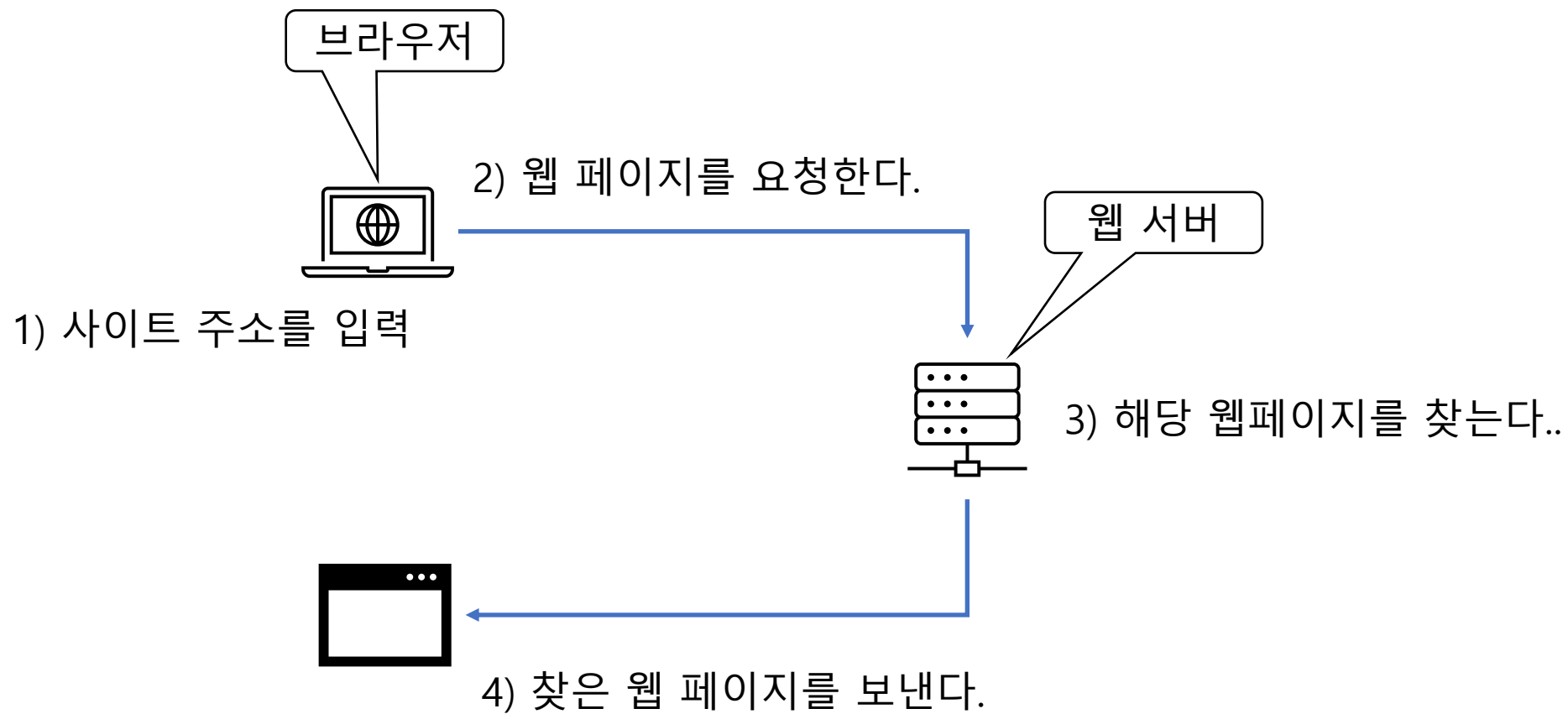


01강 웹개발 개요

웹 프로그래밍이란



클라이언트 측

서버 측

웹 프로그래밍이란

- 기본적인 인터넷 동작원리
 - [클라이언트] 사이트 주소를 입력하고 웹 페이지 요청
 - [서버] 요청받은 웹페이지를 검색
 - [서버] 검색된 웹페이지를 응답
 - [클라이언트] 웹페이지를 본다.
-
- 요청과 응답은 HTTP라는 약속된 절차를 통해서 진행된다.

웹 프로그래밍이란

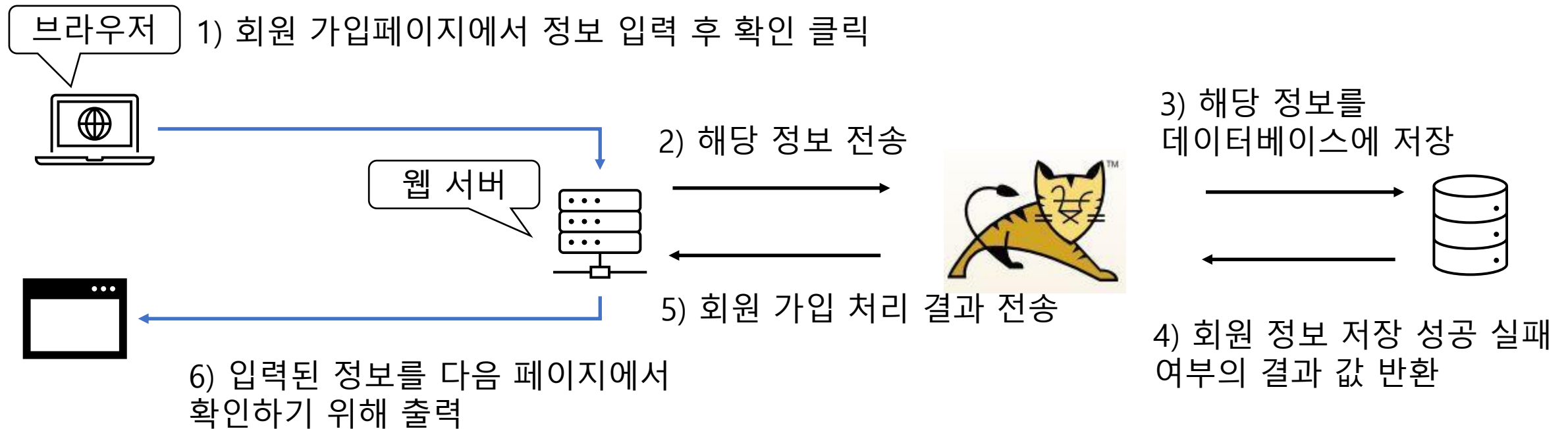
- 이때 요청하고 응답하는 페이지는 HTML문서로 이루어진 웹 문서이다.
- HTML문서로 이루어진 문서는 정보만 보여주는 정적인 페이지이다.
- 대표적인 예시 : 위키 백과

웹 프로그래밍이란

- 그러나 우리가 사용하는 인터넷은 새로운 정보를 제공해야 하는데 정적 페이지인 HTML 문서만으로는 이런 역할을 하기에는 불충분하다.
- 이런 단점을 보완하기 위해서 나온 것이 동적 웹페이지이다.
- 다양하고 새로운 정보를 제공하기 위해서는 방대한 양의 정보가 필요한데 이를 위한 정보 저장소 => 데이터 베이스가 필요하게 되고 이런 정보를 가져와서 '동적'으로 보여 줄 기술이 필요하다
- 이런 의미로 등장한 언어가 PHP, ASP, 서블릿/JSP이다.

웹 프로그래밍이란

- 이런 언어로 개발된 애플리케이션이 웹 애플리케이션이다.



클라이언트 측

서버측

웹 프로그래밍이란

- 서버는 클라이언트의 요청을 처리해서 결과를 보여 주기 위한 웹서버(WS)
- 실질적으로 페이지 로직이나 데이터베이스 연동을 위한 비즈니스 로직이 구현된 웹 애플리케이션 서버(WAS)로 구성된다.
- 웹서버의 종류 : Apache, Nginx, IIS 등
- 웹 애플리케이션 서버의 종류: WebLogic, WebSphere, iPlanet, 9iAS, Jboss, Tomcat 등
- 우리는 이중에서 Apache-Tomcat으로 학습하기로 한다.

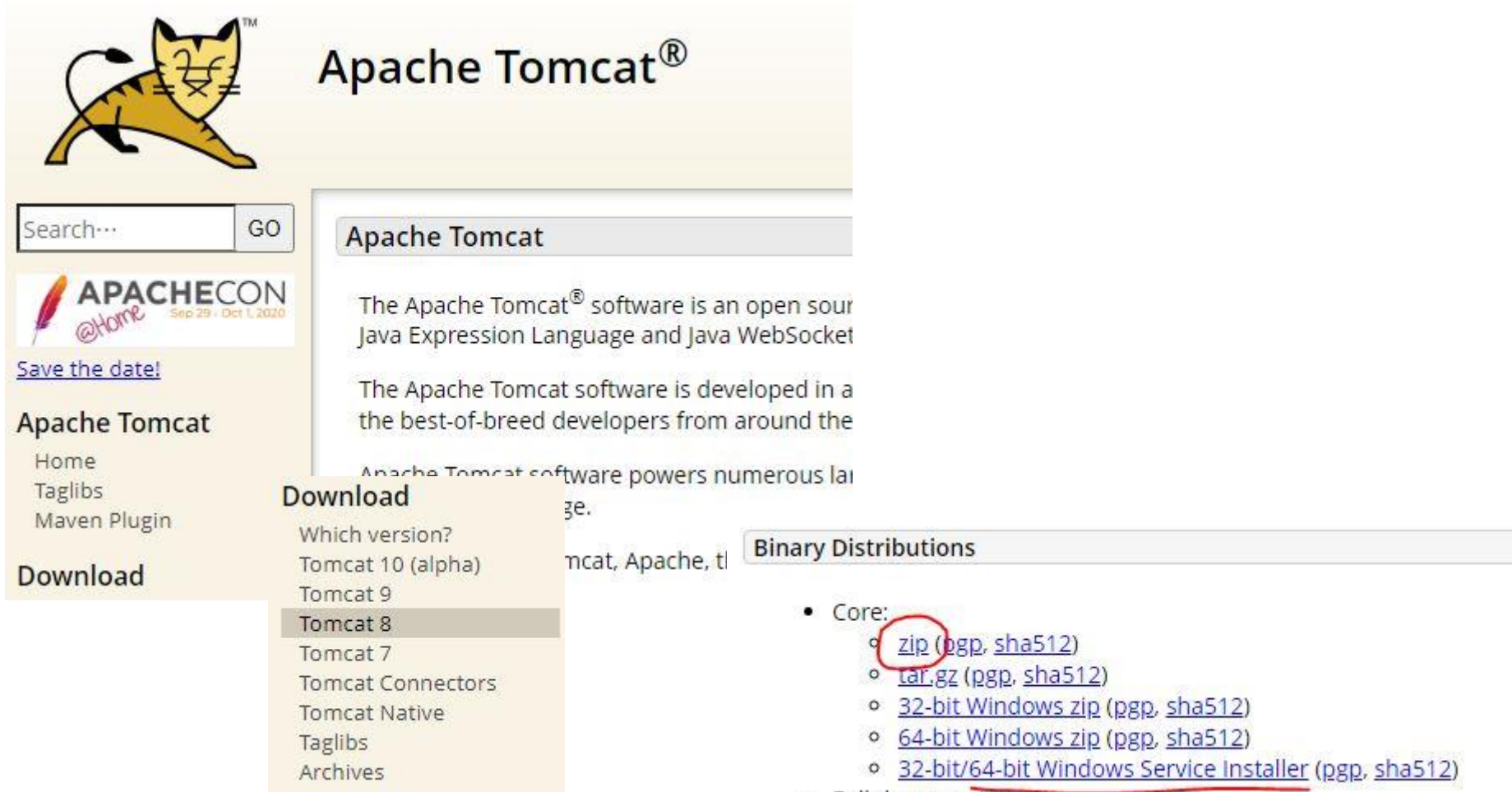
웹 애플리케이션 개발 환경 구축하기

• 환경 구축하기

설치 프로그램	설명	버전
JDK	자바 개발도구	1.8 이후
Eclipse	종합 개발 도구(IDE)	2020-03 이후
Apache Tomcat	웹서버+웹 애플리케이션서버	8.0 이후

웹 애플리케이션 개발 환경 구축하기

- <http://tomcat.apache.org/>



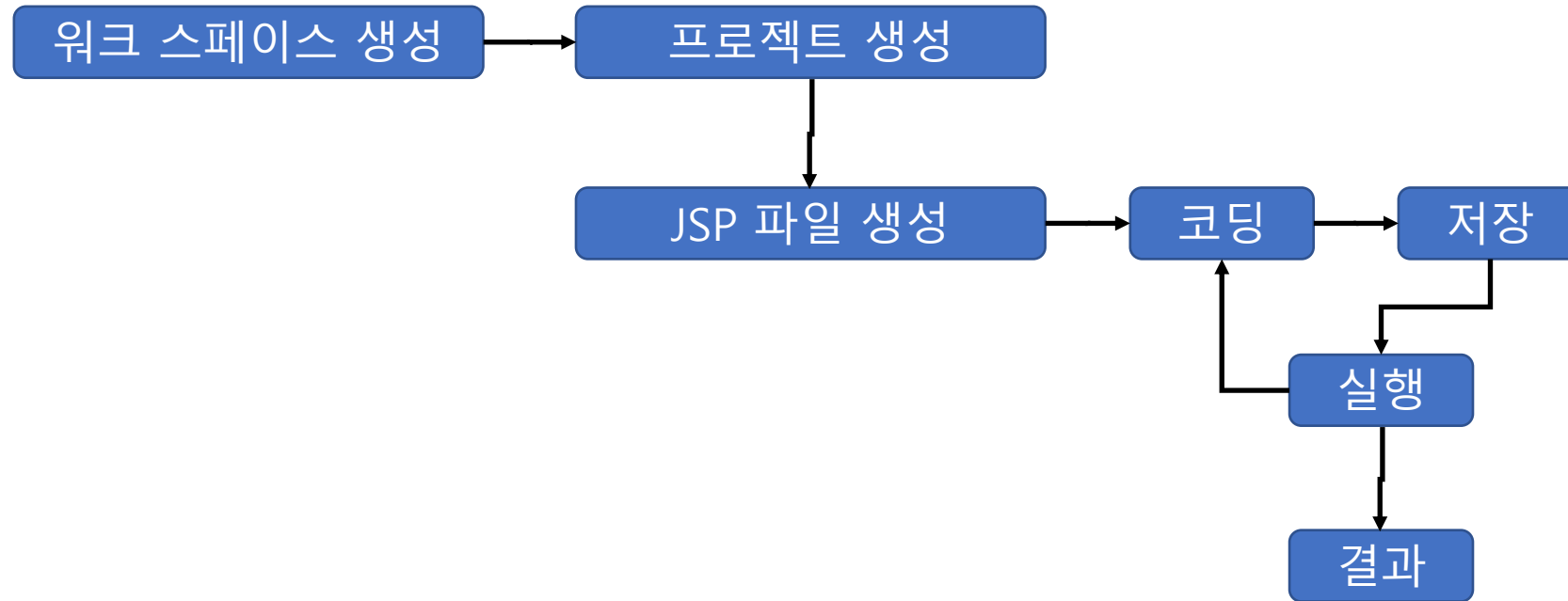
The screenshot shows the Apache Tomcat website. At the top left is the Tomcat logo (a yellow cat). To its right is the text "Apache Tomcat®". Below the logo is a search bar with the text "Search..." and a "GO" button. To the left of the main content area is a sidebar with the "APACHECON @Home" logo (dated Sep 29 - Oct 1, 2020) and a link "Save the date!". Below this is the "Apache Tomcat" section with links for "Home", "Taglibs", and "Maven Plugin". The "Download" section is highlighted, showing a list of versions: "Which version?", "Tomcat 10 (alpha)", "Tomcat 9", "Tomcat 8" (which is selected and highlighted), "Tomcat 7", "Tomcat Connectors", "Tomcat Native", "Taglibs", and "Archives". To the right of the sidebar, the main content area has a heading "Apache Tomcat" and text describing the software as an open source Java Expression Language and Java WebSocket container. Below this is a section titled "Binary Distributions" which lists the following links:

- Core:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
 - [32-bit Windows zip \(pgp, sha512\)](#)
 - [64-bit Windows zip \(pgp, sha512\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:

웹 애플리케이션 개발 환경 구축하기

- zip파일은 무설치 버전
- Install파일은 설치 버전
- 실제 웹 서비스를 제공하려면 설치 버전을 받는 것이 관리하기 편하다.
- 그러나 우리는 실습을 위해서 이클립스에게 실습용 WAS를 구동을 시킬 예정이므로 무설치 버전을 사용한다.
- 적절한 위치에 압축을 푼다.

이클립스로 첫 웹 애플리케이션 작성하기

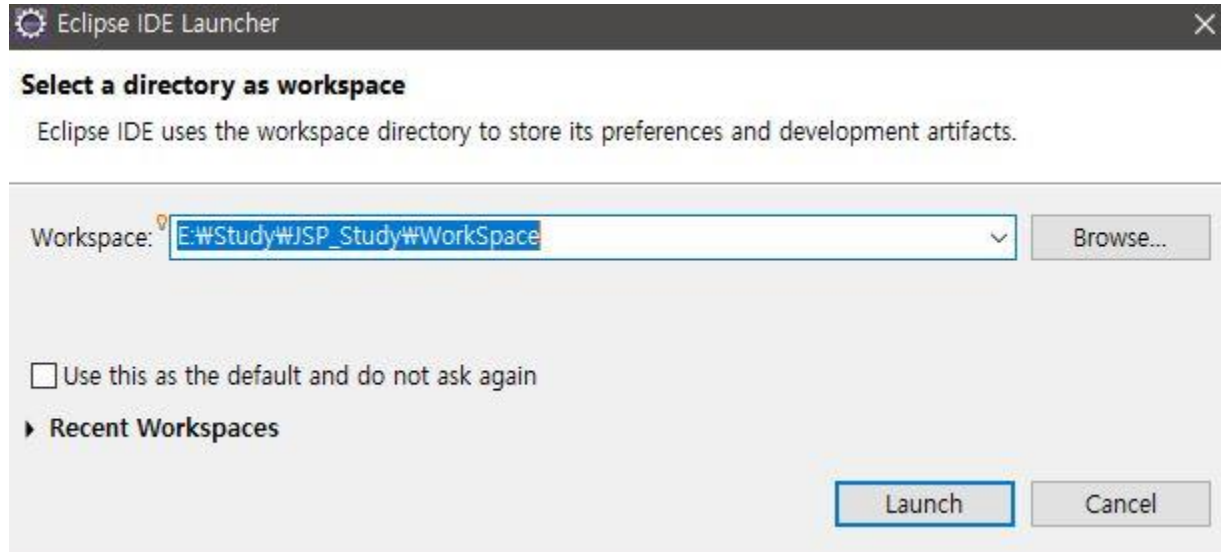


이클립스로 첫 웹 애플리케이션 작성하기

- 프로젝트는 하나의 결과물을 만들어 내기 위한 작업물의 집합체이다.
- 하나의 프로젝트는 작업물 파일들을 담는 폴더를 구성한다.
- 워크스페이스는 서로 다른 프로젝트들을 저장하는 공간(폴더)를 구성한다.
- 이클립스는 워크스페이스 단위로 폴더를 관리한다.

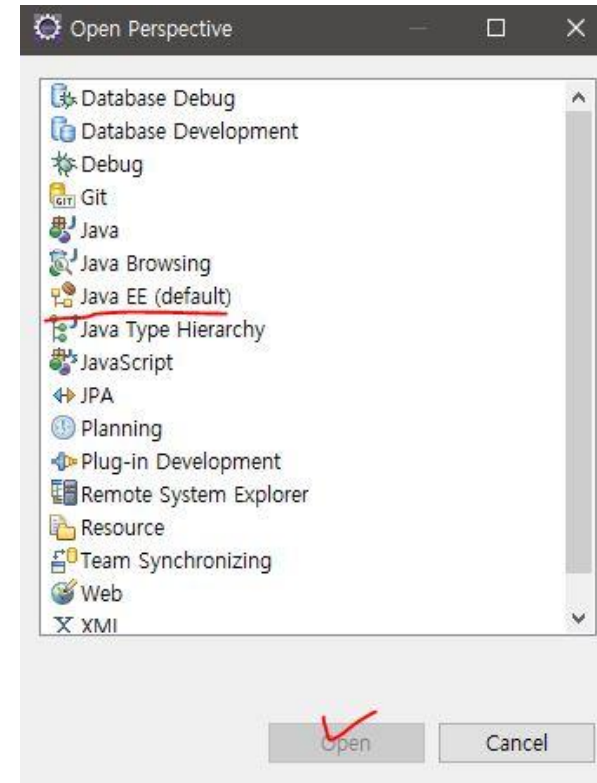
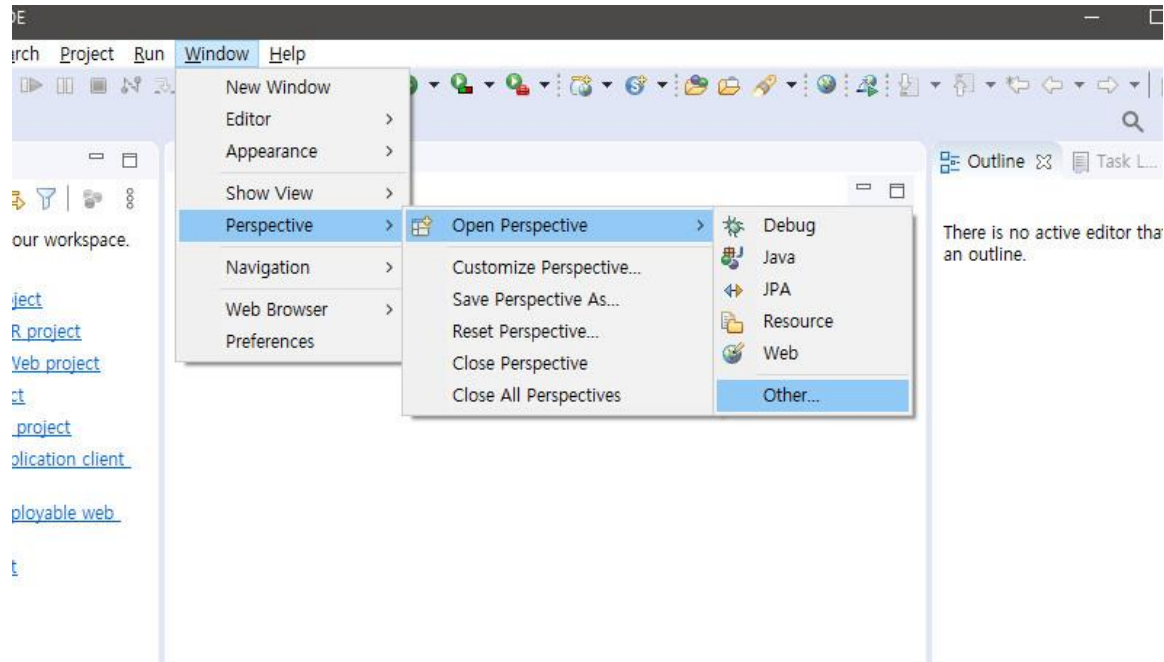
이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스는 워크스페이스 단위로 폴더를 관리한다.



이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스에 톰캣 서버를 등록한다.
 - 혹시 Perspertime가 JAVA EE가 아니라면 바꿔준다
 - windows메뉴->perspective탭 -> open perspertime-> others ->JAVA EE

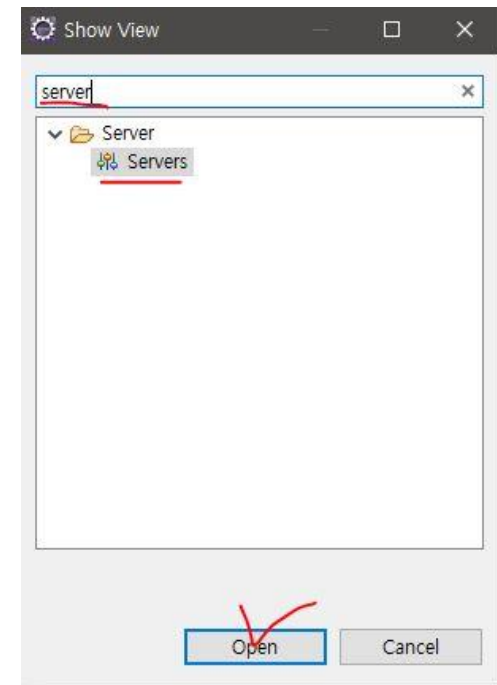
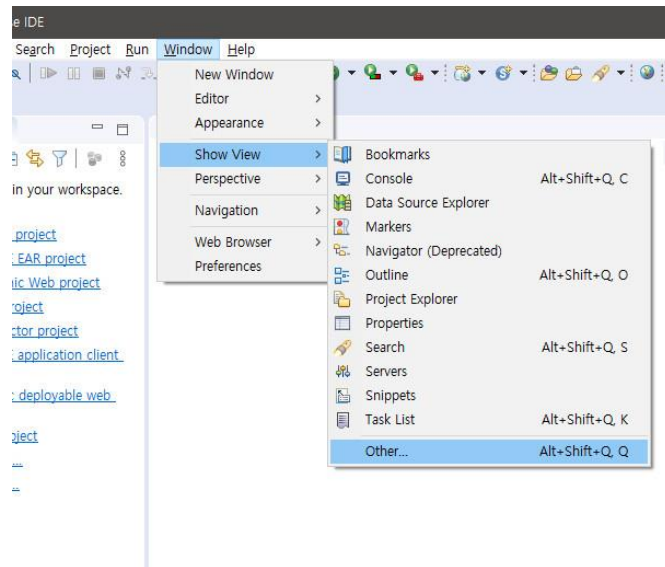


이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스에 톰캣 서버를 등록한다.
 - 하단에 Server탭을 클릭한다.

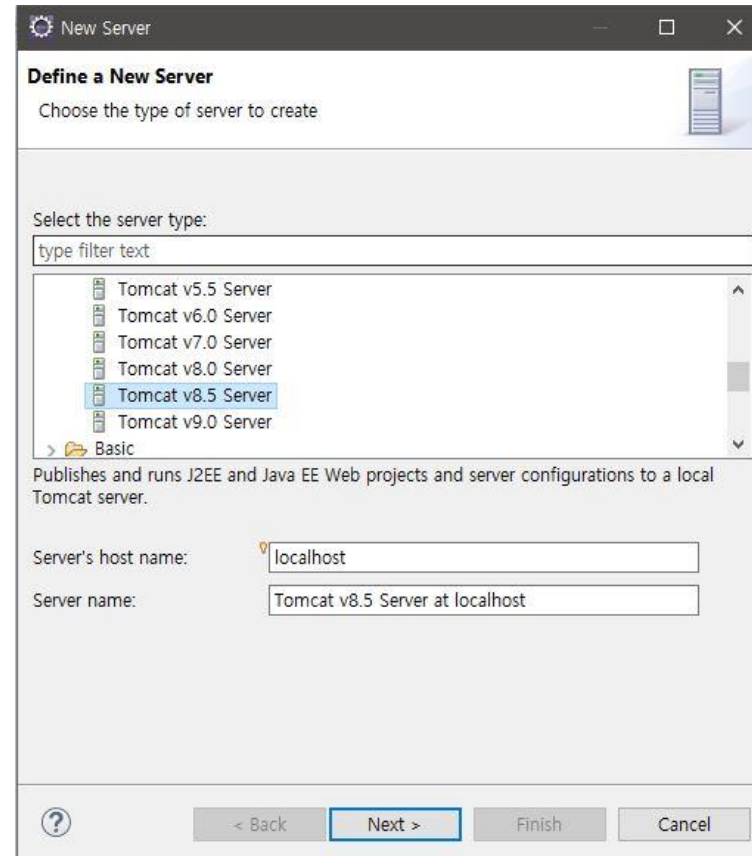


- server탭이 없다면
 - windows메뉴->show view-> others ->server검색



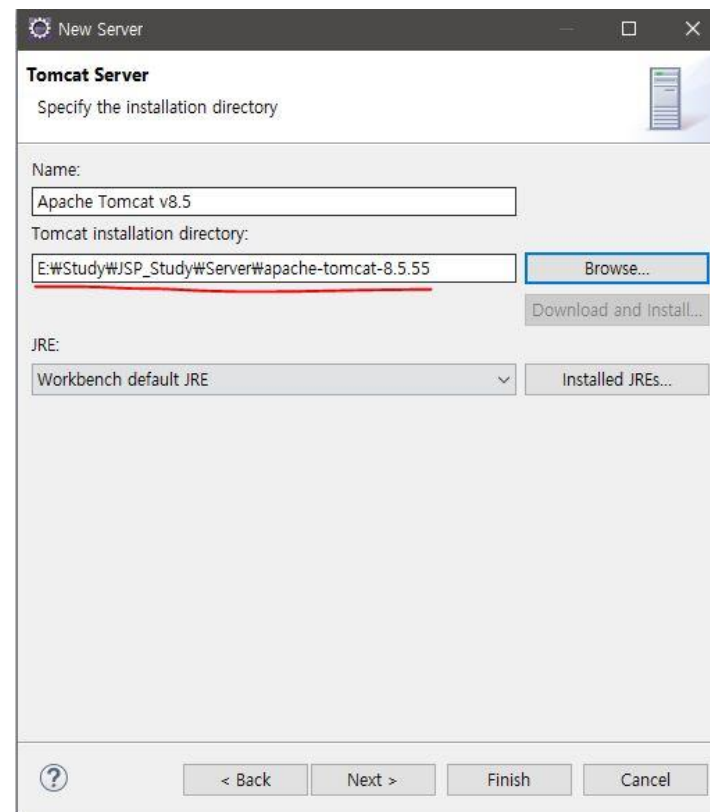
이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스에 톰캣 서버를 등록한다.
 - 위 No servers~~~의 파란 색 링크를 클릭한다.
 - 목록에서 Tomcat8.5를 선택후 Next클릭



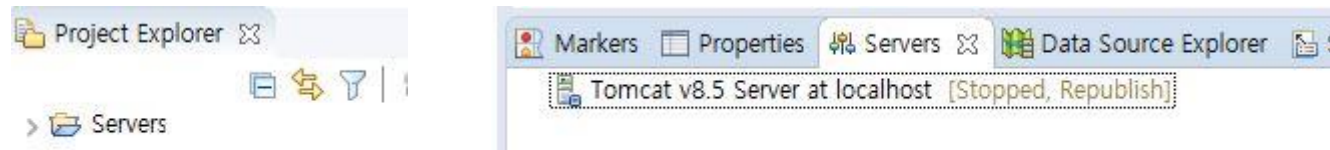
이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스에 톰캣 서버를 등록한다.
 - 붉은 색 위치에 톰캣 압축 푼 위치를 알려준다.
- 기존의 프로젝트가 있다면 next
- 기존의 프로젝트가 없다면 Finish



이클립스로 첫 웹 애플리케이션 작성하기

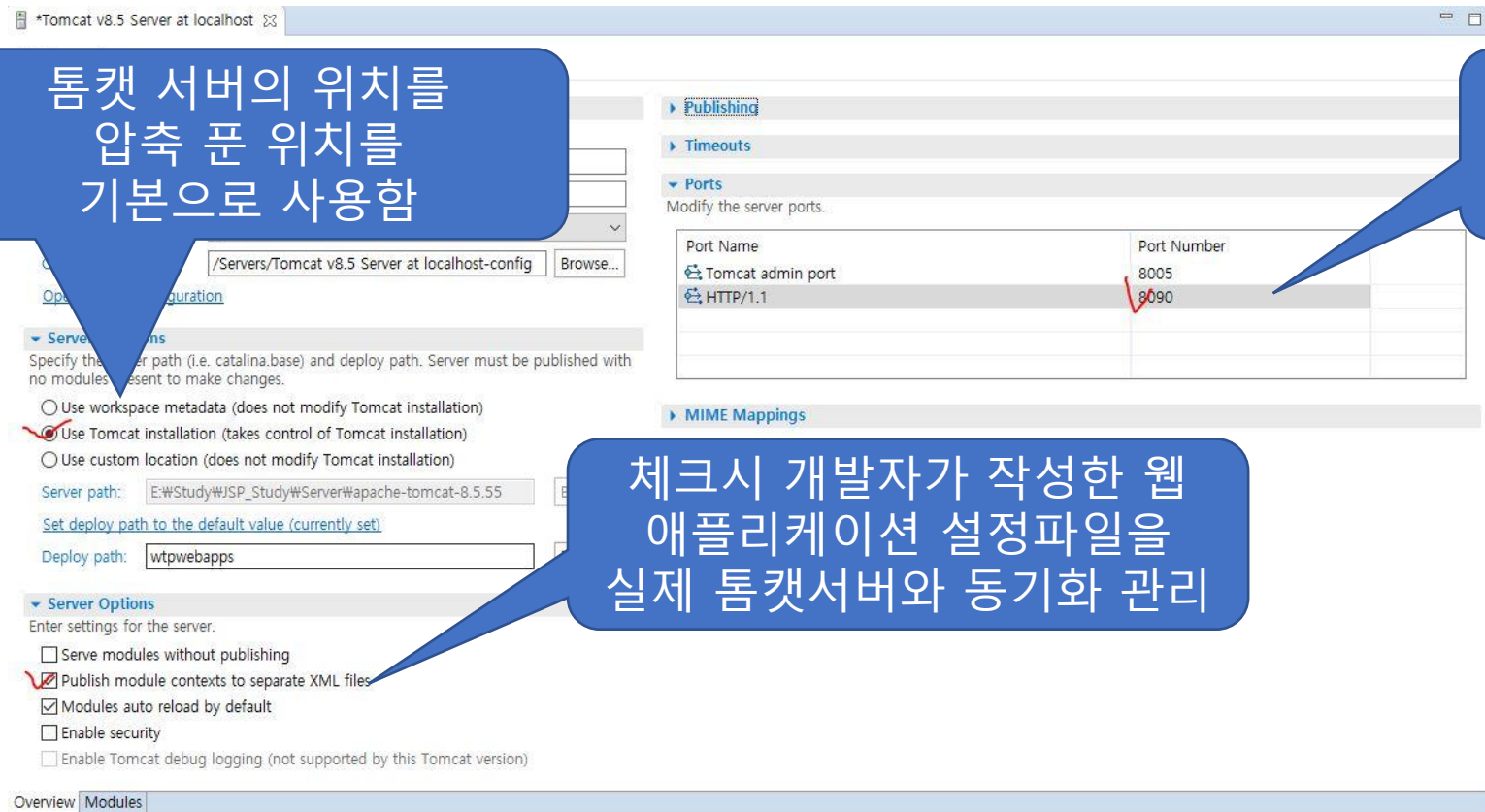
- 이클립스에 톰캣 서버를 등록한다.
- 서버등록이 완료되면 다음과 같은 화면이 나오고 project explorer에 Servers프로젝트가 등록된 것을 볼 수 있다.



- Tomcat v8.5 Server....부분을 더블 클릭하면 톰캣 설정 창이 나온다.

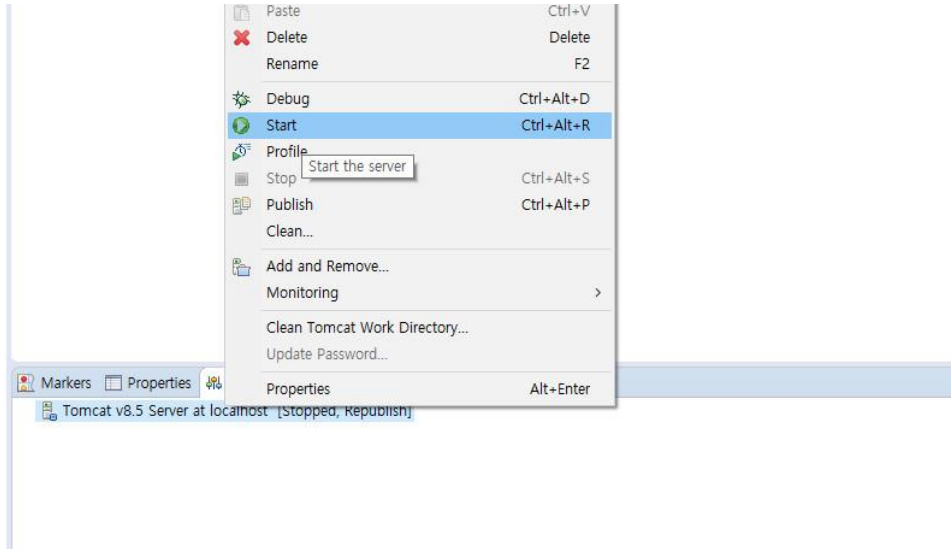
이클립스로 첫 웹 애플리케이션 작성하기

- Tomcat v8.5 Server....부분을 더블 클릭하면 톰캣 설정 창이 나온다.

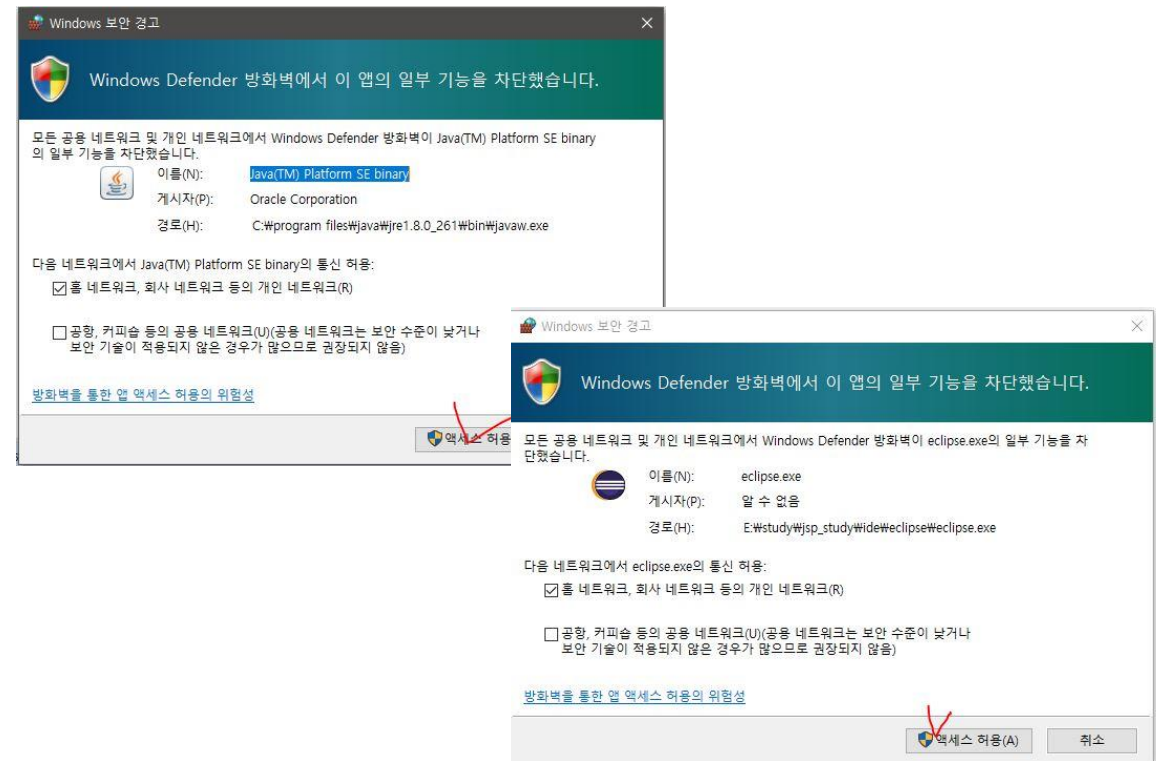


이클립스로 첫 웹 애플리케이션 작성하기

- 이클립스에서 톰캣 서버를 구동해보자

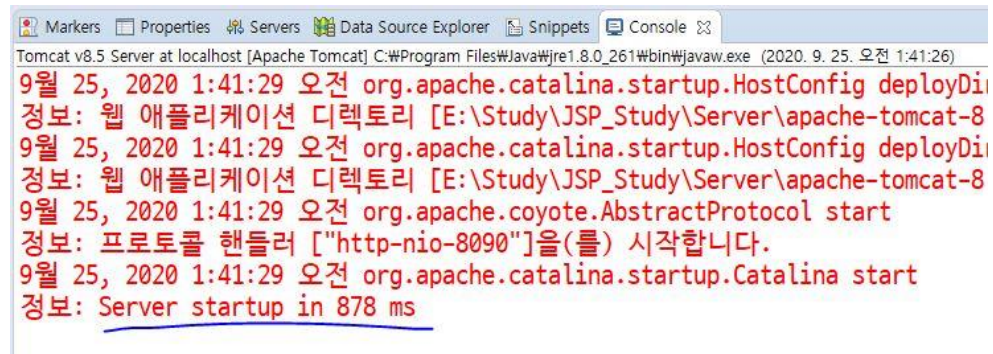


- 보안 경고가 나오면 액세스 허용을 클릭



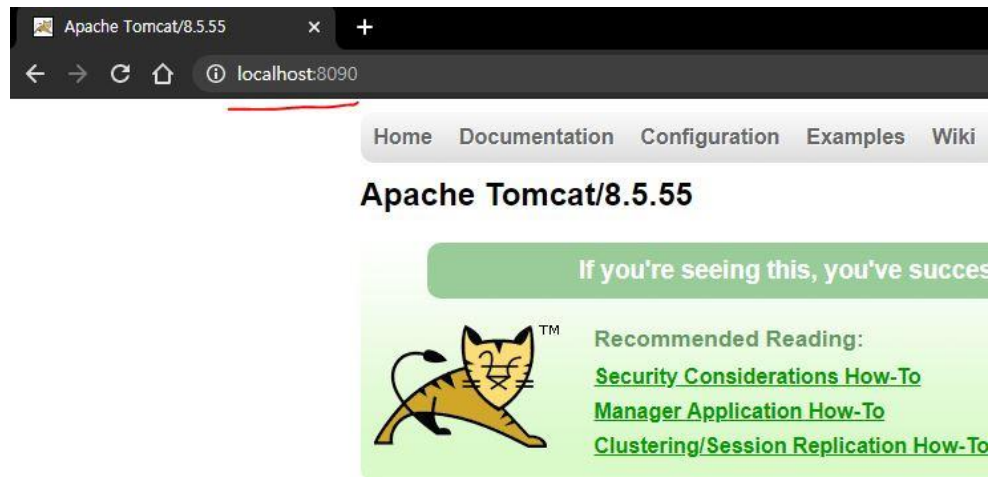
이클립스로 첫 웹 애플리케이션 작성하기

- 콘솔 탭에서 다음과 같은 화면이 나오면 톰캣 구동이 성공한 것이다.



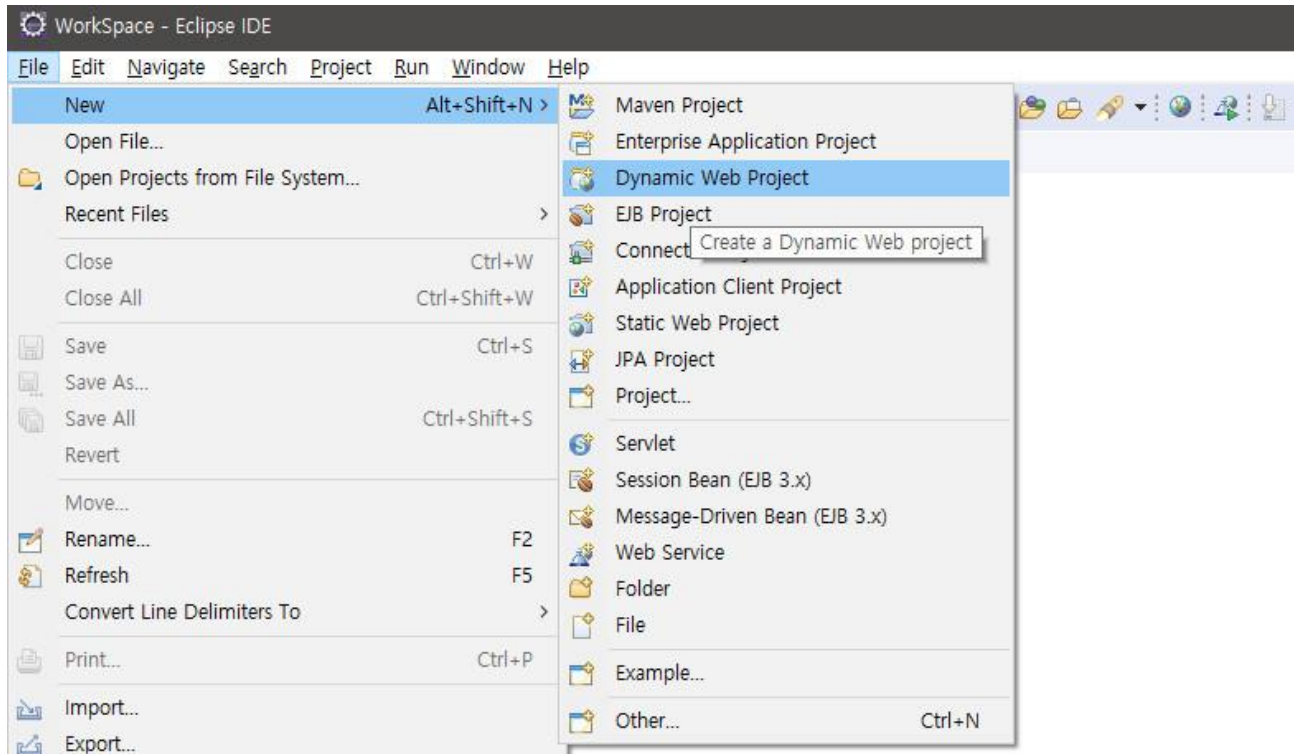
```
Markers Properties Servers Data Source Explorer Snippets Console
Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (2020. 9. 25. 오전 1:41:26)
9월 25, 2020 1:41:29 오전 org.apache.catalina.startup.HostConfig deployDir
정보: 웹 애플리케이션 디렉토리 [E:\Study\JSP_Study\Server\apache-tomcat-8
9월 25, 2020 1:41:29 오전 org.apache.catalina.startup.HostConfig deployDir
정보: 웹 애플리케이션 디렉토리 [E:\Study\JSP_Study\Server\apache-tomcat-8
9월 25, 2020 1:41:29 오전 org.apache.coyote.AbstractProtocol start
정보: 프로토콜 핸들러 ["http-nio-8090"]을(를) 시작합니다.
9월 25, 2020 1:41:29 오전 org.apache.catalina.startup.Catalina start
정보: Server startup in 878 ms
```

- 웹 브라우저로 접속해보자 - localhost:8090 (= > 설정해둔 포트번호)



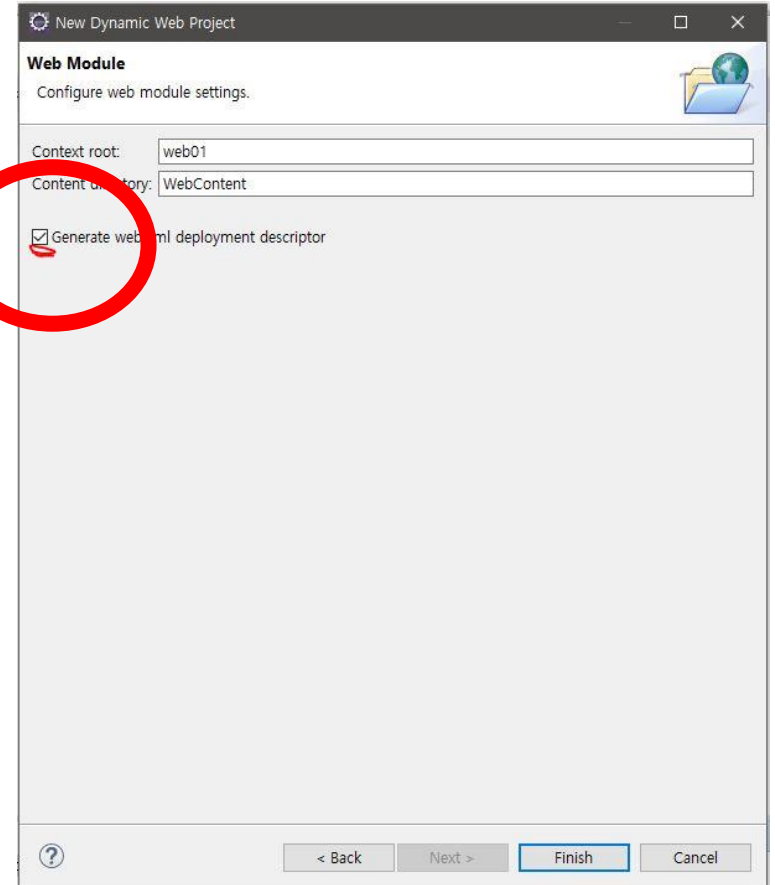
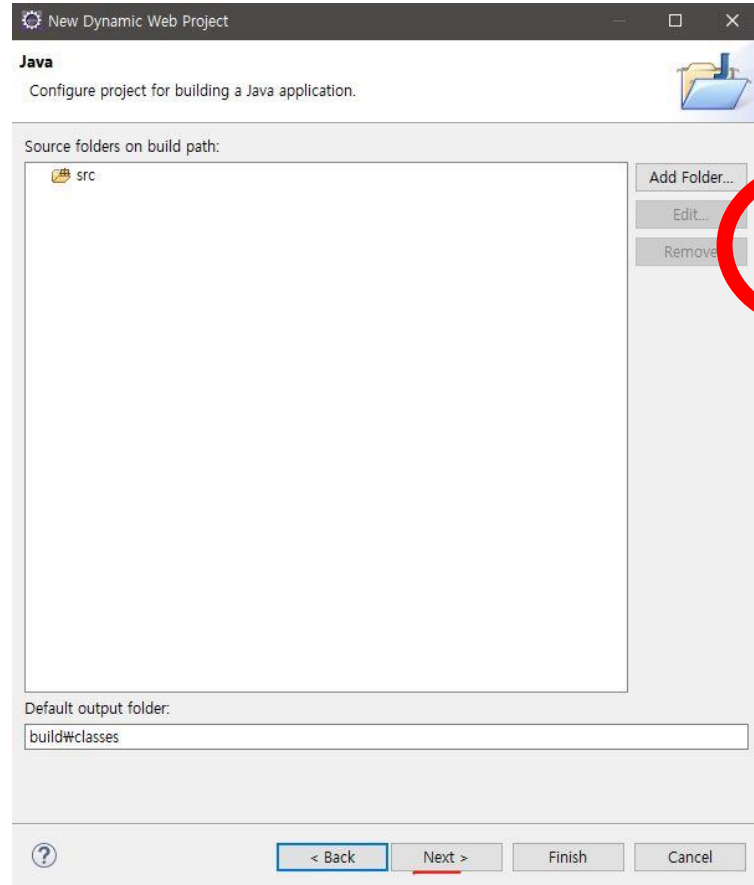
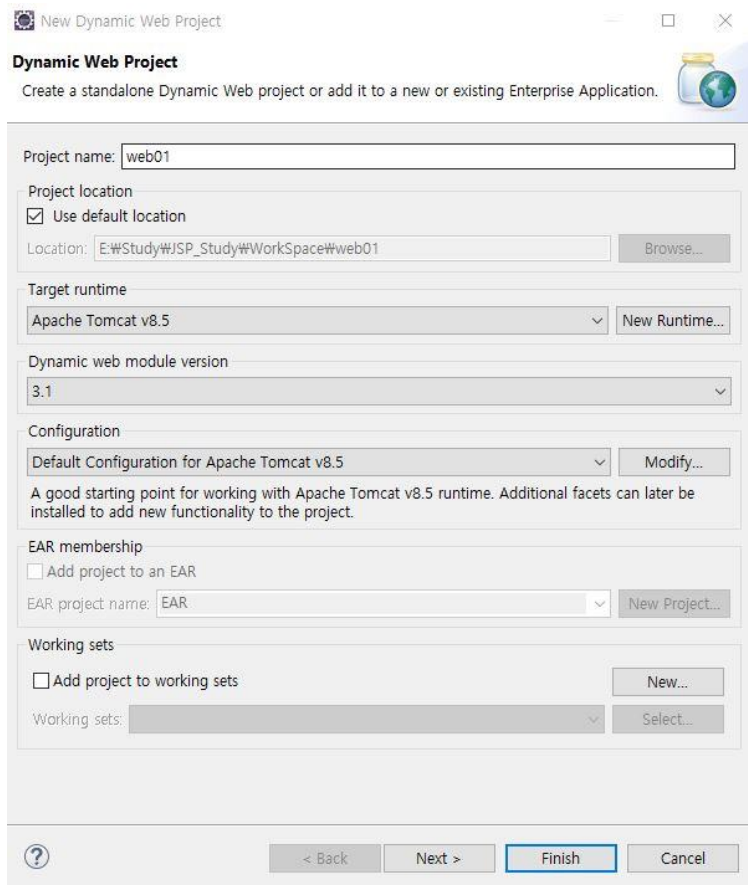
이클립스로 첫 웹 애플리케이션 작성하기

- 이제 첫번째 웹 프로그램을 제작해 봅니다.
- 메뉴에서 file -> new -> Dynamic Web Project를 선택합니다



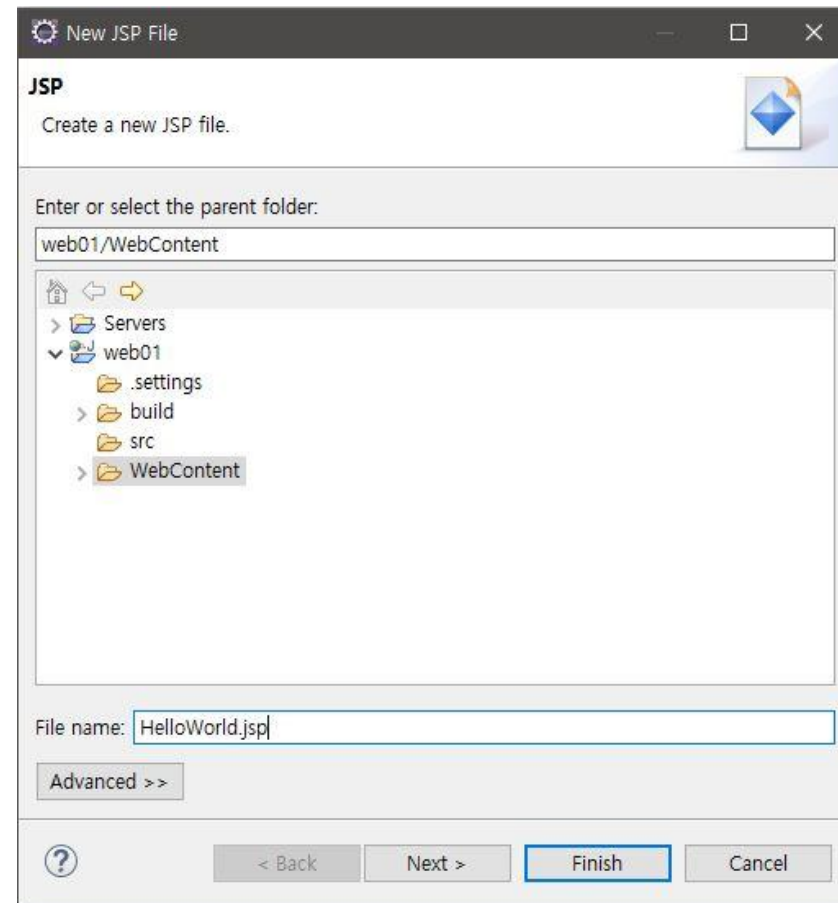
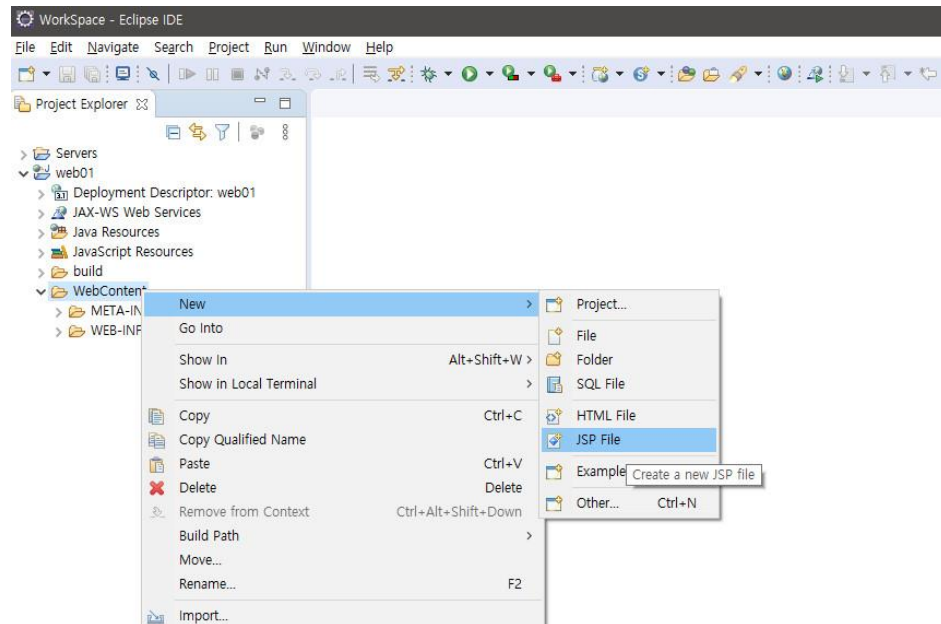
이클립스로 첫 웹 애플리케이션 작성하기

- 프로젝트 이름은 web01로 하고 web.xml 만들어 줍니다.



이클립스로 첫 웹 애플리케이션 작성하기

- WebContent에서 JSP 파일을 생성합니다. (파일명 HelloWorld.jsp)



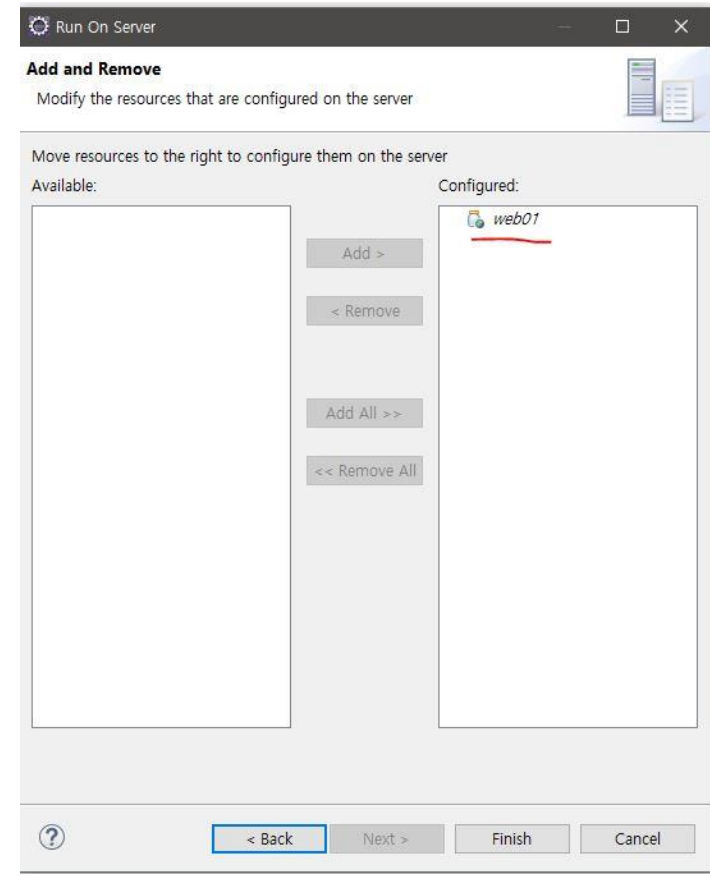
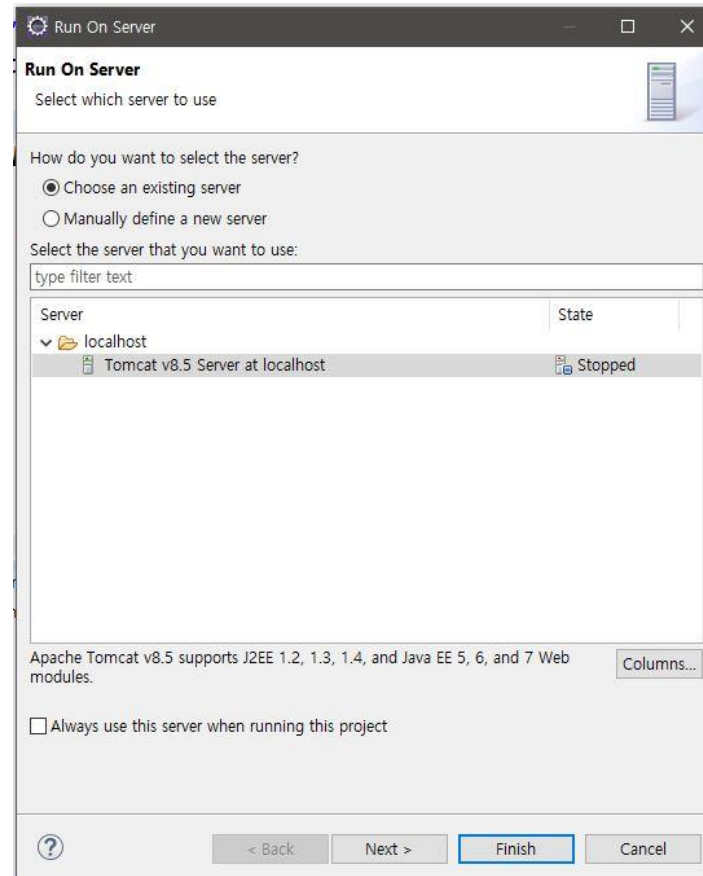
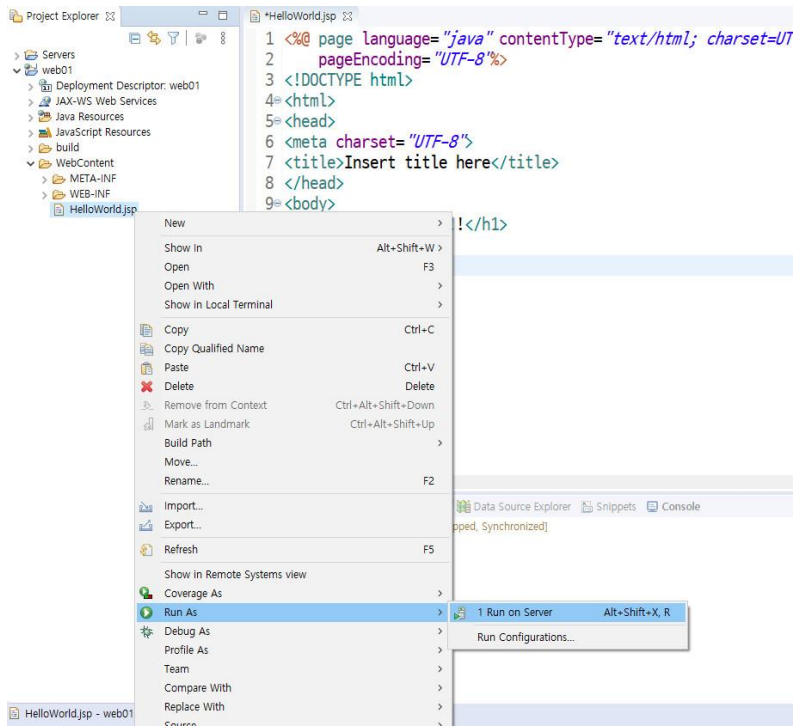
이클립스로 첫 웹 애플리케이션 작성하기

- 생성된 JSP 파일 내부에 다음과 같은 코드를 작성해본다.(저장한다)

```
*HelloWorld.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 </head>
9 <body>
10   <h1>Hello World!!</h1>
11 </body>
12 </html>
```

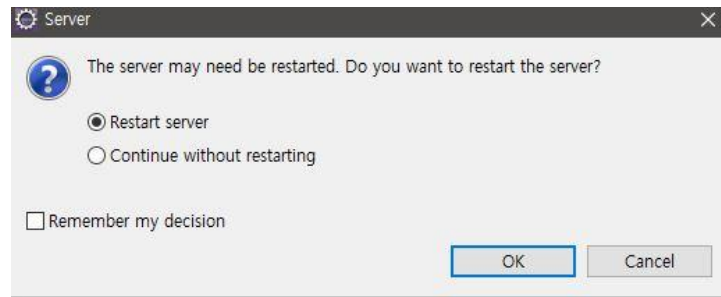
이클립스로 첫 웹 애플리케이션 작성하기

- 생성된 파일을 우클릭해서 서버 실행을 하면 웹 페이지가 정상적으로 나온다.

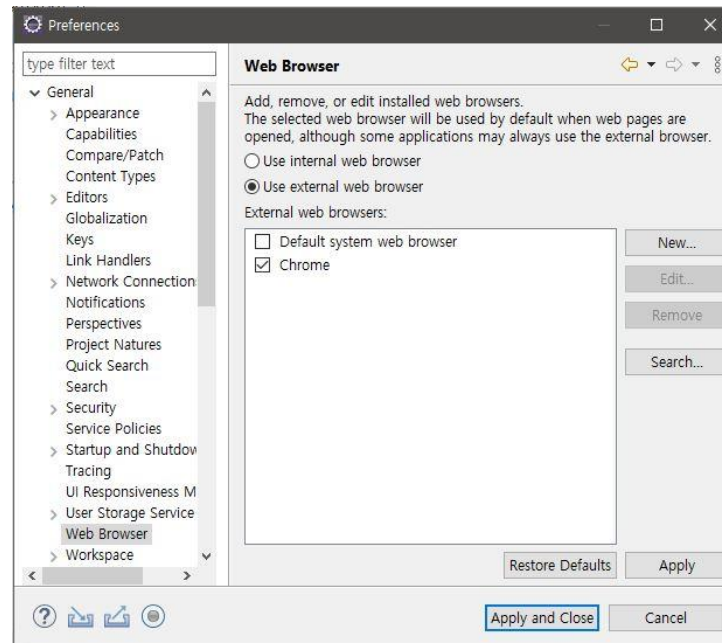


이클립스로 첫 웹 애플리케이션 작성하기

- 이미 서버가 실행중이라면 서버 재시작 메시지가 나오는데 그냥 재시작하면 된다.



- 또한 웹 페이지가 이클립스에서 나온다면 설정을 통해 바꿔 줄 수 있다
메뉴의 window->Preferences
General->Web browser에서
변경가능

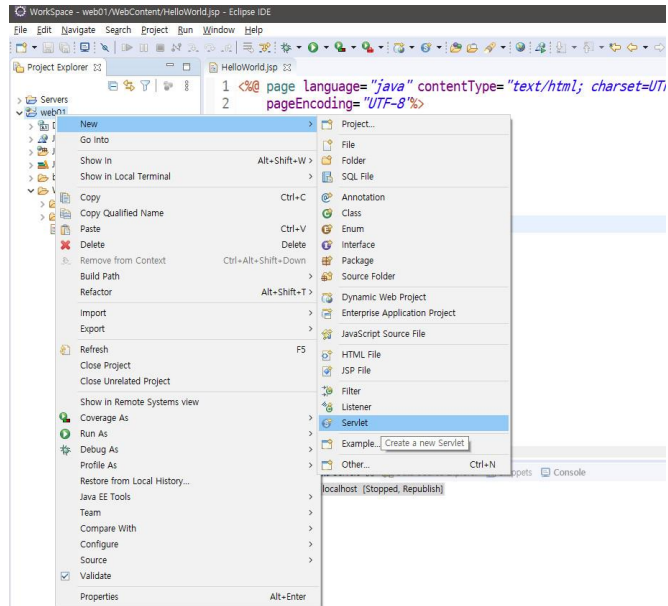


서블릿과 JSP의 기초 개념

- 서블릿(Servlet)이란 Server+Applet의 합성어로 서버에서 실행되는 Applet란 의미로 웹에서 실행하는 프로그램을 작성하는 기술을 의미한다.
- 보통 서블릿은 자바 클래스 형태의 웹 애플리케이션이다
- 이것을 브라우저를 통해서 실행되도록 하기 위해서는 javax.servlet.http패키지에서 제공하는 httpServlet 클래스를 상속받아서 구현해야 한다.
- 위 클래스를 상속받은 서브 클래스를 서블릿클래스라고 한다.
- JDK에는 웹 애플리케이션을 제작할 수 있는 클래스를 제공하지 않고 톰캣을 설치하면 해당 클래스를 제공한다.

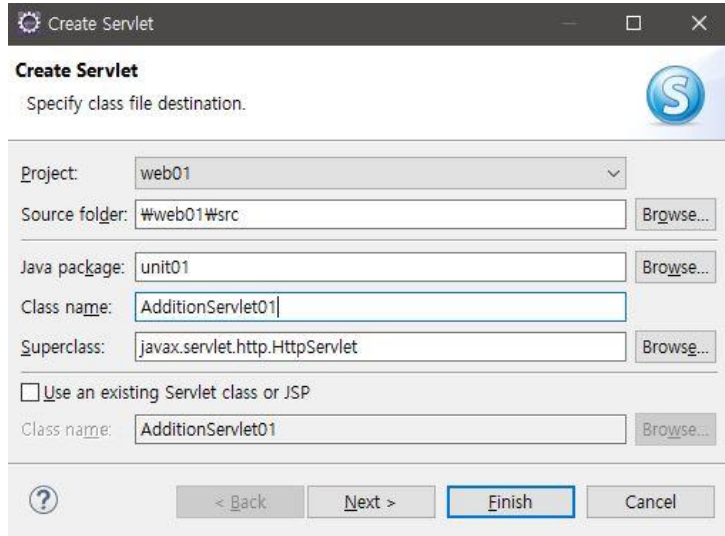
서블릿과 JSP의 기초 개념

- 서블릿 프로그램을 만들어 본다.
- 프로젝트에서 우클릭 -> new -> servlet을 선택한다.



서블릿과 JSP의 기초 개념

- 나오는 화면에서 패키지는 unit01, 서블릿 클래스 이름은 AdditionServlet01로 입력후 finish를 클릭.



- Java Resources->src->unit01->AdditionServlet01.java가 생성된것을 볼수 있다

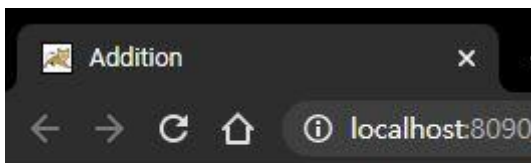
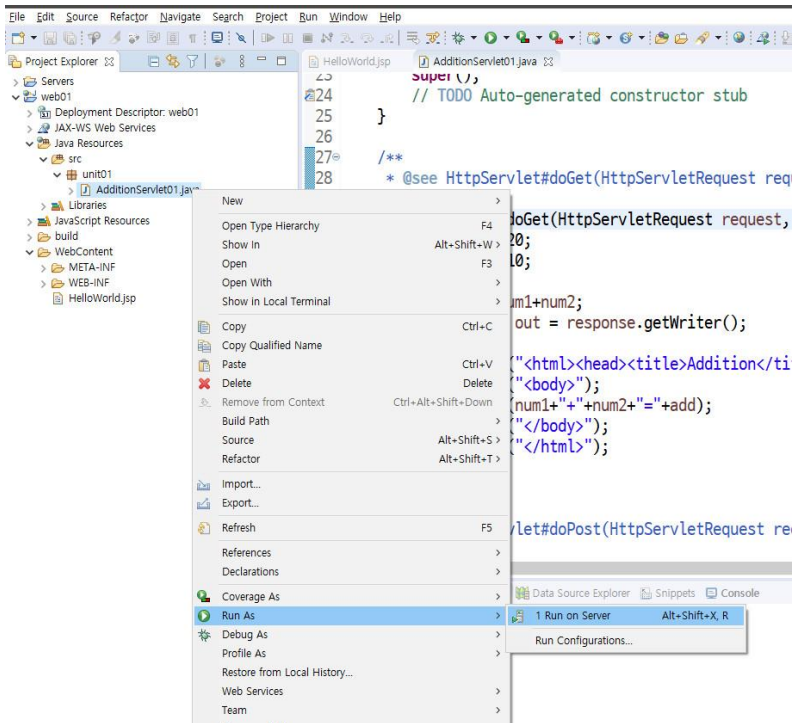
서블릿과 JSP의 기초 개념

- 해당 파일에서 doGet 메소드에 다음과 같은 코드를 작성한다.
두수의 합을 구하는 서블릿 코드

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    int num1 = 20;  
    int num2 = 10;  
  
    int add = num1+num2;  
    PrintWriter out = response.getWriter();  
  
    out.println("<html><head><title>Addition</title></head>");  
    out.println("<body>");  
    out.println(num1+"+"+num2+"="+add);  
    out.println("</body>");  
    out.println("</html>");  
}
```

서블릿과 JSP의 기초 개념

- 해당 서블릿 파일을 선택 후 우클릭으로 서버 실행을 해본다



20+10=30

서블릿과 JSP의 기초 개념

- 서블릿의 응답 방식을 보면 doGet과 doPost방식이 있는데 이것은 클라이언트의 요청방식에 대응한다.

전송 방식	설명
Get방식	데이터를 주소창으로 넘어간다. 보안에 취약하고 255이하의 적은 양의 데이터를 전송
Post방식	Html header에 데이터를 담아서 넘긴다. 255자 이상 대용량 데이터를 전송한다.

(아무런 선택이 없으면 기본값으로 get방식 요청을 하게 된다)

- 서블릿 입장에서는 Post로 전송되나 Get방식으로 전송되나 처리방식은 같다.
- 단지 doGet이냐 doPost에서 처리하느냐의 차이이다.
- 요청정보는 HttpServletRequest객체에 담아서 전달된다.
- 그리고 응답정보는 HttpServletResponse 객체를 사용한다.

서블릿과 JSP의 기초 개념

- HttpServletResponse 객체를 이용해서 HTML문서로 응답하기 위해서는 출력 객체를 사용해야 한다.
- PrintWriter 출력 스트림 객체로 HTML 문서를 출력한다.

서블릿과 JSP의 기초 개념

- JSP는 JAVA SERVER PAGE의 준말로 자바로 웹 페이지를 작성하기 위한 언어이다.
- 서버 페이지를 실행되는 로직을 구현하기 위해 프로그래밍언어가 필요한데 JSP는 그중에서 Java를 사용하고 있다.

두수의 합을 구하는 JSP 예제

```
<body>
  <%
    int num1 = 20;
    int num2 = 10;
    int add = num1 + num2;
  %>

  <%=num1%>+<%=num2%>=<%=add%>
</body>
```

서블릿과 JSP의 기초 개념

- 서블릿과 JSP의 차이를 보면
- 서블릿은 자바 코드 내부에 HTML코드가 들어가는 구조이지만
- JSP는 HTML코드 내부에 자바 코드를 삽입하는 형태이다.

서블릿과 JSP의 기초 개념

- 서블릿 보다 JSP만 사용하는 것이 적합 할 것 같지만 복잡한 로직을 구현하는데 JSP에 작성하면 프론트 엔드 개발자가 실수로 코드를 건드려 문제가 발생하는 경우가 있으므로 적절한 조화가 필요하다.

두 수의 합을 구하는 예제(서블릿 : AdditionServlet03.java , JSP : addition03.jsp)

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    int num1 = 20;  
    int num2 = 10;  
    int add = num1+num2;  
  
    request.setAttribute("num1",num1);  
    request.setAttribute("num2",num2);  
    request.setAttribute("add",add);  
  
    RequestDispatcher dispatcher=  
        request.getRequestDispatcher("addition03.jsp");  
    dispatcher.forward(request, response);  
}
```

```
<title>Addition03</title>  
</head>  
<body>  
    ${num1}+${num2}=${add}  
</body>
```

위와 같이 표현 부분과 로직 부분을 분할 하는 코드 방식을 MVC라고 한다.