```
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

```java
    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
    public List<MemberVO> readAll() {
        return this.sqlSession.selectList("Member.select");
    }

4)MemberServiceImpl.java

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

5)Postman
    GET http://localhost:8080/RestfulDemo/members Send
    Body

    {
        "code": "success",
        "data": [
            {
                "userId": "jimin",
                "name": "한지민",
                "gender": "여",
                "city": "서울"
            },
            {
                "userId": "example",
                "name": "조용필",
                "gender": "남성",
                "city": "부산"
            },
            {
                "userId": "javaexpert",
                "name": "이미자",
                "gender": "여성",
                "city": "광주"
            }
        ]
    }

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

7)WebContent/index.html

    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8">
            <title>Welcome</title>
            <script src="js/jquery-1.12.4.js"></script>
            <script>
```

```
         $(document).ready(function(){
            $.ajax({
               url:"/RestfulDemo/members",
               type : "GET",
               dataType : "json",
               success : function(data){
                  var str = "";
                  var members = data.data;
                  for(var i = 0 ; i < members.length ; i++){
                     str += "<tr>";
                     var userid = members[i].userid;
                     str += "<td><a href='view.html?userid=" + userid + "'>" + userid + "</a></td>" +
                           "<td>" + members[i].name + "</td>" +
                           "<td>" + members[i].gender + "</td>" +
                           "<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&nbsp;&nbsp;
                <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" : "소녀시대",
      "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;" +
            "<input type='radio' class='gender' value='여성' />여성";
        }else{
          str = "<input type='radio' class='gender' name='gender' value='남성' />남성 &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) {
        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>소녀시대</name>
    <gender>남성</gender>
    <city>부산</city>
  </member>
  <member userid="jimin">
    <name>박지민</name>
    <gender>여성</gender>
    <city>서울</city>
  </member>
</memberList>
```