1. 소스 분석

WebApplicationContext

```
public interface WebApplicationContext extends ApplicationContext {
}
```

```
package org.springframework.context;
public interface ApplicationContext extends EnvironmentCapable …
```

```
package org.springframework.web.servlet;
public class DispatcherServlet extends FrameworkServlet {
        …
        public DispatcherServlet(WebApplicationContext webApplicationContext) {
                super(webApplicationContext);
                setDispatchOptionsRequest(true);
        }
        …
        protected void initStrategies(ApplicationContext context) {
                …
                initLocaleResolver(context);
                …
                initHandlerMappings(context);
                …
                initViewResolvers(context);
        }
        …
        private void initHandlerMappings(ApplicationContext context) {
                …
        }
        …
        private void initViewResolvers(ApplicationContext context) {
                …
        }
        …
}
```

```
package org.springframework.web.servlet;
public abstract class FrameworkServlet extends HttpServletBean implements
ApplicationContextAware {
        public static final String DEFAULT_NAMESPACE_SUFFIX = "-servlet";
        public static final Class<?> DEFAULT_CONTEXT_CLASS = XmlWebApplicationContext.class;
        public static final String SERVLET_CONTEXT_PREFIX = FrameworkServlet.class.getName()
+ ".CONTEXT.";
        …
        public String getNamespace() {
                return (this.namespace != null ? this.namespace : getServletName() +
DEFAULT_NAMESPACE_SUFFIX);
        …
        @Override
        protected final void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
                processRequest(request, response);
        }
        @Override
        protected final void doPost(HttpServletRequest request, HttpServletResponse
```

```
response) throws ServletException, IOException {
                processRequest(request, response);
        }
        …
        protected final void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                …
                try {
                        doService(request, response);
                }
                …
        }
        …
        protected abstract void doService(HttpServletRequest request, HttpServletResponse
response) throws Exception;
        …
}
```

```
package org.springframework.web.servlet;
public abstract class HttpServletBean extends HttpServlet implements
EnvironmentCapable, EnvironmentAware {
        …
        @Override
        public final void init() throws ServletException {
                …
                initServletBean();
        }
        …
}
```

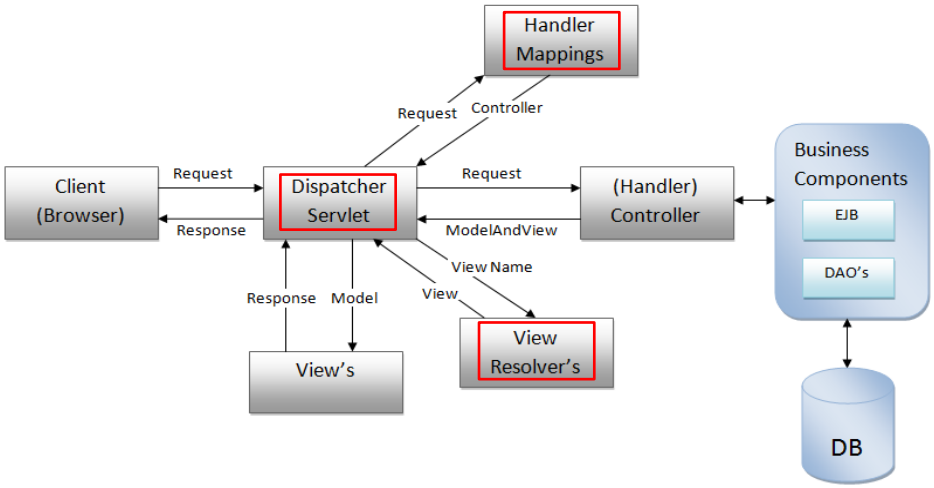
```
package javax.servlet.http;
public abstract class HttpServlet extends GenericServlet {
}
```

```
package javax.servlet;
public abstract class GenericServlet
        implements Servlet, ServletConfig, java.io.Serializable {
}
```

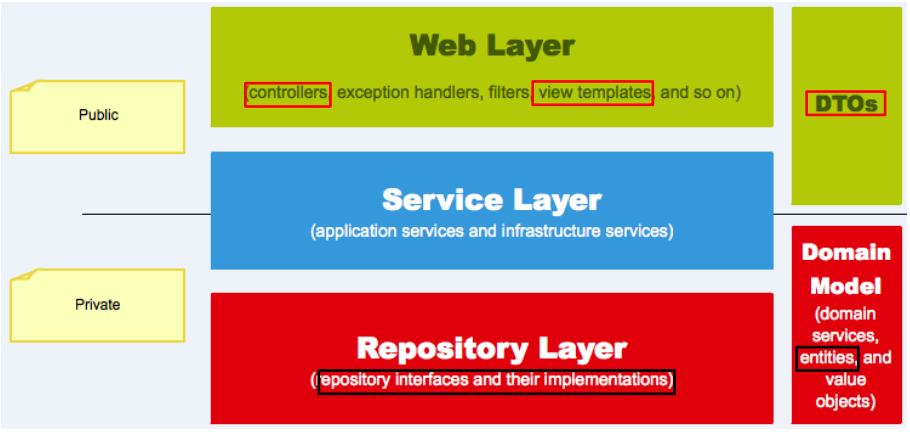
javax javax javax springframework springframework springframework

Servlet -> GenericServlet -> HttpServlet -> HttpServletBean -> FrameworkServlet -> DispatcherServlet

2. 스프링 웹 개념도



3. 소프트웨어 아키텍처



4. 스프링 소개

가. Spring MVC 다이내믹 웹 어플리케이션을 디자인 하는데 사용

나. 내부적으로 사용하는 패턴

- Front Controller
- Handler Mapper
- View Resolver
- View Resolver pattern: 뷰의 이름들과 실제 뷰(파일)들 사이에 매핑을 제공

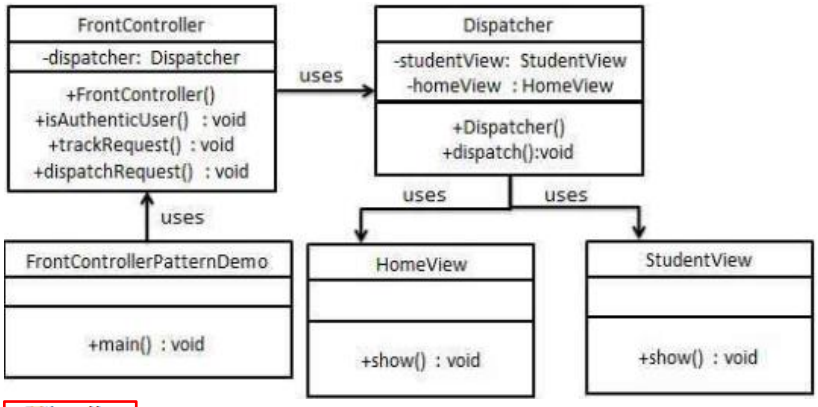
***** Front Controller Pattern *****

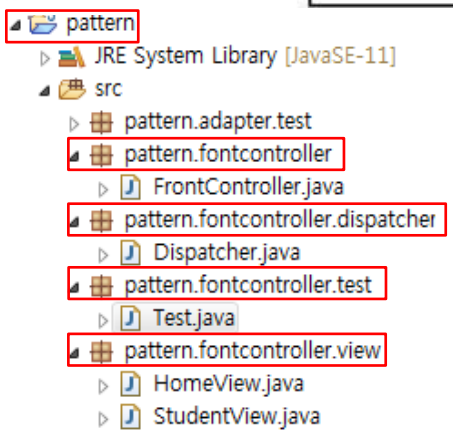
모든 요청을 단일 핸들러에 의해 처리되도록 하기 위해 집중화된 요청처리 메커니즘을 제공하는 디자인 패턴

프런트 컨트롤러(Front Controller): 어플리케이션으로 들어오는 모든 요청에 대한 단일 핸들러

디스패처(Dispatcher): 대등되는 특정 핸들에게 요청을 보낼 수 있는 오브젝트

뷰(View): 요청들이 완성되는 오브젝트





HomeView.java

```java
package pattern.fontcontroller.view;

public class HomeView {
    public void show()
    {
        System.out.println("Displaying Home Page");
    }
}
```

StudentView.java

```java
package pattern.fontcontroller.view;

public class StudentView {
    public void show()
    {
        System.out.println("Displaying Student Page");
    }
}
```

Dispatcher.java

```java
package pattern.fontcontroller.dispatcher;

import pattern.fontcontroller.view.HomeView;

public class Dispatcher {

    private HomeView homeView;
    private StudentView studentView;

    public Dispatcher()
    {
        homeView = new HomeView();
        studentView = new StudentView();
    }

    public void dispatch(String request)
    {
        if(request.equalsIgnoreCase("STUDENT"))
        {
            studentView.show();
        }
        else
        {
            homeView.show();
        }
    }
}
```

FrontController.java

```java
package pattern.fontcontroller;

import pattern.fontcontroller.dispatcher.Dispatcher;

public class FrontController {
    private Dispatcher dispatcher;

    public FrontController()
    {
        dispatcher = new Dispatcher();
    }

    private boolean isAuthenticUser()
    {
        System.out.println("User is authenticated successfully");
        return true;
    }

    private void trackRequest(String request)
    {
        System.out.println("Page requested: " + request);
    }

    public void dispatchRequest(String request)
    {
        // log each request
        trackRequest(request);

        // authenticate the user
        if(isAuthenticUser())
        {
            dispatcher.dispatch(request);
        }
    }
}
```

Test.java

```java
package pattern.fontcontroller.test;

import pattern.fontcontroller.FrontController;

public class Test {

    public static void main(String[] args) {
        FrontController frontController = new FrontController();
        frontController.dispatchRequest("HOME");
        frontController.dispatchRequest("STUDENT");

    }

}
```

Problems @ Javadoc Declaration

```
<terminated> Test (24) [Java Application] D:₩
Page requested: HOME
User is authenticated successfully
Displaying Home Page
Page requested: STUDENT
User is authenticated successfully
Displaying Student Page
```

5. Spring MVC 소개

가. DispatcherServlet

- Front controller design pattern 구현체
- web.xml에 설정
- 어떤 컨트롤러가 호출되어야 하는지를 아는 HanderMapper 사용

나. HanderMapper

- URL패턴을 사용해서 요청과 컨트롤러를 매핑

다. Controller 클래스

- POJO 클래스
- 스프링 프레임워크에서 제공하는 스테레오 타입의 @Controller 애노테이션 사용
- ModelAndView를 생성하는 메소드 구현

라. Model

- 어플리케이션 내의 데이터를 표현

마. View

- 최종 사용자에게 display 되어야 할 페이지
- 물리적인 실제 파일이름이 아닌 이름을 지칭

바. ModelAndView 타입으로 최종적으로 dispatcher에게 되돌려 짐

사. DispatcherServlet

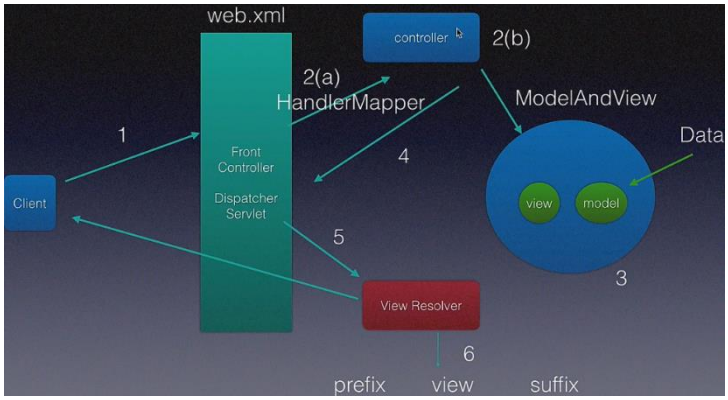
- view 이름을 받음
- View Resolver 호출

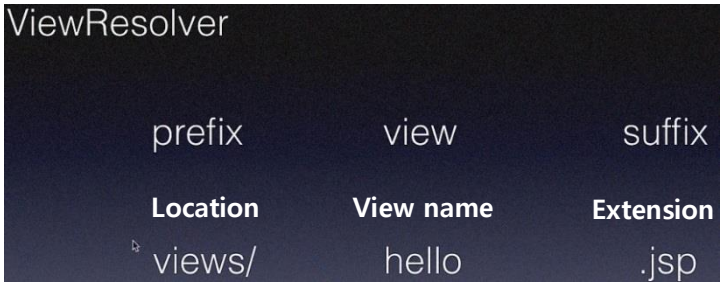
아. 뷰 리졸버(View Resolver)

- 뷰 이름(view name) 받아와 뷰 이름에 접두사와 접미사를 추가
- 접두사: 서버상의 뷰의 위치
- 접미사: 뷰의 확장자
- 최종적으로 DispatcherServlet에게 위치+뷰이름+확장자의 완성된 형태로 되돌려 줌

자. DispatcherServlet

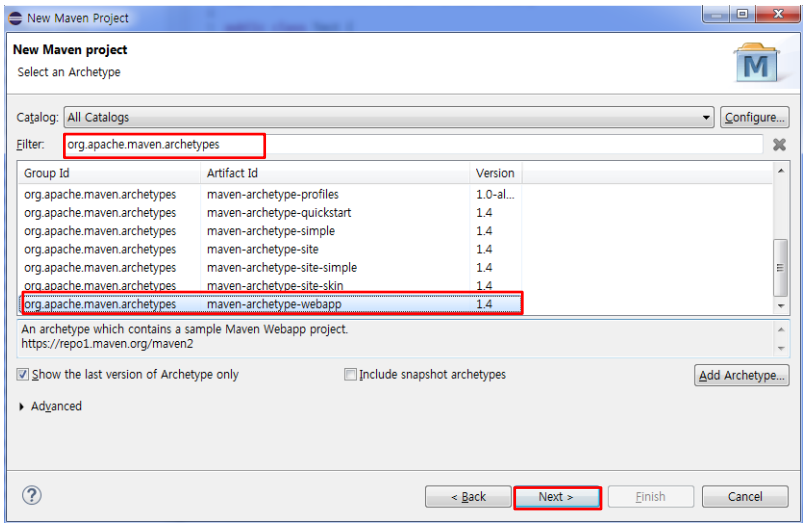
- 모델에 데이터가 있다면 뷰에게 데이터를 줌

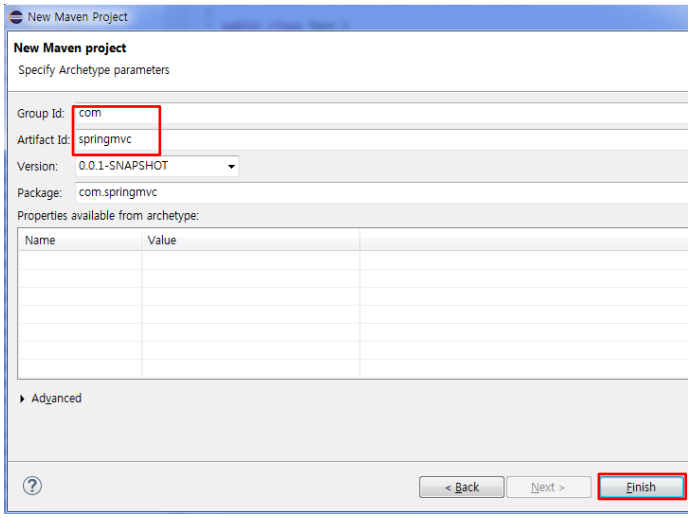




6. 메이븐 프로젝트 만들기

가. 프로젝트 생성





나. 프로젝트 생성 최초 구조

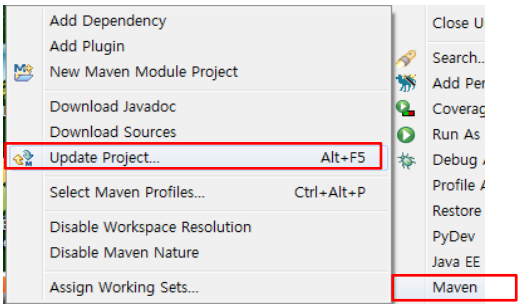
다. 스프링 mvc 모듈 추가: pom.xml

springmvc/pom.xml

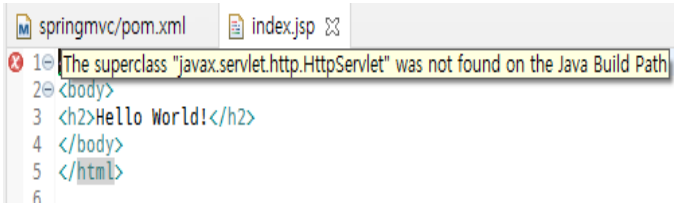
```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apa
  <modelVersion>4.0.0</modelVersion>

    <groupId>com</groupId>
    <artifactId>springmvc</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <name>springmvc Maven Webapp</name>
    <url>http://www.example.com</url>

    <properties>
        <springframework.version>4.3.6.RELEASE</springframework.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${springframework.version}</version>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.8.0</version>
                    <configuration>
                        <source>11</source>
                        <target>11</target>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

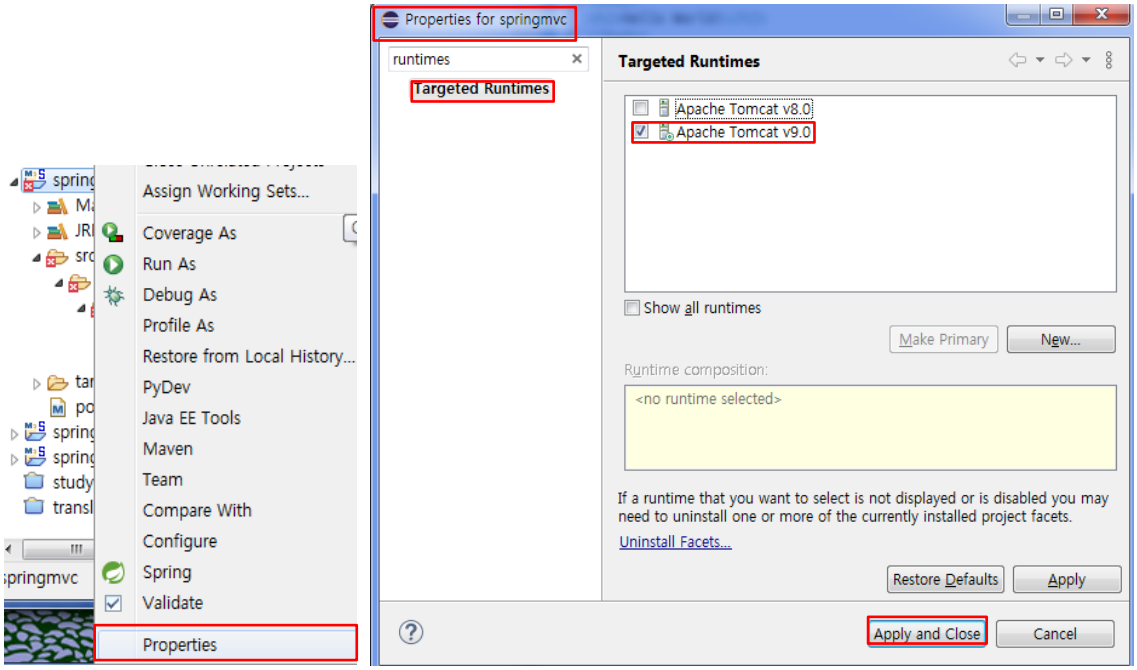
Libraries
- JRE System Library [JavaSE-1.7]
- Maven Dependencies
  - spring-webmvc-4.3.6.RELEASE.jar -
  - spring-aop-4.3.6.RELEASE.jar - C:₩
  - spring-beans-4.3.6.RELEASE.jar -
  - spring-context-4.3.6.RELEASE.jar -
  - spring-core-4.3.6.RELEASE.jar - C:₩
  - commons-logging-1.2.jar - C:₩Use
  - spring-expression-4.3.6.RELEASE.jar
  - spring-web-4.3.6.RELEASE.jar - C:₩

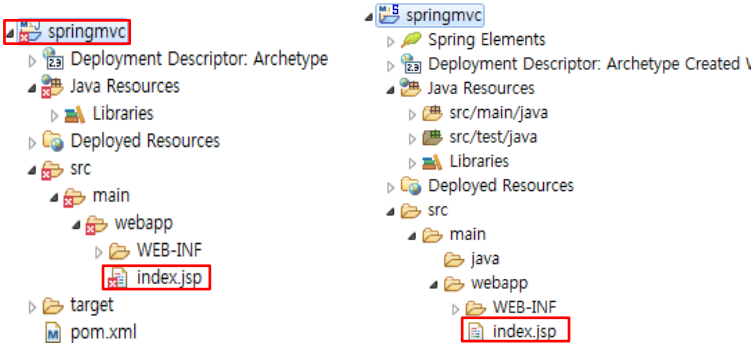
라. pom.xml 변경에 따른 메이븐 업데이트



마. 서블릿 런타임 추가

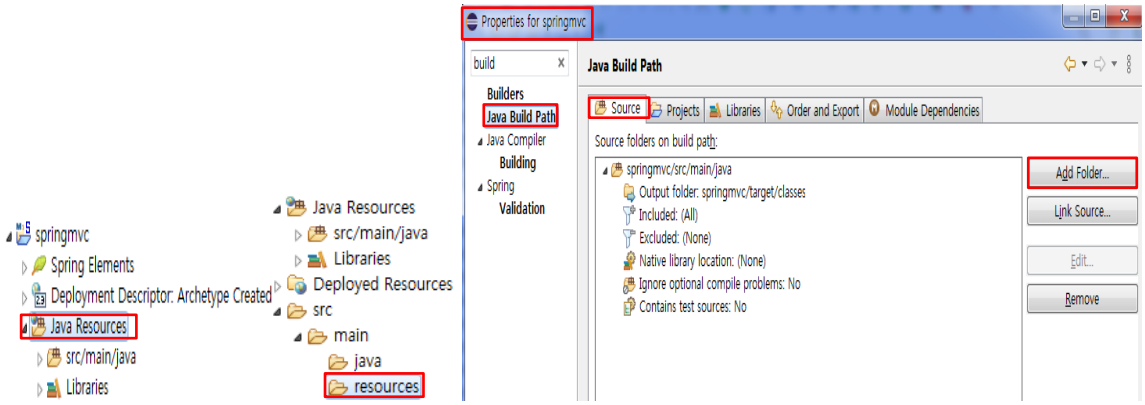


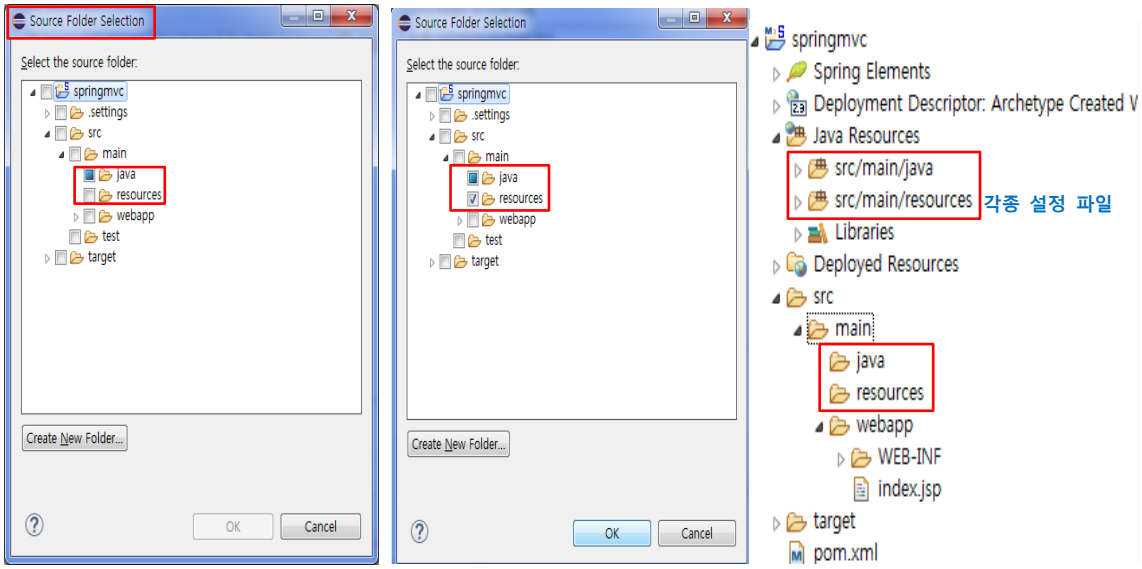


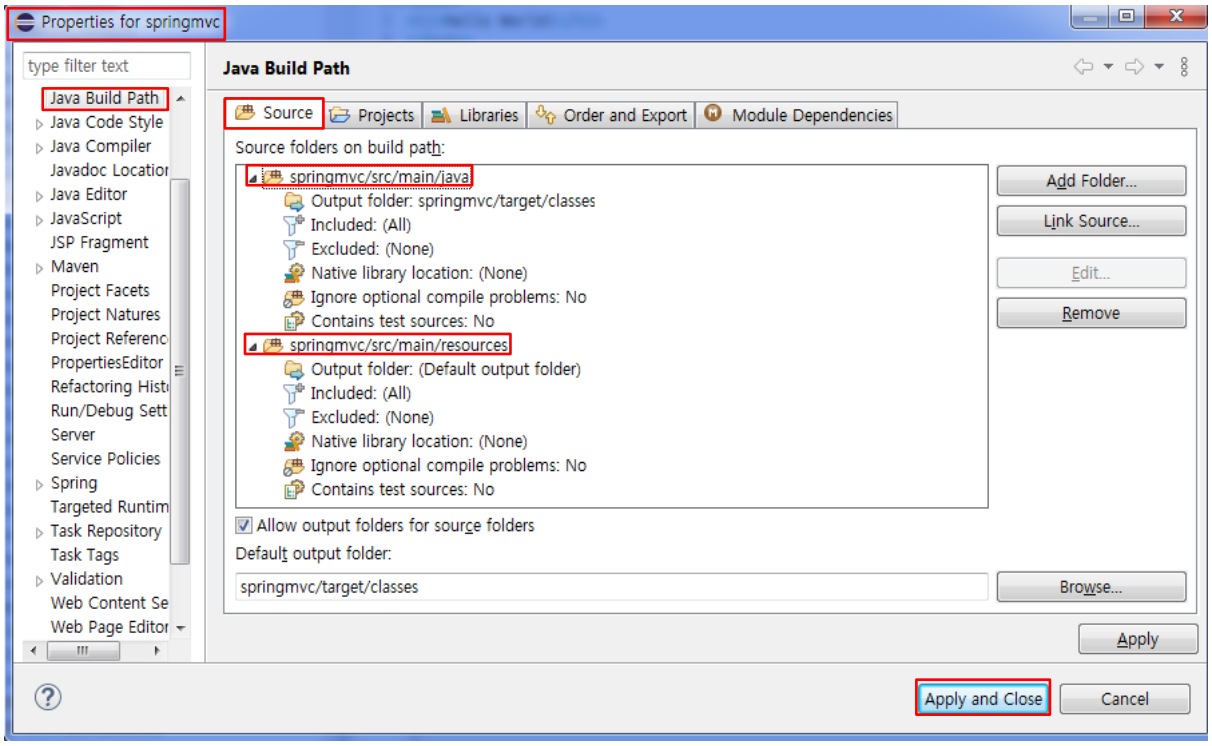


바. 테스트 폴더 삭제

사. 리소스 폴더 생성 및 클래스 패스 설정







7. 스프링 MVC 애플리케이션 생성 단계

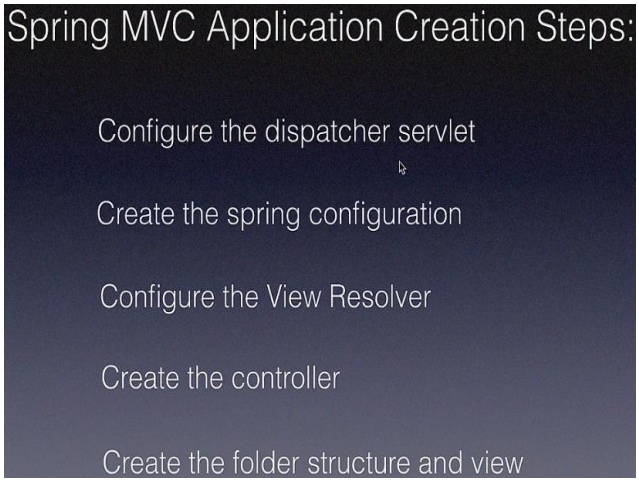
가. 디스패처 서블릿 설정: web.xml

나. 스프링 설정 파일 만들기: WEB-INF 폴더 아래

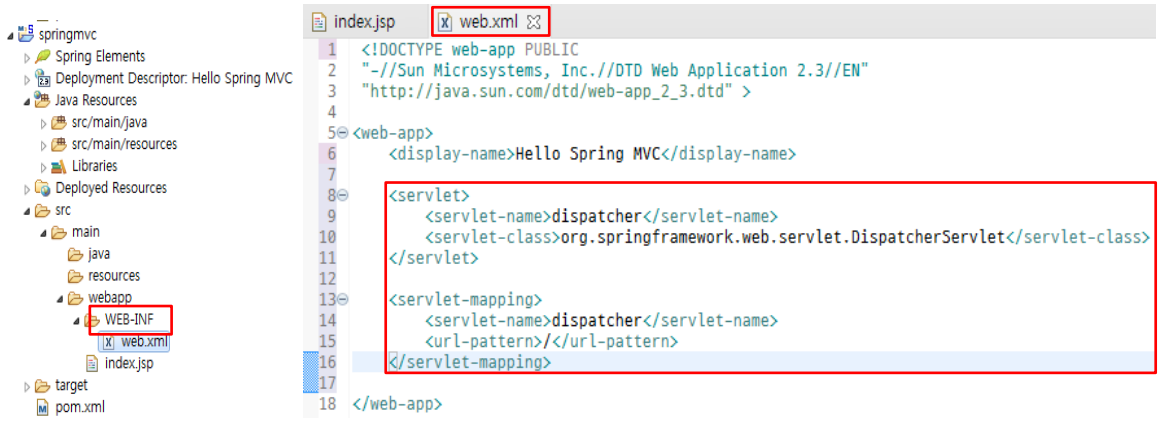
다. 뷰 리졸버(View Resolver) 설정

라. 컨트롤러 만들기

마. WEB-INF 아래 폴더 및 JSP 페이지 생성



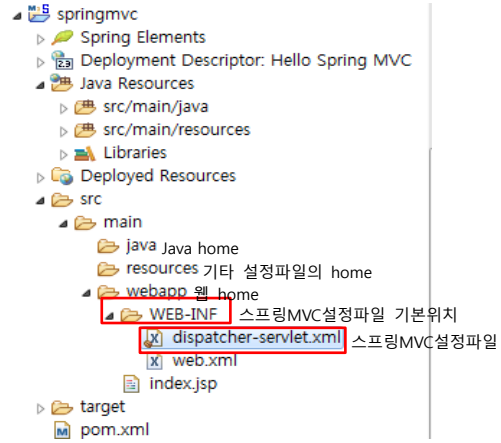
8. 디스패처 서블릿(사용자의 모든 요청을 수신) 설정: web.xml



9. 스프링 설정파일 만들기

가. 스프링 설정 파일 생성 및 이름 변경(스프링MVC Conventional Name으로)

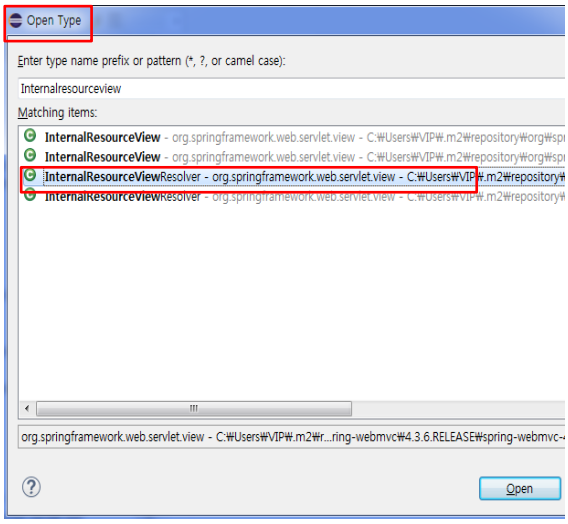
나. 최종 결과



10. 뷰 리졸버(View Resolver) 설정

dispatcher-servlet.xml | InternalResourceViewResolver.class

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:conte
4      xmlns:p="http://www.springframework.org/schema/p" xmlns:c="http:/
5      xmlns:tx="http://www.springframework.org/schema/tx"
6      xsi:schemaLocation="http://www.springframework.org/schema/beans
7      http://www.springframework.org/schema/beans/spring-beans.xsd
8      http://www.springframework.org/schema/context
9      http://www.springframework.org/schema/context/spring-context.xsd
10     http://www.springframework.org/schema/tx
11     http://www.springframework.org/schema/tx/spring-tx.xsd">
12
13     <bean class=""></bean>  뷰리졸버 클래스 설정
14
15 </beans>
```



Ctrl+Shift+t

```
public class InternalResourceViewResolver extends UrlBasedViewResolver
public class UrlBasedViewResolver extends AbstractCachingViewResolver
public abstract class AbstractCachingViewResolver extends WebApplicationObjectSupport implements ViewResolver
public abstract class WebApplicationObjectSupport extends ApplicationObjectSupport implements ServletContextAware
```

dispatcher-servlet.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"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        name="viewResolver">

            뷰리졸버 프로퍼티 세팅

    </bean>

</beans>
```

hello.jsp / dispatcher-servlet.xml

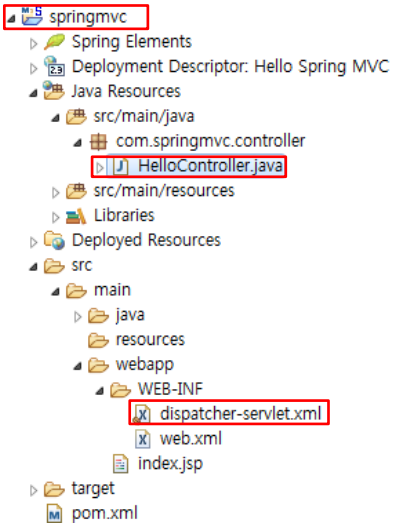
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"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

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        name="viewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>

        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

View 파일이 저장된 폴더 위치

View 파일의 확장자

개발자가 임의의 바꾸는 것이 가능(예: .html, . do 등)

11. 컨트롤러 만들고 설정하기



hello.jsp / dispatcher-servlet.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"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        name="viewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>

        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

뷰파일 위치

뷰파일 확장자

dispatcher-servlet.xml | HelloController.java

```java
package com.springmvc.controller;

import org.springframework.web.servlet.ModelAndView;

public class HelloController {

    public ModelAndView hello()
    {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("hello");
        return modelAndView;
    }
}
```

스프링 설정파일인 dispatcher-servlet.xml을 참조해

뷰리졸버는/WEB-INF/views/hello.jsp 페이지 리턴

hello.jsp | dispatcher-servlet.xml

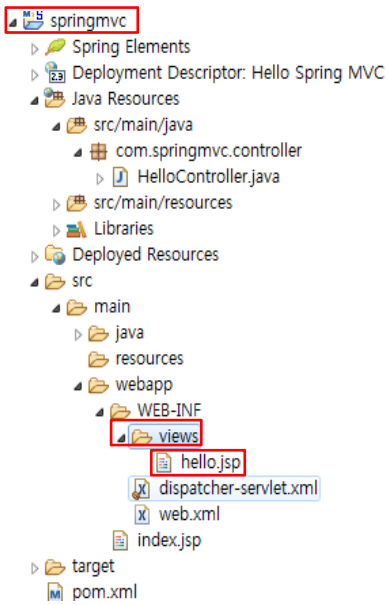
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"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

    <context:component-scan base-package="com.springmvc.controller" />

    수프링 컨테이너가 관리하는 컴포넌트 애노테이션 활성화

    <bean
        class="org.springframework.web.servlet.view.InternalResourceVi
        name="viewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

dispatcher-servlet.xml | HelloController.java

```java
package com.springmvc.controller;

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

@Controller 컴포넌트 애노테이션
public class HelloController {

    @RequestMapping("/hello") 요청 URL에 따른 핸들러 매핑
    public ModelAndView hello()
    {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("hello");
        return modelAndView;
    }
}
```

DispatcherServlet이 Handler Mapper 호출

12. 뷰 폴더와 뷰 페이지 만들기

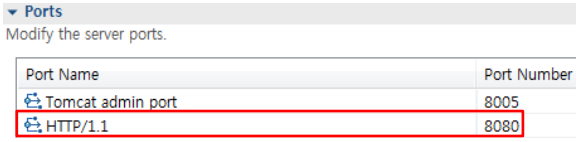


hello.jsp

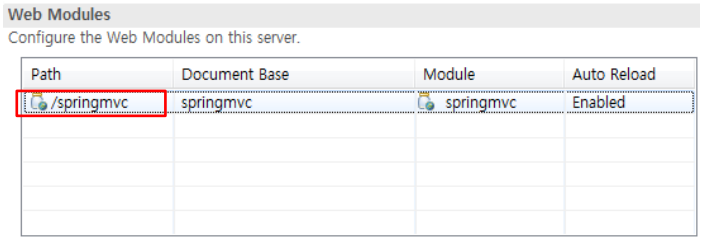
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hello</title>
    </head>
    <body>
    <h1>Welcome from spring MVC</h1>
    </body>
</html>
```

13. 서버에서 애플리케이션 실행

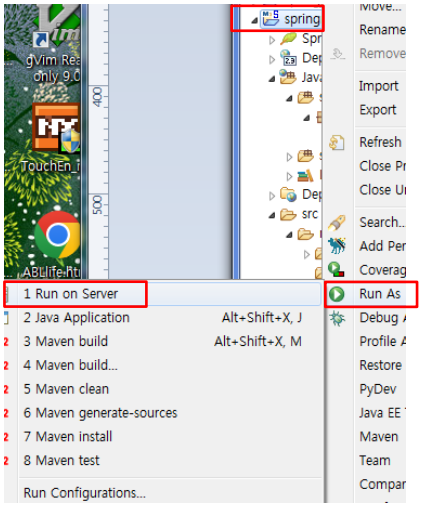
가. 실행포트 확인

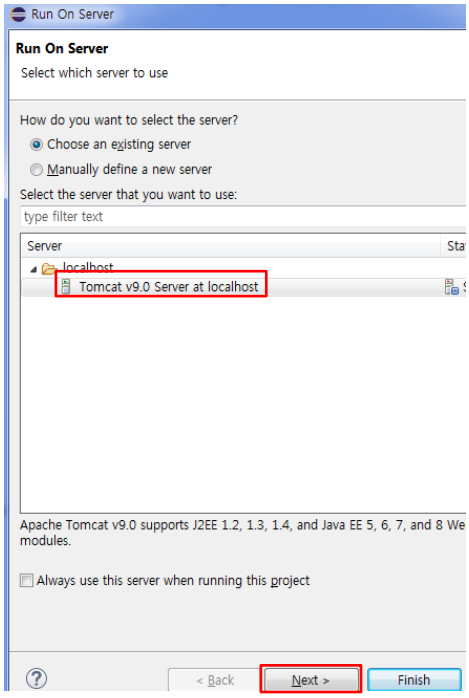


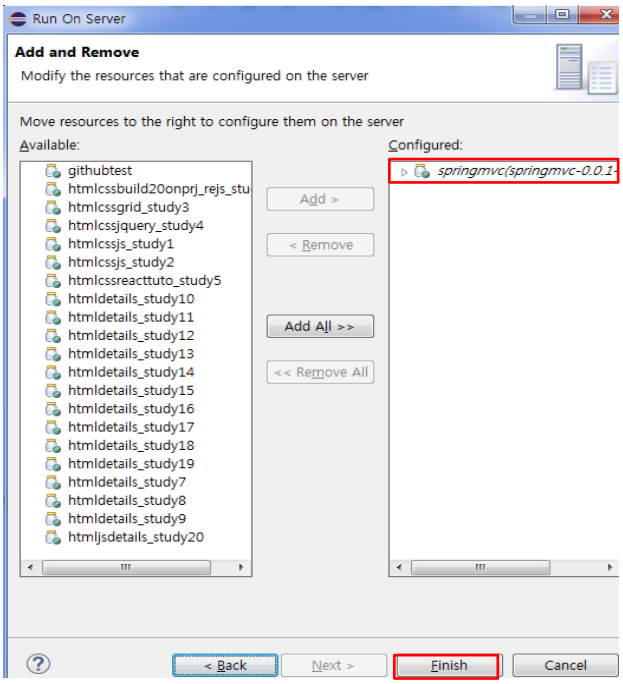
나. 실행 컨텍스트 확인

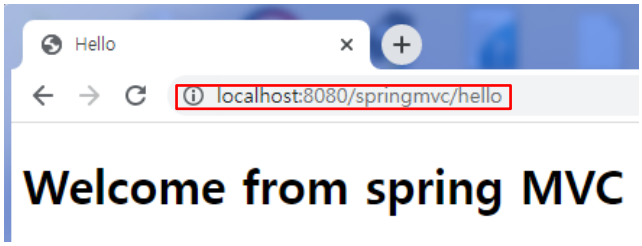


다. 실행








Hello
localhost:8080/springmvc/hello
Welcome from spring MVC