SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 삼성 청년 SW 아카데미

Front-End Programming

## 학습목표

- 웹프로젝트의 CRUD를 구성한다.
- REST 설계하는 기법을 익힌다.

Confidential

## 스프링 프레임워크란?

자바 엔터프라이즈 개발을 위한 오픈소스 경량 애플리케이션 프레임워크다.

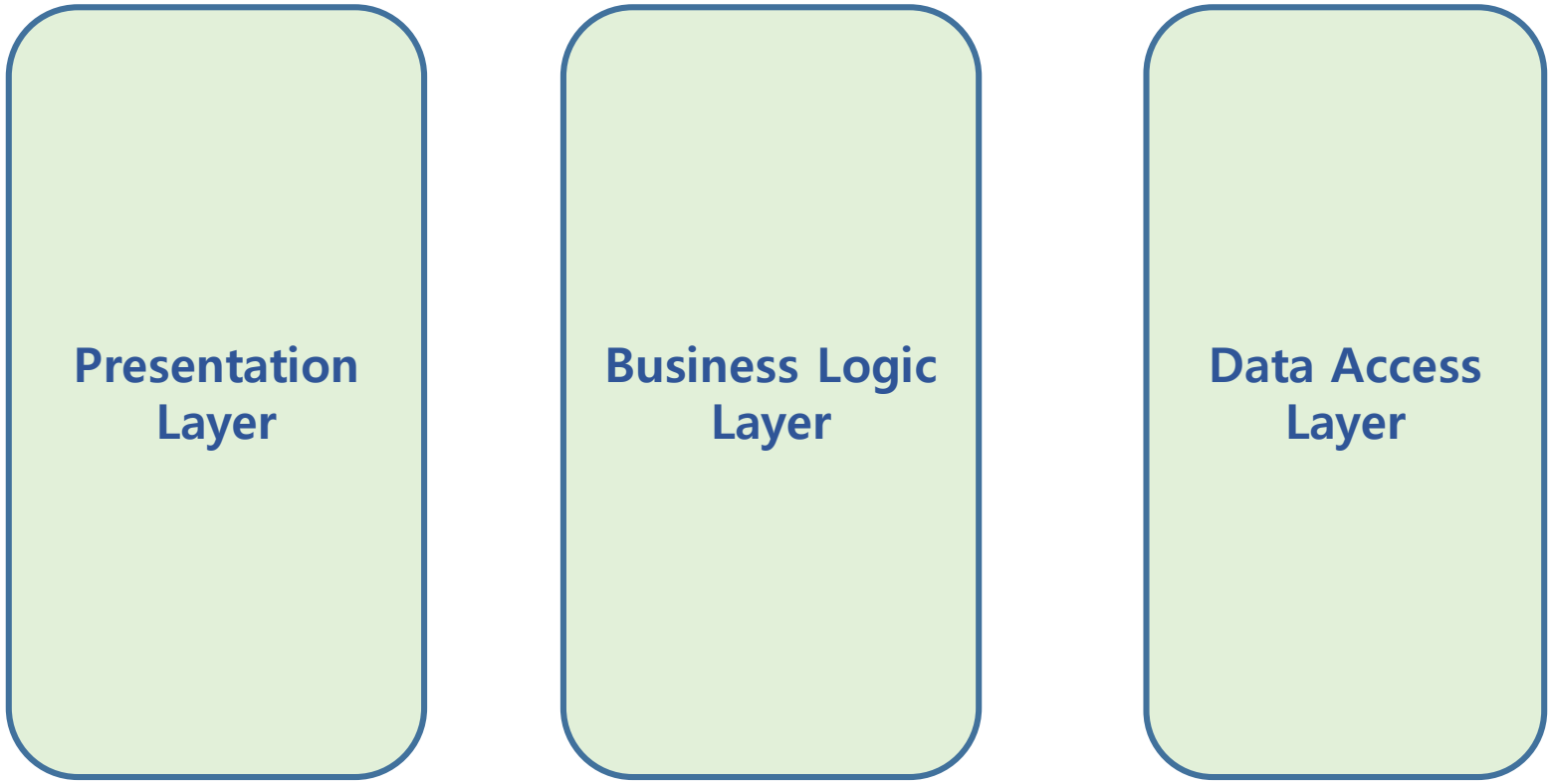
공통 프로그래밍 모델 및 Configuration 모델을 제공한다.

프레임워크가 애플리케이션 수준의 인프라 구조를 제공

개발자가 귀찮은 일에 신경 쓰지 않고 비즈니스 로직 개발에 전념

## 웹 프로젝트 3 레이어(tier)

Confidential



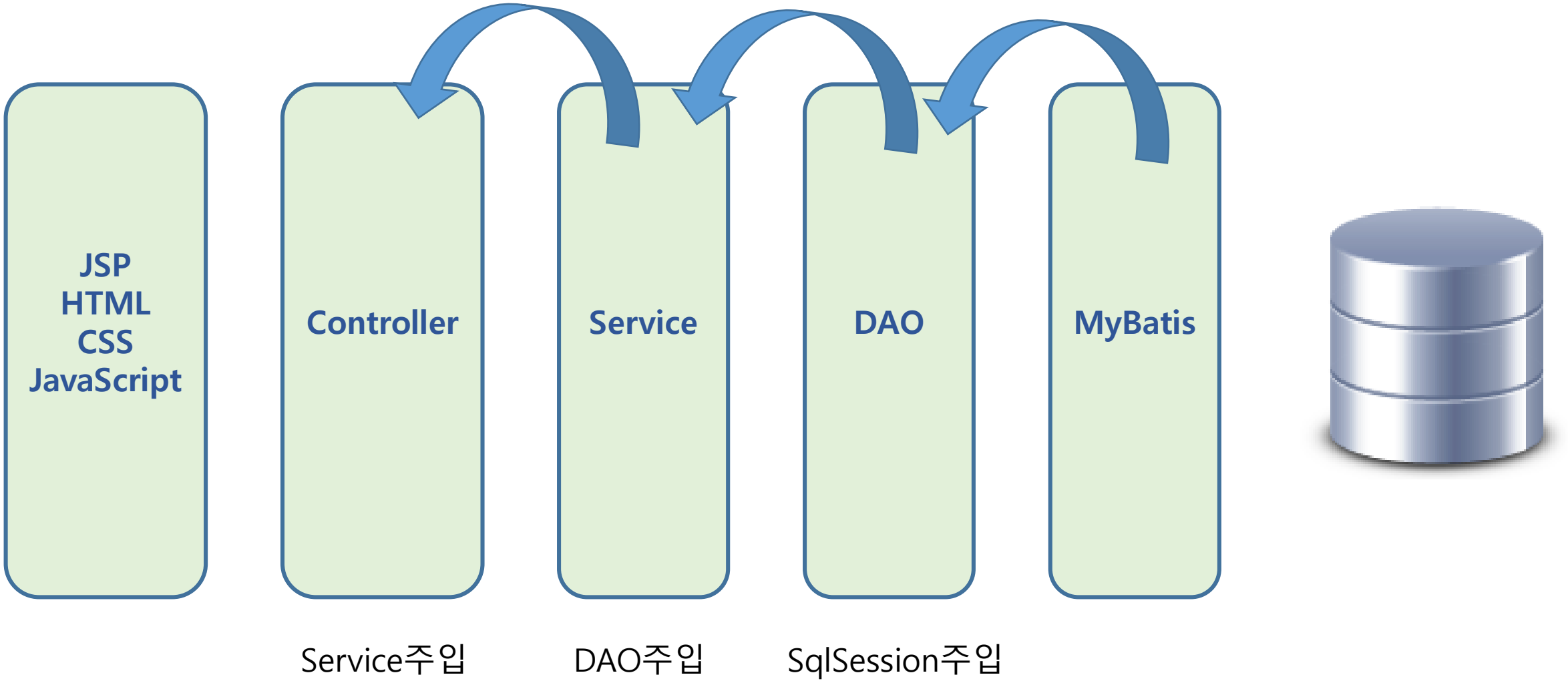

UI담당 구성요소

서비스계층
고객 요구사항반영

Persistence계층
데이터처리

## 세분화된 프로젝트 구성

Confidential



Confidential

# MyBatis

- 개발자가 지정한 SQL, 저장프로시저 그리고 몇가지 고급 매핑을 지원하는 퍼시스턴스 프레임워크
- JDBC코드와 수동으로 셋팅하는 파라미터와 결과 매핑을 제거
- 데이터베이스 레코드에 원시타입과 Map인터페이스 그리고 자바 POJO를 설정하고 매핑하기 위해 XML과 애노테이션을 사용

Confidential

```java
@Override
public MemberDto login(String userid, String userpwd) throws SQLException {
    MemberDto memberDto = null;
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    try {
        conn = DBUtil.getConnection();
        StringBuilder sql = new StringBuilder();
        sql.append("select username, userid, email \n");
        sql.append("from ssafy_member \n");
        sql.append("where userid = ? and userpwd = ?");
        pstmt = conn.prepareStatement(sql.toString());
        pstmt.setString(1, userid);
        pstmt.setString(2, userpwd);
        rs = pstmt.executeQuery();
        if(rs.next()) {
            memberDto = new MemberDto();
            memberDto.setUserid(rs.getString("userid"));
            memberDto.setUsername(rs.getString("username"));
            memberDto.setEmail(rs.getString("email"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
        memberDto = null;
    } finally {
        DBUtil.close(rs);
        DBUtil.close(pstmt);
        DBUtil.close(conn);
    }
    return memberDto;
}
```

MyBatis

```xml
<select id="selectLogin">
    select username, userid, email
        from ssafy_member
        where userid = #{userid} and userpwd = #{userpwd}
</select>
```

Confidential

## ▣ REST(Representational State Transfer)방식

❖ 모바일 시대 이전 : 서버의 데이터를 소비하는 주체는 '브라우저'!!

❖ 모바일 시대 이후 : '브라우저'와 함께 스마트 폰에의 '앱(App)'에서 서버의 데이터를 소비.

✓ 서버는 점점 더 순수하게 데이터에 대한 처리를 목적으로 하는 형태로 진화.

✓ return 'HTML페이지'에서 return '데이터'로 진화!!

## ▣ REST(Representational State Transfer)방식

하나의 URI는 하나의 고유한 리소스(Resource)를 대표하도록 설계된다는 개념에
전송방식을 결합해서 원하는 작업 지정.

URI + POST/GET/PUT/DELETE

이전방식 : /board?action=insert

/board?action=list

**REST방식: /board/123 + POST**

**/board/all + GET**

```
@Conroller
public class MyController{

    @RequestMapping("/gildong")
    public String m1(){
        return "hello";   --->   /WEB-INF/views/hello.jsp 응답
    }

    @RequestMapping("/lime")
    public @ResponseBody String m2(){
        return "hello";   --->  "hello"문자열 응답
    }
}
```

## ▣ REST방식

```
@RestConroller // 컨트롤러내의 모든 요청매핑은
               // JSP페이지가 아닌  데이터(텍스트,JSON,XML,배열)를  응답
(모두 @ResponseBody한것처럼)
               // ==> Ajax요청 전용!!
public class YouController{

    @RequestMapping("/gildong")
    public String m1(){
        return "hello";   ---> "hello"문자열 응답
    }

    @RequestMapping("/lime")
    public String m2(){
        Person vo = new Person("홍길동",13,"학생");
        return vo;        ---> {"name":"홍길동", "age":13, "job":"학생"}
JSON 리턴
    }

}
```

▣ @RestConroller를 사용할 때 특이점

1. 모든 메소드의 요청매핑에 대한 응답은 데이터다!! (text,json,xml데이터를 클라이언트에게 전달)
   ---> @ReponseBody를 명시하지 않아도 모든 메소드의 리턴이 @ResponseBody

2. @PathVariable을 사용하여 요청경로에 데이터를 전달시킬 수 있다.

```
@RequestMapping("/board/{no}/{uname}") //요청경로상에 사용된 {no},{uname}을 '경로변수'라 함!!
        public String getPath(@PathVariable("no")    Integer no,
                                                @PathVariable("uname") String uname){
         //'경로 변수'를 사용하는 경우  {}중괄호의 갯수 만큼 메소드 매개변수에 @PathVariable선언이
있어야 하고
         //'경로 변수'와 같은 이름이 정의되어 있어야 함.
           return  no+"잘했어요!!"+uname;
        }
```

3. 응답데이터로 ReponseEntity<리턴할 데이터의 타입> 를 사용할 수 있다.

```
@RequestMapping("/hello")
public String m1(){
    return "안녕하세요";    ==> 클라이언트(HTML내의 JavaScript)에게 문자열 전달
}

@RequestMapping("/good")
public ReponseEntity<String> m2(){
    return new ResponseEntity<>("좋아요", HttpStatus.OK) ;
    return new ResponseEntity<>("좋아요", HttpStatus.INTERNAL_SERVER_ERROR) ;
                                                    ----------------------------------
                                                              임의로 변경가능

    ==> 클라이언트(HTML내의 JavaScript)에게 문자열뿐만 아니라 조작된 서버의 상태도 전달하는 것이 가능
}
```

4. @PostMapping(value = "/start", consumes = "application/json")

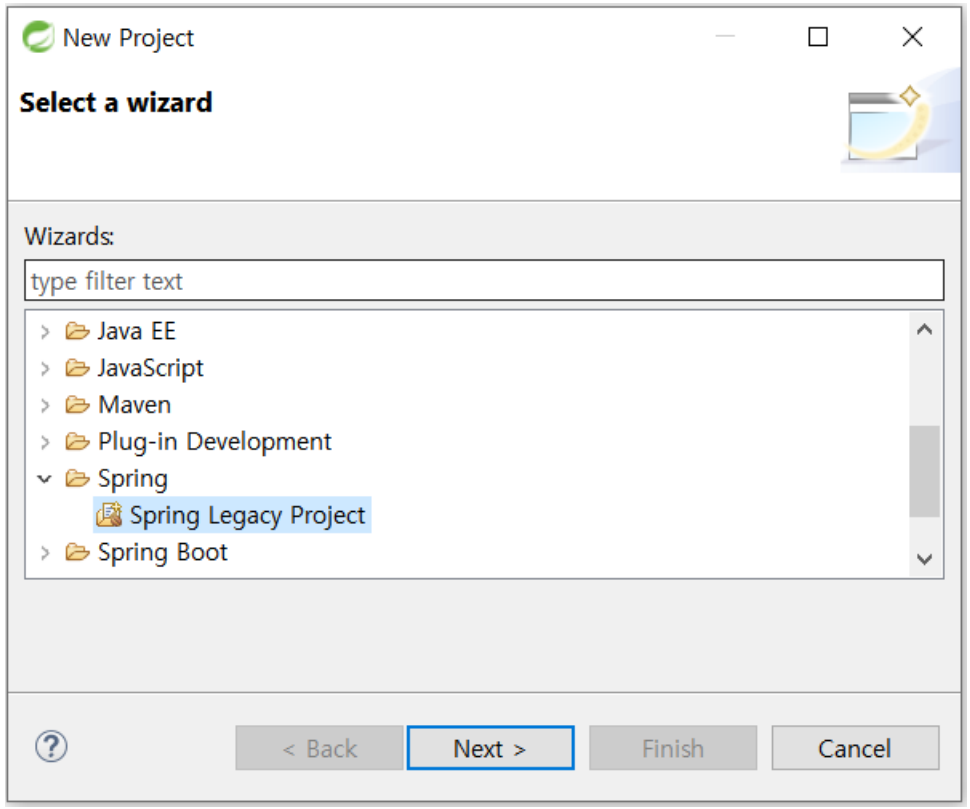
```
public String create(@RequestBody ReplyVO vo) {
  return "시작";
}
```

==> @RequestBody는 클라이언트(내의 JavaScript)가 보낸 JSON데이터를 VO로 변환하는 역할을 한다.

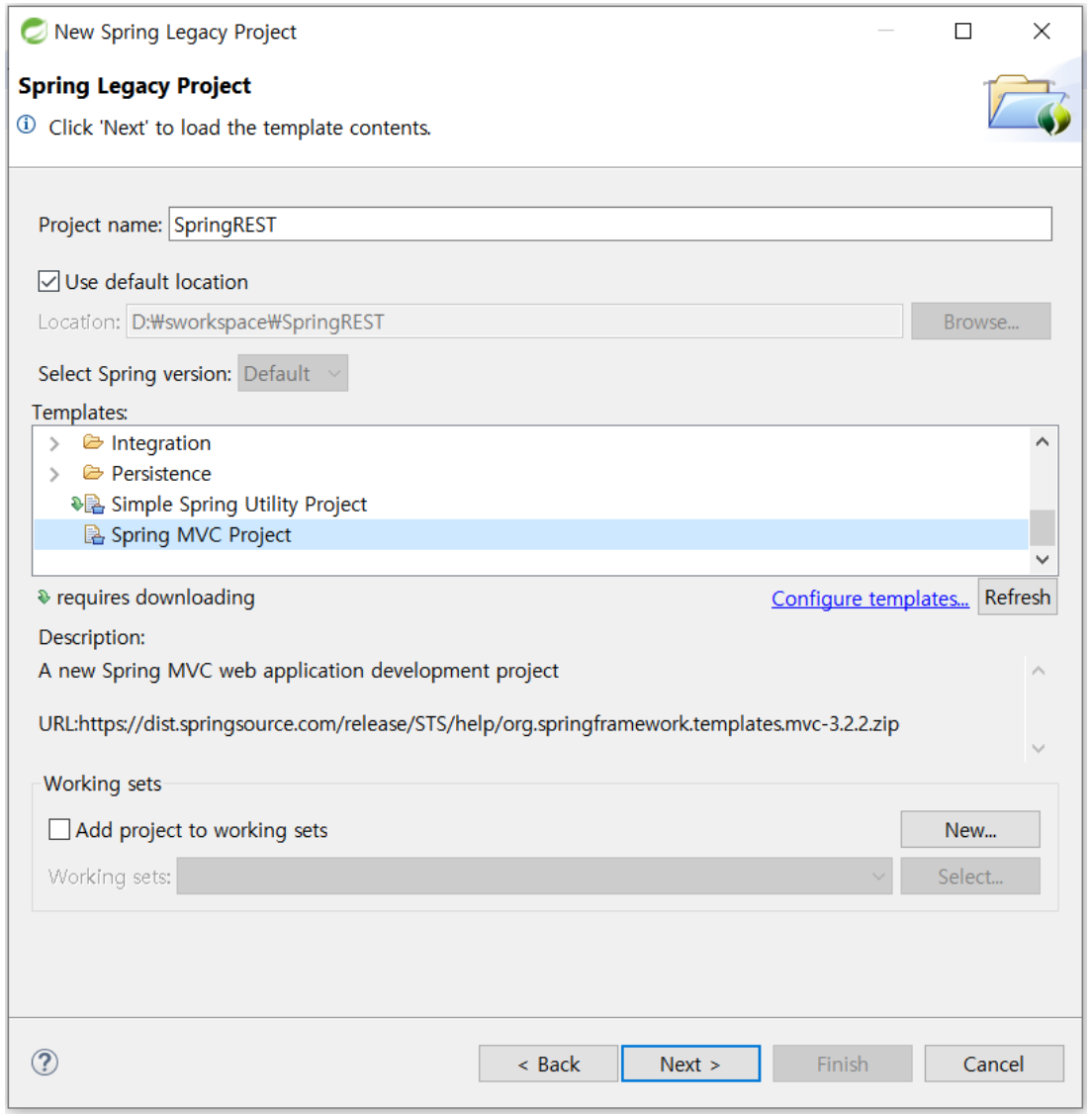
5. pom.xml에 XML 컨버터 추가시 변화

```
<!-- JSON Converter(컨트롤러에서 전달한 VO데이터를 JSON으로 변경하기 위해 사용) -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.9.6</version>
        </dependency>

<!-- XML Converter(컨트롤러에서 전달한 VO데이터를 XML로 변경하기 위해 사용) -->
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-xml</artifactId>
            <version>2.9.6</version>
        </dependency>
```

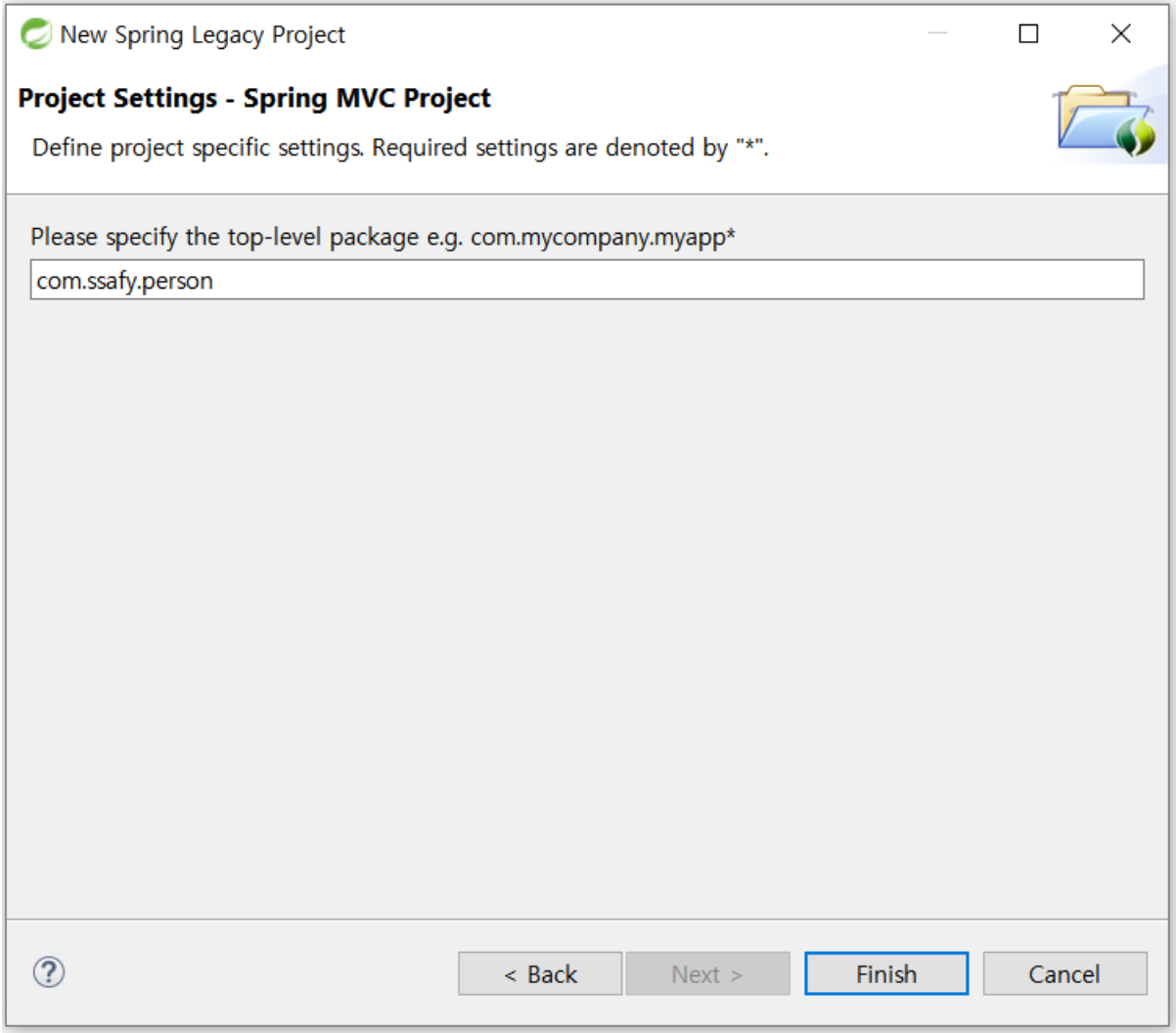
Confidential



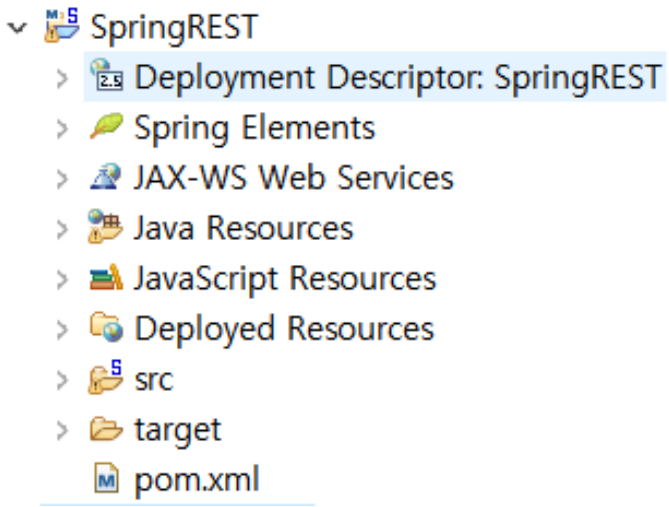
Confidential



Confidential



Confidential



```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.2.6.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

```
<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
```

Confidential

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

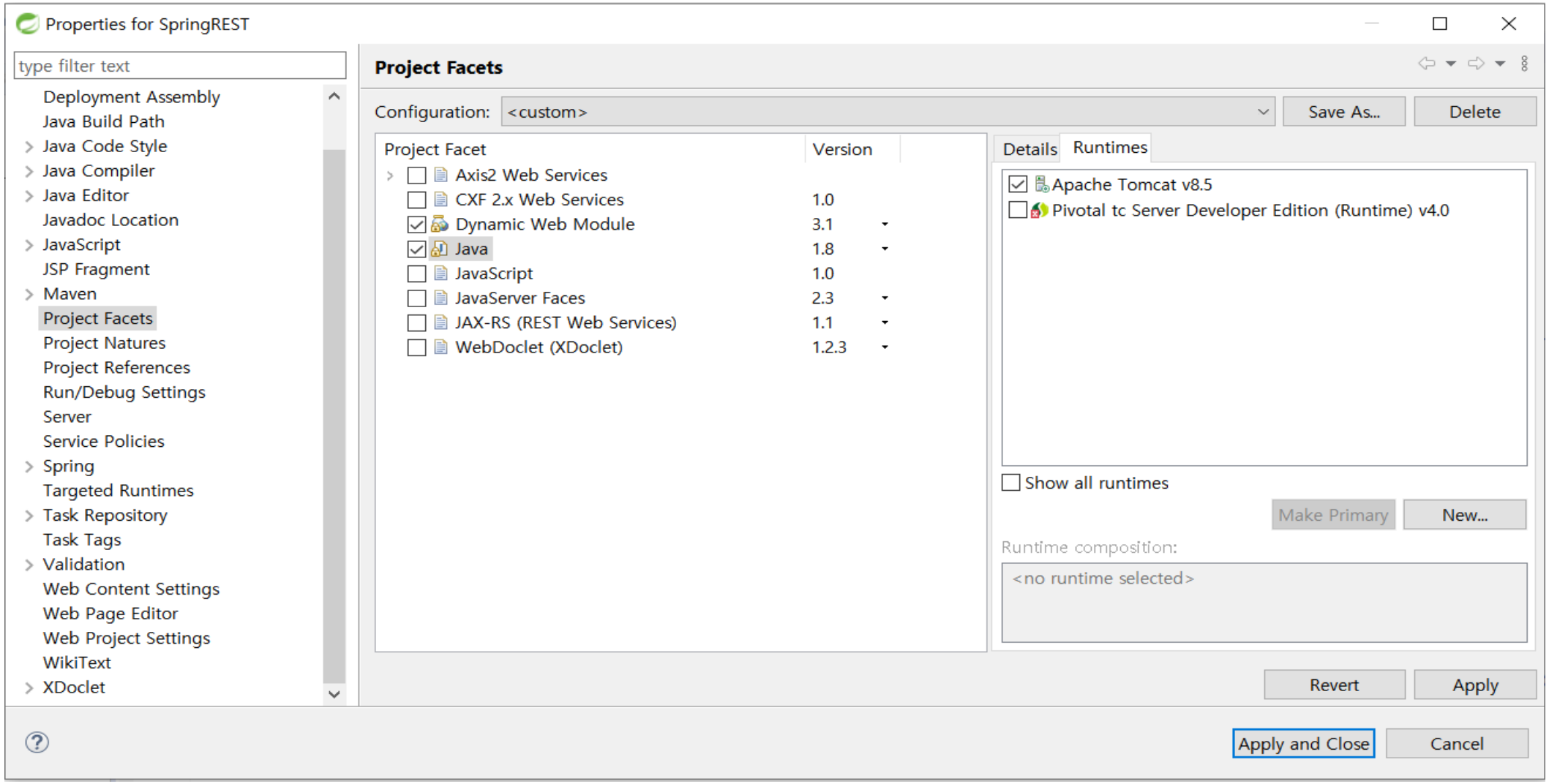
Confidential

```
<!-- DB관련부분 -->

<!-- MySQL 드라이버추가 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.13</version>
</dependency>

<!-- 스프링 JDBC연동 -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
```

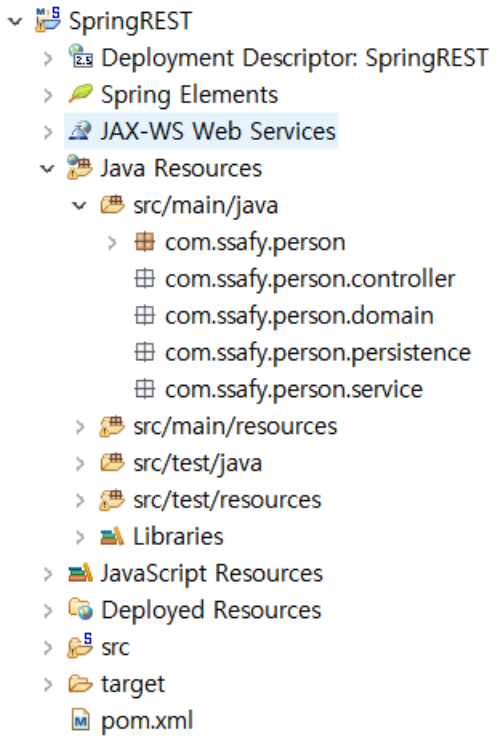
Confidential

```
<!-- MyBatis라이브러리 -->
 <dependency>
   <groupId>org.mybatis</groupId>
   <artifactId>mybatis</artifactId>
   <version>3.4.1</version>
 </dependency>  <!-- mybatis.3.4.1.jar -->


 <!-- 스프링 MyBatis연동 -->
 <dependency>
   <groupId>org.mybatis</groupId>
   <artifactId>mybatis-spring</artifactId>
   <version>1.3.0</version>
 </dependency>

 <!-- 스프링 Transaction 처리 -->
 <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-tx</artifactId>
   <version>${org.springframework-version}</version>
 </dependency>
```

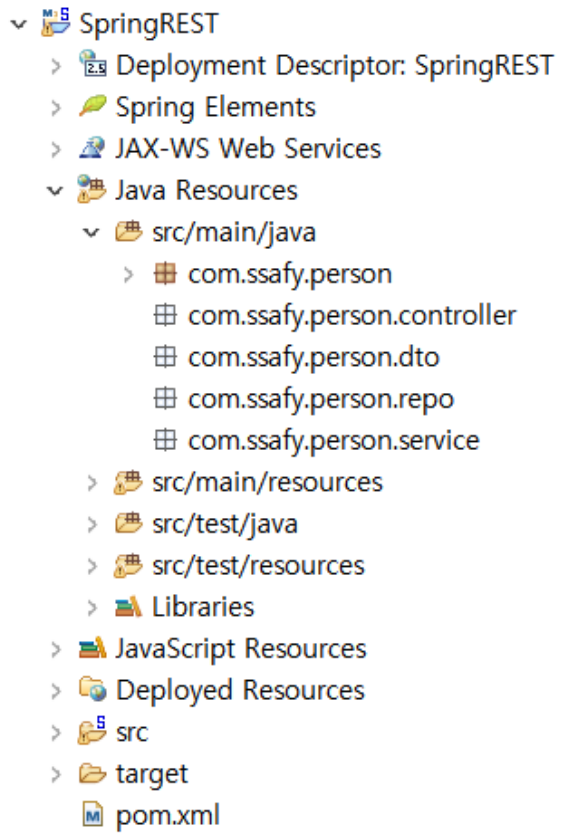
Confidential



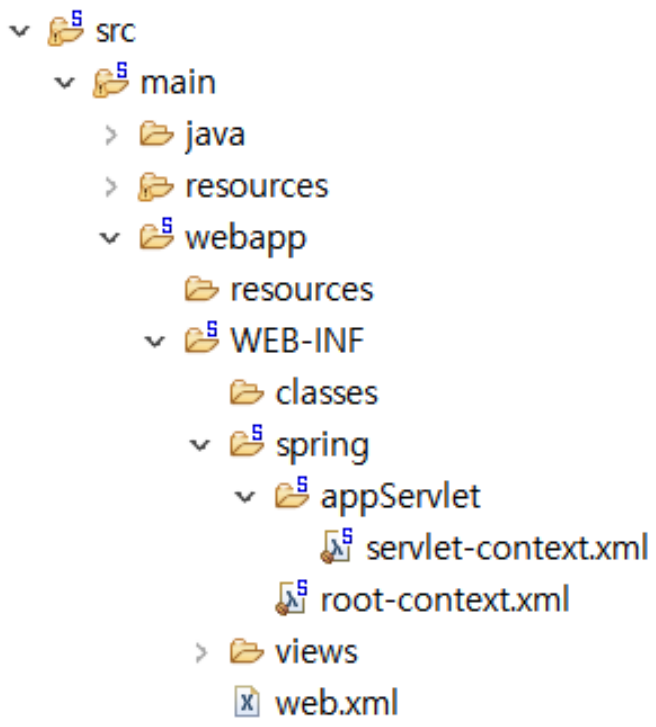
Confidential



Confidential



Confidential



Confidential

servlet-context.xml

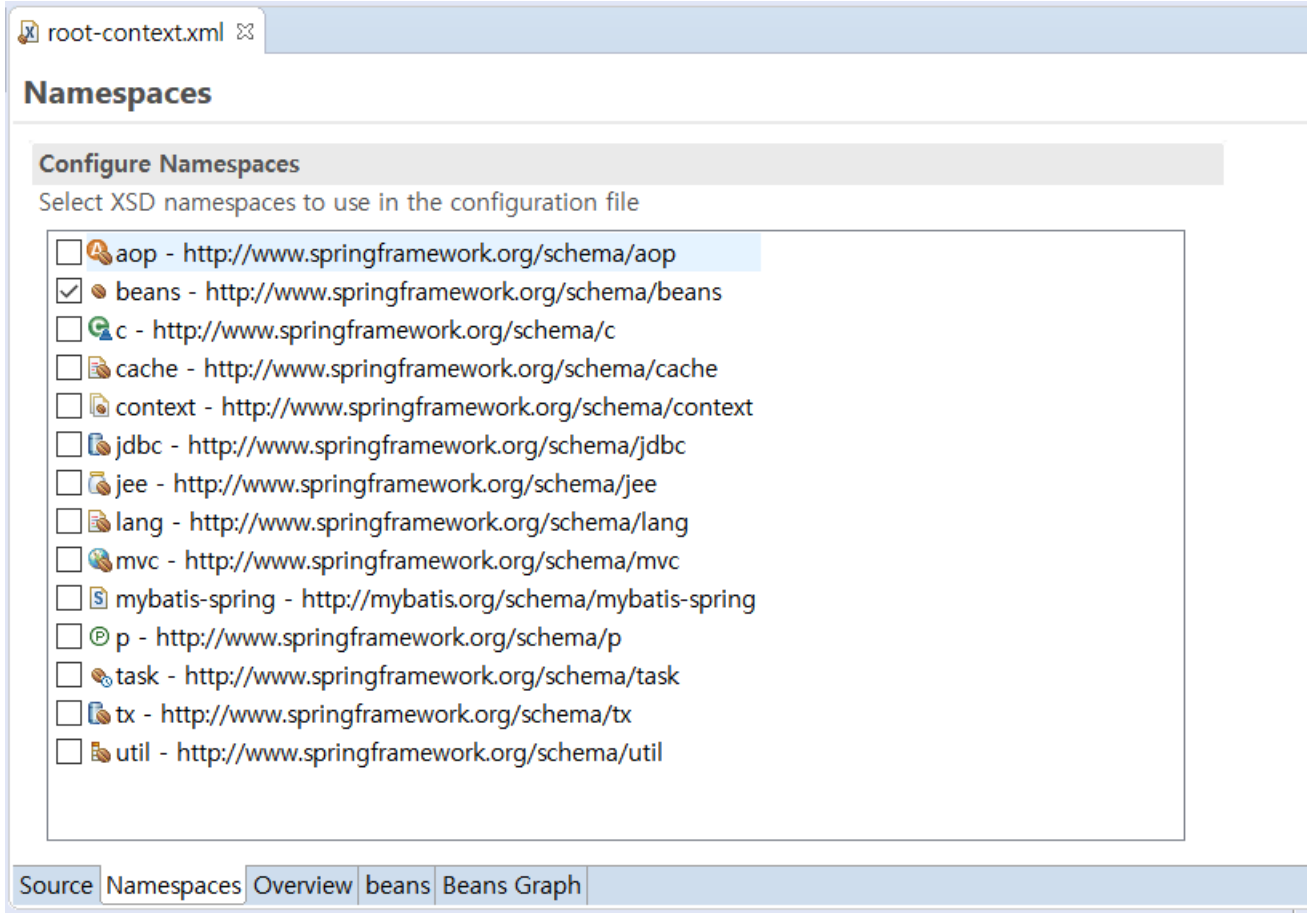
```
11
12      <!-- Enables the Spring MVC @Controller programming model -->
13      <annotation-driven />
14
15      <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resourc
16      <resources mapping="/resources/**" location="/resources/" />
17
18      <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-
19      <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
20          <beans:property name="prefix" value="/WEB-INF/views/" />
21          <beans:property name="suffix" value=".jsp" />
22      </beans:bean>
23
24      <context:component-scan base-package="com.ssafy.person" />
25
26      <!-- 모든 Controller클래스는 controller 패키지 밑에 작성하겠음!! -->
27      <context:component-scan base-package="com.ssafy.person.controller" />
28
29
30  </beans:beans>
```

Confidential

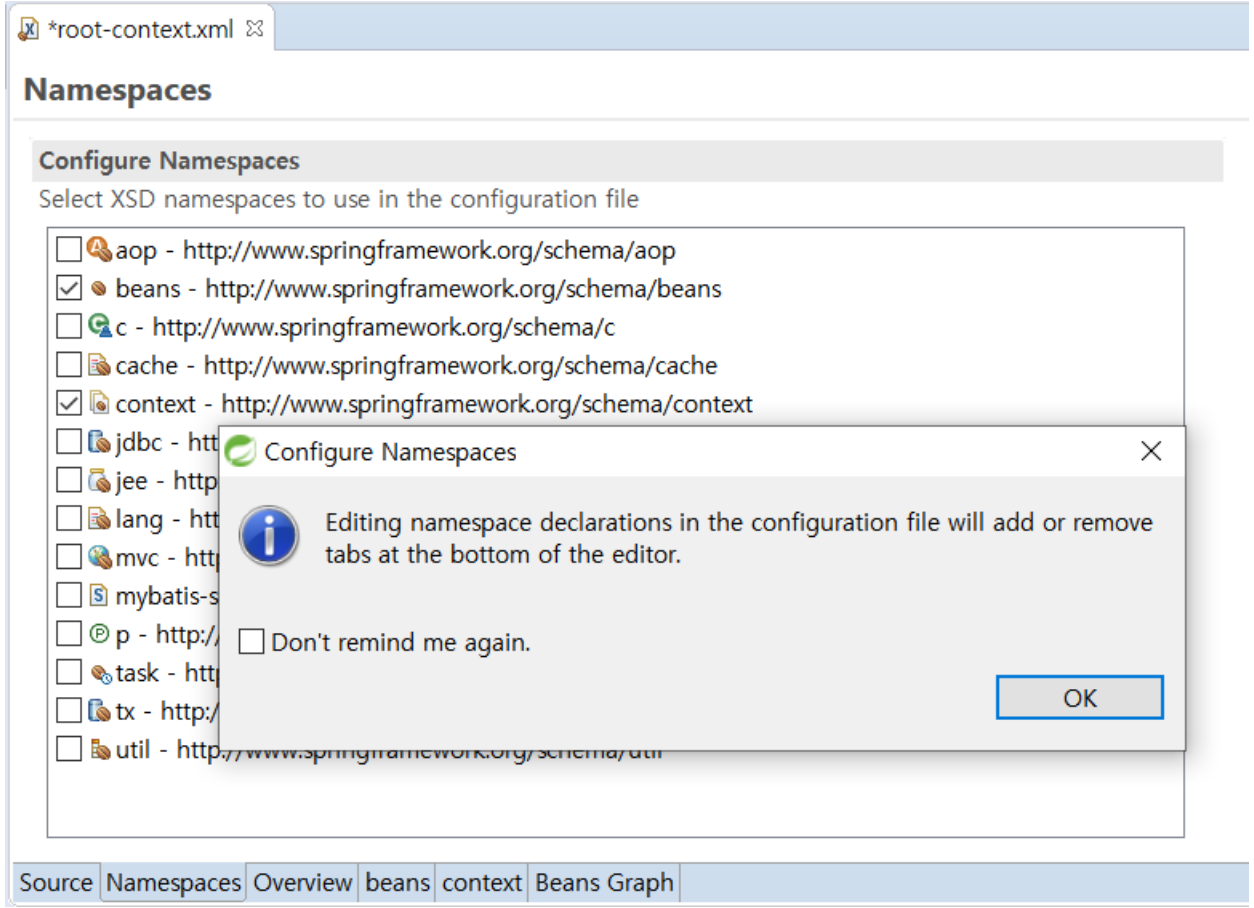
root-context.xml

```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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    https://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->
        <!--
         SpringBoard/ root-context.xml
         ==> 모델관련된 클래스(객체)들에 대한 등록,관리
           예) DAO, DBCP (관련클래스: DataSource)  ...
     -->
```

Confidential



Confidential



Confidential

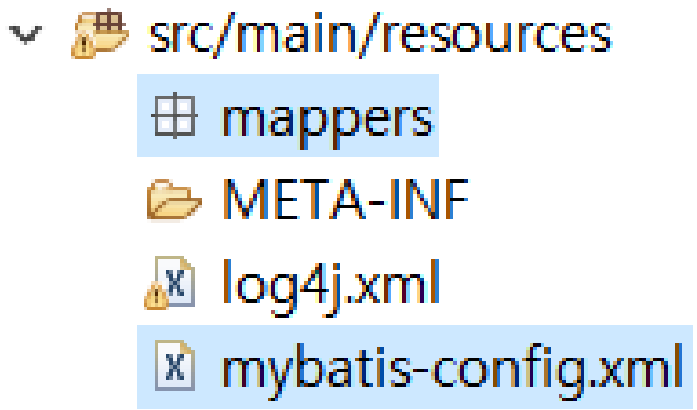
root-context.xml

```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
                https://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">

    <!-- Root Context: defines shared resources visible to all other web components -->
        <!--
         SpringBoard/ root-context.xml
         ==> 모델관련된 클래스(객체)들에 대한 등록,관리
           예) DAO, DBCP (관련클래스: DataSource)  ...
     -->

```

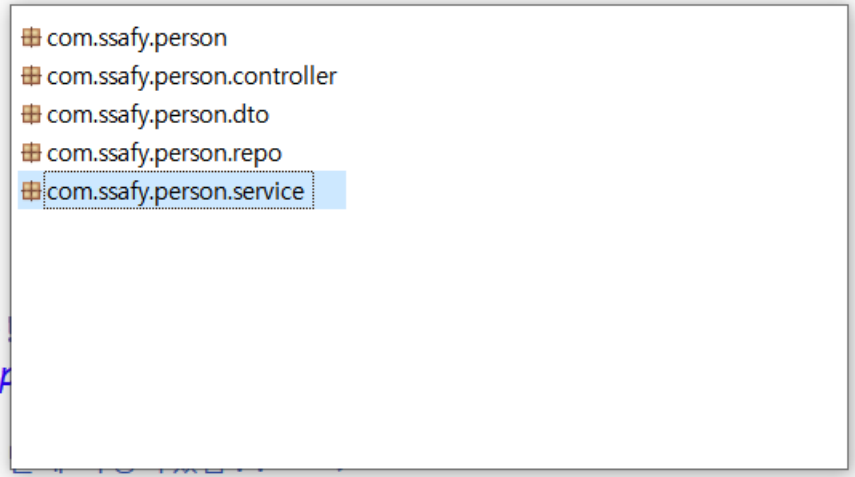
Confidential

```xml
<bean class="org.springframework.jdbc.datasource.DriverManagerDataSource"
      id="dataSource">
    <property name="driverClassName"
              value="com.mysql.cj.jdbc.Driver"></property>
    <property name="url"
              value="jdbc:mysql://127.0.0.1:3306/ssafydb?serverTimezone=UTC&amp;
    <property name="username" value="ssafy"></property>
    <property name="password" value="ssafy"></property>
</bean>


<!-- XML내에 작성된 sql문을 호출하는 객체: SqlMapClient(iBatis), SqlSession(MyBatis) -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean"
      id="sqlSessionFactory">
   <property name="dataSource" ref="dataSource"></property>
   <property name="configLocation"
             value="classpath:/mybatis-config.xml"></property>

   <!-- sql문 작성된 mapperXML문서 등록 -->
   <property name="mapperLocations"
             value="classpath:mappers/*.xml"></property>
</bean>

<bean class="org.mybatis.spring.SqlSessionTemplate"
      destroy-method="clearCache">
      <constructor-arg ref="sqlSessionFactory"></constructor-arg>
</bean>
```

Confidential



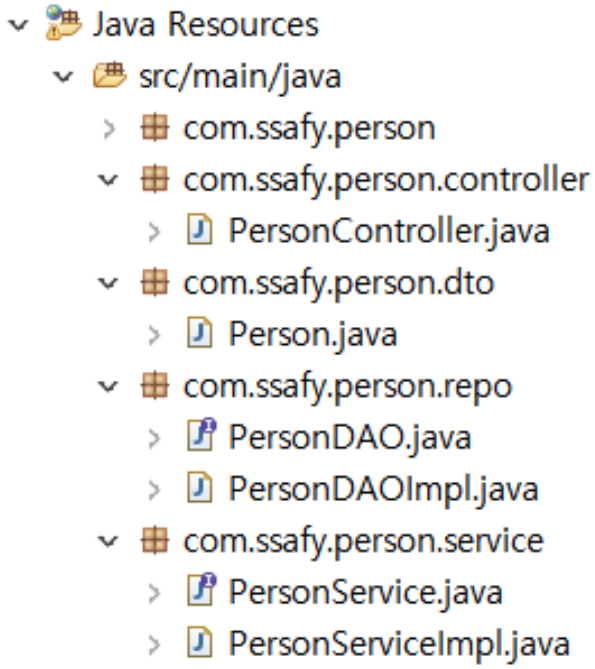
Confidential



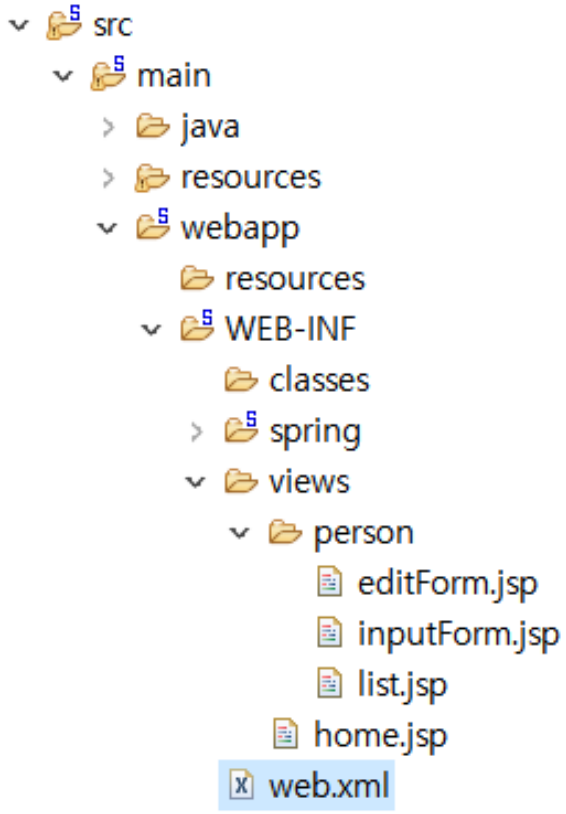

```
<!-- 모든 DAO, DAOImpl클래스는 repo 패키지 밑에 작성하겠음!
<context:component-scan base-package="com.ssafy.p

<!-- 모든 Service, ServiceImpl클래스는 service 패키지
<context:component-scan base-package="com.ssafy."/>
```

```
<bean class="org.mybatis.spring.SqlSessionTemplate"
      destroy-method="clearCache">
      <constructor-arg ref="sqlSessionFactory"></constructor-arg>
</bean>

<!-- 모든 DAO, DAOImpl클래스는 repo 패키지 밑에 작성하겠음!! -->
<context:component-scan base-package="com.ssafy.person.repo"/>

<!-- 모든 Service, ServiceImpl클래스는 service 패키지 밑에 작성하겠음!! -->
<context:component-scan base-package="com.ssafy.person.service"/>
```

Confidential



Confidential



Confidential

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>
    <!-- Creates the Spring Container shared by all Servlets and Filters -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- Processes application requests -->
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

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

```
27      <servlet-mapping>
28          <servlet-name>appServlet</servlet-name>
29          <url-pattern>/</url-pattern>
30      </servlet-mapping>
31
32      <!-- 한글 파라미터 처리 -->
33      <filter>
34        <filter-name>encoding</filter-name>
35        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
36        <init-param>
37           <param-name>encoding</param-name>
38           <param-value>UTF-8</param-value>
39        </init-param>
40      </filter>
41
42      <filter-mapping>
43        <filter-name>encoding</filter-name>
44        <url-pattern>/*</url-pattern>
45      </filter-mapping>
46  </web-app>
```

Confidential

```sql
-- no,name,age,job
-- 번호, 이름, 나이, 직업

-- person.sql
drop table person;

create table person(
    no          int     primary key auto_increment,
    name        varchar(20)  not null,
    age         int          not null,
    job         varchar(30)  not null
);
```

```java
package com.ssafy.person.dto;

public class Person {
    private int no;
    private String name;
    private int age;
    private String job;
```

```java
package com.ssafy.person.dto;

public class Person {
    private int no;
    private String name;
    private int age;
    private String job;

    public Person() {
    }

    public Person(int no, String name, int age, String job) {
        super();
        this.no = no;
        this.name = name;
        this.age = age;
        this.job = job;
    }

    public int getNo() {
        return no;
    }

    public void setNo(int no) {
        this.no = no;
    }
```

```java
package com.ssafy.person.controller;

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

@Controller //나 컨트롤러!! ---> servlet-context.xml등록
public class PersonController {

    /*
      컨트롤러내의 메소드 : MVC패턴 Servlet클래스의 service,doGet,doPost메소드역할
        <컨트롤러의 역할>
        1. 요청분석
        2. 사용자 입력데이터 얻어오기
        3. 서비스객체생성,호출
        4. 데이터 영역저장(뷰와 공유)
        5. 이동페이지 설정(forward,redirect)
     */
```

```
@RequestMapping("/m1")  //단순페이지 포워딩 (http://localhost:8080/person/m1 요청시)
                                                                  //가상경로
public String m1() {
  return "person/m1";
}
//prefix==> "/WEB-INF/views/"+"person/m1" + //suffix==> ".jsp"
// "/WEB-INF/views/person/m1.jsp"

@RequestMapping("/m2")  //단순페이지 포워딩 (http://localhost:8080/person/m2 요청시)
public void m2() {}
//만약 요청URL과 결과JSP경로가 일치한다면 return 생략가능
// "/WEB-INF/views/m2.jsp"
```

```java
@RequestMapping("/m3")  //리다이렉트 준비
public String m3() {
  return "person/m3";
}// "/WEB-INF/views/person/m3.jsp"

@RequestMapping("/m4")  //리다이렉트 이동
public String m4() {
  return "redirect:/m3";
}
```

Confidential

```
@RequestMapping("/m5")
public String m5(String name) {
/*
http://localhost:8080/person/m5?name=gildong

 또는

<form action="/person/m5" method=post>
   <input type=text name=name><br>
   <input type=submit value=전송>
</form>
*/
    System.out.println(name); //gildong출력
    return "person/m5";
}
```

```
@RequestMapping("/m6")
public String m6(String name, int age) {
    /*
http://localhost:8080/person/m6?name=gildong&age=13

 또는

<form action="/person/m6" method=post>
   <input type=text name=name><br>
   <input type=text name=age><br>
   <input type=submit value=전송>
</form>
     */
    System.out.println(name); //gildong출력
    System.out.println(age); //13출력
    return "person/m6";
}
```

```
@RequestMapping("/m7")
public String m7(Person p) {
    /*
http://localhost:8080/person/m7?name=gildong&age=13

 또는

<form action="/person/m7" method=post>
   <input type=text name=name><br>
   <input type=text name=age><br>
   <input type=submit value=전송>
</form>

스프링프레임워크에서는  Person p = new Person();
   p.setName(request.getParameter("name"));
   p.setAge(Integer.parseInt(request.getParameter("age"))); 코드생성
     */
    System.out.println(p.getName()); //gildong출력
    System.out.println(p.getAge()); //13출력
    return "person/m7";
}
```

Confidential

```java
@Controller //나 컨트롤러!! ---> servlet-context.xml등록
public class PersonController {

    @Autowired
    PersonService service;


    @RequestMapping("/m8")
    public String m8(int no, HttpServletRequest request) {

        request.setAttribute("person",service.find(no));

        return "person/m8";
    }

    @RequestMapping("/m8_2")
    public String m8(int no, Model model) {

        model.addAttribute("person",service.find(no));

        return "person/m8";
    }
```

```java
package com.ssafy.person.service;

import java.util.List;

import com.ssafy.person.dto.Person;

public interface PersonService {
    public int registry(Person person);
    public int modify(Person person);
    public int remove(int no);
    public Person find(int no);
    public List<Person> findAll();
}
```

```java
package com.ssafy.person.service;

import java.util.List;

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

import com.ssafy.person.dto.Person;
import com.ssafy.person.repo.PersonDAO;

@Service
public class PersonServiceImpl implements PersonService{

    @Autowired
    PersonDAO dao;
```

```java
package com.ssafy.person.repo;

import java.util.List;

import com.ssafy.person.dto.Person;

public interface PersonDAO {
    public int insert(Person vo);
    public List<Person> selectAll();
    public Person select(int no);
    public int update(Person vo);
    public int delete(int no);
}
```

Confidential

```java
package com.ssafy.person.repo;

import java.util.List;

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

import com.ssafy.person.dto.Person;

@Repository
public class PersonDAOImpl implements PersonDAO{

    @Autowired
    SqlSession sqlSession;
```

```java
@RequestMapping("/m9")
public String m9() {
    return "person/m9";
}

@RequestMapping("/m10")
public String m10(int no, HttpSession session) {

    session.setAttribute("person",service.find(no));

    return "redirect:/m9";
}
```

Confidential

```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>
 <!-- mybatis-config.xml -->
   <typeAliases>
      <package name="com.ssafy.person.dto" />
   </typeAliases>

</configuration>
```

Confidential

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

   <!-- no,name,age,job
        번호, 이름, 나이, 직업 -->

   <!-- 사람 입력 -->
   <insert id="insert" >
       insert into person (name,age,job)
       values (#{name},#{age},#{job})
   </insert>
```

```
<!-- 전체 사람 조회 -->
<select id="selectAll" resultType="Person">
<!-- parameterType속성 생략, resultType속성 필수!! -->
    select no,name,age,job
    from person
    order by no desc
</select>
```

```
<!-- (수정폼에 출력할) 사람 조회 -->
<select id="select" resultType="Person">
    select no,name,age,job
    from person
    where no=#{no}
</select>
```

```
    <!-- (수정폼에 입력된) 사람 수정 -->
    <update id="update">
        update person
        set name=#{name}, age=#{age}, job=#{job}
        where no=#{no}
    </update>

    <!-- (번호로 구분하는 )사람 삭제 -->
    <delete id="delete">
        delete from person
        where no=#{no}
    </delete>

</mapper>
```

Confidential

```java
@Repository
public class PersonDAOImpl implements PersonDAO{

    @Autowired
    SqlSession sqlSession;


    @Override
    public int insert(Person vo) {
        return sqlSession.insert("person.insert",vo);
    }

    @Override
    public List<Person> selectAll() {
        return sqlSession.selectList("person.selectAll");
    }
```

```java
    @Override
    public Person select(int no) {
        return sqlSession.selectOne("person.select",no);
    }

    @Override
    public int update(Person vo) {
        return sqlSession.update("person.update",vo);
    }

    @Override
    public int delete(int no) {
        return sqlSession.delete("person.delete",no);
    }

}
```

Confidential

```java
@Service
public class PersonServiceImpl implements PersonService{

    @Autowired
    PersonDAO dao;

    @Override
    public int registry(Person person) {
        return dao.insert(person);
    }

    @Override
    public int modify(Person person) {
        return dao.update(person);
    }
```

```java
    @Override
    public int remove(int no) {
        return dao.delete(no);
    }

    @Override
    public Person find(int no) {
        return dao.select(no);
    }

    @Override
    public List<Person> findAll() {
        return dao.selectAll();
    }
}
```

```java
@Controller //나 컨트롤러!! ---> servlet-context.xml등록
public class PersonController {

    @Autowired
    PersonService service;


    @RequestMapping(value = "/form", method = RequestMethod.GET) //입력폼보기
    public String form() {
      return "person/inputForm";
    }

    @RequestMapping(value = "/form", method = RequestMethod.POST) //DB입력
    public String formInsert(Person vo) {
        service.registry(vo);
        return "redirect:/list";
    }


    @RequestMapping("/list")
    public String list(Model m) {
        m.addAttribute("list",service.findAll());//뷰와 공유할 데이터를 영역에 저장
        return "person/list";//JSP페이지 포워딩
    }
```

```java
@RequestMapping(value = "/upform" , method = RequestMethod.GET)//수정폼 보이기
public String upform(int no, Model m) {
    m.addAttribute("person",service.find(no));
    return "person/editForm";
}

@RequestMapping(value = "/upform" , method = RequestMethod.POST)//DB수정하기
public String update(Person vo) {
    service.modify(vo);
    return "redirect:/list";
}

@RequestMapping("/delete")//DB삭제하기
public String delete(int no) {
    service.remove(no);
    return "redirect:/list";
}
```

Confidential

## 입력폼

[목록보기]

| | |
|---|---|
| 이름 | |
| 나이 | |
| 직업 | |
| 입력 취소 | |

# CRUD:리스트

---

사람정보 쓰기

| 번호 | 이름 | 나이 | 직업 |
|---|---|---|---|
| 3 | 김주원 | 17 | 학생 |
| 2 | 길라임 | 16 | 학생 |
| 1 | 홍길동 | 15 | 학생 |

## 수정폼

[목록보기]

| | |
|---|---|
| 이름 | 홍길동 |
| 나이 | 13 |
| 직업 | 학생 |
| 입력 취소 | |

Confidential

SAMSUNG
SW
ACADEMY
FOR
YOUTH

함께가요 미래로!
Enabling People

# 내일 방송에서 만나요!

삼성 청년 SW 아카데미