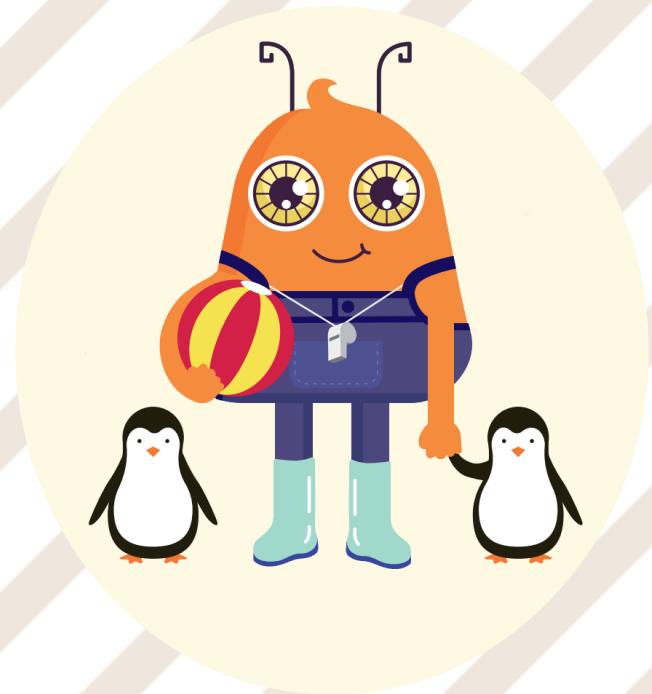


9

CHAPTER

War 제작 및 deploy



Contents

01 war제작

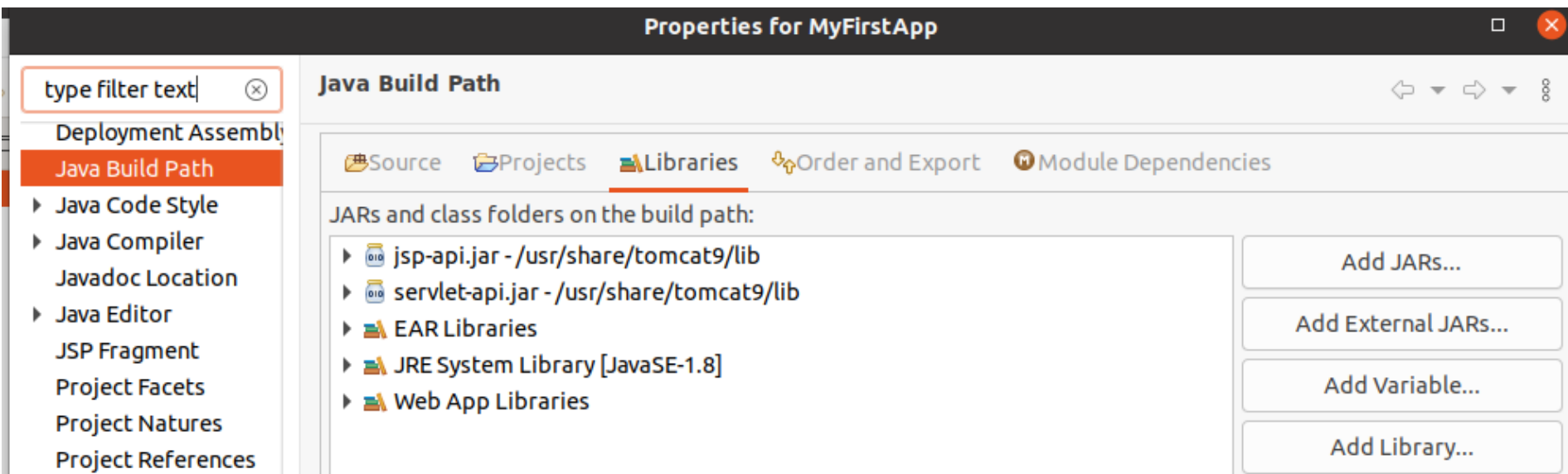
02 web-was 설정 변경

학습목표

- Servlet을 실행하고 war를 만든다.
- 그에 맞는 web/was 설정을 수정한다.

개발전 미리 해야 할일 - jdk 버전 맞추기

- Tomcat에서 사용하는 jdk버전, servlet build할때 jdk버전 맞춰야 함
- Servlet을 build 하려면 Servlet-api.jar, jsp-api.jar 라이브러리 필요
 - JAVA EE 라이브러리라서 따로 찾음
 - \$TOMCAT_HOME/lib/servlet-api.jar, jsp-api.jar 에서 copy함
 - \$TOMCAT_HOME은 tomcat이 동작하는 곳 - /usr/share/tomcat9
- Eclipse 설정 변경
 - Project – Properties – Java Build Path – JRE System Library [Java SE 1.8을 선택]
 - Jsp-api.jar, servlet-api.jar 가져오기



개발전 미리 해야 할일 - jdk 버전 맞추기

- JRE 버전 변경하기 JavaSE-17 => JavaSE-1.8

The screenshot shows the Eclipse IDE interface. The 'Properties for Greeting' dialog is open, with the 'Java Build Path' tab selected. In the 'Libraries' section, 'JRE System Library [JavaSE-17]' is highlighted. An 'Edit Library' dialog is also open, showing the 'JRE System Library' configuration. The 'Execution environment' is set to 'JavaSE-1.8 (jre)'. The 'System library' section has three options: 'Execution environment' (selected), 'Alternate JRE', and 'Workspace default JRE'. The 'Execution environment' dropdown is currently set to 'JavaSE-1.8 (jre)'. The 'Edit Library' dialog also includes buttons for 'Environments...', 'Installed JREs...', and 'Migrate JAR File...'. The 'Apply' button is at the bottom right of the 'Edit Library' dialog.

Properties for Greeting

Java Build Path

JARs and class folders on the build path:

- Modulepath
 - JRE System Library [JavaSE-17]
- Classpath
 - EAR Libraries
 - Web App Libraries

Edit Library

JRE System Library

Select JRE for the project build path.

System library

☒ Execution environment: JavaSE-1.8 (jre) Environments...

☐ Alternate JRE: Installed JREs...

☐ Workspace default JRE (jre)

Apply

Eclipse와 tomcat 연동

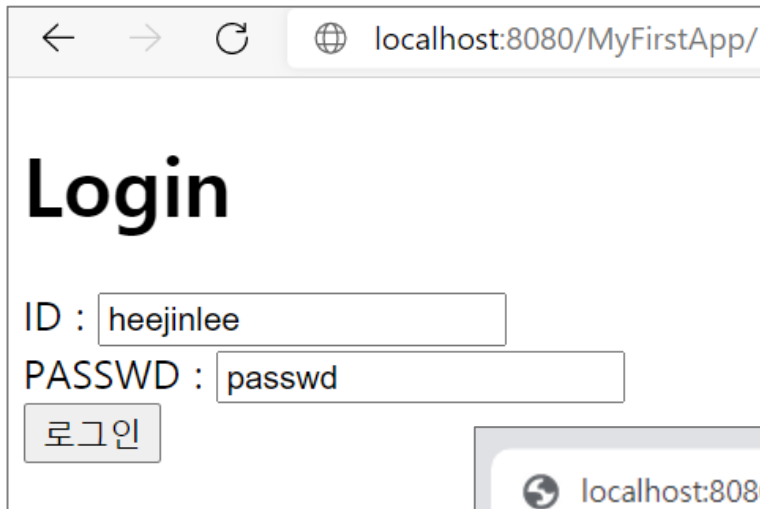
- Eclipse에서 미리 test해보고 war를 생성한다.
 - Run on server - tomcat 9
 - Apache-tomcat과 연동된 가상머신 말고, 다른 가상머신에서 test할것
 - 설정이 충돌나서 시간이 많이 걸림
- 정상 동작을 확인하고 war를 만들어서 was에 배포한다.

[실습 1] login servlet

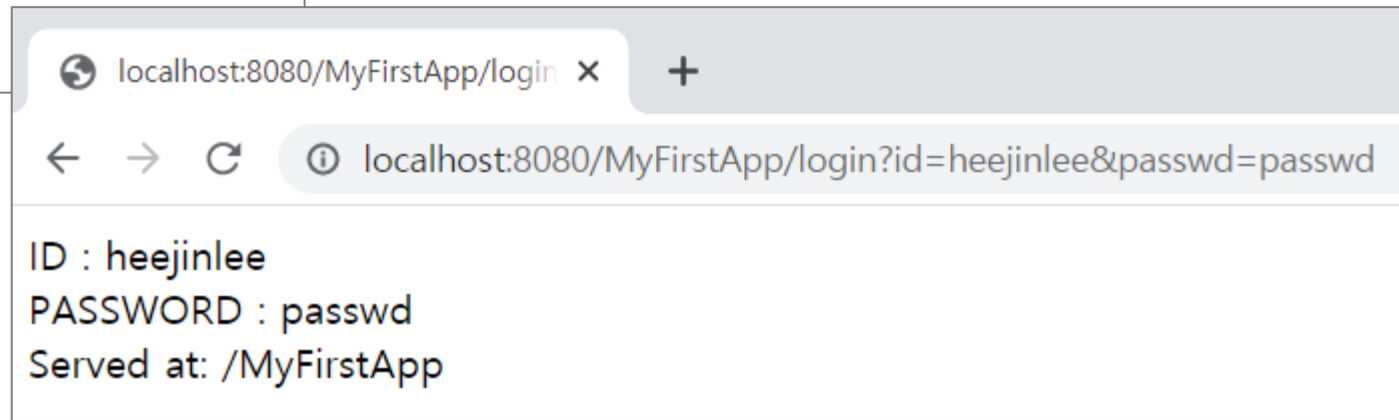
1. login servlet 실행

```
http://localhost:8080/MyFirstApp/index.html  
http://localhost:8080/MyFirstApp
```

```
http://localhost:8080/MyFirstApp/login?id=heejinlee&  
passwd=passwd
```



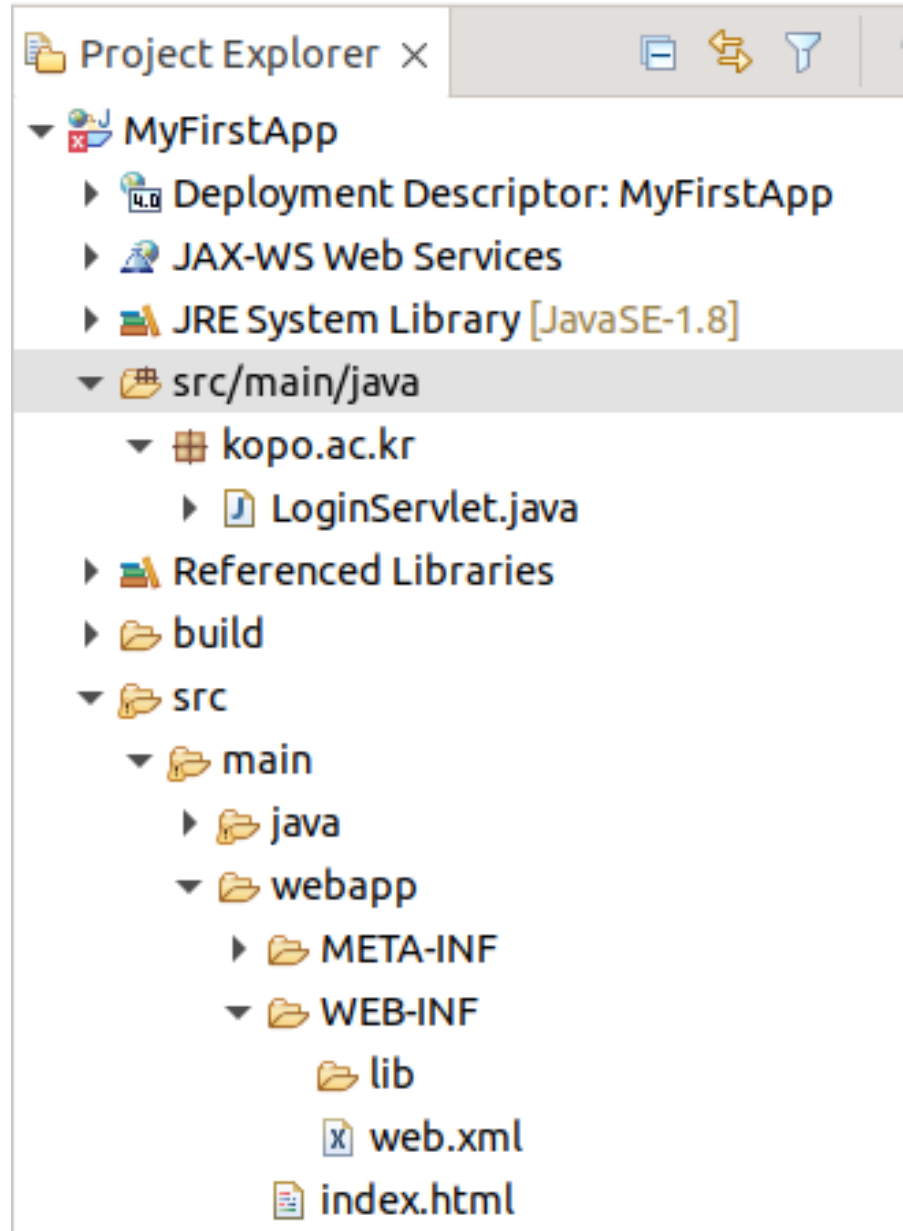
A screenshot of a web browser window. The address bar shows 'localhost:8080/MyFirstApp/'. The page content includes a large 'Login' heading, followed by 'ID : heejinlee' and 'PASSWD : passwd' with corresponding input fields. A '로그인' (Login) button is at the bottom.



A screenshot of a web browser window showing the login page after a successful login. The address bar shows 'localhost:8080/MyFirstApp/login?id=heejinlee&passwd=passwd'. The page content displays 'ID : heejinlee', 'PASSWORD : passwd', and 'Served at: /MyFirstApp'.

[실습 1]

- Servlet 을 작성해 보자.
- New- Dynamic Web Project
 - Generate web.xml



[실습 1] index.html

■ Src/main/webapp/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title> MyFirstApp-Login </title>
</head>
<body>

<h1> Login </h1>
<form action="login" method="get">
<label id= "id_id"> ID    : </label>
<input type = "text" id="id_id" name="id" /><br>
<label id= "id_passwd"> PASSWD    : </label>
<input type = "text" id="id_passwd" name="passwd"/><br>
<input type = "submit" value = "로그인"/>
</form>

</body>
</html>
```

[실습 1] LoginServlet.java

- Src/main/java/kopo.ac.kr/LoginServlet.java

```
package kopo.ac.kr;
public class LoginServlet extends HttpServlet{
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String id = request.getParameter("id");
    String passwd = request.getParameter("passwd");

    response.setContentType("text/html;charset=euc-kr");
    PrintWriter out = response.getWriter();
    out.println("ID : " + id + "<br>");
    out.println("PASSWORD : " + passwd + "<br>");

    response.getWriter().append("Served at:
").append(request.getContextPath());
}
}
```

[실습 1] WEB-INF/web.xml

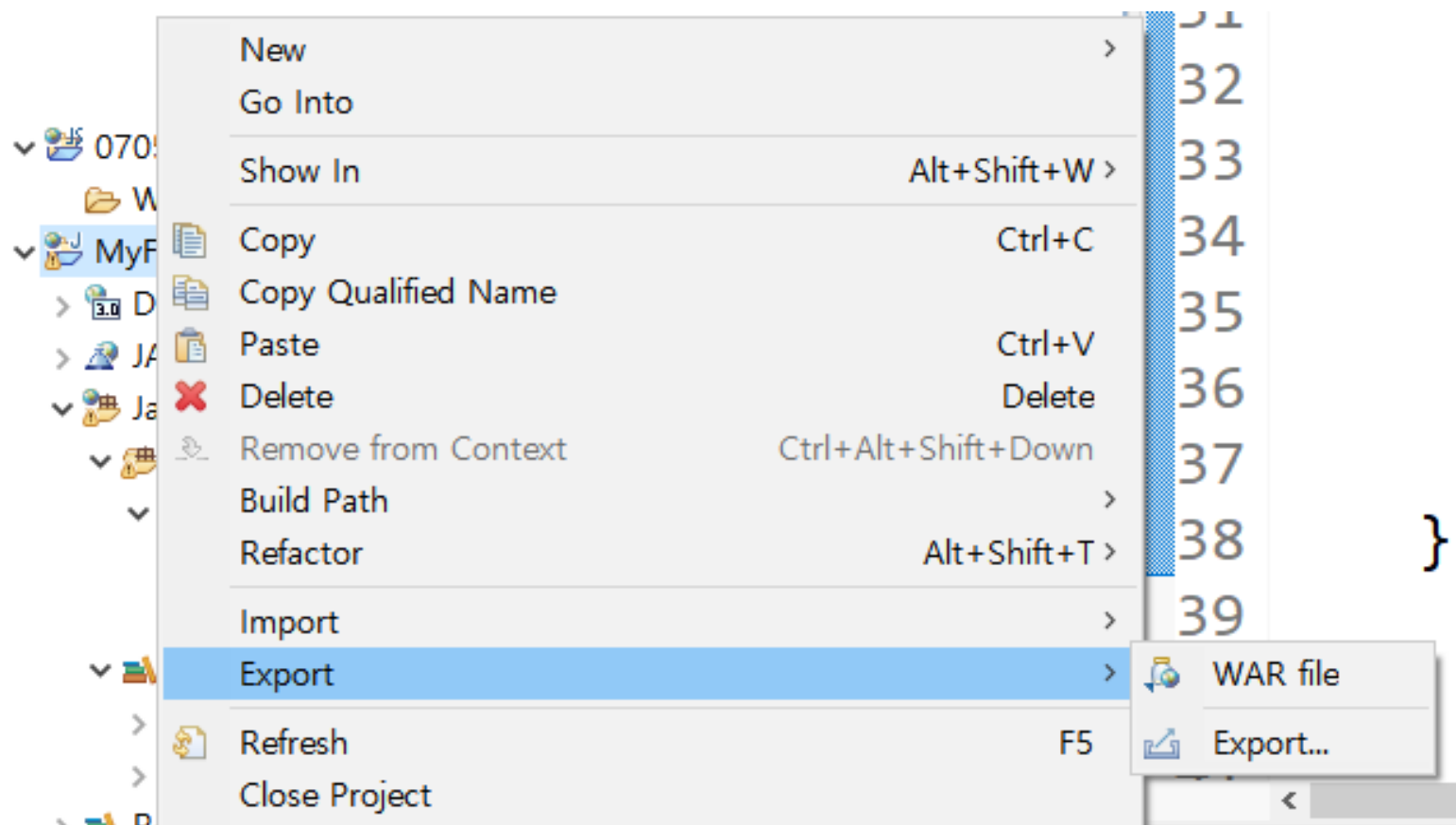
- webapp/WEB-INF/web.xml
- Web.xml 에 servlet mapping 넣어줌

```
.....
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  .....
</welcome-file-list>

<servlet>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-class>kopo.ac.kr.LoginServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/login</url-pattern>
</servlet-mapping>
</web-app>
```

[실습 1] War export



[실습 1]

- Servlet을 호출할 때, Tomcat의 주소를 쓰지 않아도 된다.
 - Tomcat의 주소가 노출되지 않음. Ip, port모두
 - Tomcat의 주소가 변경가능

[실습 2] apache와 tomcat 둘다 활용

- DocumentRoot는 apache에서 처리.
- MyFirstApp은 Tomcat인 ajp13_Worker가 처리

```
# value is not decisive as it is used as a last  
# However, you must set it for any further vir  
ServerName 127.0.0.1
```

```
ServerAdmin webmaster@localhost  
DocumentRoot /home/heejinlee/public_html
```

```
#JkMount /* ajp13_worker  
#JkMount / ajp13_worker
```

```
JkMount /MyFirstApp/ ajp13_worker  
JkMount /MyFirstApp/* ajp13_worker  
# Available loglevels: trace8, ..., trace1, de  
# error crit alert emerg
```

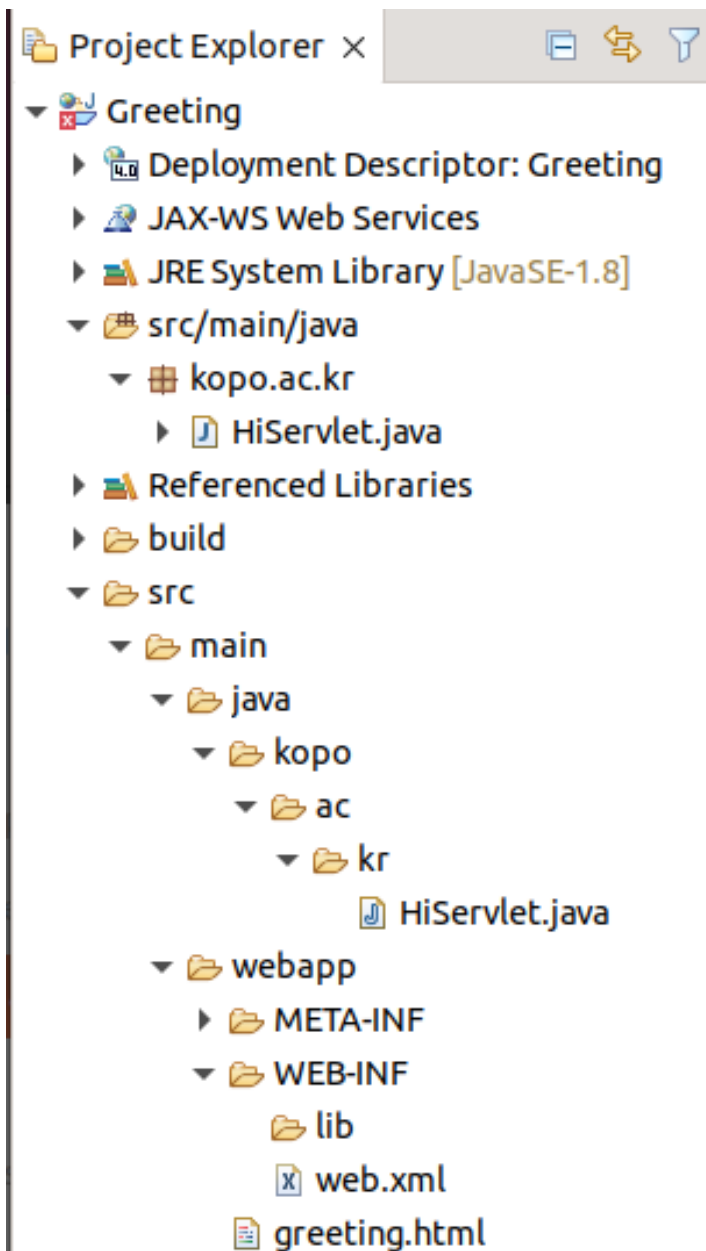
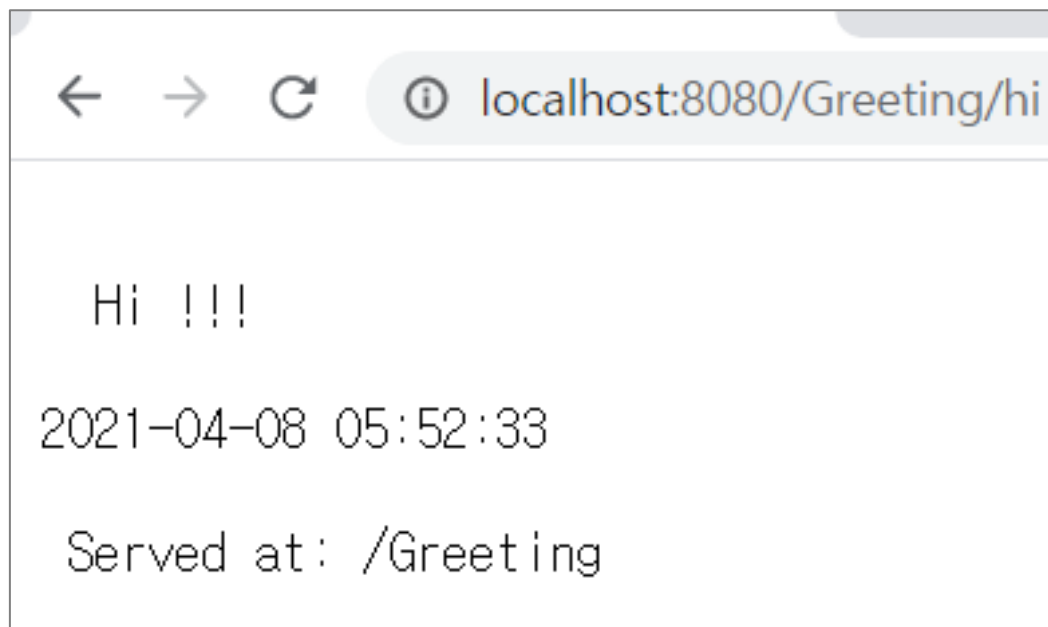
[실습 2] apache와 tomcat 둘다 활용

- Apache
 - Index.html
 - Static file, apache document root 하위의 directory에 대한 Link들
 - <http://localhost> 접속하면 apache의 index.html이 보임
- Tomcat
 - MyFirstApp deploy
 - Greeting deploy
- 설정
 - MyFirstApp, Greeting인 경우 tomcat으로 요청 보내기
 - 나머지는 apache에서 담당

[실습 2] apache와 tomcat 둘다 활용

- Greeting app
 - 인사와 시간을 찍어주는 servlet 하나만 존재

<http://localhost:8080/Greeting/hi>



[실습 2] apache와 tomcat 둘다 활용

- Greeting app
 - Src/HiServlet.java

```
protected void doGet
(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub

SimpleDateFormat format1 =
    new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
Date time = new Date();
String time1 = format1.format(time);

response.getWriter().println("\n Hi !!! \n");
response.getWriter().println(time1);

response.getWriter().append("\n Served at: ")
    .append(request.getContextPath());

}
```

[실습 2] apache와 tomcat 둘다 활용

- Greeting app
 - Src/main/webapp/WEB-INF/Web.xml

```
<servlet>
  <servlet-name>HiServlet</servlet-name>
  <servlet-class> kopo.ac.kr.HiServlet </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HiServlet</servlet-name>
  <url-pattern>/hi</url-pattern>
</servlet-mapping>
```

[실습 2] apache와 tomcat 둘다 활용

■ Greeting app

- Src/main/webapp/index.html
 - 동작 확인용

Success!!
IP/Greeting/hi 를 치세요.

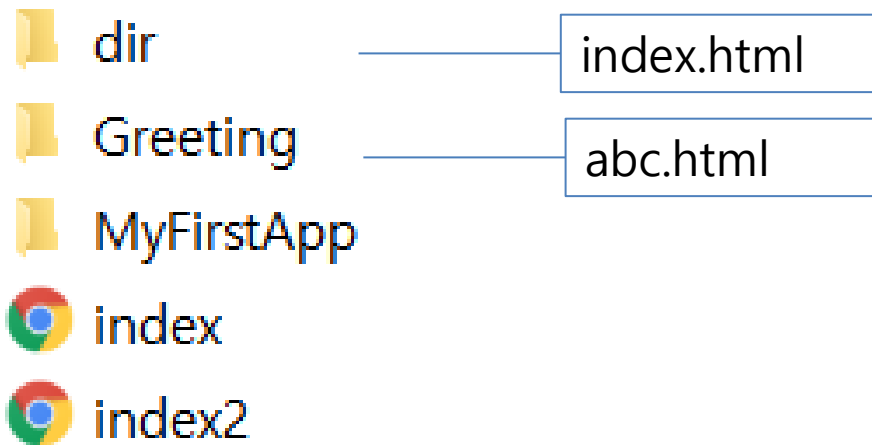
- Src/main/webapp/greeting.html
 - 동작 확인용

안녕하세요.
TOMCAT에 설치된 Greeting application 안에 있는
greeting.html입니다.

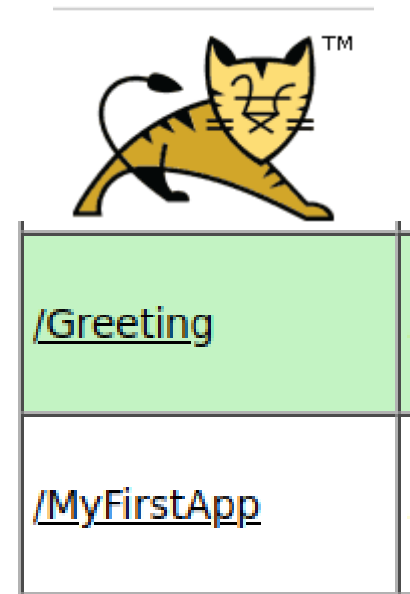
IP/Greeting/greeting.html

[실습 2] apache와 tomcat 둘다 활용

- Apache Document Root에 index.html과 directory 추가
 - dir 추가
 - 요청이 tomcat으로 가는지, apache로 오는지 확인
 - Directory MyFirstApp 추가
 - Directory Greeting 추가 – 하위에 abc.html 추가
 - Directory dir 추가 – 하위에 index.html추가



<Apache DocumentRoot directory>



<Tomcat 에 deploy된 app들>

[실습 2] apache와 tomcat 둘다 활용

■ Apache DocumentRoot/index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Hello Hello</title>
</head>
<body>
    <p> <a href="index2.html">index2.html in the same directory</a> </p>
    <p> <a href="/MyFirstApp/">Login</a> </p>
    <p> <a href="/Greeting/">context path가 Greeting인 경우 - Greeting app에
    index.html이 없는경우 동작 안함, 현재는 index.html이 있기 때문에 동작함</a>
    </p>
    <p> <a href="/Greeting/greeting.html">/Greeting/greeting.html 실행 - 정상
    동작</a></p>
    <p> <a href="/Greeting/hi">/Greeting/hi 실행 - 정상 동작</a> </p>
    <p> <a href="/Greeting/abc.html">apache에는 존재하는 Greeting/abc.html -
    동작 안함</a></p>
    <p> <a href="/dir">index.html in the director "dir"- Apache - 정상 동작
    </a></p>
</body>
</html>
```

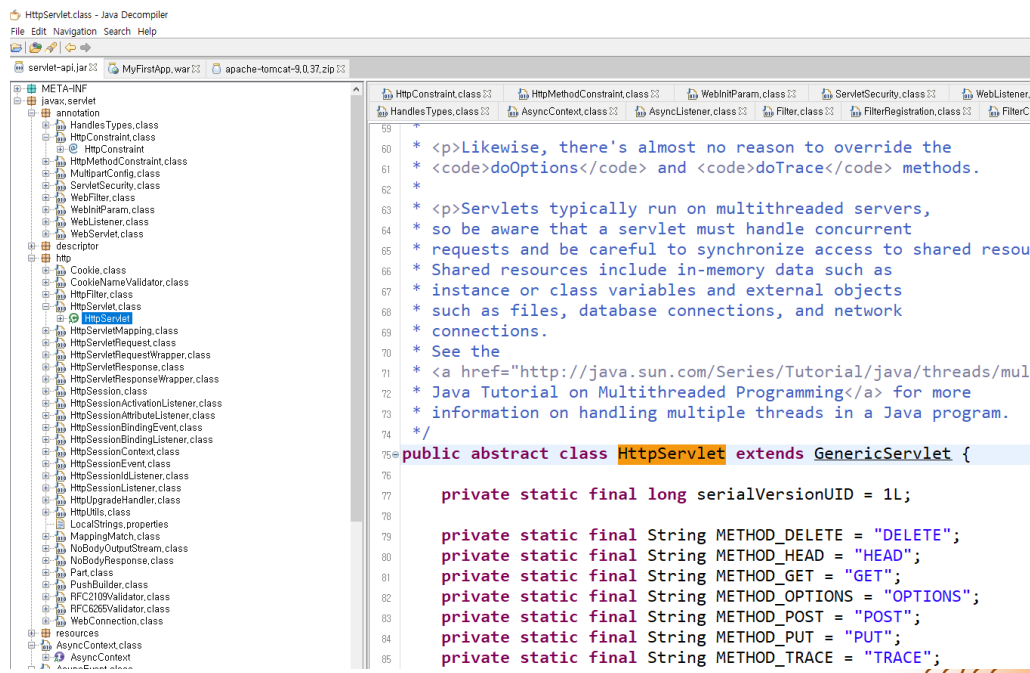
[실습 3] Decompiler

- War는 compile된 java 코드인 class로 이루어짐.
- War로 배포시 java source code를 같이 배포하지 않는경우가 대부분
 - 보안상의 문제 – 코드를 알면 취약점 파악 용이
 - 회사 대외비 – 코드를 알면 원천 기술 공개되므로 (solution 회사들)

■ Java Decompiler

- Class로 이루어진 war를 java source file로 볼수 있게 해주는 것

- <http://java-decompiler.github.io/>



[실습 4] Weblogic 설치 해보자

- 스스로 설치 해보자.

[실습 5] JEUS 설치

[실습 6] Google cloud 에 나만의 웹서버 구축



Thank You
