WEB-INF안에 *.JSP를 넣어라?

코드리뷰

A Table of Contents.

Part 1,

WEB-INF 안에 JSP

Part 1,

WEB-INF 안에 JSP

WEB-INF 안에 JSP를 넣는 이유

- JSP/Servlet은 WEB-INF 디렉토리 이하를 보안상의 문제로 웹 브라우저를 통한 접근을 금지하고 있다.
  하지만 포워딩을 통한 접근은 웹 브라우저를 통하지 않기 때문에 가능하다.

자료출처 : http://egloos.zum.com/kwon37xi/v/2850553

Setting

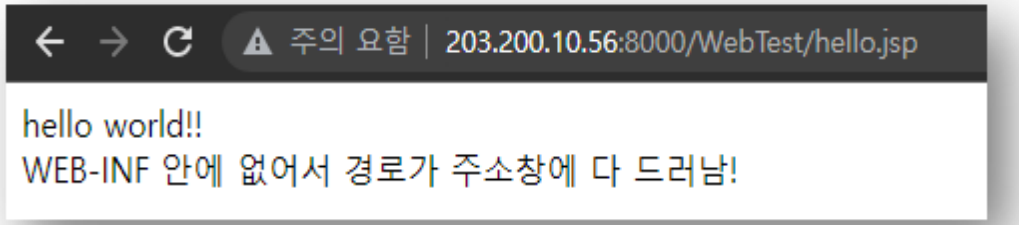
- Web.xml에서 Servlet Mapping을 통해 컨트롤러를 호출한다
- 컨트롤러에서

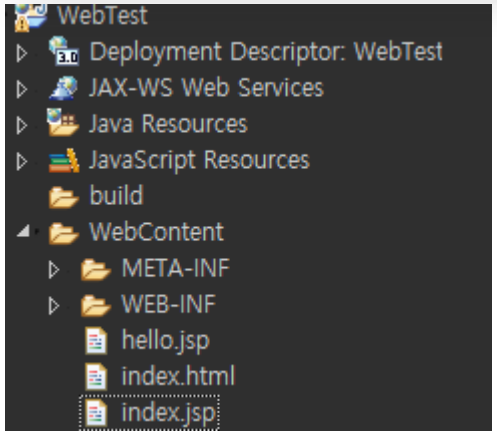
```java
public class hello extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/html/hello.jsp");
        dispatcher.forward(request, response);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }

}
```

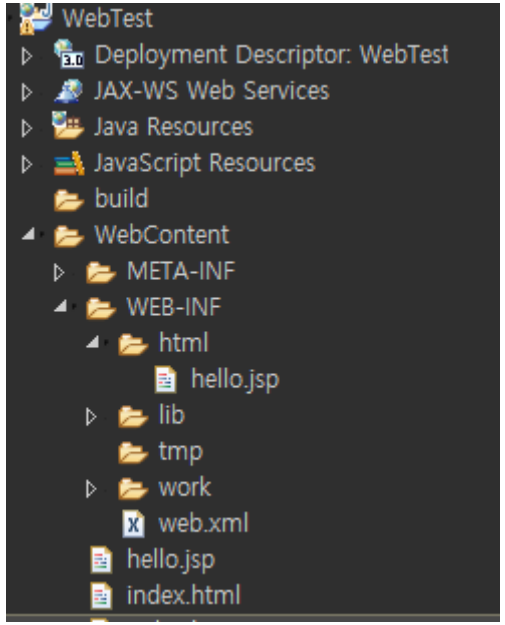
Part 1,

WEB-INF 안에 JSP

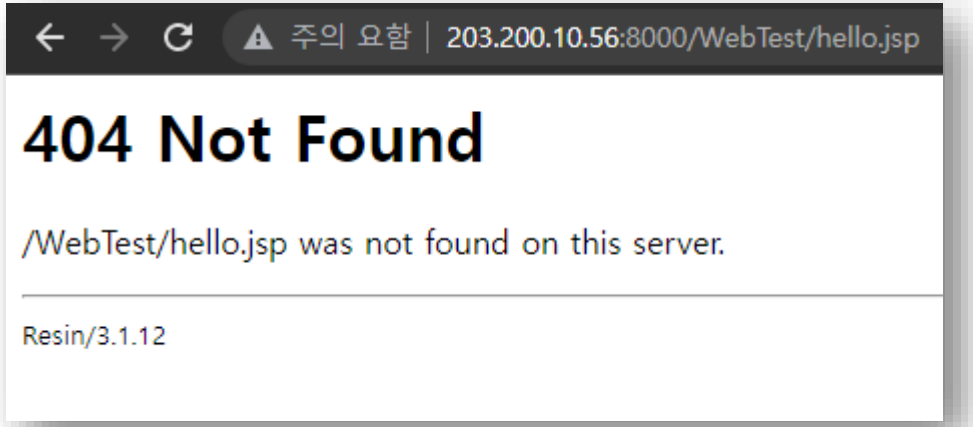
WEB-INF 안에 JSP를 넣지 않았을 때



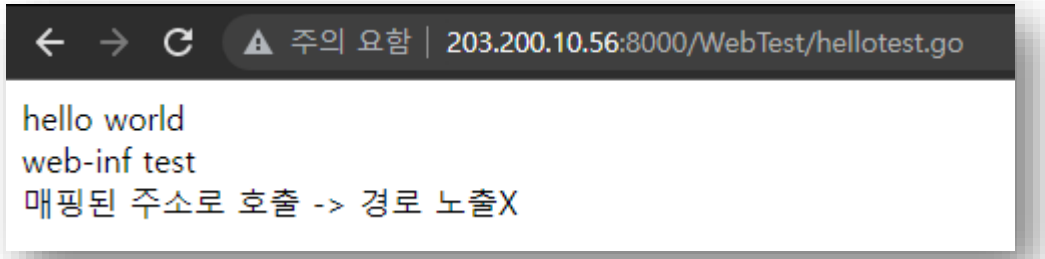




WEB-INF 안에 JSP를 넣고 브라우저 직접 호출 시



WEB-INF 안에 JSP를 넣고 컨트롤러를 통한 호출 시



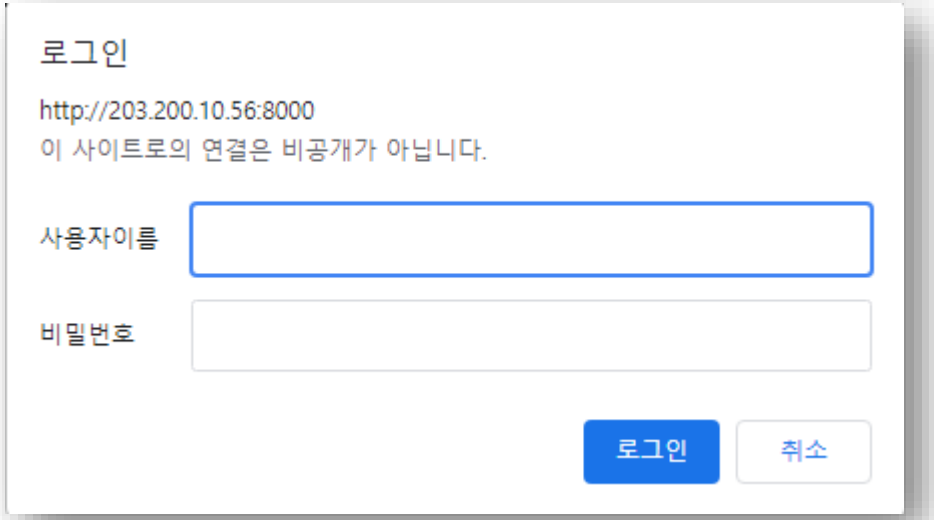

Q.
꼭 WEB-INF안에 넣어야 할까?

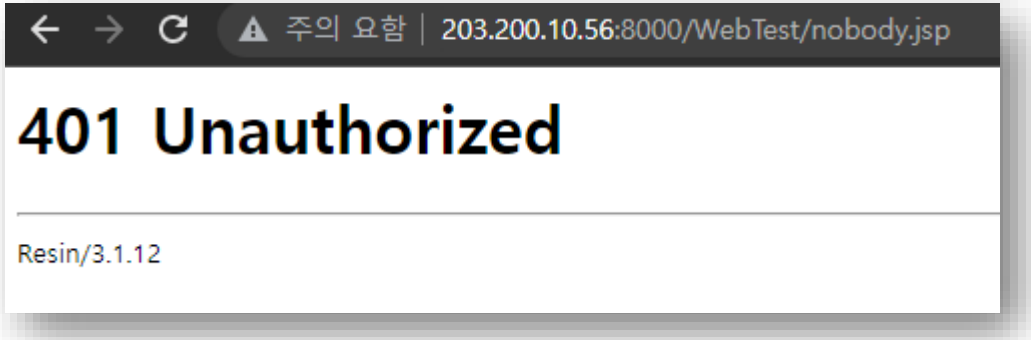
Part 1,

WEB-INF 안에 JSP

Web.xml 안에 경로를 입력시켜주거나 보안처리를 한다

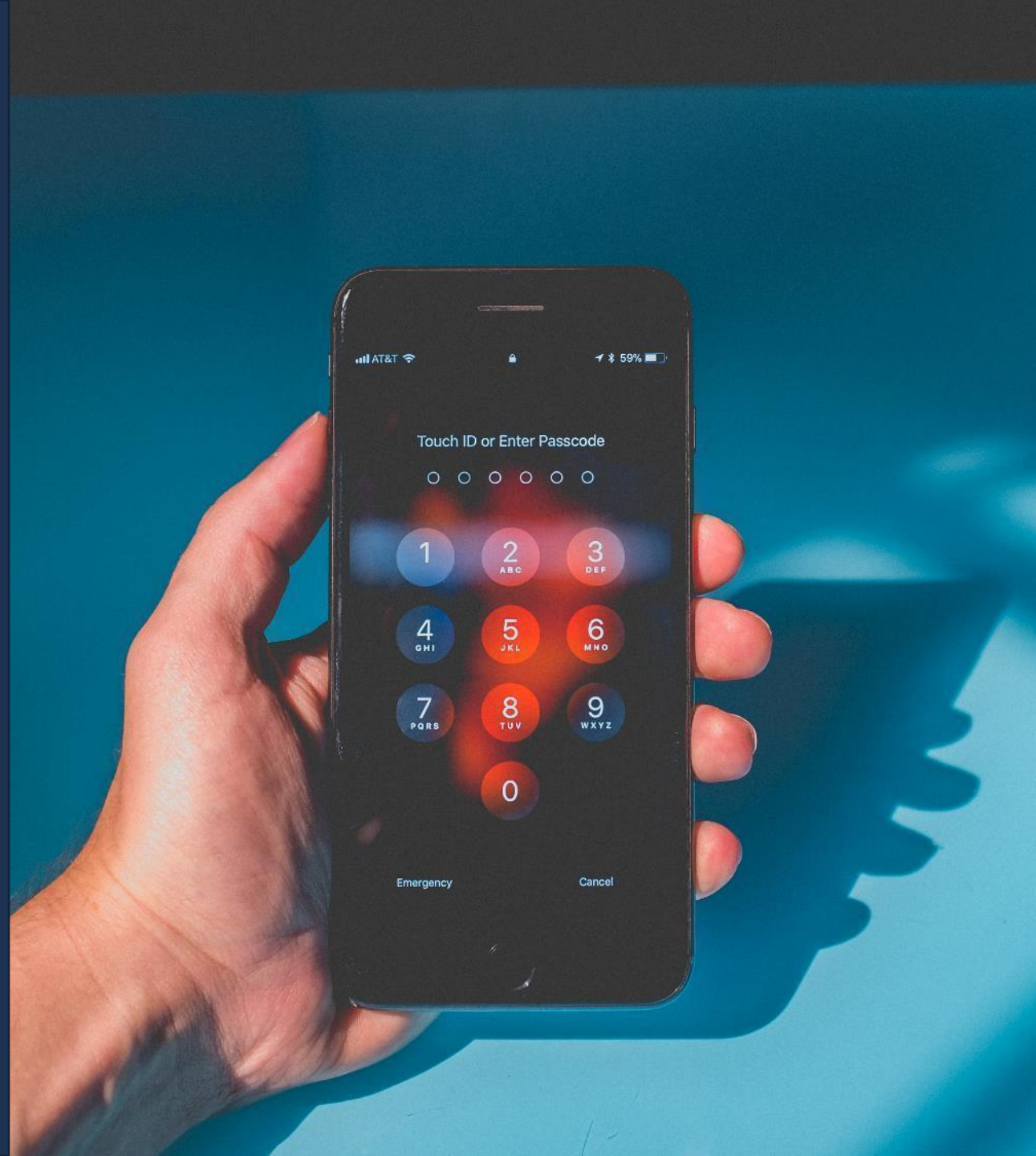
```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>PreventViewingJSPs</web-resource-name>
        <description>브라우저로 접속한 사용자가 JSP파일로 직접 접근할 수 없도록 한다.</description>
        <url-pattern>*.jsp</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name></role-name> <!-- 여기에 role-name을 적어도 된다. 존재하지 않는 것으로 -->
    </auth-constraint>
</security-constraint>
```





Part 2,

Web.xml



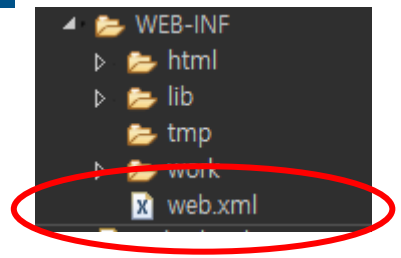
Part 2,

Web.xml

- Web application의 설정을 위한 deployment description
- 모든 Web Application은 반드시 하나의 web.xml파일을 가져야 한다
- Web.xml 파일의 설정들은 Web Application 시작 시 메모리에 로딩 된다
  (수정을 할 경우 **web application**을 재시작 해야함.)

존재 이유 WAS 구동 시, /WEB-INF 안의 Web.xml을 읽어 Web Application 의 설정을 구성하기 위해 존재

위치 **WEB-INF 폴더 아래**



사용이유 https://wanna-b.tistory.com/87 참고

Part 2,

Web.xml

예제

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
  <display-name>WebTest</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>controller.hello</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hellotest.go</url-pattern>
    </servlet-mapping>

</web-app>
```

1. 알리아스 설정

<servlet-class>controller.hello</servlet-class>은
실제 작성된 클래스 이름으로 하여야 한다

2. 매핑

- Url을 서블릿 이름에 연결
- <servlet-class>controller.hello</servlet-class>은
  클라이언트의 요청 url에서 프로젝트 이름 뒤에 오는 부분으로
  슬래시('/')로 시작되어야 한다

참고 클라이언트(browser)가 요청하는 URL 정보

- 요청을 보낼 서버의 IP 주소 : Port 번호 / App 이름 / 달라고 요청하는 HTML
  - Ex. `localhost:8080/FormHandlingServlet/LoginForm.html`

Part 3,

Web.xml 설정

Part 3,

Web.xml 설정

DispatcherServlet

클라이언트의 요청을 전달받는 객체

1. 클라이언트의 요청을 처리해 줄 컨트롤러를 찾는다.
   (Handler Mapping)
2. 컨트롤러를 실행한다.
   (Handler Adapter)
3. 클라이언트에게 보여줄 View를 찾는다.
   (View Resolver)
4. 응답데이터와 View를 클라이언트에게 전달한다

ContextLoaderListener

Appleication context 단위의 설정 처리

encodingFilter

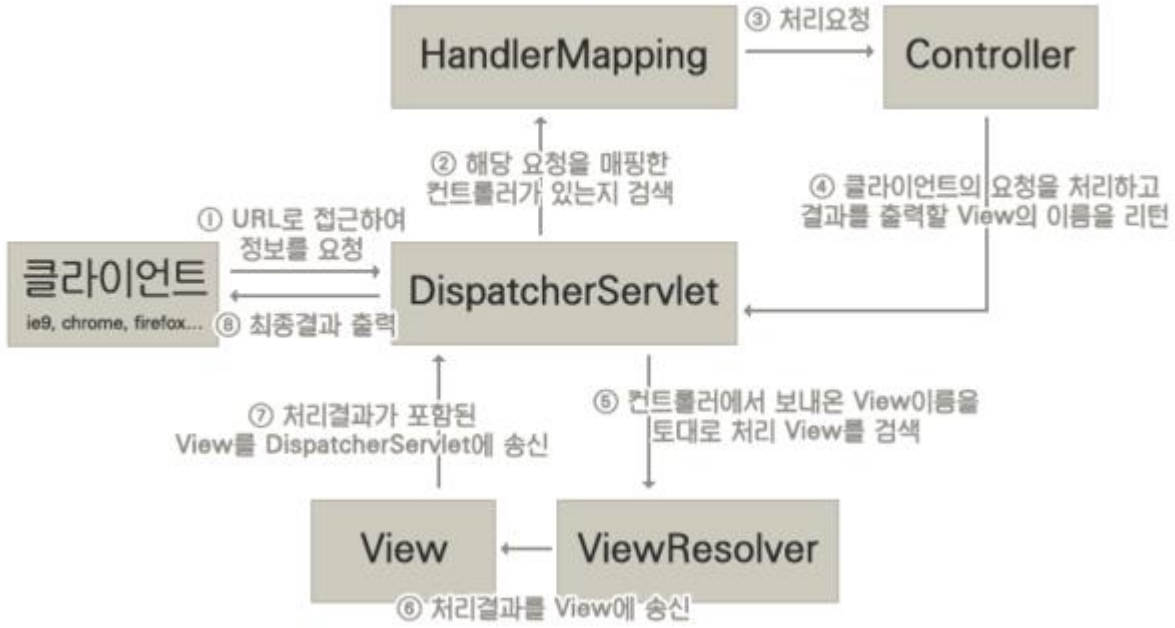
클라이언트에서 온 요청을
Dispatcher Servlet이 받기 전 거치는 부분

Part 3,

Web.xml 설정 - DispatcherServlet

DispatcherServlet

[Dispatcher-Servlet의 흐름]



사진출처 : https://mangkyu.tistory.com/18

1. URL로 접근하여 정보를 요청
2. 해당 요청을 매칭한 컨트롤러가 있는지 검색
3. 처리요청
4. 클라이언트의 요청을 처리하고 결과를 출력할 View의 이름을 리턴
5. 컨트롤러에서 보내온 View이름을 토대로 처리 View를 검색
6. 처리결과를 View에 송신
7. 처리결과가 포함된 View를 DispatcherServlet에 송신
8. 최종결과 출력

Part 3,

Web.xml 설정 - DispatcherServlet

DispatcherServlet 단점

모든 요청을 처리하다보니 정적인 파일이나 HTML 파일을 불러오는 요청마저 전부 Controller로 넘겨버리게 된다.

게다가 JSP파일 안의 JAVASCRIPT나 CSS 파일에 대한 요청까지도 가로채기 때문에 불러오지 못하는 상황도 발생하기도 한다.

해결방안

1. 정적 자원에 대한 요청과 어플리케이션에 대한 요청을 분리
2. 어플리케이션에 대한 요청을 처리하고 없으면 정적 자원에 대한 요청으로 처리

도움 : https://jaeho214.tistory.com/50


Q&A

감사합니다.