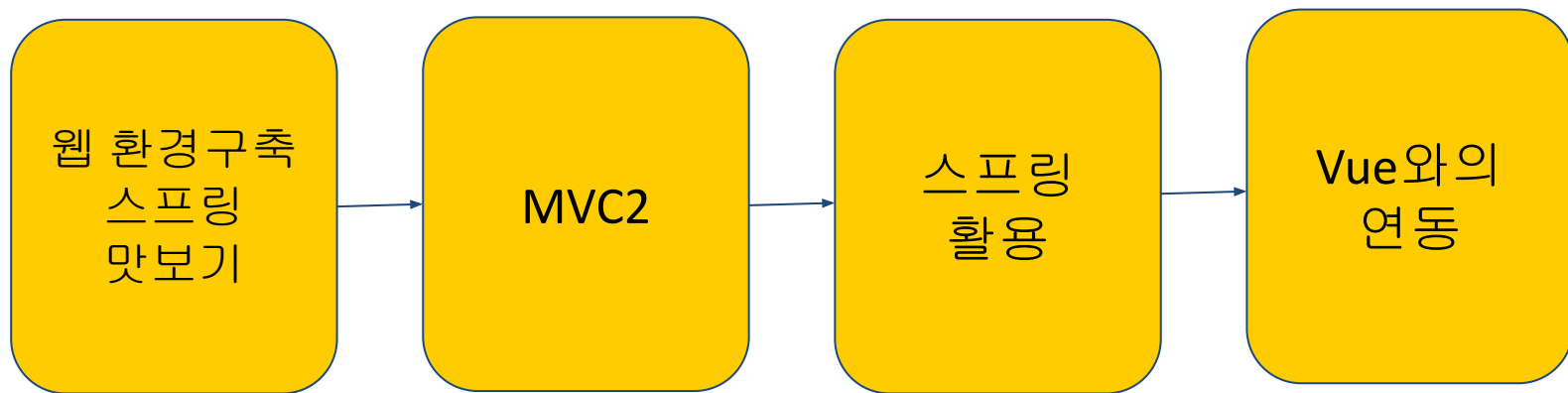
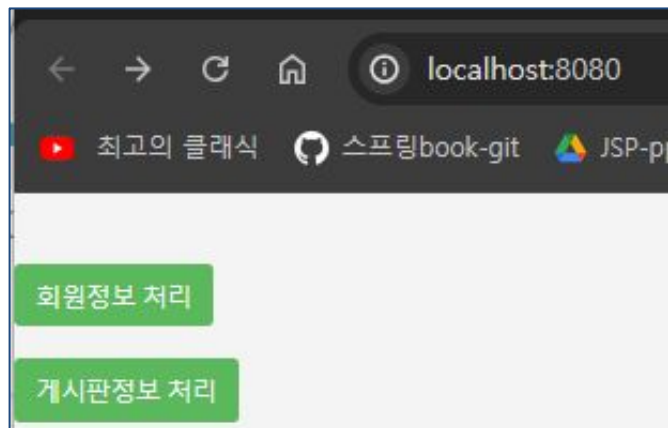


2024년 상반기 K-디지털 트레이닝

SPRING03 - myBatis

[KB] IT's Your Life

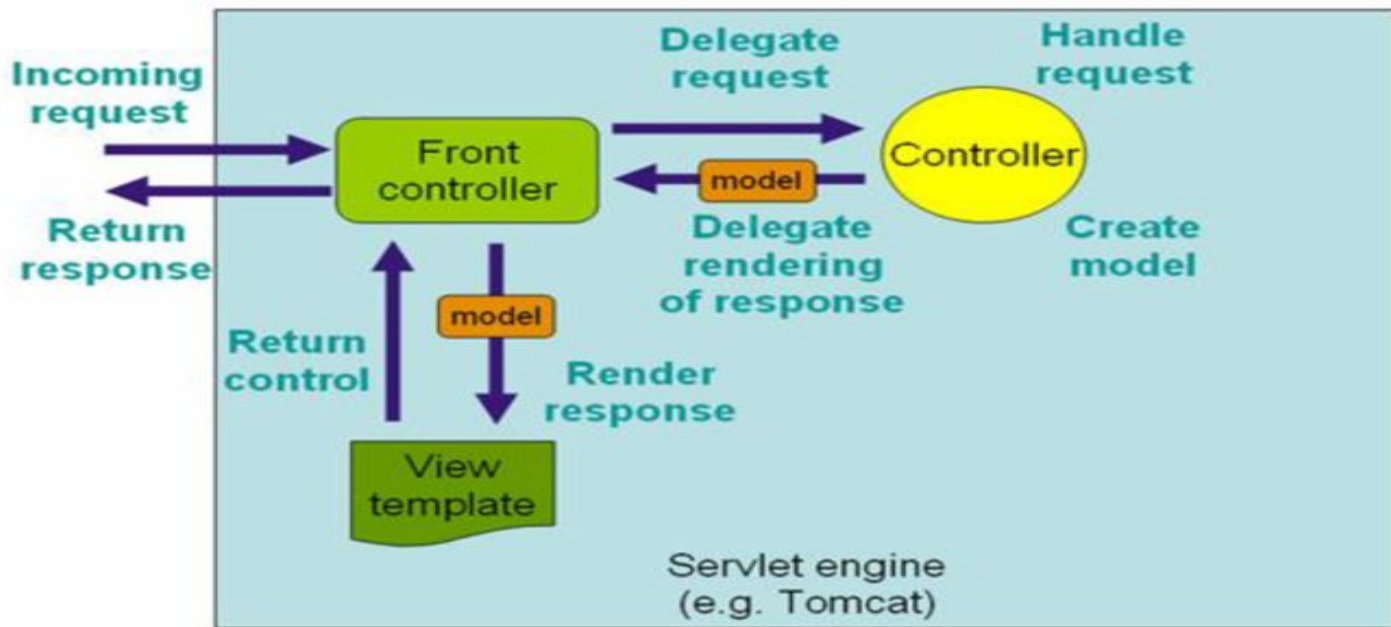


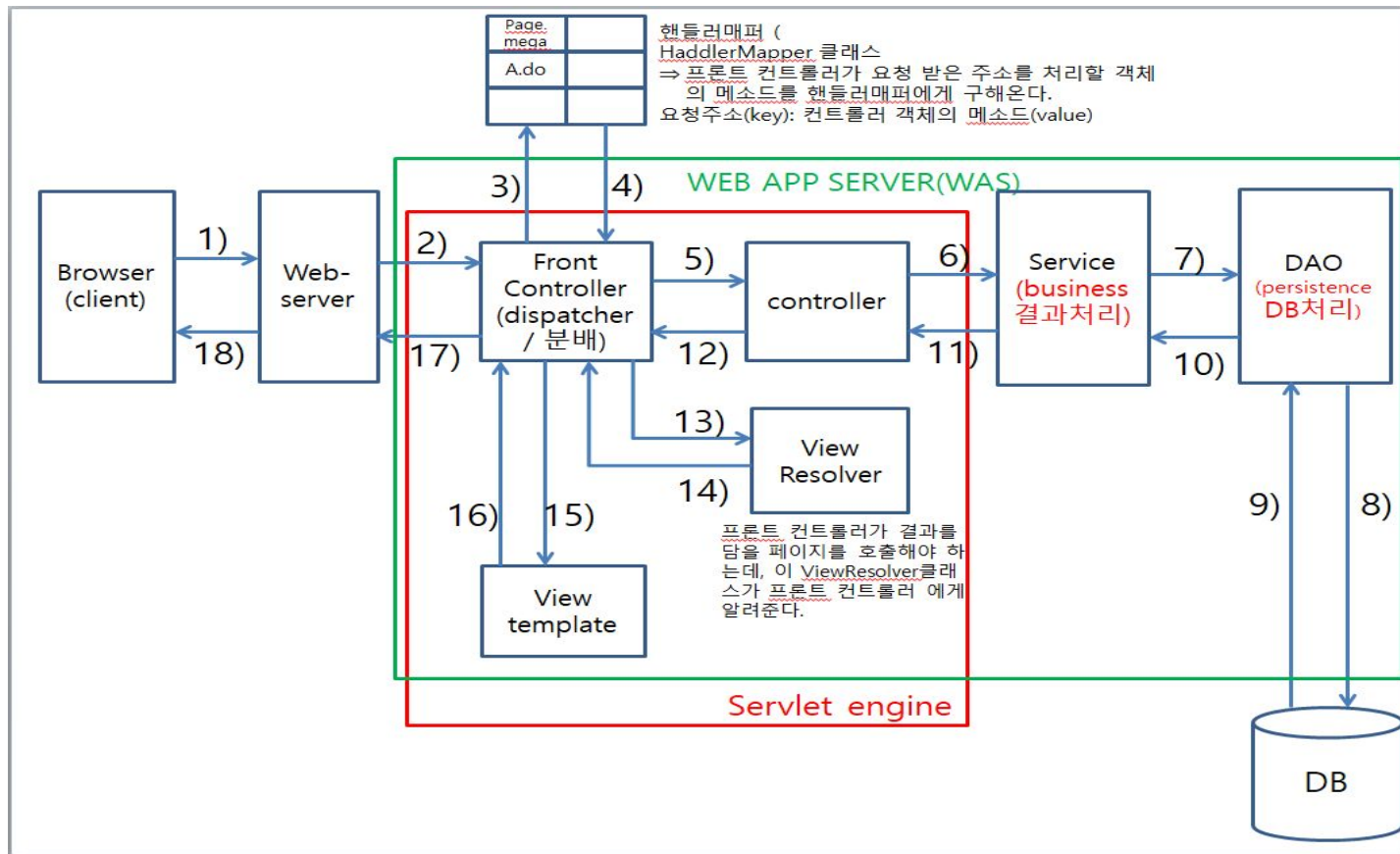


A screenshot of a web browser window showing a 'Member CRUD Form'. The browser address bar shows 'localhost:8080/member'. The form is titled 'Member CRUD Form' and contains several sections:

- ID**: A text input field.
- Password**: A text input field.
- Name**: A text input field.
- Telephone**: A text input field.
- Add Member**: A green button.
- 회원탈퇴** (Member Deletion): A section with a text input field containing 'apple' and a yellow 'Delete Member' button.
- 회원정보 수정** (Member Information Modification): A section with a text input field for 'Update Member by ID' and a text input field for 'New Telephone', followed by a green 'Update Member' button.
- 회원정보 검색** (Member Information Search): A section with a text input field containing 'apple' and a green 'Search' button.
- 회원 전체 검색** (Search All Members): A section with a green 'Search All' button.

myBatis





- myBatis 프레임워크

- 자바 기반의 퍼시스턴스 프레임워크
- SQL, 저장 프로시저 및 고급 매핑 기능을 지원
- MyBatis는 객체 관계 매핑(ORM) 프레임워크인 Hibernate와 달리 SQL에 더 많은 제어권을 제공
- MyBatis는 XML 파일이나 애노테이션을 사용하여 SQL 쿼리를 작성하고, 자바 객체와 데이터베이스의 테이블 간의 매핑을 처리
- 장점
 - 직관적인 **SQL 사용**: 개발자가 SQL 쿼리를 직접 작성할 수 있으므로, 복잡한 쿼리를 쉽게 처리
 - 유연한 매핑: XML 파일을 사용하여 자바 객체와 데이터베이스 테이블 간의 매핑을 유연하게 관리
 - 동적 **SQL 지원**: 다양한 조건과 반복문을 사용하여 동적 SQL을 생성
 - 확장성: 필요에 따라 매퍼와 SQL 문을 확장하고 재사용
- 단점
 - 복잡성: XML 매핑 파일을 관리해야 하므로, 프로젝트가 커지면 복잡
 - **SQL 의존성**: SQL을 직접 작성해야 하므로, 데이터베이스 변경 시 수정 작업이 많이 필요
 - 학습 곡선: MyBatis의 다양한 기능과 설정을 이해하는 데 시간이 필요

항목	MyBatis	JPA
접근 방식	SQL 직접 작성	객체-관계 매핑(ORM)
설정 방식	XML 매핑 파일, 애노테이션	애노테이션, XML 설정
유연성	높음	중간
자동화 수준	낮음	높음
쿼리 언어	SQL	JPQL
데이터베이스 의존성	높음	낮음
성능 최적화	가능 (SQL 직접 작성)	제한적 (ORM 추상화)
학습 곡선	중간	높음
디버깅 및 유지보수	쉬움	어려움

AppConfig

- db연결 정보 설정

mybatis-config.xml

- 전체설정

DAO.java

- DB처리

Mapper.java

- 인터페이스

Mapper.xml

- sql문 작성

- mybatis-config.xml

- MyBatis 설정 파일
- MyBatis의 전역 설정을 정의하는 파일
- MyBatis가 데이터베이스와 상호작용하는 방법을 구성
- 데이터베이스 연결 정보, 매퍼 파일의 위치, 트랜잭션 관리 설정, 캐시 설정 등을 포함
- MyBatis의 동작 방식을 설정하며, MyBatis 인스턴스가 초기화될 때 로드

- DAO.java

- 데이터베이스 접근 객체 (DAO): DAO(Data Access Object)는 데이터베이스와 상호작용하는 객체
- 애플리케이션의 다른 부분이 데이터베이스와 직접 상호작용하지 않도록 하며, 데이터베이스 접근 로직을 캡슐화
- MyBatis를 이용해 매퍼 XML 파일에 정의된 SQL 문장을 호출
- SQL 쿼리를 실행하여 데이터베이스로부터 데이터를 가져오거나 저장, 수정, 삭제 등의 작업을 수행

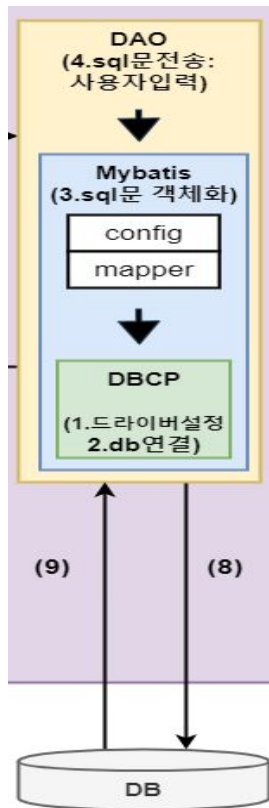
- Mapper.java

- MyBatis 매퍼 인터페이스
- MyBatis 매퍼는 SQL 쿼리를 Java 메서드에 매핑하는 인터페이스
- 주로 SQL 쿼리의 id와 동일한 메서드 이름을 사용하여 정의
- 매퍼 XML 파일에서 정의된 SQL 문장을 호출하는 메서드를 선언
- MyBatis는 런타임 시 이 인터페이스의 구현체를 생성하여 SQL을 실행

- Mapper.xml

- MyBatis 매퍼 XML 설정 파일
- SQL 쿼리와 매퍼 인터페이스 메서드를 매핑
- SQL 쿼리는 XML 형식으로 작성되며, 각 쿼리는 <select>, <insert>, <update>, <delete> 태그로 정의
- SQL 문장과 Java 메서드를 매핑하여, SQL 문장을 별도로 관리하고 코드와 SQL 문장을 분리
- 유지보수가 용이해지고, SQL 변경 시 Java 코드를 수정할 필요가 없음

- myBatis 동작 방식



DAO

myBatis(SqlSessionTemplate)

config(SqlSessionFactory)

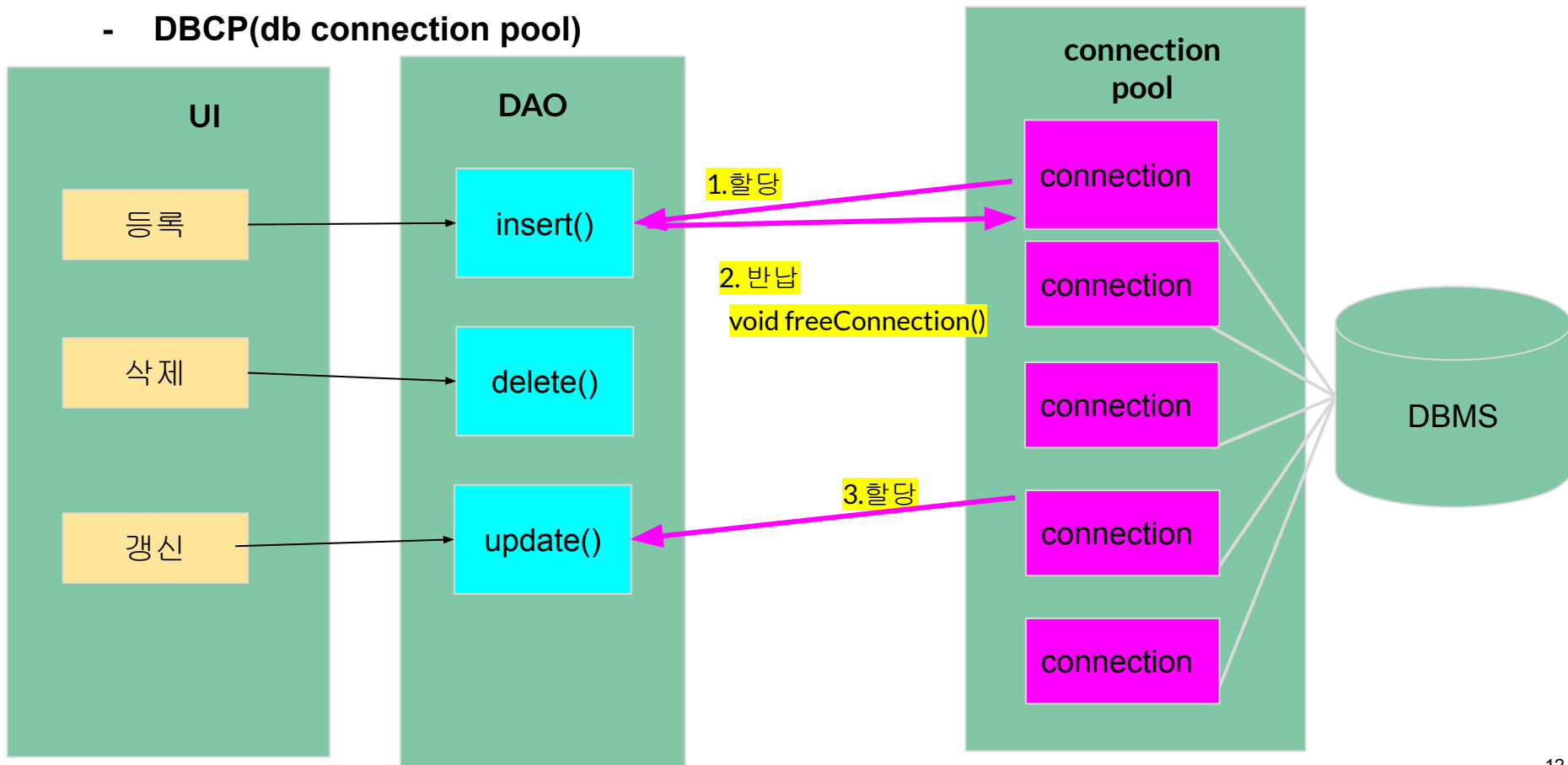
mybatis-config.xml

- alias
- mapper list

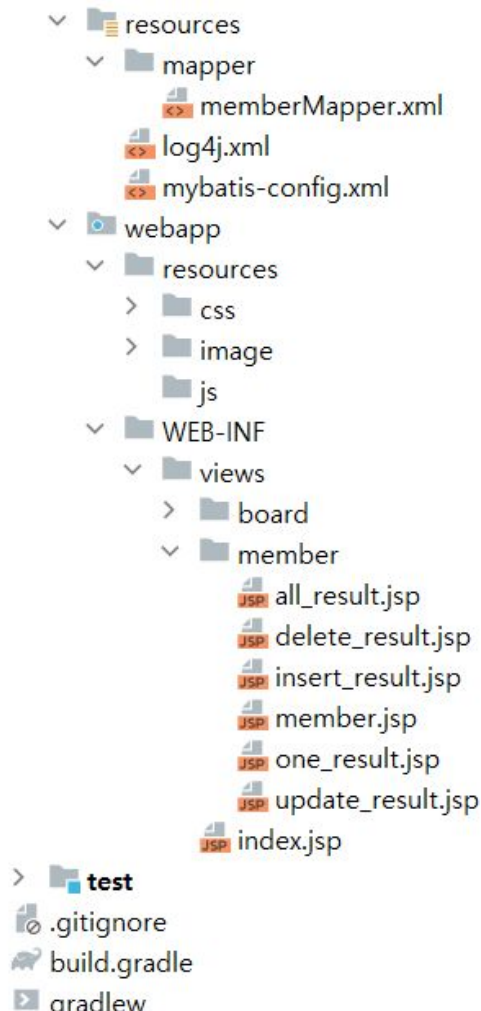
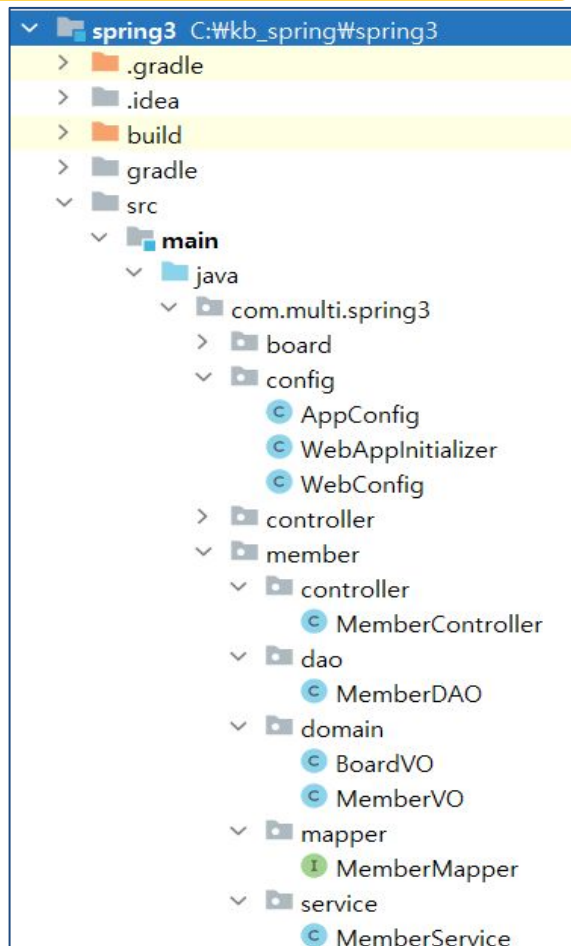
datasource

- driver
- db연결

- DBCP(db connection pool)



스프링 프로젝트 만들기



파일명	설명
MemberVO.java	회원 정보를 담는 Value Object 클래스 Lombok 어노테이션을 사용하여 getter, setter, 생성자를 자동 생성
MemberDAO.java	메모리 내 데이터베이스 역할을 하며 회원 정보의 CRUD 작업을 담당

model

파일명	설명
member.jsp	회원 정보를 입력받는 폼을 제공
all_result.jsp	모든 회원 정보를 테이블 형식으로 출력
insert_result.jsp	회원 추가 결과를 출력
update_result.jsp	회원 수정 결과를 출력
delete_result.jsp	회원 삭제 결과를 출력
one_result.jsp	단일 회원 정보를 출력

view

파일명	설명
MemberController.java	회원 정보와 관련된 요청을 처리
RootController.java	루트 경로와 관련된 요청을 처리
BoardController.java	게시판 관련 요청을 처리

control

파일명	설명
MemberService.java	비즈니스 로직을 처리하며 DAO를 통해 데이터베이스 작업을 수행

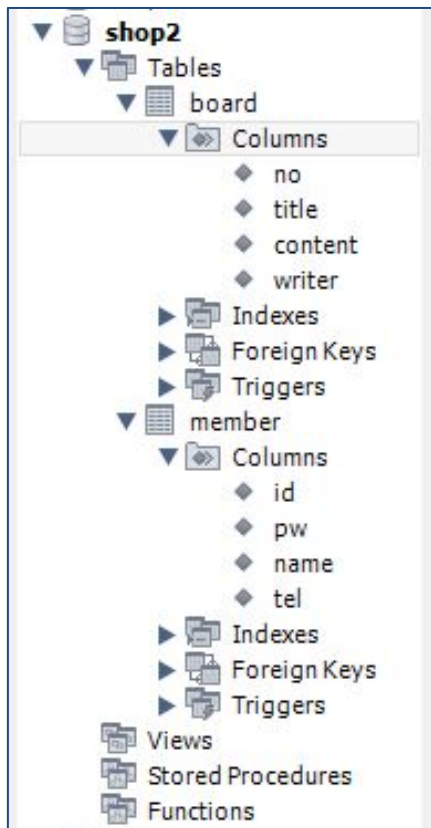
service

파일명	설명
AppConfig.java	애플리케이션의 기본 설정을 담당
WebConfig.java	Spring MVC 설정을 담당하며, 뷰 리졸버를 설정
WebAppInitializer.java	서블릿 컨텍스트를 초기화

config

src
main
java
com.multi.spring3
board
config
AppConfig
WebAppInitializer
WebConfig
controller
member
controller
MemberController
dao
MemberDAO
domain
BoardVO
MemberVO
mapper
MemberMapper
service
MemberService
resources
mapper
memberMapper.xml
log4j.xml
mybatis-config.xml

설명	파일 이름	구분	패키지 및 경로
데이터베이스 접근 객체 (DAO)	MemberDAO	Java 파일	com.multi.spring3.dao
MyBatis 매퍼 인터페이스	MemberMapper	Java 파일	com.multi.spring3.mapper
MyBatis 매퍼 XML 설정 파일	memberMapper.xml	XML 파일	resources.mapper
MyBatis 설정 파일	mybatis-config.xml	XML 파일	resources

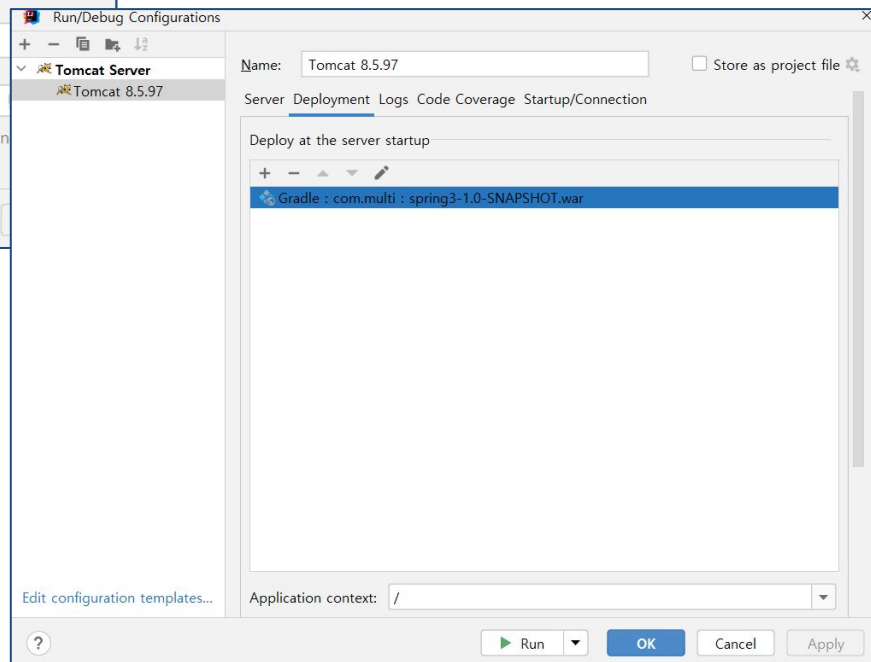
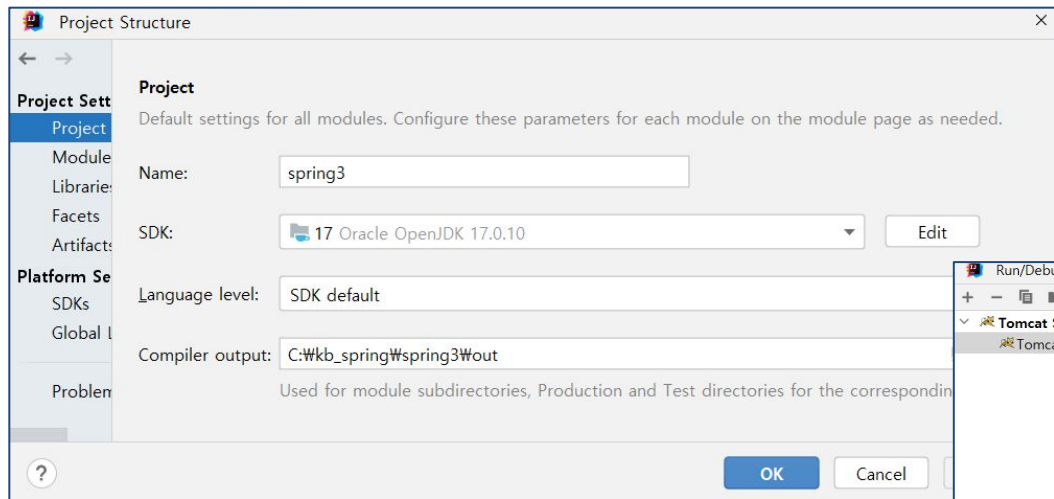


```
use shop2;  
CREATE TABLE `member` (  
  `id` varchar(45) NOT NULL,  
  `pw` varchar(45) NOT NULL,  
  `name` varchar(45) DEFAULT NULL,  
  `tel` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
);
```

```
desc member;
```

```
CREATE TABLE `board` (  
  `no` int NOT NULL AUTO_INCREMENT,  
  `title` varchar(45) NOT NULL,  
  `content` varchar(45) NOT NULL,  
  `writer` varchar(45) NOT NULL,  
  PRIMARY KEY (`no`)  
);
```

```
desc board;
```



```
dependencies {
    // 스프링
    implementation ("org.springframework:spring-context:5.3.37")
    { exclude group: 'commons-logging', module: 'commons-logging' }
    implementation "org.springframework:spring-webmvc:5.3.37"
    implementation 'javax.inject:javax.inject:1'
    implementation 'org.springframework:spring-web:5.3.37'

    //mybatis
    implementation 'org.springframework:spring-jdbc:5.3.37'
    implementation 'org.mybatis:mybatis:3.5.10'
    implementation 'org.mybatis:mybatis-spring:2.0.7'
    runtimeOnly 'mysql:mysql-connector-java:8.0.26'
    implementation 'org.apache.commons:commons-dbcp2:2.9.0'

    // AOP
    implementation 'org.aspectj:aspectjrt:1.9.20'
    implementation 'org.aspectj:aspectjweaver:1.9.20'

    // JSP, SERVLET, JSTL
    implementation('javax.servlet:javax.servlet-api:4.0.1')
    compileOnly 'javax.servlet.jsp:jsp-api:2.1'
    implementation 'javax.servlet:jstl:1.2'

    // Logging
    implementation 'org.slf4j:slf4j-api:2.0.9'
    runtimeOnly 'org.slf4j:jcl-over-slf4j:2.0.9'
    runtimeOnly 'org.slf4j:slf4j-log4j12:2.0.9'
    implementation 'log4j:log4j:1.2.17'

    // xml내 한글 처리
    implementation 'xerces:xercesImpl:2.12.2'

    compileOnly 'org.projectlombok:lombok:1.18.28'
    annotationProcessor 'org.projectlombok:lombok:1.18.28'
}
```

```
ext {
    junitVersion = '5.9.2'
    springVersion = '5.3.37'
}

sourceCompatibility = '17'
targetCompatibility = '17'

tasks.withType(JavaCompile) {
    options.encoding = 'UTF-8'
}

dependencies {
    // 스프링
    implementation ("org.springframework:spring-context:5.3.37")
    { exclude group: 'commons-logging', module: 'commons-logging' }
    implementation "org.springframework:spring-webmvc:5.3.37"
    implementation 'javax.inject:javax.inject:1'
    implementation 'org.springframework:spring-web:5.3.37'

    //mybatis
    implementation 'org.springframework:spring-jdbc:5.3.37'
    implementation 'org.mybatis:mybatis:3.5.10'
    implementation 'org.mybatis:mybatis-spring:2.0.7'
    runtimeOnly 'mysql:mysql-connector-java:8.0.26'
    implementation 'org.apache.commons:commons-dbcp2:2.9.0'

    // AOP
    implementation 'org.aspectj:aspectjrt:1.9.20'
    implementation 'org.aspectj:aspectjweaver:1.9.20'

    // JSP, SERVLET, JSTL
    implementation('javax.servlet:javax.servlet-api:4.0.1')
    compileOnly 'javax.servlet.jsp:jsp-api:2.1'
    implementation 'javax.servlet:jstl:1.2'

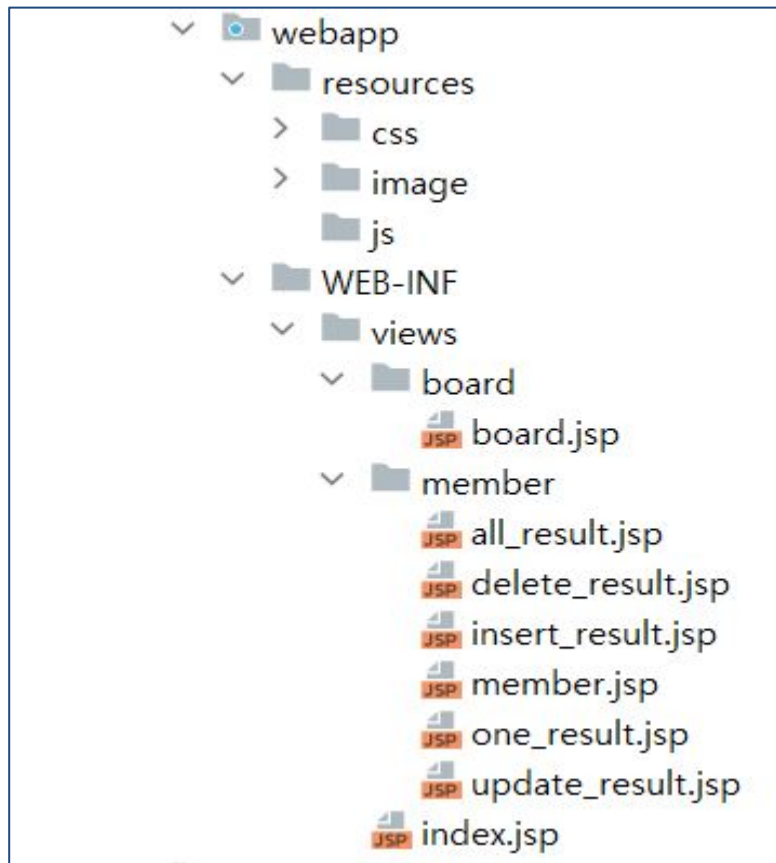
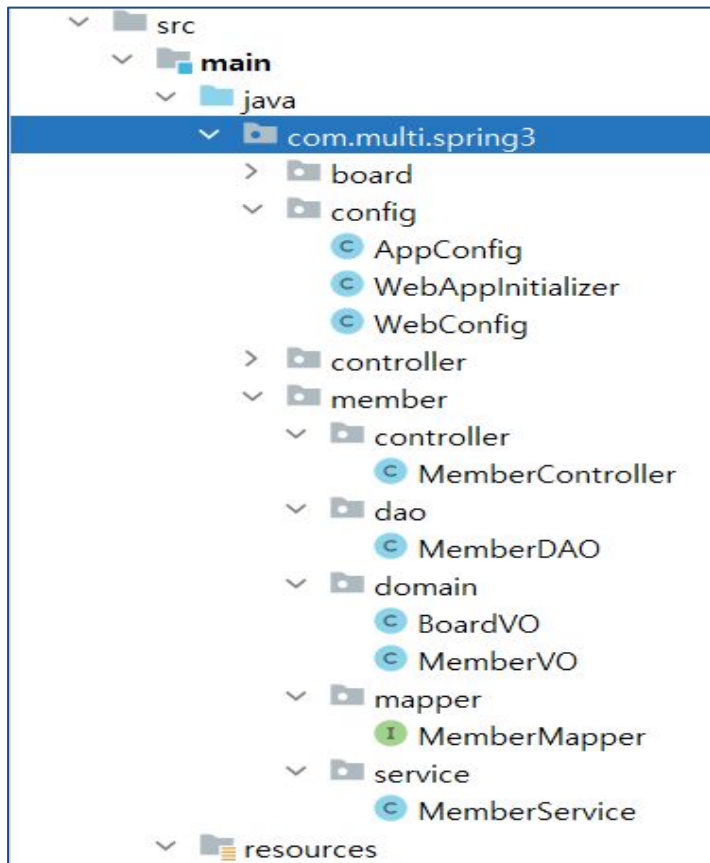
    // Logging
    implementation 'org.slf4j:slf4j-api:2.0.9'
    runtimeOnly 'org.slf4j:jcl-over-slf4j:2.0.9'
    runtimeOnly 'org.slf4j:slf4j-log4j12:2.0.9'
    implementation 'log4j:log4j:1.2.17'

    // xml내 한글 처리
    implementation 'xerces:xercesImpl:2.12.2'

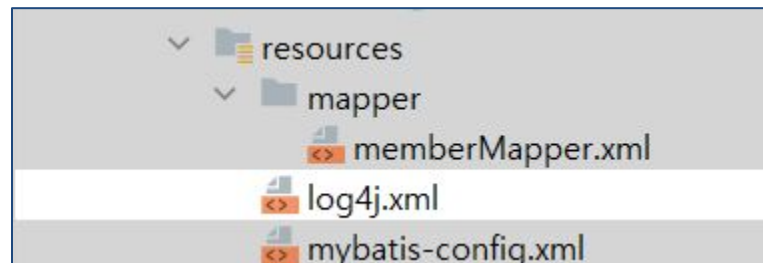
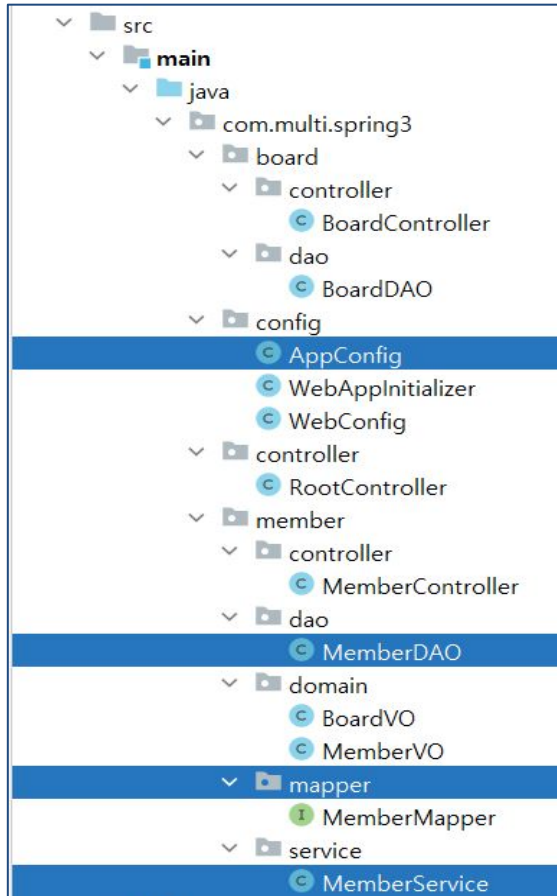
    compileOnly 'org.projectlombok:lombok:1.18.28'
    annotationProcessor 'org.projectlombok:lombok:1.18.28'

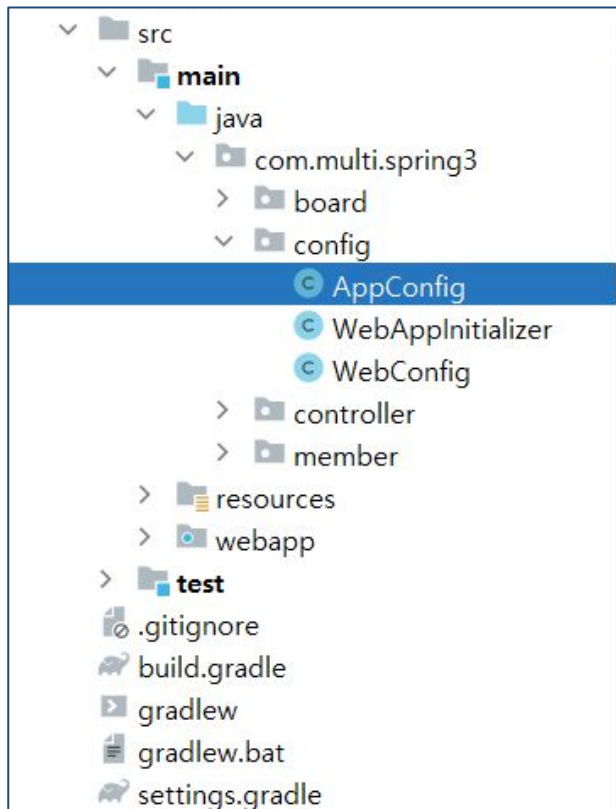
    testImplementation "org.springframework:spring-test:${springVersion}"
    testCompileOnly 'org.projectlombok:lombok:1.18.28'
    testAnnotationProcessor 'org.projectlombok:lombok:1.18.28'
    testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")
}
```

- 폴더 만들기



- 폴더 만들기





```

@Configuration
@ComponentScan(basePackages = "com.multi")
public class AppConfig {

    public AppConfig() {
        System.out.println("AppConfig created");
    }

    @Bean
    public DataSource dataSource() {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");

        dataSource.setUrl("jdbc:mysql://localhost:3306/shop2?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC&characterEncoding=UTF-8&useUnicode=true");

        dataSource.setUsername("root");
        dataSource.setPassword("1234");
        dataSource.setInitialSize(5); // 초기 커넥션 수
        dataSource.setMaxTotal(10); // 최대 커넥션 수
        return dataSource;
    }

    @Bean
    public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        sessionFactory.setConfigLocation(new ClassPathResource("mybatis-config.xml"));
        return sessionFactory.getObject();
    }

    @Bean
    public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory) {
        return new SqlSessionTemplate(sqlSessionFactory);
    }
}
    
```



```
1 package com.multi.spring3.config;
2
3 import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;
4
5 public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
6
7     public WebAppInitializer() { System.out.println("WebAppInitializer created"); }
8
9
10
11     @Override 5 usages
12     protected Class<?>[] getRootConfigClasses() { return new Class[] { AppConfig.class }; }
13
14
15
16     @Override 4 usages
17     protected Class<?>[] getServletConfigClasses() { return new Class[] { WebConfig.class }; }
18
19
20
21     @Override 2 usages
22     protected String[] getServletMappings() { return new String[] { "/" }; }
23
24
25 }
26
```



```
package com.multi.spring3.config;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    public WebAppInitializer() {
        System.out.println("WebAppInitializer created");
    }

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { AppConfig.class };
    }

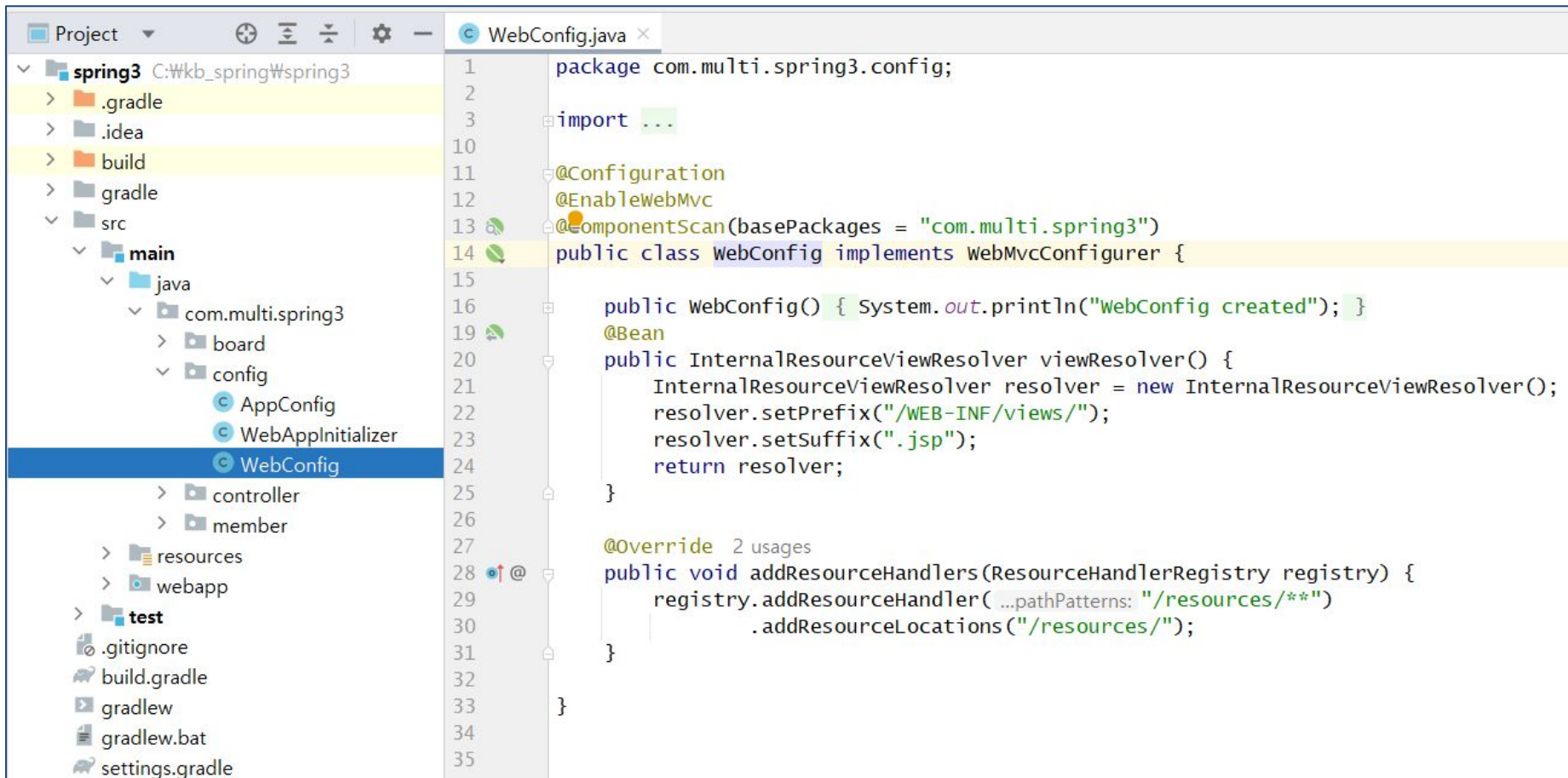
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { WebConfig.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }
}
```

- 한글 처리 필터

```
import javax.servlet.Filter;

@Override
protected Filter[] getServletFilters() {
    CharacterEncodingFilter characterEncodingFilter = new
    CharacterEncodingFilter();
    characterEncodingFilter.setEncoding("UTF-8");
    characterEncodingFilter.setForceEncoding(true);
    return new Filter[] { characterEncodingFilter };
}
```



The screenshot shows an IDE with a project named `spring3` located at `C:\wkb_spring\spring3`. The project structure includes `.gradle`, `.idea`, `build`, `gradle`, `src`, and `test` directories. The `src` directory contains a `main` folder with `java` and `resources` subfolders. The `java` folder contains `com.multi.spring3`, which has `board` and `config` subfolders. The `config` folder contains `AppConfig`, `WebAppInitializer`, and `WebConfig` (selected). The `WebConfig.java` file is open, showing the following code:

```
1 package com.multi.spring3.config;
2
3 import ...
4
5 @Configuration
6 @EnableWebMvc
7 @ComponentScan(basePackages = "com.multi.spring3")
8 public class WebConfig implements WebMvcConfigurer {
9
10     public WebConfig() { System.out.println("WebConfig created"); }
11
12     @Bean
13     public InternalResourceViewResolver viewResolver() {
14         InternalResourceViewResolver resolver = new InternalResourceViewResolver();
15         resolver.setPrefix("/WEB-INF/views/");
16         resolver.setSuffix(".jsp");
17         return resolver;
18     }
19
20     @Override 2 usages
21     public void addResourceHandlers(ResourceHandlerRegistry registry) {
22         registry.addResourceHandler(...pathPatterns: "/resources/**")
23             .addResourceLocations("/resources/");
24     }
25 }
26
27
28
29
30
31
32
33
34
35
```

```
package com.multi.spring3.config;

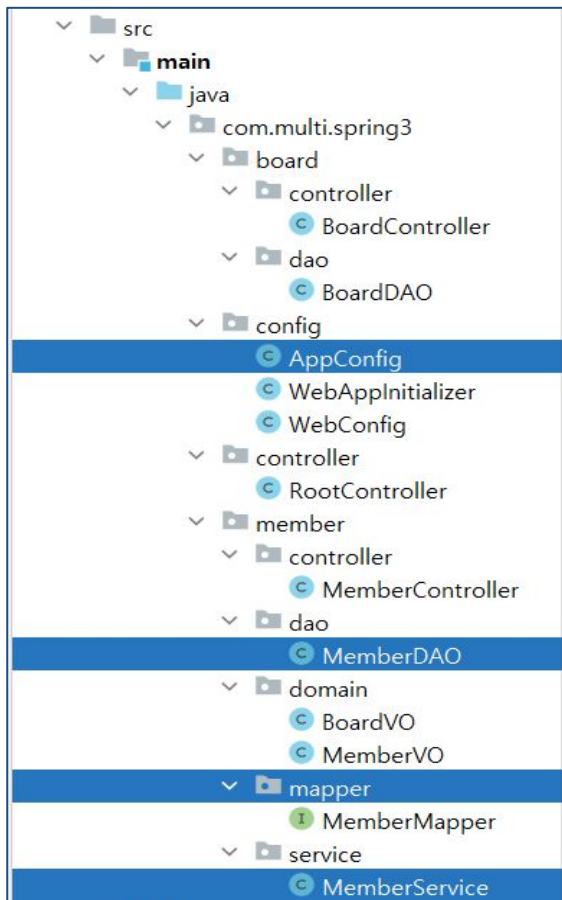
import org.springframework.context.annotation. Bean;
import org.springframework.context.annotation. ComponentScan;
import org.springframework.context.annotation. Configuration;
import org.springframework.web.servlet.config.annotation. EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

@Configuration
@EnableWebMvc
@ComponentScan (basePackages = "com.multi.spring3")
public class WebConfig implements WebMvcConfigurer {

    public WebConfig() {
        System.out.println("WebConfig created");
    }

    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix( "/WEB-INF/views/" );
        resolver.setSuffix( ".jsp" );
        return resolver;
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler( "/resources/**" )
            .addResourceLocations( "/resources/" );
    }
}
```



```
package com.multi.spring3.member.mapper;
```

```
import com.multi.spring3.member.domain.MemberVO;
import org.apache.ibatis.annotations.Mapper;
import java.util.List;
```

```
@Mapper
```

```
public interface MemberMapper {
```

```
    //mapper의 id와 맞아야함.
```

```
    int insert(MemberVO memberVO); //<insert id="insert" ~~>
```

```
    int update(MemberVO memberVO);
```

```
    int delete(String id);
```

```
    MemberVO one(String id); // <select id="one" ~~>
```

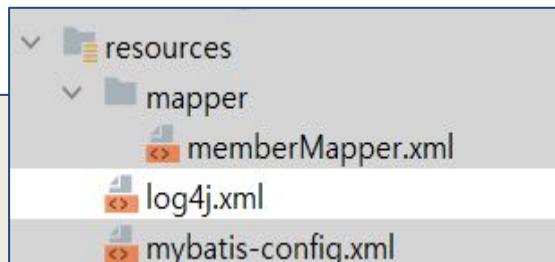
```
    List<MemberVO> all();
```

```
}
```

```
<?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>
    <package name="com.multi.spring3.member.domain"/>
  </typeAliases>

  <typeHandlers>
    <typeHandler javaType="String" handler="com.multi.spring3.config.StringTypeHandler"/>
    <typeHandler javaType="Integer" handler="com.multi.spring3.config.IntegerTypeHandler"/>
    <typeHandler javaType="Boolean" handler="com.multi.spring3.config.BooleanTypeHandler"/>
  </typeHandlers>

  <mappers>
    <mapper resource="mapper/memberMapper.xml"/>
  </mappers>
</configuration>
```



myBatis 설정 파일

```
<settings>
```

```
  <setting name="logImpl" value="SLF4J"/>
```

```
  <!-- MyBatis가 로그를 기록할 유형을 결정 -->
```

```
  <setting name="cacheEnabled" value="true"/>
```

```
  <!-- MyBatis에서 2차 캐시를 사용할지 여부를 결정
```

```
  - true: 매퍼블로 2차 캐시가 활성화
```

SQL 쿼리 결과를 메모리에 캐시하여, 동일한 쿼리가 반복적으로 실행될 때 데이터베이스를 다시 조회하지 않고 캐시된 결과를 사용
성능을 크게 향상시킬 수 있지만, 캐시된 데이터가 최신 상태가 아닐 수 있는 위험이 있음.

```
  - false: 2차 캐시가 비활성화, 모든 쿼리는 항상 데이터베이스에서 직접 실행됩니다.
```

```
  -->
```

```
  <setting name="lazyLoadingEnabled" value="true"/>
```

```
  <!-- 관계형 데이터를 지연 로딩(lazy loading)할지 여부를 결정
```

```
  - true: 관계형 데이터(예: 다른 테이블과의 관계를 나타내는 필드)는 실제로 그 데이터가 필요할 때(즉, 해당 필드에 접근할 때)만 로드  
    성능을 향상시킬 수 있으나, 예상치 못한 시점에 데이터베이스 접근이 발생
```

```
  - false: 모든 관계형 데이터는 해당 객체가 로드될 때 즉시 함께 로드.
```

```
  - 지연 로딩이 없기 때문에 예측 가능성이 높지만, 불필요한 데이터를 로드하여 메모리 사용이 증가
```

```
  -->
```

```
  <setting name="multipleResultSetsEnabled" value="true"/>
```

```
  <!--하나의 쿼리에서 다중 결과 집합을 허용할지 여부를 결정-->
```

```
  <setting name="useGeneratedKeys" value="true"/>
```

```
  <!--
```

데이터베이스에서 자동 생성된 키(예: AUTO INCREMENT 또는 SERIAL 필드)를 MyBatis가 자동으로 사용할지 여부를 결정

```
  - true: MyBatis는 데이터베이스에서 생성된 키를 자동으로 가져와서 사용
```

```
  - 예를 들어, INSERT 문 이후에 생성된 키를 자동으로 받아와서 해당 객체의 필드에 설정
```

```
  - false: MyBatis는 자동으로 키를 가져오지 않으며, 개발자가 직접 키를 가져와야 함.
```

```
  -->
```

```
  <setting name="defaultExecutorType" value="SIMPLE"/>
```

```
  <!-- MyBatis가 SQL을 실행할 때 사용할 기본 실행기(executor) 유형을 결정 -->
```

```
  <setting name="autoMappingBehavior" value="PARTIAL"/>
```

```
  <!-- MyBatis가 결과 집합의 열을 객체의 필드에 자동으로 매핑할지 여부를 결정 -->
```

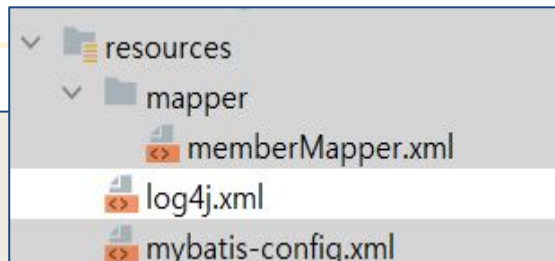
```
  <!--
```

NONE: 자동 매핑을 하지 않음. 개발자가 매핑을 명시적으로 설정

PARTIAL: 결과 집합의 필드가 객체의 필드명과 일치할 경우에만 자동 매핑을 수행

FULL: 결과 집합의 모든 열을 객체의 필드와 자동으로 매핑. 필드명과 일치하지 않는 경우에도 시도.-->

```
</settings>
```



- config
 - AppConfig
 - BooleanTypeHandler
 - IntegerTypeHandler
 - StringTypeHandler**
 - WebAppInitializer
 - WebConfig

```
<resultMap id="orm rule1" type="com.multi.spring3.member.domain.MemberVO">
  <result property="id" column="id"
  typeHandler="com.multi.spring3.config.StringTypeHandler"/>
  <result property="pw" column="pw"/>
  <result property="name" column="name"/>
  <result property="tel" column="tel"/>
</resultMap>

<select id="getUser" resultType="MemberVO">
  SELECT id,
         COALESCE(name, 'N/A') AS name, -- name null일 경우 'N/A'로 처리
         COALESCE(age, 0) AS age -- age null일 경우 0으로 처리
  FROM member
  WHERE id = #{id}
</select>
```

```
package com.multi.spring3.config;

import org.apache.ibatis.type.BaseTypeHandler;
import org.apache.ibatis.type.JdbcType;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

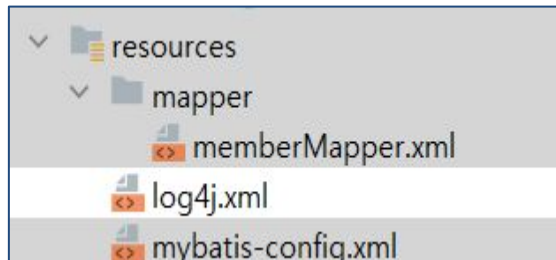
public class StringTypeHandler extends BaseTypeHandler<String> {

    @Override
    public void setNonNullParameter(PreparedStatement ps, int i, String parameter, JdbcType
    jdbcType) throws SQLException {
        ps.setString(i, parameter);
    }

    @Override
    public String getNullableResult(ResultSet rs, String columnName) throws SQLException {
        String result = rs.getString(columnName);
        return result == null ? "" : result;
    }

    @Override
    public String getNullableResult(ResultSet rs, int columnIndex) throws SQLException {
        String result = rs.getString(columnIndex);
        return result == null ? "" : result;
    }

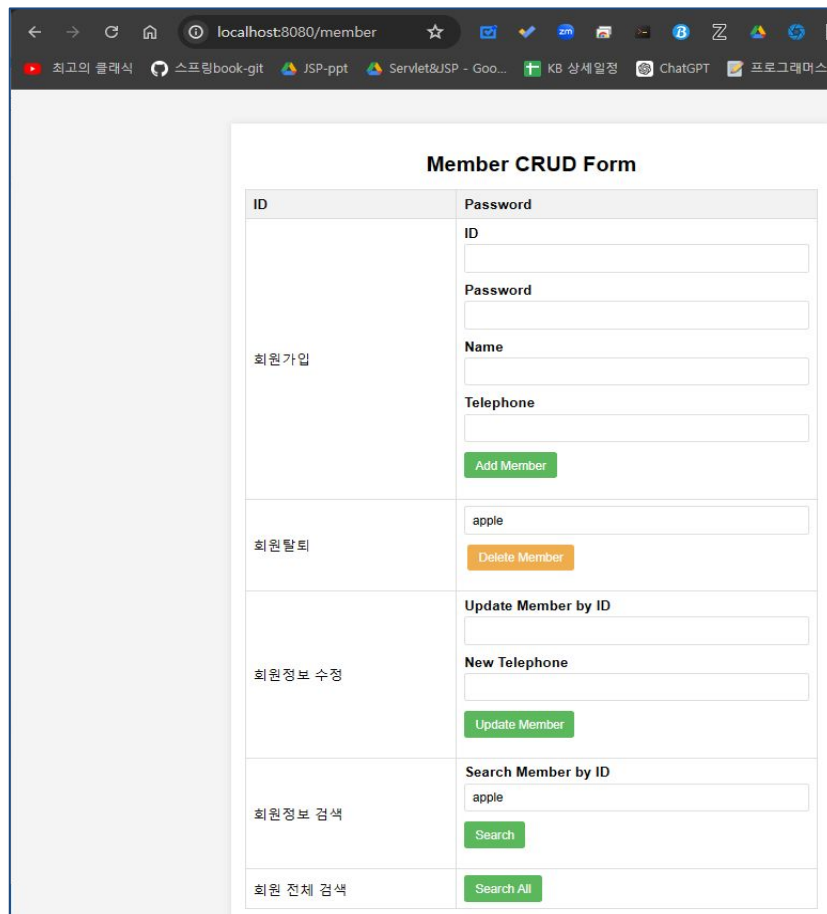
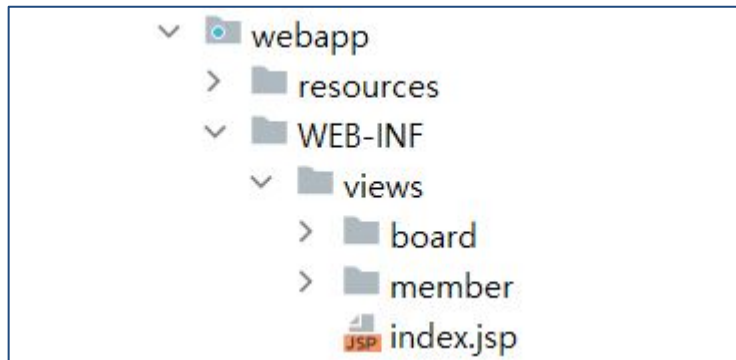
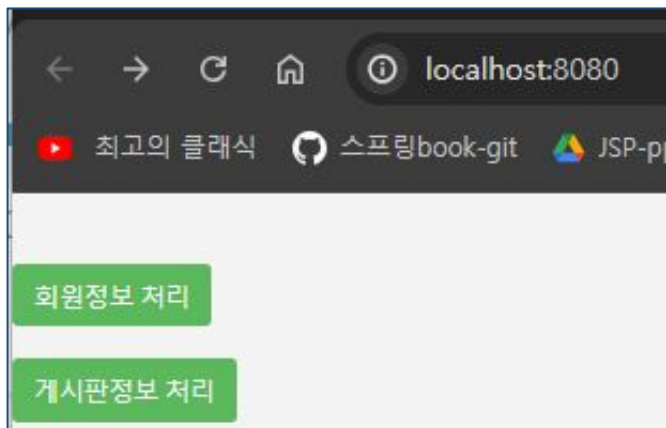
    @Override
    public String getNullableResult(java.sql.CallableStatement cs, int columnIndex) throws
    SQLException {
        String result = cs.getString(columnIndex);
        return result == null ? "" : result;
    }
}
```

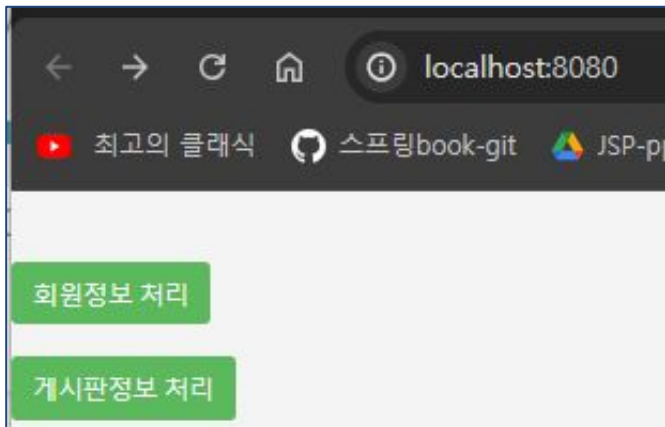



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.multi.spring3.member.mapper.MemberMapper">
    <insert id="insert" parameterType="com.multi.spring3.member.domain.MemberVO">
        insert into member
        values ( #{id}, #{pw}, #{name}, #{tel} );
    </insert>
    <update id="update" parameterType="com.multi.spring3.member.domain.MemberVO">
        update member set tel =  #{tel}
        where id =  #{id}
    </update>
    <delete id="delete" parameterType="String">
        delete from member
        where id =  #{id}
    </delete>
    <select id="one" parameterType="String"
        resultType="com.multi.spring3.member.domain.MemberVO">
        select * from member
        where id =  #{id}
    </select>
    <select id="all" resultType="com.multi.spring3.member.domain.MemberVO">
        select * from member
    </select>
</mapper>
```

태그	설명	예시 쿼리
<insert>	새로운 레코드 삽입	insert into member (id, pw, name, tel) values ({id}, {pw}, {name}, {tel});
<update>	기존 레코드 업데이트	update member set tel = {tel} where id = {id};
<delete>	레코드 삭제	delete from member where id = {id};
<select>	단일 또는 다중 레코드 조회	select * from member where id = {id}; select * from member;
namespace	고유 네임스페이스 정의	com.multi.spring3.member.mapper.MemberMapper
id	쿼리의 고유 ID	id="insert", id="update", id="delete", id="one", id="all"
parameterType	파라미터 타입 지정	parameterType="com.multi.spring3.member.domain.MemberVO", parameterType="String"
resultType	결과 타입 지정	resultType="com.multi.spring3.member.domain.MemberVO"
resultMap	복잡한 매핑을 위한 결과 매핑 정의	<resultMap id="memberResultMap" type="com.multi.spring3.member.domain.MemberVO"> <id property="id" column="member_id"/> <result property="name" column="member_name"/> <result property="tel" column="member_tel"/> </resultMap>





```

Connected to server
[2024-07-26 09:47:05,282] I
26-Jul-2024 21:47:09.385 I
webAppInitializer created
26-Jul-2024 21:47:09.679 I
26-Jul-2024 21:47:10.681 I
26-Jul-2024 21:47:10.785 I
WebConfig created
RootController created
MemberService created
MemberController created
MemberDAO created.

```

```

<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
  <title>Spring</title>
  <link rel="stylesheet" href="resources/css/out.css">
</head>
<body>
<h1></h1>
<br/>
<table>
  <tbody>
    <tr>
      <td style="background: lemonchiffon; text-align: center">
        <a href="member">
          <button>회원정보 처리</button>
        </a>
      </td>
    </tr>
    <tr>
      <td style="background: lemonchiffon; text-align: center">
        <a href="board"><button>게시판정보 처리</button></a><br>
      </td>
    </tr>
  </tbody>
</table>
</body>
</html>

```

▼ **spring3** C:\wkb_spring\spring3

- > .gradle
- > .idea
- > build
- > gradle
- ▼ src
 - ▼ main
 - ▼ java
 - ▼ com.multi.spring3
 - > board
 - > config
 - ▼ controller
 - RootController**
 - > member
 - > resources
 - > webapp

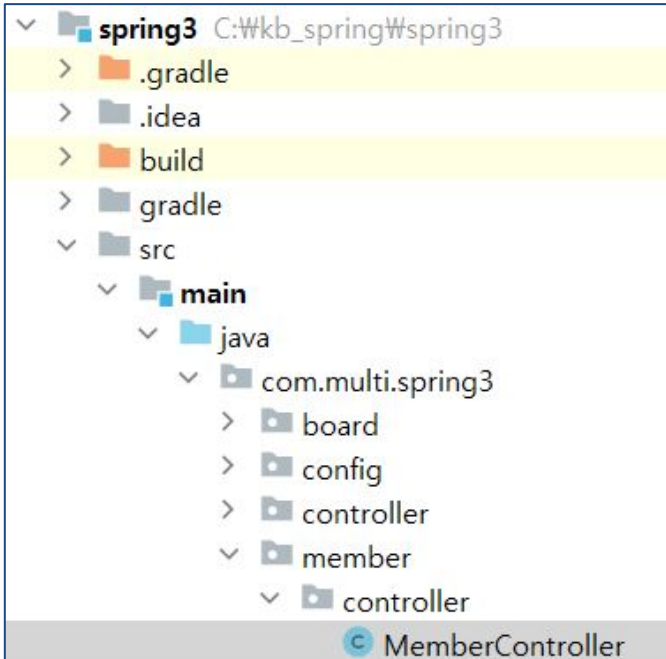
```
package com.multi.spring3.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class RootController {

    public RootController() {
        System.out.println("RootController created");
    }

    @GetMapping("/")
    public String root() {
        return "index";
    }
}
```



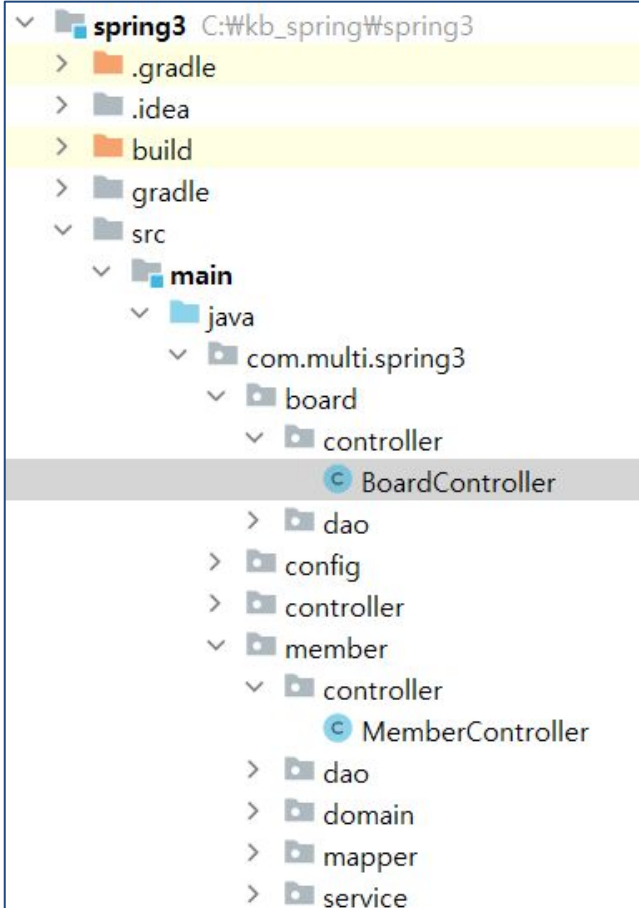
```
package com.multi.spring3.member.controller;

import com.multi.spring3.member.domain.MemberVO;
import com.multi.spring3.member.service.MemberService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/member")
public class MemberController {

    @GetMapping
    public String index() {
        return "member/member";
    }
}
```

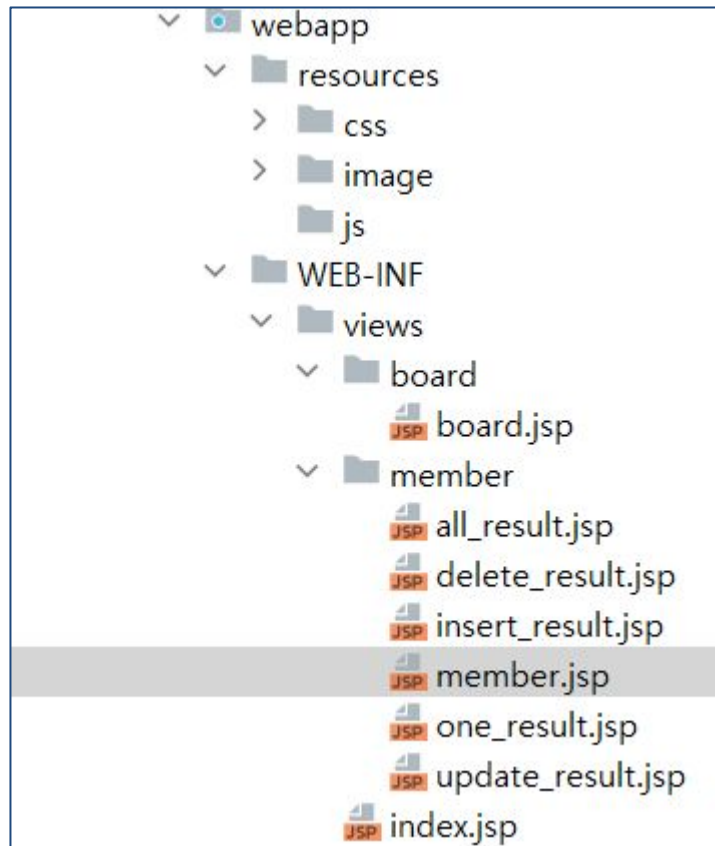


```
package com.multi.spring3.board.controller;

import com.multi.spring3.member.domain.MemberVO;
import com.multi.spring3.member.service.MemberService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/board")
public class BoardController {

    @RequestMapping
    public String boardHome() {
        return "board/board";
    }
}
```



localhost:8080/member

최고의 블렉식 스프링book-git JSP-ppt Servlet&JSP - Goo... KB 상세일정 ChatGPT 프로그래머스

Member CRUD Form

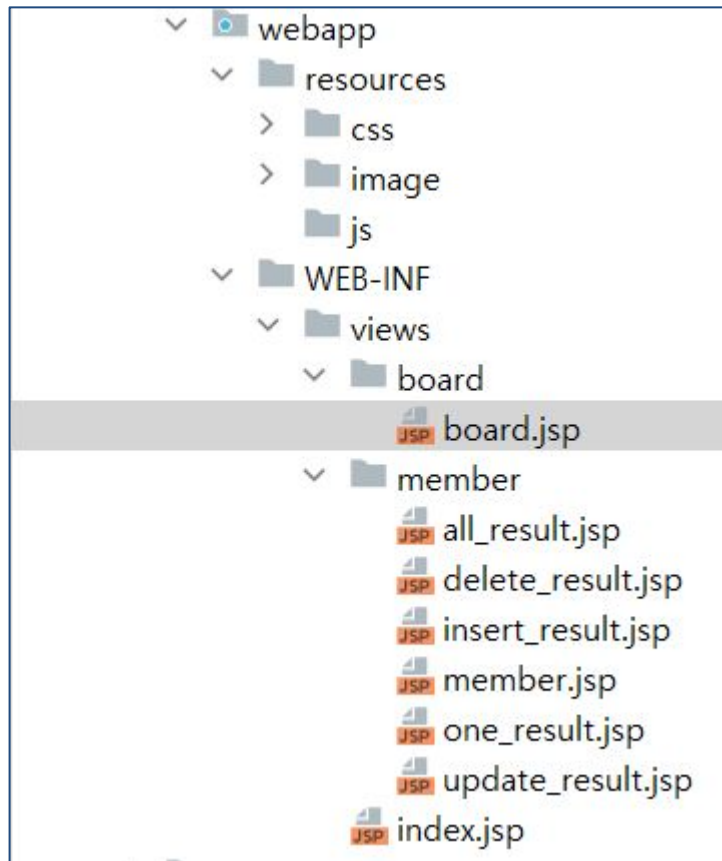
ID	Password
회원가입	<input type="text"/> <input type="password"/> <input type="password"/> <input type="text"/> <input type="text"/> <input type="button" value="Add Member"/>
회원탈퇴	<input type="text" value="apple"/> <input type="button" value="Delete Member"/>
회원정보 수정	<p>Update Member by ID</p> <input type="text"/> <p>New Telephone</p> <input type="text"/> <input type="button" value="Update Member"/>
회원정보 검색	<p>Search Member by ID</p> <input type="text" value="apple"/> <input type="button" value="Search"/>
회원 전체 검색	<input type="button" value="Search All"/>

member.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Member CRUD Form</title>
  <link rel="stylesheet" href="../resources/css/out.css">
</head>
<body>
<div class="container">
  <h2>Member CRUD Form</h2>

  <!-- Member Table -->
  <table>
    <thead>
      <tr>
        <th>항목명</th>
        <th>입력폼</th>
      </tr>
    </thead>
    <tbody>
      <!-- Sample data row for demonstration -->
      <tr>
        <td class="center">회원가입</td>
        <td>
          <!-- Member Insert Form -->
          <form action="member/insert" method="post">
            <label for="id">ID</label>
            <input type="text" id="id" name="id" value="apple"
required>
            <label for="pw">Password</label>
            <input type="password" id="pw" name="pw" value="1234"
required>
            <label for="name">Name</label>
            <input type="text" id="name" name="name" value="apple">
            <label for="tel">Telephone</label>
            <input type="text" id="tel" name="tel" value="011">
            <button type="submit">Add Member</button>
          </form>
        </td>
      </tr>
    </tbody>
  </table>
</div>
</body>
</html>
```

```
<tr>
  <td class="center">회원탈퇴</td>
  <td>
    <!-- Member Delete Form -->
    <form action="member/delete" method="post">
      <input type="text" name="id" value="apple">
      <button type="submit" class="btn-edit">Delete Member</button>
    </form>
  </td>
</tr>
<tr>
  <td class="center">회원정보 수정</td>
  <td>
    <!-- Member Update Form -->
    <form action="member/update" method="post">
      <label for="id">Update Member by ID</label>
      <input type="text" id="id" name="id" required>
      <label for="tel">New Telephone</label>
      <input type="text" id="tel" name="tel">
      <button type="submit">Update Member</button>
    </form>
  </td>
</tr>
<tr>
  <td class="center">회원정보 검색</td>
  <td>
    <!-- Member Search Form -->
    <form action="member/one" method="get">
      <label for="id">Search Member by ID</label>
      <input type="text" name="id" value="apple" required>
      <button type="submit">Search</button>
    </form>
  </td>
</tr>
<tr>
  <td class="center">회원 전체 검색</td>
  <td>
    <!-- Member All -->
    <a href="member/all">
      <button>Search All</button>
    </a>
  </td>
</tr>
</tbody></table></div>\</body></html>
```



localhost:8080/board

Board CRUD Form

항목명	입력폼
게시판글쓰기	<p>title</p> <input type="text"/> <p>content</p> <input type="text"/> <p>Writer</p> <input type="text"/> <p>Add Board</p>
게시판 글 삭제	<p>Delete Board</p>
게시판 글 수정	<p>Update Board by NO</p> <input type="text"/> <p>New Content</p> <input type="text"/> <p>Update Board</p>
게시판 정보 검색	<p>Search Board by NO</p> <input type="text"/> <p>Search</p>
게시판 전체 검색	<p>Search All</p>

insert/update/delete

회원가입

ID

apple

Password

....

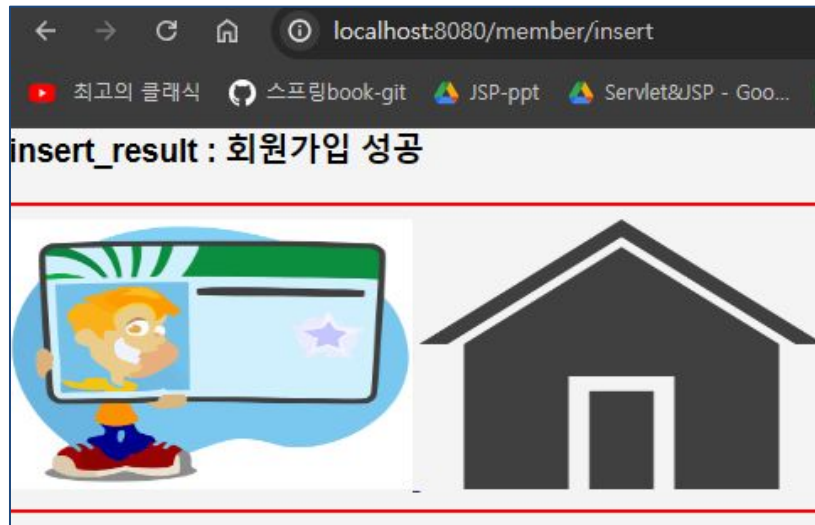
Name

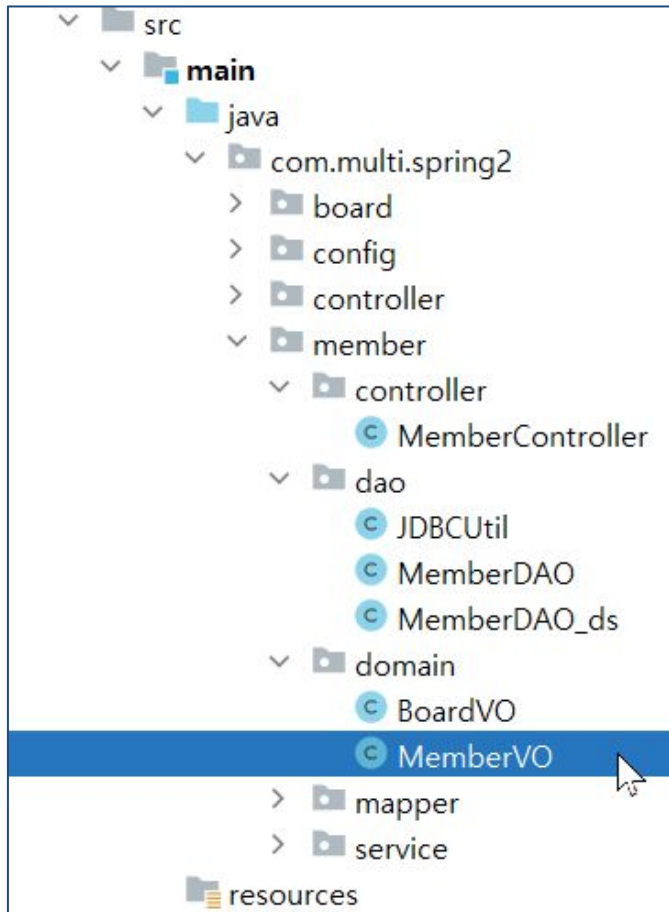
apple

Telephone

011

Add Member





```
package com.multi.spring3.member.domain;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class MemberVO {
```

```
    private String id;
    private String pw;
    private String name;
    private String tel;
```

```
    /*
        use shop2;
        CREATE TABLE `member` (
            `id` varchar(45) NOT NULL,
            `pw` varchar(45) NOT NULL,
            `name` varchar(45) DEFAULT NULL,
            `tel` varchar(45) DEFAULT NULL,
            PRIMARY KEY (`id`)
        );
```

```
        desc member;
```

```
    */
```

```
}
```

spring3 C:\wkb_spring\spring3

> .gradle

> .idea

> build

> gradle

src

main

java

com.multi.spring3

> board

> config

> controller

member

> controller

dao

MemberDAO

> domain

> mapper

> service

@Repository

public class MemberDAO {

private final SqlSessionTemplate sqlSessionTemplate;

@Autowired

public MemberDAO(SqlSessionTemplate sqlSessionTemplate) {

this.sqlSessionTemplate = sqlSessionTemplate;

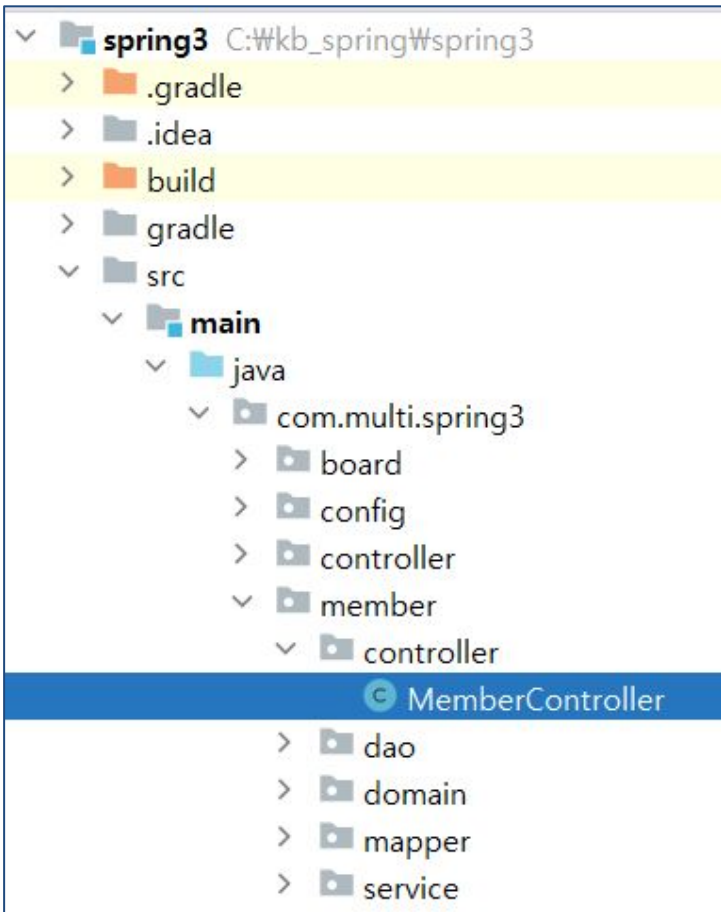
}

public int insert(MemberVO memberVO) {

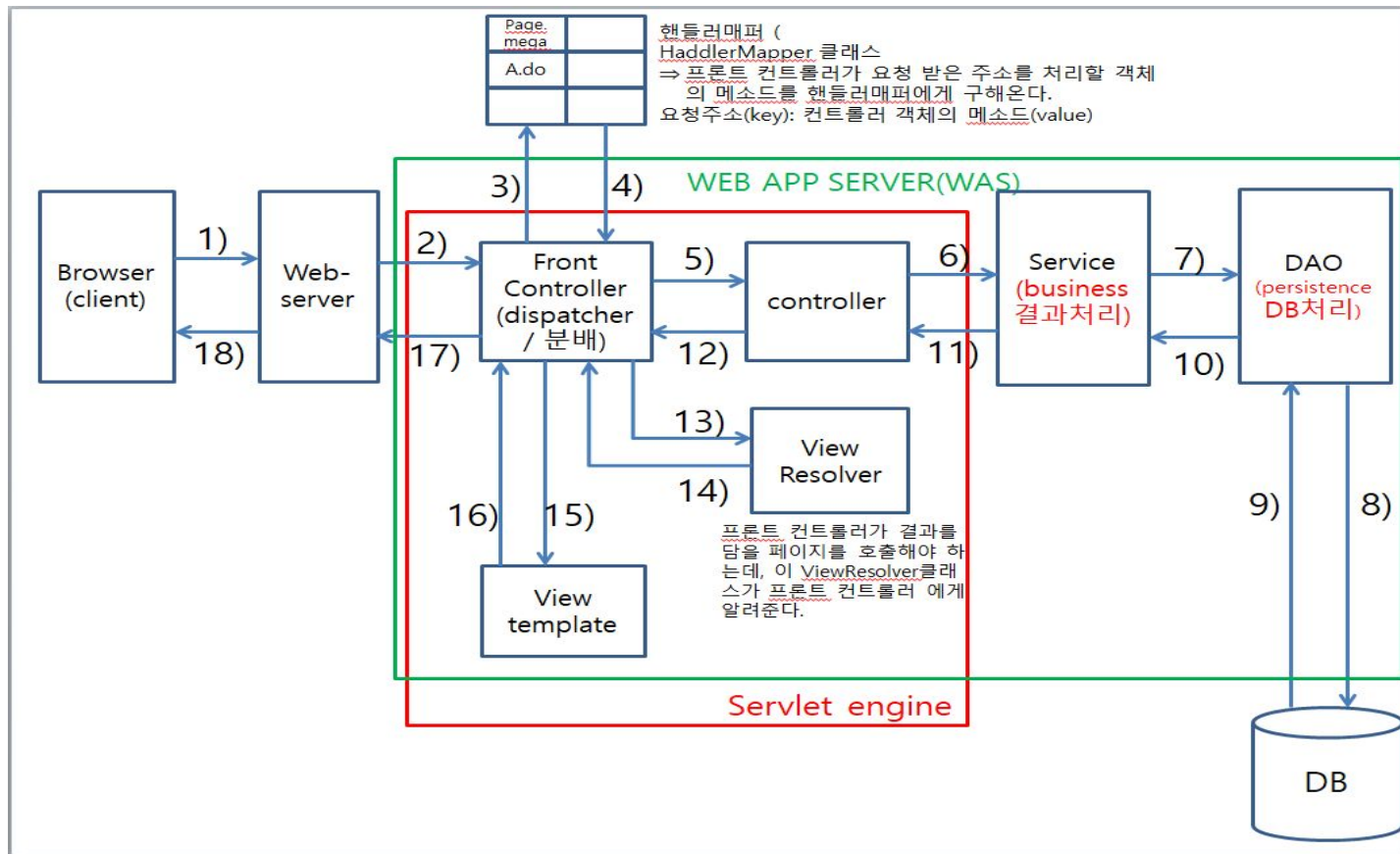
return

sqlSessionTemplate.getMapper(MemberMapper.class).insert(memberVO);

}



```
@PostMapping("/insert")
public ModelAndView insert(MemberVO memberVO) {
    String result = memberService.insert(memberVO);
    System.out.println("----->> " + result);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("result", result);
    modelAndView.setViewName("member/insert_result");
    return modelAndView;
}
```



controller

```
ModelAndView modelAndView = new ModelAndView();
```

```
modelAndView.addObject("result", result);
```

→ 모델 속성으로 지정 (이름, 값);

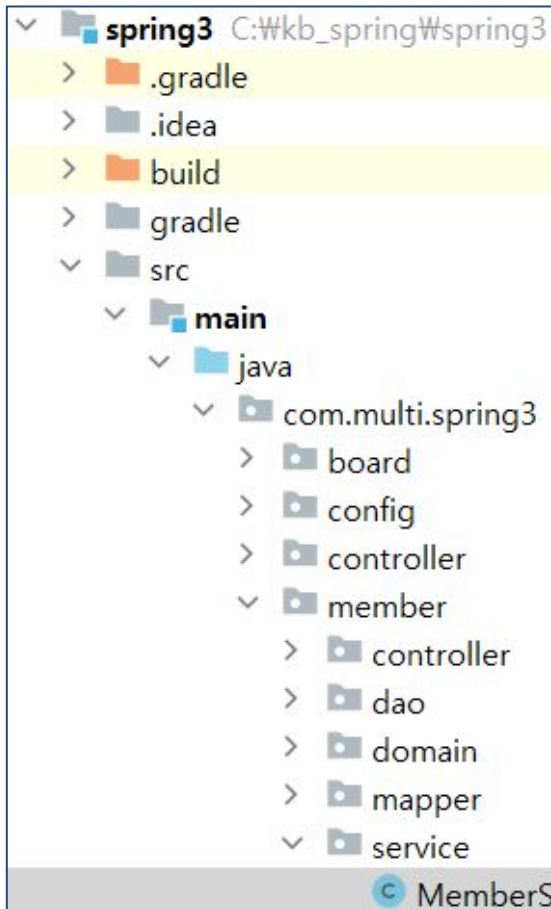
```
modelAndView.setViewName("member/insert_result");
```

→ 결과를 출력할 views/jsp파일 이름

views/jsp

```
<%  
    String result = (String) request.getAttribute("result");  
%>  
<h3>insert_result : <%= result %></h3>
```

\${result} → 모델 속성으로 지정한 이름 출력



```
package com.multi.spring3.member.service;

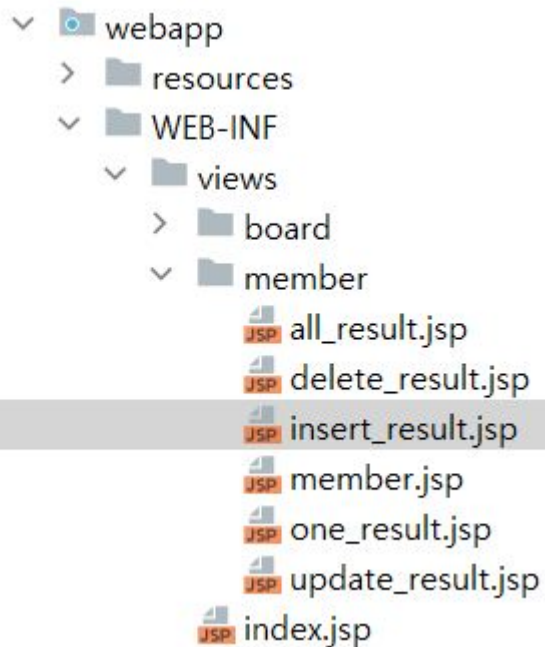
import com.multi.spring3.member.dao.MemberDAO;
import com.multi.spring3.member.domain.MemberVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class MemberService {

    private final MemberDAO memberDAO;

    @Autowired
    public MemberService(MemberDAO memberDAO) {
        this.memberDAO = memberDAO;
        System.out.println("MemberService created");
    }

    public String insert(MemberVO memberVO) {
        String result = "회원가입 실패";
        if (memberDAO.insert(memberVO) == 1) {
            result = "회원가입 성공";
        };
        return result;
    }
}
```



```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Member CRUD Form</title>
    <link rel="stylesheet" href="../resources/css/out.css">
</head>
<body>
<%
    String result = (String)request.getAttribute("result");
%>
    <h3>insert result : <%= result %></h3>
    <hr color="red">
    <a href="/member">
        
    </a>
    <a href="/">
        
    </a>
    <hr color="red">
</body>
</html>
```

회원정보 수정

Update Member by ID

apple2

New Telephone

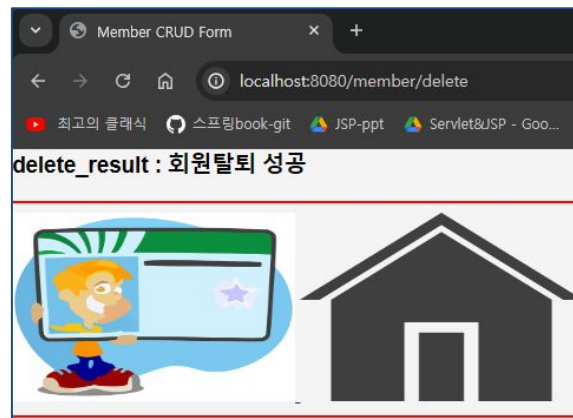
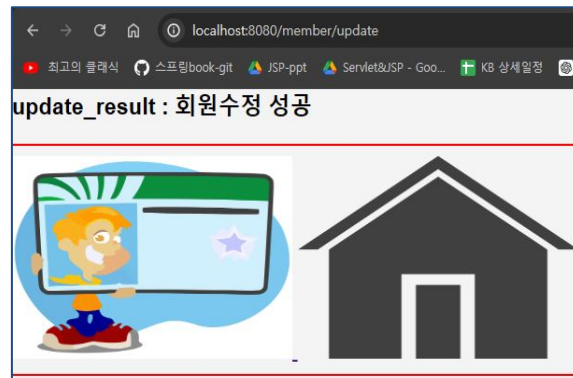
9999

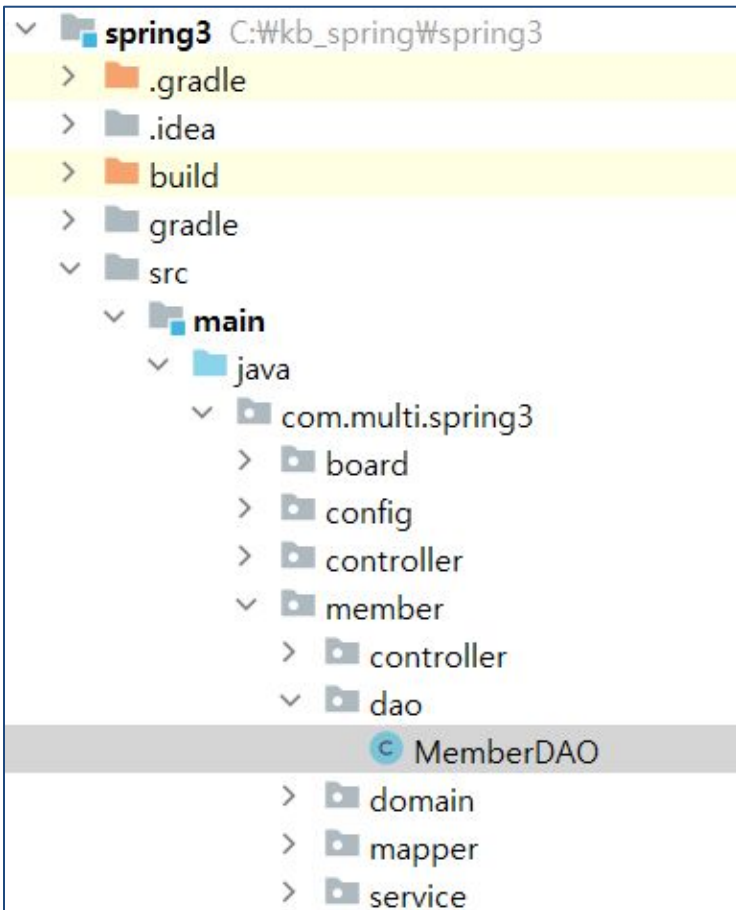
Update Member

회원탈퇴

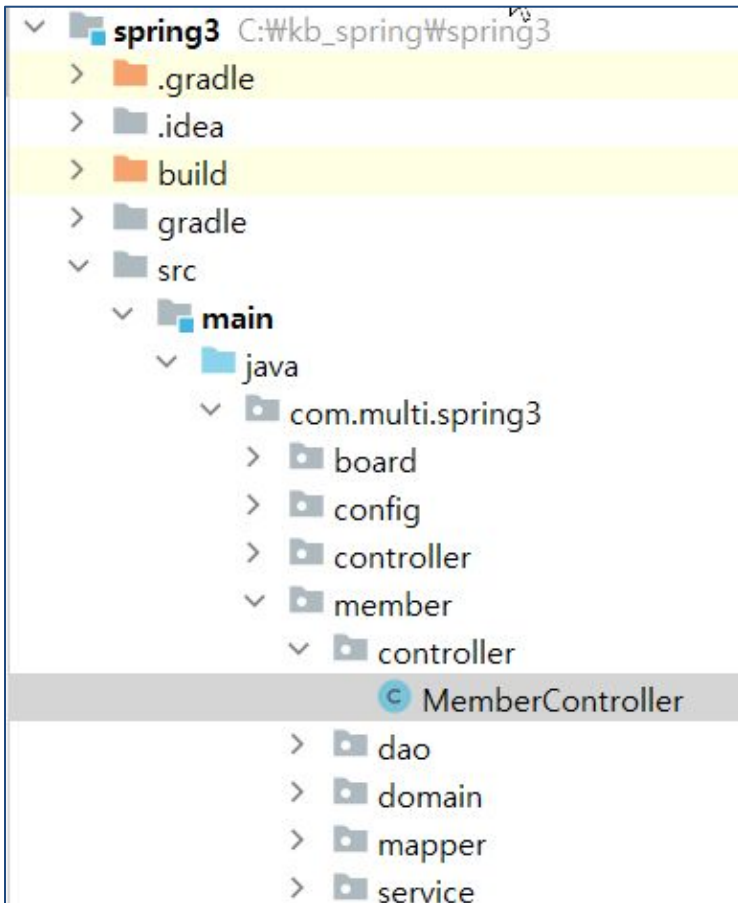
apple

Delete Member



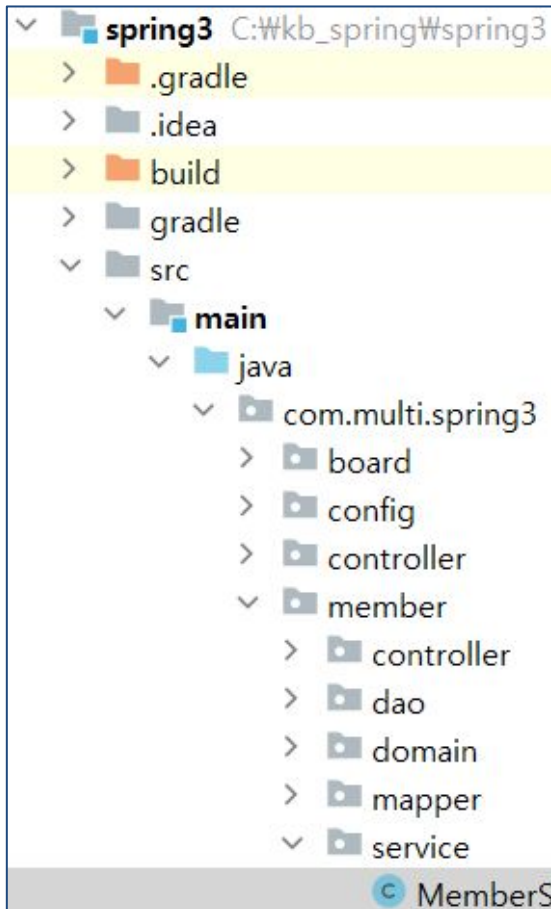


```
public int update(MemberVO memberVO) {  
    return  
    sqlSessionTemplate.getMapper(MemberMapper.class).update(memberVO);  
}  
  
public int delete(String id) {  
    return  
    sqlSessionTemplate.getMapper(MemberMapper.class).delete(id);  
}
```



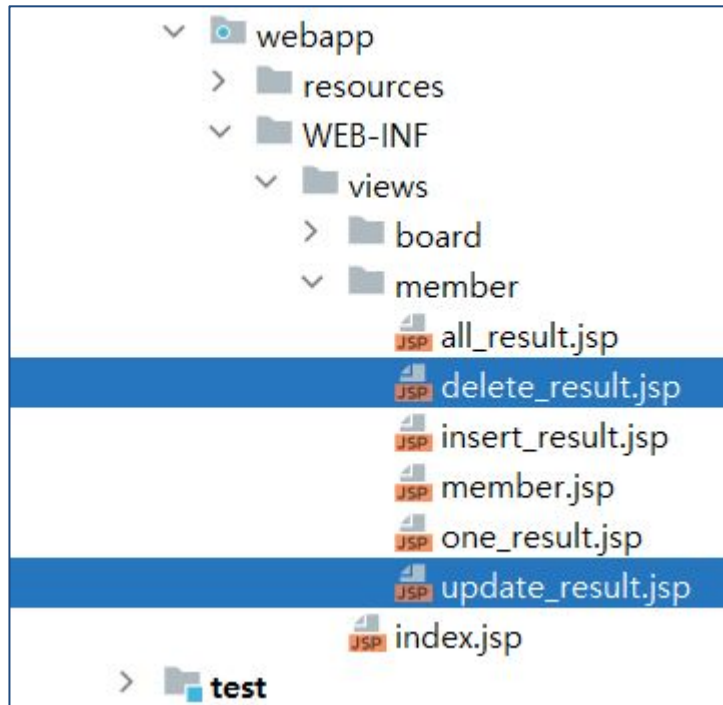
```
@PostMapping ("/update")
public ModelAndView update(MemberVO memberVO) {
    String result = memberService.update(memberVO);
    System.out.println("----->> " + result);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("result", result);
    modelAndView.setViewName("member/update_result");
    return modelAndView;
}

@PostMapping ("/delete")
public ModelAndView delete(@RequestParam("id") String
id) {
    String result = memberService.delete(id);
    System.out.println("----->> " + result);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("result", result);
    modelAndView.setViewName("member/delete_result");
    return modelAndView;
}
```



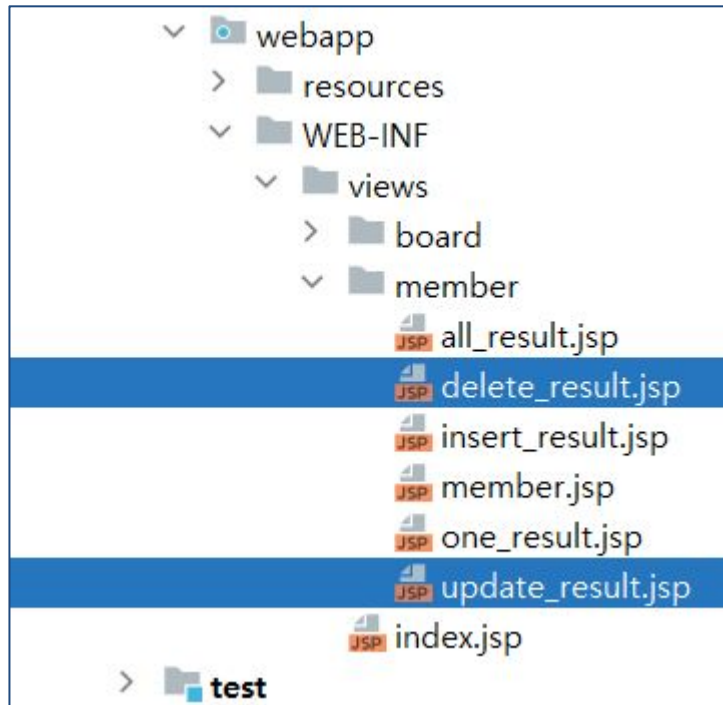
```
public String update(MemberVO memberVO) {  
    String result = "회원수정 실패";  
    if (memberDAO.update(memberVO) != 0) {  
        result = "회원수정 성공";  
    };  
    return result;  
}
```

```
public String delete(String id) {  
    String result = "회원탈퇴 실패";  
    if (memberDAO.delete(id) != 0) {  
        result = "회원탈퇴 성공";  
    };  
    return result;  
}
```



```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Member CRUD Form</title>
  <link rel="stylesheet" href="../resources/css/out.css">
</head>
<body>

<h3>delete_result : ${result}</h3>
<hr color="red">
<a href="/member">
  
</a>
<a href="/">
  
</a>
<hr color="red">
</body>
</html>
```

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Member CRUD Form</title>
  <link rel="stylesheet" href="../resources/css/out.css">
</head>
<body>
  <h3>update_result : ${result}</h3>
  <hr color="red">
  <a href="/member">
    
  </a>
  <a href="/">
    
  </a>
  <hr color="red">
</body>
</html>
```


select

<p>회원정보 검색</p>	<p>Search Member by ID</p> <div data-bbox="722 274 1692 359"> <input type="text" value="apple2"/> </div> <div data-bbox="722 390 892 473"> <p>Search</p> </div>
<p>회원 전체 검색</p>	<div data-bbox="732 607 946 687"> <p>Search All</p> </div>

localhost:8080/member/all

Member List


ID	Password	Name	Telephone
apple2	1234	apple	555
apple3	1234	apple	011
apple4	1234	apple	011
apple5	1234	apple	011
apple6	1234	apple	011

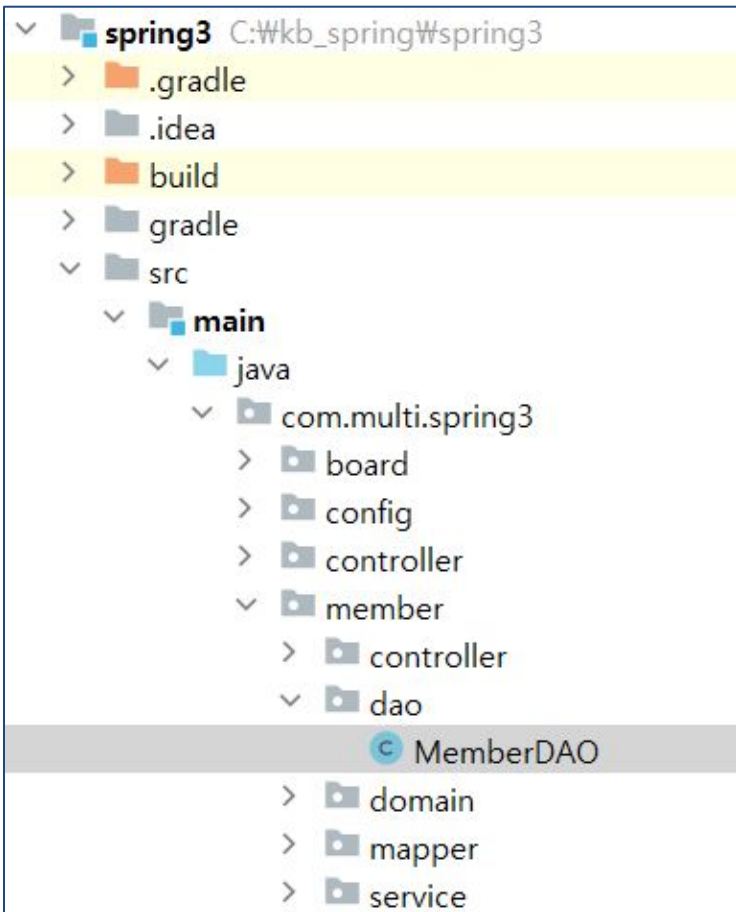


localhost:8080/member/one?id=apple2

가입한 정보 확인

항목명	항목값
가입한 ID	apple2
가입한 PW	1234
가입한 NAME	apple
가입한 TEL	555

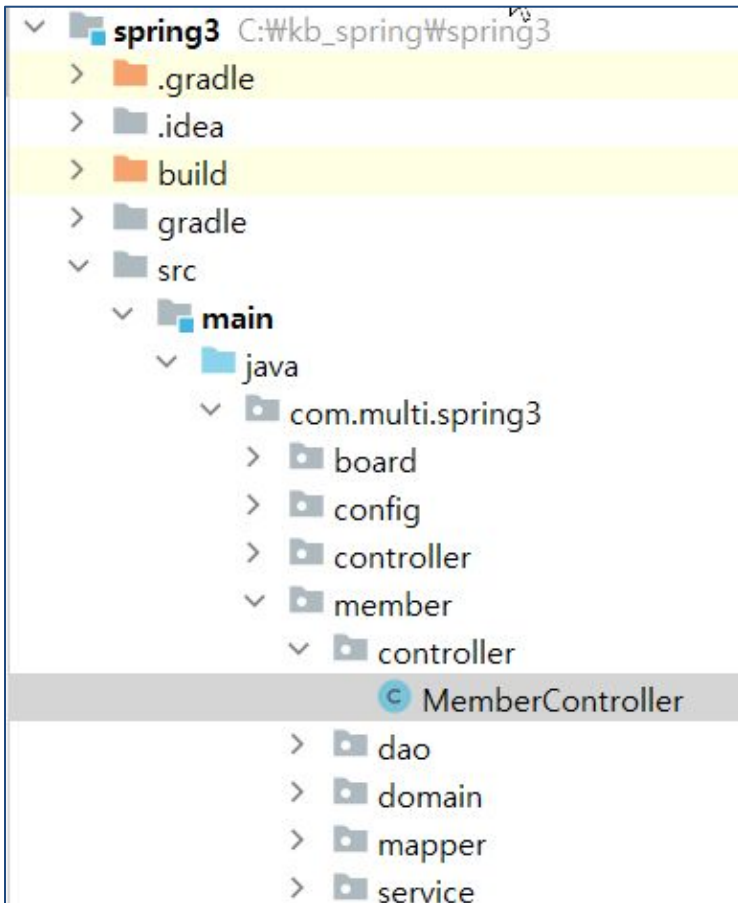




```

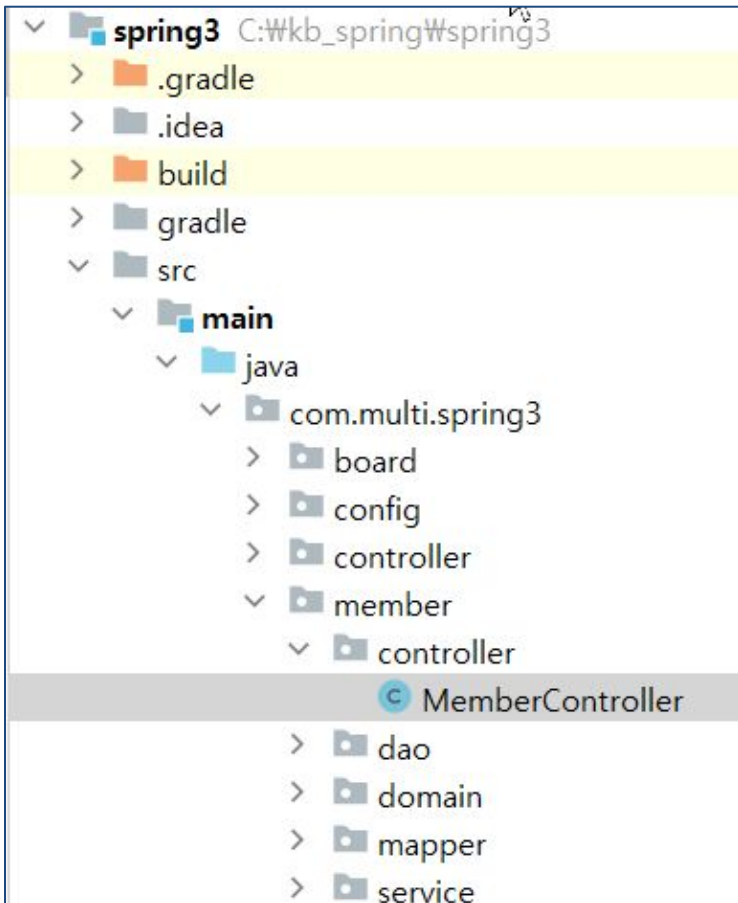
public MemberVO one(String id) {
    return sqlSessionTemplate.getMapper(MemberMapper.class).one(id);
}

public List<MemberVO> all() {
    return sqlSessionTemplate.getMapper(MemberMapper.class).all();
}
  
```



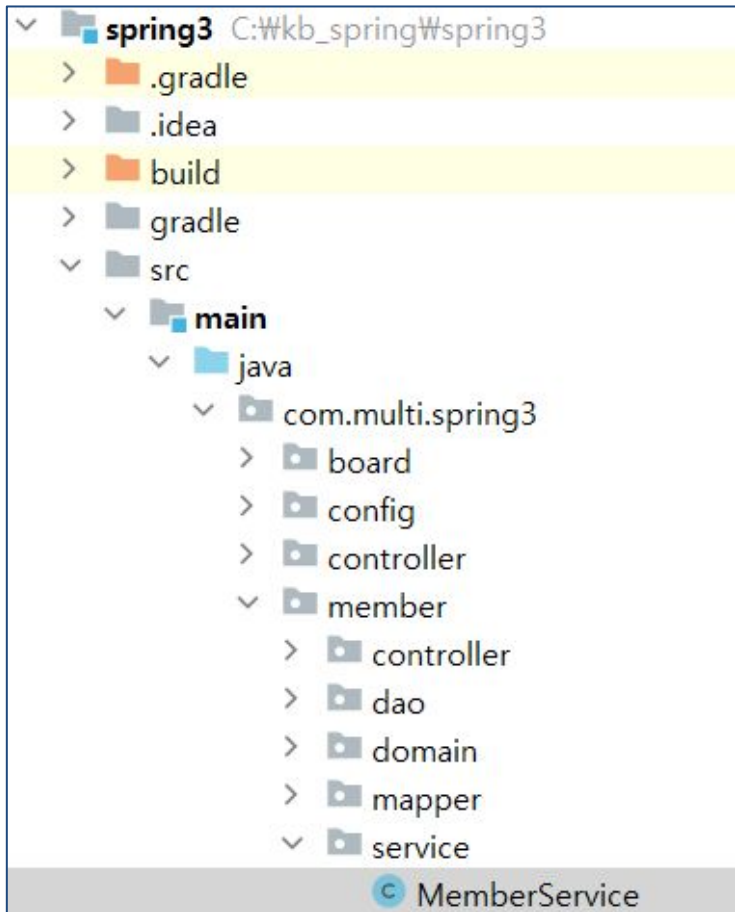
```
@GetMapping("/one")
public ModelAndView one(String id) {
    MemberVO memberVO = memberService.one(id);
    System.out.println("----->> " + memberVO);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("memberVO", memberVO);
    modelAndView.setViewName("member/one_result");
    return modelAndView;
}

@GetMapping("/all")
public ModelAndView all() {
    List<MemberVO> all = memberService.all();
    System.out.println("----->> " + all);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("all", all);
    modelAndView.setViewName("member/all_result");
    return modelAndView;
}
```



```
@PostMapping("/update")
public ModelAndView update(MemberVO memberVO) {
    String result = memberService.update(memberVO);
    System.out.println("----->> " + result);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("result", result);
    modelAndView.setViewName("member/update_result");
    return modelAndView;
}

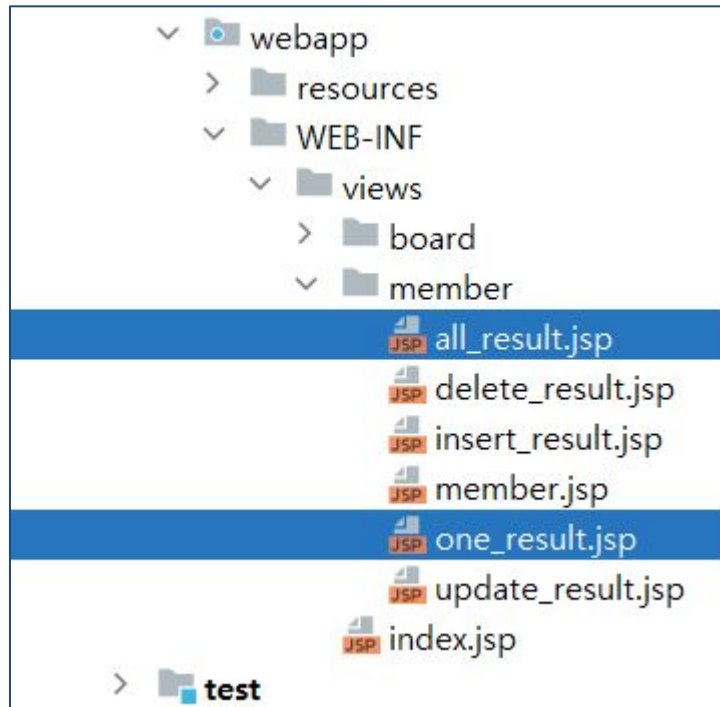
@PostMapping("/delete")
public ModelAndView delete(@RequestParam("id") String id) {
    String result = memberService.delete(id);
    System.out.println("----->> " + result);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("result", result);
    modelAndView.setViewName("member/delete_result");
    return modelAndView;
}
```



```
public MemberVO one(String name){
    return memberDAO.one(name);
}

public List<MemberVO> all(){
    return memberDAO.all();
}
```


one/all 회원 정보 검색



```
<body>

<%
    MemberVO memberVO = (MemberVO)request.getAttribute("memberVO");
%>
<div class="container">

    <h2>가입한 정보 확인</h2>
    <table>
        <thead>
            <tr>
                <th>항목명</th>
                <th>항목값</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>가입한 ID</td>
                <td><%= memberVO.getId() %></td>
            </tr>
            <tr>
                <td>가입한 PW</td>
                <td><%= memberVO.getPw() %></td>
            </tr>
            <tr>
                <td>가입한 NAME</td>
                <td><%= memberVO.getName() %></td>
            </tr>
            <tr>
                <td>가입한 TEL</td>
                <td><%= memberVO.getTel() %></td>
            </tr>
            <tr>
                <td colspan="2">
                    <hr color="red">
                    <a href="/member">
                        
                    </a>
                    <a href="/">
                        
                    </a>
                    <hr color="red">
                </td>
            </tr>
        </tbody>
    </table>
</div>
</body>
```

```

<body>

<div class="container">
  <h1>Member List</h1>
  <table>
    <thead>
      <tr>
        <th>ID</th>
        <th>Password</th>
        <th>Name</th>
        <th>Telephone</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach var="member"
items="${all}">
        <c:if test="${member != null}">
          <tr>
            <td>${member.id}</td>
            <td>${member.pw}</td>
            <td>${member.name}</td>
            <td>${member.tel}</td>
          </tr>
        </c:if>

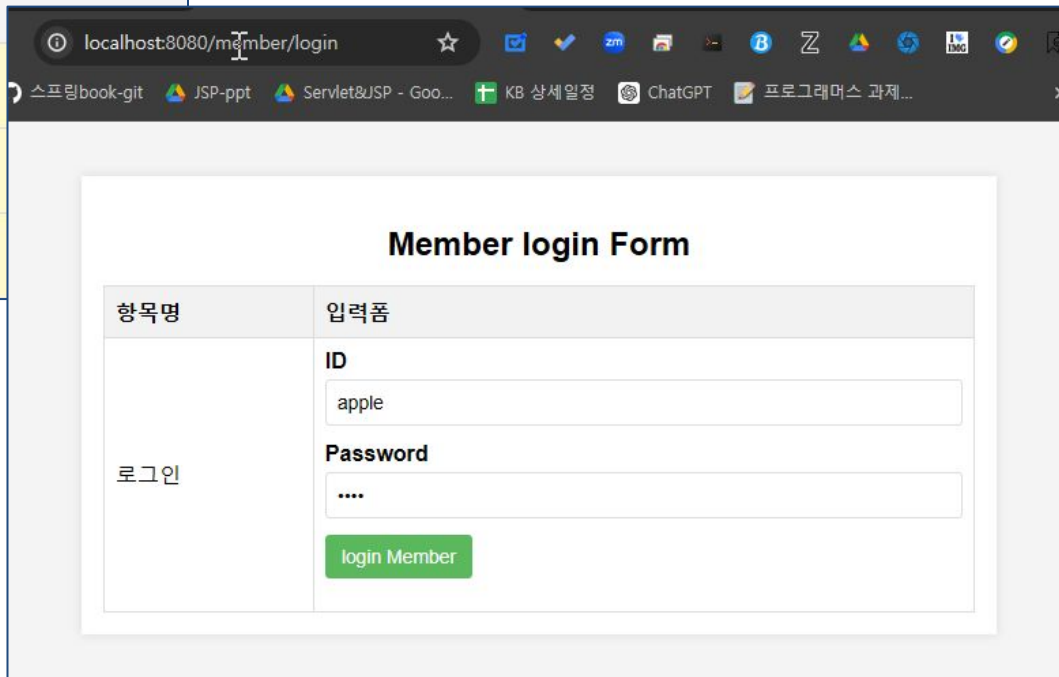
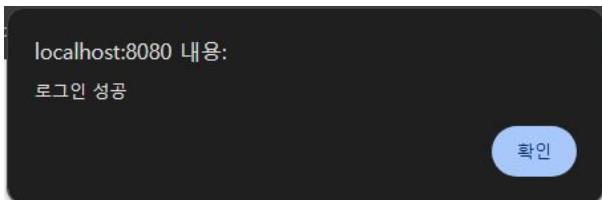
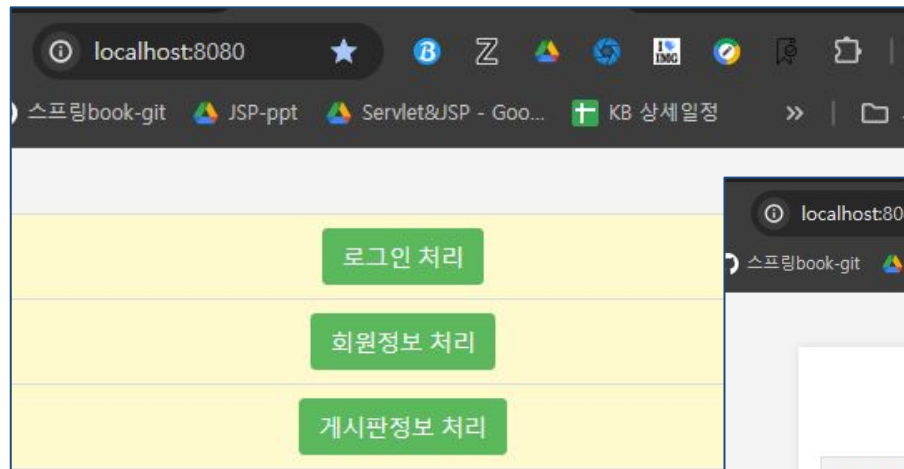
```

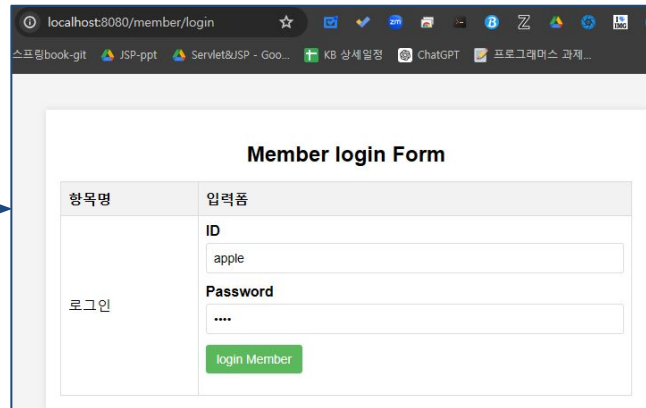
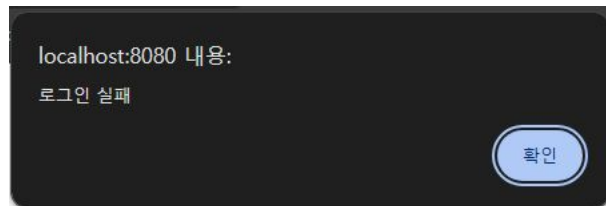
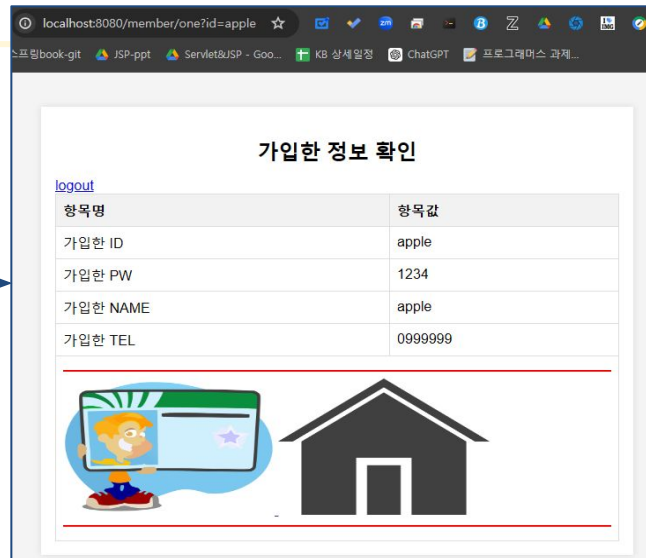
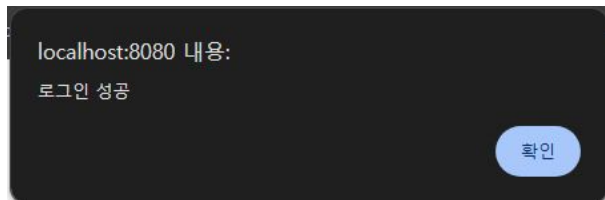
```

<c:if test="${member == null}">
  <tr>
    <td colspan="4">No members found</td>
  </tr>
</c:if>
</c:forEach>
<tr>
  <td colspan="4">
    <hr color="red">
    <a href="/member">
      
    </a>
    <a href="/">
      
    </a>
    <hr color="red">
  </td>
</tr>
</tbody>
</table>
</div>
</body>

```

login/logout





```
@GetMapping("/login")
public String login() {
    return "member/login";
}

@PostMapping("/loginProcess")
public String loginProcess(MemberVO memberVO, HttpSession session) {
    int result = memberService.login(memberVO);
    System.out.println("----->> " + result);
    if (result == 1) {
        session.setAttribute("login_id", memberVO.getId());
        return "member/login_ok";
    } else {
        return "member/login_no";
    }
}

@GetMapping("/logout")
public String logout(HttpSession session) {
    //세션끊기
    session.invalidate();
    return "redirect:/";
}
```

xml

```
<select id="login"
        parameterType="com.multi.spring3.member.domain.MemberVO"
        resultType="int">
    select count(id) from member
    where id = #{id} and pw = #{pw}
</select>
```

interface

```
@Mapper
public interface MemberMapper {
    //mapper의 id와 맞아야함.
    int insert(MemberVO memberVO); //<insert id="insert" ~~>
    int update(MemberVO memberVO);
    int delete(String id);
    MemberVO one(String id); // <select id="one" ~~>
    List<MemberVO> all();
    int login(MemberVO memberVO);
}
```

Service

```
public int login(MemberVO memberVO) {  
    return memberDAO.login(memberVO);  
}
```

DAO

```
public int login(MemberVO memberVO) {  
    return sqlSessionTemplate.getMapper(MemberMapper.class).login(memberVO);  
}
```


login.jsp

```
<div class="container">
  <h2>Member login Form</h2>

  <!-- Member Table -->
  <table>
    <thead>
      <tr>
        <th>항목명</th>
        <th>입력폼</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td class="center">로그인</td>
        <td>
          <!-- Member login Form -->
          <form action="loginProcess" method="post">
            <label for="id">ID</label>
            <input type="text" id="id" name="id" value="apple" required>
            <label for="pw">Password</label>
            <input type="password" id="pw" name="pw" value="1234" required>
            <button type="submit">login Member</button>
          </form>
        </td>
      </tr>
    </tbody>
  </table>
</div>
```

index.jsp

```
<tr>
  <td style="background: lemonchiffon; text-align: center">
    <a href="member/login">
      <button>로그인 처리</button>
    </a>
  </td>
</tr>
```

login_no.jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Member CRUD Form</title>
  <link rel="stylesheet" href="../../resources/css/out.css">
</head>
<body>
  <script>
    alert("로그인 실패");
    location.href = "login";
  </script>
</body>
</html>
```

login_ok.jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Member CRUD Form</title>
  <link rel="stylesheet" href="../../resources/css/out.css">
</head>
<body>
  <script>
    alert("로그인 성공");
    location.href = "one?id=${login_id}";
  </script>
</body>
</html>
```

search

localhost:8080

최고의 클래식 스프링book-git JSP-ppt Servlet&JSP - Goo...

로그인 처리

find id :

1

id 포함 검색

회원정보 처리

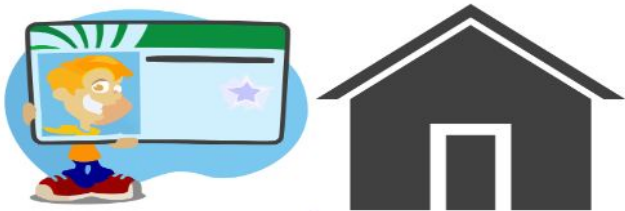
게시판정보 처리

localhost:8080/member/find?word=1

스프링book-git JSP-ppt Servlet&JSP - Goo... KB 상세일정 ChatGPT 프로그래머스 과제...

Member find List

ID	Password	Name	Telephone
apple14	1234	í□□ê,□	011
apple15	1234	appleí□□ê,□	011
apple16	1234	appleí□□ê,□	011
apple17	1234	appleí□□ê,□	011
apple18	1234	apple	011
apple19	1234	appleí□□ê,□	011



```
@GetMapping("/find")
public ModelAndView find(@RequestParam("word") String word) {
    List<MemberVO> all = memberService.find(word);
    System.out.println("----->> " + all);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("all", all);
    modelAndView.setViewName("member/find_result");
    return modelAndView;
}
```

xml

```
<select id="find"
        parameterType="String"
        resultType="com.multi.spring3.member.domain.MemberVO">
    select * from member
    where id LIKE CONCAT('%', #{word}, '%')
</select>
```

interface

```
@Mapper
public interface MemberMapper {
    //mapper의 id와 맞아야함.
    int insert(MemberVO memberVO); //<insert id="insert" ~~>
    int update(MemberVO memberVO);
    int delete(String id);
    MemberVO one(String id); // <select id="one" ~~>
    List<MemberVO> all();
    int login(MemberVO memberVO);

    List<MemberVO> find(String word);
}
```

Service

```
public List<MemberVO> find(String word) {  
    return memberDAO.find(word);  
}
```

DAO

```
public List<MemberVO> find(String word) {  
    return sqlSessionTemplate.getMapper(MemberMapper.class).find(word);  
}
```

index.jsp

```
<tr>
  <td style="background: lemonchiffon; text-align: center">
    <form action="member/find" method="get">
      find id : <input type="text" name="word" value="1">
      <button>id 포함 검색</button>
    </form>
  </td>
</tr>
```


find_result.jsp

```
<%@ page import="com.multi.spring3.member.domain.MemberVO" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page suffix=".jsp" uri="http://java.sun.com/jsp/jstl/core" %>

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Member CRUD Form</title>
<link rel="stylesheet" href="../resources/css/out.css">
</head>
<body>

<div class="container">
  <h1>Member find List</h1>
  <table>
    <thead>
      <tr>
        <th>ID</th>
        <th>Password</th>
        <th>Name</th>
        <th>Telephone</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach var="member" items="${all}">
        <c:if test="${member != null}">
          <tr>
            <td>${member.id}</td>
            <td>${member.pw}</td>
            <td>${member.name}</td>
            <td>${member.tel}</td>
          </tr>
        </c:if>
        <c:if test="${member == null}">
          <tr>
            <td colspan="4">No members found</td>
          </tr>
        </c:if>
      </c:forEach>
    </tbody>
  </table>
</div>
```

1. spring framework내 db처리

2. myBatis framework

a. 설정 파일 myBatis-config.xml

b. 설정 파일 mapper파일

c. mapper interface

3. SqlSessionTemplate

4. JSTL, EL