

```
1  HOL : Restful API in Spring
2  -----
3  Task1. Using Restful API
4  1. In J2EE Perspective
5  2. Project Explorer > right-click > New > Dynamic Web Project
6  3. Project name : RestfulDemo > Next > Check [Generate web.xml deployment descriptor] > Finish
7  4. Convert to Maven Project
8     project right-click > Configure > Convert to Maven Project > Finish
9
10 5. Add Spring Project Nature
11    project right-click > Spring > Add Spring Project Nature
12
13 6. 새로 생성된 pom.xml file에 필요한 library 추가 > Maven Clean > Maven Install
14    <dependencies>
15      <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
16      <dependency>
17        <groupId>org.springframework</groupId>
18        <artifactId>spring-context</artifactId>
19        <version>5.2.0.RELEASE</version>
20      </dependency>
21      <!-- https://mvnrepository.com/artifact/junit/junit -->
22      <dependency>
23        <groupId>junit</groupId>
24        <artifactId>junit</artifactId>
25        <version>4.12</version>
26        <scope>test</scope>
27      </dependency>
28      <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
29      <dependency>
30        <groupId>org.springframework</groupId>
31        <artifactId>spring-jdbc</artifactId>
32        <version>5.2.0.RELEASE</version>
33      </dependency>
34      <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
35      <dependency>
36        <groupId>org.springframework</groupId>
37        <artifactId>spring-webmvc</artifactId>
38        <version>5.2.0.RELEASE</version>
39      </dependency>
40      <dependency>
41        <groupId>javax.servlet</groupId>
42        <artifactId>jstl</artifactId>
43        <version>1.2</version>
44      </dependency>
45      <dependency>
46        <groupId>com.oracle</groupId>
47        <artifactId>ojdbc6</artifactId>
48        <version>11.2</version>
49      </dependency>
50      <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
51      <dependency>
52        <groupId>com.fasterxml.jackson.core</groupId>
```

```

53     <artifactId>jackson-databind</artifactId>
54     <version>2.10.0</version>
55 </dependency>
56 <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
57 <dependency>
58     <groupId>com.fasterxml.jackson.core</groupId>
59     <artifactId>jackson-core</artifactId>
60     <version>2.10.0</version>
61 </dependency>
62 <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
63 <dependency>
64     <groupId>com.fasterxml.jackson.core</groupId>
65     <artifactId>jackson-annotations</artifactId>
66     <version>2.10.0</version>
67 </dependency>
68 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
69 <dependency>
70     <groupId>org.mybatis</groupId>
71     <artifactId>mybatis-spring</artifactId>
72     <version>2.0.3</version>
73 </dependency>
74 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
75 <dependency>
76     <groupId>org.mybatis</groupId>
77     <artifactId>mybatis</artifactId>
78     <version>3.5.3</version>
79 </dependency>
80 </dependencies>

```

82 1)pom.xml > right-click > Run As > Maven install
83 [INFO] BUILD SUCCESS

86 7. Build path에 config Foler 추가

- 87 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
- 88 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
- 89 3)Folder name : config > Finish > OK > Apply and Close
- 90 4)Java Resources > config folder 확인

93 8. config Folder에 applicationContext.xml file 생성

- 94 1)config Folder > right-click > New > Other > Spring > Spring Bean Configuration File
- 95 2)File name : applicationContext.xml
- 96 3)생성시 beans,context, mvc check

```

97 <?xml version="1.0" encoding="UTF-8"?>
98 <beans xmlns="http://www.springframework.org/schema/beans"
99     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
100     xmlns:context="http://www.springframework.org/schema/context"
101     xmlns:mvc="http://www.springframework.org/schema/mvc"
102     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

```

```
104 http://www.springframework.org/schema/beans/spring-beans.xsd  
105 http://www.springframework.org/schema/context  
106 http://www.springframework.org/schema/context/spring-context-4.3.xsd">  
107  
108  
109 9. ContextLoaderListener class 설정  
110 1)Business logic의 Spring 설정 file (ex:applicationContext.xml)을 작성했기 때문에 listener로  
111 ContextLoaderListener class를 정의해야 한다.  
112 2)ContextLoaderListener class는 Spring 설정 file(default에서 file명 applicationContext.xml)을 load하면  
113 ServletContextListener interface를 구현하고 있기 때문에 ServletContext instance 생성시(Tomcat으로  
114 application이 load된 때)에 호출된다.  
115 3)즉, ContextLoaderListener class는 DispatcherServlet class의 load보다 먼저 동작하여 business logic층을  
116 정의한 Spring 설정 file을 load한다.  
117 4)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener  
118 -ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입  
119  
120 <!-- needed for ContextLoaderListener -->  
121 <context-param>  
122 <param-name>contextConfigLocation</param-name>  
123 <param-value>location</param-value>  
124 </context-param>  
125  
126 <!-- Bootstraps the root web application context before servlet initialization -->  
127 <listener>  
128 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
129 </listener>  
130  
131 5)아래 code로 변환  
132 <context-param>  
133 <param-name>contextConfigLocation</param-name>  
134 <param-value>classpath:applicationContext.xml</param-value>  
135 </context-param>  
136  
137 10. DispatcherServlet Class 추가  
138 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet -DispatcherServlet  
139 declaration] 선택하면 아래의 code가 자동 추가된다.  
140  
141 <!-- The front controller of this Spring Web application, responsible for handling all application  
142 requests -->  
143 <servlet>  
144 <servlet-name>springDispatcherServlet</servlet-name>  
145 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
146 <init-param>  
147 <param-name>contextConfigLocation</param-name>  
148 <param-value>location</param-value>  
149 </init-param>  
150 <load-on-startup>1</load-on-startup>  
151 </servlet>  
152  
153 <!-- Map all requests to the DispatcherServlet for handling -->
```

```
147     <servlet-mapping>
148         <servlet-name>springDispatcherServlet</servlet-name>
149         <url-pattern>url</url-pattern>
150     </servlet-mapping>
151
152 2)아래의 code로 변환
153     <servlet>
154         <servlet-name>springDispatcherServlet</servlet-name>
155         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
156         <init-param>
157             <param-name>contextConfigLocation</param-name>
158             <param-value>classpath:beans.xml</param-value>
159         </init-param>
160         <load-on-startup>1</load-on-startup>
161     </servlet>
162
163     <servlet-mapping>
164         <servlet-name>springDispatcherServlet</servlet-name>
165         <url-pattern>/</url-pattern>
166     </servlet-mapping>
```

11. MemberVO class 생성

1)src/com.example.vo package 생성

2)src/com.example.vo.MemberVO class 생성

```
172
173 package com.example.vo;
174
175 public class MemberVO {
176     private String name, userid, gender, city;
177     public MemberVO() {}
178     public MemberVO(String name, String userid, String gender, String city) {
179         this.name = name;
180         this.userid = userid;
181         this.gender = gender;
182         this.city = city;
183     }
184     public String getName() {
185         return name;
186     }
187     public void setName(String name) {
188         this.name = name;
189     }
190     public String getUserid() {
191         return userid;
192     }
193     public void setUserid(String userid) {
194         this.userid = userid;
195     }
196     public String getGender() {
197         return gender;
198     }
```

```
199     public void setGender(String gender) {
200         this.gender = gender;
201     }
202     public String getCity() {
203         return city;
204     }
205     public void setCity(String city) {
206         this.city = city;
207     }
208     @Override
209     public String toString() {
210         return "MemberVO [name=" + name + ", userid=" + userid + ", gender=" + gender + ", city="
211             + city + "]\n";
212     }
213 }
214
```

12. MemberDao 객체 생성

- 1)src/com.example.dao package 생성
- 2)src/com.example.dao.MemberDao interface

```
218
219     package com.example.dao;
220
221     import java.util.List;
222
223     import com.example.vo.MemberVO;
224
225     public interface MemberDao {
226         void create(MemberVO member);
227         List<MemberVO> readAll();
228         MemberVO read(String userid);
229         void update(MemberVO member);
230         void delete(String userid);
231     }
232
```

3)src/com.example.dao.MemberDaoImpl.java 생성

```
234
235     package com.example.dao;
236
237     import java.util.List;
238
239     import org.springframework.beans.factory.annotation.Autowired;
240     import org.springframework.stereotype.Repository;
241     import org.apache.ibatis.session.SqlSession;
242
243     import com.example.vo.MemberVO;
244
245     @Repository("memberDao")
246     public class MemberDaoImpl implements MemberDao {
247         @Autowired
248         private SqlSession sqlSession;
249
```

```
250     @Override
251     public void create(MemberVO member) {
252
253     }
254
255     @Override
256     public List<MemberVO> readAll() {
257         return null;
258     }
259
260     @Override
261     public MemberVO read(String userid) {
262         return null;
263     }
264
265     @Override
266     public void update(MemberVO member) {
267
268     }
269
270     @Override
271     public void delete(String userid) {
272
273     }
274 }
275
276
277 13. MemberService 객체 생성
278 1)src/com.example.service package 생성
279 2)src/com.example.service.MemberService interface
280
281     package com.example.service;
282
283     import java.util.List;
284
285     import com.example.vo.MemberVO;
286
287     public interface MemberService {
288         void insertMember(MemberVO member);
289         List<MemberVO> select();
290         MemberVO selectMember(String userid);
291         void updateMember(MemberVO member);
292         void deleteMember(String userid);
293     }
294
295 3)src/com.example.service.MemberServiceImpl.java
296
297     package com.example.service;
298
299     import java.util.List;
300
301     import org.springframework.beans.factory.annotation.Autowired;
```

```
302 import org.springframework.stereotype.Service;
303
304 import com.example.dao.MemberDao;
305 import com.example.vo.MemberVO;
306
307 @Service("memberService")
308 public class MemberServiceImpl implements MemberService {
309     @Autowired
310     private MemberDao memberDao;
311
312     @Override
313     public void insertMember(MemberVO member) {
314
315     }
316
317     @Override
318     public List<MemberVO> select() {
319         return null;
320     }
321
322     @Override
323     public MemberVO selectMember(String userid) {
324         return null;
325     }
326
327     @Override
328     public void updateMember(MemberVO member) {
329
330     }
331
332     @Override
333     public void deleteMember(String userid) {
334
335     }
336 }
```

339 14. HomeController 객체 생성

```
340 1)src/com.example.controller package 생성
341 2)com.example.controller.HomeController class 생성
342
343 package com.example.controller;
344
345 import org.springframework.beans.factory.annotation.Autowired;
346 import org.springframework.stereotype.Controller;
347 import org.springframework.ui.Model;
348 import org.springframework.web.bind.annotation.RequestMapping;
349 import org.springframework.web.bind.annotation.RequestParam;
350
351 import com.example.service.MemberService;
352 import com.example.vo.MemberVO;
353
```

```
354     @Controller
355     public class HomeController {
356         @Autowired
357         private MemberService service;
358
359     }
360
361
```

362 15. config/dbinfo.properties file 생성

```
363
364     db.driver=oracle.jdbc.driver.OracleDriver
365     db.url=jdbc:oracle:thin:@localhost:1521:XE
366     db.username=hr
367     db.password=hr
368
369
```

370 16. applicationContext.xml

```
371     <context:property-placeholder location="classpath:dbinfo.properties"/>
372     <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
373         <property name="driverClass" value="${db.driver}"/>
374         <property name="url" value="${db.url}"/>
375         <property name="username" value="${db.username}"/>
376         <property name="password" value="${db.password}"/>
377     </bean>
378
379
```

380 17. config > right-click > New > Other > Spring > Spring Bean configuration File >

- 381 1)File name : beans.xml
- 382 2)mvc, context check

```
383
384     <context:component-scan base-package="com.example" />
385     <mvc:annotation-driven />
386     <mvc:default-servlet-handler/>
387
388
```

389 18. config/mybatis-config.xml

```
390
391     <?xml version="1.0" encoding="UTF-8" ?>
392     <!DOCTYPE configuration
393         PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
394         "http://mybatis.org/dtd/mybatis-3-config.dtd">
395
396     <configuration>
397         <typeAliases>
398             <typeAlias type="com.example.vo.MemberVO" alias="memberVO"/>
399         </typeAliases>
400     </configuration>
401
402
```

403 19. config/member-mapper.xml

```
404
405     <?xml version="1.0" encoding="UTF-8" ?>
```



```
406 <!DOCTYPE mapper
407     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
408     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
409 <mapper namespace="Member">
410
411 </mapper>
412
413
414 20. applicationContext.xml 아래 code 추가
415
416 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
417     <property name="dataSource" ref="dataSource" />
418     <property name="configLocation" value="classpath:mybatis-config.xml" />
419     <property name="mapperLocations">
420         <list>
421             <value>classpath:member-mapper.xml</value>
422         </list>
423     </property>
424 </bean>
425 <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
426     <constructor-arg ref="sqlSessionFactory" />
427 </bean>
```

428
429
430 21. 전체 사용자 조회하기

```
431 1)HomeController 객체 code 추가
432
433 @RequestMapping(value = "/members", method = RequestMethod.GET)
434 @ResponseBody
435 public Map members() {
436     List<MemberVO> list = this.service.select();
437     Map<String, Object> map = new HashMap<String, Object>();
438     map.put("code", "success");
439     map.put("data", list);
440     return map;
441 }
```

442
443 2)mybatis-mapper.xml

```
444
445 <resultMap type="memberVO" id="selectMap">
446     <result property="name" column="name"/>
447     <result property="userid" column="userid"/>
448     <result property="gender" column="gender" />
449     <result property="city" column="city"/>
450 </resultMap>
451
452 <select id="select" resultMap="selectMap">
453     SELECT * FROM Member
454 </select>
```

455
456 3)MemberDaoImpl.java

457

```
458     @Override
459     public List<MemberVO> readAll() {
460         return this.sqlSession.selectList("Member.select");
461     }
462
463 4)MemberServiceImpl.java
464
465     @Override
466     public List<MemberVO> select() {
467         return this.memberDao.readAll();
468     }
469
470 5)Postman
471 GET http://localhost:8080/RestfulDemo/members Send
472 Body
473
474 {
475     "code": "success",
476     "data": [
477         {
478             "userId": "jimin",
479             "name": "한지민",
480             "gender": "여",
481             "city": "서울"
482         },
483         {
484             "userId": "example",
485             "name": "조용필",
486             "gender": "남성",
487             "city": "부산"
488         },
489         {
490             "userId": "javaexpert",
491             "name": "이미자",
492             "gender": "여성",
493             "city": "광주"
494         }
495     ]
496 }
497
498 6)WebContent > right-click > New > Folder > js
499   -js/jquery-1.12.4.js
500
501 7)WebContent/index.html
502
503 <!DOCTYPE html>
504 <html>
505     <head>
506         <meta charset="UTF-8">
507         <title>Welcome</title>
508         <script src="js/jquery-1.12.4.js"> </script>
509         <script>
```

```

510     $(document).ready(function(){
511         $.ajax({
512             url:"/RestfulDemo/members",
513             type : "GET",
514             dataType : "json",
515             success : function(data){
516                 var str = "";
517                 var members = data.data;
518                 for(var i = 0 ; i < members.length ; i++){
519                     str += "<tr>";
520                     var userid = members[i].userid;
521                     str += "<td> <a href='view.html?userid=" + userid + "'> " + userid + "</a></td>" +
522                         "<td>" + members[i].name + "</td>" +
523                         "<td>" + members[i].gender + "</td>" +
524                         "<td>" + members[i].city + "</td>";
525                     str += "</tr>";
526                 }
527                 $("#result").html(str);
528             }
529         });
530     });
531 </script>
532 </head>
533 <body>
534     <h1>Member List</h1>
535     <div style="text-align:center">
536         <a href="register.html">Member Add</a>
537     </div>
538     <table border="1">
539         <thead>
540             <tr>
541                 <th>아이디 </th> <th>이름 </th>
542                 <th>성별 </th> <th>거주지 </th>
543             </tr>
544         </thead>
545         <tbody id="result">
546         </tbody>
547     </table>
548 </body>
549 </html>

```

8)index.html > right-click > Run As > Run on Server

22. 특정 사용자 조회하기

1)HomeController.java

```

557 @RequestMapping(value = "/members/{userid}", method = RequestMethod.GET)
558 @ResponseBody
559 public Map memberInfo(@PathVariable String userid) {
560     //System.out.println("userid = " + userid);
561     MemberVO member = this.service.selectMember(userid);

```

```
562     Map<String, Object> map = new HashMap<String, Object>();
563     map.put("code", "success");
564     map.put("data", member);
565     return map;
566 }
567
568 2)mybatis-mapper.xml
569
570 <select id="selectMember" parameterType="String" resultType="memberVO">
571     SELECT * FROM Member WHERE userid = #{userid}
572 </select>
573
574 3)MemberDaoImpl.java
575
576 @Override
577 public MemberVO read(String userid) {
578     return this.sqlSession.selectOne("Member.selectMember", userid);
579 }
580
581 4)MemberServiceImpl.java
582
583 @Override
584 public MemberVO selectMember(String userid) {
585     return this.memberDao.read(userid);
586 }
587
588 5)Postman
589 GET http://localhost:8080/RestfulDemo/members/jimin Send
590 Body
591
592 {
593     "code": "success",
594     "data": {
595         "userId": "jimin",
596         "name": "한지민",
597         "gender": "여",
598         "city": "서울"
599     }
600 }
601
602 6)WebContent/view.html
603
604 <!DOCTYPE html>
605 <html>
606     <head>
607         <meta charset="UTF-8">
608         <title>회원 정보 페이지</title>
609         <script src="js/jquery-1.12.4.js"></script>
610         <script>
611             var userid = null;
612
613             $(function(){
```

```

614         userid = location.search.substring(1).split("=")[1];
615         $.ajax({
616             url : "/RestfulDemo/members/" + userid,
617             type : "GET",
618             success : function(data){
619                 var member = data.data;
620                 $("#userid").text(member.userid);
621                 $("#name").text(member.name);
622                 $("#gender").text(member.gender);
623                 $("#city").text(member.city);
624             }
625         });
626     });
627 </script>
628 </head>
629 <body>
630     <h1>Member Information</h1>
631     <ul>
632         <li>아이디 : <span id="userid"></span></li>
633         <li>이름 : <span id="name"></span></li>
634         <li>성별 : <span id="gender"></span></li>
635         <li>거주지 : <span id="city"></span></li>
636     </ul>
637     <a href = "javascript:void(0)" onclick="javascript:history.back();">목록으로</a>
638 </body>
639 </html>

```

640
641 7)view.html > right-click > Run As > Run on Server
642 8)view.html?userid=jimin or index.html에서 jimin link click

643 644 645 23. 사용자 등록 구현하기

646 1)HomeController.java

```

647
648 @RequestMapping(value = "/members", method = RequestMethod.POST)
649 @ResponseBody
650 public Map insert(@RequestBody MemberVO memberVO) {
651     System.out.println(memberVO);
652     this.service.insertMember(memberVO);
653     Map<String, Object> map = new HashMap<String, Object>();
654     map.put("code", "success");
655     return map;
656 }

```

657 658 2)mabatis-mapper.xml

```

659
660 <insert id="insert" parameterType="memberVO">
661     INSERT INTO Member(name,userid,gender,city)
662     VALUES(#{name}, #{userid}, #{gender}, #{city})
663 </insert>

```

664 665 3)MemberDaoImpl.java

```
666
667     @Override
668     public void create(MemberVO member) {
669         this.sqlSession.insert("Member.insert", member);
670     }
671
672 4)MemberServiceImpl.java
673
674     @Override
675     public void insertMember(MemberVO member) {
676         this.memberDao.create(member);
677     }
678
679 5)web.xml에 한글이 post방식으로 처리할 때 깨짐 방지 하기 위한 fileter 복사해서 넣기
680
681     <filter>
682         <filter-name>encodingFilter</filter-name>
683         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
684         <init-param>
685             <param-name>encoding</param-name>
686             <param-value>UTF-8</param-value>
687         </init-param>
688     </filter>
689     <filter-mapping>
690         <filter-name>encodingFilter</filter-name>
691         <url-pattern>/*</url-pattern>
692     </filter-mapping>
693
694 6)Postman
695     POST http://localhost:8080/RestfulDemo/members
696     Body > raw > JSON(application/json)
697
698     {
699         "userid" : "girlsage",
700         "name" : "소녀시대",
701         "gender" : "여성",
702         "city" : "수원"
703     }
704
705     Send 버튼 클릭하면
706
707     Body
708     {"code": "success"}
709
710 7)WebContent/register.html
711
712     <!DOCTYPE html>
713     <html>
714         <head>
715             <meta charset="UTF-8">
716             <title>Member Add</title>
717             <script src="js/jquery-1.12.4.js"></script>
```

[illegible]

8)register.html > right-click > Run As > Run on Server

24. 사용자 정보 수정 구현하기

1)HomeController.java

```
762 @RequestMapping(value = "/members", method = RequestMethod.PUT)
763 @ResponseBody
764 public Map update(@RequestBody MemberVO memberVO) {
765     this.service.updateMember(memberVO);
766     Map<String, Object> map = new HashMap<String, Object>();
767     map.put("code", "success");
768     return map;
769 }
```



```

822         "<input type='radio' class='gender' value='여성' />여성";
823     }else{
824         str = "<input type='radio' class='gender' name='gender' value='남성' />남성
            &nbsp;&nbsp;&nbsp;";
825         "<input type='radio' class='gender' name='gender' value='여성' checked />여성";
826     }
827     $("span#gender").replaceWith(str);
828     var city = $("#city").text();
829     $("span#city")
830     .replaceWith("<input type='text' id='city' value='" + city + "'/>");
831     flag = true;
832 }else{
833     $.ajax({
834         url : "/RestfulDemo/members",
835         type : "PUT",
836         data : JSON.stringify({
837             "userid" : userid,
838             "name" : $("#name").val(),
839             "gender" : $(".gender:checked").val(),
840             "city" : $("#city").val()
841         }),
842         contentType : "application/json;charset=utf-8",
843         success : function(data){
844             alert(data.code);
845             location.reload();
846         }
847     });
848     flag = false;
849 }
850 }

```

852 25. 사용자 정보 삭제 구현하기

853 1)HomeController.java

```

854
855
856 @RequestMapping(value = "/members/{userid}", method = RequestMethod.DELETE)
857 @ResponseBody
858 public Map delete(@PathVariable String userid) {
859     this.service.deleteMember(userid);
860     Map<String, Object> map = new HashMap<String, Object>();
861     map.put("code", "success");
862     return map;
863 }

```

864 2)mabatis-mapper.xml

```

865
866
867 <delete id="delete" parameterType="String">
868     DELETE FROM Member WHERE userid = #{userid}
869 </delete>

```

870 3)MemberDaoImpl.java

872

```
873     @Override
874     public void delete(String userid) {
875         this.sqlSession.delete("Member.delete", userid);
876     }
877
878 4)MemberServiceImpl.java
879
880     @Override
881     public void deleteMember(String userid) {
882         this.memberDao.delete(userid);
883     }
884
885 5)Postman
886     DELETE http://localhost:8080/RestfulDemo/members/girlsage
887
888     Send button click하면
889
890     Body
891     {"code": "success"}
892
893 6)WebContent/view.html 수정
894     -아래의 code를 추가한다.
895
896     <a href = "javascript:void(0)" onclick="javascript:member_delete()">삭제하기</a>
897
898     function member_delete(){
899         $.ajax({
900             url : "/RestfulDemo/members/" + userid,
901             type : "DELETE",
902             success : function(data){
903                 alert(data.code);
904                 location.href = "/RestfulDemo/";
905             }
906         });
907     }
908
909
910 -----
911 Task2. Using Restful API with XML
912 1. Maven Repository에서 'spring oxm'로 검색
913 2. Spring Object/XML Marshalling에서 4.3.24.RELEASE 선택
914 3. pom.xml에 아래 dependency 추가 > Maven Clean > Mavan Install
915
916 <!-- https://mvnrepository.com/artifact/org.springframework/spring-oxm -->
917 <dependency>
918     <groupId>org.springframework</groupId>
919     <artifactId>spring-oxm</artifactId>
920     <version>5.2.0.RELEASE</version>
921 </dependency>
922
923
924 4. Maven Repository에서 'jaxb'로 검색, Jaxb Api에서 2.3.1
```

```
925
926 5. 아래의 dependency를 pom.xml에 추가 > Maven Clean > Mavan Install
927
928 <dependency>
929     <groupId>javax.xml.bind</groupId>
930     <artifactId>jaxb-api</artifactId>
931     <version>2.3.1</version>
932 </dependency>
933
934
935 6. src/com.example.vo/MemberListVO.java 생성
936
937 package com.example.vo;
938
939 import java.util.List;
940
941 import javax.xml.bind.annotation.XmlAccessType;
942 import javax.xml.bind.annotation.XmlAccessorType;
943 import javax.xml.bind.annotation.XmlElement;
944 import javax.xml.bind.annotation.XmlRootElement;
945
946 import org.springframework.stereotype.Component;
947
948 @XmlRootElement(name="memberList")
949 @XmlAccessorType(XmlAccessType.FIELD)
950 @Component
951 public class MemberListVO {
952     @XmlElement(name="member")
953     private List<MemberVO> memberList;
954
955     public List<MemberVO> getUserList() {
956         return memberList;
957     }
958
959     public void setUserList(List<MemberVO> memberList) {
960         this.memberList = memberList;
961     }
962 }
963
964
965 1)XML 문서는 반드시 단 하나의 root element를 가져야 한다.
966 2)여러 UserVO를 표현하려면 root element로 사용할 또 다른 Java class가 필요하다.
967 3)새로 생성한 MemberListVO객체는 이 객체가 root element에 해당하는 객체이며, root element 이름을
968   memberList로 설정하겠다는 의미로 @XmlRootElement(name="memberList") 설정을 추가했다.
969 4)그리고 memberList 변수 위에도 @XmlElement(name="member")를 추가했는데, MemberVO 마다
970   element 이름을 member로 변경할 것이다.
971
972 7. src/com.example.vo.MemberVO.java 수정
973
974 package com.example.vo;
```

```

975 import javax.xml.bind.annotation.XmlAccessType;
976 import javax.xml.bind.annotation.XmlAccessorType;
977 import javax.xml.bind.annotation.XmlAttribute;
978 import javax.xml.bind.annotation.XmlRootElement;
979
980 import org.springframework.stereotype.Component;
981
982 @XmlRootElement(name="member")
983 @XmlAccessorType(XmlAccessType.FIELD)
984 @Component
985 public class MemberVO {
986     @XmlAttribute
987     private String userid;
988     private String name, gender, city;
989
990     public MemberVO() {}
991
992 1)VO class에 선언된 @XmlAccessorType은 MemberVO 객체를 XML로 변환할 수 있다는 의미이다.
993 2)그리고 XmlAccessType.FIELD 때문에 이 객체가 가진 field, 즉 변수들은 자동으로 자식 element로 표현
994  된다.
995 3)하지만, 이 중에서 userid에만 @XmlAttribute가 붙었는데, 이는 userid를 속성으로 표현하라는 의미이다.
996 4)만일 JSON 변환시 @JsonIgnore가 변환시 제외하는 것처럼, XML변환시에도 제외할 변수는
997   @XmlTransient를 붙이면 된다.
998 5)마지막으로 변환시 변수가 참조형이면 반드시 기본 생성자가 있어야만 한다.
999
1000 8. Spring 설정 File(beans.xml)에서 p와 oxm check후, 아래 code 추가
1001 1)JSON 변환시 Java 객체를 JSON response body로 변환해주는 MappingJackson2HttpMessageConverter
1002  를 Spring 설정 file에 추가해야 하는데, 설정 file에 <mvc:annotation-driven />으로 대체했었다.
1003 2)마찬가지로 Java 객체를 XML response body로 변환할 때는 아래의 code를 추가한다.
1004
1005     <bean id="xmlViewer" class="org.springframework.web.servlet.view.xml.MarshallingView">
1006         <constructor-arg>
1007             <bean class="org.springframework.xml.jaxb.Jaxb2Marshaller"
1008                 p:classesToBeBound="com.example.vo.MemberListVO"/>
1009         </constructor-arg>
1010     </bean>
1011
1012 9. com.example.controller.UserController.java
1013
1014 package com.example.controller;
1015
1016 import org.springframework.beans.factory.annotation.Autowired;
1017 import org.springframework.stereotype.Controller;
1018 import org.springframework.web.bind.annotation.RequestMapping;
1019 import org.springframework.web.bind.annotation.ResponseBody;
1020
1021 import com.example.service.MemberService;
1022 import com.example.vo.MemberListVO;
1023
1024 @Controller

```

```
1023 public class UserController {
1024     @Autowired
1025     private MemberService service;
1026
1027     @RequestMapping(value="/memberlist.do", produces="application/xml")
1028     @ResponseBody
1029     public MemberListVO userlist(){
1030         MemberListVO listVO = new MemberListVO();
1031         listVO.setUserList(this.service.select());
1032         return listVO;
1033     }
1034 }
```

```
1035
1036
1037 10. http://localhost:8080/RestfulDemo/memberlist.do
1038 <memberList>
1039     <member userid="girlsage">
1040         <name>소년시대</name>
1041         <gender>남성</gender>
1042         <city>부산</city>
1043     </member>
1044     <member userid="jimin">
1045         <name>박지민</name>
1046         <gender>여성</gender>
1047         <city>서울</city>
1048     </member>
1049 </memberList>
```