



서블릿 이해하기

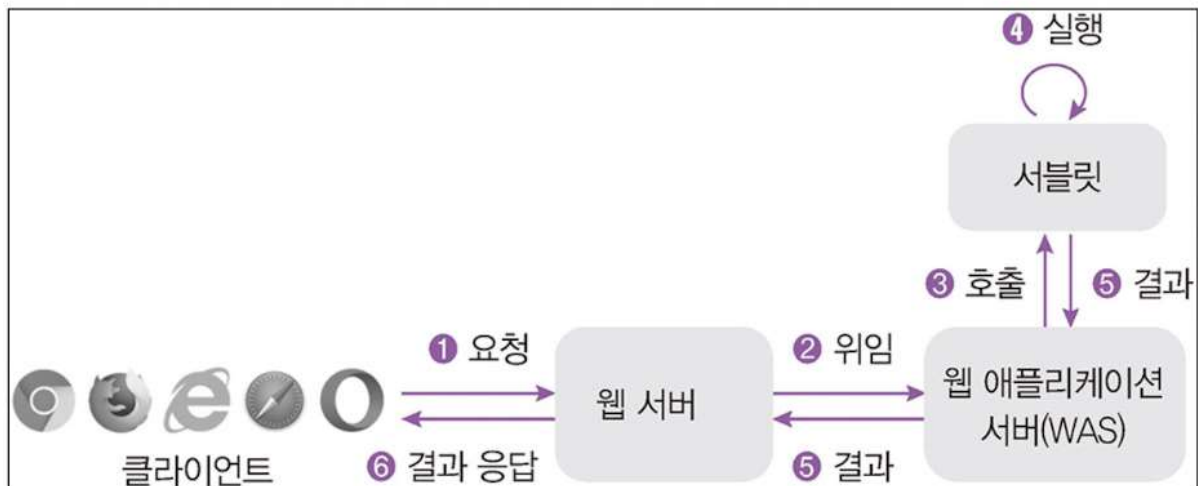
서블릿이란?

서블릿은 서버쪽에서 실행되면서 클라이언트의 요청에 따라 동적으로 서비스를 제공하는 자바 클래스입니다.

서블릿은 자바로 작성되어 있으므로 자바의 일반적인 특징을 모두 가집니다.

서블릿은 일반 자바 프로그램과 다르게 독자적으로 실행되지 못하고 **톰캣과 같은 JSP/Servlet 컨테이너에서 실행**된다는 점에서 차이가 있습니다.

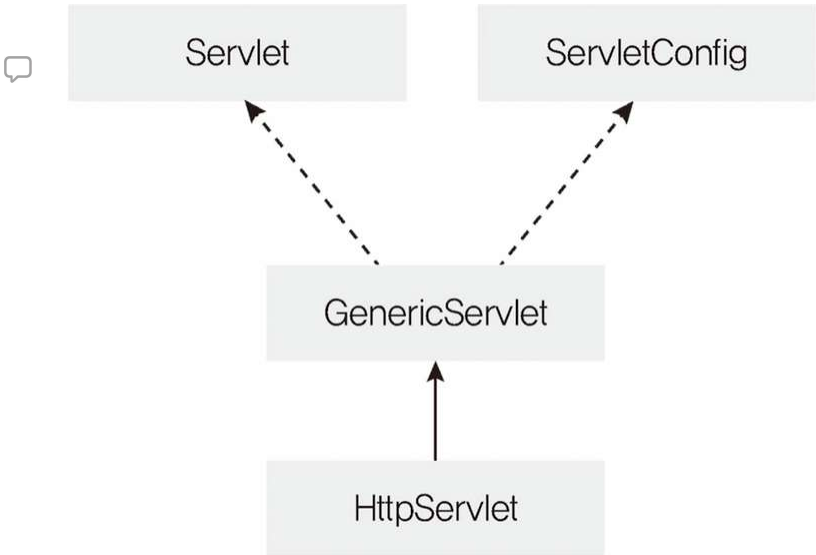
서블릿은 서버에서 실행되다가 웹 브라우저에서 요청을 하면 해당 기능을 수행한 후 웹 브라우저에 결과를 전송 합니다. 서버에서 실행되므로 보안과 관련된 기능도 훨씬 안전하게 수행 할 수 있습니다.



서블릿 API 계층 구조와 기능

서블릿은 자바로 만들어졌으므로 당연히 클래스 간의 계층 구조를 가집니다.

(자바의 인터페이스 및 추상 클래스에 대한 이해가 필요 합니다.)



서블릿 API 기능

서블릿 API 구성 요소 특징

서블릿 구성 요소	기능
Servlet 인터페이스	<ul style="list-style-type: none">• javax.servlet 패키지에 선언되어 있습니다.• Servlet 관련 추상 메서드를 선언합니다.• init(), service(), destroy(), getServletInfo(), getServletConfig()를 선언합니다.
ServletConfig 인터페이스	<ul style="list-style-type: none">• javax.servlet 패키지에 선언되어 있습니다.• Servlet 기능 관련 추상 메서드가 선언되어 있습니다.• getInitParameter(), getInitParameterNames(), getServletContext(), getServletName()이 선언되어 있습니다.
GenericServlet 클래스	<ul style="list-style-type: none">• javax.servlet 패키지에 선언되어 있습니다.• 상위 두 인터페이스를 구현하여 일반적인 서블릿 기능을 구현한 클래스입니다.• GenericServlet을 상속받아 구현한 사용자 서블릿은 사용되는 프로토콜에 따라 각각 service()를 오버라이딩해서 구현합니다.
HttpServlet 클래스	<ul style="list-style-type: none">• javax.servlet.http 패키지에 선언되어 있습니다.• GenericServlet을 상속받아 http 프로토콜을 사용하는 웹 브라우저에서 서블릿 기능을 수행합니다.• 웹 브라우저 기반 서비스를 제공하는 서블릿을 만들 때 상속받아 사용합니다.• 요청 시 service()가 호출되면서 요청 방식에 따라 doGet()이나 doPost()가 차례대로 호출됩니다.

HttpServlet의 여러 가지 메서드 기능

메서드	기능
<code>protected doDelete(HttpServletRequest req, HttpServletResponse resp)</code>	서블릿이 DELETE request를 수행하기 위해 <code>service()</code> 를 통해서 호출됩니다.
<code>protected doGet(HttpServletRequest req, HttpServletResponse resp)</code>	서블릿이 GET request를 수행하기 위해 <code>service()</code> 를 통해서 호출됩니다.
<code>protected doHead(HttpServletRequest req, HttpServletResponse resp)</code>	서블릿이 HEAD request를 수행하기 위해 <code>service()</code> 를 통해서 호출됩니다.
<code>protected doPost(HttpServletRequest req, HttpServletResponse resp)</code>	서블릿이 POST request를 수행하기 위해 <code>service()</code> 를 통해서 호출됩니다.
<code>protected service (ServletRequest req, ServletResponse resp)</code>	request를 <code>public service()</code> 에서 전달받아 <code>doXX()</code> 메서드를 호출합니다.
<code>public service (ServletRequest req, ServletResponse resp)</code>	클라이언트의 request를 <code>protected service()</code> 에게 전달합니다.

서블릿의 생명주기 메소드

서블릿도 자바 클래스이므로 실행하면 당연히 초기화 과정 메모리에 인스턴스를 생성하여 서비스를 수행한 후 다시 소멸하는 과정을 거칩니다.

서블릿 생명주기 메소드란 서블릿이 실행 단계마다 호출되어 기능을 수행하는 콜백 메소드를 의미 합니다.

생명주기 단계	호출 메서드	특징
초기화	<code>init()</code>	<ul style="list-style-type: none"> 서블릿 요청 시 맨 처음 한 번만 호출됩니다. 서블릿 생성 시 초기화 작업을 주로 수행합니다.
작업 수행	<code>doGet()</code> <code>doPost()</code>	<ul style="list-style-type: none"> 서블릿 요청 시 매번 호출됩니다. 실제로 클라이언트가 요청하는 작업을 수행합니다.
종료	<code>destroy()</code>	<ul style="list-style-type: none"> 서블릿이 기능을 수행하고 메모리에서 소멸될 때 호출됩니다. 서블릿의 마무리 작업을 주로 수행합니다.

👉 `do`로 시작하는 메소드는 서블릿의 핵심 기능을 처리하므로 반드시 구현 해야 합니다.

FirstServlet을 이용한 실습



사용자 정의 서블릿 만들기

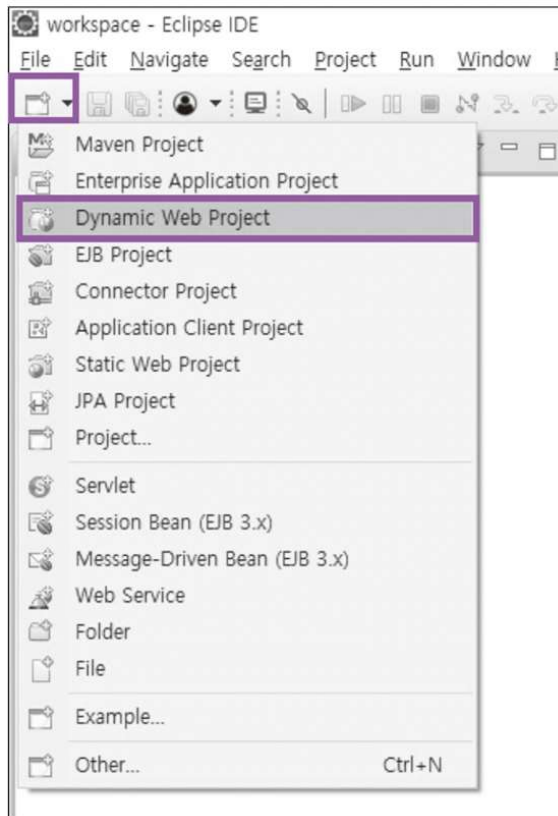
```

public class FirstServlet extends HttpServlet { @Override public void
init() throws ServletException { System.out.println("init 메소드 호출"); }
@Override protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
System.out.println("doGet 메소드 호출"); } @Override public void destroy()
{ System.out.println("destroy 메소드 호출"); } }
  
```

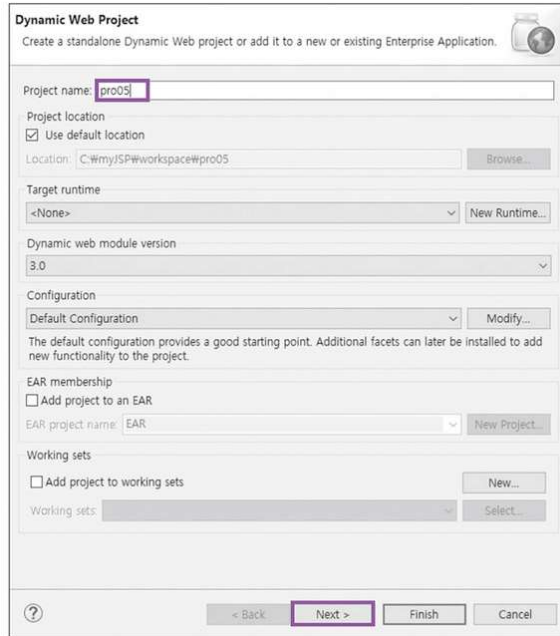
톰캣의 servlet-api.jar 클래스 패스 설정 하기

서블릿 API들은 톰캣의 servlet-api.jar 라이브러리로 제공되므로 이클립스의 프로젝트에서 서블릿을 사용하려면 반드시 클래스를 설정해야 합니다.

1. 이클립스 상단의 New 아이콘을 클릭한 후 Dynamic Web Project 선택



2. 프로젝트 이름을 pro05로 입력 한후 Next를 클릭



Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location
☒ Use default location
Location:

Target runtime
<None>

Dynamic web module version
3.0

Configuration
Default Configuration
The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name:

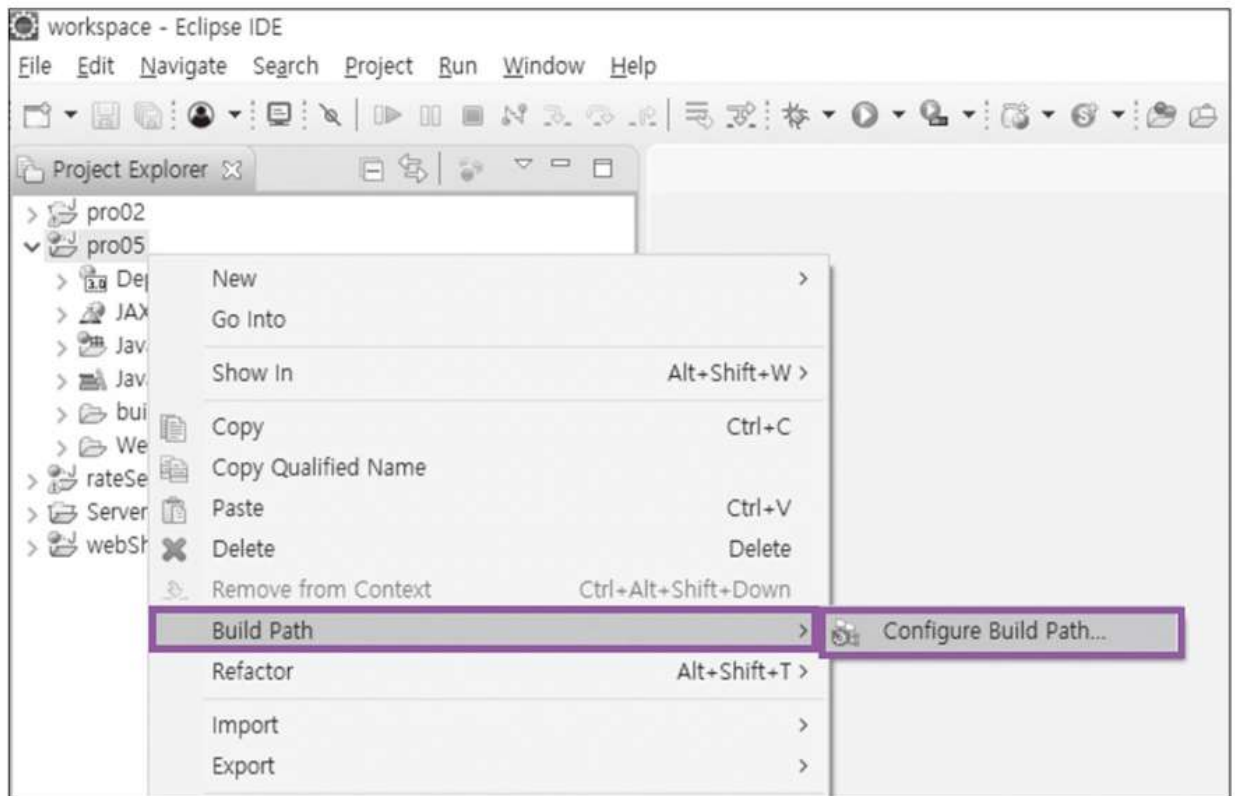
Working sets
☐ Add project to working sets
Working sets:

3. 경로 확인 후 Next

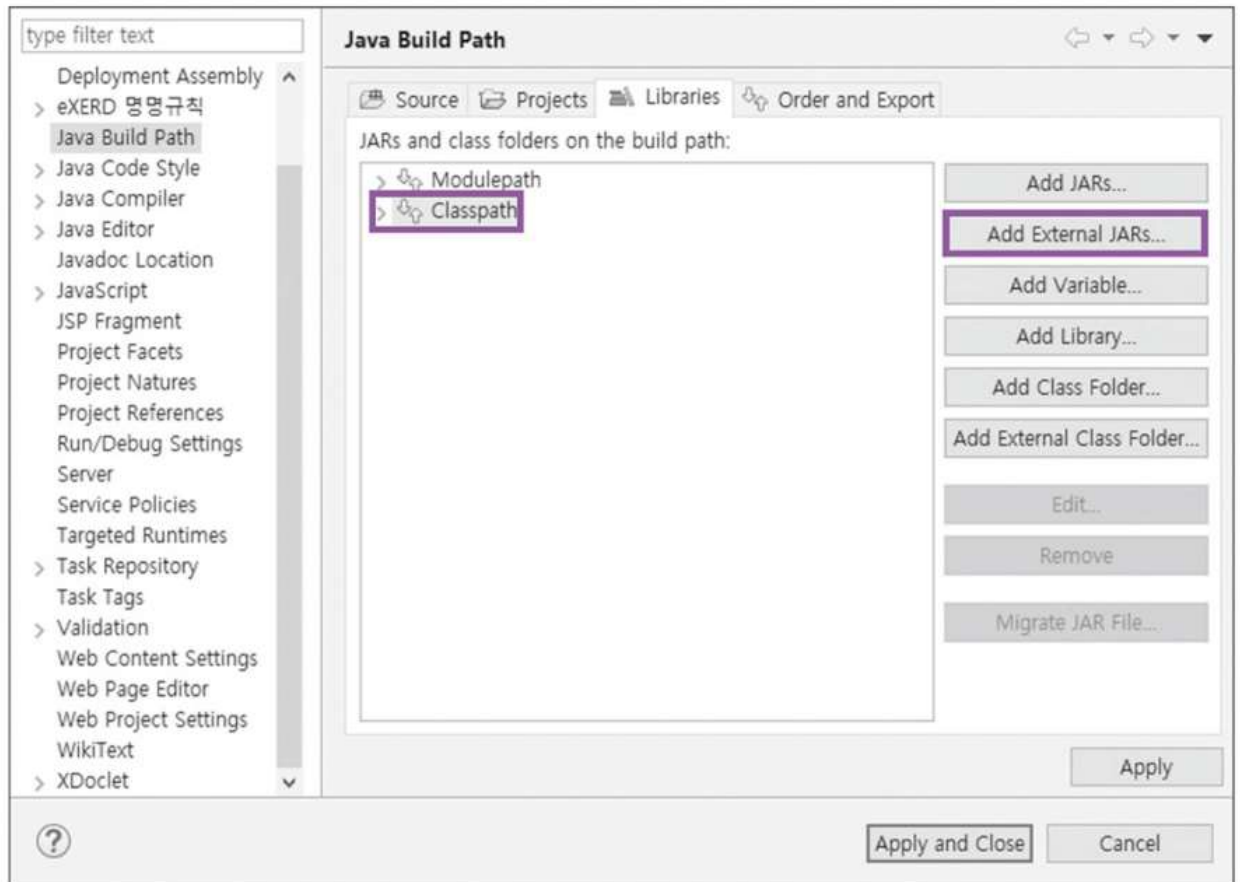
4. **Generate web.xml development descriptor** 옵션의 체크 박스에 체크한후 Finish
클릭



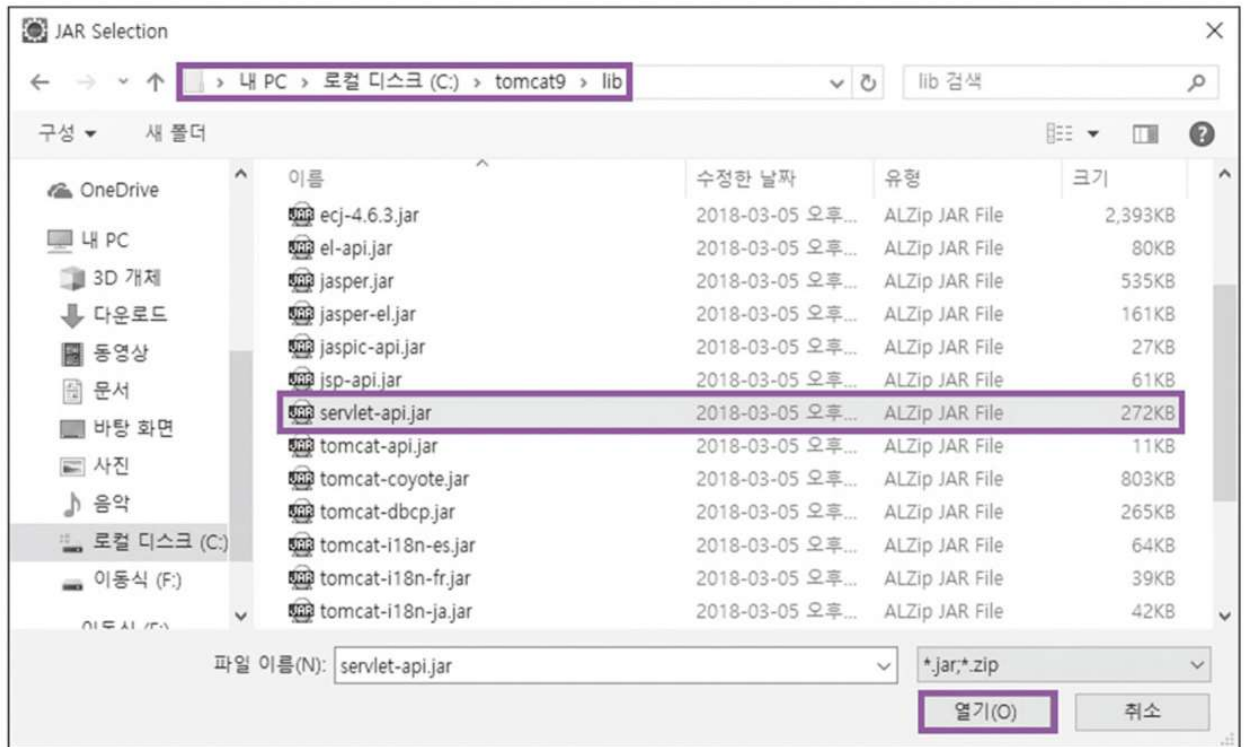
5. 프로젝트 이름을 선택하고 마우스 오른쪽 버튼을 클릭한 후 **Build Path > Configure Build Path...**를 선택



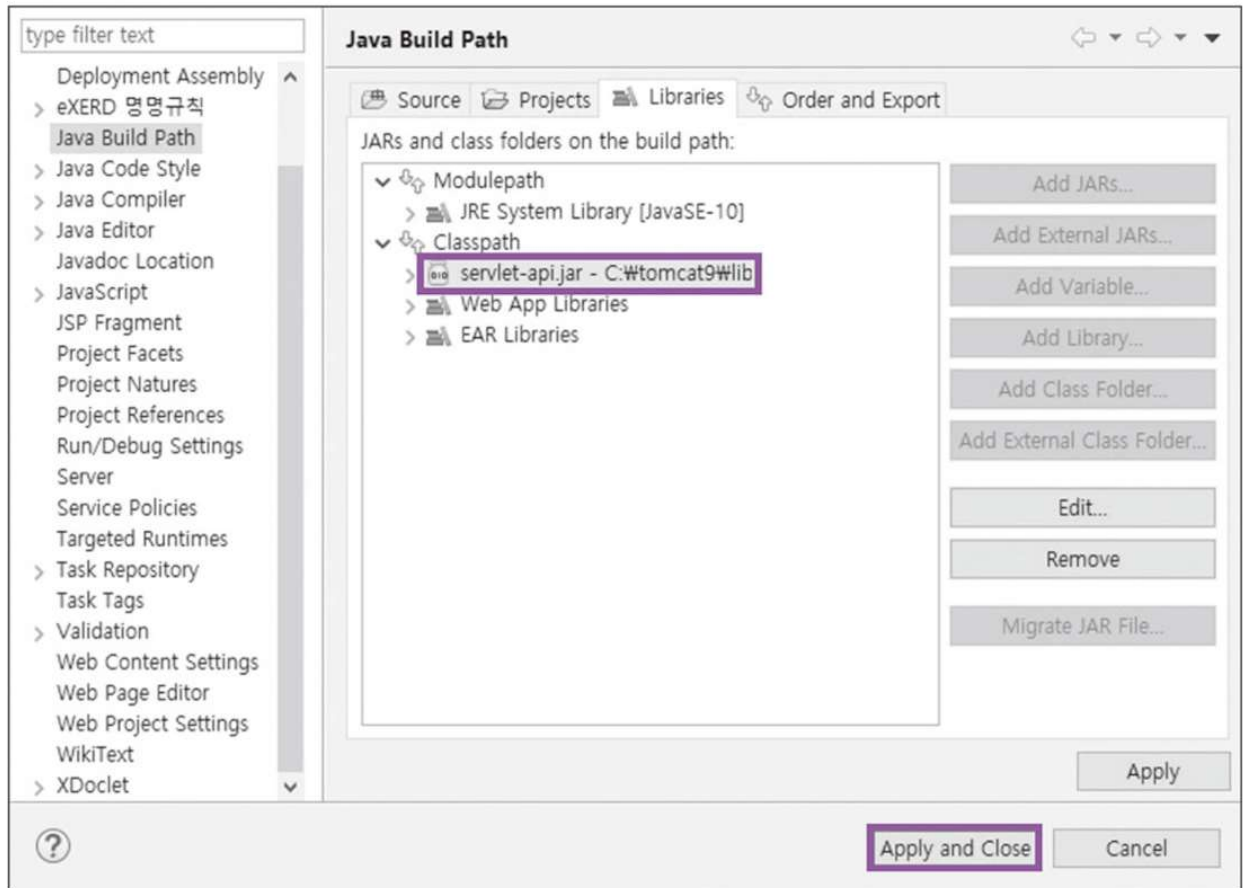
6. 설정창에서 Libraries 탭을 클릭하고 Classpath를 선택 한 후 Add External JARs...를 클릭



7. 톰캣이 설치된 디렉토리의 lib 디렉토리에 있는 `servlet-api.jar` 선택 한 후 열기 클릭



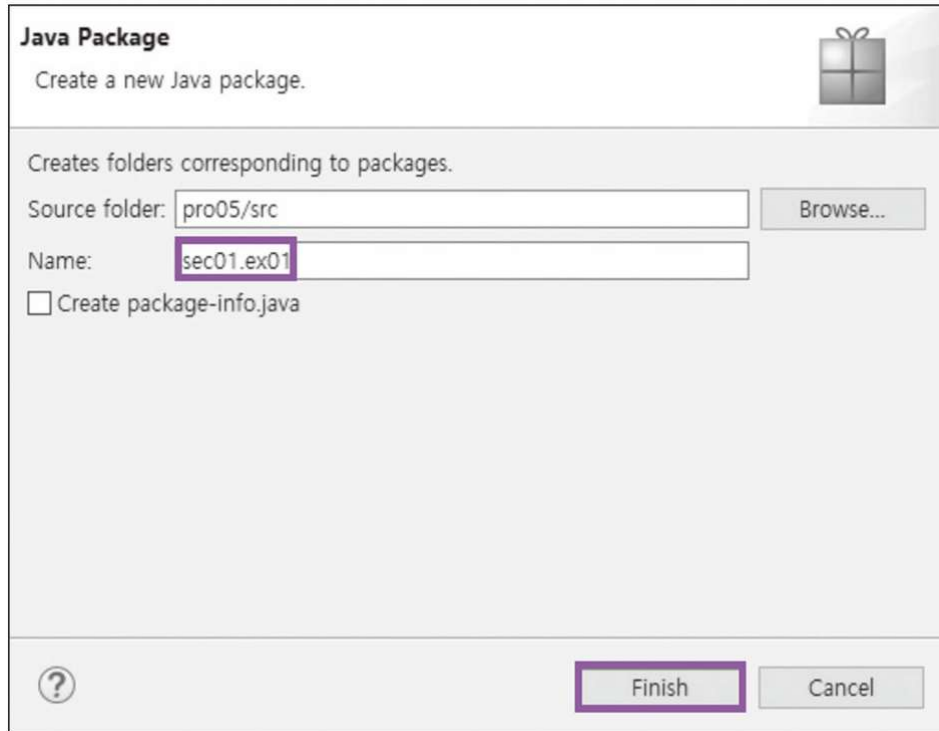
8. servlet-api.jar 클렛의 패스 설정 확인 후 Apply and Close를 클릭



첫 번째 서블릿 만들기

실제로 브라우저의 요청을 처리하는 첫 번째 서블릿을 만들어 보겠습니다.
(FirstServlet 클래스 생성)

1. sec01.ex01 패키지 생성



2. FirstServlet Class 생성

- HttpServlet으로 부터 상속 받습니다.
- doGet 메소드를 통해서 브라우저의 요청을 처리 합니다.
- HttpServlet을 상속받아 3개의 생명주기 메소드를 구현 합니다.

```
package sec01.ex01; import java.io.IOException; import
javax.servlet.ServletException; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; public class FirstServlet extends
HttpServlet { @Override public void init() throws ServletException {
System.out.println("init 메소드 호출"); } @Override protected void
doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException { System.out.println("doGet 메소드 호출"); }
@Override public void destroy() { System.out.println("destroy 메소드 호
출"); } }
```

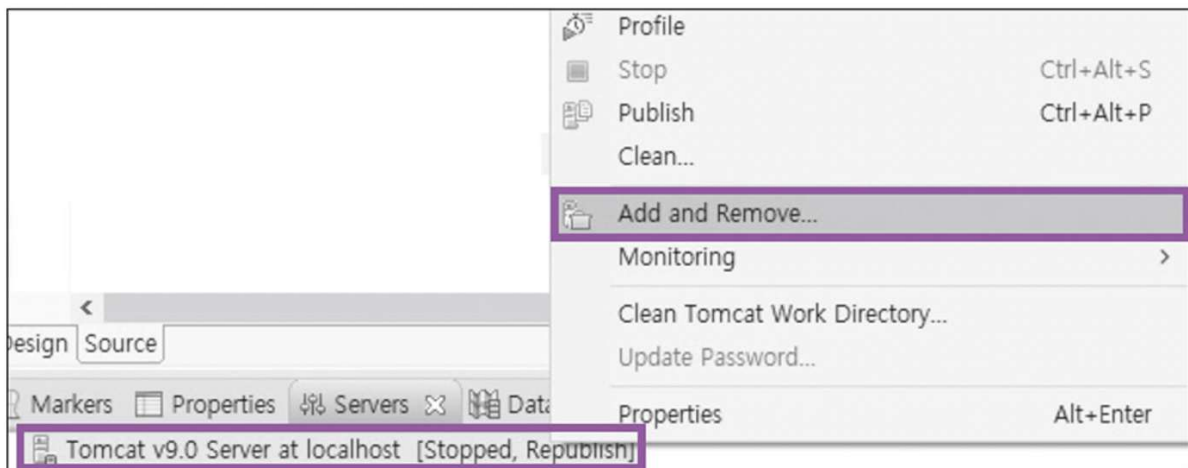
서블릿 매핑하기

브라우저에서 서블릿 이름으로 요청하는 방법을 의미 합니다.

1. 각 프로젝트에 있는 web.xml에서 설정 합니다.
2. <servlet>태그와 <servlet-mapping> 태그를 이용 합니다.
3. 여러개의 서블릿 매핑 시에는 <servlet>태그를 먼저 정의하고 <servlet-mapping> 정의 합니다.

```
<servlet> <servlet-name>aaa</servlet-name> <servlet-  
class>sec01.ex01.FirstServlet</servlet-class> </servlet> <servlet-  
mapping> <servlet-name>aaa</servlet-name> <url-pattern>/first</url-  
pattern> </servlet-mapping>
```

톰캣에 프로젝트 실행



브라우저에서 서블릿 요청하기

- 요청 형식 : **http://IP주소:포트번호/프로젝트이름(컨텍스트이름)/서블릿매핑이름**
- http://127.0.0.1:8090/pro05/first
- http://localhost:8090/pro05/first

다수의 서블릿 매핑하기

온라인 쇼핑몰과 같은 경우 대부분은 상품 조회, 주문, 회원 관리 등의 기능으로 이루어져 있습니다. 만약 이런 기능을 모두 서블릿 하나에 만들어서 제공한다면 소스가 복잡해져 관리하기가 불편 합니다.

따라서 일반적인 웹 애플리케이션은 각 기능에 대한 서블릿을 따로 만들어서 서비스를 제공합니다.

SecondServlet 추가 하기

1. New > Class를 선택
2. 클래이 이름으로 SecondServlet을 입력하고 Finish를 클릭

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

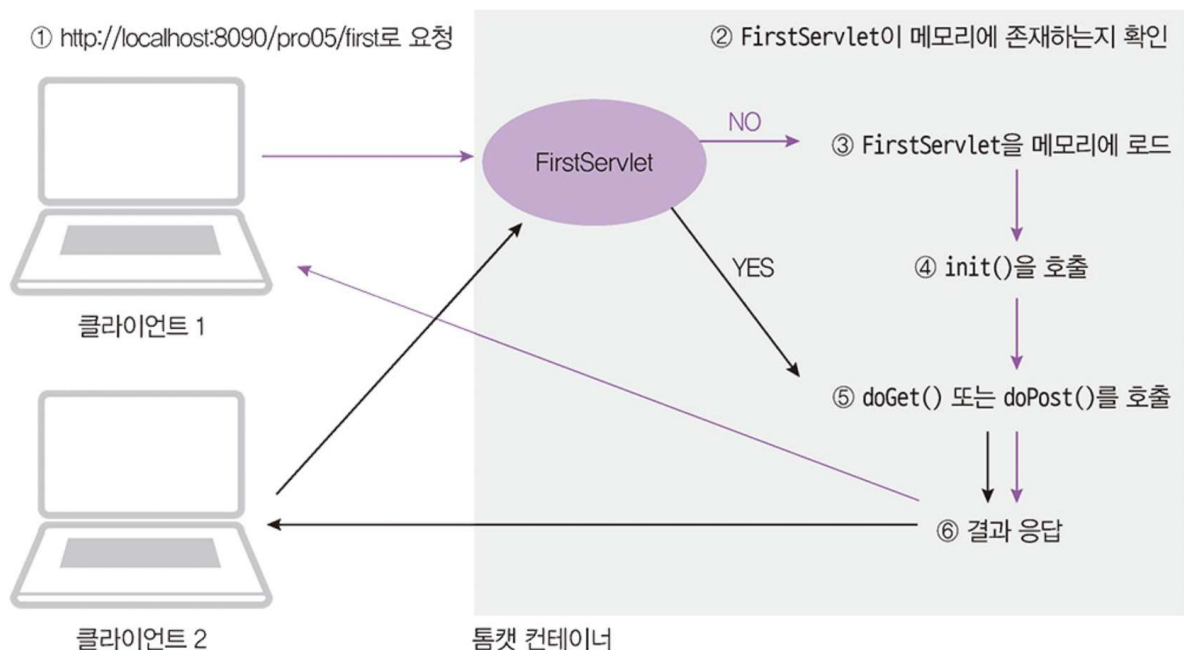
Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

```
package sec01.ex01; import java.io.IOException; import
javax.servlet.ServletException; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; public class SecondServlet
extends HttpServlet { @Override public void init() throws
ServletException { System.out.println("init 메소드 호출"); } @Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException { System.out.println("doGet 메소드 호
출"); } @Override public void destroy() { System.out.println("destroy 메소
드 호출"); } }
```

3. SecondServlet.java를 web.xml에 매핑

```
<servlet> <servlet-name>aaa</servlet-name> <servlet-
class>sec01.ex01.FirstServlet</servlet-class> </servlet> <servlet>
<servlet-name>bbb</servlet-name> <servlet-
class>sec01.ex01.SecondServlet</servlet-class> </servlet> <servlet-
mapping> <servlet-name>aaa</servlet-name> <url-pattern>/first</url-
pattern> </servlet-mapping> <servlet-mapping> <servlet-name>bbb</servlet-
name> <url-pattern>/second</url-pattern> </servlet-mapping>
```

서블릿 동작 과정



클라이언트1이 요청하면 톰캣은 FirstServlet이 메모리에 로드 되었는지 확인 합니다.
최초 요청이므로 init() 메소드가 호출하여 FirstServlet 인스턴스를 메모리에 로드 합니다.

그런 다음 doGet()이나 doPost()메소드를 호출 합니다.

클라이언트2가 다시 동일한 서블릿을 요청하면 톰캣은 다시 FirstServlet이 메모리에 로드 되었는지 확인 합니다.

이번에는 메모리에 있는 것이 확인되므로 바로 doGet()이나 doPost()메소드를 호출하여 서비스 합니다;

애너테이션을 이용한 서블릿 매핑

앞에서 실습한것처럼 여러 서블릿을 web.xml에 설정할 경우 복잡해진다는 단점이 있습니다.

각 서블릿 클래스에 기호@를 이용해서 서블릿을 표시를 해주면 훨씬 가독성이 좋아 집니다.

이처럼 소스 코드에 직접 기능을 설정하는 방법을 애너테이션이라고 합니다. (톰캣 7 이후에 추가된 기능)

애너테이션을 이용한 서블릿 매핑

@WebServlet 이용하면 되고, 애너테이션이 적용되는 클래스는 반드시 HttpServlet 클래스를 상속 받아야 합니다.

```
@WebServlet("/third") public class ThirdServlet extends HttpServlet { ...
}
```

애너테이션을 이용한 서블릿 매핑 실습

1. ThirdServlet을 입력하고 Next

2. URL mapping 이름을 선택한 후 매핑 이름을 수정하기 위해 Edit..를 클릭

Create Servlet

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

URL mappings:

/ThirdServlet

☐ Asynchronous Support

Buttons: Add..., Edit..., Remove (for both sections)

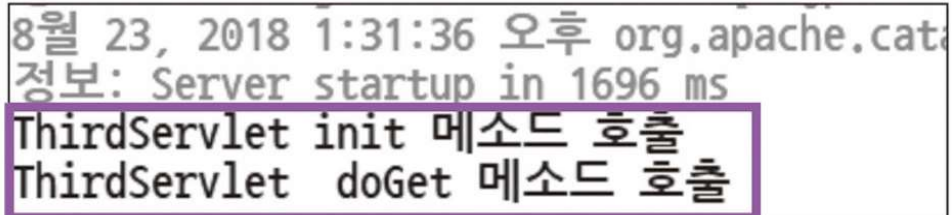
Navigation: < Back, Next >, Finish, Cancel

3. 자동 생성된 코드 확인

```
@WebServlet("/third") public class ThirdServlet extends HttpServlet {
    public void init(ServletConfig config) throws ServletException {
        System.out.println("ThirdServlet init 메소드 호출");
    } public void
    destroy() { } protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        System.out.println("ThirdServlet doGet 메소드 호출");
    } protected void
    doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException { System.out.println("ThirdServlet doPost 메
        소드 호출"); } }
```

4. 톰캣 중지 이후에 재 실행

- <http://localhost:8090/pro05/third>



8월 23, 2018 1:31:36 오후 org.apache.catalina.core.StandardEngine
정보: Server startup in 1696 ms
ThirdServlet init 메소드 호출
ThirdServlet doGet 메소드 호출

- 매핑 이름이 이미 사용된 다른 매핑 이름과 중복되지 않도록 주의 해야 합니다.