

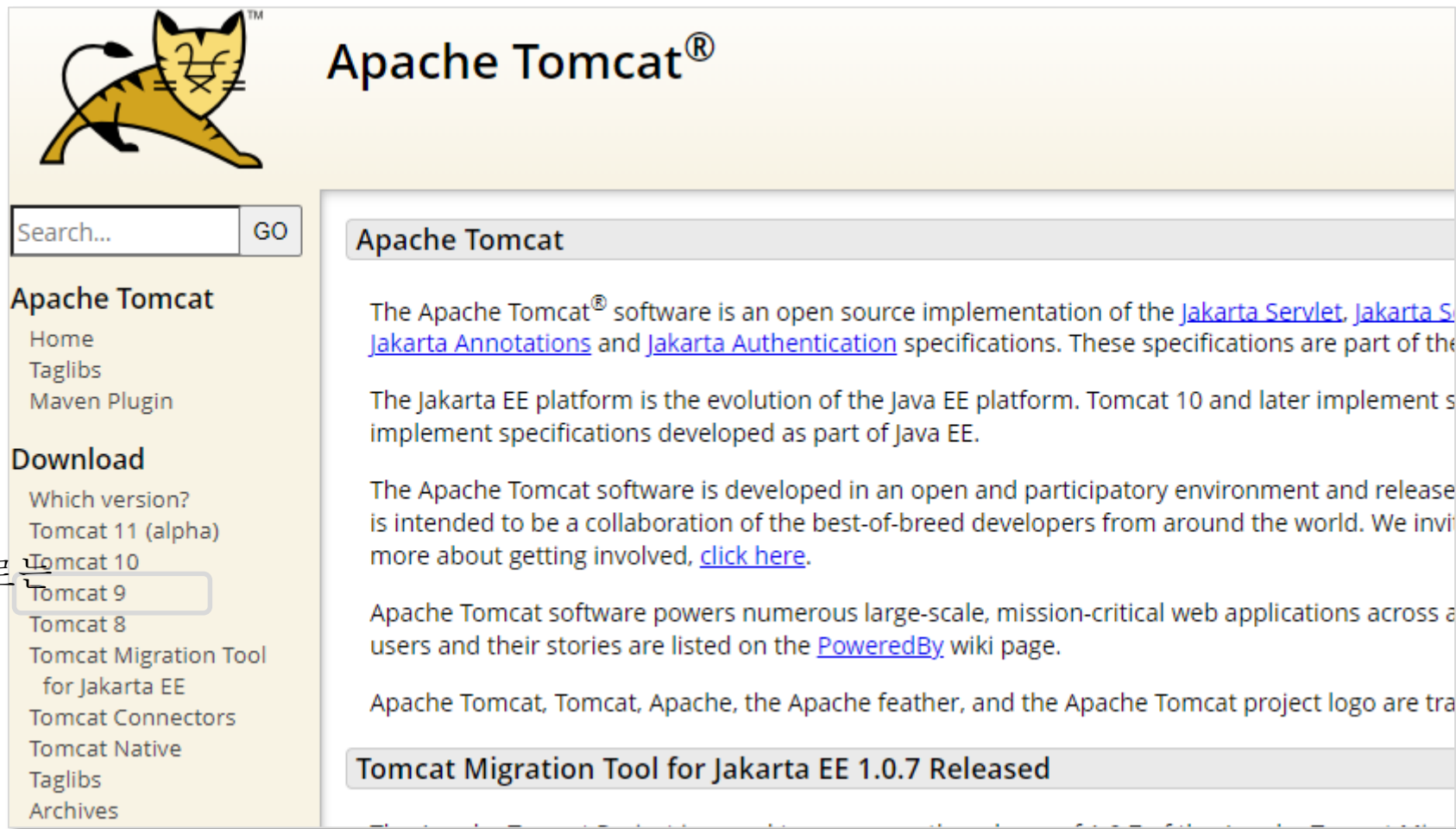
2025년 상반기 K-디지털 트레이닝

웹 어플리케이션 프로젝트

[KB] IT's Your Life

✓ 톰캣 다운로드

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



JAVA EE가
오라클 말고
아파치 재단에서
관리함.

Tomcat 8 버전 밑으로는
JDK 버전과 같음

9부터는 아파치가
한거여서
따로

✓ 톰캣 다운로드

9.0.74

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- 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:
 - [tar.gz](#) ([pgp](#), [sha512](#))
- Deployer:
 - [zip](#) ([pgp](#), [sha512](#))
 - [tar.gz](#) ([pgp](#), [sha512](#))
- Embedded:
 - [tar.gz](#) ([pgp](#), [sha512](#))
 - [zip](#) ([pgp](#), [sha512](#))

✓ 압축 해제

○ C:\wapache-tomcat-9.0.74

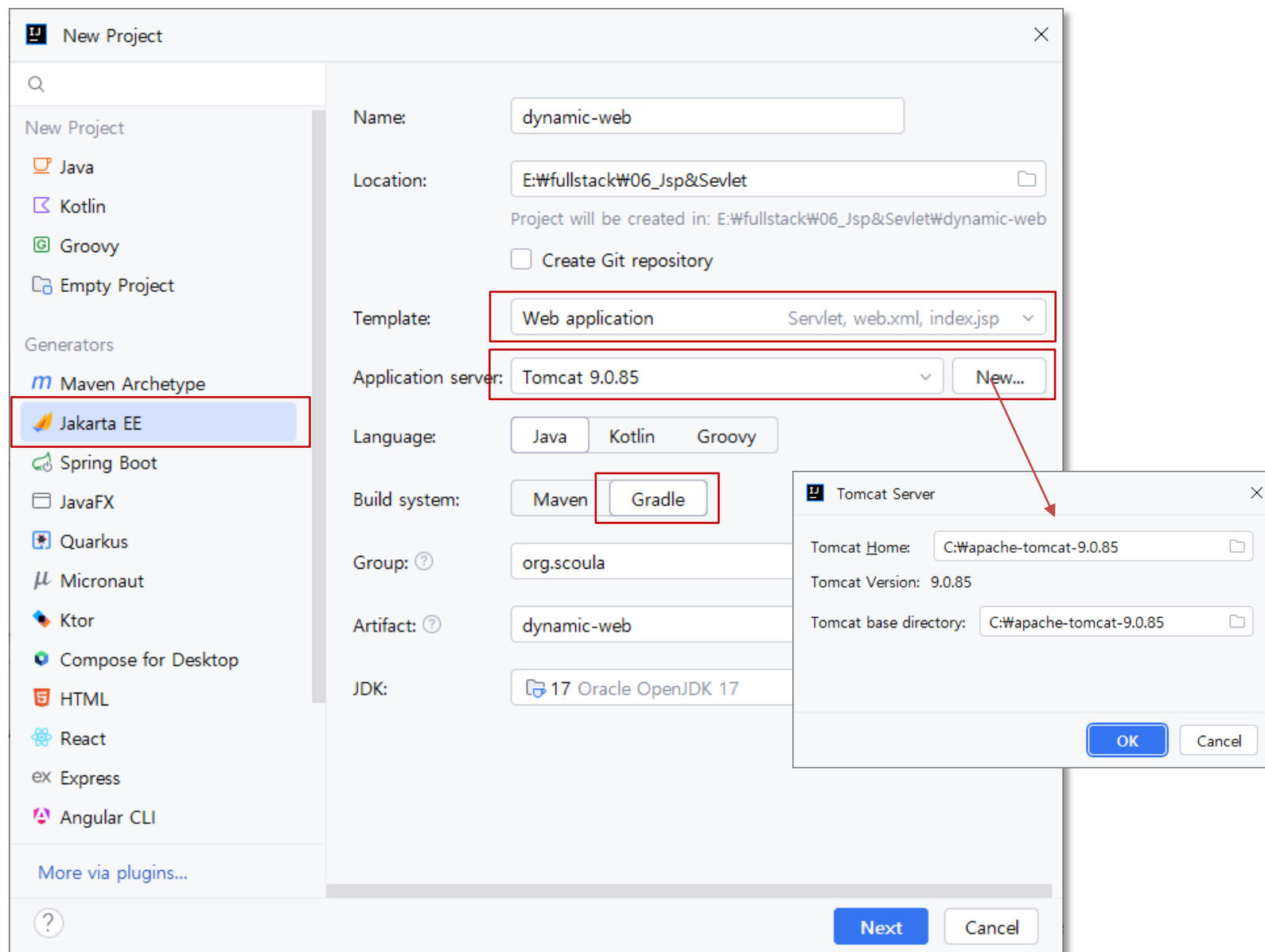
내 PC > 로컬 디스크 (C:) > apache-tomcat-9.0.74 >

이름	수정한 날짜
bin → Tomcat 실행 파일	2023-05-03 오후 2:51
conf — 설정	2023-05-03 오후 2:51
lib → JSP/Servlet 라이브러리	2023-05-03 오후 2:51
logs	2023-04-13 오전 8:10
temp	2023-05-03 오후 2:51
* webapps → 웹 어플리케이션 관리	2023-05-03 오후 2:51
work	2023-04-13 오전 8:10
BUILDING.txt	2023-04-13 오전 8:10
CONTRIBUTING.md	2023-04-13 오전 8:10
LICENSE	2023-04-13 오전 8:10
NOTICE	2023-04-13 오전 8:10
README.md	2023-04-13 오전 8:10
RELEASE-NOTES	2023-04-13 오전 8:10
RUNNING.txt	2023-04-13 오전 8:10

✓ 프로젝트 만들기

○ File > New Project

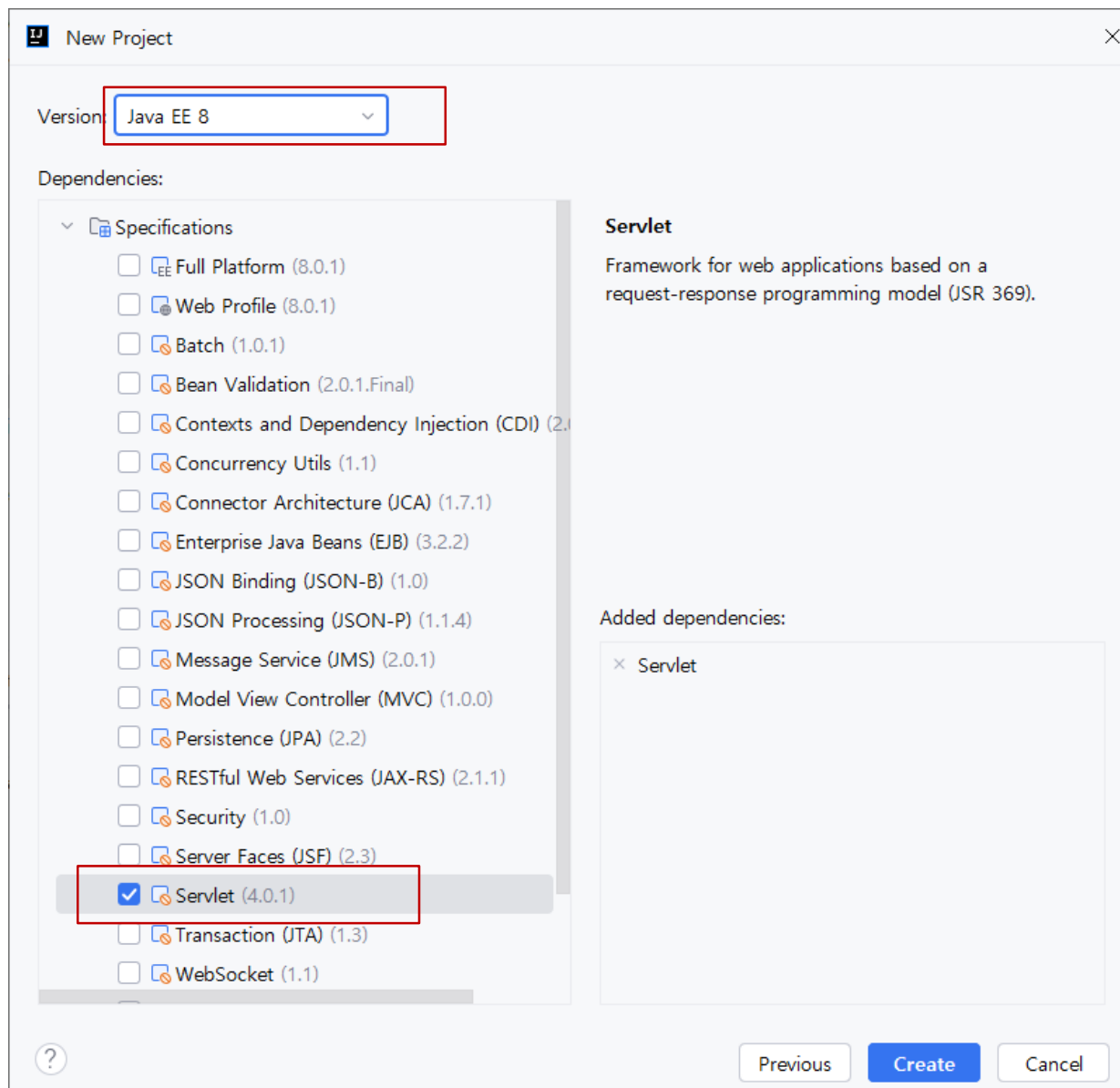
- Name: dynamic-web
- Location: C:\Wfullstack\W06_Jsp&Servlet
- Template: Web application
- Build system: Gradle
- GroupId: org.scoula
- ArtifactId: dynamic-web



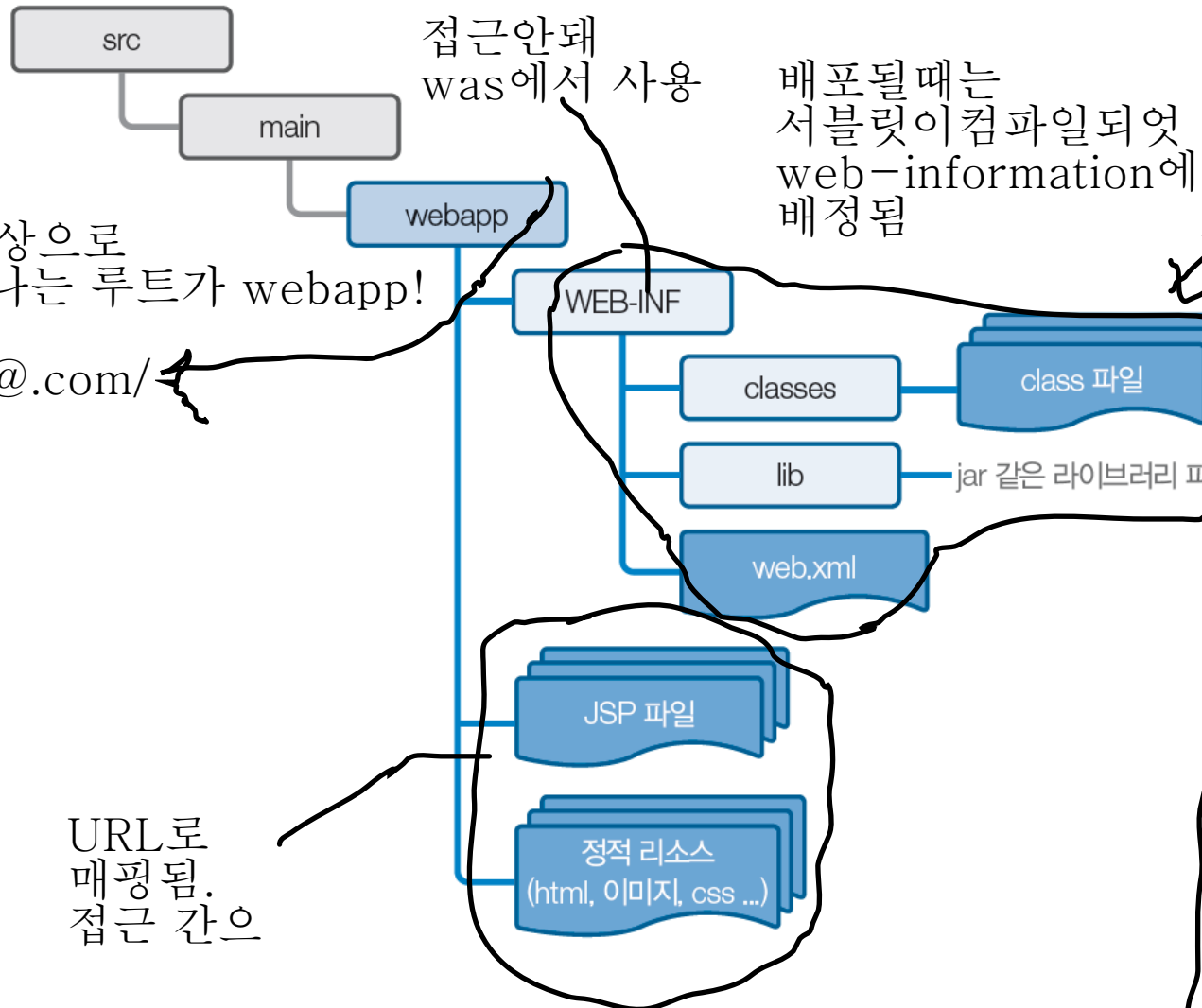
✓ 프로젝트 만들기

도커로
톰캣을 구동하면
불편함.

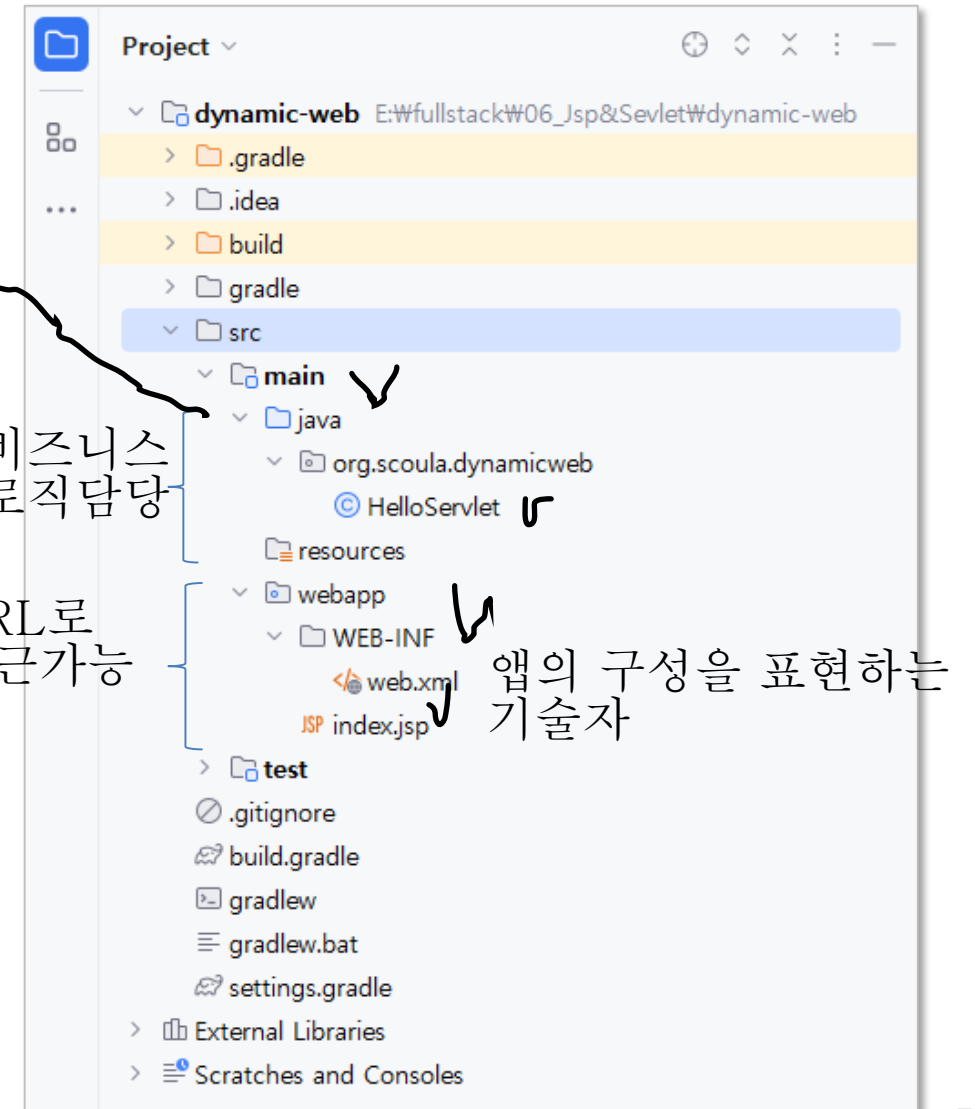
개발용으로
was를
도커에 구성하면
안 좋다. 왜?



프로젝트 기본구조




개발당시 구조



✎ **webapp/WEB-INF/web.xml** 복잡한 구성이므로 yaml보단 xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0">
</web-app>
```

servlet 4.0을 쓰겠다는 말



cf] Servlet 3.1 버전인 경우

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  id="WebApp_ID" version="3.1">
</web-app>
```


build.gradle

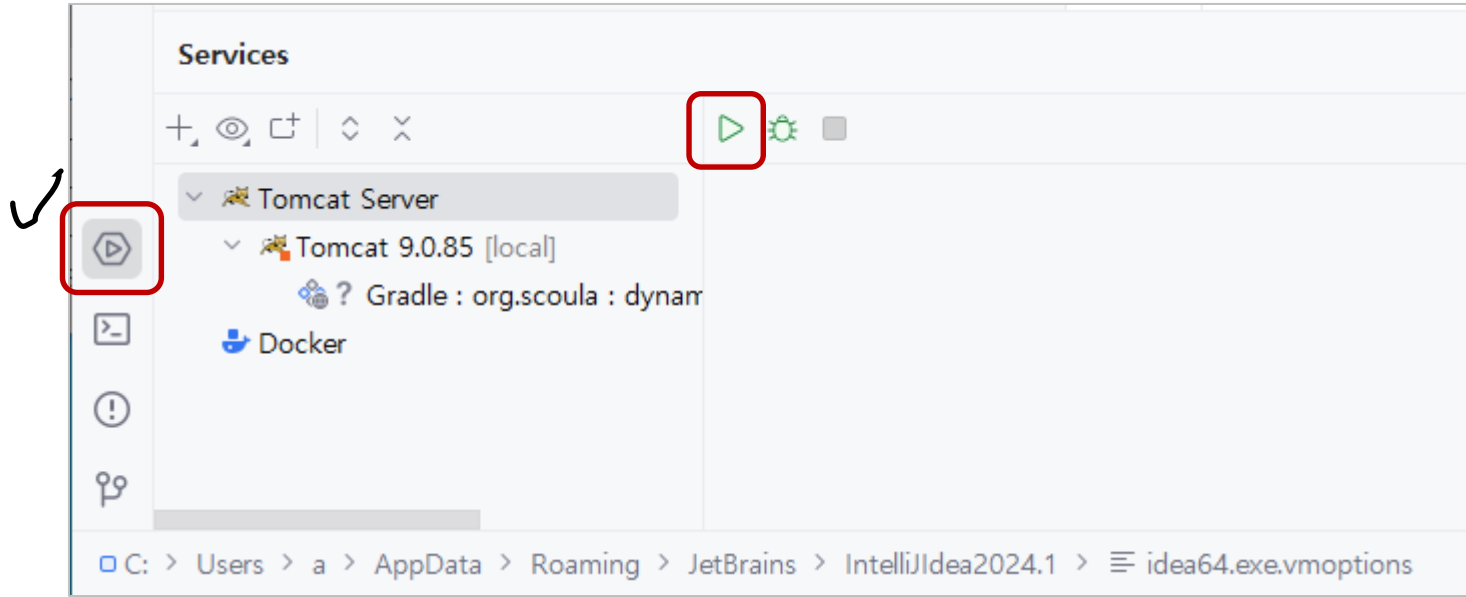
```
plugins {  
    id 'java'  
    id 'war' ✓  
}  
  
group 'org.scoula'  
version '1.0-SNAPSHOT'  
  
repositories {  
    mavenCentral()  
}  
  
ext {  
    junitVersion = '5.9.2'  
}  
  
sourceCompatibility = '1.8' ✓  
targetCompatibility = '1.8' ✓  
  
tasks.withType(JavaCompile) {  
    options.encoding = 'UTF-8'  
}
```

자바 패키지 배포시 jar로 함. java archive. war은 web archive약자.
web app용 archive. 일종의 jar이지만 웹앱 정보가 들어있어서.
war은 Tomcat배포 단위다. app배포본이다.
실제 배포시war을 weapp폴더에 두면된다

17로 변경 가능 자바 호환버전을 바꿀수 있다

```
dependencies {  
    compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
  
    testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")  
    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")  
}  
  
test {  
    useJUnitPlatform()  
}
```

✓ 톰캣의 실행 및 재기동



Context Path(Application name)

http://localhost:8080/Gradle__org_scoula__dynamic_web_1_0_SNAPSHOT_war/

Hello World!

[Hello Servlet](#)

src>main>webapp>index.jsp

webapp/index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
  <title>JSP - Hello World</title>
</head>
<body>
<h1><%= "Hello World!" %></h1>
<br/>
<a href="hello-servlet">Hello Servlet</a>
</body>
</html>
```

비즈니스 로직의 결과가
해당 변수?에 들어가서
JSP가 구성되어
렌더링된 결과

Context Path(Application name)

http://localhost:8080/Gradle__org_scoula__dynamic_web_1_0_SNAPSHOT_war/

index.jsp 렌더링 결과

Hello World!

[Hello Servlet](#)

클릭시 http://localhost:8080/Gradle__org_scoula__dynamic_web_1_0_SNAPSHOT_war/hello-servlet
처럼 뒤에 /hello-servlet이 붙음

📄 HelloServlet.java

```
package org.scoula.dynamicweb;
```

```
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
```

```
@WebServlet(name = "helloServlet", value = "/hello-servlet")
```

```
public class HelloServlet extends HttpServlet {
    private String message;
```

```
    public void init() {
        message = "Hello World!";
    }
```

상대경로이고 (context 경로 제외하며 표시)
해당 위치의 hello-servlet의 서블릿이
담당하는 URL이 무냐가 중요

```
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.setContentType("text/html");
```

그렇다면 post요청시에는
doPost메소드가 호출되겠네

GET 요청이 왔을 때 해당 메소드를 실행시킴. doGet이 실행됨.

```
    // Hello
```

```
    PrintWriter out = resp.getWriter();
```

```
    out.println("<html><body>");
```

```
    out.println("<h1>" + message + "</h1>");
```

```
    out.println("</body></html>");
```

Context Path(Application name)

http://localhost:8080/Gradle__org_scoula__dynamic_web_1.0.SNAPSHOT_war/hello-servlet

Hello World!

hello-servlet X

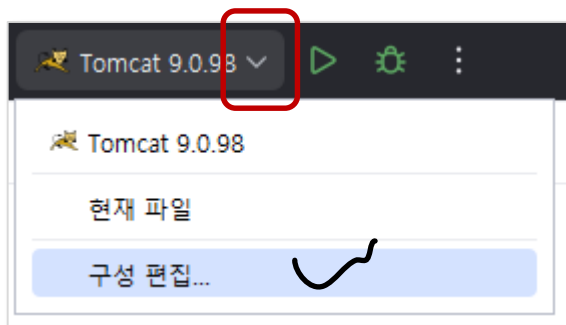
```
1 <html><body>
2 <h1>Hello World!</h1>
3 </body></html>
4
```

```
    public void destroy() {
    }
```

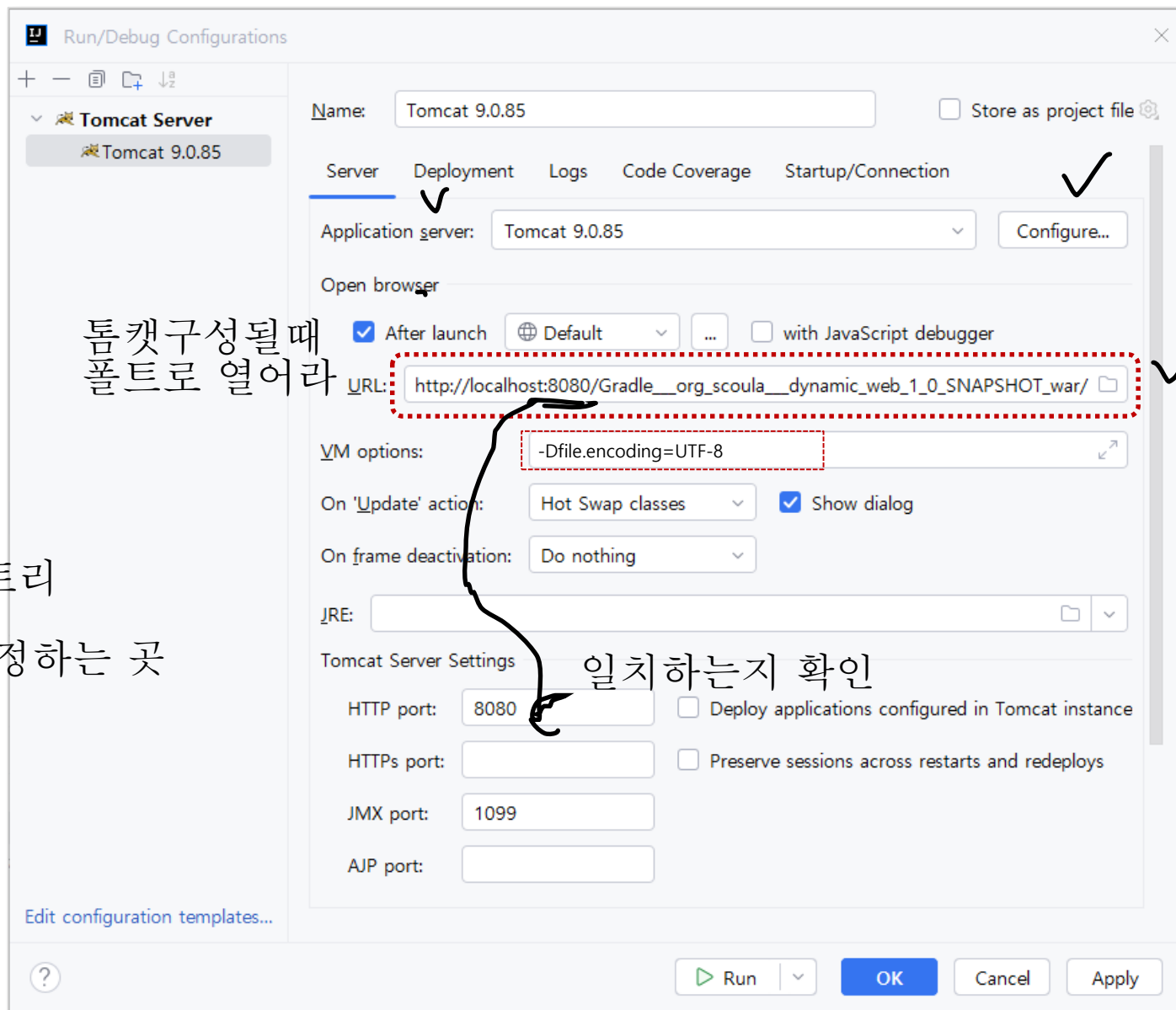
서블릿 종료시

✓ 톰캣 서버 설정

○ context path 확인

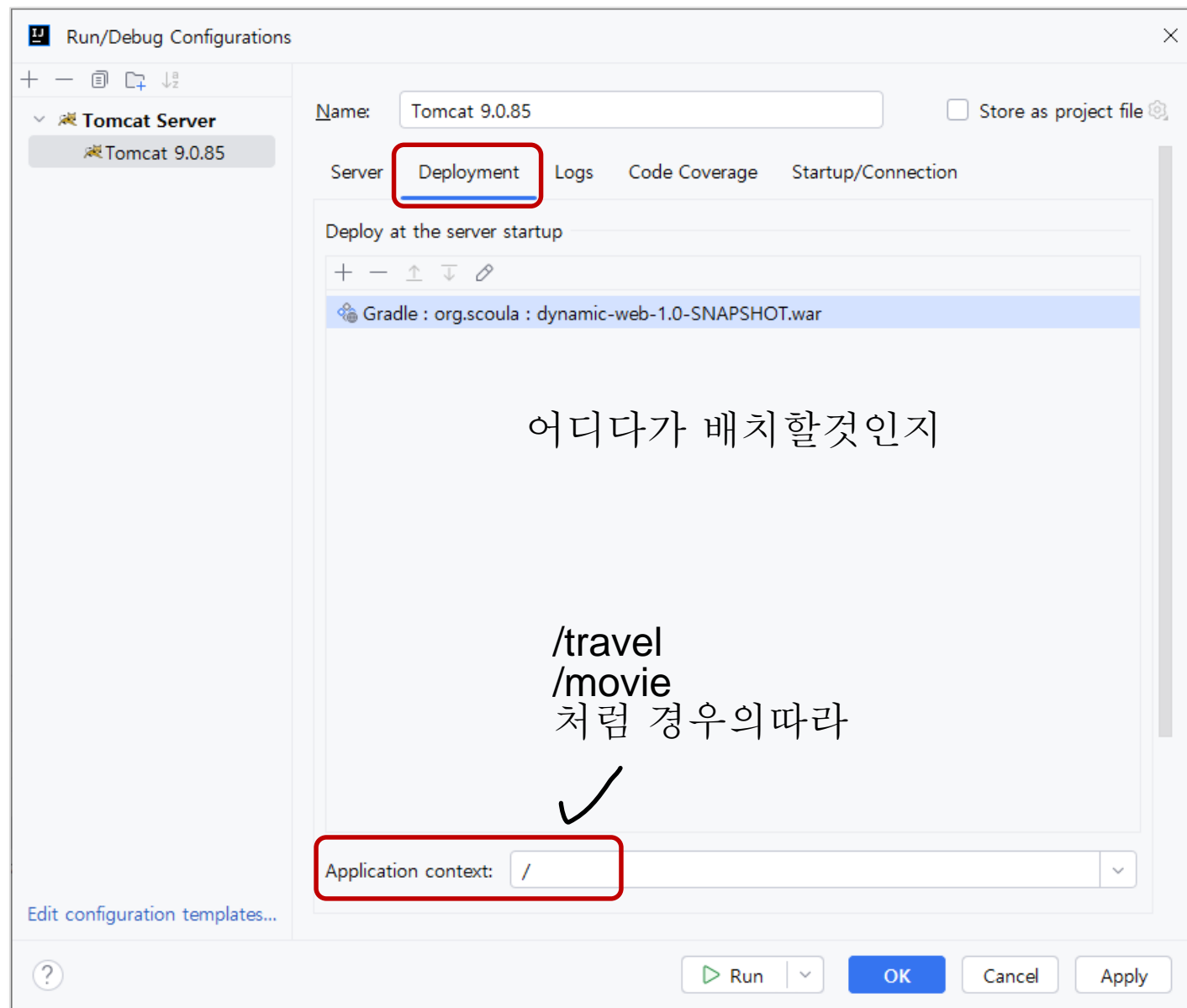


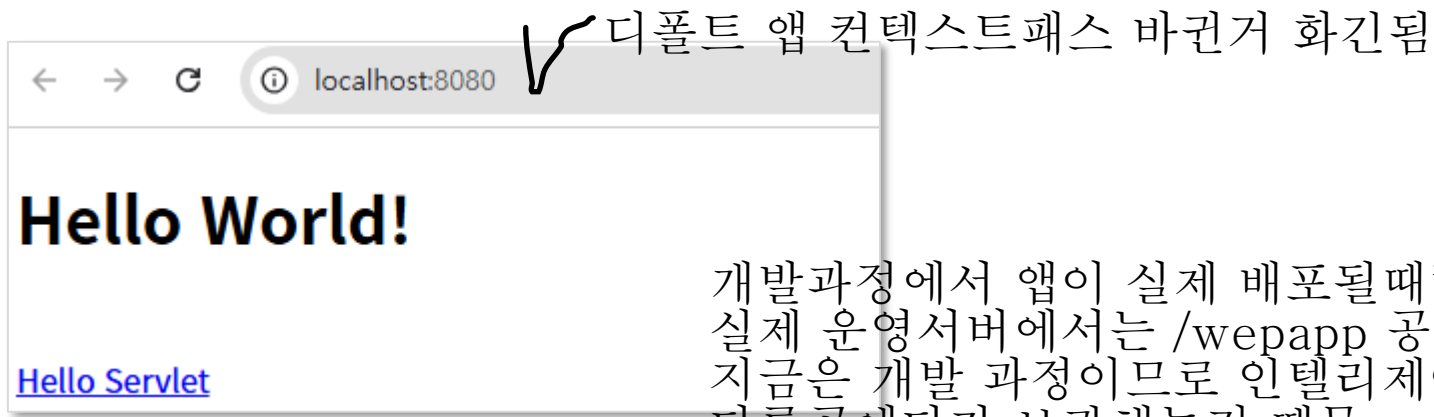
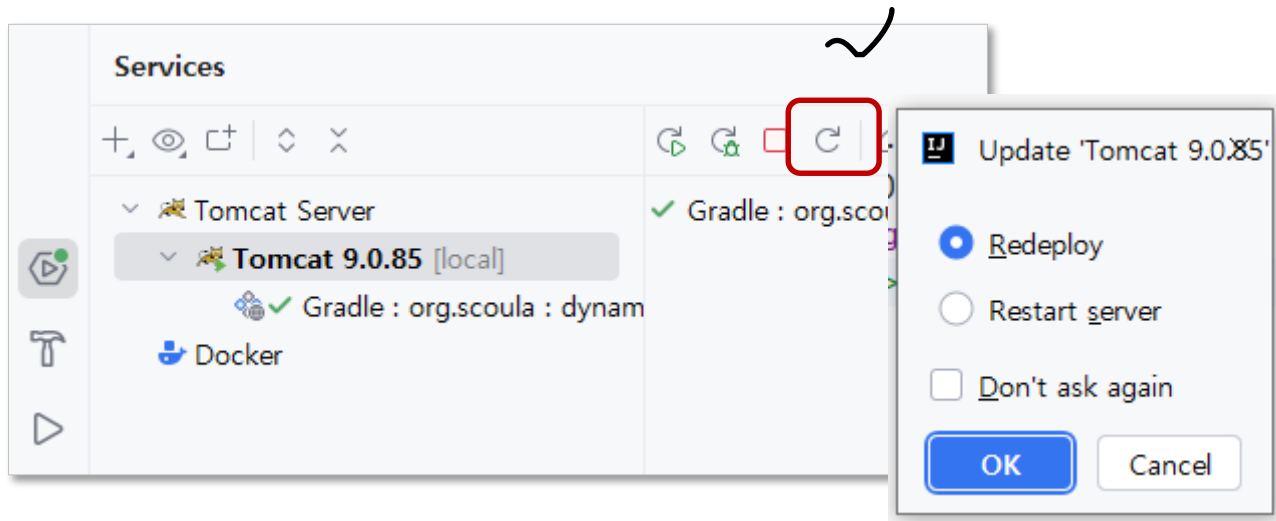
초기 워킹 디렉토리
인자 넘기는 거
뭐로 할 거냐 설정하는 곳



✓ 톰캣 서버 설정

○ context path 변경





개발과정에서 앱이 실제 배포될때랑은 다르게 구성된다.
실제 운영서버에서는 /webapp 공간에 war들이 있지만
지금은 개발 과정이므로 인텔리제이가 자동적으로
다른곳에다가 보관해놓기 때문.
그래서 실제로 배포할때는 수동으로 war들을 /webapp공간에
 옮겨놓아야함