HOL : Restful API in Spring

Task1. Using Restful API

1. In J2EE Perspective
2. Project Explorer > right-click > New > Dynamic Web Project
3. Project name : RestfulDemo > Next > Check [Generate web.xml deployment descriptor] > Finish
4. Convert to Maven Project
   project right-click > Configure > Convert to Maven Project > Finish

5. Add Spring Project Nature
   project right-click > Spring > Add Spring Project Nature

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

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.0.RELEASE</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.2.0.RELEASE</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.2.0.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <dependency>
        <groupId>com.oracle</groupId>
        <artifactId>ojdbc6</artifactId>
        <version>11.2</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.10.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <version>2.10.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-annotations</artifactId>
        <version>2.10.0</version>
    </dependency>
```

```
      <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
      <dependency>
         <groupId>org.mybatis</groupId>
         <artifactId>mybatis-spring</artifactId>
         <version>2.0.3</version>
      </dependency>
      <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
      <dependency>
         <groupId>org.mybatis</groupId>
         <artifactId>mybatis</artifactId>
         <version>3.5.3</version>
      </dependency>
   </dependencies>

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


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


8. config Folder에 applicationContext.xml file 생성
   1)config Folder > right-click > New > Other > Spring > Spring Bean Configuration File
   2)File name :  applicationContext.xml
   3)생성시 beans,context, mvc check
      <?xml version="1.0" encoding="UTF-8"?>
      <beans xmlns="http://www.springframework.org/schema/beans"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:context="http://www.springframework.org/schema/context"
         xmlns:mvc="http://www.springframework.org/schema/mvc"
         xsi:schemaLocation="http://www.springframework.org/schema/mvc
         http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
            http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-4.3.xsd">

      </beans>


9. ContextLoaderListener class 설정
   1)Business logic의 Spring 설정 file (ex:applicationContext.xml)을 작성했기 때문에 listener로 ContextLoaderListener class를 정의해야 한다.
   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.
   3)즉, ContextLoaderListener class는 DispatcherServlet class의 load보다 먼저 동작하여 business logic층을 정의한 Spring 설정 file을 load한다.
   4)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입
      <!-- needed for ContextLoaderListener -->
      <context-param>
         <param-name>contextConfigLocation</param-name>
         <param-value>location</param-value>
      </context-param>

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

   5)아래 code로 변환
      <context-param>
```

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

10. DispatcherServlet Class 추가

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

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

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

2)아래의 code로 변환

```
    <servlet>
        <servlet-name>springDispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:beans.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

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

11. MemberVO class 생성

1)src/com.example.vo package 생성

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

```
    package com.example.vo;

    public class MemberVO {
        private String name, userid, gender, city;
        public MemberVO() {}
        public MemberVO(String name, String userid, String gender, String city) {
            this.name = name;
            this.userid = userid;
            this.gender = gender;
            this.city = city;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getUserid() {
            return userid;
```

```
    }
    public void setUserid(String userid) {
        this.userid = userid;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    @Override
    public String toString() {
        return "MemberVO [name=" + name + ", userid=" + userid + ", gender=" + gender + ",
        city=" + city + "]";
    }
}
```

12. MemberDao 객체 생성

1)src/com.example.dao package 생성

2)src/com.example.dao.MemberDao interface

```
package com.example.dao;

import java.util.List;

import com.example.vo.MemberVO;

public interface MemberDao {
    void create(MemberVO member);
    List<MemberVO> readAll();
    MemberVO read(String userid);
    void update(MemberVO member);
    void delete(String userid);
}
```

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

```
package com.example.dao;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.apache.ibatis.session.SqlSession;

import com.example.vo.MemberVO;

@Repository("memberDao")
public class MemberDaoImpl implements MemberDao {
    @Autowired
    private SqlSession sqlSession;

    @Override
    public void create(MemberVO member) {

    }

    @Override
    public List<MemberVO> readAll() {
        return null;
```

```
    }

    @Override
    public MemberVO read(String userid) {
        return null;
    }

    @Override
    public void update(MemberVO member) {

    }

    @Override
    public void delete(String userid) {

    }
}
```

13. MemberService 객체 생성

1)src/com.example.service package 생성

2)src/com.example.service.MemberService interface

```
package com.example.service;

import java.util.List;

import com.example.vo.MemberVO;

public interface MemberService {
    void insertMember(MemberVO member);
    List<MemberVO> select();
    MemberVO selectMember(String userid);
    void updateMember(MemberVO member);
    void deleteMember(String userid);
}
```

3)src/com.example.service.MemberServiceImpl.java

```
package com.example.service;

import java.util.List;

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

import com.example.dao.MemberDao;
import com.example.vo.MemberVO;

@Service("memberService")
public class MemberServiceImpl implements MemberService {
    @Autowired
    private MemberDao memberDao;

    @Override
    public void insertMember(MemberVO member) {

    }

    @Override
    public List<MemberVO> select() {
        return null;
    }

    @Override
    public MemberVO selectMember(String userid) {
        return null;
```

```
        }

        @Override
        public void updateMember(MemberVO member) {

        }

        @Override
        public void deleteMember(String userid) {

        }
    }
```

14. HomeController 객체 생성
    1)src/com.example.controller package 생성
    2)com.example.controller.HomeController class 생성

```
    package com.example.controller;

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

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

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

    }
```

15. config/dbinfo.properties file 생성

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

16. applicationContext.xml

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

17. config > right-click > New > Other > Spring > Spring Bean configuration File >
    1)File name : beans.xml
    2)mvc, context check

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

18. config/mybatis-config.xml

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

```
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
  <typeAliases>
    <typeAlias type="com.example.vo.MemberVO" alias="memberVO"/>
  </typeAliases>
</configuration>
```

19. config/member-mapper.xml

```
<?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="Member">

</mapper>
```

20. applicationContext.xml 아래 code 추가

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="configLocation" value="classpath:mybatis-config.xml" />
  <property name="mapperLocations">
    <list>
      <value>classpath:member-mapper.xml</value>
    </list>
  </property>
</bean>
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg ref="sqlSessionFactory" />
</bean>
```

21. 전체 사용자 조회하기

1)HomeController 객체 code 추가

```
@RequestMapping(value = "/members", method = RequestMethod.GET)
@ResponseBody
public Map members() {
  List<MemberVO> list = this.service.select();
  Map<String, Object> map = new HashMap<String, Object>();
  map.put("code", "success");
  map.put("data", list);
  return map;
}
```

2)mybatis-mapper.xml

```
<resultMap type="memberVO" id="selectMap">
  <result property="name" column="name"/>
  <result property="userid" column="userid"/>
  <result property="gender" column="gender" />
  <result property="city" column="city"/>
</resultMap>

<select id="select" resultMap="selectMap">
  SELECT * FROM Member
</select>
```

3)MemberDaoImpl.java

```
@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>";
```

```
                        str += "</tr>";
                    }
                    $("#result").html(str);
                }
            });
        });
    </script>
</head>
<body>
    <h1>Member List</h1>
    <div style="text-align:center">
        <a href="register.html">Member Add</a>
    </div>
    <table border="1">
        <thead>
            <tr>
                <th>아이디</th><th>이름</th>
                <th>성별</th><th>거주지</th>
            </tr>
        </thead>
        <tbody id="result">
        </tbody>
    </table>
</body>
</html>
```

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

22. 특정 사용자 조회하기

1)HomeController.java

```
@RequestMapping(value = "/members/{userid}", method = RequestMethod.GET)
@ResponseBody
public Map memberInfo(@PathVariable String userid) {
    //System.out.println("userid = " + userid);
    MemberVO member = this.service.selectMember(userid);
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("code", "success");
    map.put("data", member);
    return map;
}
```

2)mybatis-mapper.xml

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

3)MemberDaoImpl.java

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

4)MemberServiceImpl.java

```
@Override
public MemberVO selectMember(String userid) {
    return this.memberDao.read(userid);
}
```

5)Postman

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

Body

```
        {
           "code": "success",
           "data": {
              "userId": "jimin",
              "name": "한지민",
              "gender": "여",
              "city": "서울"
           }
        }
```

6)WebContent/view.html

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>회원 정보 페이지</title>
      <script src="js/jquery-1.12.4.js"></script>
      <script>
         var userid = null;

         $(function(){
            userid = location.search.substring(1).split("=")[1];
            $.ajax({
               url : "/RestfulDemo/members/" + userid,
               type : "GET",
               success : function(data){
                  var member = data.data;
                  $("#userid").text(member.userid);
                  $("#name").text(member.name);
                  $("#gender").text(member.gender);
                  $("#city").text(member.city);
               }
            });
         });
      </script>
   </head>
   <body>
      <h1>Member Information</h1>
      <ul>
         <li>아이디 : <span id="userid"></span></li>
         <li>이름 : <span id="name"></span></li>
         <li>성별 : <span id="gender"></span></li>
         <li>거주지 : <span id="city"></span></li>
      </ul>
      <a href = "javascript:void(0)" onclick="javascript:history.back();">목록으로</a>
   </body>
</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

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

2)mabatis-mapper.xml

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

3)MemberDaoImpl.java

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

4)MemberServiceImpl.java

```
    @Override
    public void insertMember(MemberVO member) {
      this.memberDao.create(member);
    }
```

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

```
    <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>
```

6)Postman

POST http://localhost:8080/RestfulDemo/members

Body > raw > JSON(application/json)

```
    {
      "userid" : "girlsage",
      "name" : "소녀시대",
      "gender" : "여성",
      "city" : "수원"
    }
```

Send 버튼 클릭하면

Body

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

7)WebContent/register.html

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Member Add</title>
        <script src="js/jquery-1.12.4.js"></script>
        <script>
          $(function(){
            $("input[type='button']").bind("click", function(){
              $.ajax({
                url : "/RestfulDemo/members",
                contentType : "application/json;charset=utf-8",
                type : "POST",
                data : JSON.stringify({
```

```
                        "userid" : $("#userid").val(),
                        "name" : $("#name").val(),
                        "gender" : $(".gender:checked").val(),
                        "city" : $("#city").val()
                     }),
                     dataType : "json",
                     success : function(data){
                        alert(data.code);
                        location.href = "/RestfulDemo/";
                     }
                  });
               });
            });
         </script>
      </head>
      <body>
         <h1>Member Add</h1>
         <ul>
            <li>Name : <input type="text" id="name" /></li>
            <li>ID : <input type="text" id="userid" /></li>
            <li>Gender :
               <input type="radio" class="gender" name="gender" value="남성">남성  
               <input type="radio" class="gender" name="gender" value="여성">여성
            </li>
            <li>City : <input type="text" id="city" /></li>
         </ul>
         <input type="button" value="가입하기" />
      </body>
   </html>
```

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  " +
           "<input type='radio' class='gender' value='여성' />여성";
      }else{
         str = "<input type='radio' class='gender' name='gender' value='남성'
         />남성  " +
           "<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) {
```

```
        this.service.deleteMember(userid);
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("code", "success");
        return map;
    }

2)mabatis-mapper.xml

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

3)MemberDaoImpl.java

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

4)MemberServiceImpl.java

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

5)Postman
    DELETE http://localhost:8080/RestfulDemo/members/girlsage

    Send button click하면

        Body
            {"code": "success"}

6)WebContent/view.html 수정
    -아래의 code를 추가한다.

        <a href = "javascript:void(0)" onclick="javascript:member_delete()">삭제하기</a>

        function member_delete(){
            $.ajax({
                url : "/RestfulDemo/members/" + userid,
                type : "DELETE",
                success : function(data){
                    alert(data.code);
                    location.href = "/RestfulDemo/";
                }
            });
        }


--------------------------------------------------------------------------------
Task2. Using Restful API with XML
1. Maven Repository에서 'spring oxm'로 검색
2. Spring Object/XML Marshalling에서 4.3.24.RELEASE 선택
3. pom.xml에 아래 dependency 추가 > Maven Clean > Mavan Install

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


4. Maven Repository에서 'jaxb'로 검색, Jaxb Api에서 2.3.1
```

5. 아래의 dependency를 pom.xml에 추가 > Maven Clean > Mavan Install

```
<dependency>
   <groupId>javax.xml.bind</groupId>
   <artifactId>jaxb-api</artifactId>
   <version>2.3.1</version>
</dependency>
```

6. src/com.example.vo/MemberListVO.java 생성

```
package com.example.vo;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

import org.springframework.stereotype.Component;

@XmlRootElement(name="memberList")
@XmlAccessorType(XmlAccessType.FIELD)
@Component
public class MemberListVO {
   @XmlElement(name="member")
   private List<MemberVO> memberList;

   public List<MemberVO> getUserList() {
      return memberList;
   }

   public void setUserList(List<MemberVO> memberList) {
      this.memberList = memberList;
   }
}
```

1)XML 문서는 반드시 단 하나의 root element를 가져야 한다.
2)여러 UserVO를 표현하려면 root element로 사용할 또 다른 Java class가 필요하다.
3)새로 생성한 MemberListVO객체는 이 객체가 root element에 해당하는 객체이며, root element 이름을 memberList로 설정하겠다는 의미로 @XmlRootElement(name="memberList") 설정을 추가했다.
4)그리고 memberList 변수 위에도 @XmlElement(name="member")를 추가했는데, MemberVO 마다 element 이름을 member로 변경할 것이다.

7. src/com.example.vo.MemberVO.java 수정

```
package com.example.vo;

import javax.xml.bind.annotation.XmlAccessType;
mport javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;

import org.springframework.stereotype.Component;

@XmlRootElement(name="member")
@XmlAccessorType(XmlAccessType.FIELD)
@Component
public class MemberVO {
   @XmlAttribute
   private String userid;
   private String name, gender, city;

   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

2. Select Spring MVC Project
3. Project name : SpringRestfulBbs > Next
4. Enter a topLevelPackage : com.example.biz > Finish

5. Version check
   1)Project > right-click > Propeties
   2)Java Build Path
      -Libraries Tab> JRE System Library [JavaSE-1.6] select
      -Edit button click > [Workspace default JRE (jdk1.8.0_251) select
      -Finish > Apply click
   3)Java Compiler
      -Compiler compliance level : 1.6 -> 1.8로 변환
   4)Project Facets
      -Java > 1.8로 변환
      -Runtimes tab > Apache Tomcat v9.0 check
   5)Apply and Close
   6)혹시 [Compiler Settings Changed]창이 나오면 [Yes] click.

6. pom.xml 수정하기
```
<properties>
   <java-version>1.8</java-version>
   <org.springframework-version>5.2.7.RELEASE</org.springframework-version>
   <org.aspectj-version>1.9.5</org.aspectj-version>
   <org.slf4j-version>1.7.30</org.slf4j-version>
</properties>
   ...
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>javax.servlet-api</artifactId>
   <version>4.0.1</version>
   <scope>provided</scope>
</dependency>
<dependency>
   <groupId>javax.servlet.jsp</groupId>
   <artifactId>javax.servlet.jsp-api</artifactId>
   <version>2.3.3</version>
   <scope>provided</scope>
</dependency>
<dependency>
   <groupId>org.junit.jupiter</groupId>
   <artifactId>junit-jupiter-api</artifactId>
   <version>5.6.2</version>
   <scope>test</scope>
</dependency>
```

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

8. Database 작업
   1)bbs 계정 생성

```
ALTER SESSION SET "_ORACLE_SCRIPT"=true;

CREATE USER bbs IDENTIFIED BY bbs
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;

ALTER USER bbs
DEFAULT TABLESPACE USERS
QUOTA UNLIMITED ON USERS;

GRANT resource, connect TO bbs;
conn bbs/bbs
```

2)Create Table

```
CREATE TABLE Bbs
(
  idx        NUMBER(4),
  username    VARCHAR2(20)    NOT NULL,
  title       VARCHAR2(50)    NOT NULL,
  contents    VARCHAR2(1000)  NOT NULL,
  email       VARCHAR2(100),
  readnum     NUMBER(4)       NOT NULL,
  writeday    DATE            NOT NULL,
  CONSTRAINT bbs_idx_pk PRIMARY KEY(idx)
)
```

3)Create Sequence

```
CREATE SEQUENCE bbs_seq
  START WITH 1
  INCREMENT BY 1
  MAXVALUE  9999
  NOCYCLE;
```

9. src/main/webapp/static folder 생성

1)src/main/webapp/static/css folder

-bootstrap-theme.min.css

-bootstrap.min.css

-style.css

```
/* CSS Document */
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
    font-size:12px;
}
```

2)src/main/webapp/static/fonts folder

-glyphicons-halflings-regular.eot

-glyphicons-halflings-regular.svg

-glyphicons-halflings-regular.ttf

-glyphicons-halflings-regular.woff

-glyphicons-halflings-regular.woff2

3)src/main/webapp/static/images folder

4)src/main/webapp/static/js folder

-jquery-3.5.1.min.js

-bootstrap.min.js

5)src/main/webapp/static/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to example.com</title>
    <link rel="stylesheet" href="static/css/bootstrap.min.css">
</head>
<body>
    <div class="jumbotron">
        <h1>Hello, world!</h1>
        <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>
        <p><a class="btn btn-primary btn-lg" href="static/list.html" role="button">BBS</a></p>
```

```
        </div>
    </body>
    </html>


10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
   <resources mapping="/static/**" location="/static/" /> 추가
   <context:component-scan base-package="com.example" /> 수정


11. web.xml 수정하기
   1)다음 코드 추가

      <welcome-file-list>
         <welcome-file>/static/index.html</welcome-file>
      </welcome-file-list>

   2)서버 시작
      -Project > right-click > Run on Server

         http://localhost:8080/biz/ --> index.html로 연결 확인


12. src/main/resources/oracle.properties
   db.driverClass=oracle.jdbc.driver.OracleDriver
   db.url=jdbc:oracle:thin:@localhost:1521:ORCL
   db.username=bbs
   db.password=bbs


13. Spring jdbc 설치하기
   1)mvnrepository.com에서 'spring jdbc'로 검색하여 아래 코드 pom.xml에 추가한다.

      <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-jdbc</artifactId>
         <version>5.2.7.RELEASE</version>
      </dependency>

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


14. HikariCP 설치하기
   1)HikariCP를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

      <!-- https://mvnrepository.com/artifact/com.zaxxer/HikariCP -->
      <dependency>
         <groupId>com.zaxxer</groupId>
         <artifactId>HikariCP</artifactId>
         <version>3.4.5</version>
      </dependency>

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


15. Oracle Jdbc Driver library 검색 및 설치
   1)지난번에 설치한 Oracle Driver를 등록한다.

      <dependency>
         <groupId>com.oracle</groupId>
         <artifactId>ojdbc8</artifactId>
         <version>12.2</version>
      </dependency>
```

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

16. MyBatis 설치하기
1)mvnrepository.com에서 'MyBatis'로 검색

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.5</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.0.5</version>
</dependency>
```

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

17. Lombok library 설치하기
1)mvnrepository.com에서 'lombok'로 검색

```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.12</version>
    <scope>provided</scope>
</dependency>
```

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:property-placeholder location="classpath:oracle.properties"/>

    <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource">
        <property name="driverClassName" value="${db.driverClass}" />
        <property name="jdbcUrl" value="${db.url}" />
        <property name="username" value="${db.username}" />
        <property name="password" value="${db.password}" />
        <property name="maximumPoolSize" value="30" />
        <property name="minimumIdle" value="5" />
        <property name="autoCommit" value="true" />
    </bean>

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="configLocation" value="classpath:mybatis-config.xml" />
        <property name="mapperLocations">
            <list>
```

```
                <value>classpath:bbs-mapper.xml</value>
            </list>
        </property>
    </bean>
    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg index="0" ref="sqlSessionFactory" />
    </bean>
</beans>
```

19. MyBatis 관련 xml 파일 설정하기

1)src/main/resources

-mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
    PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <typeAliases>
        <typeAlias type="com.example.vo.BbsVO" alias="bbsVo"/>
    </typeAliases>
</configuration>
```

-bbs-mapper.xml

```
<?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="com.example.vo.BbsVO">

    <select id="currentDate" resultType="java.util.Date">
        SELECT SYSDATE FROM dual
    </select>

</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

```
package com.example.vo;

import java.util.Date;

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

@Setter
@Getter
@NoArgsConstructor
@ToString
public class BbsVO {
    private int idx, readnum;
    private String username, title, contents, email;
    private Date writeday;
}
```

22. Project 실행
1)Project > right-click > Run On Server

http://localhost:8080/biz/

2)현재 index.html이 문제없이 rendering되어야 한다.

23. Spring TestContext Framework 설치
1)mvnrepository.com에서 'spring test'로 검색하여 아래의 dependency를 설치한다.

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-test</artifactId>
   <version>5.2.7.RELEASE</version>
   <scope>test</scope>
</dependency>
```

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

24. Spring Test로 검사하기
1)src/test/java/com.example.biz/TestApp class 생성
-com.example.biz > right-click > New > JUnit Test Case
-Select [New JUnit Jupiter test]
-Name : TestApp
-Finish

```
package com.example.biz;

import java.util.Date;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit.jupiter.SpringExtension;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@ExtendWith(SpringExtension.class)
@ContextConfiguration(locations =
{"file:src/main/webapp/WEB-INF/spring/root-context.xml"})
class TestApp {
   @Autowired
   private SqlSessionTemplate sqlSession;

   @Test
   void test() {
      Date today = (Date)this.sqlSession.selectOne("currentDate");
      log.info("Today is " + today);
   }

}
```

2)Run as > JUnit Test > Green bar

```
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : com.example.biz.TestApp - Today is Thu Jun 25 12:44:06 KST 2020
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

25. com/example.dao

1)BbsDao interface

```
package com.example.dao;

import java.util.Map;

import com.example.vo.BbsVO;

public interface BbsDao {
   void insert(BbsVO bbsVo);
   void select(Map map);
   void selectAll(Map map);
   void update(BbsVO bbsVo);
   void delete(int idx);
}
```

2)BbsDaoImpl.java

```
package com.example.dao;

import java.util.Map;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.example.vo.BbsVO;

@Repository("bbsDao")
public class BbsDaoImpl implements BbsDao {
   @Autowired
   private SqlSessionTemplate sqlSession;

   @Override
   public void insert(BbsVO bbsVo) {
   }

   @Override
   public void select(Map map) {
   }

   @Override
   public void selectAll(Map map) {
   }

   @Override
   public void update(BbsVO bbsVo) {
   }

   @Override
   public void delete(int idx) {
   }
}
```

26. com.example.service

1)BbsService interface

```
package com.example.service;

import java.util.Map;

import com.example.vo.BbsVO;

public interface BbsService {
   void create(BbsVO bbsVo);
   void read(Map map);
   void readAll(Map map);
```

```
        void update(BbsVO bbsVo);
        void delete(int idx);
    }
```

2)BbsServiceImpl.java

```
    package com.example.service;

    import java.util.Map;

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

    import com.example.dao.BbsDao;
    import com.example.vo.BbsVO;

    @Service("bbsService")
    public class BbsServiceImpl implements BbsService {
        @Autowired
        private BbsDao bbsDao;

        @Override
        public void create(BbsVO bbsVo) {
        }

        @Override
        public void read(Map map) {
        }

        @Override
        public void readAll(Map map) {
        }

        @Override
        public void update(BbsVO bbsVo) {
        }

        @Override
        public void delete(int idx) {
        }
    }
```

27. com.example.controller

1)MainController.java

```
    package com.example.controller;

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

    import com.example.service.BbsService;

    import lombok.extern.slf4j.Slf4j;

    @Slf4j
    @Controller
    public class MainController {
        @Autowired
        private BbsService bbsService;
    }
```

28. Jackson Library 추가하기

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

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

```
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.11.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <version>2.11.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-annotations</artifactId>
        <version>2.11.0</version>
    </dependency>
```

2)pom.xml에 붙여 넣고 Maven Install 하기

[INFO] BUILD SUCCESS

29. 전체 게시판 글 조회하기

1)Stored Procedure 작성하기

```
CREATE OR REPLACE PROCEDURE sp_bbs_select_all
(
  bbs_record    OUT     SYS_REFCURSOR
)
AS
BEGIN
  OPEN bbs_record FOR
  SELECT idx, username, email, title, writeday, readnum
  FROM BBS
  ORDER BY idx DESC;
END;
```

2)MainController 수정

```
@RequestMapping(value = "/bbs", method = RequestMethod.GET)
@ResponseBody
public Map list() {
    Map<String, Object> map = new HashMap<String, Object>();
    this.bbsService.readAll(map);
    map.put("code", "success");
    List<BbsVO> list = (List<BbsVO>)map.get("results");
    log.info("list = " + list);
    map.remove("results");
    map.put("data", list);
    return map;
}
```

3)BbsServiceImpl.java

```
@Override
public void readAll(Map map) {
    this.bbsDao.selectAll(map);
}
```

4)BbsDaoImpl.java

```
@Override
public void selectAll(Map map) {
    this.sqlSession.selectList("selectAll", map);
}
```

5)bbs-mapper.xml

```
    <resultMap type="bbsVo" id="BbsResultMap">
      <result property="idx" javaType="java.lang.Integer"
        column="idx" jdbcType="INTEGER" />
      <result property="username" javaType="java.lang.String"
        column="username" jdbcType="VARCHAR" />
      <result property="email" javaType="java.lang.String"
        column="email" jdbcType="VARCHAR" />
      <result property="title" javaType="java.lang.String"
        column="title" jdbcType="VARCHAR" />
      <result property="contents" javaType="java.lang.String"
        column="contents" jdbcType="VARCHAR" />
      <result property="readnum" javaType="java.lang.Integer"
        column="readnum" jdbcType="INTEGER" />
      <result property="writeday" javaType="java.util.Date"
        column="writeday" jdbcType="DATE" />
    </resultMap>

    <parameterMap type="bbsVo" id="selectAllParameterMap">
      <parameter property="results" javaType="ResultSet"
        jdbcType="CURSOR" mode="OUT" resultMap="BbsResultMap" />
    </parameterMap>
    <select id="selectAll" parameterMap="selectAllParameterMap"
      statementType="CALLABLE">
      { call sp_bbs_select_all(?) }
    </select>

  6)Postman
    GET http://localhost:8080/biz/bbs Send
    Body

    {
      "code": "success",
      "data": []
    }

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

    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>게시판</title>
      <style type="text/css">
        table{ margin:auto;border:0px solid;width:800px}
        th, td{border:0px solid}
      </style>
      <script src="js/jquery-3.5.1.min.js"></script>
      <script>
        $(function(){
          $.ajax({
            url : '/biz/bbs',
            type : 'GET',
            dataType : 'json',
            success : function(data){
              var result = data.data;
              str = "";
              if(result.length == 0){
                str = "<tr><td colspan='5' style='text-align:center'>NO Data</td></tr>";
              }else{
                for(var i = 0 ; i < result.length ; i++){
                  var mydata = result[i];
                  var title = mydata.title;
                  title = title.replace(/''/g, "'");
                  title = title.replace(/&lt;/g, "<");
                  title = title.replace(/&gt;/g, ">");
                  var writeday = new Date(mydata.writeday);
```

```
                        var idx = mydata.idx;
                        str += "<tr onmouseover='myover(this)' onmouseout='myout(this)'>";
                        str += "<td style='text-align:center'>" + idx + "</td>";
                        str += "<td style='text-align:center'><a style='text-decoration:none'
                        href='mailto:" + mydata.email +"'>" + mydata.username + "</a></td>";
                        str += "<td style='text-align:center'><a href='view.html?idx=" + idx +
                        "'>" + title + "</a></td>";
                        str += "<td style='text-align:center'>" + writeday.toLocaleDateString() +
                        "</td>";
                        str += "<td style='text-align:center'>" + mydata.readnum + "</td>";
                        str += "</tr>";
                     }
                  }
                  $("#results").html(str);
               }
            });
         });
         function myover(t){
               t.style.backgroundColor='yellow';
         }
         function myout(t){
            t.style.backgroundColor = 'white';
         }
      </script>
   </head>
   <body>
      <h1 style="text-align: center;color:#0000ff">자유 게시판</h1>
      <table>
         <tr>
            <td align="left"><a href="write.html"><img src="images/write.gif" /></a></td>
            <td align="right">()</td>
         </tr>
      </table>
      <table>
         <thead>
            <tr>
               <td colspan="5" height="1" style="background-color:#1F4F8F"></td>
            </tr>
             <tr style="background-color:#87E8FF">
                <th width="10%">번호</th>
                <th width="15%">등록자</th>
                <th width="50%">제목</th>
                <th width="15%">날짜</th>
                <th width="10%">조회수</th>
            </tr>
            <tr style="text-align:center">
               <td colspan="5" height="1" style="background-color:#1F4F8F"></td>
            </tr>
         </thead>
         <tbody id="results">

         </tbody>
      </table>
   </body>
   </html>
```

8)index.html 수정하기

```
   <div class="jumbotron">
      <h1>Hello, world!</h1>
      <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>
      <p><a class="btn btn-primary btn-lg" href="static/list.html" role="button">BBS</a></p>
    </div>
```

30. 게시판 글 입력하기

1)src/main/webapp/static/write.html

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

      .bg {
         background-color: #1F4F8F
      }

      #bg1 {
         background-color: #DFEDFF
      }

      div {
         width: 600px;
         margin: auto;
      }

      .tdbg {
         background-color: #f4f4f4;
         text-align: center;
      }
   </style>
   <script src="js/jquery-3.5.1.min.js"></script>
   <script>
      $(function(){
         $("input[type='image']").bind("click", function(){
            if(!$("#username").val()){
               alert("이름이 빠졌습니다.");
               $("#username").focus();
               return false;
            }
            if(!$("#title").val()){
               alert("제목이 빠졌습니다.");
               $("#title").focus();
               return false;
            }
            if(!$("#contents").val()){
               alert("내용이 빠졌습니다.");
               $("#contents").focus();
               return false;
            }

            $.ajax({
               url: '/biz/bbs',
               type : 'POST',
               dataType : 'json',
               contentType : 'application/json;charset=utf-8',
               data : JSON.stringify({
                  'username' : $('#username').val(),
                  'email' : $('#email').val(),
                  'title' : $('#title').val(),
                  'contents' : $('#contents').val()
               }),
               success : function(data){
                  //alert(data.code);   //success
                  location.href = "/biz/static/list.html";
               }
```

```
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">  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"
                    id="title"></td>
1881             </tr>
1882             <tr>
1883               <td width="124" height="162" style="padding-top:7px;" class="tdbg">내용</td>
1884               <td width="494" valign="top" style="padding:5 0 5 10">
1885                 <textarea cols="65" rows="10" name="contents" maxlength="2000"
                    class="textarea_style1" id="contents"></textarea>
1886               </td>
1887             </tr>
1888             <tr>
1889               <td colspan="2" height="1" class='button'></td>
1890             </tr>
1891             <tr>
1892               <td colspan="2" height="1" class="bg"></td>
1893             </tr>
1894             <tr>
1895               <td colspan="2" height="10"></td>
1896             </tr>
1897             <tr>
1898               <td colspan="2">
1899                 <table width="100%" cellpadding="0" cellspacing="0">
1900                   <tr>
1901                     <td width="28%"> </td>
1902                     <td width="51%"> </td>
1903                     <td width="12%">
1904                      <a href="javascript:history.back()">
1905                        <img src="images/cancle.gif" width="46" height="20">
1906                      </a>
1907                      </td>
1908                     <td width="9%">
1909                      <input type="image" src="images/ok.gif" width="46" height="20">
1910                      </td>
1911                   </tr>
1912                 </table>
1913               </td>
```

```
            </tr>
          </table>
          </div>
        </body>
        </html>

2)com.example.biz/MainController.java

    @RequestMapping(value = "/bbs", method = RequestMethod.POST)
    @ResponseBody
    public Map insert(@RequestBody BbsVO bbsVo) {
        String username = this.convert(bbsVo.getUsername());
        String email = this.convert(bbsVo.getEmail());
        String title = this.convert(bbsVo.getTitle());
        String contents = this.convert(bbsVo.getContents());
        bbsVo.setUsername(username);
        bbsVo.setEmail(email);
        bbsVo.setTitle(title);
        bbsVo.setContents(contents);
        //System.out.println(bbsVo);
        this.bbsService.create(bbsVo);
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("code", "success");
        return map;
    }

    private String convert(String oldStr) {
        String newStr = oldStr.replace("'", "''");
        newStr = newStr.replace("<", "&lt;");
        newStr = newStr.replace(">", "&gt;");
        newStr = newStr.replace("\n", "<br />");
        return newStr;
    }

3)com.example.service/BbsServiceImpl.java

    @Override
    public void create(BbsVO bbsVo) {
        this.bbsDao.insert(bbsVo);
    }

4)com.example.dao.BbsDaoImpl.java

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

5)bbs-mapper.xml

    <parameterMap type="bbsVo" id="insertParameterMap">
        <parameter property="username" javaType="java.lang.String" jdbcType="VARCHAR"
        mode="IN" />
        <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" />
    </parameterMap>
    <insert id="insert" parameterMap="insertParameterMap" parameterType="bbsVo"
    statementType="CALLABLE">
        { call sp_bbs_insert(?, ?, ?, ?) }
    </insert>

6)Store Procedure 생성

    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
```

```
    public void read(Map map) {
        this.bbsDao.select(map);
    }

3)BbsDaoImpl.java

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

4)bbs-mapper.xml

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

5)Stored PROCEDURE

    CREATE OR REPLACE PROCEDURE sp_bbs_select
    (
      v_idx        IN      bbs.idx%TYPE,
      bbs_record    OUT     SYS_REFCURSOR
    )
    AS
    BEGIN
      OPEN bbs_record FOR
      SELECT idx, username, email, title, writeday, readnum, contents
      FROM BBS
      WHERE idx = v_idx;

      UPDATE BBS SET readnum = readnum + 1
      WHERE idx = v_idx;
    END;

6)view.html

    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>글 읽기</title>
        <link href="css/style.css" rel="stylesheet" type="text/css">
        <script src="js/jquery-3.5.1.min.js"></script>
        <script>
            $(function(){
                var idx = location.search.substring(1).split("=")[1];
                $.ajax({
                    url : '/biz/bbs/' + idx,
                    type : 'GET',
                    dataType : 'json',
                    success : function(data){
                        mydata = data.data;
                        var title = mydata.title;
                        title = title.replace(/''/g, "'");
                        title = title.replace(/&lt;/g, '<');
                        title = title.replace(/&gt;/g, '>');
                        $('#title').text(title);

                        $('#username').text(mydata.username);
```

```
                    var writeday = new Date(mydata.writeday);
                    $('#writeday').text(writeday.toLocaleDateString());
                    $('#readnum').text(mydata.readnum);

                    var contents = mydata.contents;
                    contents = contents.replace(/''/g, "'");
                    contents = contents.replace(/&lt;/g, '<');
                    contents = contents.replace(/&gt;/g, '>');
                    $('#contents').html(contents);

                    if(mydata.email){
                       var email = "<a href='mailto:" + mydata.email + "'>" + mydata.email +
                       "</a>";
                    }else{
                       var email = "";
                    }
                    $('#email').html(email);
                 }
              });
           });
        </script>
     </head>
     <body>
     <h1 style="text-align:center">게시글 읽기</h1>
        <table width="600" style="margin:auto" cellspacing="0" cellpadding="2">
           <tr>
              <td height="22"> </td>
           </tr>
           <tr>
              <td height="1" bgcolor="#1F4F8F"></td>
           </tr>
           <tr bgcolor="#DFEDFF">
              <td bgcolor="#DFEDFF">
                 <div><strong id="title"></strong></div>
              </td>
           </tr>
           <tr bgcolor="#FFFFFF">
              <td bgcolor="#F4F4F4">
                 <table width="100%" border="0" cellpadding="0" cellspacing="4" height="1">
                    <tr bgcolor="#F4F4F4">
                       <td width="13%" height="7"></td>
                       <td width="51%" height="7">글쓴이 : <span id="username"></span>
                       (<span id="email"></span>)</td>

                       <td width="25%" height="7"></td>
                       <td width="11%" height="7"></td>
                    </tr>
                    <tr bgcolor="#F4F4F4">
                       <td width="13%"></td>
                       <td width="51%">작성일 : <span id='writeday'></span></td>
                       <td width="25%">조회수 : <span id='readnum'></span></td>
                       <td width="11%"></td>
                    </tr>
                 </table>
              </td>
           </tr>
           <tr align="center">
              <td bgcolor="#1F4F8F" height="1"></td>
           </tr>
           <tr>
              <td style="padding:20 0 20 0">
                 <br />
                 <span style="color:#333333" id='contents'></span>
                <br />
              </td>
           </tr>
           <tr align="center">
```

```
            <td class="button" height="1"></td>
         </tr>
         <tr align="center">
            <td bgcolor="#1F4F8F" height="1"></td>
         </tr>
      </table>
      <table width="600" style="margin:auto" border="0" cellpadding="0" cellspacing="5">
         <tr>
            <td align="right" width="450"><a href="list.html"><img
            src="images/list.jpg"></a></td>
            <td width="70" align="right">
               <a href="#"><img src="images/reply.jpg"></a>
            </td>
            <td width="70" align="right"><img src="images/edit.jpg" id="updateBtn"></td>
            <td width="70" align="right"><img src="images/del.jpg" id="delBtn"></td>
         </tr>
      </table>
   </body>
   </html>
```

7)Postman에서 테스트

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

```
{
   "code": "success",
   "data": {
      "idx": 2,
      "readnum": 1,
      "username": "한지민",
      "title": "''첫'' 글",
      "contents": "이것은<br /><br />''첫'' 글입니다.",
      "email": null,
      "writeday": 1593129238000
   }
}
```

32. 게시판 글 삭제하기

1)MainController.java

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

2)BbsServiceImpl.java

```
@Override
public void delete(int idx) {
   this.bbsDao.delete(idx);
}
```

3)BbsDaoImpl.java

```
@Override
public void delete(int idx) {
   this.sqlSession.delete("delete", idx);
}
```

4)bbs-mapper.xml

```
<delete id="delete" parameterType="java.lang.Integer" statementType="CALLABLE">
   { 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 {
```

```
            background-color: #DFEDFF
        }

        div {
            width: 600px;
            margin: auto;
        }

        .tdbg {
            background-color: #f4f4f4;
            text-align: center;
        }
    </style>
    <script src="js/jquery-3.5.1.min.js"></script>
    <script>
        $(function(){
            var idx = location.search.substring(1).split("=")[1];
            $.ajax({
                url : '/biz/bbs/' + idx,
                type : 'GET',
                dataType : 'json',
                success : function(data){
                    var mydata = data.data;
                    $('#username').val(mydata.username);
                    if(mydata.email){
                        $('#email').val(mydata.email);
                    }else{
                        $('#email').val("");
                    }
                    var title = mydata.title;
                    title = title.replace(/''/g, "'");
                    title = title.replace(/&lt;/g, '<');
                    title = title.replace(/&gt;/g, '>');
                    $('#title').val(title);

                    var contents = mydata.contents;
                    contents = contents.replace(/''/g, "'");
                    contents = contents.replace(/&lt;/g, '<');
                    contents = contents.replace(/&gt;/g, '>');
                    contents = contents.replace(RegExp("<br />", "g"), "\n");
                    $('#contents').val(contents);
                }
            });

            $("input[type='image']").bind("click", function(){
                $.ajax({
                    url : '/biz/bbs',
                    type : 'PUT',
                    contentType : 'application/json;charset=utf-8',
                    data : JSON.stringify({
                        "email" : $('#email').val(),
                        "title" : $('#title').val(),
                        "contents" : $('#contents').val(),
                        "idx" : idx
                    }),
                    dataType : 'json',
                    success : function(data){
                        location.href = "view.html?idx=" + idx;
                    }
                });
            });
        });
    </script>
</head>

<body>
    <div>
```

```
<table width="600" cellspacing="0" cellpadding="2">
  <tr>
    <td colspan="2" height="1" class="bg"></td>
  </tr>
  <tr>
    <td colspan="2" height="20" class="notice" id="bg1">  게시판 글 수정하기</td>
  </tr>
  <tr>
    <td colspan="2" height="1" class="bg"></td>
  </tr>
  <tr>
    <td width="124" height="30" class="tdbg">작성자</td>
    <td width="494" style="padding:0 0 0 10">
      <input type="text" name="username" class="input_style1" id="username"
      readonly></td>
  </tr>
  <tr>
    <td width="124" height="30" class="tdbg">Email</td>
    <td width="494" style="padding:0 0 0 10">
      <input type="text" name="email" class="input_style1" id="email"></td>
  </tr>
  <tr>
    <td width="124" class="tdbg">제목</td>
    <td width="494" style="padding:0 0 0 10" height="25">
      <input type="text" name="title" size="60" maxlength="20" class="input_style2"
      id="title"></td>
  </tr>
  <tr>
    <td width="124" height="162" style="padding-top:7px;" class="tdbg">내용</td>
    <td width="494" valign="top" style="padding:5 0 5 10">
      <textarea cols="65" rows="10" name="contents" maxlength="2000"
      class="textarea_style1" id="contents"></textarea>
    </td>
  </tr>
  <tr>
    <td colspan="2" height="1" class='button'></td>
  </tr>
  <tr>
    <td colspan="2" height="1" class="bg"></td>
  </tr>
  <tr>
    <td colspan="2" height="10"></td>
  </tr>
  <tr>
    <td colspan="2">
      <table width="100%" cellpadding="0" cellspacing="0">
        <tr>
          <td width="28%"> </td>
          <td width="51%"> </td>
          <td width="12%">
           <a href="javascript:history.back()">
              <img src="images/cancle.gif" width="46" height="20">
           </a>
           </td>
          <td width="9%">
           <input type="image" src="images/ok.gif" width="46" height="20">
           </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</div>
</body>
</html>
```

Task4. Spring Restful BBS(댓글게시판)

1. 계속 위에서 생성한 Task4의 단순게시판을 이용한다.
2. Reply Table 생성

```
CREATE TABLE Reply
(
  replyidx      NUMBER(4),
  idx           NUMBER(4),
  replyer       VARCHAR2(20)   NOT NULL,
  replytext     VARCHAR2(1000)  NOT NULL,
  replydate     DATE   DEFAULT SYSDATE    NOT NULL,
  CONSTRAINTS reply_replyidx_pk  PRIMARY KEY(replyidx),
  CONSTRAINTS reply_idx_fk FOREIGN KEY(idx) REFERENCES BBS(idx)
)
```

3. Reply Table용 Sequence 생성

```
CREATE SEQUENCE reply_seq
  START WITH 1
  INCREMENT BY 1
  MAXVALUE  9999
  NOCYCLE
```

4. 댓글처리를 위한 객체 생성

1)com.example.vo.ReplyVO.java

```
package com.example.vo;

import java.util.Date;

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

@Getter
@Setter
@NoArgsConstructor
@ToString
public class ReplyVO {
   private int replyidx, idx;
   private String replyer, replytext;
   private Date replydate;
}
```

2)com.example.dao.ReplyDao.java

```
package com.example.dao;

import java.util.Map;

import com.example.vo.ReplyVO;

public interface ReplyDao {
   void insert(ReplyVO replyVo);
   void select(Map map);
}
```

3)com.example.dao.ReplyDaoImpl.java

```
package com.example.dao;

import java.util.Map;
```

```
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.example.vo.ReplyVO;

@Repository("replyDao")
public class ReplyDaoImpl implements ReplyDao {
   @Autowired
   private SqlSessionTemplate sqlSession;

   @Override
   public void insert(ReplyVO replyVo) {
      // TODO Auto-generated method stub

   }

   @Override
   public void select(Map map) {
      // TODO Auto-generated method stub

   }

}
```

4)com.example.service.ReplyService.java

```
package com.example.service;

import java.util.Map;

import com.example.vo.ReplyVO;

public interface ReplyService {
   void select(Map map);
   void create(ReplyVO replyVo);
}
```

5)com.example.service.ReplyServiceImpl.java

```
package com.example.service;

import java.util.Map;

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

import com.example.dao.ReplyDao;
import com.example.vo.ReplyVO;

@Service("replyService")
public class ReplyServiceImpl implements ReplyService {
   @Autowired
   private ReplyDao replyDao;

   @Override
   public void select(Map map) {
      // TODO Auto-generated method stub

   }

   @Override
   public void create(ReplyVO replyVo) {
      // TODO Auto-generated method stub

   }
}
```

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.");
```

```
            $('#replytext').focus();
            return false;
        }
        $.ajax({
            url : '/biz/reply',
            type : 'POST',
            contentType : 'application/json;charset=utf-8',
            data : JSON.stringify({
                "replyer" : $('#replyer').val(),
                "replytext" : $('#replytext').val(),
                "idx" : idx
            }),
            success : function(data){
                alert('댓글이 저장됐습니다.');
                $('#replyer').val("");
                $('#replytext').val("");
            }
        });
    });
```

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

```
    @RequestMapping(value="/reply", method=RequestMethod.POST)
    @ResponseBody
    public Map replyInsert(@RequestBody ReplyVO replyVo) {
        String replyer = this.convert(replyVo.getReplyer());
        String replytext = this.convert(replyVo.getReplytext());
        replyVo.setReplyer(replyer);
        replyVo.setReplytext(replytext);
        this.replyService.create(replyVo);
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("code", "success");
        return map;
    }

    private String convert(String oldStr) {
        String newStr = oldStr.replace("'", "''");
        newStr = newStr.replace("<", "&lt;");
        newStr = newStr.replace(">", "&gt;");
        newStr = newStr.replace("\n", "<br />");
        return newStr;
    }
```

4)ReplyServiveImpl.java 코드 수정

```
    @Override
    public void create(ReplyVO replyVo) {
        this.replyDao.insert(replyVo);
    }
```

5)ReplyDaoImpl.java 코드 수정

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

6)reply-mapper.xml 코드 추가

```
    <parameterMap type="replyVo" id="replyInsertParameterMap">
        <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER" mode="IN"/>
        <parameter property="replyer" javaType="java.lang.String" jdbcType="VARCHAR"
        mode="IN"/>
        <parameter property="replytext" javaType="java.lang.String" jdbcType="VARCHAR"
        mode="IN"/>
    </parameterMap>
    <insert id="replyInsert" parameterMap="replyInsertParameterMap" statementType="CALLABLE">
```

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

7)Stored Procedure 생성

```
    CREATE OR REPLACE PROCEDURE sp_reply_insert
    (
        v_idx          Reply.idx%TYPE,
        v_replyer      Reply.replyer%TYPE,
        v_replytext    Reply.replytext%TYPE
    )
    IS
    BEGIN
      INSERT INTO Reply(replyidx, idx, replyer, replytext, replydate)
      VALUES(reply_seq.NEXTVAL, v_idx, v_replyer, v_replytext, SYSDATE);
    END;
```

8)Postman 에서 테스트

POST : http://localhost:8080/biz/reply

```
    {
        "replyer" : "jimin",
        "replytext" : "감동입니다.",
        "idx" : 4
    }

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

9)view.html에서 입력으로 테스트하기

7. 댓글 목록 보기

1)ReplyController.java

```
    @RequestMapping(value="/reply/{idx}", method=RequestMethod.GET)
    @ResponseBody
    public Map replylist(@PathVariable int idx) {
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("idx", idx);
        this.replyService.select(map);
        List<ReplyVO> list = (List<ReplyVO>)map.get("results");
        map.remove("results");
        map.remove("idx");
        map.put("code", "success");
        map.put("data", list);
        return map;
    }
```

2)ReplyServiceImpl.java

```
    @Override
    public void select(Map map) {
        this.replyDao.select(map);
    }
```

3)ReplyDaoImpl.java

```
    @Override
    public void select(Map map) {
        this.sqlSession.selectList("Reply.replySelectAll", map);
    }
```

4)reply-mapper.xml

```
    <resultMap type="replyVo" id="replyResultMap">
      <result property="replyer" javaType="java.lang.String" column="replyer"
      jdbcType="VARCHAR"/>
      <result property="replytext" javaType="java.lang.String" column="replytext"
      jdbcType="VARCHAR"/>
      <result property="replydate" javaType="java.util.Date" column="replydate"
      jdbcType="DATE"/>
    </resultMap>
    <parameterMap type="replyVo" id="replySelectParameterMap">
      <parameter property="idx" javaType="java.lang.Integer" jdbcType="INTEGER" mode="IN"/>
      <parameter property="results" javaType="ResultSet" jdbcType="CURSOR" mode="OUT"
      resultMap="replyResultMap"/>
    </parameterMap>
    <select id="replySelectAll" parameterMap="replySelectParameterMap"
    statementType="CALLABLE">
      { call sp_reply_select(?, ?) }
    </select>
```

5)Stored Procedure 생성

```
CREATE OR REPLACE PROCEDURE sp_reply_select
(
  v_idx           IN       Reply.idx%TYPE,
  reply_record    OUT      SYS_REFCURSOR
)
AS
BEGIN
  OPEN reply_record FOR
  SELECT replyer, replytext, replydate
  FROM Reply
  WHERE idx = v_idx
  ORDER BY replyidx ASC;
END;
```

6)Postman에서 테스트

GET : http://localhost:8080/biz/reply/4

```
{
  "code": "success",
  "data": [
    {
      "replyidx": 0,
      "idx": 0,
      "replyer": "jimin",
      "replytext": "감동입니다.",
      "replydate": 1593132977000
    },
    {
      "replyidx": 0,
      "idx": 0,
      "replyer": "javasoft",
      "replytext": "Very Good.",
      "replydate": 1593133151000
    }
  ]
}
```

7)view.html 수정

-jQuery 영역 밖의 새로운 함수 생성

```
  function listReply(idx){
    $.ajax({
      url : '/biz/reply/' + idx,
      type : 'GET',
      dataType : 'json',
      success : function(data){
```

```
                var mydata = data.data;
                var str = "<ul style='list-style-type:none'>";
                for(var i = 0 ; i < mydata.length ; i++){
                   str += "<li>";
                   str += "<p>작성자 : " + mydata[i].replyer + "<br />";
                   var now = new Date(mydata[i].replydate);
                   str += "작성날짜 : " + now.toLocaleString() + "</p>";
                   str += "<p>" + mydata[i].replytext + "</p>";
                   str += "</li>";
                   str += "<hr style='width:600px;'/>"
                }
                str += "</ul>";
                $('#replylist').html(str);
             }
          });
       }

    -기존 코드에 위의 함수 호출하는 코드 추가

       listReply(idx);  <---추가

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

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

       success : function(data){
          alert('댓글이 저장됐습니다.');
          $('#replyer').val("");
          $('#replytext').val("");
          listReply(idx);   <----추가
       }
```