

```
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>

```

81
82 1)pom.xml > right-click > Run As > Maven install
83 [INFO] BUILD SUCCESS
84
85

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
103     http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
104     http://www.springframework.org/schema/beans
105     http://www.springframework.org/schema/beans/spring-beans.xsd
106     http://www.springframework.org/schema/context
107     http://www.springframework.org/schema/context/spring-context-4.3.xsd">
108
109 </beans>

```

109 9. ContextLoaderListener class 설정

- 110 1)Business logic의 Spring 설정 file (ex:applicationContext.xml)을 작성했기 때문에 listener로 ContextLoaderListener class를 정의해야 한다.
- 111 2)ContextLoaderListener class는 Spring 설정 file(default에서 file명 applicationContext.xml)을 load하면 ServletContextListener interface를 구현하고 있기 때문에 ServletContext instance 생성시(Tomcat으로 application이 load된 때)에 호출된다.
- 112 3)즉, ContextLoaderListener class는 DispatcherServlet class의 load보다 먼저 동작하여 business logic층을 정의한 Spring 설정 file을 load한다.
- 113 4)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입

```

114 <!-- needed for ContextLoaderListener -->
115 <context-param>
116   <param-name>contextConfigLocation</param-name>
117   <param-value>location</param-value>
118 </context-param>
119
120 <!-- Bootstraps the root web application context before servlet initialization -->
121 <listener>
122   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
123 </listener>

```

- 125 5)아래 code로 변환

```

126 <context-param>

```

```
127     <param-name>contextConfigLocation</param-name>
128     <param-value>classpath:applicationContext.xml</param-value>
129 </context-param>
130
131
```

10. DispatcherServlet Class 추가

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

```
134
135     <!-- The front controller of this Spring Web application, responsible for handling all application
136         requests -->
137     <servlet>
138         <servlet-name>springDispatcherServlet</servlet-name>
139         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
140         <init-param>
141             <param-name>contextConfigLocation</param-name>
142             <param-value>location</param-value>
143         </init-param>
144         <load-on-startup>1</load-on-startup>
145     </servlet>
146
147     <!-- Map all requests to the DispatcherServlet for handling -->
148     <servlet-mapping>
149         <servlet-name>springDispatcherServlet</servlet-name>
150         <url-pattern>url</url-pattern>
151     </servlet-mapping>
```

2) 아래의 code로 변환

```
152
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>
167
168
```

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 + ",
211             city=" + city + "]\n";
212     }
213 }
214

```

215 12. MemberDao 객체 생성

- 216 1)src/com.example.dao package 생성
- 217 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

```

233 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 }
337
338

```

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>

```

19. config/member-mapper.xml

```

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>

```

20. applicationContext.xml 아래 code 추가

```

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>

```

21. 전체 사용자 조회하기

1) HomeController 객체 code 추가

```

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 }

```

2) mybatis-mapper.xml

```

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>

```

3) MemberDaoImpl.java

```

457
458 @Override

```

```

459     public List<MemberVO> readAll() {
460         return this.sqlSession.selectList("Member.select");
461     }

```

4)MemberServiceImpl.java

```

465     @Override
466     public List<MemberVO> select() {
467         return this.memberDao.readAll();
468     }

```

5)Postman

GET http://localhost:8080/RestfulDemo/members Send
Body

```

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     }

```

6)WebContent > right-click > New > Folder > js -js/jquery-1.12.4.js

7)WebContent/index.html

```

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 +
522                                     "</a></td>" +
523                                     "<td>" + members[i].name + "</td>" +
524                                     "<td>" + members[i].gender + "</td>" +
525                                     "<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 }

```

2)mybatis-mapper.xml

```

570 <select id="selectMember" parameterType="String" resultType="memberVO">
571     SELECT * FROM Member WHERE userid = #{userid}
572 </select>

```

3)MemberDaoImpl.java

```

576 @Override
577 public MemberVO read(String userid) {
578     return this.sqlSession.selectOne("Member.selectMember", userid);
579 }

```

4)MemberServiceImpl.java

```

583 @Override
584 public MemberVO selectMember(String userid) {
585     return this.memberDao.read(userid);
586 }

```

5)Postman

```

589 GET http://localhost:8080/RestfulDemo/members/jimin Send
590 Body

```

```

592     {
593         "code": "success",
594         "data": {
595             "userId": "jimin",
596             "name": "한지민",
597             "gender": "여",
598             "city": "서울"
599         }
600     }

```

6)WebContent/view.html

```

603 <!DOCTYPE html>
604 <html>
605     <head>
606         <meta charset="UTF-8">
607         <title>회원 정보 페이지</title>
608         <script src="js/jquery-1.12.4.js"></script>
609         <script>
610             var userid = null;
611
612             $(function(){
613                 userid = location.search.substring(1).split("=")[1];
614                 $.ajax({
615                     url : "/RestfulDemo/members/" + userid,
616                     type : "GET",
617                     success : function(data){
618                         var member = data.data;
619                         $("#userid").text(member.userid);
620                         $("#name").text(member.name);
621                         $("#gender").text(member.gender);
622                         $("#city").text(member.city);
623                     }
624                 });
625             });
626         </script>
627     </head>
628     <body>
629         <h1>Member Information</h1>
630         <ul>
631             <li>아이디 : <span id="userid"></span></li>
632             <li>이름 : <span id="name"></span></li>
633             <li>성별 : <span id="gender"></span></li>
634             <li>거주지 : <span id="city"></span></li>
635         </ul>
636         <a href = "javascript:void(0)" onclick="javascript:history.back();">목록으로</a>
637     </body>
638 </html>

```

7)view.html > right-click > Run As > Run on Server

8)view.html?userid=jimin or index.html에서 jimin link click

23. 사용자 등록 구현하기

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 }

```

2)mabatis-mapper.xml

```

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

```

3)MemberDaoImpl.java

```

667 @Override
668 public void create(MemberVO member) {
669     this.sqlSession.insert("Member.insert", member);
670 }

```

4)MemberServiceImpl.java

```

674 @Override
675 public void insertMember(MemberVO member) {
676     this.memberDao.create(member);
677 }

```

5)web.xml에 한글이 post방식으로 처리할 때 깨짐 방지 하기 위한 fileter 복사해서 넣기

```

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>

```

6)Postman

POST <http://localhost:8080/RestfulDemo/members>
 Body > raw > JSON(application/json)

```

698 {
699     "userid" : "girlsage",
700     "name" : "소녀시대",
701     "gender" : "여성",
702     "city" : "수원"
703 }

```

Send 버튼 클릭하면

```

707 Body
708     {"code": "success"}

```

7)WebContent/register.html

```

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>
718         <script>
719             $(function(){
720                 $("input[type='button']").bind("click", function(){
721                     $.ajax({
722                         url : "/RestfulDemo/members",
723                         contentType : "application/json;charset=utf-8",
724                         type : "POST",
725                         data : JSON.stringify({

```

[illegible]

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

24. 사용자 정보 수정 구현하기

1)HomeController.java

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

2)mabatis-mapper.xml

```
<update id="update" parameterType="memberVO">
    UPDATE Member SET name = #{name}, gender = #{gender}, city = #{city}
    WHERE userid = #{userid}
</update>
```

3)MemberDaoImpl.java

```
@Override
public void update(MemberVO member) {
    this.sqlSession.update("Member.update", member);
}
```

4)MemberServiceImpl.java

```
@Override
public void updateMember(MemberVO member) {
    this.memberDao.update(member);
}
```

5) Postman

```
PUT http://localhost:8080/RestfulDemo/members
Body > raw > JSON(application/json)
```

```
{
  "userid" : "girlsage",
  "name" : "소년시대",
  "gender" : "남성",
  "city" : "부산"
}
```

Send 버튼 클릭하면

```
Body
{"code": "success"}
```

6)WebContent/view.html 수정

-아래 code를 추가한다.

```
<a href = "javascript:void(0)" onclick="javascript:member_update()">수정하기</a>
```

```
var flag = false;
function member_update(){
  if(!flag){ //수정하기를 click하면
    var name = $("#name").text();
    $("#span#name")
    .replaceWith("<input type='text' id='name' value='" + name + "'/>");
    var gender = $("#gender").text();
    var str = null;
    if(gender == "남성"){
      str = "<input type='radio' class='gender' value='남성' checked/>남성&nbsp;&nbsp;&nbsp;" +
        "<input type='radio' class='gender' value='여성' />여성";
    }else{
      str = "<input type='radio' class='gender' name='gender' value='남성'
        />남성&nbsp;&nbsp;&nbsp;" +
        "<input type='radio' class='gender' name='gender' value='여성' checked />여성";
    }
    $("#span#gender").replaceWith(str);
    var city = $("#city").text();
    $("#span#city")
    .replaceWith("<input type='text' id='city' value='" + city + "'/>");
    flag = true;
  }else{
    $.ajax({
      url : "/RestfulDemo/members",
      type : "PUT",
      data : JSON.stringify({
        "userid" : userid,
        "name" : $("#name").val(),
        "gender" : $(".gender:checked").val(),
        "city" : $("#city").val()
      }),
      contentType : "application/json;charset=utf-8",
      success : function(data){
        alert(data.code);
        location.reload();
      }
    });
    flag = false;
  }
}
```

25. 사용자 정보 삭제 구현하기

1)HomeController.java

```
@RequestMapping(value = "/members/{userid}", method = RequestMethod.DELETE)
@ResponseBody
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

```

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

```

870 3)MemberDaoImpl.java

```

872     @Override
873     public void delete(String userid) {
874         this.sqlSession.delete("Member.delete", userid);
875     }

```

877 4)MemberServiceImpl.java

```

879     @Override
880     public void deleteMember(String userid) {
881         this.memberDao.delete(userid);
882     }

```

884 5)Postman

886 DELETE http://localhost:8080/RestfulDemo/members/girlsage

888 Send button click하면

```

889     Body
890     {"code": "success"}

```

892 6)WebContent/view.html 수정

894 -아래의 code를 추가한다.

```

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     }

```

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

```

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>

```

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 memberList로 설정하겠다는 의미로 @XmlRootElement(name="memberList") 설정을 추가했다.
968 4)그리고 memberList 변수 위에도 @XmlElement(name="member")를 추가했는데, MemberVO 마다 element 이름을 member로 변경할 것이다.

969
970
971 7. src/com.example.vo.MemberVO.java 수정

```
972
973 package com.example.vo;
974
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() {}
```

- 1)VO class에 선언된 @XmlAccessorType은 MemberVO 객체를 XML로 변환할 수 있다는 의미이다.
2)그리고 XmlAccessType.FIELD 때문에 이 객체가 가진 field, 즉 변수들은 자동으로 자식 element로 표현된다.
3)하지만, 이 중에서 userid에만 @XmlAttribute가 붙었는데, 이는 userid를 속성으로 표현하라는 의미이다.
4)만일 JSON 변환시 @JsonIgnore가 변환시 제외하는 것처럼, XML변환시에도 제외할 변수는 @XmlTransient를 붙이면 된다.
5)마지막으로 변환시 변수가 참조형이면 반드시 기본 생성자가 있어야만 한다.

8. Spring 설정 File(beans.xml)에서 p와 oxm check후, 아래 code 추가

- 1)JSON 변환시 Java 객체를 JSON response body로 변환해주는 MappingJackson2HttpMessageConverter를 Spring 설정 file에 추가해야 하는데, 설정 file에 <mvc:annotation-driven />으로 대체했었다.
2)마찬가지로 Java 객체를 XML response body로 변환할 때는 아래의 code를 추가한다.

```
<bean id="xmlViewer" class="org.springframework.web.servlet.view.xml.MarshallingView">
    <constructor-arg>
        <bean class="org.springframework.oxm.jaxb.Jaxb2Marshaller"
            p:classesToBeBound="com.example.vo.MemberListVO"/>
    </constructor-arg>
</bean>
```

9. com.example.controller.UserController.java

```
package com.example.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import com.example.service.MemberService;
import com.example.vo.MemberListVO;

@Controller
public class UserController {
    @Autowired
    private MemberService service;

    @RequestMapping(value="/memberlist.do", produces="application/xml")
    @ResponseBody
    public MemberListVO userList(){
        MemberListVO listVO = new MemberListVO();
        listVO.setUserList(this.service.select());
        return listVO;
    }
}
```

10. <http://localhost:8080/RestfulDemo/memberlist.do>

```
<memberList>
    <member userid="girlsage">
        <name>소년시대</name>
        <gender>남성</gender>
        <city>부산</city>
    </member>
    <member userid="jimin">
        <name>박지민</name>
        <gender>여성</gender>
        <city>서울</city>
    </member>
</memberList>
```

Task3. Spring Restful BBS(단순게시판)

1. Package Explorer > right-click > New > Spring Legacy Project

1055 2. Select Spring MVC Project
1056 3. Project name : SpringRestfulBbs > Next
1057 4. Enter a topLevelPackage : com.example.biz > Finish
1058
1059 5. Version check
1060 1)Project > right-click > Properties
1061 2)Java Build Path
1062 -Libraries Tab> JRE System Library [JavaSE-1.6] select
1063 -Edit button click > [Workspace default JRE (jdk1.8.0_251) select
1064 -Finish > Apply click
1065 3)Java Compiler
1066 -Compiler compliance level : 1.6 -> 1.8로 변환
1067 4)Project Facets
1068 -Java > 1.8로 변환
1069 -Runtimes tab > Apache Tomcat v9.0 check
1070 5)Apply and Close
1071 6)혹시 [Compiler Settings Changed]창이 나오면 [Yes] click.
1072
1073
1074 6. pom.xml 수정하기
1075 <properties>
1076 <java-version>1.8</java-version>
1077 <org.springframework-version>5.2.7.RELEASE</org.springframework-version>
1078 <org.aspectj-version>1.9.5</org.aspectj-version>
1079 <org.slf4j-version>1.7.30</org.slf4j-version>
1080 </properties>
1081 ...
1082 <dependency>
1083 <groupId>javax.servlet</groupId>
1084 <artifactId>javax.servlet-api</artifactId>
1085 <version>4.0.1</version>
1086 <scope>provided</scope>
1087 </dependency>
1088 <dependency>
1089 <groupId>javax.servlet.jsp</groupId>
1090 <artifactId>javax.servlet.jsp-api</artifactId>
1091 <version>2.3.3</version>
1092 <scope>provided</scope>
1093 </dependency>
1094 <dependency>
1095 <groupId>org.junit.jupiter</groupId>
1096 <artifactId>junit-jupiter-api</artifactId>
1097 <version>5.6.2</version>
1098 <scope>test</scope>
1099 </dependency>
1100
1101
1102 7. pom.xml > right-click > Run As > Maven install
1103 [INFO] BUILD SUCCESS
1104
1105
1106 8. Database 작업
1107 1)bbs 계정 생성
1108
1109 ALTER SESSION SET "_ORACLE_SCRIPT"=true;
1110
1111 CREATE USER bbs IDENTIFIED BY bbs
1112 DEFAULT TABLESPACE USERS
1113 TEMPORARY TABLESPACE TEMP;
1114
1115 ALTER USER bbs
1116 DEFAULT TABLESPACE USERS
1117 QUOTA UNLIMITED ON USERS;
1118
1119 GRANT resource, connect TO bbs;
1120 conn bbs/bbs
1121

```

1122 2)Create Table
1123
1124 CREATE TABLE Bbs
1125 (
1126     idx          NUMBER(4),
1127     username     VARCHAR2(20) NOT NULL,
1128     title        VARCHAR2(50) NOT NULL,
1129     contents     VARCHAR2(1000) NOT NULL,
1130     email        VARCHAR2(100),
1131     readnum      NUMBER(4) NOT NULL,
1132     writeday     DATE NOT NULL,
1133     CONSTRAINT bbs_idx_pk PRIMARY KEY(idx)
1134 )
1135
1136 3)Create Sequence
1137
1138 CREATE SEQUENCE bbs_seq
1139 START WITH 1
1140 INCREMENT BY 1
1141 MAXVALUE 9999
1142 NOCYCLE;
1143
1144
1145 9. src/main/webapp/static folder 생성
1146 1)src/main/webapp/static/css folder
1147 -bootstrap-theme.min.css
1148 -bootstrap.min.css
1149 -style.css
1150
1151 /* CSS Document */
1152 body {
1153     margin-left: 0px;
1154     margin-top: 0px;
1155     margin-right: 0px;
1156     margin-bottom: 0px;
1157     font-size:12px;
1158 }
1159
1160 2)src/main/webapp/static/fonts folder
1161 -glyphicons-halflings-regular.eot
1162 -glyphicons-halflings-regular.svg
1163 -glyphicons-halflings-regular.ttf
1164 -glyphicons-halflings-regular.woff
1165 -glyphicons-halflings-regular.woff2
1166
1167 3)src/main/webapp/static/images folder
1168
1169 4)src/main/webapp/static/js folder
1170 -jquery-3.5.1.min.js
1171 -bootstrap.min.js
1172
1173 5)src/main/webapp/static/index.html
1174 <!DOCTYPE html>
1175 <html lang="en">
1176 <head>
1177     <meta charset="UTF-8">
1178     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1179     <title>Welcome to example.com</title>
1180     <link rel="stylesheet" href="static/css/bootstrap.min.css">
1181 </head>
1182 <body>
1183     <div class="jumbotron">
1184         <h1>Hello, world!</h1>
1185         <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Sit cupiditate aliquam eaque
obcaecati dolorem nulla nihil natus corrupti repellendus! Praesentium, quae itaque quasi
soluta sunt accusamus cumque incidunt quam dignissimos.</p>
1186         <p><a class="btn btn-primary btn-lg" href="static/list.html" role="button">BBS</a></p>

```

```

1187         </div>
1188     </body>
1189 </html>
1190
1191
1192 10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
1193     <resources mapping="/static/**" location="/static/" /> 추가
1194     <context:component-scan base-package="com.example" /> 수정
1195
1196
1197 11. web.xml 수정하기
1198     1)다음 코드 추가
1199
1200         <welcome-file-list>
1201             <welcome-file>/static/index.html</welcome-file>
1202         </welcome-file-list>
1203
1204     2)서버 시작
1205         -Project > right-click > Run on Server
1206
1207         http://localhost:8080/biz/ --> index.html로 연결 확인
1208
1209
1210 12. src/main/resources/oracle.properties
1211     db.driverClass=oracle.jdbc.driver.OracleDriver
1212     db.url=jdbc:oracle:thin:@localhost:1521:ORCL
1213     db.username=bbs
1214     db.password=bbs
1215
1216
1217 13. Spring jdbc 설치하기
1218     1)mvnrepository.com에서 'spring jdbc'로 검색하여 아래 코드 pom.xml에 추가한다.
1219
1220         <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
1221         <dependency>
1222             <groupId>org.springframework</groupId>
1223             <artifactId>spring-jdbc</artifactId>
1224             <version>5.2.7.RELEASE</version>
1225         </dependency>
1226
1227     2)pom.xml에 붙여 넣고 Maven Install 하기
1228         [INFO] BUILD SUCCESS
1229
1230
1231 14. HikariCP 설치하기
1232     1)HikariCP를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
1233
1234         <!-- https://mvnrepository.com/artifact/com.zaxxer/HikariCP -->
1235         <dependency>
1236             <groupId>com.zaxxer</groupId>
1237             <artifactId>HikariCP</artifactId>
1238             <version>3.4.5</version>
1239         </dependency>
1240
1241     2)pom.xml에 붙여 넣고 Maven Install 하기
1242         [INFO] BUILD SUCCESS
1243
1244
1245 15. Oracle Jdbc Driver library 검색 및 설치
1246     1)지난번에 설치한 Oracle Driver를 등록한다.
1247
1248         <dependency>
1249             <groupId>com.oracle</groupId>
1250             <artifactId>ojdbc8</artifactId>
1251             <version>12.2</version>
1252         </dependency>
1253

```

1254 2)pom.xml에 붙여 넣고 Maven Install 하기
1255 [INFO] BUILD SUCCESS
1256
1257

1258 16. MyBatis 설치하기

1259 1)mvnrepository.com에서 'MyBatis'로 검색
1260

```
1261 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->  
1262 <dependency>  
1263   <groupId>org.mybatis</groupId>  
1264   <artifactId>mybatis</artifactId>  
1265   <version>3.5.5</version>  
1266 </dependency>  
1267 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->  
1268 <dependency>  
1269   <groupId>org.mybatis</groupId>  
1270   <artifactId>mybatis-spring</artifactId>  
1271   <version>2.0.5</version>  
1272 </dependency>
```

1273
1274 2)pom.xml에 붙여 넣고 Maven Install 하기
1275 [INFO] BUILD SUCCESS
1276
1277

1278 17. Lombok library 설치하기

1279 1)mvnrepository.com에서 'lombok'로 검색
1280

```
1281 <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->  
1282 <dependency>  
1283   <groupId>org.projectlombok</groupId>  
1284   <artifactId>lombok</artifactId>  
1285   <version>1.18.12</version>  
1286   <scope>provided</scope>  
1287 </dependency>
```

1288
1289 2)pom.xml에 붙여 넣고 Maven Install 하기
1290 [INFO] BUILD SUCCESS
1291
1292

1293 18. root-context.xml 수정

1294 1)src/main/webapp/WEB-INF/spring/root-context.xml

```
1295 <?xml version="1.0" encoding="UTF-8"?>  
1296 <beans xmlns="http://www.springframework.org/schema/beans"  
1297   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1298   xmlns:context="http://www.springframework.org/schema/context"  
1299   xsi:schemaLocation="http://www.springframework.org/schema/beans  
1300     http://www.springframework.org/schema/beans/spring-beans.xsd  
1301     http://www.springframework.org/schema/context  
1302     http://www.springframework.org/schema/context/spring-context-4.3.xsd">  
1303  
1304   <context:property-placeholder location="classpath:oracle.properties"/>  
1305  
1306   <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource">  
1307     <property name="driverClassName" value="${db.driverClass}" />  
1308     <property name="jdbcUrl" value="${db.url}" />  
1309     <property name="username" value="${db.username}" />  
1310     <property name="password" value="${db.password}" />  
1311     <property name="maximumPoolSize" value="30" />  
1312     <property name="minimumIdle" value="5" />  
1313     <property name="autoCommit" value="true" />  
1314   </bean>  
1315  
1316   <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
1317     <property name="dataSource" ref="dataSource" />  
1318     <property name="configLocation" value="classpath:mybatis-config.xml" />  
1319     <property name="mapperLocations">  
1320       <list>
```

```

1319         <value>classpath:bbs-mapper.xml</value>
1320     </list>
1321 </property>
1322 </bean>
1323 <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
1324     <constructor-arg index="0" ref="sqlSessionFactory" />
1325 </bean>
1326 </beans>

```

19. MyBatis 관련 xml 파일 설정하기

1)src/main/resources
-mybatis-config.xml

```

1333 <?xml version="1.0" encoding="UTF-8"?>
1334 <!DOCTYPE configuration
1335     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
1336     "http://mybatis.org/dtd/mybatis-3-config.dtd">
1337 <configuration>
1338     <typeAliases>
1339         <typeAlias type="com.example.vo.BbsVO" alias="bbsVo"/>
1340     </typeAliases>
1341 </configuration>

```

-bbs-mapper.xml

```

1345 <?xml version="1.0" encoding="UTF-8"?>
1346 <!DOCTYPE mapper
1347     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
1348     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
1349 <mapper namespace="com.example.vo.BbsVO">
1350
1351     <select id="currentDate" resultType="java.util.Date">
1352         SELECT SYSDATE FROM dual
1353     </select>
1354
1355 </mapper>

```

20. package 생성

1)src/main/java/com.example.vo
2)src/main/java/com.example.dao
3)src/main/java/com.example.service
4)src/main/java/com.example.controller

21. com.example.vo.BbsVO 생성

```

1366 package com.example.vo;
1367
1368 import java.util.Date;
1369
1370 import lombok.Getter;
1371 import lombok.NoArgsConstructor;
1372 import lombok.Setter;
1373 import lombok.ToString;
1374
1375 @Setter
1376 @Getter
1377 @NoArgsConstructor
1378 @ToString
1379 public class BbsVO {
1380     private int idx, readnum;
1381     private String username, title, contents, email;
1382     private Date writeday;
1383 }

```

```
1386 22. Project 실행
1387     1)Project > right-click > Run On Server
1388
1389     http://localhost:8080/biz/
1390
1391     2)현재 index.html이 문제없이 rendering되어야 한다.
1392
1393
1394 23. Spring TestContext Framework 설치
1395     1)mvnrepository.com에서 'spring test'로 검색하여 아래의 dependency를 설치한다.
1396
1397     <!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
1398     <dependency>
1399         <groupId>org.springframework</groupId>
1400         <artifactId>spring-test</artifactId>
1401         <version>5.2.7.RELEASE</version>
1402         <scope>test</scope>
1403     </dependency>
1404
1405     2)pom.xml에 붙여 넣고 Maven Install 하기
1406     [INFO] BUILD SUCCESS
1407
1408
1409 24. Spring Test로 검사하기
1410     1)src/test/java/com.example.biz/TestApp class 생성
1411         -com.example.biz > right-click > New > JUnit Test Case
1412         -Select [New JUnit Jupiter test]
1413         -Name : TestApp
1414         -Finish
1415
1416         package com.example.biz;
1417
1418         import java.util.Date;
1419
1420         import org.junit.jupiter.api.Test;
1421         import org.junit.jupiter.api.extension.ExtendWith;
1422         import org.mybatis.spring.SqlSessionTemplate;
1423         import org.springframework.beans.factory.annotation.Autowired;
1424         import org.springframework.test.context.ContextConfiguration;
1425         import org.springframework.test.context.junit.jupiter.SpringExtension;
1426
1427         import lombok.extern.slf4j.Slf4j;
1428
1429         @Slf4j
1430         @ExtendWith(SpringExtension.class)
1431         @ContextConfiguration(locations =
1432             {"file:src/main/webapp/WEB-INF/spring/root-context.xml"})
1433         class TestApp {
1434             @Autowired
1435             private SqlSessionTemplate sqlSession;
1436
1437             @Test
1438             void test() {
1439                 Date today = (Date)this.sqlSession.selectOne("currentDate");
1440                 log.info("Today is " + today);
1441             }
1442         }
1443
1444     2)Run as > JUnit Test > Green bar
1445
1446     INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
1447     INFO : com.example.biz.TestApp - Today is Thu Jun 25 12:44:06 KST 2020
1448     INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
1449     INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
1450
1451
```

```

1452 25. com/example.dao
1453     1)BbsDao interface
1454
1455         package com.example.dao;
1456
1457         import java.util.Map;
1458
1459         import com.example.vo.BbsVO;
1460
1461         public interface BbsDao {
1462             void insert(BbsVO bbsVo);
1463             void select(Map map);
1464             void selectAll(Map map);
1465             void update(BbsVO bbsVo);
1466             void delete(int idx);
1467         }
1468
1469     2)BbsDaoImpl.java
1470         package com.example.dao;
1471
1472         import java.util.Map;
1473
1474         import org.mybatis.spring.SqlSessionTemplate;
1475         import org.springframework.beans.factory.annotation.Autowired;
1476         import org.springframework.stereotype.Repository;
1477
1478         import com.example.vo.BbsVO;
1479
1480         @Repository("bbsDao")
1481         public class BbsDaoImpl implements BbsDao {
1482             @Autowired
1483             private SqlSessionTemplate sqlSession;
1484
1485             @Override
1486             public void insert(BbsVO bbsVo) {
1487             }
1488
1489             @Override
1490             public void select(Map map) {
1491             }
1492
1493             @Override
1494             public void selectAll(Map map) {
1495             }
1496
1497             @Override
1498             public void update(BbsVO bbsVo) {
1499             }
1500
1501             @Override
1502             public void delete(int idx) {
1503             }
1504         }
1505
1506
1507 26. com.example.service
1508     1)BbsService interface
1509         package com.example.service;
1510
1511         import java.util.Map;
1512
1513         import com.example.vo.BbsVO;
1514
1515         public interface BbsService {
1516             void create(BbsVO bbsVo);
1517             void read(Map map);
1518             void readAll(Map map);

```

```

1519         void update(BbsVO bbsVo);
1520         void delete(int idx);
1521     }
1522
1523 2)BbsServiceImpl.java
1524     package com.example.service;
1525
1526     import java.util.Map;
1527
1528     import org.springframework.beans.factory.annotation.Autowired;
1529     import org.springframework.stereotype.Service;
1530
1531     import com.example.dao.BbsDao;
1532     import com.example.vo.BbsVO;
1533
1534     @Service("bbsService")
1535     public class BbsServiceImpl implements BbsService {
1536         @Autowired
1537         private BbsDao bbsDao;
1538
1539         @Override
1540         public void create(BbsVO bbsVo) {
1541         }
1542
1543         @Override
1544         public void read(Map map) {
1545         }
1546
1547         @Override
1548         public void readAll(Map map) {
1549         }
1550
1551         @Override
1552         public void update(BbsVO bbsVo) {
1553         }
1554
1555         @Override
1556         public void delete(int idx) {
1557         }
1558     }

```

1561 27. com.example.controller

1562 1)MainController.java

```

1563
1564     package com.example.controller;
1565
1566     import org.springframework.beans.factory.annotation.Autowired;
1567     import org.springframework.stereotype.Controller;
1568
1569     import com.example.service.BbsService;
1570
1571     import lombok.extern.slf4j.Slf4j;
1572
1573     @Slf4j
1574     @Controller
1575     public class MainController {
1576         @Autowired
1577         private BbsService bbsService;
1578     }

```

1581 28. Jackson Library 추가하기

1582 1)mvnrepository.com에서 'jackson'으로 검색하여 아래의 dependency를 추가한다.

```

1583
1584     <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
1585     <dependency>

```



```

1586     <groupId>com.fasterxml.jackson.core</groupId>
1587     <artifactId>jackson-databind</artifactId>
1588     <version>2.11.0</version>
1589 </dependency>
1590 <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
1591 <dependency>
1592     <groupId>com.fasterxml.jackson.core</groupId>
1593     <artifactId>jackson-core</artifactId>
1594     <version>2.11.0</version>
1595 </dependency>
1596 <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
1597 <dependency>
1598     <groupId>com.fasterxml.jackson.core</groupId>
1599     <artifactId>jackson-annotations</artifactId>
1600     <version>2.11.0</version>
1601 </dependency>

```

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

29. 전체 게시판 글 조회하기

1)Stored Procedure 작성하기

```

1610 CREATE OR REPLACE PROCEDURE sp_bbs_select_all
1611 (
1612     bbs_record OUT SYS_REFCURSOR
1613 )
1614 AS
1615 BEGIN
1616     OPEN bbs_record FOR
1617     SELECT idx, username, email, title, writeday, readnum
1618     FROM BBS
1619     ORDER BY idx DESC;
1620 END;

```

2)MainController 수정

```

1624 @RequestMapping(value = "/bbs", method = RequestMethod.GET)
1625 @ResponseBody
1626 public Map list() {
1627     Map<String, Object> map = new HashMap<String, Object>();
1628     this.bbsService.readAll(map);
1629     map.put("code", "success");
1630     List<BbsVO> list = (List<BbsVO>)map.get("results");
1631     log.info("list = " + list);
1632     map.remove("results");
1633     map.put("data", list);
1634     return map;
1635 }

```

3)BbsServiceImpl.java

```

1639 @Override
1640 public void readAll(Map map) {
1641     this.bbsDao.selectAll(map);
1642 }

```

4)BbsDaoImpl.java

```

1646 @Override
1647 public void selectAll(Map map) {
1648     this.sqlSession.selectList("selectAll", map);
1649 }

```

5)bbs-mapper.xml

```

1653 <resultMap type="bbsVo" id="BbsResultMap">
1654   <result property="idx" javaType="java.lang.Integer"
1655     column="idx" jdbcType="INTEGER" />
1656   <result property="username" javaType="java.lang.String"
1657     column="username" jdbcType="VARCHAR" />
1658   <result property="email" javaType="java.lang.String"
1659     column="email" jdbcType="VARCHAR" />
1660   <result property="title" javaType="java.lang.String"
1661     column="title" jdbcType="VARCHAR" />
1662   <result property="contents" javaType="java.lang.String"
1663     column="contents" jdbcType="VARCHAR" />
1664   <result property="readnum" javaType="java.lang.Integer"
1665     column="readnum" jdbcType="INTEGER" />
1666   <result property="writeday" javaType="java.util.Date"
1667     column="writeday" jdbcType="DATE" />
1668 </resultMap>
1669
1670 <parameterMap type="bbsVo" id="selectAllParameterMap">
1671   <parameter property="results" javaType="ResultSet"
1672     jdbcType="CURSOR" mode="OUT" resultMap="BbsResultMap" />
1673 </parameterMap>
1674 <select id="selectAll" parameterMap="selectAllParameterMap"
1675   statementType="CALLABLE">
1676   { call sp_bbs_select_all(?) }
1677 </select>

```

6)Postman

```

1680 GET http://localhost:8080/biz/bbs Send
1681 Body

```

```

1682 {
1683   "code": "success",
1684   "data": []
1685 }

```

7)src/main/webapp/static/list.html

```

1690 <!DOCTYPE html>
1691 <html lang="en">
1692 <head>
1693   <meta charset="UTF-8">
1694   <meta name="viewport" content="width=device-width, initial-scale=1.0">
1695   <title>게시판</title>
1696   <style type="text/css">
1697     table{ margin:auto;border:0px solid;width:800px}
1698     th, td{border:0px solid}
1699   </style>
1700   <script src="js/jquery-3.5.1.min.js"></script>
1701   <script>
1702     $(function(){
1703       $.ajax({
1704         url : '/biz/bbs',
1705         type : 'GET',
1706         dataType : 'json',
1707         success : function(data){
1708           var result = data.data;
1709           str = "";
1710           if(result.length == 0){
1711             str = "<tr><td colspan='5' style='text-align:center'>NO Data</td></tr>";
1712           }else{
1713             for(var i = 0 ; i < result.length ; i++){
1714               var mydata = result[i];
1715               var title = mydata.title;
1716               title = title.replace(/"/g, "");
1717               title = title.replace(/&lt;/g, "<");
1718               title = title.replace(/&gt;/g, ">");
1719               var writeday = new Date(mydata.writeday);

```

```

1720         var idx = mydata.idx;
1721         str += "<tr onmouseover='myover(this)' onmouseout='myout(this)'>";
1722         str += "<td style='text-align:center'>" + idx + "</td>";
1723         str += "<td style='text-align:center'><a style='text-decoration:none'";
1724         href='mailto:' + mydata.email + "'>" + mydata.username + "</a></td>";
1725         str += "<td style='text-align:center'><a href='view.html?idx=" + idx +";
1726         "'>" + title + "</a></td>";
1727         str += "<td style='text-align:center'>" + writeday.toLocaleDateString() +
1728         "</td>";
1729         str += "<td style='text-align:center'>" + mydata.readnum + "</td>";
1730         str += "</tr>";
1731     }
1732     }
1733     $("#results").html(str);
1734 }
1735 });
1736 function myover(t){
1737     t.style.backgroundColor='yellow';
1738 }
1739 function myout(t){
1740     t.style.backgroundColor = 'white';
1741 }
1742 </script>
1743 </head>
1744 <body>
1745 <h1 style="text-align: center;color: #0000ff">자유 게시판</h1>
1746 <table>
1747 <tr>
1748 <td align="left"><a href="write.html"></a></td>
1749 <td align="right">()</td>
1750 </tr>
1751 </table>
1752 <table>
1753 <thead>
1754 <tr>
1755 <td colspan="5" height="1" style="background-color: #1F4F8F"></td>
1756 </tr>
1757 <tr style="background-color: #87E8FF">
1758 <th width="10%">번호</th>
1759 <th width="15%">등록자</th>
1760 <th width="50%">제목</th>
1761 <th width="15%">날짜</th>
1762 <th width="10%">조회수</th>
1763 </tr>
1764 <tr style="text-align:center">
1765 <td colspan="5" height="1" style="background-color: #1F4F8F"></td>
1766 </tr>
1767 </thead>
1768 <tbody id="results">
1769 </tbody>
1770 </table>
1771 </body>
1772 </html>

```

8)index.html 수정하기

```

1773 <div class="jumbotron">
1774 <h1>Hello, world!</h1>
1775 <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Sit cupiditate aliquam eaque
1776 obcaecati dolorem nulla nihil natus corrupti repellendus! Praesentium, quae itaque quasi
1777 soluta sunt accusamus cumque incidunt quam dignissimos.</p>
1778 <p><a class="btn btn-primary btn-lg" href="static/list.html" role="button">BBS</a></p>
1779 </div>
1780
1781

```

```

1782 30. 게시판 글 입력하기
1783 1)src/main/webapp/static/write.html
1784
1785 <!DOCTYPE html>
1786 <html lang="en">
1787 <head>
1788 <meta charset="UTF-8">
1789 <title>게시판 글 쓰기</title>
1790 <link href="css/style.css" rel="stylesheet" type="text/css">
1791 <style>
1792     body {
1793         margin-top: 50px;
1794     }
1795
1796     .bg {
1797         background-color: #1F4F8F
1798     }
1799
1800     #bg1 {
1801         background-color: #DFEDFF
1802     }
1803
1804     div {
1805         width: 600px;
1806         margin: auto;
1807     }
1808
1809     .tdbg {
1810         background-color: #f4f4f4;
1811         text-align: center;
1812     }
1813 </style>
1814 <script src="js/jquery-3.5.1.min.js"></script>
1815 <script>
1816     $(function(){
1817         $("input[type='image']").bind("click", function(){
1818             if(!$("#username").val()){
1819                 alert("이름이 빠졌습니다.");
1820                 $("#username").focus();
1821                 return false;
1822             }
1823             if(!$("#title").val()){
1824                 alert("제목이 빠졌습니다.");
1825                 $("#title").focus();
1826                 return false;
1827             }
1828             if(!$("#contents").val()){
1829                 alert("내용이 빠졌습니다.");
1830                 $("#contents").focus();
1831                 return false;
1832             }
1833
1834             $.ajax({
1835                 url: '/biz/bbs',
1836                 type: 'POST',
1837                 dataType: 'json',
1838                 contentType: 'application/json;charset=utf-8',
1839                 data: JSON.stringify({
1840                     'username': $('#username').val(),
1841                     'email': $('#email').val(),
1842                     'title': $('#title').val(),
1843                     'contents': $('#contents').val()
1844                 }),
1845                 success: function(data){
1846                     //alert(data.code); //success
1847                     location.href = "/biz/static/list.html";
1848                 }
1849             });
1850         });
1851     });
1852 </script>

```

```
1849         });
1850     });
1851 });
1852 </script>
1853 </head>
1854
1855 <body>
1856     <div>
1857         <table width="600" cellspacing="0" cellpadding="2">
1858             <tr>
1859                 <td colspan="2" height="1" class="bg"></td>
1860             </tr>
1861             <tr>
1862                 <td colspan="2" height="20" class="notice" id="bg1">&nbsp;&nbsp;&nbsp;게시판 새 글쓰기</td>
1863             </tr>
1864             <tr>
1865                 <td colspan="2" height="1" class="bg"></td>
1866             </tr>
1867             <tr>
1868                 <td width="124" height="30" class="tdbg">작성자</td>
1869                 <td width="494" style="padding:0 0 0 10">
1870                     <input type="text" name="username" class="input_style1" id="username"></td>
1871             </tr>
1872             <tr>
1873                 <td width="124" height="30" class="tdbg">Email</td>
1874                 <td width="494" style="padding:0 0 0 10">
1875                     <input type="text" name="email" class="input_style1" id="email"></td>
1876             </tr>
1877             <tr>
1878                 <td width="124" class="tdbg">제목</td>
1879                 <td width="494" style="padding:0 0 0 10" height="25">
1880                     <input type="text" name="title" size="60" maxlength="20" class="input_style2"
1881                         id="title"></td>
1882             </tr>
1883             <tr>
1884                 <td width="124" height="162" style="padding-top:7px;" class="tdbg">내용</td>
1885                 <td width="494" valign="top" style="padding:5 0 5 10">
1886                     <textarea cols="65" rows="10" name="contents" maxlength="2000"
1887                         class="textarea_style1" id="contents"></textarea>
1888                 </td>
1889             </tr>
1890             <tr>
1891                 <td colspan="2" height="1" class='button'></td>
1892             </tr>
1893             <tr>
1894                 <td colspan="2" height="1" class="bg"></td>
1895             </tr>
1896             <tr>
1897                 <td colspan="2" height="10"></td>
1898             </tr>
1899             <tr>
1900                 <td colspan="2">
1901                     <table width="100%" cellpadding="0" cellspacing="0">
1902                         <tr>
1903                             <td width="28%">&nbsp;&nbsp;&nbsp;</td>
1904                             <td width="51%">&nbsp;&nbsp;&nbsp;</td>
1905                             <td width="12%">
1906                                 <a href="javascript:history.back()">
1907                                     
1908                                 </a>
1909                             </td>
1910                             <td width="9%">
1911                                 <input type="image" src="images/ok.gif" width="46" height="20">
1912                             </td>
1913                         </tr>
1914                     </table>
1915                 </td>
1916             </tr>
1917         </table>
1918     </div>
1919 </body>
1920 </html>
```

```

1914         </tr>
1915     </table>
1916 </div>
1917 </body>
1918 </html>
1919

```

2)com.example.biz/MainController.java

```

1921 @RequestMapping(value = "/bbs", method = RequestMethod.POST)
1922 @ResponseBody
1923 public Map insert(@RequestBody BbsVO bbsVo) {
1924     String username = this.convert(bbsVo.getUsername());
1925     String email = this.convert(bbsVo.getEmail());
1926     String title = this.convert(bbsVo.getTitle());
1927     String contents = this.convert(bbsVo.getContents());
1928     bbsVo.setUsername(username);
1929     bbsVo.setEmail(email);
1930     bbsVo.setTitle(title);
1931     bbsVo.setContents(contents);
1932     //System.out.println(bbsVo);
1933     this.bbsService.create(bbsVo);
1934     Map<String, Object> map = new HashMap<String, Object>();
1935     map.put("code", "success");
1936     return map;
1937 }
1938
1939 private String convert(String oldStr) {
1940     String newStr = oldStr.replace("'", "");
1941     newStr = newStr.replace("<", "&lt;");
1942     newStr = newStr.replace(">", "&gt;");
1943     newStr = newStr.replace("\n", "<br />");
1944     return newStr;
1945 }
1946

```

3)com.example.service/BbsServiceImpl.java

```

1949 @Override
1950 public void create(BbsVO bbsVo) {
1951     this.bbsDao.insert(bbsVo);
1952 }
1953

```

4)com.example.dao.BbsDaoImpl.java

```

1956 @Override
1957 public void insert(BbsVO bbsVo) {
1958     this.sqlSession.insert("insert", bbsVo);
1959 }
1960

```

5)bbs-mapper.xml

```

1963 <parameterMap type="bbsVo" id="insertParameterMap">
1964     <parameter property="username" javaType="java.lang.String" jdbcType="VARCHAR"
1965         mode="IN" />
1966     <parameter property="email" javaType="java.lang.String" jdbcType="VARCHAR"
1967         mode="IN" />
1968     <parameter property="title" javaType="java.lang.String" jdbcType="VARCHAR" mode="IN" />
1969     <parameter property="contents" javaType="java.lang.String" jdbcType="VARCHAR"
1970         mode="IN" />
1971 </parameterMap>
1972 <insert id="insert" parameterMap="insertParameterMap" parameterType="bbsVo"
1973     statementType="CALLABLE">
1974     { call sp_bbs_insert(?, ?, ?, ?) }
1975 </insert>
1976

```

6)Store Procedure 생성

```

1976 CREATE OR REPLACE PROCEDURE sp_bbs_insert

```

```
(
  v_username IN bbs.username%TYPE,
  v_email IN bbs.email%TYPE,
  v_title IN bbs.title%TYPE,
  v_contents IN bbs.contents%TYPE
)
IS
BEGIN
  INSERT INTO BBS(idx, username, title, contents, email, readnum, writeday)
  VALUES(bbs_seq.NEXTVAL, v_username, v_title, v_contents, v_email, 0, SYSDATE);
END;
```

7)POST 발송시 한글 깨짐 처리하기
-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)Postman test

```
POST : http://localhost:8080/biz/bbs
BODY > raw > JSON(application/json)
{
  "username" : "박지민",
  "email" : "",
  "title" : "게시판 타이틀",
  "contents" : "이것은 게시판 내용입니다."
}

{
  "code": "success"
}
```

9)http://localhost:8080/biz/static/write.html에서 게시판 글쓰기 테스트

31. 게시판 한 개의 글 선택하기 및 조회수 증가하기

1)MainController.java

```
@RequestMapping(value="/bbs/{idx}", method=RequestMethod.GET)
@ResponseBody
public Map bbs(@PathVariable int idx) {
  Map<String, Object> map = new HashMap<String, Object>();
  map.put("idx", idx);
  this.bbsService.read(map);
  List<Object> array = (List<Object>)map.get("result");
  BbsVO bbsVo = (BbsVO)array.get(0);
  map.remove("idx");
  map.remove("result");
  map.put("code", "success");
  map.put("data", bbsVo);
  return map;
}
```

2)BbsServiceImpl.java

```
@Override
```

```

2044     public void read(Map map) {
2045         this.bbsDao.select(map);
2046     }

```

3)BbsDaoImpl.java

```

2049     @Override
2050     public void select(Map map) {
2051         this.sqlSession.selectOne("select", map);
2052     }

```

4)bbs-mapper.xml

```

2056     <parameterMap type="bbsVo" id="selectParameterMap">
2057         <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER"
2058             mode="IN"/>
2059         <parameter property="result" javaType="ResultSet" jdbcType="CURSOR"
2060             mode="OUT" resultMap="BbsResultMap"/>
2061     </parameterMap>
2062     <select id="select" parameterMap="selectParameterMap" statementType="CALLABLE">
2063         { call sp_bbs_select(?, ?) }
2064     </select>

```

5)Stored PROCEDURE

```

2068     CREATE OR REPLACE PROCEDURE sp_bbs_select
2069     (
2070         v_idx      IN      bbs.idx%TYPE,
2071         bbs_record OUT    SYS_REFCURSOR
2072     )
2073     AS
2074     BEGIN
2075         OPEN bbs_record FOR
2076         SELECT idx, username, email, title, writeday, readnum, contents
2077         FROM BBS
2078         WHERE idx = v_idx;
2079
2080         UPDATE BBS SET readnum = readnum + 1
2081         WHERE idx = v_idx;
2082     END;

```

6)view.html

```

2086     <!DOCTYPE html>
2087     <html lang="en">
2088     <head>
2089         <meta charset="UTF-8">
2090         <meta name="viewport" content="width=device-width, initial-scale=1.0">
2091         <title>글 읽기</title>
2092         <link href="css/style.css" rel="stylesheet" type="text/css">
2093         <script src="js/jquery-3.5.1.min.js"></script>
2094         <script>
2095             $(function(){
2096                 var idx = location.search.substring(1).split("=")[1];
2097                 $.ajax({
2098                     url : '/biz/bbs/' + idx,
2099                     type : 'GET',
2100                     dataType : 'json',
2101                     success : function(data){
2102                         mydata = data.data;
2103                         var title = mydata.title;
2104                         title = title.replace(/"/g, "");
2105                         title = title.replace(/&lt;/g, '<');
2106                         title = title.replace(/&gt;/g, '>');
2107                         $('#title').text(title);
2108
2109                         $('#username').text(mydata.username);

```



```

2111         var writeday = new Date(mydata.writeday);
2112         $('#writeday').text(writeday.toLocaleDateString());
2113         $('#readnum').text(mydata.readnum);
2114
2115         var contents = mydata.contents;
2116         contents = contents.replace(/"/g, "");
2117         contents = contents.replace(/&lt;/g, '<');
2118         contents = contents.replace(/&gt;/g, '>');
2119         $('#contents').html(contents);
2120
2121         if(mydata.email){
2122             var email = "<a href='mailto:" + mydata.email + "'>" + mydata.email +
                "</a>";
2123         }else{
2124             var email = "";
2125         }
2126         $('#email').html(email);
2127     }
2128     });
2129 });
2130 </script>
2131 </head>
2132 <body>
2133 <h1 style="text-align:center">게시글 읽기</h1>
2134 <table width="600" style="margin:auto" cellspacing="0" cellpadding="2">
2135     <tr>
2136         <td height="22">&nbsp;</td>
2137     </tr>
2138     <tr>
2139         <td height="1" bgcolor="#1F4F8F"></td>
2140     </tr>
2141     <tr bgcolor="#DFEDFF">
2142         <td bgcolor="#DFEDFF">
2143             <div><strong id="title"></strong></div>
2144         </td>
2145     </tr>
2146     <tr bgcolor="#FFFFFF">
2147         <td bgcolor="#F4F4F4">
2148             <table width="100%" border="0" cellpadding="0" cellspacing="4" height="1">
2149                 <tr bgcolor="#F4F4F4">
2150                     <td width="13%" height="7"></td>
2151                     <td width="51%" height="7">글쓴이 : <span id="username"></span>
2152                     (<span id="email"></span>)</td>
2153
2154                     <td width="25%" height="7"></td>
2155                     <td width="11%" height="7"></td>
2156                 </tr>
2157                 <tr bgcolor="#F4F4F4">
2158                     <td width="13%"></td>
2159                     <td width="51%">작성일 : <span id='writeday'></span></td>
2160                     <td width="25%">조회수 : <span id='readnum'></span></td>
2161                     <td width="11%"></td>
2162                 </tr>
2163             </table>
2164         </td>
2165     </tr>
2166     <tr align="center">
2167         <td bgcolor="#1F4F8F" height="1"></td>
2168     </tr>
2169     <tr>
2170         <td style="padding:20 0 20 0">
2171             <br />
2172             <span style="color:#333333" id='contents'></span>
2173             <br />
2174         </td>
2175     </tr>
2176     <tr align="center">

```

```

2177         <td class="button" height="1"></td>
2178     </tr>
2179     <tr align="center">
2180         <td bgcolor="#1F4F8F" height="1"></td>
2181     </tr>
2182 </table>
2183 <table width="600" style="margin:auto" border="0" cellpadding="0" cellspacing="5">
2184     <tr>
2185         <td align="right" width="450"><a href="list.html"></a></td>
2187         <td width="70" align="right">
2188             <a href="#"></a>
2189         </td>
2190         <td width="70" align="right"></td>
2191         <td width="70" align="right"></td>
2192     </tr>
2193 </table>
2194 </body>
2195 </html>

```

7)Postman에서 테스트

GET : <http://localhost:8080/biz/bbs/2>

```

2200 {
2201     "code": "success",
2202     "data": {
2203         "idx": 2,
2204         "readnum": 1,
2205         "username": "한지민",
2206         "title": "'첫' 글",
2207         "contents": "이것은<br /><br />'첫' 글입니다.",
2208         "email": null,
2209         "writeday": 1593129238000
2210     }
2211 }

```

32. 게시판 글 삭제하기

1)MainController.java

```

2216 @RequestMapping(value="/bbs/{idx}", method=RequestMethod.DELETE)
2217 @ResponseBody
2218 public Map delete(@PathVariable int idx) {
2219     this.bbsService.delete(idx);
2220     Map<String, Object> map = new HashMap<String, Object>();
2221     map.put("code", "success");
2222     return map;
2223 }

```

2)BbsServiceImpl.java

```

2227 @Override
2228 public void delete(int idx) {
2229     this.bbsDao.delete(idx);
2230 }

```

3)BbsDaoImpl.java

```

2234 @Override
2235 public void delete(int idx) {
2236     this.sqlSession.delete("delete", idx);
2237 }

```

4)bbs-mapper.xml

```

2241 <delete id="delete" parameterType="java.lang.Integer" statementType="CALLABLE">
2242     { call sp_bbs_delete(#{idx}) }

```

</delete>

5)Stored Procedure

```
CREATE OR REPLACE PROCEDURE sp_bbs_delete
(
    v_idx      IN      bbs.idx%TYPE
)
IS
BEGIN
    DELETE FROM BBS
    WHERE idx = v_idx;
END;
```

6)Postman에서 test

DELETE : http://localhost:8080/biz/bbs/3

```
{
  "code": "success"
}
```

7)view.html 코드 추가후 테스트

```
$('#delBtn').bind("click", function(){
    $.ajax({
        url : '/biz/bbs/' + idx,
        type : 'DELETE',
        dataType : 'json',
        success : function(data){
            //alert(data.code); //success
            location.href = 'list.html';
        }
    });
});
```

33. 게시판 글 수정하기

1)MainController.java

```
@RequestMapping(value="/bbs", method=RequestMethod.PUT)
@ResponseBody
public Map update(@RequestBody BbsVO bbsVo) {
    String title = this.convert(bbsVo.getTitle());
    String contents = this.convert(bbsVo.getContents());
    bbsVo.setTitle(title);
    bbsVo.setContents(contents);
    this.bbsService.update(bbsVo);
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("code", "success");
    return map;
}
```

2)BbsServiceImpl.java

```
@Override
public void update(BbsVO bbsVo) {
    this.bbsDao.update(bbsVo);
}
```

3)BbsDaoImpl.java

```
@Override
public void update(BbsVO bbsVo) {
    this.sqlSession.update("update", bbsVo);
}
```

4)bbs-mapper.xml

```
<parameterMap type="bbsVo" id="updateParameterMap">
  <parameter property="email" javaType="java.lang.String" jdbcType="VARCHAR"
    mode="IN"/>
  <parameter property="title" javaType="java.lang.String" jdbcType="VARCHAR" mode="IN"/>
  <parameter property="contents" javaType="java.lang.String" jdbcType="VARCHAR"
    mode="IN"/>
  <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER" mode="IN"/>
</parameterMap>
<update id="update" parameterMap="updateParameterMap" statementType="CALLABLE">
  { call sp_bbs_update(?,?,?,?) }
</update>
```

5)Stored Procedure

```
CREATE OR REPLACE PROCEDURE sp_bbs_update
(
  v_email    IN      BBS.email%TYPE,
  v_title    IN      BBS.title%TYPE,
  v_contents IN      BBS.contents%TYPE,
  v_idx      IN      BBS.idx%TYPE
)
IS
BEGIN
  UPDATE BBS SET email = v_email, title = v_title, contents = v_contents
  WHERE idx = v_idx;
END;
```

6)Postman에서 테스트

```
PUT : http://localhost:8080/biz/bbs/
{
  "email" : "aaa@aaa.com",
  "title" : "게시판 타이틀 수정입니다.",
  "contents" : "이것은 게시판 내용 수정사항입니다.",
  "idx":2
}

{
  "code": "success"
}
```

7)view.html 코드 추가

```
$('#updateBtn').bind("click", function(){
  location.href = 'update.html?idx=' + idx;
});
```

8)update.html 추가 후 테스트하기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>게시판 글 수정하기</title>
  <link href="css/style.css" rel="stylesheet" type="text/css">
  <style>
    body {
      margin-top: 50px;
    }

    .bg {
      background-color: #1F4F8F
    }

    #bg1 {
```

```

2375         background-color: #DFEDFF
2376     }
2377
2378     div {
2379         width: 600px;
2380         margin: auto;
2381     }
2382
2383     .tdbg {
2384         background-color: #f4f4f4;
2385         text-align: center;
2386     }
2387 </style>
2388 <script src="js/jquery-3.5.1.min.js"></script>
2389 <script>
2390     $(function(){
2391         var idx = location.search.substring(1).split("=")[1];
2392         $.ajax({
2393             url : '/biz/bbs/' + idx,
2394             type : 'GET',
2395             dataType : 'json',
2396             success : function(data){
2397                 var mydata = data.data;
2398                 $('#username').val(mydata.username);
2399                 if(mydata.email){
2400                     $('#email').val(mydata.email);
2401                 }else{
2402                     $('#email').val("");
2403                 }
2404                 var title = mydata.title;
2405                 title = title.replace(/"/g, "");
2406                 title = title.replace(/&lt;/g, '<');
2407                 title = title.replace(/&gt;/g, '>');
2408                 $('#title').val(title);
2409
2410                 var contents = mydata.contents;
2411                 contents = contents.replace(/"/g, "");
2412                 contents = contents.replace(/&lt;/g, '<');
2413                 contents = contents.replace(/&gt;/g, '>');
2414                 contents = contents.replace(RegExp("<br />", "g"), "\n");
2415                 $('#contents').val(contents);
2416             }
2417         });
2418
2419         $("input[type='image']").bind("click", function(){
2420             $.ajax({
2421                 url : '/biz/bbs',
2422                 type : 'PUT',
2423                 contentType : 'application/json;charset=utf-8',
2424                 data : JSON.stringify({
2425                     "email" : $('#email').val(),
2426                     "title" : $('#title').val(),
2427                     "contents" : $('#contents').val(),
2428                     "idx" : idx
2429                 }),
2430                 dataType : 'json',
2431                 success : function(data){
2432                     location.href = "view.html?idx=" + idx;
2433                 }
2434             });
2435         });
2436     });
2437 </script>
2438 </head>
2439
2440 <body>
2441     <div>

```

```
2442 <table width="600" cellspacing="0" cellpadding="2">
2443 <tr>
2444 <td colspan="2" height="1" class="bg"></td>
2445 </tr>
2446 <tr>
2447 <td colspan="2" height="20" class="notice" id="bg1">&nbsp;&nbsp;&nbsp;게시판 글 수정하기</td>
2448 </tr>
2449 <tr>
2450 <td colspan="2" height="1" class="bg"></td>
2451 </tr>
2452 <tr>
2453 <td width="124" height="30" class="tdbg">작성자</td>
2454 <td width="494" style="padding:0 0 0 10">
2455 <input type="text" name="username" class="input_style1" id="username"
2456 </td>
2457 </tr>
2458 <tr>
2459 <td width="124" height="30" class="tdbg">Email</td>
2460 <td width="494" style="padding:0 0 0 10">
2461 <input type="text" name="email" class="input_style1" id="email"></td>
2462 </tr>
2463 <tr>
2464 <td width="124" class="tdbg">제목</td>
2465 <td width="494" style="padding:0 0 0 10" height="25">
2466 <input type="text" name="title" size="60" maxlength="20" class="input_style2"
2467 </td>
2468 </tr>
2469 <tr>
2470 <td width="124" height="162" style="padding-top:7px;" class="tdbg">내용</td>
2471 <td width="494" valign="top" style="padding:5 0 5 10">
2472 <textarea cols="65" rows="10" name="contents" maxlength="2000"
2473 </td>
2474 </tr>
2475 <tr>
2476 <td colspan="2" height="1" class="button"></td>
2477 </tr>
2478 <tr>
2479 <td colspan="2" height="1" class="bg"></td>
2480 </tr>
2481 <tr>
2482 <td colspan="2" height="10"></td>
2483 </tr>
2484 <tr>
2485 <td colspan="2">
2486 <table width="100%" cellpadding="0" cellspacing="0">
2487 <tr>
2488 <td width="28%">&nbsp;&nbsp;&nbsp;</td>
2489 <td width="51%">&nbsp;&nbsp;&nbsp;</td>
2490 <td width="12%">
2491 <a href="javascript:history.back()">
2492 
2493 </a>
2494 </td>
2495 <td width="9%">
2496 <input type="image" src="images/ok.gif" width="46" height="20">
2497 </td>
2498 </tr>
2499 </table>
2500 </td>
2501 </tr>
2502 </table>
2503 </div>
2504 </body>
2505 </html>
```

```

2506 -----
2507 Task4. Spring Restful BBS(댓글게시판)
2508 1. 계속 위에서 생성한 Task4의 단순게시판을 이용한다.
2509 2. Reply Table 생성
2510
2511 CREATE TABLE Reply
2512 (
2513     replyidx    NUMBER(4),
2514     idx         NUMBER(4),
2515     replyer     VARCHAR2(20) NOT NULL,
2516     replytext   VARCHAR2(1000) NOT NULL,
2517     replydate   DATE DEFAULT SYSDATE NOT NULL,
2518     CONSTRAINTS reply_replyidx_pk PRIMARY KEY(replyidx),
2519     CONSTRAINTS reply_idx_fk FOREIGN KEY(idx) REFERENCES BBS(idx)
2520 )
2521
2522
2523 3. Reply Table용 Sequence 생성
2524
2525 CREATE SEQUENCE reply_seq
2526     START WITH 1
2527     INCREMENT BY 1
2528     MAXVALUE 9999
2529     NOCYCLE
2530
2531
2532 4. 댓글처리를 위한 객체 생성
2533     1)com.example.vo.ReplyVO.java
2534
2535         package com.example.vo;
2536
2537         import java.util.Date;
2538
2539         import lombok.Getter;
2540         import lombok.NoArgsConstructor;
2541         import lombok.Setter;
2542         import lombok.ToString;
2543
2544         @Getter
2545         @Setter
2546         @NoArgsConstructor
2547         @ToString
2548         public class ReplyVO {
2549             private int replyidx, idx;
2550             private String replyer, replytext;
2551             private Date replydate;
2552         }
2553
2554     2)com.example.dao.ReplyDao.java
2555
2556         package com.example.dao;
2557
2558         import java.util.Map;
2559
2560         import com.example.vo.ReplyVO;
2561
2562         public interface ReplyDao {
2563             void insert(ReplyVO replyVo);
2564             void select(Map map);
2565         }
2566
2567     3)com.example.dao.ReplyDaoImpl.java
2568
2569         package com.example.dao;
2570
2571         import java.util.Map;
2572

```

```

2573 import org.mybatis.spring.SqlSessionTemplate;
2574 import org.springframework.beans.factory.annotation.Autowired;
2575 import org.springframework.stereotype.Repository;
2576
2577 import com.example.vo.ReplyVO;
2578
2579 @Repository("replyDao")
2580 public class ReplyDaoImpl implements ReplyDao {
2581     @Autowired
2582     private SqlSessionTemplate sqlSession;
2583
2584     @Override
2585     public void insert(ReplyVO replyVo) {
2586         // TODO Auto-generated method stub
2587
2588     }
2589
2590     @Override
2591     public void select(Map map) {
2592         // TODO Auto-generated method stub
2593
2594     }
2595
2596 }

```

4)com.example.service.ReplyService.java

```

2599 package com.example.service;
2600
2601 import java.util.Map;
2602
2603 import com.example.vo.ReplyVO;
2604
2605 public interface ReplyService {
2606     void select(Map map);
2607     void create(ReplyVO replyVo);
2608 }
2609

```

5)com.example.service.ReplyServiceImpl.java

```

2612 package com.example.service;
2613
2614 import java.util.Map;
2615
2616 import org.springframework.beans.factory.annotation.Autowired;
2617 import org.springframework.stereotype.Service;
2618
2619 import com.example.dao.ReplyDao;
2620 import com.example.vo.ReplyVO;
2621
2622 @Service("replyService")
2623 public class ReplyServiceImpl implements ReplyService {
2624     @Autowired
2625     private ReplyDao replyDao;
2626
2627     @Override
2628     public void select(Map map) {
2629         // TODO Auto-generated method stub
2630
2631     }
2632
2633     @Override
2634     public void create(ReplyVO replyVo) {
2635         // TODO Auto-generated method stub
2636
2637     }
2638
2639 }

```


6)com.example.controller.ReplyController.java

```
package com.example.controller;

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

import com.example.service.ReplyService;

@Controller
public class ReplyController {
    @Autowired
    private ReplyService replyService;
}
```

5. 댓글 처리를 위한 MyBatis 설정 파일 생성 및 추가

1)src/main/resources/reply-mapper.xml 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="Reply">

</mapper>
```

2)src/main/webapp/WEB-INF/spring/root-context 수정

```
<property name="mapperLocations">
    <list>
        <value>classpath:bbs-mapper.xml</value>
        <value>classpath:reply-mapper.xml</value> <-- 추가
    </list>
</property>
```

3)mybatis-config.xml 코드 추가

```
<typeAlias type="com.example.vo.ReplyVO" alias="replyVo"/>
```

6. 댓글 쓰기

1)view.html 코드 추가

-다음과 같이 게시판 테이블 아래에 댓글 입력을 위한 div를 추가한다.

```
<div style="text-align:center; width:600px;margin:auto;">
    <p><label for="replyer">Nickname : </label>
        <input type="text" id="replyer" />
        <input type="button" value="댓글등록" id="replyBtn"/>
    </p>
    <p>
        <textarea rows="3" cols="60" placeholder="댓글을 달아주세요" id="replytext"></textarea>
    </p>
</div>
<div id="replylist"></div>
```

2)view.html의 \$.ajax() 안에 기존코드 다음에 다음과 같은 코드를 추가한다.

```
$('#replyBtn').bind("click", function(){
    if(!$('#replyer').val()){
        alert("댓글 작성자의 Nickname이 없습니다.");
        $('#replyer').focus();
        return false;
    }
    if(!$('#replytext').val()){
        alert("댓글의 내용이 없습니다.");
    }
});
```

```

2707         $('#replytext').focus();
2708     return false;
2709 }
2710 $.ajax({
2711     url : '/biz/reply',
2712     type : 'POST',
2713     contentType : 'application/json;charset=utf-8',
2714     data : JSON.stringify({
2715         "replyer" : $('#replyer').val(),
2716         "replytext" : $('#replytext').val(),
2717         "idx" : idx
2718     }),
2719     success : function(data){
2720         alert('댓글이 저장됐습니다.');
```

3)com.example.controller.ReplyController.java 코드 추가

```

2729     @RequestMapping(value="/reply", method=RequestMethod.POST)
2730     @ResponseBody
2731     public Map replyInsert(@RequestBody ReplyVO replyVo) {
2732         String replyer = this.convert(replyVo.getReplyer());
2733         String replytext = this.convert(replyVo.getReplytext());
2734         replyVo.setReplyer(replyer);
2735         replyVo.setReplytext(replytext);
2736         this.replyService.create(replyVo);
2737         Map<String, Object> map = new HashMap<String, Object>();
2738         map.put("code", "success");
2739         return map;
2740     }
2741
2742     private String convert(String oldStr) {
2743         String newStr = oldStr.replace("'", "");
2744         newStr = newStr.replace("<", "&lt;");
2745         newStr = newStr.replace(">", "&gt;");
2746         newStr = newStr.replace("\n", "<br />");
2747         return newStr;
2748     }
2749 
```

4)ReplyServiveImpl.java 코드 수정

```

2752     @Override
2753     public void create(ReplyVO replyVo) {
2754         this.replyDao.insert(replyVo);
2755     }
2756 
```

5)ReplyDaoImpl.java 코드 수정

```

2759     @Override
2760     public void insert(ReplyVO replyVo) {
2761         this.sqlSession.insert("Reply.replyInsert", replyVo);
2762     }
2763 
```

6)reply-mapper.xml 코드 추가

```

2766     <parameterMap type="replyVo" id="replyInsertParameterMap">
2767         <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER" mode="IN"/>
2768         <parameter property="replyer" javaType="java.lang.String" jdbcType="VARCHAR"
2769             mode="IN"/>
2770         <parameter property="replytext" javaType="java.lang.String" jdbcType="VARCHAR"
2771             mode="IN"/>
2772     </parameterMap>
2773     <insert id="replyInsert" parameterMap="replyInsertParameterMap" statementType="CALLABLE">
```

```
2772         { call sp_reply_insert(?, ?, ?) }
2773     </insert>
```

2774 7)Stored Procedure 생성

```
2776
2777     CREATE OR REPLACE PROCEDURE sp_reply_insert
2778     (
2779         v_idx          Reply.idx%TYPE,
2780         v_replyer       Reply.replyer%TYPE,
2781         v_replytext     Reply.replytext%TYPE
2782     )
2783     IS
2784     BEGIN
2785         INSERT INTO Reply(replyidx, idx, replyer, replytext, replydate)
2786         VALUES(reply_seq.NEXTVAL, v_idx, v_replyer, v_replytext, SYSDATE);
2787     END;
```

2788 8)Postman 에서 테스트

2789 POST : <http://localhost:8080/biz/reply>

```
2790
2791
2792     {
2793         "replyer" : "jimin",
2794         "replytext" : "감동입니다.",
2795         "idx" : 4
2796     }
2797
2798     {
2799         "code": "success"
2800     }
2801
2802
```

2803 9)view.html에서 입력으로 테스트하기

2804 7. 댓글 목록 보기

2805 1)ReplyController.java

```
2806
2807
2808
2809     @RequestMapping(value="/reply/{idx}", method=RequestMethod.GET)
2810     @ResponseBody
2811     public Map replylist(@PathVariable int idx) {
2812         Map<String, Object> map = new HashMap<String, Object>();
2813         map.put("idx", idx);
2814         this.replyService.select(map);
2815         List<ReplyVO> list = (List<ReplyVO>)map.get("results");
2816         map.remove("results");
2817         map.remove("idx");
2818         map.put("code", "success");
2819         map.put("data", list);
2820         return map;
2821     }
2822
```

2823 2)ReplyServiceImpl.java

```
2824
2825
2826     @Override
2827     public void select(Map map) {
2828         this.replyDao.select(map);
2829     }
2830
```

2831 3)ReplyDaoImpl.java

```
2832
2833     @Override
2834     public void select(Map map) {
2835         this.sqlSession.selectList("Reply.replySelectAll", map);
2836     }
2837
```

2838 4)reply-mapper.xml

```

2839 <resultMap type="replyVo" id="replyResultMap">
2840   <result property="replyer" javaType="java.lang.String" column="replyer"
      jdbcType="VARCHAR"/>
2841   <result property="replytext" javaType="java.lang.String" column="replytext"
      jdbcType="VARCHAR"/>
2842   <result property="replydate" javaType="java.util.Date" column="replydate"
      jdbcType="DATE"/>
2843 </resultMap>
2844 <parameterMap type="replyVo" id="replySelectParameterMap">
2845   <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER" mode="IN"/>
2846   <parameter property="results" javaType="ResultSet" jdbcType="CURSOR" mode="OUT"
      resultMap="replyResultMap"/>
2847 </parameterMap>
2848 <select id="replySelectAll" parameterMap="replySelectParameterMap"
2849   statementType="CALLABLE">
2850   { call sp_reply_select(?, ?) }
2851 </select>
2852

```

2853 5)Stored Procedure 생성

```

2854
2855 CREATE OR REPLACE PROCEDURE sp_reply_select
2856 (
2857   v_idx          IN      Reply.idx%TYPE,
2858   reply_record   OUT     SYS_REFCURSOR
2859 )
2860 AS
2861 BEGIN
2862   OPEN reply_record FOR
2863   SELECT replyer, replytext, replydate
2864   FROM Reply
2865   WHERE idx = v_idx
2866   ORDER BY replyidx ASC;
2867 END;
2868

```

2869 6)Postman에서 테스트

```

2870
2871 GET : http://localhost:8080/biz/reply/4
2872
2873 {
2874   "code": "success",
2875   "data": [
2876     {
2877       "replyidx": 0,
2878       "idx": 0,
2879       "replyer": "jimin",
2880       "replytext": "감동입니다.",
2881       "replydate": 1593132977000
2882     },
2883     {
2884       "replyidx": 0,
2885       "idx": 0,
2886       "replyer": "javasoft",
2887       "replytext": "Very Good.",
2888       "replydate": 1593133151000
2889     }
2890   ]
2891 }
2892

```

2893 7)view.html 수정

```

2894 -jQuery 영역 밖의 새로운 함수 생성
2895
2896 function listReply(idx){
2897   $.ajax({
2898     url : '/biz/reply/' + idx,
2899     type : 'GET',
2900     dataType : 'json',
2901     success : function(data){

```

```

2902         var mydata = data.data;
2903         var str = "<ul style='list-style-type:none'>";
2904         for(var i = 0 ; i < mydata.length ; i++){
2905             str += "<li>";
2906             str += "<p>작성자 : " + mydata[i].replyer + "<br />";
2907             var now = new Date(mydata[i].replydate);
2908             str += "작성날짜 : " + now.toLocaleString() + "</p>";
2909             str += "<p>" + mydata[i].replytext + "</p>";
2910             str += "</li>";
2911             str += "<hr style='width:600px;' />"
2912         }
2913         str += "</ul>";
2914         $('#replylist').html(str);
2915     }
2916 });
2917 }
2918

```

-기존 코드에 위의 함수 호출하는 코드 추가

```

2920         listReply(idx); <---추가
2921
2922         $('#delBtn').bind("click", function(){
2923             $.ajax({
2924                 url : '/Bbs/gesipan/' + idx,
2925                 type : 'DELETE',
2926                 dataType : 'json',
2927                 success : function(data){
2928                     //alert(data.code); //success
2929                     location.href = 'bbslist.html';
2930                 }
2931             });
2932         });
2933     });
2934

```

-댓글 작성 후 replyer와 replytext를 clear하는 부분에 위의 함수 호출 코드 추가

```

2936
2937         success : function(data){
2938             alert('댓글이 저장됐습니다.');
```