



## 5장 서블릿 이해하기

---

5.1 서블릿이란?

5.2 서블릿 API 계층 구조와 기능

5.3 서블릿의 생명 주기 메서드

5.4 FirstServlet을 이용한 실습

5.5 서블릿 동작 과정

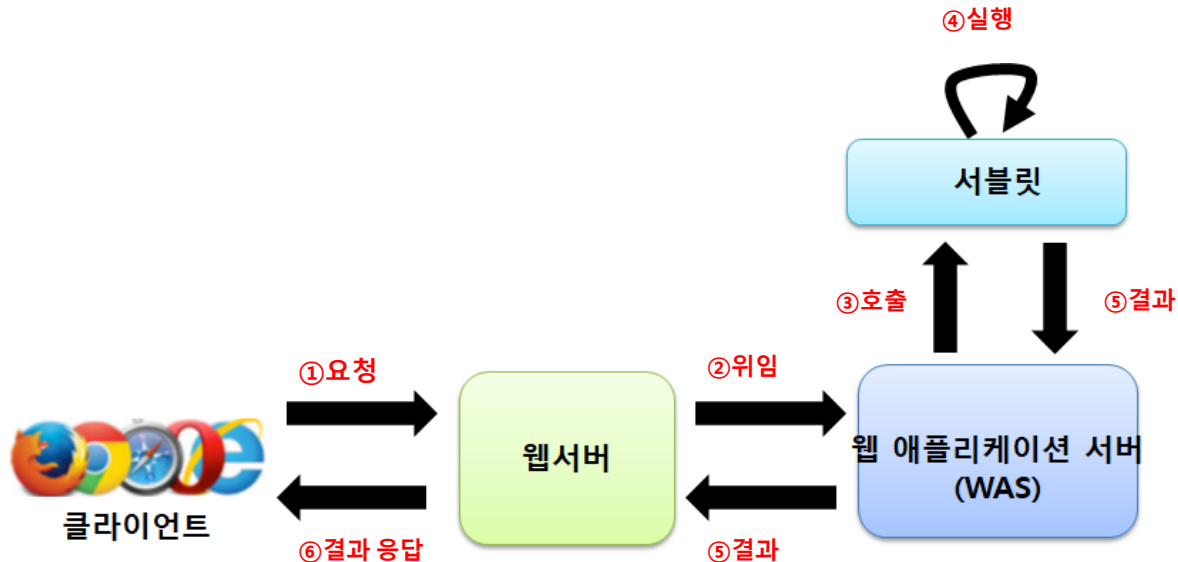
5.6 애너테이션을 이용한 서블릿 매핑

## 5.1 서블릿이란

- 서블릿이란?

➤ 서버 쪽에서 실행되면서 클라이언트의 요청에 따라 동적으로 서비스를 제공하는 자바 클래스

- 서블릿 동작 과정





## 5.1 서블릿이란

### • 서블릿 특징

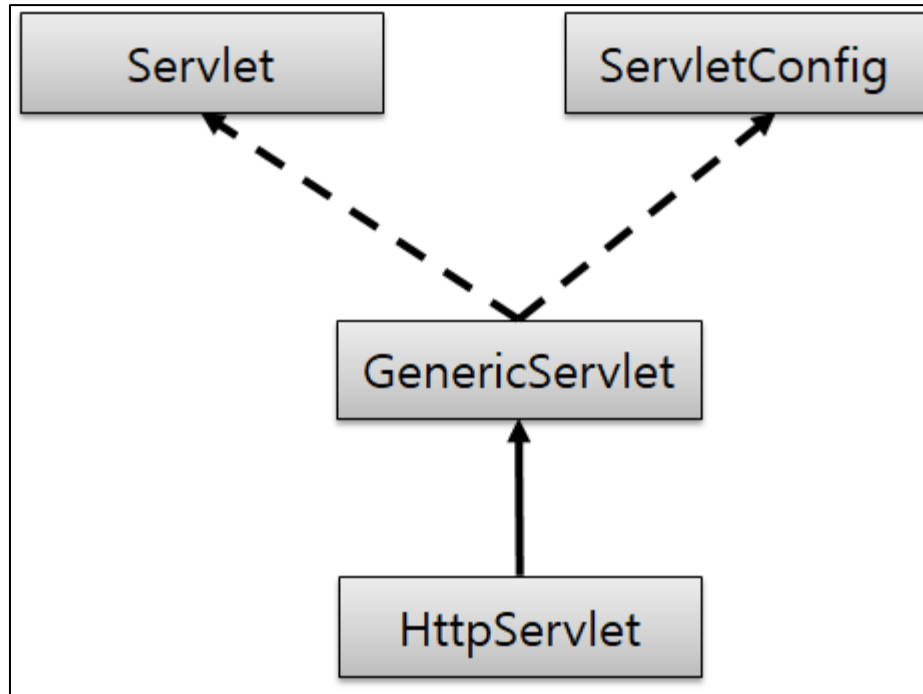
- 서버 쪽에서 실행되면서 기능을 수행함.
- 기존의 정적인 웹 프로그램의 문제점을 보완하여 동적인 여러 가지 기능을 제공함.
- 스레드 방식으로 실행됨.
- 자바로 만들어져 자바의 특징( 객체 지향)을 가짐.
- 컨테이너에서 실행됨,
- 컨테이너 종류에 상관없이 실행됨( 플랫폼 독립적).
- 보안 기능을 적용하기 쉬움.
- 웹 브라우저에서 요청 시 기능을 수행함.

### ❖ Note

이 책에서는 JSP/Servlet 컨테이너로 자바 기반 오픈 소스로 제공되는 톰캣을 사용합니다. 그 외 자바 기반의 JEUS와 WebLogic, 웹스피어( WebShpere)는 IBM 등 특정 소프트웨어 회사가 개발해서 유료로 제공하는 JSP/Servlet 컨테이너도 있습니다. 또 다른 자바 기반 오픈 소스 JSP/Servlet 컨테이너로 JBOSS는 지금은 거의 사용하지 않는 EJB 컨테이너 기능도 제공합니다.

## 5.2 서블릿 API 계층 구조와 기능

- 서블릿 클래스 계층 구조



- `GenericServlet` 추상클래스는 `Servlet`과 `ServletConfig` 인터페이스를 구현함
- `HttpServlet`은 `GenericServlet` 추상클래스를 상속받음



## 5.2 서블릿 API 계층 구조와 기능

### • 5.2.1 서블릿 API 기능

#### 서블릿 API 구성 요소 특징

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

- GenericServlet 클래스는 여러 통신 프로토콜에 대한 서블릿 기능을 구현하는함.
- GenericServlet 클래스를 상속받는 HttpServlet 클래스는 HTTP프로토콜을 사용하는 서블릿 기능을 수행함.



## 5.2 서블릿 API 계층 구조와 기능

### HttpServlet 클래스의 여러 가지 메서드 기능

메서드	기능
protected doDelete(HttpServletRequest req, HttpServletResponse resp)	서블릿이 DELETE request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doGet(HttpServletRequest req, HttpServletResponse resp)	서블릿이 GET request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doHead(HttpServletRequest req, HttpServletResponse resp)	서블릿이 HEAD request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doPost(HttpServletRequest req, HttpServletResponse resp)	서블릿이 POST request를 수행하기 위해 service()를 통해서 호출됩니다.
protected service (HttpServletRequest req, HttpServletResponse resp)	표준 HTTP request를 public service()에서 전달받아 doXXX() 메서드를 호출합니다.
public service (HttpServletRequest req, HttpServletResponse resp)	클라이언트의 request를 protected service()에게 전달합니다.

클라이언트 요청 → public service() 호출 → protected service() 호출 → doXXX() 호출



## 5.3 서블릿의 생명주기 메서드

- 서블릿 생명주기(Life Cycle) 메서드

➤ 서블릿 실행 단계마다 호출되어 기능을 수행하는 콜백 메서드

### 서블릿의 생명 주기 메서드 기능

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



init() 와 destroy() 메서드는 생략 가능하나 doXXX() 메서드는 반드시 구현해야함.



## 5.4 FirstServlet을 이용한 실습

- 서블릿 생성 과정

사용자 정의 서블릿 클래스 만들기



서블릿 생명주기 메서드 구현



서블릿 매핑 작업



웹 브라우저에서 서블릿 매핑 이름으로 요청하기





## 5.4 FirstServlet을 이용한 실습

- 5.4.1 사용자 정의 서블릿 만들기

반드시 HttpServlet 클래스를  
상속받아야 합니다.

코드 사용자 정의 서블릿 형식

```
public class FirstServlet extends HttpServlet {  
    @Override  
    public void init() {  
        ...  
    }  
    @Override  
protected public void doGet(HttpServletRequest req, HttpServletResponse resp) {  
        ...  
    }  
  
    @Override  
    public void destroy() {  
        ...  
    }  
}
```



init(), doGet() 또는 doPost(), destroy() 메서드를 오버라이딩해서 구현합니다.

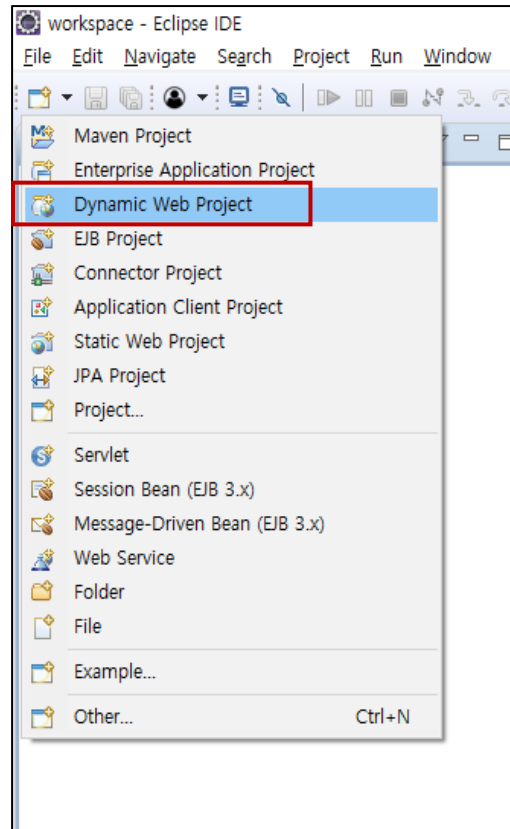


## 5.4 FirstServlet을 이용한 실습

- 5.4.2 톰캣의 servlet-api.jar 클래스 패스 설정하기

➤ 서블릿 API는 톰캣의 servlet-api.jar 라이브러리로 제공되므로 클래스 패스를 설정해야 사용 할수 있음

1. 이클립스 상단의 New 아이콘을 클릭한 후 Dynamic Web Project를 선택합니다.





## 5.4 FirstServlet을 이용한 실습

2. 프로젝트 이름을 pro05로 입력한 후 Next를 클릭합니다.

**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:

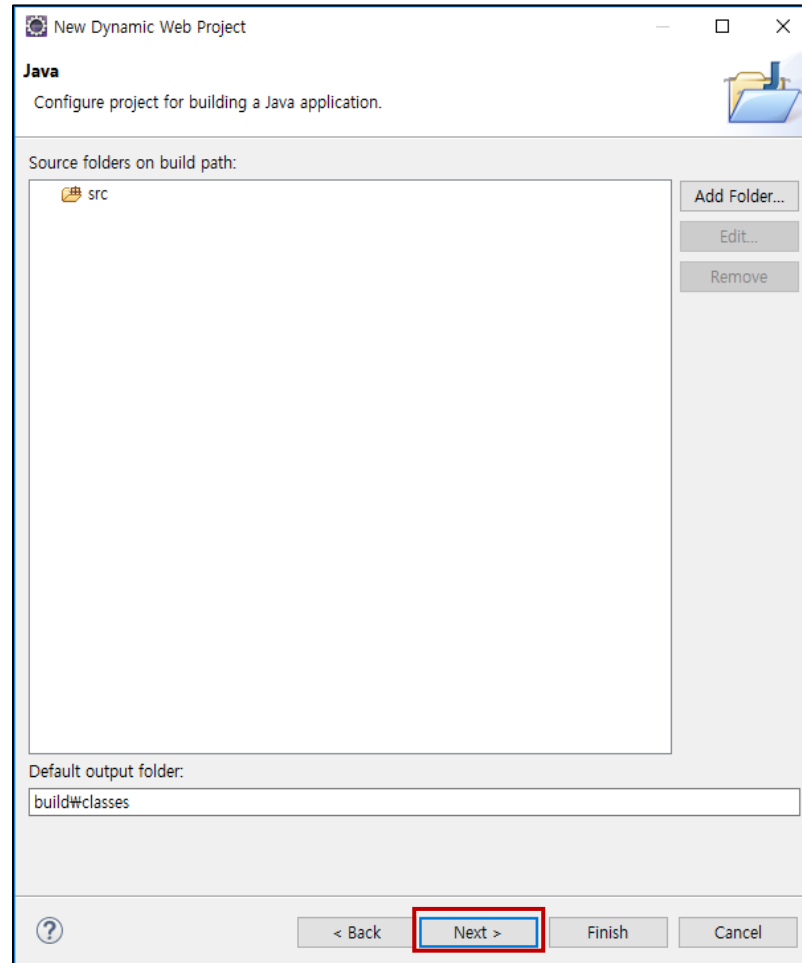
Working sets  
☐ Add project to working sets   
Working sets:

? < Back **Next >** Finish Cancel



## 5.4 FirstServlet을 이용한 실습

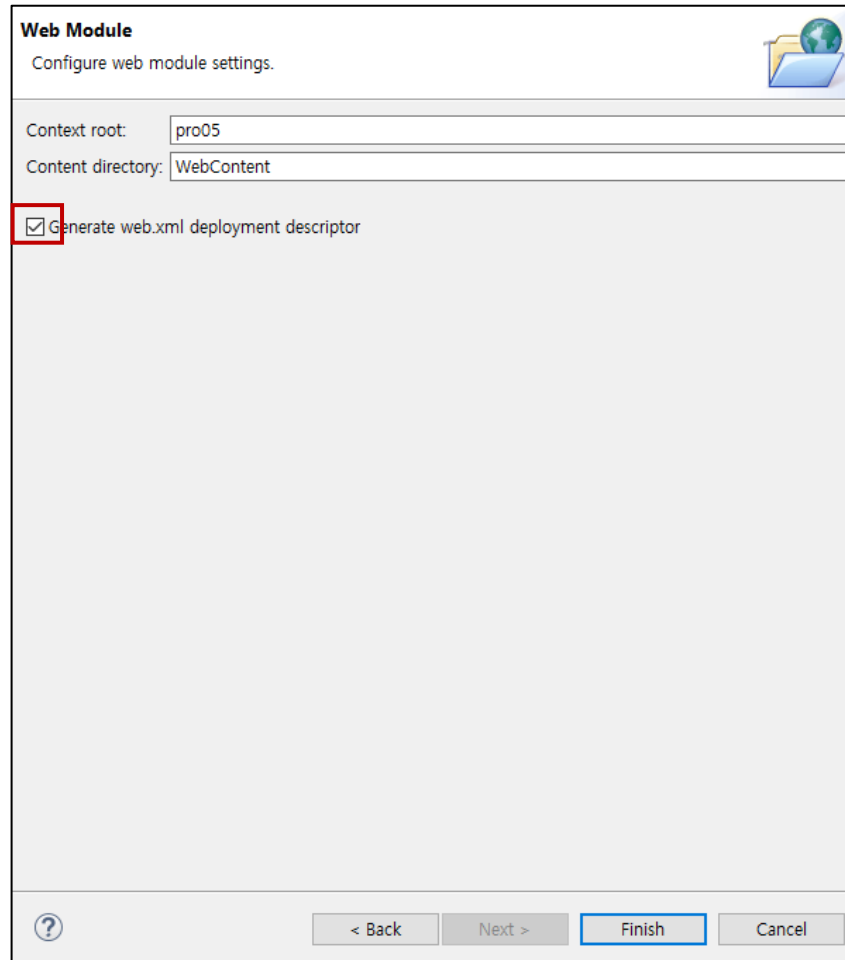
3. 경로를 확인한 후 Next를 클릭합니다.





## 5.4 FirstServlet을 이용한 실습

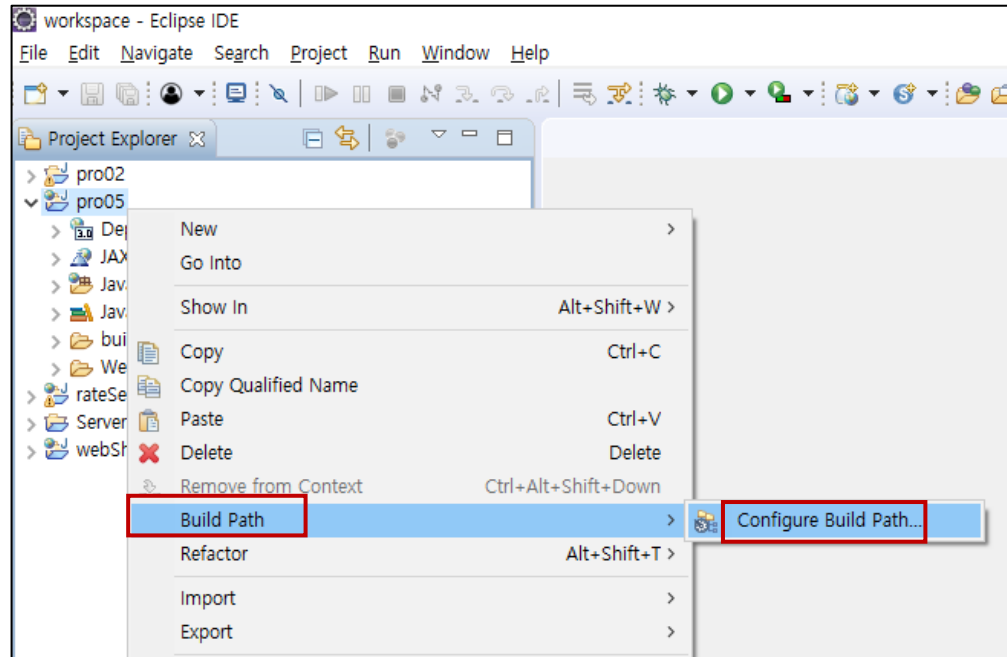
4. Generate web.xml deployment descriptor 옵션의 체크박스에 체크한 후 Finish를 클릭합니다.



The image shows a 'Web Module' configuration dialog box. At the top, it says 'Web Module' and 'Configure web module settings.' Below this, there are two text fields: 'Context root:' with the value 'pro05' and 'Content directory:' with the value 'WebContent'. Below these fields is a checkbox labeled 'Generate web.xml deployment descriptor', which is checked. At the bottom of the dialog, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish' (which is highlighted with a blue border). A 'Cancel' button is also present.

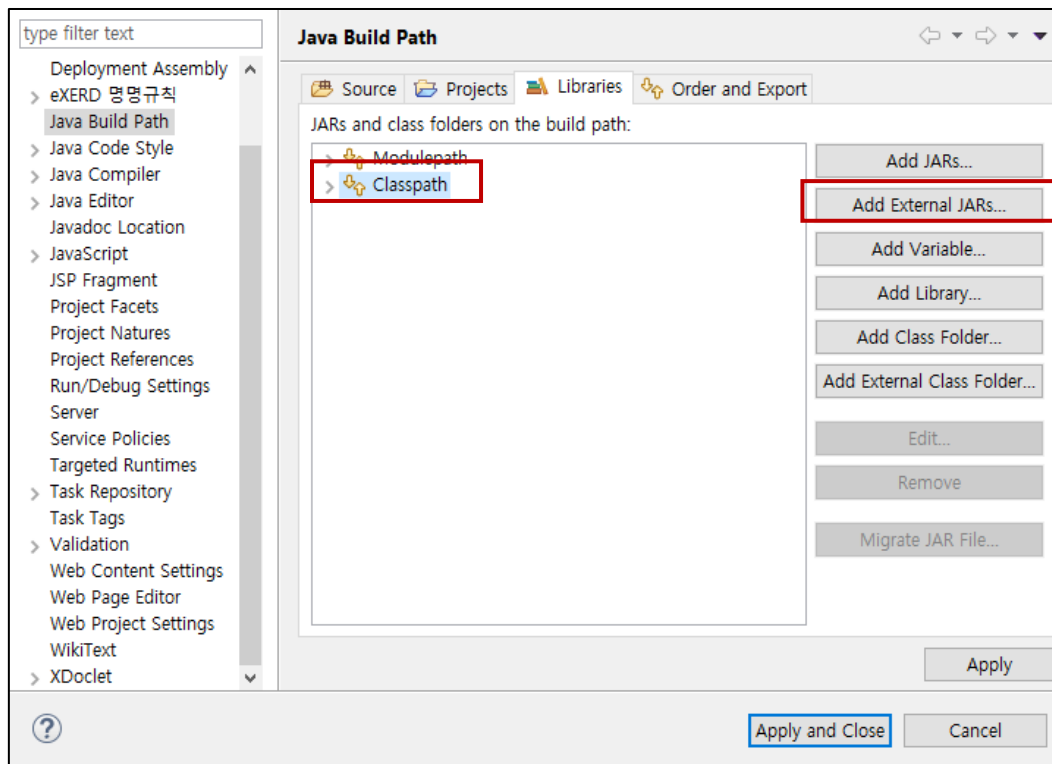
## 5.4 FirstServlet을 이용한 실습

5. 프로젝트 이름을 선택하고 마우스 오른쪽 버튼을 클릭한 후 Build Path > Configure BuildPath...를 선택합니다.



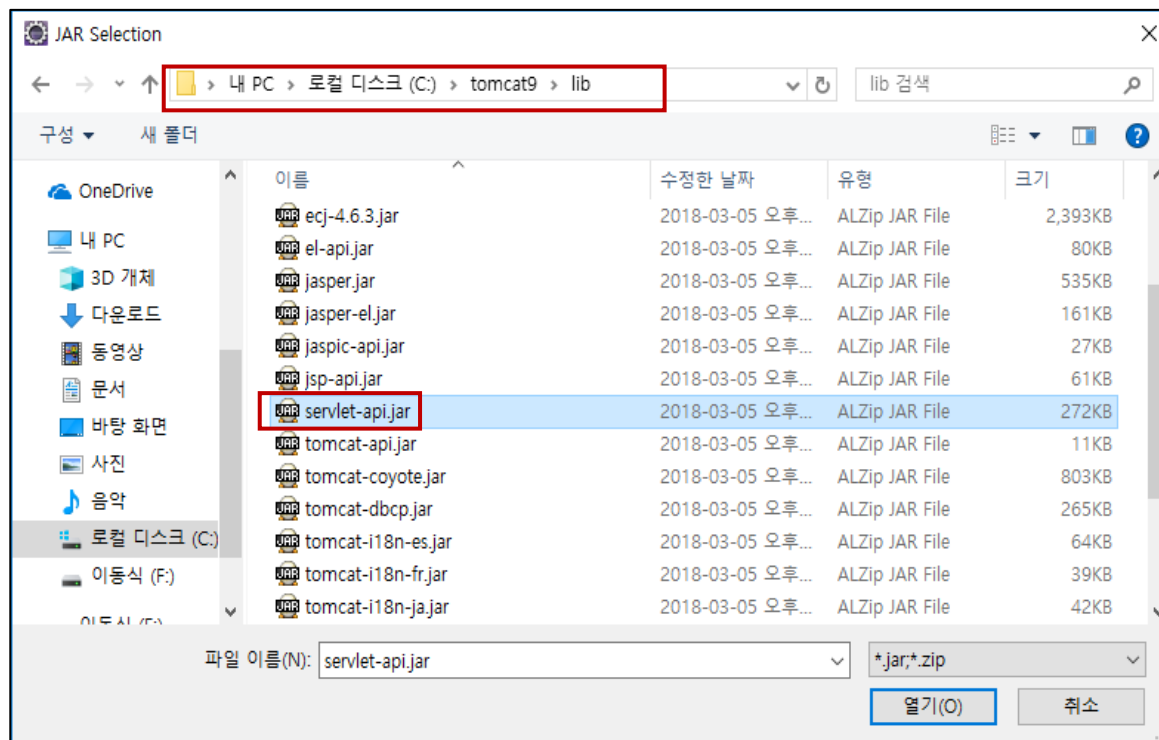
## 5.4 FirstServlet을 이용한 실습

6. 설정창에서 Libraries 탭을 클릭하고 Classpath를 선택한 후 Add External JARs...를 클릭합니다.



## 5.4 FirstServlet을 이용한 실습

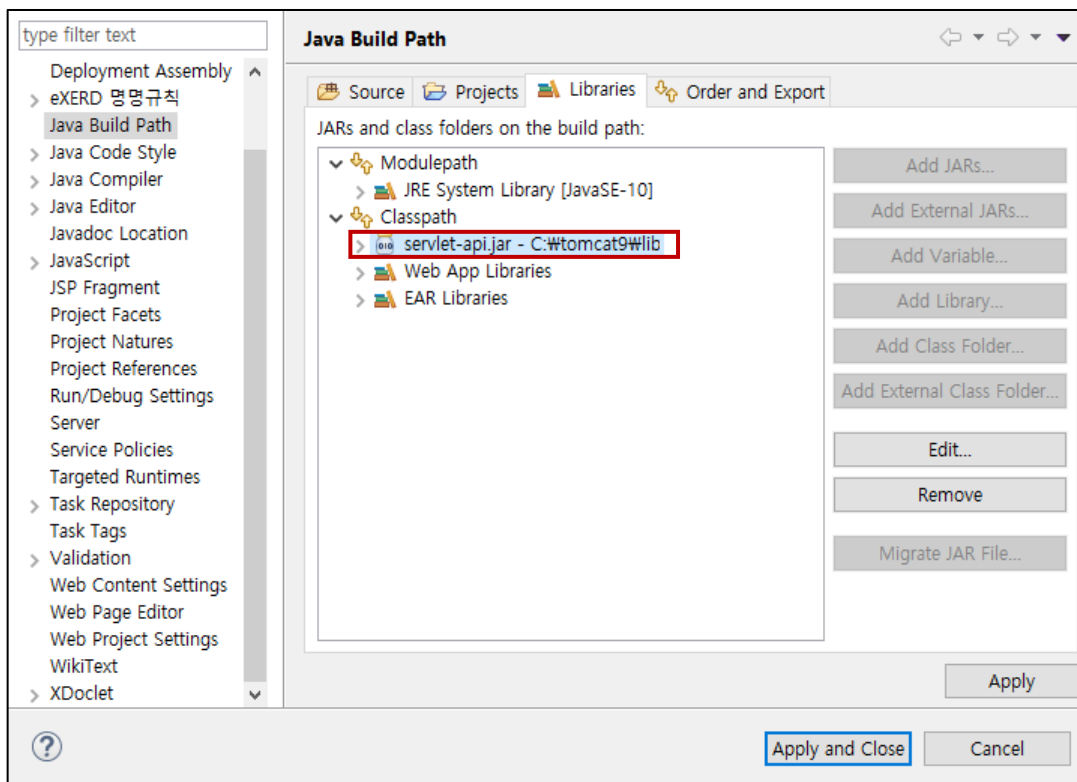
7. CATALINA\_HOME(톰캣 루트 디렉터리)의 lib 디렉터리에 있는 servlet-api.jar를 선택한 후 열기를 클릭합니다.





## 5.4 FirstServlet을 이용한 실습

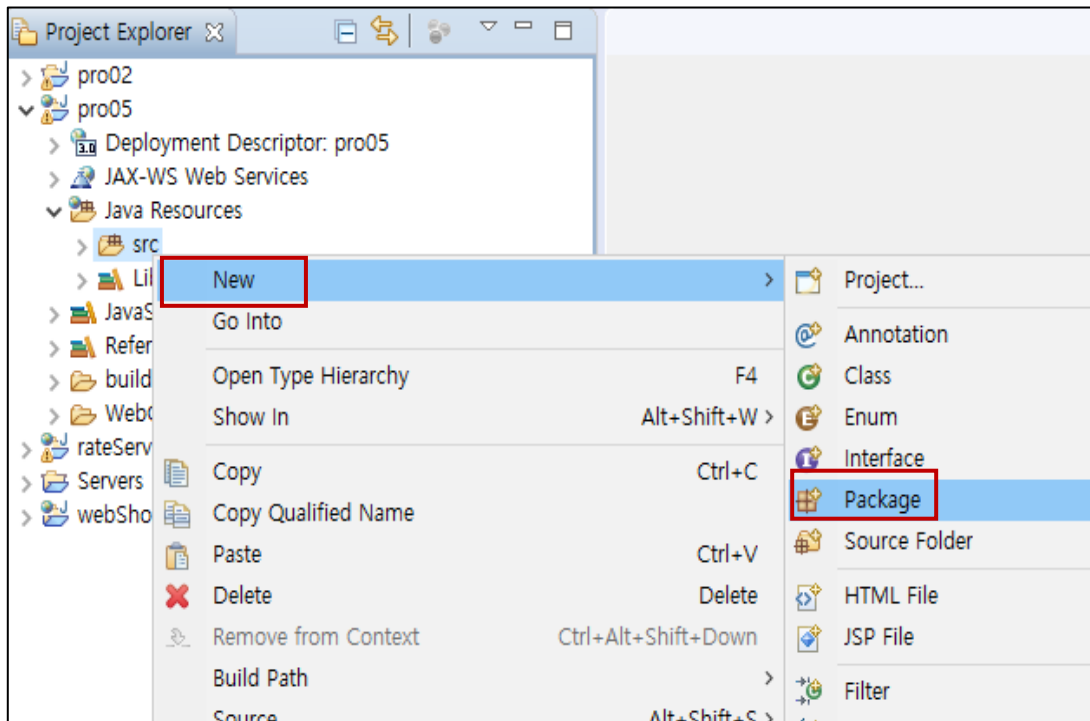
8. servlet-api.jar 클래스의 패스 설정을 확인한 후 Apply and Close를 클릭해 종료합니다.



## 5.4 FirstServlet을 이용한 실습

### • 5.4.3 첫번째 서블릿 만들기

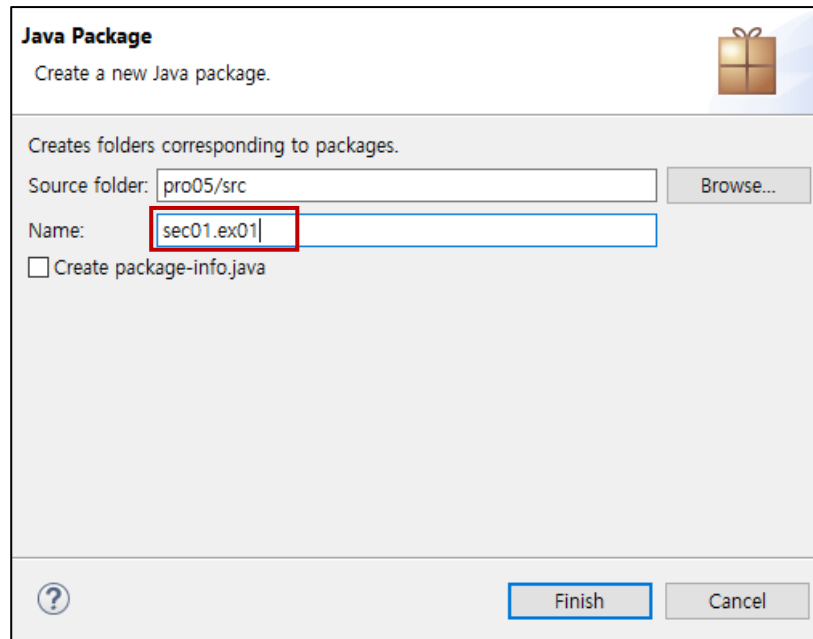
1. pro05 프로젝트의 Java Resources 디렉터리 하위의 src를 선택하고 마우스 오른쪽 버튼을 클릭한 후 New > Package를 선택합니다.





## 5.4 FirstServlet을 이용한 실습

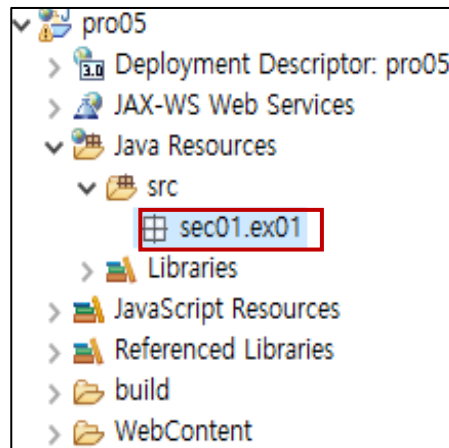
2. sec01.ex01이라는 이름으로 패키지를 생성합니다.





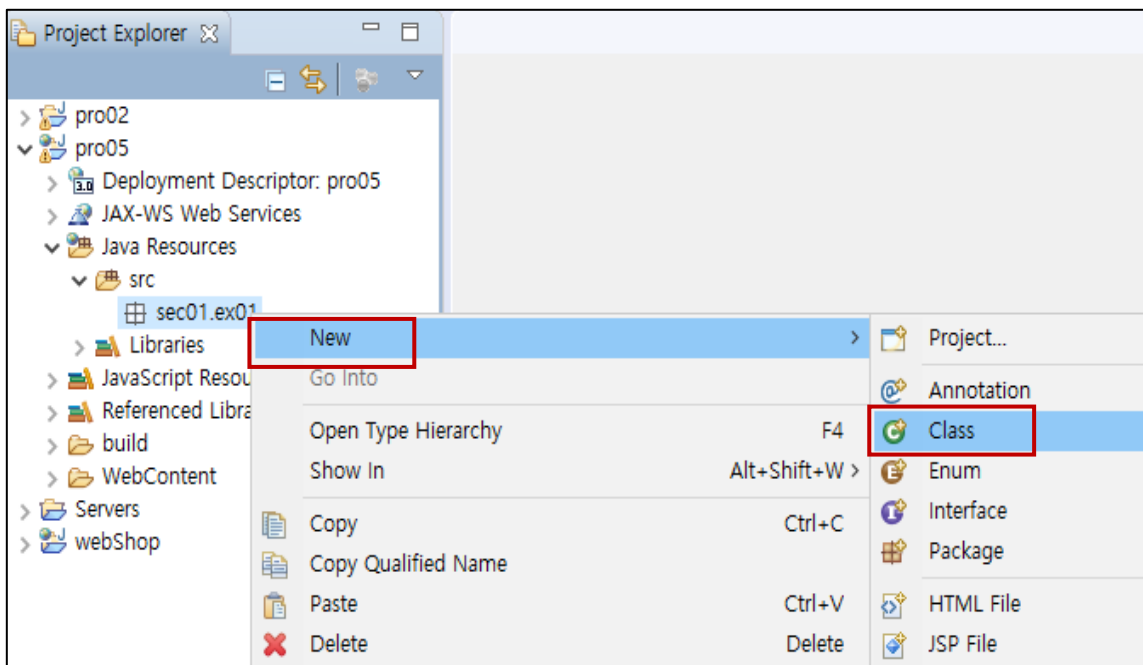
## 5.4 FirstServlet을 이용한 실습

3. Project Explorer에서 src 하위에 sec01.ex01이라는 패지지가 생긴 것을 확인할 수 있습니다.



## 5.4 FirstServlet을 이용한 실습

4. 이 패키지 이름 위에서 마우스 오른쪽 버튼을 클릭한 후 New > Class를 선택합니다.





## 5.4 FirstServlet을 이용한 실습

5. 클래스 이름으로 FirstServlet을 입력한 후 Finish를 클릭합니다.

**Java Class**  
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

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

Superclass:

Interfaces:

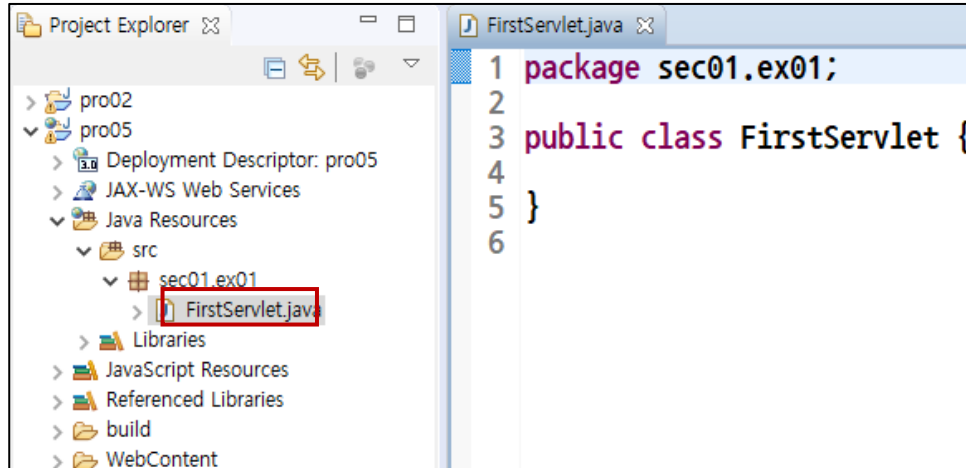
Which method stubs would you like to create?  
☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

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



## 5.4 FirstServlet을 이용한 실습

6. FirstServlet.java가 생성된 것을 확인할 수 있습니다.





## 5.4 FirstServlet을 이용한 실습

코드 5-1 pro05/src/sec01/ex01/FirstServlet.java

```
package sec01.ex01;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
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 메서드 호출");
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("doGet 메서드 호출");
    }

    @Override
    public void destroy() {
        System.out.println("destroy 메서드 호출");
    }
}
```

HttpServlet를 상속받습니다.

브라우저의 요청을 처리합니다.





## 5.4 FirstServlet을 이용한 실습

- 5.4.4 서블릿 매핑하기

`http://주소:포트번호/프로젝트명/패키지명이 포함된 클래스명`



`http://127.0.0.1:8090/pro05/sec01.ex01.FirstServlet`

- 클래스 이름이 길어지면 불편함.
- 클래스 이름을 사용하면 보안에도 좋지않음



서블릿 클래스에 대응하는 서블릿 매핑 이름으로 요청함



## 5.4 FirstServlet을 이용한 실습

### 서블릿 매핑 방법

- 각 프로젝트에 있는 web.xml에서 설정
- <servlet> 태그와 <servlet-mapping> 태그를 이용함.
- 여러 개의 서블릿 매핑 시에는 <servlet> 태그를 먼저 정의하고 <servlet-mapping> 태그를 정의함.

코드 5-2

```
<servlet>
  <servlet-name>aaa</servlet-name>
  <servlet-class>sec01.ex01.FirstServlet</servlet-class>
</servlet>
```

<servlet> 태그와 <servlet-mapping>  
태그를 연결시켜 줍니다.

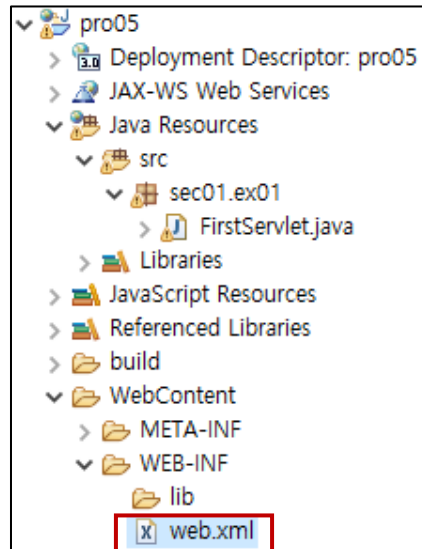
```
<servlet-mapping>
  <servlet-name>aaa</servlet-name>
  <url-pattern>/first</url-pattern>
</servlet-mapping>
```

웹 브라우저에서 요청하는 매핑 이름



## 5.4 FirstServlet을 이용한 실습

1. pro05 프로젝트의 WebContent > WEB-INF 폴더를 클릭한 후 web.xml을 선택하여 엽니다.





## 5.4 FirstServlet을 이용한 실습

2. web.xml에 <web-app> 태그의 하위 태그를 지우고 다음과 같이 서블릿 매핑을 작성합니다.

코드 5-3 pro05/WebContent/WEB-INF/web.xml

```
<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>
```

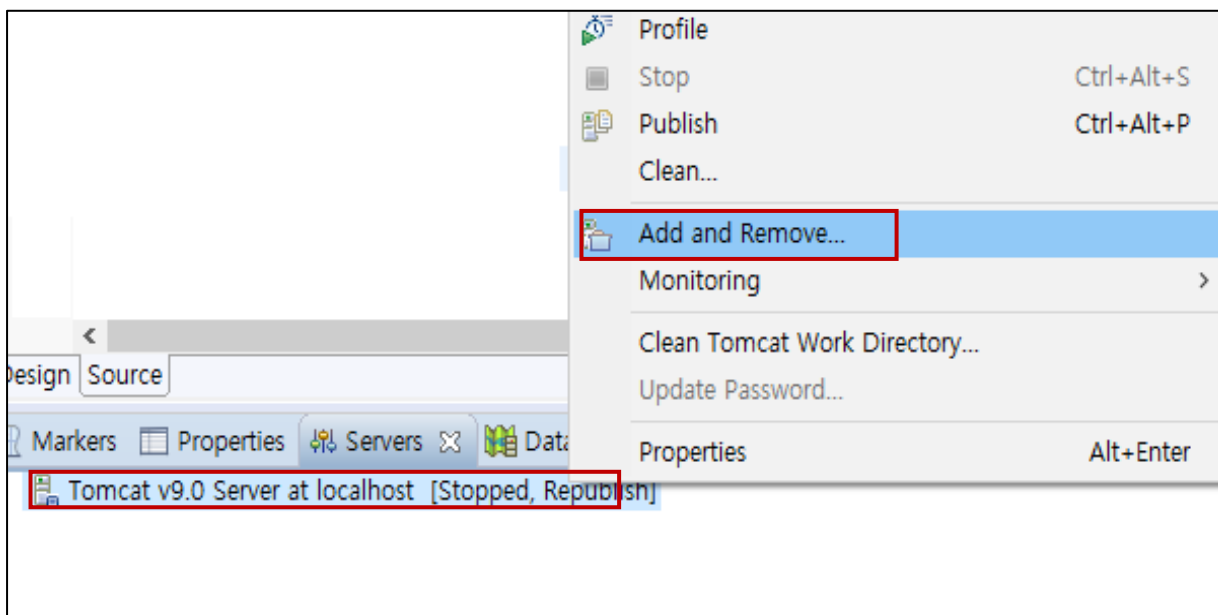
— 브라우저에서 요청하는 매핑 이름에 대해 실제로 실행하는 서블릿 클래스를 설정하는 태그입니다.  
 — <servlet-mapping> 태그의 <servlet-name> 태그와 값이 동일합니다.  
 — 브라우저에서 요청하는 매핑 이름에 대해 실제로 기능을 수행하는 서블릿 클래스를 설정합니다.  
 — 매핑 이름으로 요청 시 값이 같은 <servlet> 태그 안의 <servlet-name> 태그와 연결됩니다.  
 — 브라우저에서 요청하는 논리적인 서블릿을 설정합니다.  
 — 브라우저에서 sec01.ex01.FirstServlet을 요청하는 논리적인 서블릿 이름입니다.



## 5.4 FirstServlet을 이용한 실습

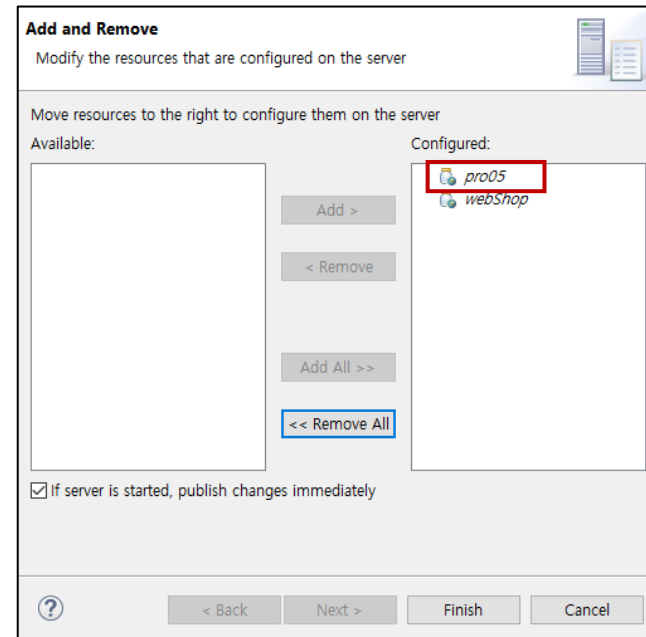
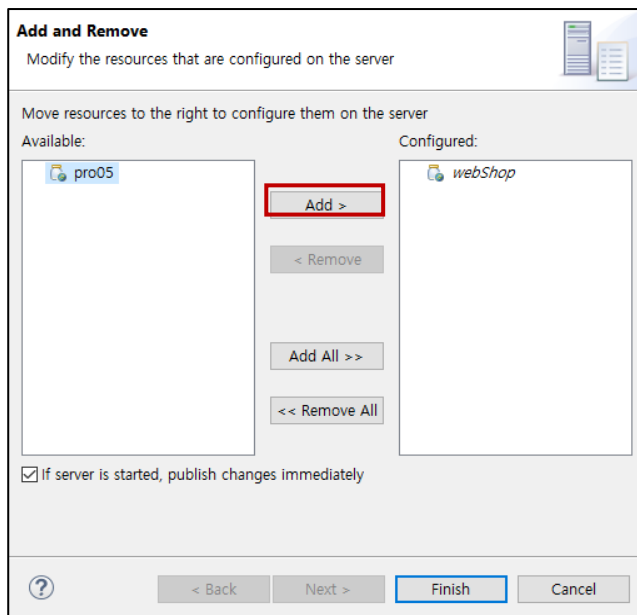
- 5.4.5 톰캣 프로젝트 실행

1. 톰캣 서버를 선택하고 마우스 오른쪽 버튼을 클릭한 후 Add and Remove...를 선택합니다.



## 5.4 FirstServlet을 이용한 실습

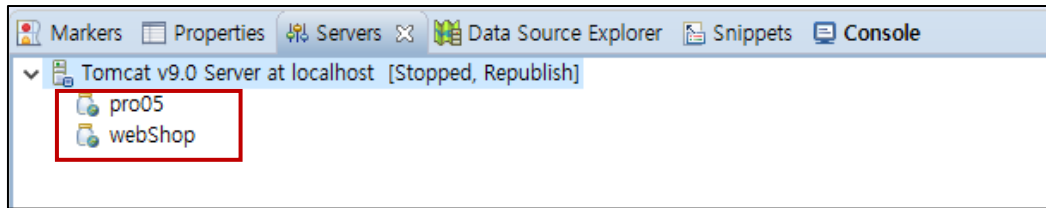
2. pro05 프로젝트를 선택한 후 Add를 클릭하여 추가하고 Finish를 클릭합니다.





## 5.4 FirstServlet을 이용한 실습

3. 톰캣에 정상적으로 새 프로젝트가 등록된 것을 확인할 수 있습니다.





## 5.4 FirstServlet을 이용한 실습

- 5.4.6 브라우저에서 서블릿 요청하기

브라우저에서 서블릿 요청 방법

- `http://IP주소:포트번호/프로젝트이름(컨텍스트이름)/서블릿매핑이름`
- 요청 예: `http://127.0.0.1:8090/pro05/first`

❖ Note

톰캣이 로컬 PC에 설치된 경우에는 다음과 같이 입력해도 됩니다.

`http://localhost:8090/pro05/first`

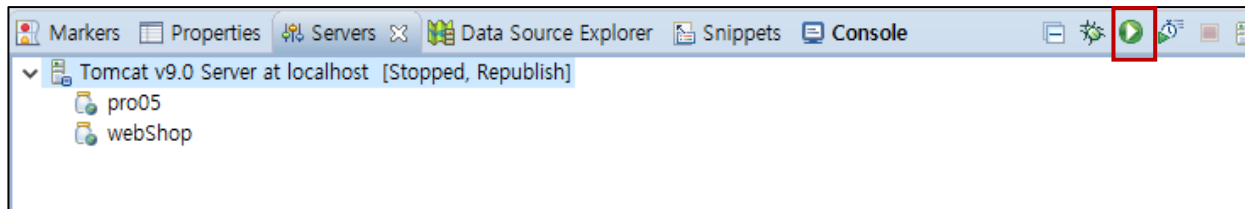
❖ Tip

자신의 IP 주소를 확인하려면 명령 프롬프트창에서 ipconfig 명령을 입력하면 됩니다.

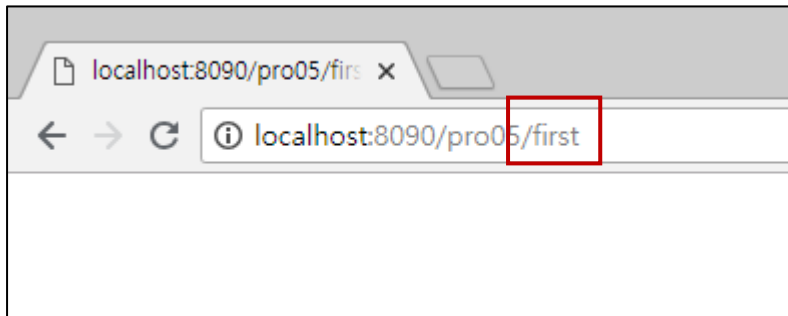


## 5.4 FirstServlet을 이용한 실습

1. 이클립스에서 톰캣을 다시 실행합니다.



2. 브라우저에서 서블릿 매핑 이름으로 요청합니다.





## 5.4 FirstServlet을 이용한 실습

3. /first로 웹 브라우저에서 요청하면 이클립스 콘솔에 각각의 메서드가 호출되면서 메시지가 출력됩니다.

```
8월 23, 2018 1:13:38 오후 org.apache.coyote.A
정보: Starting ProtocolHandler ["http-nio-8090
8월 23, 2018 1:13:38 오후 org.apache.coyote.A
정보: Starting ProtocolHandler ["ajp-nio-8009"
8월 23, 2018 1:13:38 오후 org.apache.catalina.
정보: Server startup in 1798 ms
init 메서드 호출
doGet 메서드 호출
```



## 5.4 FirstServlet을 이용한 실습

### ❖ Note

다음은 web.xml에 서블릿 매핑을 잘못된 상태에서 톰캣을 실행했을 때 나타난 오류 메시지입니다. 오류가 발생한 상태에서 웹 브라우저에 요청하면 정상적으로 실행되지 않습니다.

web.xml에 서블릿 매핑을 할 경우에는 문법이나 태그의 철자가 틀리지 않도록 대소문자까지 주의해서 입력해야 합니다. 톰캣을 실행할 때는 어떤 오류도 발생하면 안 됩니다. 오류가 발생하면 그 원인을 찾아 바로 수정한 후 다시 실행해야 합니다.

```
FirstServlet.java  web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   <servlet>
4     <servlet-name>aaa</servlet-name>
5     <servlet-class>sec01.ex01.FirstServlet</servlet-class>
6   </servlet>
7   <servlet-mapping>
8     <sevlet-name>aaa</sevlet-name>
9     <url-pattern>/first</url-pattern>
10  </servlet-mapping>
11 </web-app>
```

<servlet>을 <sevlet>로 잘못입력한 경우

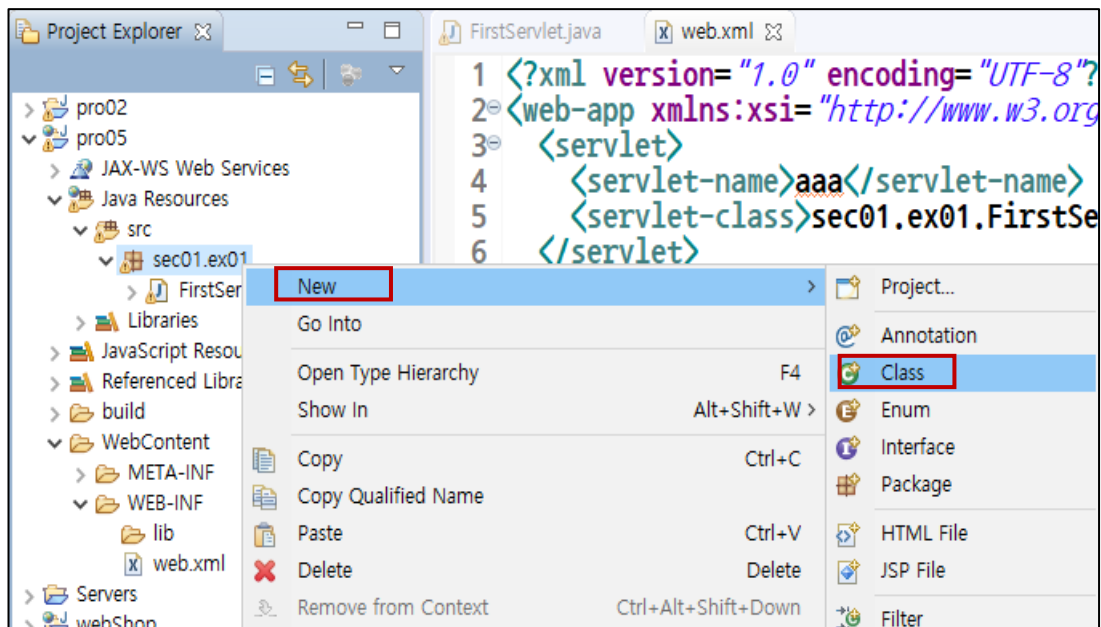
```
Markers Properties Servers Data Source Explorer Snippets Problems Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (2018. 4. 18. 오후 5:13:27)
정보: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a c
월 18, 2018 5:13:32 오후 org.apache.tomcat.util.digester.Digester endElement
실각: End event threw exception
java.lang.IllegalArgumentException: Can't convert argument: null
    at org.apache.tomcat.util.IntrospectionUtils.convert(IntrospectionUtils.java:450)
    at org.apache.tomcat.util.descriptor.web.CallMethodMultiRule.end(WebRuleSet.java:992)
    at org.apache.tomcat.util.digester.Digester.endElement(Digester.java:944)
    at java.xml/com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.endElement(Unknown Source)
    at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanEndElement(Unk
    at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl$FragmentContentD
    at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentScannerImpl.next(Unknown Source)
    at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unk
    at java.xml/com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(Unknown Source)
```

## 5.4 FirstServlet을 이용한 실습

### • 5.4.7 다수의 서블릿 매핑하기

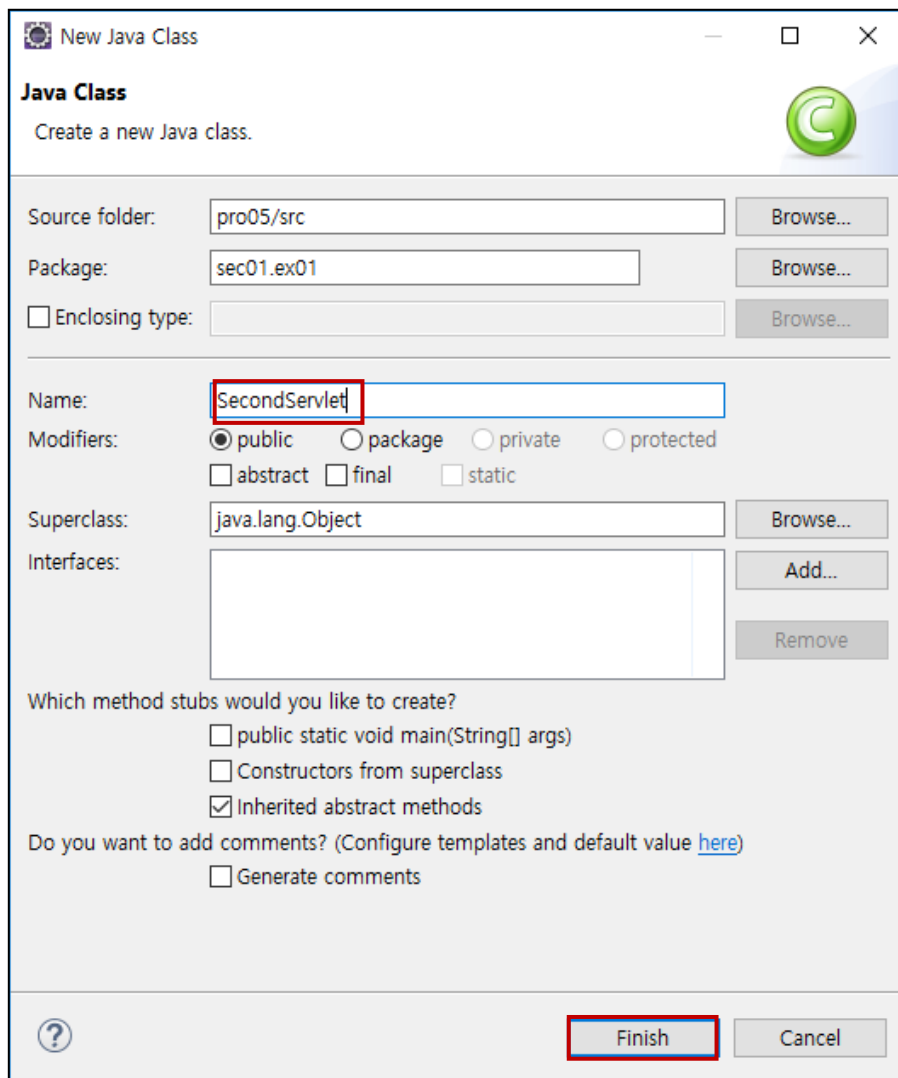
➤ 일반적인 웹 애플리케이션은 각 기능에 대한 서블릿을 따로 만들어서 서비스를 제공함.

1. 패키지 sec01.ex01을 선택하고 마우스 오른쪽 버튼을 클릭한 후 New > Class를 선택합니다.



## 5.4 FirstServlet을 이용한 실습

2. 클래스 이름으로 SecondServlet을 입력하고 Finish를 클릭합니다.



**New 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?

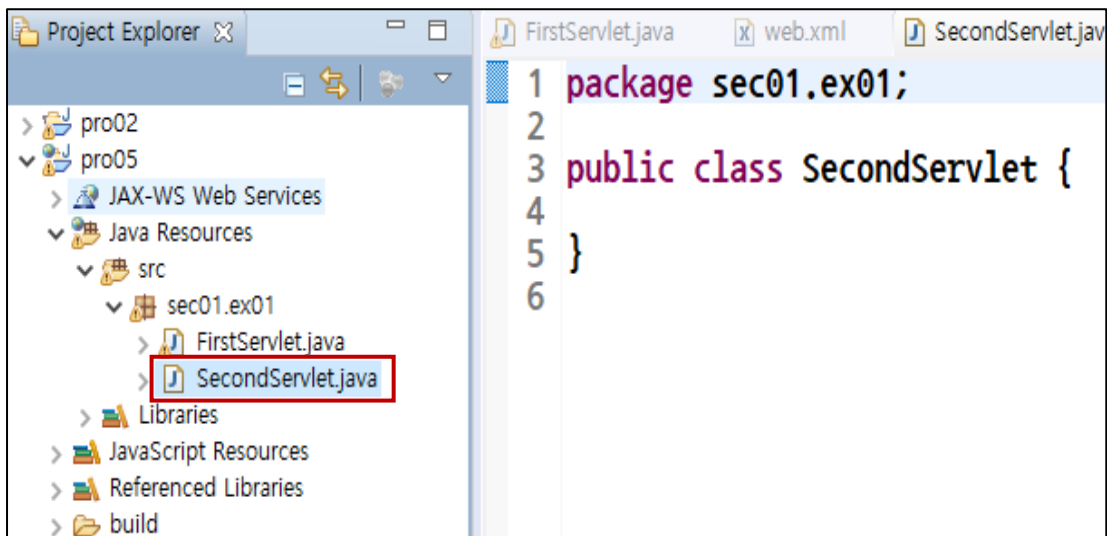
☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

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



## 5.4 FirstServlet을 이용한 실습

3. Project Explorer에 SecondServlet.java가 생성된 것을 확인할 수 있습니다.





## 5.4 FirstServlet을 이용한 실습

4. 또 다른 서블릿 기능을 하는 SecondServlet 클래스를 다음과 같이 작성합니다.

코드 5-4 pro05/src/sec01/ex01/SecondServlet.java

```
package sec01.ex01;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
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 메서드 호출>>>>");
    }
}
```



## 5.4 FirstServlet을 이용한 실습

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException
{
    System.out.println("doGet 메서드 호출>>>>");
}

@Override
public void destroy() {
    System.out.println("destroy 메서드 호출>>>>");
}
}
```

---





## 5.4 FirstServlet을 이용한 실습

5. 다시 SeocndServlet.java를 web.xml에 매핑을 합니다.

코드 5-5 pro05/WebContent/WEB\_INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
<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> 태그끼리 위치시킵니다.

— <servlet-name> 태그 값은 다른  
<servlet-name> 태그 값과 절대  
같으면 안 됩니다.

```
<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>
```

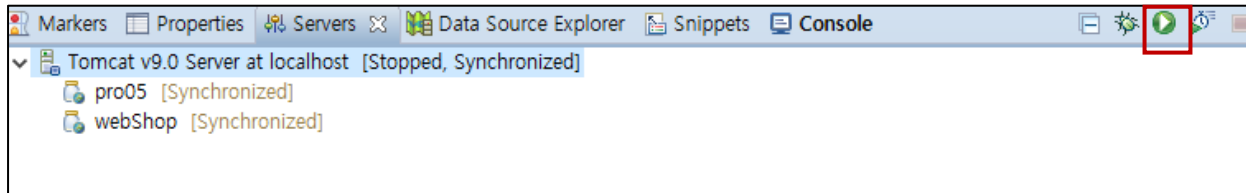
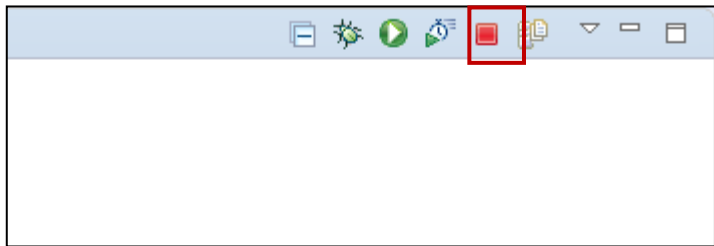
— <servlet-mapping> 태그끼리  
위치시킵니다.

```
</web-app>
```



## 5.4 FirstServlet을 이용한 실습

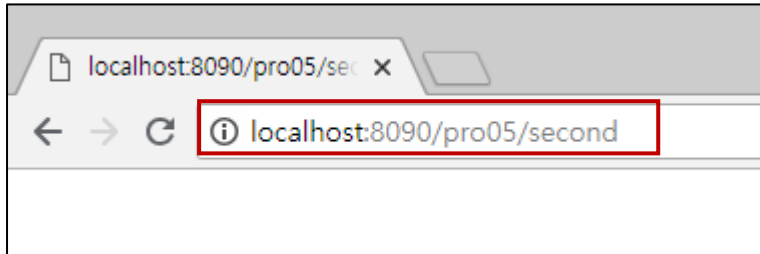
6. 프로젝트의 web.xml 변경 사항을 반영하려면 톰캣을 재실행해야 합니다.  
Servers의 빨간색 버튼을 클릭해 톰캣을 종료한 후 다시 녹색 버튼을 클릭해 톰캣을 실행합니다.





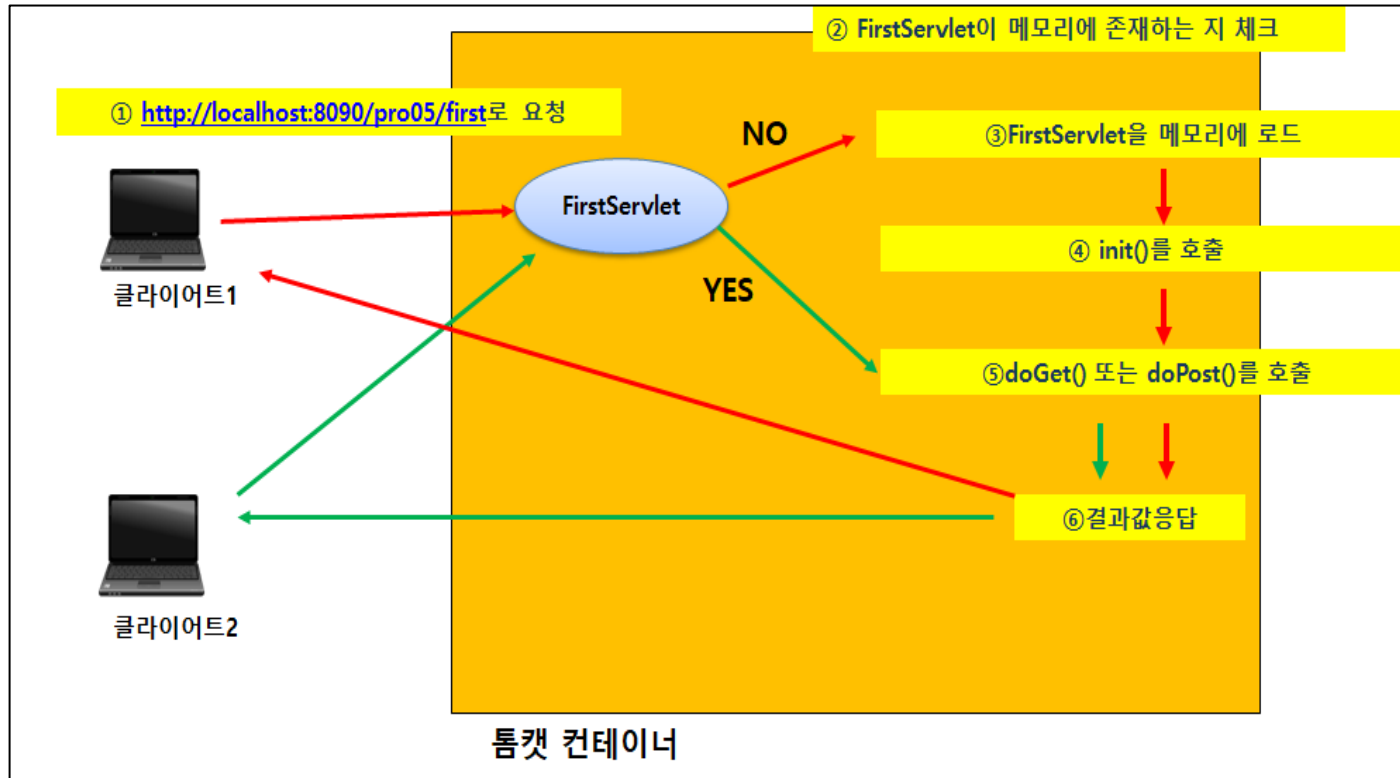
## 5.4 FirstServlet을 이용한 실습

7. 다음은 브라우저에서 /second라는 매핑 이름으로 요청했을 때의 결과입니다. 이번에는 SecondServlet 클래스들의 메서드가 호출되어 메시지를 출력합니다.



```
8월 23, 2018 1:21:21 오후 org.apache.c
정보: Starting ProtocolHandler ["ajp-n
8월 23, 2018 1:21:21 오후 org.apache.c
정보: Server startup in 1572 ms
init 메서드 호출>>>>
doGet 메서드 호출>>>>
```

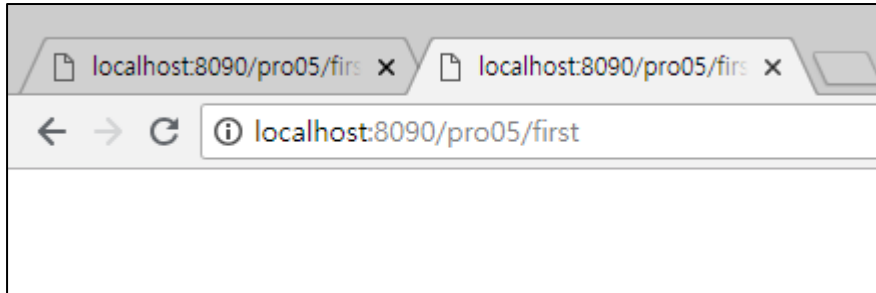
## 5.5 서블릿 동작 과정





## 5.5 서블릿 동작 과정

- 스레드로 기능을 수행하는 서블릿



```
정보: Starting ProtocolHandler ["ajp-nio-  
8월 23, 2018 1:24:41 오후 org.apache.cata  
정보: Server startup in 1544 ms  
init 메서드 호출  
doGet 메서드 호출  
doGet 메서드 호출  
doGet 메서드 호출
```

✓ 브라우저에서 최초 요청 시 한 번만 init() 메서드가 호출되며 재 요청 시 doXXX() 메서드만 호출됨



## 5.6 애너테이션을 이용한 서블릿 매핑

### • 5.6.1 애너테이션을 이용한 서블릿 매핑

- web.xml에 여러 서블릿 매핑 설정 시 복잡해짐.
- 따라서 서블릿 클래스에 직접 애너테이션으로 서블릿명을 설정하면 가독성이 좋아짐.
- @WebServlet을 이용해서 서블릿 매핑을 구현함.

#### 코드 5-6 @WebServlet 사용 방법

(서블릿 클래스 위에 선언)

```
@WebServlet("/서블릿매핑이름");  
</>
```

<코드> 애너테이션을 이용한 서블릿 매핑 예

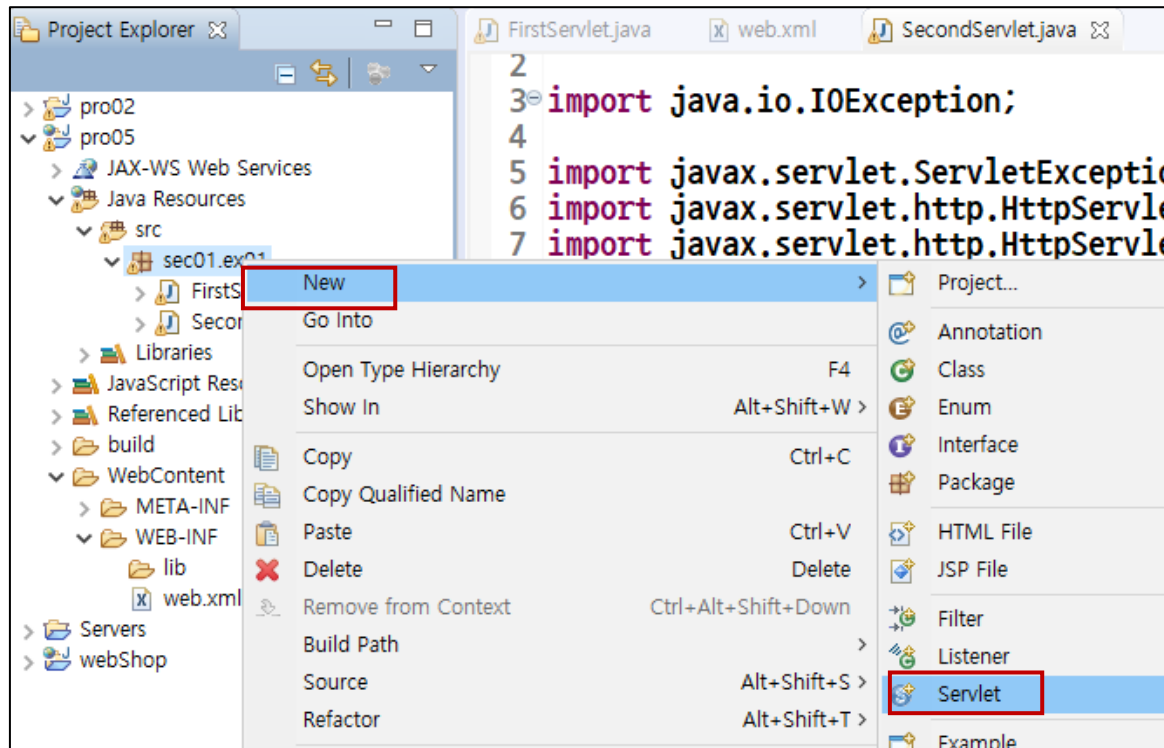
```
@WebServlet("/third")
```

```
public class ThirdServlet extends HttpServlet {  
    ...  
}
```

애너테이션을 이용해서 만드는 서블릿 클래스는 반드시 HttpServlet을 상속받아야 합니다.

## 5.6 애너테이션을 이용한 서블릿 매핑

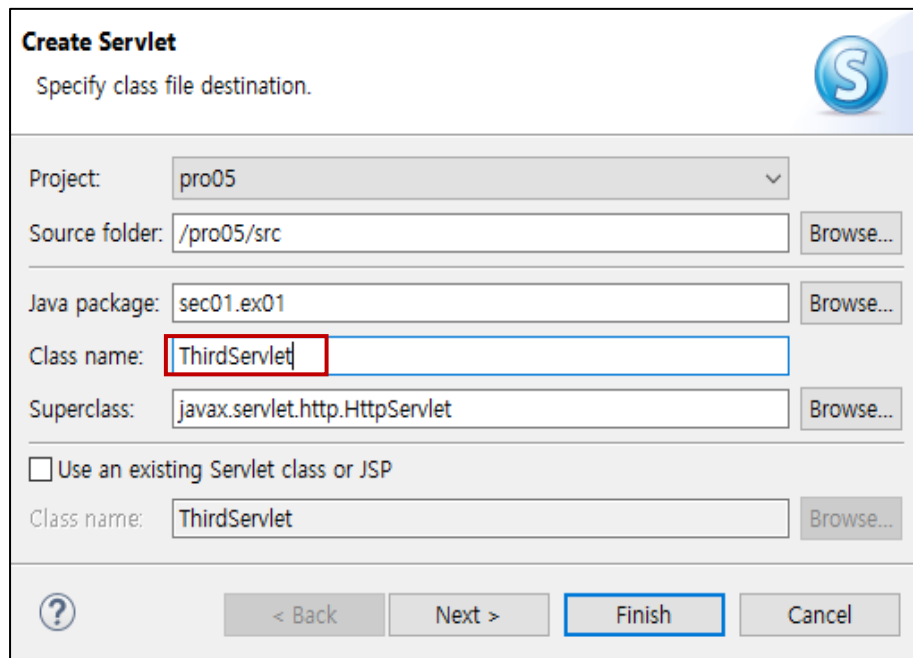
1. sec01.ex01 패키지를 선택하고 마우스 오른쪽 버튼을 클릭한 후 New > Servlet을 선택합니다.







## 5.6 애너테이션을 이용한 서블릿 매핑


2. 클래스 이름으로 ThirdServlet을 입력하고 Next를 클릭합니다.




**Create Servlet** 


Specify class file destination.

Project:  


Source folder:  


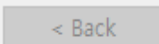
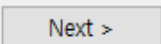
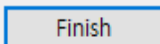
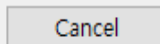
Java package:  

Class name:

Superclass:  

☐ Use an existing Servlet class or JSP

Class name:  





## 5.6 애너테이션을 이용한 서블릿 매핑

3. 우선 기본 URL mapping 이름을 선택한 후 매핑 이름을 수정하기 위해 Edit...를 클릭합니다.

**Create Servlet**  
Enter servlet deployment descriptor specific information.

Name:   
Description:

Initialization parameters:

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

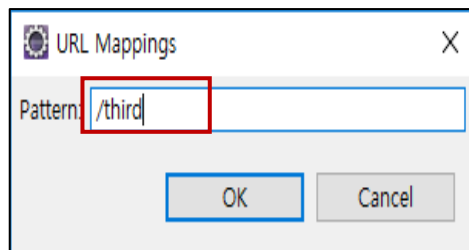
URL mappings:

<input type="text" value="/ThirdServlet"/>
--

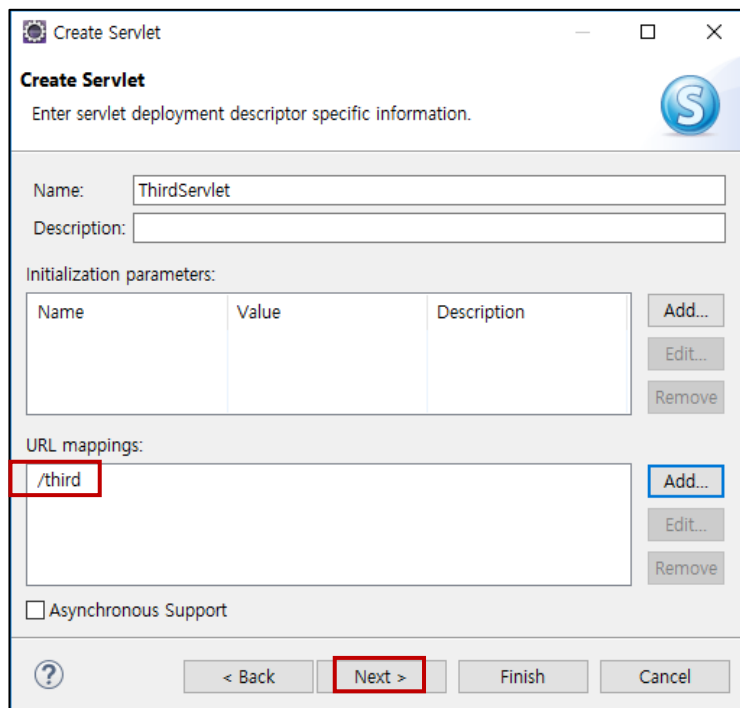
☐ Asynchronous Support

## 5.6 애너테이션을 이용한 서블릿 매핑

4. 서블릿 매핑 이름을 third로 수정하고 OK를 클릭합니다.

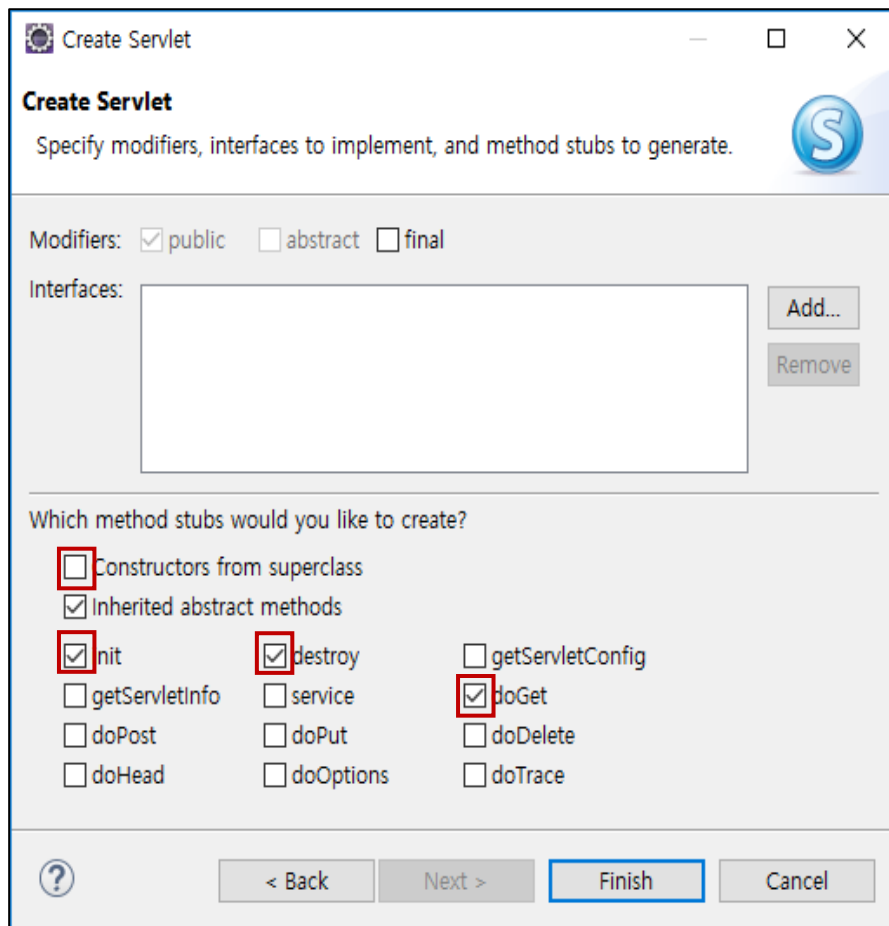


5. 매핑 이름이 수정된 것을 확인한 후 Next를 클릭합니다.



## 5.6 애너테이션을 이용한 서블릿 매핑

6. Constructors from superclass 옵션 체크박스의 체크를 해제한 후 오버라이딩할 생명주기 메서드를 추가합니다. 기본값으로 설정된 상태에서 init과 destroy, doGet에 체크하고 Finish를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. The title bar says 'Create Servlet'. Below the title bar, there's a subtitle 'Create Servlet' and a description 'Specify modifiers, interfaces to implement, and method stubs to generate.' with a blue 'S' icon. The 'Modifiers' section has checkboxes for 'public' (checked), 'abstract' (unchecked), and 'final' (unchecked). The 'Interfaces' section has an empty text box and 'Add...' and 'Remove' buttons. The 'Which method stubs would you like to create?' section has a list of checkboxes: 'Constructors from superclass' (unchecked, highlighted with a red box), 'Inherited abstract methods' (checked), 'init' (checked, highlighted with a red box), 'destroy' (checked, highlighted with a red box), 'doGet' (checked, highlighted with a red box), 'getServletInfo' (unchecked), 'service' (unchecked), 'doPost' (unchecked), 'doPut' (unchecked), 'doHead' (unchecked), 'doOptions' (unchecked), 'getServletConfig' (unchecked), 'doDelete' (unchecked), and 'doTrace' (unchecked). At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.



## 5.6 애너테이션을 이용한 서블릿 매핑

7. 애너테이션에 수정한 매핑 이름이 추가된 것을 확인할 수 있습니다.

```
ThirdServlet.java x
1 package sec01.ex01;
2
3 import java.io.IOException;
10
11 /**
12  * Servlet implementation class ThirdServlet
13  */
14 @WebServlet("/third")
15 public class ThirdServlet extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     /**
19      * @see Servlet#init(ServletConfig)
20      */
21     public void init(ServletConfig config) throws ServletException {
22         // TODO Auto-generated method stub
23     }
24
25     /**
26      * @see Servlet#destroy()
27      */
28     public void destroy() {
29         // TODO Auto-generated method stub
30     }
31 }
```



## 5.6 애너테이션을 이용한 서블릿 매핑

코드 5-7 pro05/src/sec01/ex01/ThirdServlet.java

```
package sec04.ex01;

...

/**
 * Servlet implementation class ThirdServlet
 */
@WebServlet("/third")
public class ThirdServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see Servlet#init(ServletConfig)
     */
    public void init(ServletConfig config) throws ServletException {
        System.out.println("ThirdServlet init 메서드 호출");
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     *      response)
     */
}
```

서블릿 클래스 직렬화를 위해 이 클래스에서 자동으로 지정한 상수입니다. 이 책에선 사용하지 않으므로 삭제합니다.



## 5.6 애너테이션을 이용한 서블릿 매핑

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    System.out.println("ThirdServlet doGet 메서드 호출");
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    System.out.println("ThirdServlet destroy 메서드 호출");
}
}
```

---



## 5.6 애너테이션을 이용한 서블릿 매핑

9. 이클립스에서 애너테이션을 이용하여 서블릿 매핑을 설정했으니 톰캣을 중지했다가 재실행한 다음 웹 브라우저에서 서블릿 매핑 이름으로 요청해 보겠습니다.

- <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
```



## 5.6 애너테이션을 이용한 서블릿 매핑

- 애너테이션으로 설정한 매핑명을 다른 매핑명과 중복되면 안됩니다.

```
ThirdServlet.java  web.xml
3 *import java.io.IOException;
10
11 /**
12  * Servlet implementation class ThirdServlet
13  */
14 @WebServlet("/first")
15 public class ThirdServlet extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     /**
19      * @see Servlet#init(ServletConfig)
20      */
21     public void init(ServletConfig config) throws ServletException {
22         System.out.println("ThirdServlet init 메소드 호출");
23     }
24
25 }
```

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Problems  Console
<terminated> Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (2018. 4. 18. 오후 7:37:26)
경고: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of
4월 18, 2018 7:37:32 오후 org.apache.catalina.core.ContainerBase startInternal
심각: A child container failed during start
java.util.concurrent.ExecutionException: org.apache.catalina.LifecycleException: Failed to start component [StandardEngine[Catalin
at java.base/java.util.concurrent.FutureTask.report(Unknown Source)
at java.base/java.util.concurrent.FutureTask.get(Unknown Source)
at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:949)
at org.apache.catalina.core.StandardHost.startInternal(StandardHost.java:839)
at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1427)
at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1417)
```