding="UTF-8" ?>
654
655
              <!DOCTYPE mapper
656
                 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
657
                     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
658
              <mapper namespace="Member">
659
660
              </mapper>
661
662
          19)applicationContext.xml 아래 코드 추가
663
664
              <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
                  property name="dataSource" ref="dataSource" />
665
666
                  configLocation" value="classpath:mybatis-config.xml" />
                  property name="mapperLocations">
667
668
                     t>
669
                         <value>classpath:member-mapper.xml</value>
670
                     </list>
671
                  </property>
672
              </bean>
```

```
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
673
             <constructor-arg ref="sqlSessionFactory" />
674
          </bean>
675
676
677
        20)전체 사용자 조회하기
678
          -HomeController 객체 코드 추가
679
             @RequestMapping(value = "/members", method = RequestMethod.GET)
680
681
             @ResponseBody
682
             public Map members() {
683
               List<MemberVO> list = this.memberService.select();
               Map<String, Object> map = new HashMap<String, Object>();
684
               map.put("code", "success");
685
               map.put("data", list);
686
687
               return map;
688
             }
689
690
          -mybatis-mapper.xml
691
692
             <resultMap type="memberVO" id="selectMap">
                <result property="name" column="name"/>
693
                <result property="userid" column="userid"/>
694
695
                <result property="gender" column="gender" />
696
                <result property="city" column="city"/>
697
             </resultMap>
698
             <select id="select" resultMap="selectMap">
699
               SELECT * FROM Member
700
701
             </select>
702
703
          -MemberDaoImpl.java
704
705
             @Override
706
             public List<MemberVO> readAll() {
707
               return this.sqlSession.selectList("Member.select");
708
             }
709
710
          -MemberServiceImpl.java
711
712
             @Override
713
             public List<MemberVO> select() {
714
               return this.memberDao.readAll();
715
             }
716
717
          -Postman
718
             GET http://localhost:8080/RestfulDemo/members Send
719
             Body
720
721
722
                  "code": "success",
                  "data": [
723
724
                     {
                       "userId": "jimin",\
725
726
                       "name": "한지민",
                       "gender": "여",
727
                       "city": "서울"
728
729
                     },
730
                       "userId": "javasoft",
731
                       "name": "조용필",
732
```

```
733
                     "gender": "남성",
734
                     "city": "부산"
                  },
{
735
736
                     "userId": "javaexpert",
737
                     "name": "이미자",
738
                     "gender": "여성",
739
740
                     "city": "광주"
741
                  }
742
                ]
743
              }
744
745
         -WebContent/index.html
746
747
           <!DOCTYPE html>
748
           <html>
749
              <head>
750
                <meta charset="UTF-8">
751
                <title>Welcome</title>
752
                <script src="js/jquery-1.12.4.js"></script>
753
                <script>
754
                   $(document).ready(function(){
755
                     $.ajax({
                       url:"/RestfulDemo/members",
756
757
                       type: "GET",
758
                       dataType: "json",
759
                       success : function(data){
760
                         var str = "";
761
                         var members = data.data;
762
                         for(var i = 0; i < members.length; i++){
763
                            str += "";
764
                            var userid = members[i].userid;
765
                            str += "<a href='view.html?userid=" + userid + "'>" +
                            userid + "</a>" +
766
                                "" + members[i].name + "" +
767
                                "" + members[i].gender + "" +
                                "" + members[i].city + "";
768
769
                            str += "";
770
771
                         $("#result").html(str);
772
773
                     });
774
                  });
775
                </script>
776
              </head>
777
              <body>
778
                <h1>Member List</h1>
779
                <div style="text-align:center">
780
                   <a href="register.html">Member Add</a>
781
                </div>
782
                783
                   <thead>
784
                     785
                       >아이디이름
786
                       성별거주지
787
                     788
                   </thead>
789
                   790
                   791
```

```
792
               </body>
793
             </html>
794
795
        21)특정 사용자 조회하기
796
          -HomeController.java
797
798
             @RequestMapping(value = "/members/{userid}", method = RequestMethod.GET)
799
             @ResponseBody
             public Map memberInfo(@PathVariable String userid) {
800
801
               //System.out.println("userid = " + userid);
               MemberVO member = this.memberService.selectMember(userid);
802
               Map<String, Object> map = new HashMap<String, Object>();
803
               map.put("code", "success");
804
               map.put("data", member);
805
806
               return map;
             }
807
808
809
          -mybatis-mapper.xml
810
             <select id="selectMember" parameterType="String" resultType="memberVO">
811
               SELECT * FROM Member WHERE userid = #{userid}
812
813
             </select>
814
815
          -MemberDaoImpl.java
816
817
             @Override
             public MemberVO read(String userid) {
818
               return this.sqlSession.selectOne("Member.selectMember", userid);
819
820
821
822
          -MemberServiceImpl.java
823
824
             @Override
825
             public MemberVO selectMember(String userid) {
826
               return this.memberDao.read(userid);
             }
827
828
829
          -Postman
830
             GET http://localhost:8080/RestfulDemo/members/javasoft Send
831
             Body
832
833
                  "code": "success",
834
835
                  "data": {
                    "userId": "javasoft",
836
                    "name": "조용필",
837
                    "aender": "남성",
838
                    "city": "부산"
839
840
                  }
841
               }
842
843
          -WebContent/view.html
844
             <!DOCTYPE html>
845
846
             <html>
847
               <head>
848
                  <meta charset="UTF-8">
                  <title>회원 정보 페이지</title>
849
850
                  <script src="js/jquery-1.12.4.js"></script>
851
                  <script>
```

```
852
                    var userid = null;
853
854
                    $(function(){
855
                      userid = location.search.substring(1).split("=")[1];
856
                      $.ajax({
857
                         url : "/RestfulDemo/members/" + userid,
858
                         type: "GET",
859
                         success : function(data){
860
                           var member = data.data;
                           $("#userid").text(member.userid);
861
                           $("#name").text(member.name);
862
                           $("#gender").text(member.gender);
863
864
                           $("#city").text(member.city);
865
866
                      });
867
                    });
868
                 </script>
869
               </head>
870
               <body>
871
                 <h1>Member Information</h1>
872
                 <l
873
                    아이디 : <span id="userid"></span>
874
                    이름: <span id="name"></span>
875
                    성별: <span id="gender"></span>
876
                    거주지: <span id="city"></span>
877
                 <a href = "javascript:void(0)"
878
                 onclick="javascript:history.back();">목록으로</a>
879
               </body>
            </html>
880
881
882
883
       22)사용자 등록 구현하기
884
          -HomeController.java
885
886
            @RequestMapping(value = "/members", method = RequestMethod.POST)
887
            @ResponseBody
            public Map insert(@RequestBody MemberVO memberVO) {
888
889
               System.out.println(memberVO);
890
               this.memberService.insertMember(memberVO);
               Map<String, Object> map = new HashMap<String, Object>();
891
               map.put("code", "success");
892
893
               return map;
894
            }
895
896
          -mabatis-mapper.xml
897
            <insert id="insert" parameterType="memberVO">
898
899
               INSERT INTO Member(name, userid, gender, city)
900
               VALUES(#{name}, #{userid}, #{gender}, #{city})
            </insert>
901
902
903
          -MemberDaoImpl.java
904
905
            @Override
906
            public void create(MemberVO member) {
               this.sqlSession.insert("Member.insert", member);
907
908
909
910
          -MemberServiceImpl.java
```

```
911
912
             @Override
913
             public void insertMember(MemberVO member) {
914
                this.memberDao.create(member);
915
916
917
           -Postman
918
             POST <a href="http://localhost:8080/RestfulDemo/members">http://localhost:8080/RestfulDemo/members</a>
919
             Body
920
                raw
921
                {
922
                   "userId": "girlsage",
923
924
                   "name": "소녀시대",
925
                   "gender" : "여성",
                   "city": "수원"
926
927
                }
928
929
             Send 버튼 클릭하면
930
931
                Body
932
                   {"code": "success"}
933
934
           -WebContent/register.html
935
936
              <!DOCTYPE html>
937
              <html>
938
                <head>
                   <meta charset="UTF-8">
939
                   <title>Member Add</title>
940
941
                   <script src="js/jquery-1.12.4.js"></script>
                   <script>
942
                      $(function(){
943
                        $("input[type='button']").bind("click", function(){
944
                           $.ajax({
945
946
                              url: "/RestfulDemo/members",
947
                              contentType: "application/json; charset=utf-8",
948
                              type: "POST",
949
                              data : JSON.stringify({
950
                                 "userid": $("#userid").val(),
951
                                "name": $("#name").val(),
                                 "gender": $(".gender:checked").val(),
952
                                 "city": $("#city").val()
953
954
                              }),
955
                              dataType: "json",
956
                              success : function(data){
957
                                 alert(data.code);
                                location.href = "/0605/";
958
959
                              }
960
                           });
                        });
961
962
                      });
963
                   </script>
964
                </head>
965
                <body>
966
                   <h1>Member Add</h1>
967
                   Name : <input type="text" id="name" />
968
                      ID : <input type="text" id="userid" />
969
970
                      Gender:
```

```
971
                        <input type="radio" class="gender" name="gender"
                        value="남성">남성  
                        <input type="radio" class="gender" name="gender" value="여성">여성
 972
 973
 974
                      City: <input type="text" id="city" />
 975
 976
                   <input type="button" value="가입하기" />
 977
                </body>
 978
              </html>
 979
 980
         23)사용자 정보 수정 구현하기
 981
           -HomeController.java
 982
              @RequestMapping(value = "/members", method = RequestMethod.PUT)
 983
 984
              @ResponseBody
 985
              public Map update(@RequestBody MemberVO memberVO) {
                this.memberService.updateMember(memberVO);
 986
 987
                Map<String, Object> map = new HashMap<String, Object>();
                map.put("code", "success");
 988
                return map;
 989
 990
              }
 991
 992
           -mabatis-mapper.xml
 993
 994
              <update id="update" parameterType="memberVO">
 995
                UPDATE Member SET name = #{name}, gender = #{gender}, city = #{city}
 996
                WHERE userid = #{userid}
 997
              </update>
 998
 999
           -MemberDaoImpl.java
1000
              @Override
1001
              public void update(MemberVO member) {
1002
1003
                this.sqlSession.update("Member.update", member);
1004
1005
1006
           -MemberServiceImpl.java
1007
1008
              @Override
              public void updateMember(MemberVO member) {
1009
                this.memberDao.update(member);
1010
1011
              }
1012
1013
           -Postman
1014
              PUT <a href="http://localhost:8080/RestfulDemo/members">http://localhost:8080/RestfulDemo/members</a>
1015
              Body
1016
                raw
1017
1018
                   "userId": "girlsage",
1019
                   "name": "소년시대",
1020
                   "gender": "남성",
1021
1022
                   "city": "부산"
1023
                }
1024
              Send 버튼 클릭하면
1025
1026
1027
                Body
                   {"code": "success"}
1028
1029
```

```
1030
           -WebContent/view.html 수정
1031
              --아래 코드를 추가한다.
1032
                 <a href = "javascript:void(0)"
                onclick="javascript:member_update()">수정하기</a>
1033
1034
                var flag = false;
                function member_update(){
1035
                   if(!flag) { //수정하기를 클릭하면
1036
1037
                      var name = $("#name").text();
1038
                      $("span#name")
                      .replaceWith("<input type='text' id='name' value='" + name + "'/>");
1039
                      var gender = $("#gender").text();
1040
1041
                      var str = null;
                      if(gender == "남성"){
1042
1043
                        str = "<input type='radio' class='gender' value='남성'
                        checked/>남성  " +
                          "<input type='radio' class='gender' value='여성' />여성";
1044
1045
                      }else{
                        str = "<input type='radio' class='gender' name='gender' value='남성'
1046
                        />남성  " +
1047
                          "<input type='radio' class='gender' name='gender' value='여성'
                          checked />여성";
                      }
1048
1049
                      $("span#gender").replaceWith(str);
1050
                      var city = $("#city").text();
1051
                      $("span#city")
1052
                      .replaceWith("<input type='text' id='city' value='" + city + "'/>");
1053
                      flag = true;
1054
                   }else{
1055
                      $.ajax({
1056
                        url: "/RestfulDemo/members",
                        type: "PUT",
1057
1058
                        data : JSON.stringify({
1059
                           "userid": userid,
                           "name": $("#name").val(),
1060
                           "gender": $(".gender:checked").val(),
1061
                           "city": $("#city").val()
1062
1063
                        }),
1064
                        contentType : "application/json;charset=utf-8",
                        success : function(data){
1065
1066
                           alert(data.code);
1067
                           location.reload();
1068
                        }
                      });
1069
1070
                      flag = false;
1071
                   }
1072
                }
1073
1074
         24)사용자 정보 삭제 구현하기
1075
1076
           -HomeController.java
1077
1078
              @RequestMapping(value = "/members/{userid}", method =
              RequestMethod.DELETE)
1079
              @ResponseBody
              public Map delete(@PathVariable String userid) {
1080
1081
                this.memberService.deleteMember(userid);
                Map<String, Object> map = new HashMap<String, Object>();
1082
                map.put("code", "success");
1083
                return map;
1084
```

```
1085
              }
1086
1087
            -mabatis-mapper.xml
1088
              <delete id="delete" parameterType="String">
1089
                 DELETE FROM Member WHERE userid = #{userid}
1090
1091
              </delete>
1092
1093
            -MemberDaoImpl.java
1094
1095
              @Override
              public void delete(String userid) {
1096
1097
                 this.sqlSession.delete("Member.delete", userid);
1098
1099
1100
            -MemberServiceImpl.java
1101
1102
              @Override
1103
              public void deleteMember(String userid) {
1104
                 this.memberDao.delete(userid);
1105
              }
1106
1107
            -Postman
1108
              DELETE <a href="http://localhost:8080/RestfulDemo/members/girlsage">http://localhost:8080/RestfulDemo/members/girlsage</a>
1109
1110
              Send 버튼 클릭하면
1111
1112
                 Body
                    {"code": "success"}
1113
1114
1115
            -WebContent/view.html 수정
1116
              --아래의 코드를 추가한다.
1117
1118
                 <a href = "javascript:void(0)"
                 onclick="javascript:member_delete()">삭제하기</a>
1119
1120
                 function member_delete(){
1121
                    $.ajax({
1122
                      url: "/0605/members/" + userid,
1123
                      type: "DELETE",
1124
                      success : function(data){
1125
                         alert(data.code);
1126
                         location.href = "/0605/";
1127
                      }
1128
                    });
                 }
1129
1130
1131
      20. 데이터 변환 - XML로 변환
1132
         1)Maven Repository에서 'spring xml'로 검색
         2)Spring Object/XML Marshalling에서 4.3.13.RELEASE 선택
1133
1134
         3)pom.xml에 아래 dependency 추가 > Maven Clean > Mavan Install
1135
1136
            <dependency>
1137
               <groupId>org.springframework</groupId>
1138
               <artifactId>spring-oxm</artifactId>842 <version>4.3.13.RELEASE</version>
            </dependency>
1139
1140
1141
         4)Maven Repository에서 'jaxb'로 검색, Jaxb Api에서 2.3.0
         5)아래의 dependency를 pom.xml에 추가 > Maven Clean > Mavan Install
1142
1143
```

```
<dependency>
1144
              <groupId>javax.xml.bind</groupId>
1145
1146
              <artifactId>jaxb-api</artifactId>
              <version>2.3.0</version>
1147
1148
           </dependency>
1149
         6)src/com.javasoft.vo/UserListVO.java 생성
1150
1151
1152
           package com.javasoft.vo;
1153
1154
           import java.util.List;
1155
1156
           import javax.xml.bind.annotation.XmlAccessType;
           import javax.xml.bind.annotation.XmlAccessorType;
1157
1158
           import javax.xml.bind.annotation.XmlElement;
1159
           import javax.xml.bind.annotation.XmlRootElement;
1160
1161
           import org.springframework.stereotype.Component;
1162
           @XmlRootElement(name="userList")
1163
1164
           @XmlAccessorType(XmlAccessType.FIELD)
           @Component
1165
           public class UserListVO {
1166
              @XmlElement(name="user")
1167
              private List<UserVO> userList;
1168
1169
1170
              public List<UserVO> getUserList() {
1171
                return userList;
1172
              }
1173
1174
              public void setUserList(List<UserVO> userList) {
1175
                this.userList = userList;
1176
              }
           }
1177
1178
1179
           -XML 문서는 반드시 단 하나의 root element를 가져야 한다.
           -여러 UserVO를 표현하려면 root element로 사용할 또 다른 Java class가 필요하다.
1180
           -새로 생성한 UserListVO객체는 이 객체가 root element에 해당하는 객체이며, root element
1181
           이름을 userList로 설정하겠다는 의미로 @XmlRootElement(name="userList") 설정을 추가했다.
           -그리고 userList 변수 위에도 @XmlElement(name="user")를 추가했는데, UserVO 마다
1182
           element 이름을 user로 변경할 것이다.
1183
1184
         7)src/com.javasoft.vo.MemberVO.java 수정
1185
1186
           package com.javasoft.vo;
1187
1188
           import javax.xml.bind.annotation.XmlAccessType;
           mport javax.xml.bind.annotation.XmlAccessorType;
1189
1190
           import javax.xml.bind.annotation.XmlAttribute;
           import javax.xml.bind.annotation.XmlRootElement;
1191
1192
           import org.springframework.stereotype.Component;
1193
1194
1195
           @XmlRootElement(name="user")
1196
           @XmlAccessorType(XmlAccessType.FIELD)
           @Component
1197
1198
           public class UserVO {
1199
              @XmlAttribute
1200
              private String userId;
1201
```

```
-VO class에 선언된 @XmlAccessorType은 UserVO 겍체를 XML로 변환할 수 있다는 의미이다.907
1202
          -그리고 XmlAccessType.FIELD 때문에 이 객체가 가진 필드, 즉 변수들은 자동으로 자식 element로
           표현된다.
          -하지만, 이 중에서 userId에만 @XmlAttribute가 붙었는데, 이는 userId를 속성으로 표현하라는
1203
          의미이다.
          -만일 JSON 변환시 @JsonIgnore가 변환시 제외하는 것처럼, XML변환시에도 제외할 변수는
1204
          @XmlTransient를 붙이면 된다.
          -마지막으로 변환시 변수가 참조형이면 반드시 기본 생성자가 있어야만 한다.
1205
1206
        8)Spring 설정 파일에서 p와 oxm 체크후, 아래 코드 추가
1207
1208
           -JSON 변환시 Java 객체를 JSON response body로 변환해주는
          MappingJackson2HttpMessageConverter를 Spring 설정파일에 추가해야 하는데, 설정 파일에
           <mvc:annotation-driven />으로 대체했었다.
          -마찬가지로 Java 객체를 XML response body로 변환할 때는 아래의 코드를 추가한다.
1209
1210
1211
           <bean id="xmlViewer"</pre>
          class="org.springframework.web.servlet.view.xml.MarshallingView">
1212
             <constructor-arg>
1213
               <bean class="org.springframework.oxm.jaxb.Jaxb2Marshaller"</pre>
               p:classesToBeBound="com.javasoft.vo.UserListVO"/>
1214
             </constructor-arg>
1215
           </bean>
1216
1217
        9)UserController.java 코드 추가
1218
1219
           @RequestMapping(value="/userlist.do", produces="application/xml")
1220
           @ResponseBody
          public UserListVO userlist(){
1221
             UserListVO listVO = new UserListVO();
1222
1223
             listVO.setUserList(this.userService.getUserList());
1224
             return listVO;
1225
          }
1226
1227
        10)실행결과
1228
           <userList>
             <user userId="jimin">
1229
1230
               <name>하지민</name>
1231
               <gender> 여</gender>
1232
               <city>서울</city>
1233
             </user>
1234
           </userList>
```