

Korea  
Software  
Technology  
Association



# 과정명 : 웹프레임워크의 활용 스트럿츠2 와 AJAX

강사명: 김 희 숙 (넥스트리소프트 hskim@nextree.co.kr)

2.0.14 version으로 작성되었으며,  
2.1.x version에서 사용하기 위해서는  
별도의 Migration작업이 필요합니다.

# 과정 개요

## □ 교육 목표

- 오픈 소스 기반 개발환경 숙달
- 아파치의 스트럿츠2 프레임워크 적용
- MVC 모델 개념 정립
- 실무 예제를 통한 개발 역량 향상
- AJAX 개념 및 AJAX 프레임워크 활용

## □ 교육 특징

- 강의와 실습을 병행하여 수강생의 명확한 이해를 도모함
- 숙달 및 깊은 이해가 필요한 부분은 실습 수행

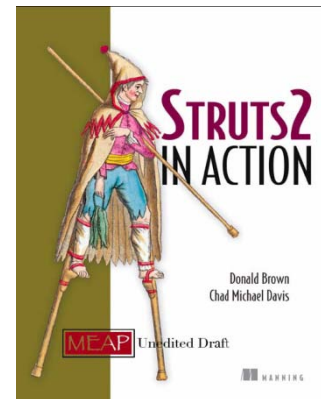
# 강의 일정

주차	일		강의 주제	비고
1	월	AJAX	AJAX 개요	
	화		Prototypejs	
	수		jQuery	
	목		jQuery	
	금		ExtJS	
2	월	Struts2	Struts2 아키텍처	
	화		Action과 Result	
	수		Interceptor	
	목		데이터 전송	
	금		Project 실습	

# 1. 스트럿츠2 개요

ONE STEP AHEAD

- 1.1 J2EE Architecture
- 1.2 모델1 vs 모델2 개발방식
- 1.3 모델2 기반 스트럿츠 프레임워크
- 1.4 스트럿츠의 문제점
- 1.5 스트럿츠2의 등장
- 1.6 스트럿츠2 바로 시작하기



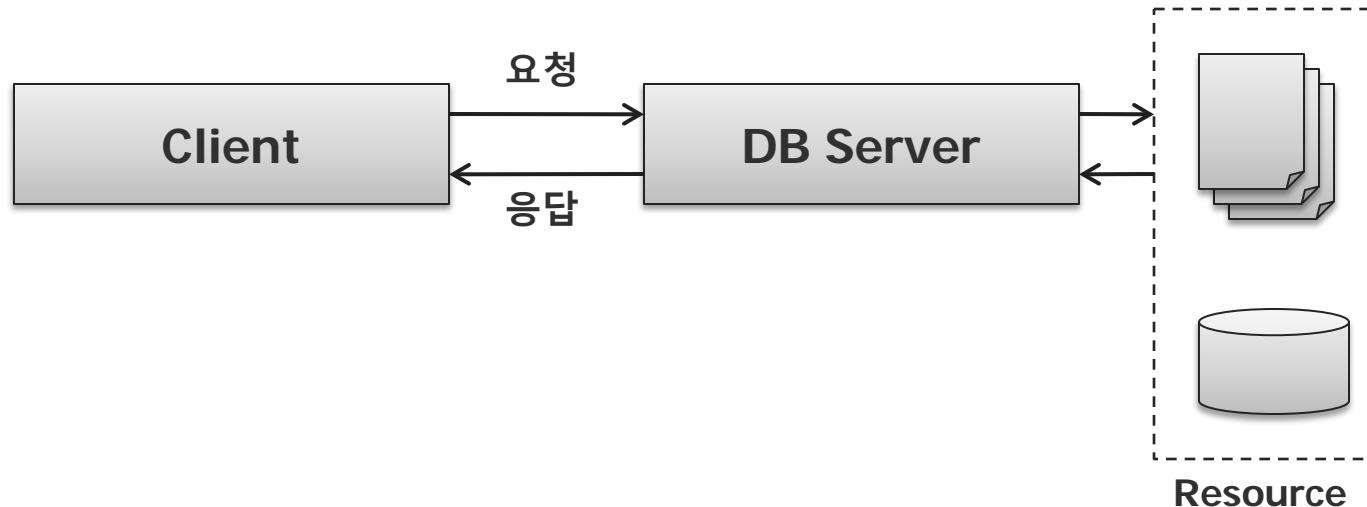
# 1.1 J2EE Architecture



# J2EE Architecture

## □ Client-Server

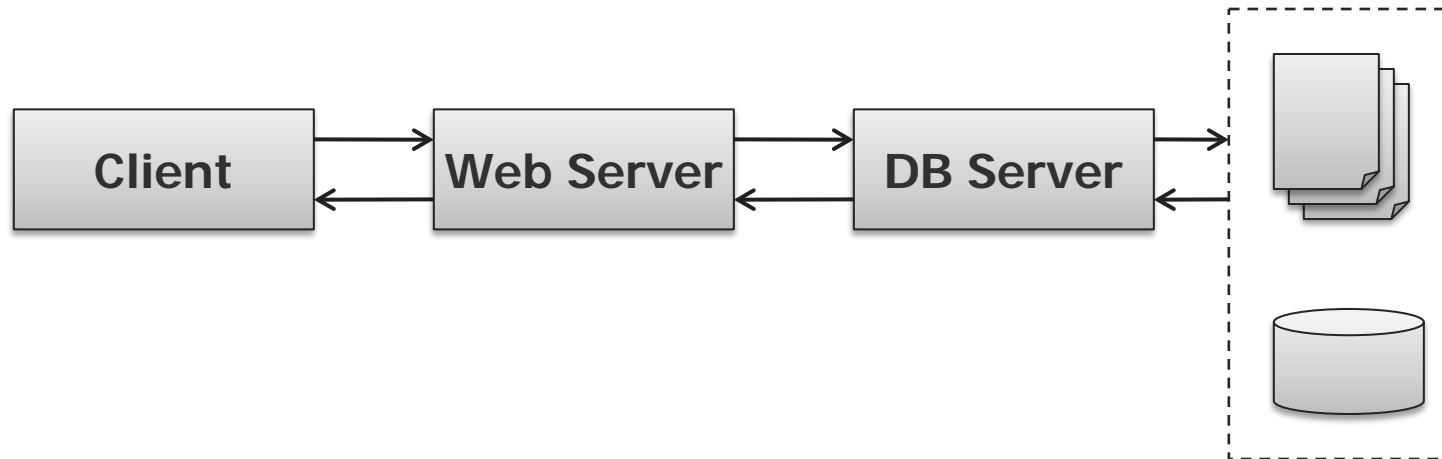
- 비즈니스 로직을 클라이언트에서 처리(클라이언트의 성능의존)
- 데이터 저장/조회시 경우만 DB Server 이용
- 비즈니스 로직을 변경시 모든 클라이언트 프로그램을 재배포해야 하는 부담이 있음



# J2EE Architecture

## □ 3-Tier

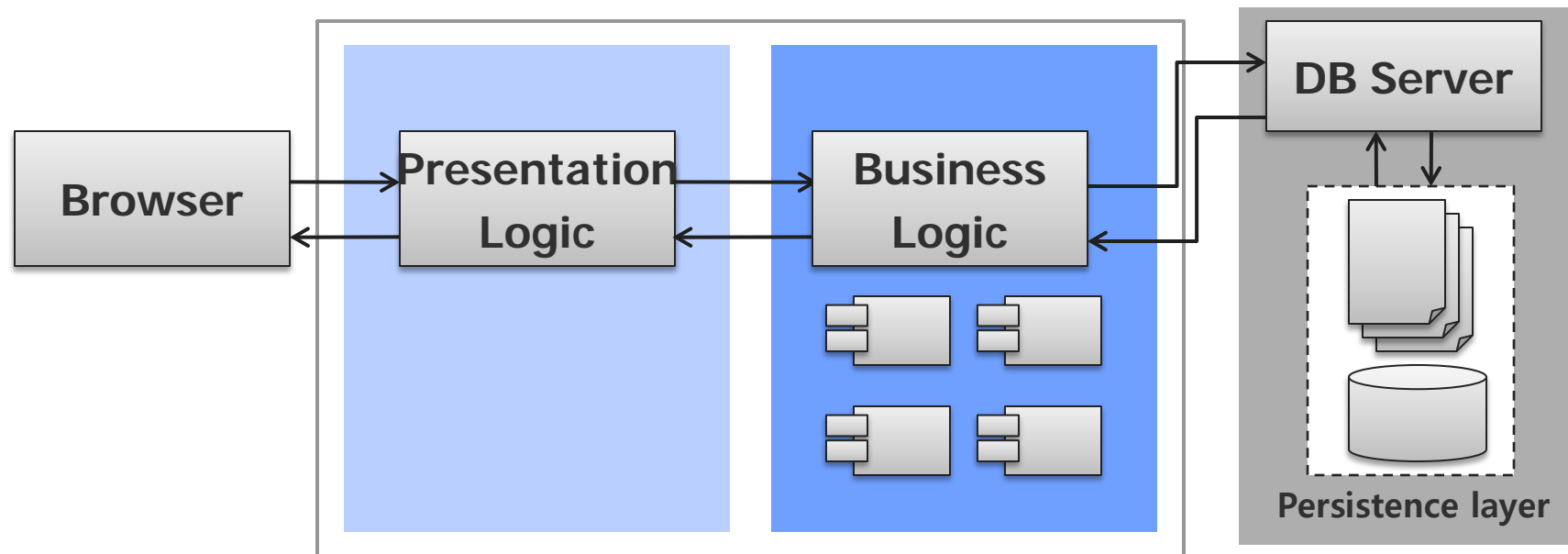
- 비즈니스 로직을 클라이언트에서 처리
- 데이터 저장/조회시 경우만 DB Server 이용
- DB접속과 클라이언트에 전송하는 부분이 하나의 구조로 되어 있어 재사용성이 어렵다



# J2EE Architecture

## □ n-Tier

- 3-tier 환경에서 코드를 재사용하는 어려움을 극복하기 위해 등장
- 비즈니스 로직을 담당하는 애플리케이션 계층을 쉽게 확장 가능

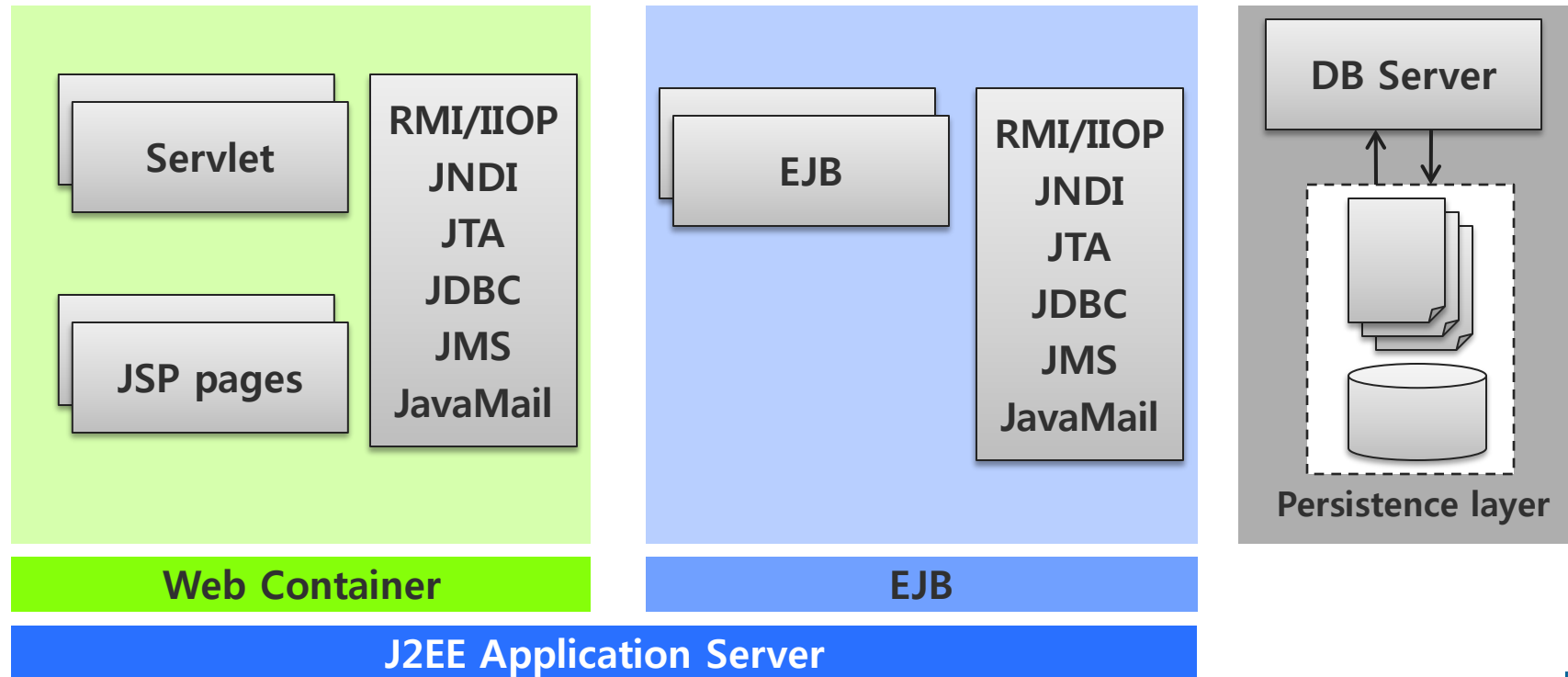




# J2EE Architecture

## □ J2EE Architecture

- EJB를 이용하여 분산 객체 시스템을 쉽게 구성가능
- 트랜잭션, 보안, 캐싱등의 지원으로 신뢰성 있는 n-tier 프로젝트를 지원



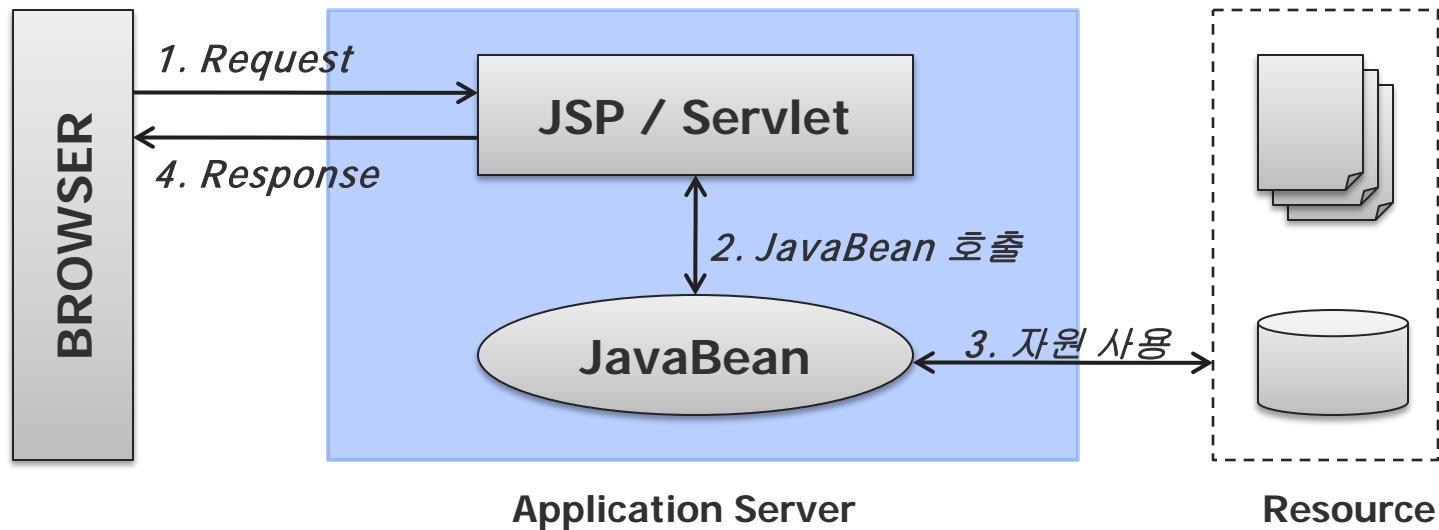
## 1.2 모델1 vs 모델2 개발방식



# 모델1 vs 모델2 개발방식

## □ 모델1 개발방식

- JSP / Servlet 만 이용하여 개발하는 경우
- JSP + Java Bean을 이용하여 개발하는 경우
- JSP + Custom Tag를 이용하여 개발하는 경우
- 위 3가지를 적절히 조합하여 개발하는 경우



# 모델1 vs 모델2 개발방식

## □ 모델1의 장단점

### ▪ 장점

- 개발자의 기술적인 숙련도가 낮아도 배우기 쉬워 빠르게 적용 가능

### ▪ 단점

- JSP 페이지에서 프레젠테이션 로직과 비즈니스 로직이 혼재되어 복잡
- 로직과 뷰의 혼합으로 인해 개발자와 디자이너의 작업 분리가 어려움
- 코드의 복잡도 증가로 인해 유지보수가 어려움

## □ 웹 애플리케이션이 복잡해지고 사용자 요구가 증가함에 따라 새로운 개발 방식을 요구

## 예) 모델1

### viewContentList.jsp

```

<%
String selectStmt = "SELECT * FROM T_BOARD" ;
Connection conn = null;
PreparedStatement pstmt = null;
ResultSet rs = null;

try {
    conn = DBUtil.getConnection();
    pstmt = conn.prepareStatement(selectStmt);
    rs = pstmt.executeQuery();
}%>

<HTML><HEAD><TITLE>게시판 목록 조회</TITLE></HEAD>
<BODY>
    <TABLE>
    <%
        while(rs.next()) {
    %>
        <TR>
            <TD><%= rs.getString(2) %></TD>
            <TD><%= rs.getString(5) %></TD>
        </TR>
    <%
        }
    %>

```

```

</TABLE>
</BODY>
</HTML>

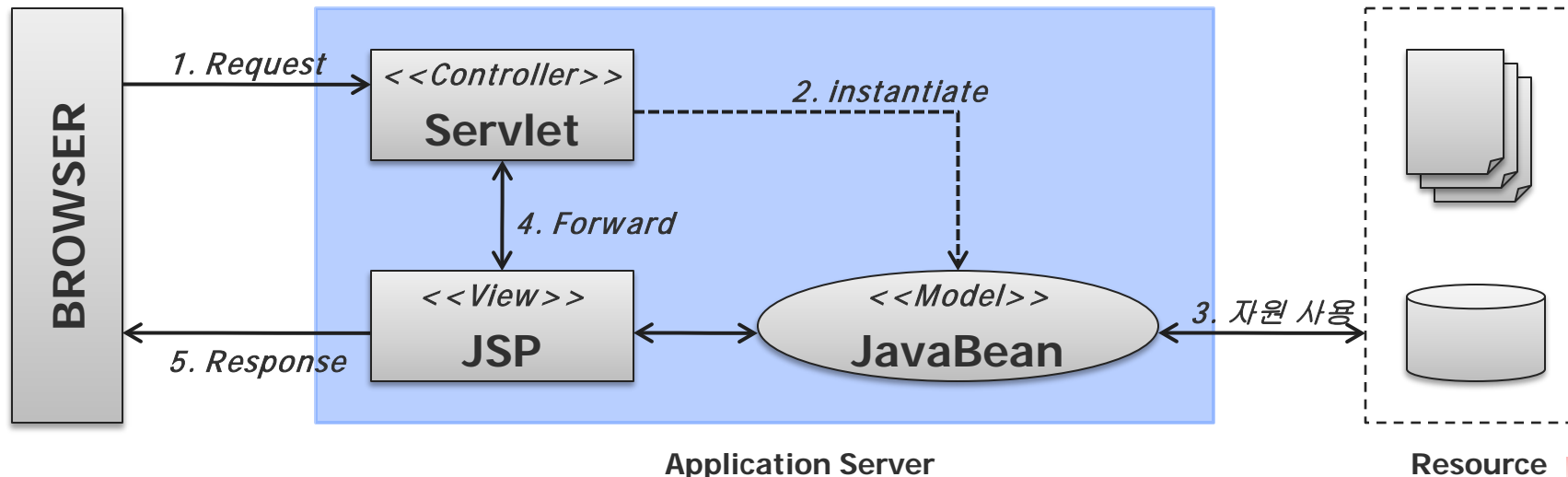
<%
    } catch(Exception e) {
        throw e;
    } finally {
        DBUtil.closeConnection(conn, pstmt, rs);
    }
}%>

```

# 모델1 vs 모델2 개발방식

## □ 모델2 개발방식

- MVC 패턴을 웹 애플리케이션에 적용하여 구현한 방식
- Model – View – Controller로 각각의 역할을 분리
  - Model: 애플리케이션 로직을 담당하는 부분으로 DB 또는 Legacy 시스템과의 로직을 처리하는 부분
  - View: 사용자가 직접 사용하는 부분으로 프레젠테이션 로직을 처리하고 결과물을 보여주는 역할을 담당
  - Controller: 비즈니스 로직을 담당하는 부분으로 사용자의 요청을 받아 작업한 후 결과에 따라 응답을 결정하는 역할을 담당



# 모델1 vs 모델2 개발방식

## □ 모델2의 장단점

### ▪ 장점

- 로직과 뷰에 대한 분리 가능으로 디자이너와 개발자의 분업이 가능
- 재사용성, 확장성, 유지보수성 증가

### ▪ 단점

- 설계 패턴에 대한 이해와 규약 필요

## □ 따라서, 대다수 웹 애플리케이션 프레임워크는 모델2 개발방식을 적용

## 예) 모델2

### viewContentList.jsp

```

<HTML>
  <HEAD>
    <TITLE>게시판 목록 조회</TITLE>
  </HEAD>

  <BODY>
    <TABLE>
      <c:forEach var="content" items="${contents}">
        <TR>
          <TD>${content.name}</TD>
          <TD>${content.writer}</TD>
        </TR>
      </c:forEach>
    </TABLE>
  </BODY>
</HTML>
    
```

### ViewContentServlet.java

```

public class ViewContentServlet
    extends javax.servlet.http.HttpServlet
    implements javax.servlet.Servlet {

    public ViewContentServlet() {
        super();
    }

    protected void doGet(HttpServletRequest req,
        HttpServletResponse res)
        throws ServletException, IOException {

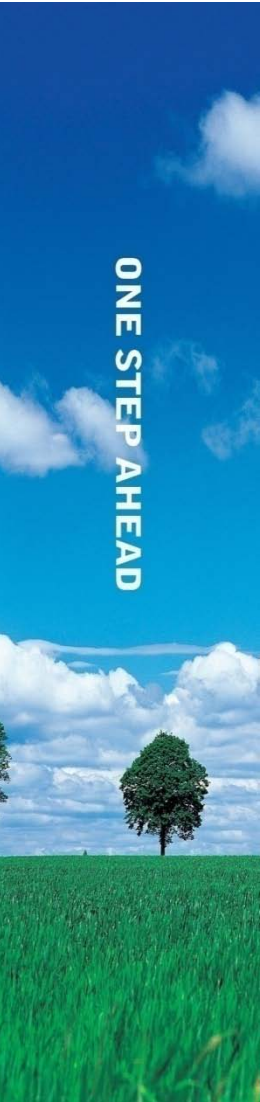
        ....
        List<Content> contents = service.getAllContents();
        req.setAttribute("contents", contents);

        ....
    }

    ....
}
    
```



## 1.3 모델2 기반 스트럿츠 프레임워크

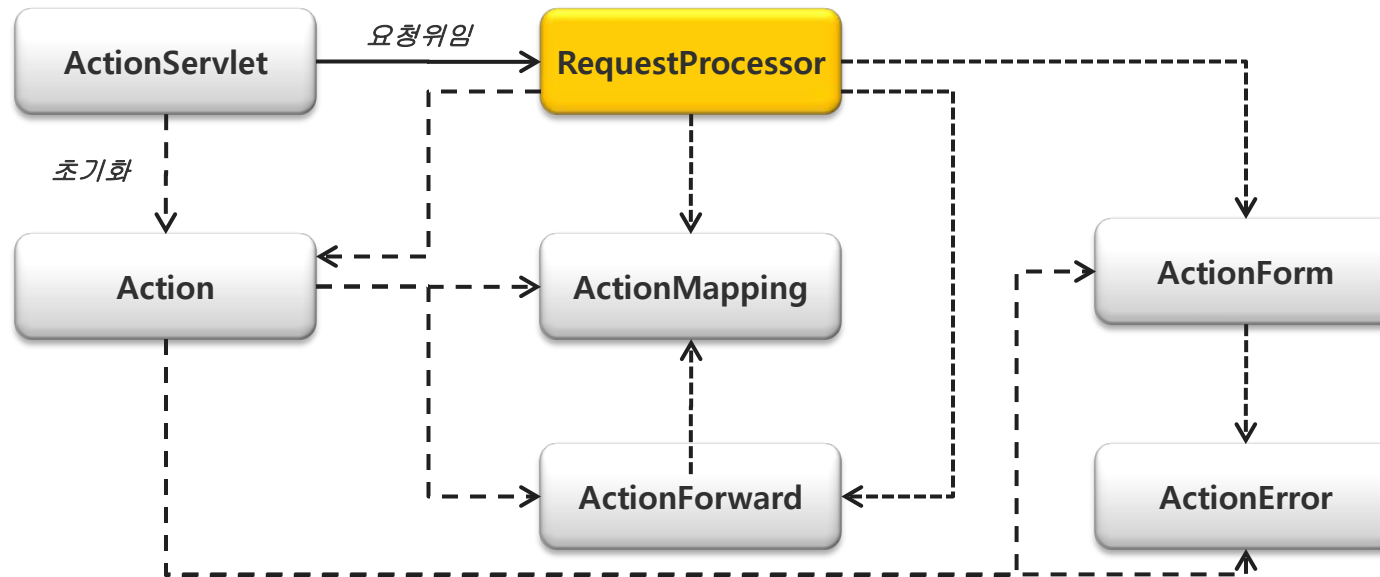


# Struts

# 모델2 기반 스트럿츠 프레임워크

## □ 스트럿츠 프레임워크의 핵심 클래스

- ActionServlet
  - 클라이언트의 모든 요청이 ActionServlet에 중앙 집중화됨
  - 초기화할 때 스트럿츠 구성정보인 struts-config.xml 파일을 읽어 정보를 로드함
  - 요청에 대한 처리는 RequestProcessor에 위임
- RequestProcessor
  - 사용자 요청을 처리하기 위한 메커니즘을 가지고 있음
  - 요청에 대한 Action클래스를 생성하여 처리하도록 위임하고 결과를 Forward함



# 스트럿스

## □ WebApplication 설정

- \*.do 의 패턴인 요청을 Struts의 ActionServlet에 매핑
- struts-config 에 설정된 정보를 ActionServlet 로딩시에 인자로 전달

web.xml

```

<web-app>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
    
```

# 스트럿츠

## □ 스트럿츠 설정

- struts-config.xml 에는 formbean, action, exception, forward 등의 Struts를 사용하기 위한 필수 정보설정
- config 파일은 모듈에 따라 복수 설정 가능

struts-config.xml

```

<struts-config>
  <form-beans>
    <form-bean name="userForm" type="my.struts.ui.formbeans.UserForm"/>
  </form-beans>
  <action-mappings>
    <action path="/Hello" forward="/pages/HelloStruts.jsp" />

    <action path="/HelloStruts" type="my.struts.ui.actions.HelloStruts" name="userForm"
      scope="request" >
      <forward name="hello" path="/pages/HelloStruts.jsp"/>
    </action>
  </action-mappings>
  <controller processorClass="my.struts.ui.processor.BaseRequestProcessor" locale="true"/>
</struts-config>
    
```

# 스트럿스

## □ ActionForm

- 요청의 인자로 넘어오는 파라미터를 보관하는 bean
- default 생성자와 인자를 가지는 생성자, setter/getter가 설정되어야 함

UserForm.java

```
package my.struts.ui.formbeans;

public class UserForm extends ActionForm implements Serializable{

    private String name;
    <!-- 생성자 -->
    public UserForm(){
    public UserForm(String name) { this.name = name;}

    <!-- getter and setter -->
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

## □ Action

- form을 action 요청의 정보로 활용, 비즈니스로직을 호출 처리
- mapping 을 이용하여 처리과정에 따라 다른 결과로 forward 가능

HelloStruts.java

```
package my.struts.ui.actions;

public class HelloStruts extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                HttpServletRequest request, HttpServletResponse response){

        UserForm userForm = (UserForm)form;
        String userName = userForm.getName();

        request.setAttribute("username", userName);
        return mapping.findForward("hello");
    }
}
```

# 스트럿스

## □ View

- Action에 요청/결과를 사용자에게 보여주기 위한 페이지

### HelloStruts.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

<html>
<head><title>Hello World</title></head>
<body>
    <form action="HelloStruts.do" method="get">
        <input type="text" name="name" value=""/> <br/>
        <input type="submit" value="Submit"/>
    </form>
    ${username} 님 환영합니다.

</body>
</html>
    
```

## 1.4 Struts의 문제점





# Struts의 문제점

## □ 단위 테스트 어려움

- 로직을 수행하는 Action 클래스가 Framework에 종속되어 있으므로 단위 테스트 수행이 어려움

## □ 설정 파일의 복잡함

- struts-config.xml 에 data-source, form-bean, exception, forward, action-mapping등의 많은 요소를 설정해야 한다.

## □ 액션 로직의 간결성 유지 곤란

- 요청 파라미터 분리등의 로직이 액션에 포함되므로 가독성 저하

## 1.5 Struts2의 등장



# Struts2 등장

## □ 새로운 아키텍처

- WebWork의 MVC 아키텍처 적용

## □ POJO 기반

- 특정 인터페이스구현이나 클래스를 확장하지 않아도 된다.
- 단위 테스트가 용이
- 별도의 FormBean없이 모델 객체를 사용

## □ Zero Configuration 지향

- 기본 값을 사용하여 많은 설정을 생략가능
- Annotation사용하여 설정파일 정의 내용 생략 가능
- 와일드카드 매핑을 통해 패턴을 가진 액션을 하나로 함축정의 가능

## □ 인터셉터

- 액션의 실행 전과 실행 후의 작업을 별도의 계층으로 구성 가능

# Struts2 등장

## □ 강력한 태그 지원

- 테마와 템플릿을 사용하여 재사용성이 높은 UI 태그 지원

## □ 손쉬운 Ajax 구현

- Dojo 프레임워크를 포함하여 Ajax 테마를 지원하는 태그 사용하여 간단히 Ajax 애플리케이션을 구현 가능

## □ 유용한 플러그 인

- JFreeChart, JasperReport, Sitemesh등을 플러그인을 통해 사용가능

## □ 의존성 주입 지원

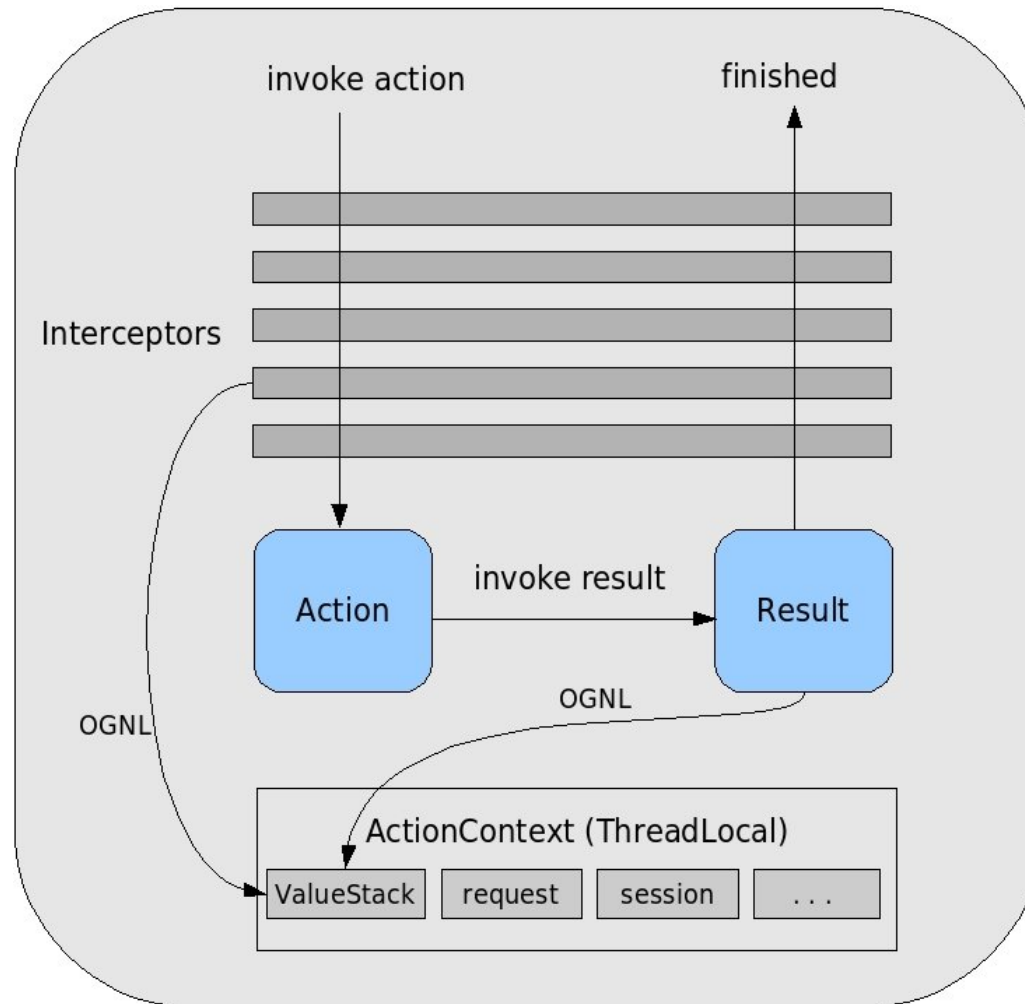
- Google Guice사용, Spring 프레임워크를 통한 DI도 지원가능

## □ 다양한 리절트 타입 지원

- JSP, FreeMarker, Velocity와 같은 다양한 뷰기술을 지원하는 리절트 타입

# 스트럿츠2 등장

## □ 스트럿츠2 Work flow



## 1.6 스트럿츠2 바로 시작하기

- ☐ 웹프로젝트 생성 – 메이븐2
- ☐ 스트럿츠2 기본 환경설정
- ☐ HelloWorld
- ☐ HelloWorld (with Annotation)

ONE STEP AHEAD



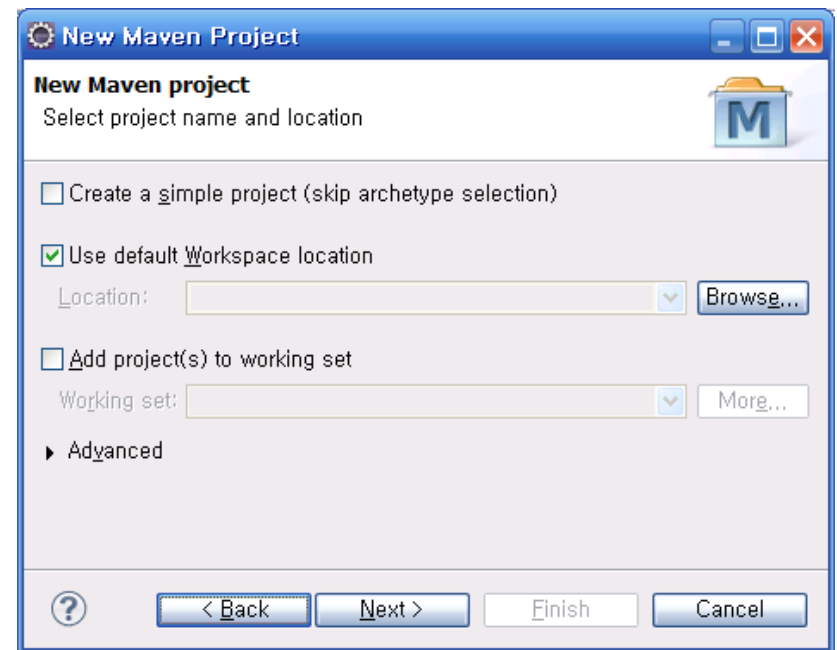
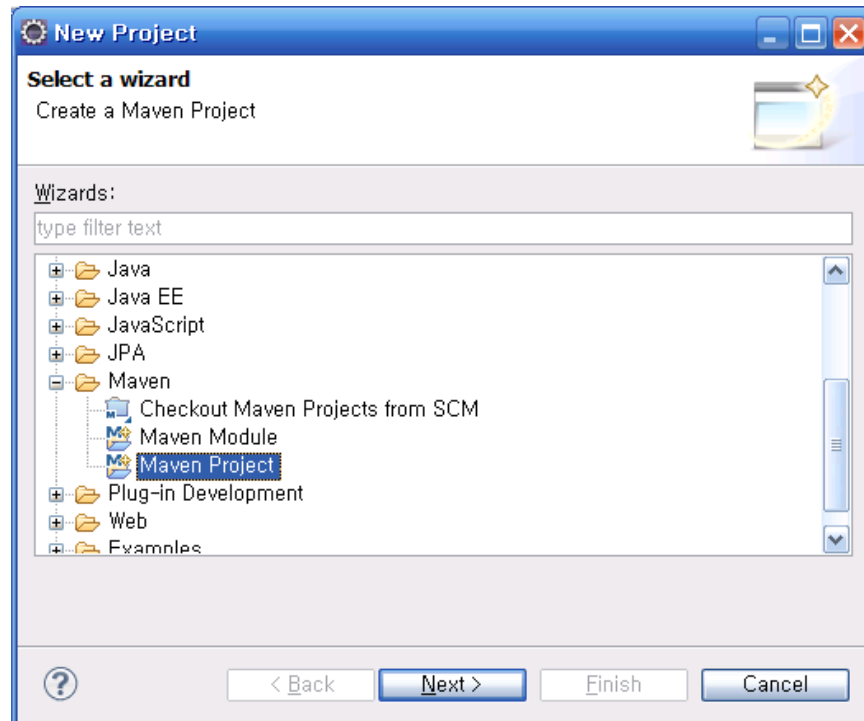
# 웹프로젝트 생성 - 메이븐2

## ❑ M2eclipse plug-in 설치

- 업데이트 사이트 : <http://m2eclipse.sonatype.org/sites/m2e>

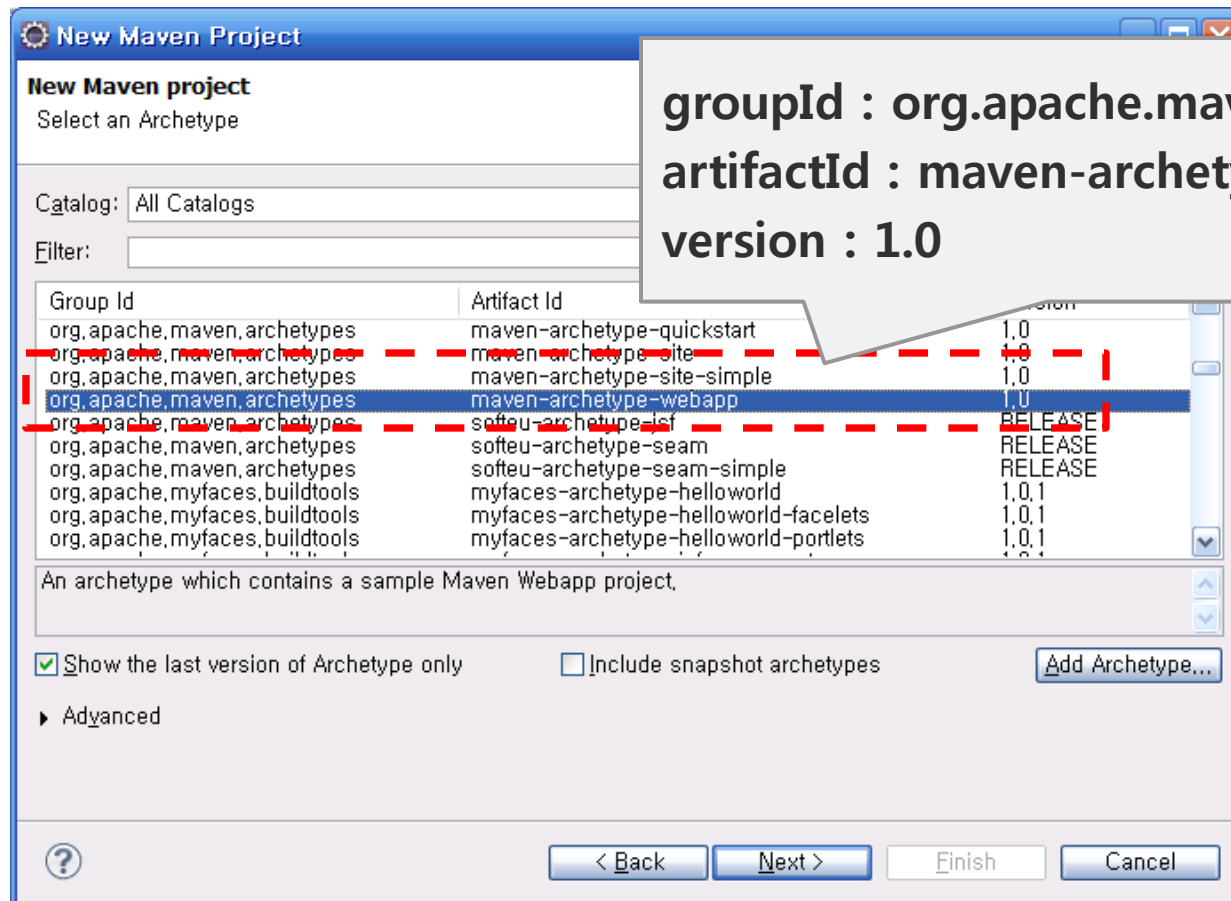
## ❑ 메이븐2를 이용한 웹프로젝트 생성

- File -> New -> Maven -> Maven Project 선택



## 웹프로젝트 생성 – 메이븐2

- archetype 선택 화면에서 maven-archetype-webapp : 1.0 선택

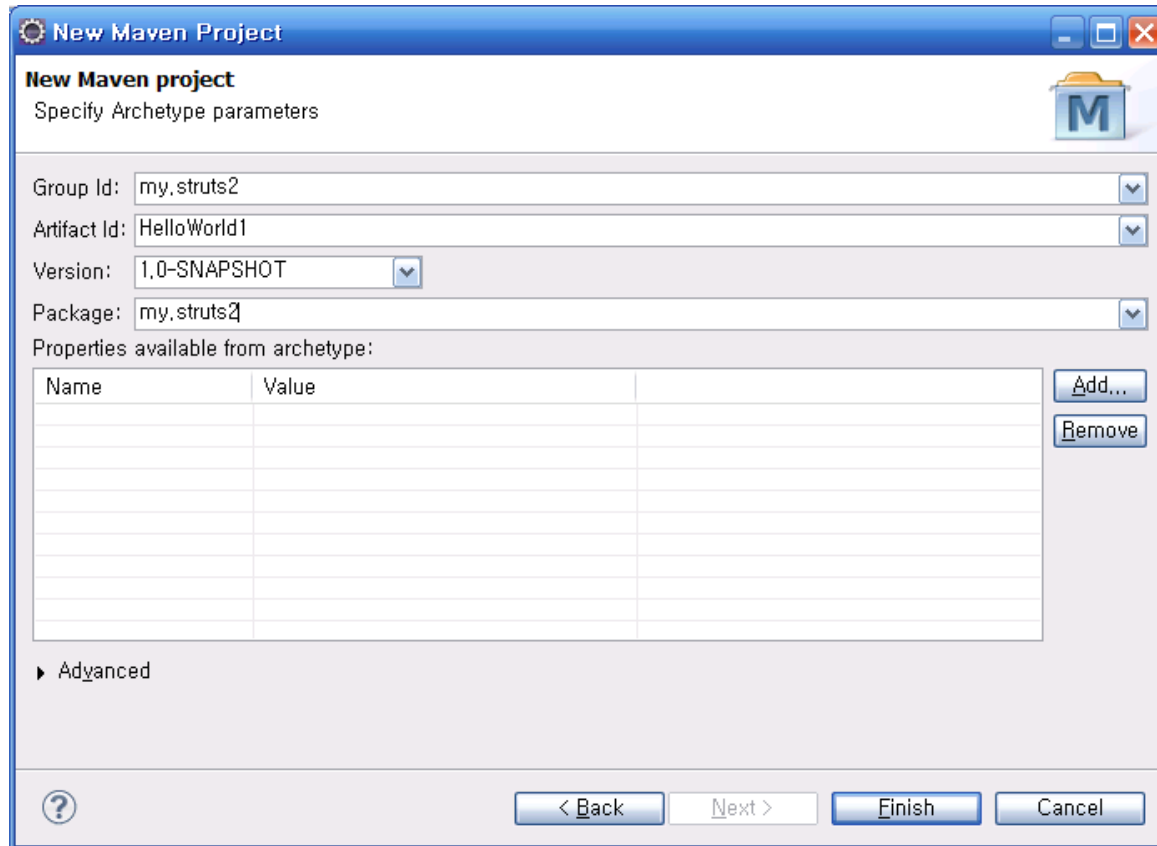


groupId : org.apache.maven.archetypes  
artifactId : maven-archetype-webapp  
version : 1.0



# 웹프로젝트 생성 – 메이븐2

- group Id, artifact Id 입력



**New Maven Project**

**New Maven project**  
Specify Archetype parameters

Group Id: my.struts2

Artifact Id: HelloWorld1

Version: 1.0-SNAPSHOT

Package: my.struts2

Properties available from archetype:

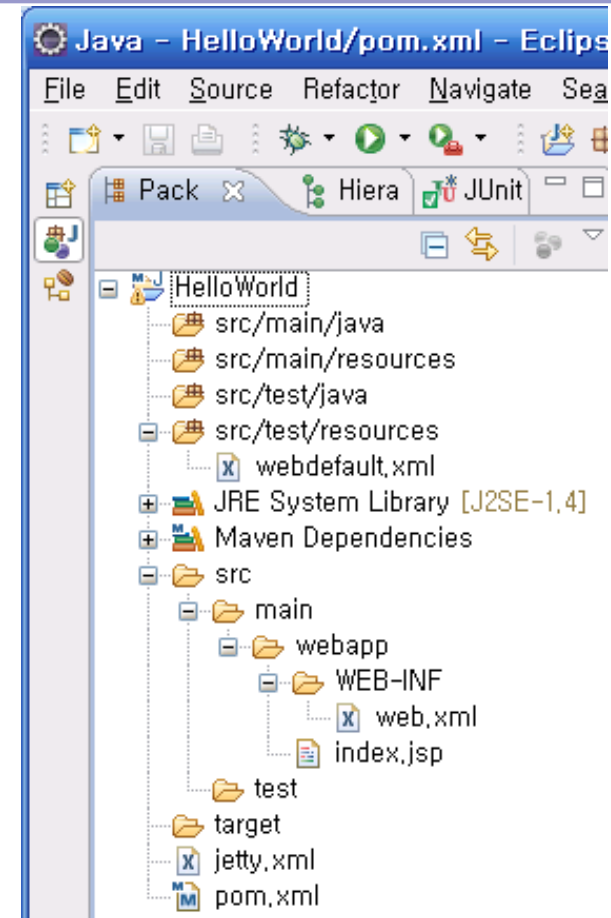
Name	Value

► Advanced

## 웹프로젝트 생성 – 메이븐2

### □ 프로젝트 구조

- src/main/java : 배포시에 필요한 소스
- src/main/resources : 배포시에 필요 자원
- src/test/java : 테스트 소스
- src/test/resources : 테스트시에 필요한 자원
- src/main/webapp : 컨텍스트 루트
- src/target : 빌드 결과물 생성 폴더



- 메이븐2 : 통합빌드툴로서 빌드, 배포, 테스트, 의존성 관리, 프로젝트 모듈화등을 지원하는 도구
- m2eclipse : sonatype사에서 공개한 이클립스 플러그인, 메이븐2의 기능을 지원
- jetty : 경량 컨테이너, 메이븐2 빌드과정에서 실행될 수 있도록 메이븐2 플러그인 형태로 지원

## 웹프로젝트 생성 - 메이븐2

### □ Jetty 컨테이너 플러그인 추가

- pom.xml 의 <build>엘리먼트 내의 내용을 다음 내용과 같이 확장

```

<build>
  <finalName>HelloWorld</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.mortbay.jetty</groupId>
      <artifactId>maven-jetty-plugin</artifactId>
      <configuration>
        <contextPath>/HelloWorld</contextPath>
        <connectors>

```

compile시 사용되는 jdk버전

version 미지정시  
최신 버전의 jetty를 사용

version 미지정시  
최신 버전의 jetty를 사용

```

      <connector implementation=
        "org.mortbay.jetty.nio.SelectChannelConnector">
        <port>8081</port>
        <maxIdleTime>60000</maxIdleTime>
      </connector>
    </connectors>
  </configuration>
</plugin>
</plugins>
</build>

```

# 웹프로젝트 생성 – 메이븐2

## □ Jetty 설정파일 추가

- datasource 및 minetype 설정등을 위해서는 jetty 플러그인에 다음 설정추가

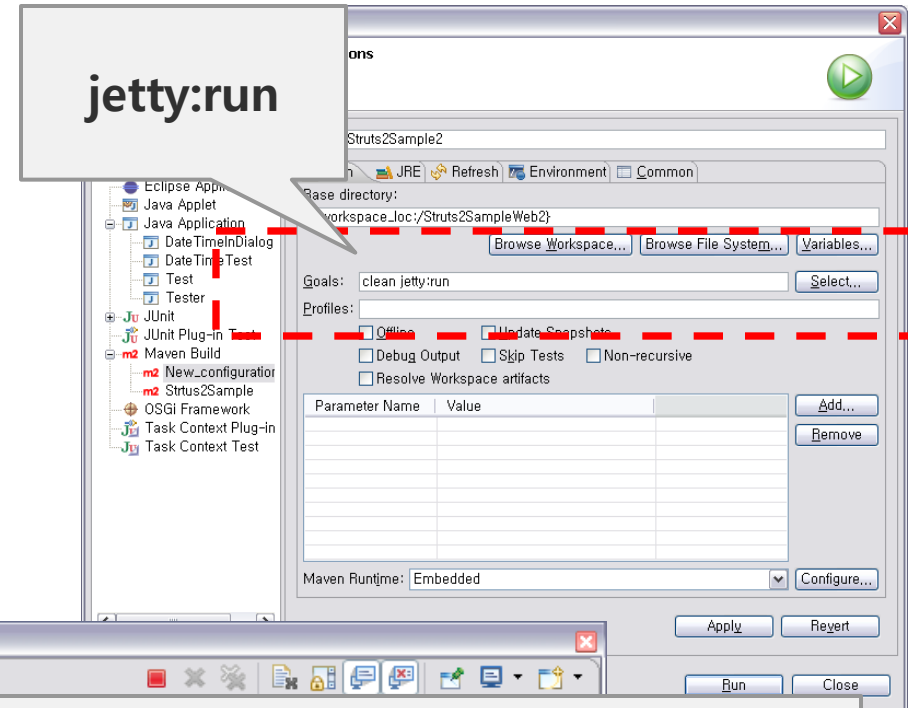
```

<plugin>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>maven-jetty-plugin</artifactId>
  <configuration>
    <contextPath>/HelloWorld</contextPath>
    <connectors>
      <connector implementation="org.mortbay.jetty.nio.SelectChannelConnector">
        <port>8081</port>
        <maxIdleTime>60000</maxIdleTime>
      </connector>
    </connectors>
    <jettyConfig>src/test/resources/jetty.xml</jettyConfig>
    <webDefaultXml>src/test/resources/webdefault.xml</webDefaultXml>
  </configuration>
</plugin>
    
```

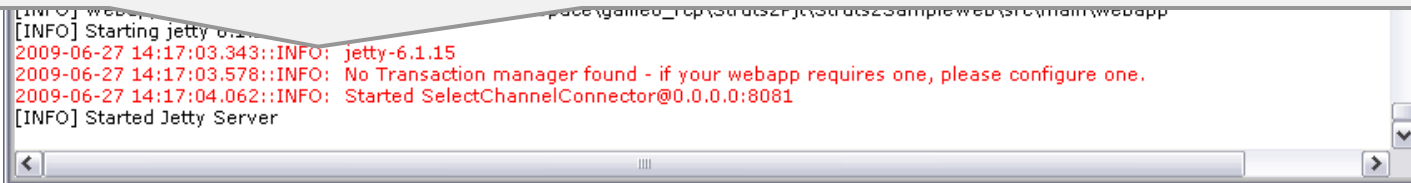
# 웹프로젝트 생성 - 메이븐2

## □ 컨테이너 실행

- Run->Run Configurations
- 메이븐 빌드 추가
- Goals 란에 jetty:run 입력
- Run 버튼 클릭
- Console 창에서 서버 구동 확인
- 브라우저를 통해 서버 접근



2009-06-27 14:17:04.062::INFO: Started SelectChannelConnector@0.0.0.0:8081  
[INFO] Started Jetty Server



# 스트럿츠2 기본 환경설정

## □ 필수 library

- /WEB-INF/lib 폴더에 jar 파일 추가
- Maven2 프로젝트일 경우는 pom.xml 파일에서 의존성 추가

### 2.0.14 버전 필수 라이브러리

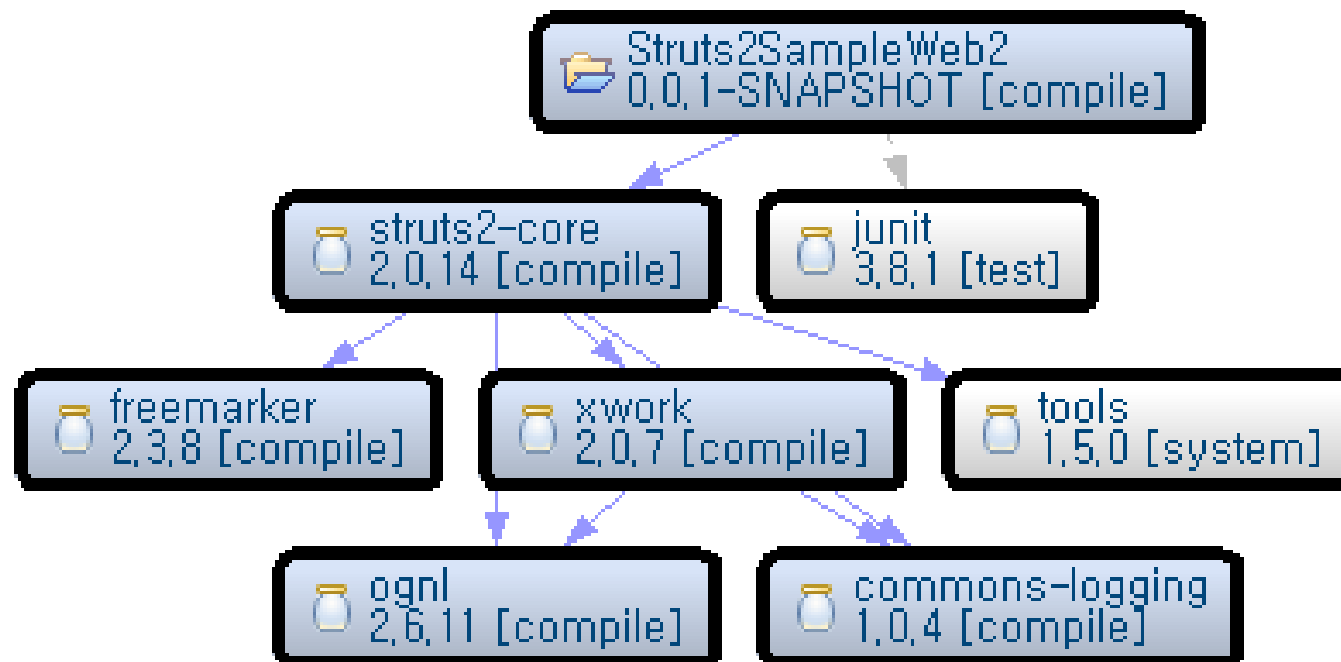
1. commons-logging : 1.0.4
2. freemarker : 2.3.8
3. ognl : 2.6.11
4. struts2-core : 2.0.14
5. xwork : 2.0.7
6. junit : 3.8.1 (test)
7. tools : 1.5.0 (system)

### 2.1.6 버전 필수 라이브러리

1. commons-fileupload : 1.2.1
2. commons-io : 1.3.2
3. commons-logging : 1.1.1
4. freemarker : 2.3.13
5. ognl : 2.6.11
6. spring-test : 2.5.6
7. struts2-core : 2.1.6
8. xwork : 2.1.2
9. junit : 3.8.1 (test)
10. tools : 1.5.0 (system)

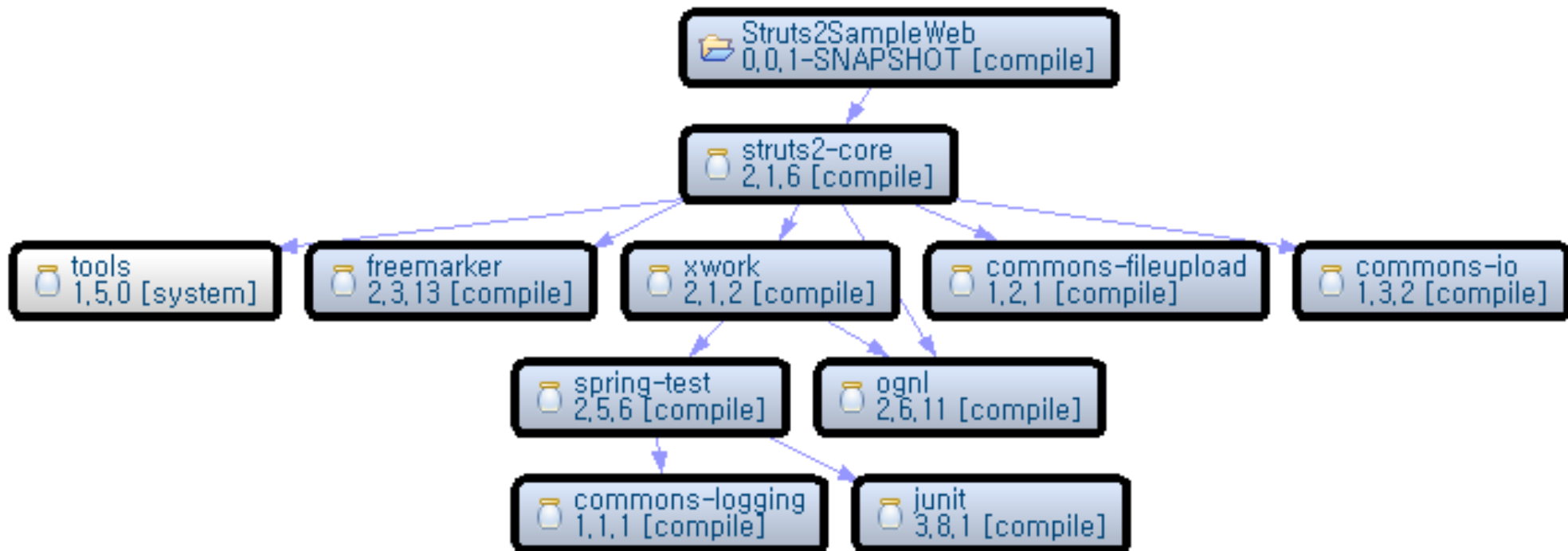
# 스트럿츠2 기본 환경설정

## □ struts2-core-2.0.14 의존성



# 스트럿츠2 기본 환경설정

## □ struts2-core-2.1.6 의존성





# 스트럿츠2 기본 환경설정

## □ 스트럿츠2 필터 등록

- /WEB-INF/web.xml 파일에 filter 설정을 추가한다.

web.xml

```

<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <filter>
        <filter-name>struts</filter-name>
        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>struts</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

</web-app>
    
```

# HelloWorld

## □ Action 클래스

- execute() 메서드와 hello라는 String 타입의 필드를 가지는 클래스 생성

src/example/HelloWorld.java

```
package kr.nexttree.struts2.quickstart.action;

public class HelloWorld {
    private String name;
    private String greeting;

    public String execute(){
        greeting = name + "님 안녕하세요";
        return "success";
    }

    //getters and setters
}
```

# HelloWorld

## □ JSP 페이지

### hello.jsp

```

<%@ page contentType="text/html;
           charset="UTF-8"%>

<html>
  <head><title>Hello world</title> </head>
  이름을 입력해 주세요
  <body>
    <form action="hello.action" method="get">
      이름은 : <input type="text" name="name"/>
    <br/>
    <input type="submit"/>
    </form>
  </body>

</html>
    
```

### greeting.jsp

```

<%@ page contentType="text/html;
           charset="UTF-8"%>

<html>
  <head><title>Hello world</title> </head>

  <body>
    ${greeting}
  </body>

</html>
    
```

# HelloWorld

## □ Action과 result 페이지 매핑

- Struts2 애플리케이션 설정파일인 struts.xml 파일을 생성한다.

src/struts.xml 또는 /WEB-INF/classes/struts.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>

    <package name="quickstart" namespace="/" extends="struts-default">

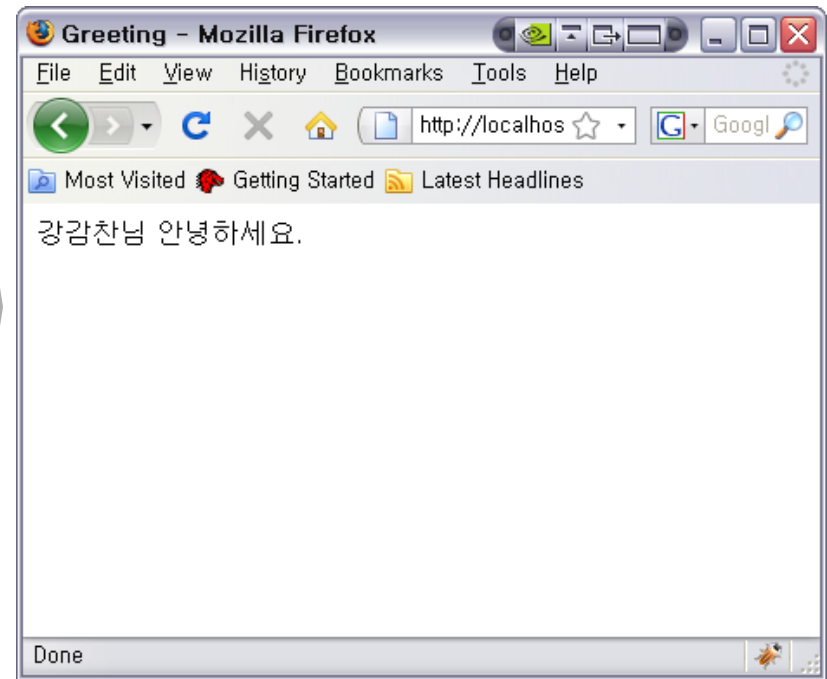
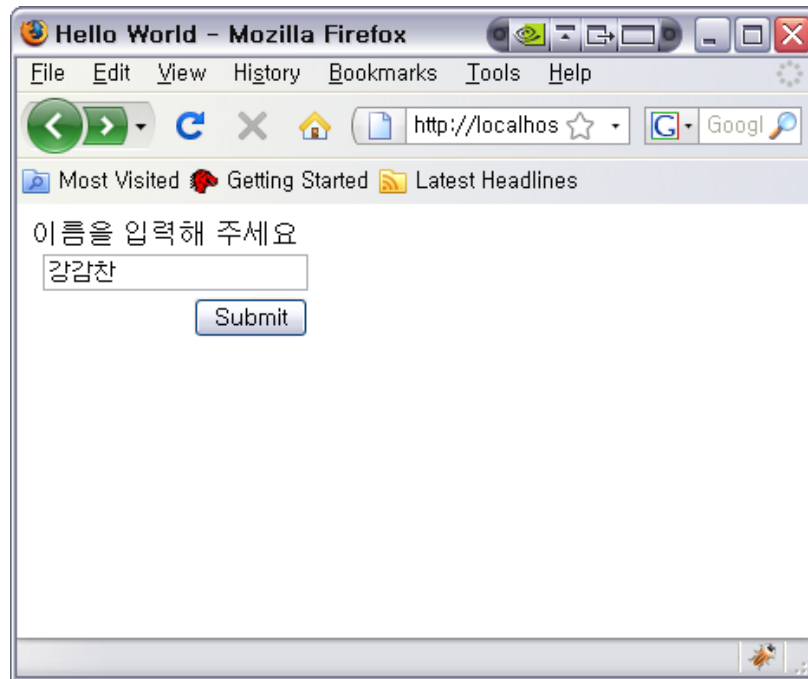
        <action name="hello" class="kr.nexttree.struts2.quickstart.action.HelloWorld">
            <result>/example/HelloWorld.jsp</result>
        </action>

    </package>
</struts>
    
```

# HelloWorld

## □ 결과

- [http://localhost:\[port\]/HelloWorld/hello.jsp](http://localhost:[port]/HelloWorld/hello.jsp) 로 요청



# HelloWorld(with Annotation)



struts-core-2.1.6

# HelloWorld(with Annotation)

## □ 의존성 변경

- struts2-core 라이브러리에서 독립적인 라이브러리로 분리
- struts2-convention-plugin-2.1.6.jar

## □ 스트럿츠2 필터 등록

- struts.xml을 이용한 설정방법과 동일, struts2 필터만 등록

web.xml

```
<web-app>
  <filter>
    <filter-name>struts</filter-name>
    <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

# HelloWorld(with Annotation)

## □ Action 클래스

- 명명규칙 : com.opensymphony.xwork2.Action 인터페이스 구현하거나 클래스 이름이 Action으로 끝남
- @ResultPath : 지정한 경로를 결과페이지의 루트 경로로 인식
- @Results : 반환값에 대한 forward 경로를 지정(name, location 속성을 보유)
- @Action : Action 클래스에 접근하는 경로를 지정



# HelloWorld(with Annotation)

## HelloWorldAction.java

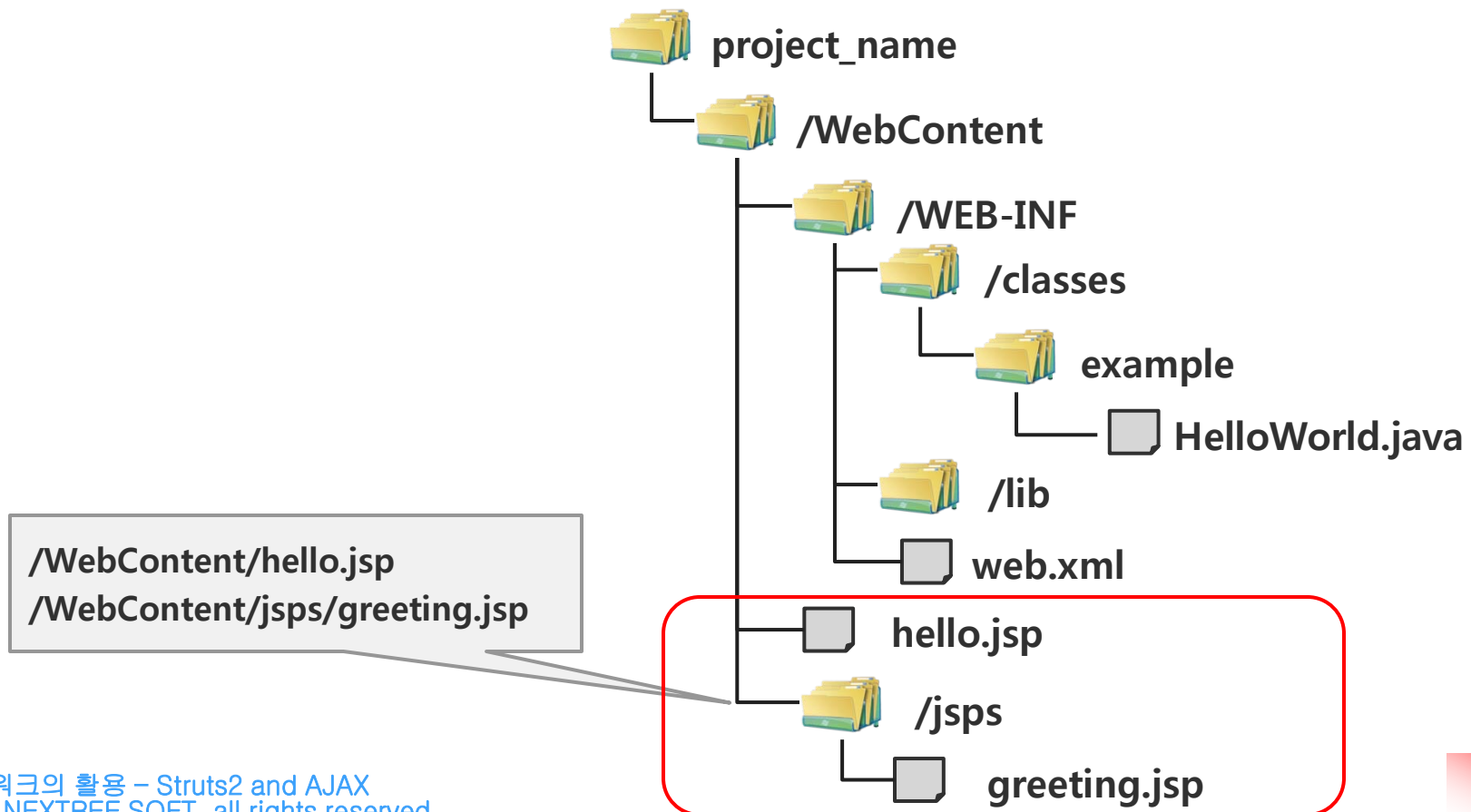
```
package kr.nexttree.struts2.quickstart.action;

@ResultPath("/jsps")
@Results({
    @Result(name="success", location="greeting.jsp")
})
public class HelloWorldAction{
    private String name;
    private String greeting;
    @Action(value="/helloWorld")
    public String execute(){
        this.greeting = this.name + "님 안녕하세요.";
        return "success";
    }
    //getters and setters
}
```

# HelloWorld(with Annotation)

## □ JSP 페이지

- 입력을 담당하는 hello.jsp의 경로는 동일
- 출력을 담당하는 greeting.jsp의 위치만 ResultPath에서 지정한 경로에 위치



# 스트럿츠2 태그 라이브러리 vs JSTL + EL

## □ 스트럿츠2 태그 라이브러리

- Struts2의 많은 기능을 Taglib 차원에서 지원
- ui 코딩량이 현저히 감소
- Dojo - Ajax Framework을 Taglib로 지원하므로 별도의 Ajax Framework 학습을 필요로 하지 않음 (최근 jQuery plug-in 추가)
- properties 파일을 이용한 국제화 지원이 쉬움
- 스트럿츠2 프레임워크에 종속적

## □ JSTL + EL

- jstl.jar, standard.jar 라이브러리 추가만으로 사용이 가능해서 이식성이 좋음
- 프리젠테이션 계층 프레임워크에 대해 독립적