

배포하기

▣ 배포전 확인사항

◆ 톰캣 버전 확인

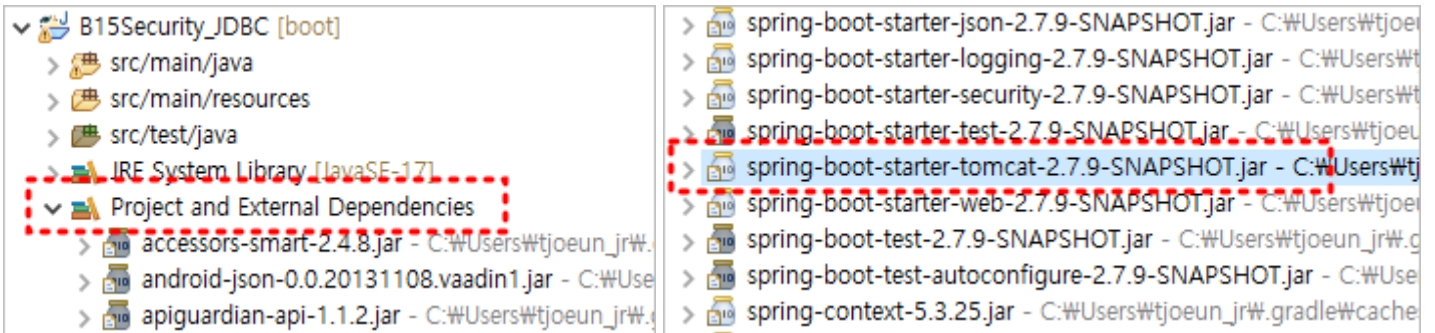
build.gradle 파일을 열어 의존설정된 부분을 확인한다.

프로젝트 생성시 Spring Web을 체크하면 추가되는 항목이다.

```
18 dependencies {
19     implementation 'org.springframework.boot:spring-boot-starter-web'
20     providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
21     testImplementation 'org.springframework.boot:spring-boot-starter-test'
22     //JSP 사용설정
23     implementation 'javax.servlet:jstl'
24     implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
25     //라이브러리 수동 의존설정
26     implementation fileTree(dir: 'libs', include: ['*.jar'])
27 }
```

버전 확인을 하려면 외부 라이브러리 목록을 봐야한다.

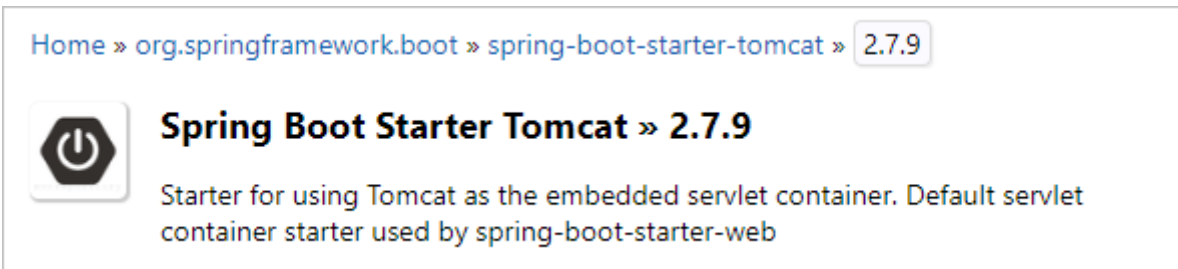
우리 프로젝트에서 사용된 톰캣 라이브러리 버전은 2.7.9 인것을 알 수 있다.







그럼 해당 라이브러리가 사용하는 톰캣 버전은 메이븐 저장소[링크]에서 확인할 수 있다.

Spring Boot Starter Tomcat [바로가기]

해당 페이지에서 2.7.9를 클릭한다.



Compile Dependencies 항목에서 톰캣 버전을 다음과 같이 확인할 수 있다.

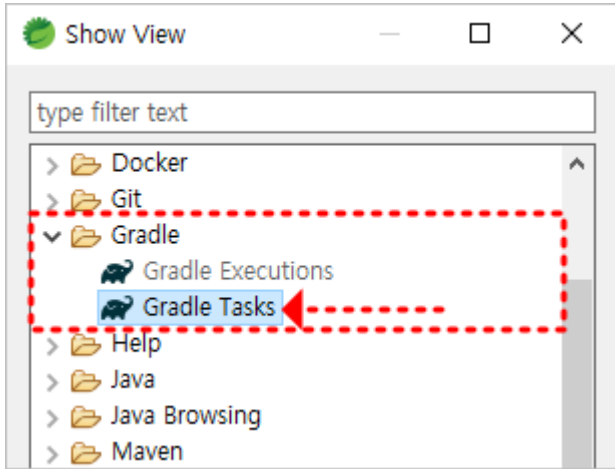
Compile Dependencies (4)				
Category/License		Group / Artifact		Version Updates
Annotation Lib EPL 2.0 GPL		jakarta.annotation » jakarta.annotation-api		1.3.5 2.1.1
Web Server Apache 2.0		org.apache.tomcat.embed » tomcat-embed-core		9.0.71 10.1.7
Apache 2.0		org.apache.tomcat.embed » tomcat-embed-el		9.0.71 10.1.7
Apache 2.0		org.apache.tomcat.embed » tomcat-embed-websocket		9.0.71 10.1.7

■ 단독 실행이 가능한 War파일로 배포

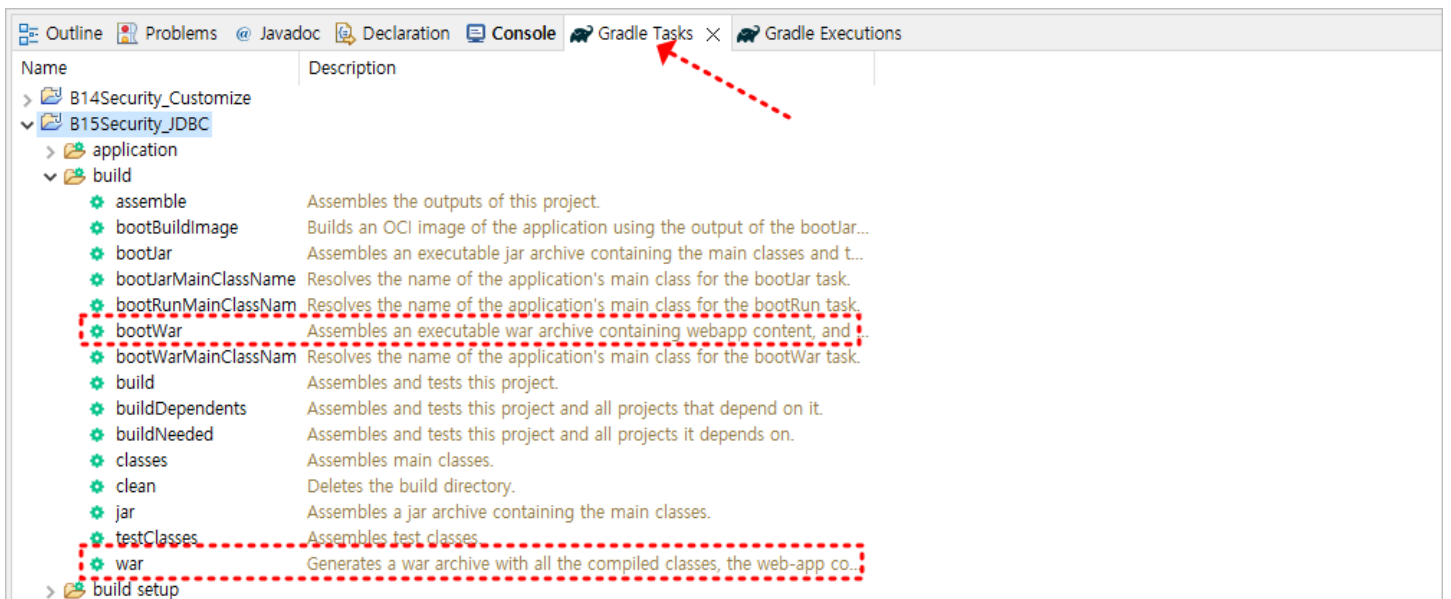
배포는 기존에 작성했던 파일업로드 프로젝트를 사용한다.

STS에서 Window ⇒ ShowView ⇒ Other를 클릭한다.

여기에서 다음 항목을 선택한 후 오픈한다.



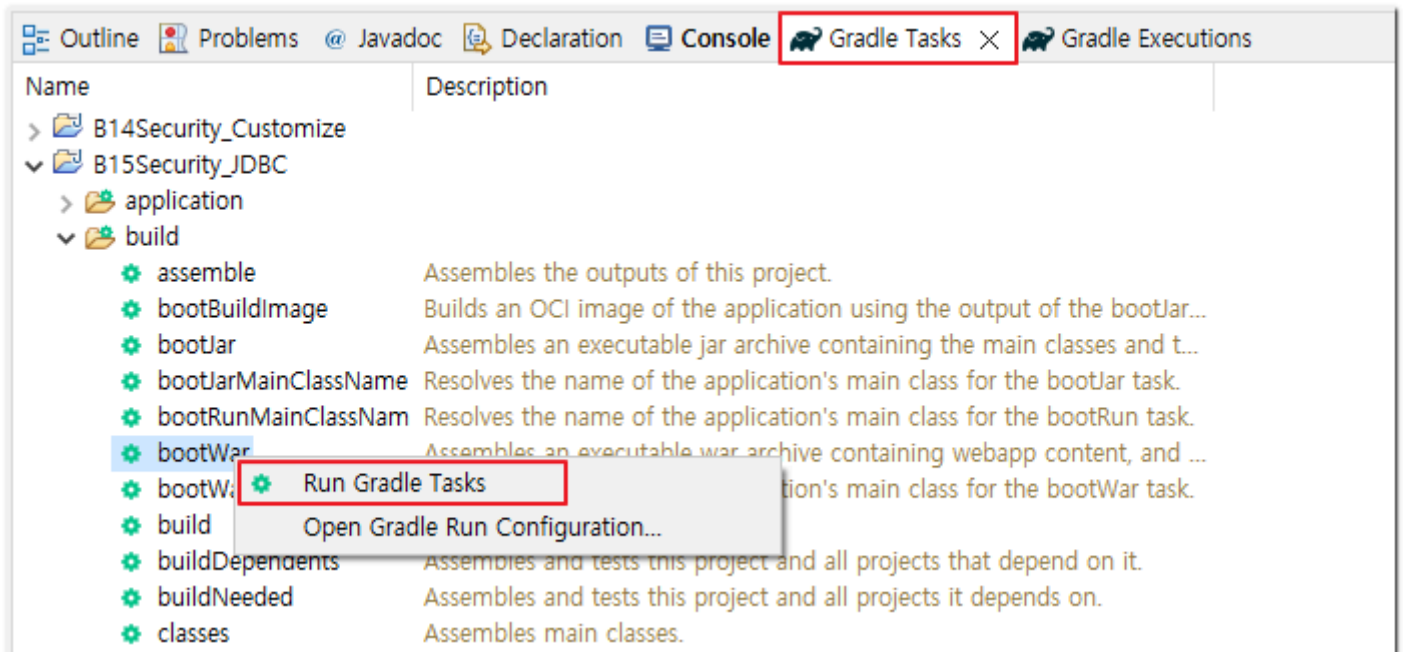
Gradle Tasks 창에서 배포할 프로젝트를 펼친다.



build 하위에는 다양한 빌드를 할 수 있는 항목들이 있는데 우리는 이중에 2가지만 해볼것이다.

- bootWar : 단독으로 실행 가능한 war 파일로 배포한다.
- war : 웹 애플리케이션 서버에 배포하기 위한 war 파일을 생성한다.

단독 실행이 가능한 war 파일을 먼저 만들어보자. bootWar를 우클릭 ⇒ Run Gradle Tasks를 클릭한다.



Gradle Executions 를 보면 그레이들의 태스크가 정상적으로 실행된것을 볼 수 있다.

The screenshot shows the Eclipse IDE's Gradle Executions view. The 'Run main tasks' section is highlighted with a red dashed box. The view displays a table of operations and their durations. The operations are listed under the 'Run build' folder, including 'Build finished for file system watching', 'Configure build', 'Calculate build tree task graph', 'Load build', 'Build started for file system watching', 'Run main tasks', 'Run tasks', and various sub-tasks like ':bootWarMainClassName', ':processResources UP-TO-DATE', 'Resolve mutations for task :processResources', ':bootWar', 'Resolve mutations for task :compileJava', 'Resolve mutations for task :bootWarMainClassName', 'Resolve mutations for task :bootWar', ':classes UP-TO-DATE', 'Resolve mutations for task :classes', and ':compileJava UP-TO-DATE'.

Operation	Duration
Run build	1.245 s
Build finished for file system watching	0.000 s
Configure build	0.078 s
Calculate build tree task graph	0.017 s
Load build	0.037 s
Build started for file system watching	0.001 s
Run main tasks	1.091 s
Run tasks	1.090 s
:bootWarMainClassName	0.191 s
:processResources UP-TO-DATE	0.007 s
Resolve mutations for task :processResources	0.000 s
:bootWar	0.492 s
Resolve mutations for task :compileJava	0.001 s
Resolve mutations for task :bootWarMainClassName	0.001 s
Resolve mutations for task :bootWar	0.001 s
:classes UP-TO-DATE	0.000 s
Resolve mutations for task :classes	0.001 s
:compileJava UP-TO-DATE	0.381 s

콘솔에서는 다음과 같은 로그가 출력된다.

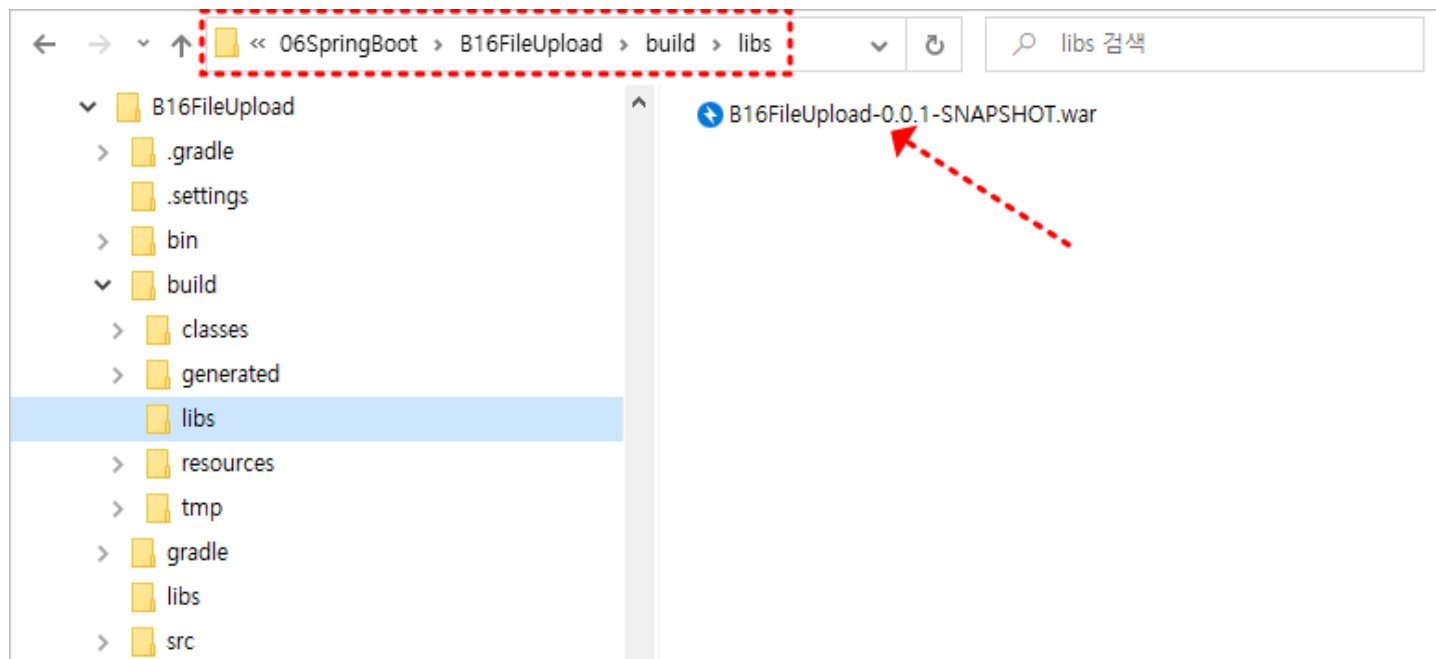
The screenshot shows the Eclipse IDE's Console view. The console output displays the Gradle build process details for the project 'B15Security_JDBC'. The output includes the working directory, Gradle user home, Gradle distribution, Gradle version, Java home, JVM arguments, program arguments, and build scans enabled status.

```

B15Security_JDBC - bootWar [Gradle Project] bootWar in C:\06SpringBoot\B15Security_JDBC (2023. 3. 8. 오후 12:45:43)
Working Directory: C:\06SpringBoot\B15Security_JDBC
Gradle user home: C:\Users\tjoeun_jr\gradle
Gradle Distribution: Gradle wrapper from target build
Gradle Version: 7.6
Java Home: C:\01Developkits\sts-4.17.2.RELEASE\plugins\org.eclipse.justj.openjdk
JVM Arguments: None
Program Arguments: None
Build Scans Enabled: false
  
```

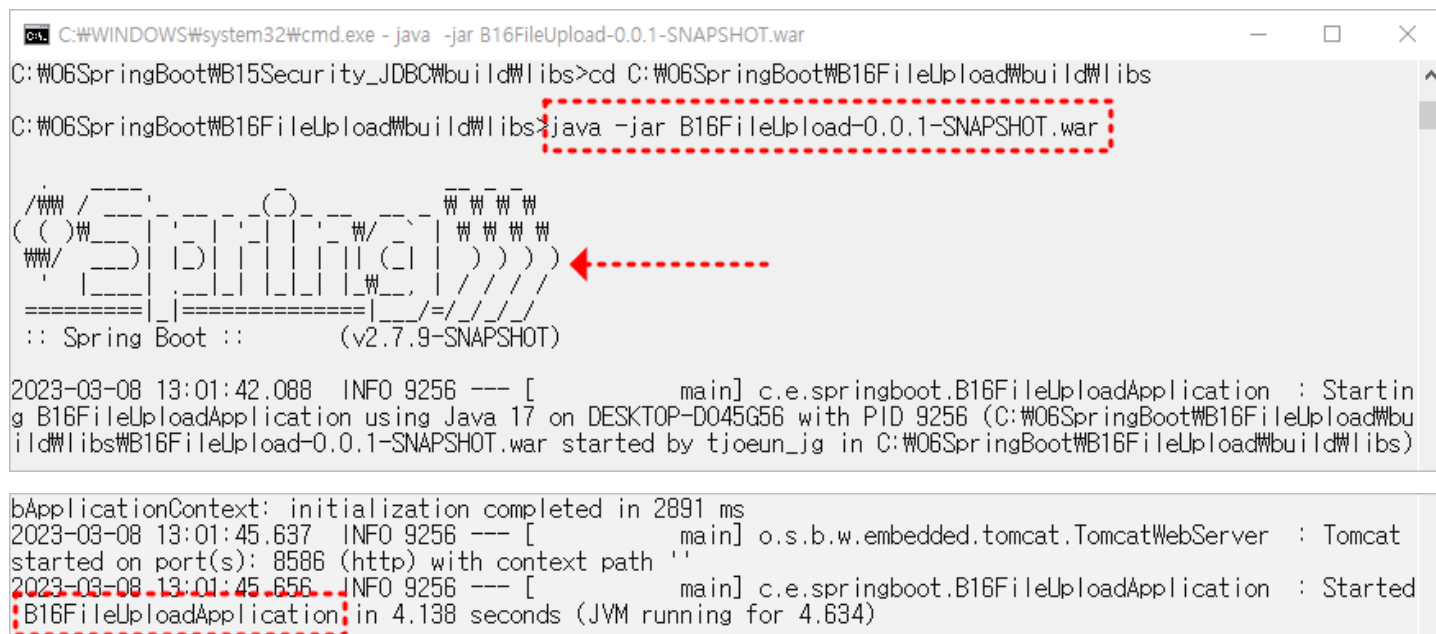
이제 윈도우 탐색기에서 배포된 war 파일을 확인해본다.

프로젝트 하위의 build/libs 폴더에 단독 실행이 가능한 war 파일이 생성된다.



실행을 위해 명령프롬프트(cmd)창을 열어 위 경로로 이동한 후 다음 명령을 실행한다.

C:\디렉토리> java -jar 프로젝트명.war



위와 같이 스프링부트 로고가 뜨면서 웹 애플리케이션이 실행된다.

실행 확인을 위해 웹브라우저에서 접속한다.

URL은 평소와 동일하게 <http://localhost:8586/> 로 하면된다.

파일 업로드 폼이 잘 뜨는걸 볼 수 있다.

← → ↺ 🏠 📄 localhost:8586/fileUpload.do

파일 업로드 폼

제목 :

파일1 :

파일 선택

선택된 파일 없음

파일2 :

파일 선택

선택된 파일 없음

전송1(JSON결과)

전송2(화면이동)

기본적인 실행에는 문제가 없지만, 파일을 업로드 해보면 에러가 발생한다.

War파일도 Zip파일과 같은 압축파일이다. 그러므로 우리가 만든것들을 실행하는것은 문제가 되지 않는다.

하지만 새로운 파일을 업로드 하면 압축파일 내부에 새로운 파일을 추가해야 하는 형태가 되기때문에
에러가 발생되는 것이다.

■ 톰캣(외부WAS) 배포용 War 파일 생성

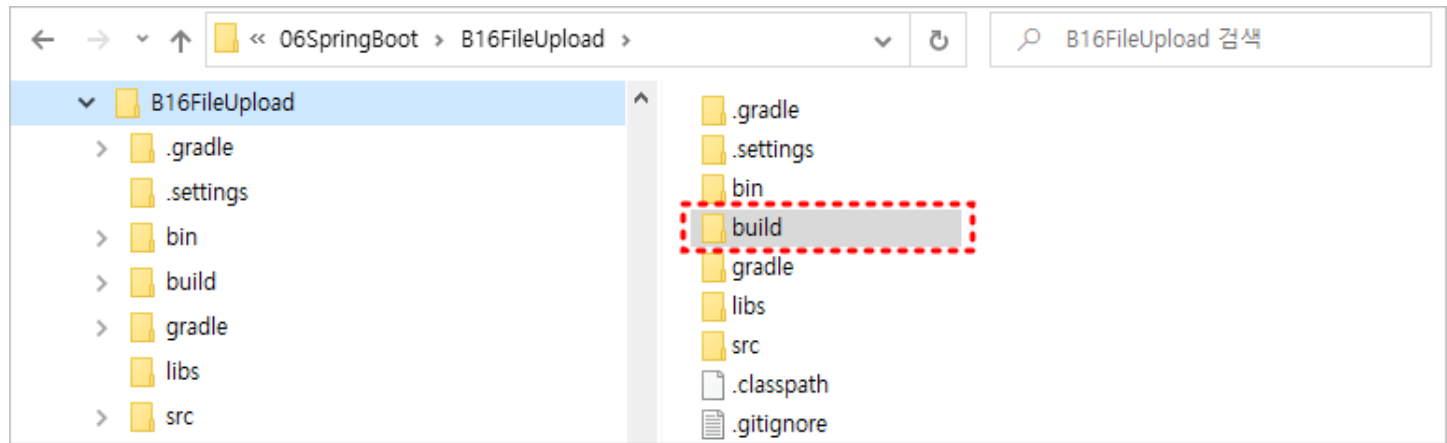
이번에는 톰캣을 내장하지 않은 외부 WAS 배포용으로 생성해 보겠다.

먼저 build.gradle에 아래의 내용을 추가한다.

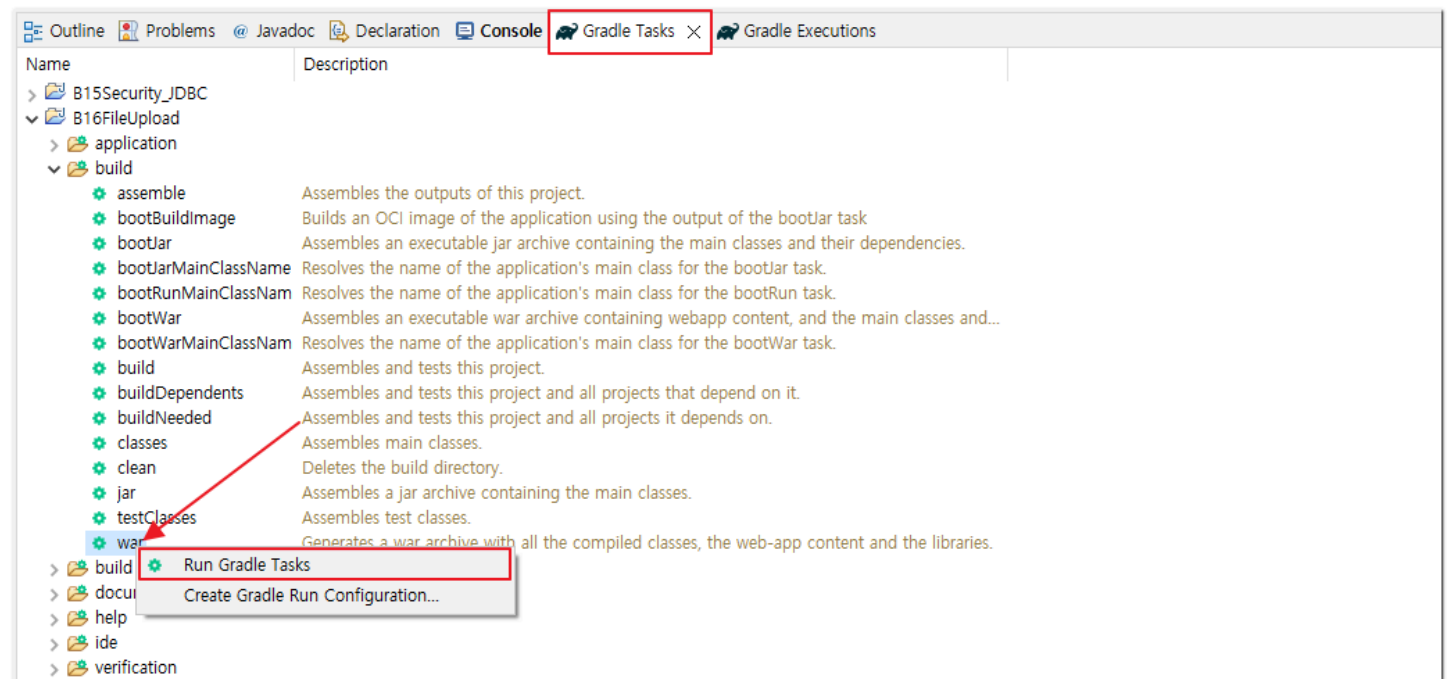
```
28
29 tasks.named('test') {
30     useJUnitPlatform()
31 }
32
33 bootWar.enabled = false
34 war.enabled = true
35
```

스프링 부트는 bootWar가 디폴트 이므로 설정을 이와같이 변경한 후 빌드해야 한다.

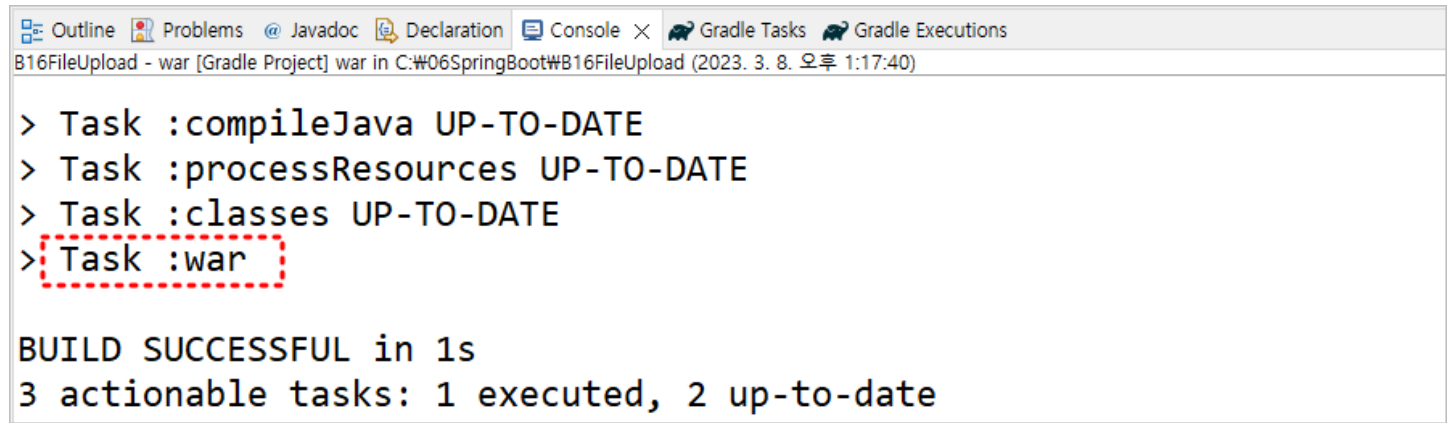
윈도우 탐색기에서 기존 빌드시 생성된 build 폴더를 삭제한다.



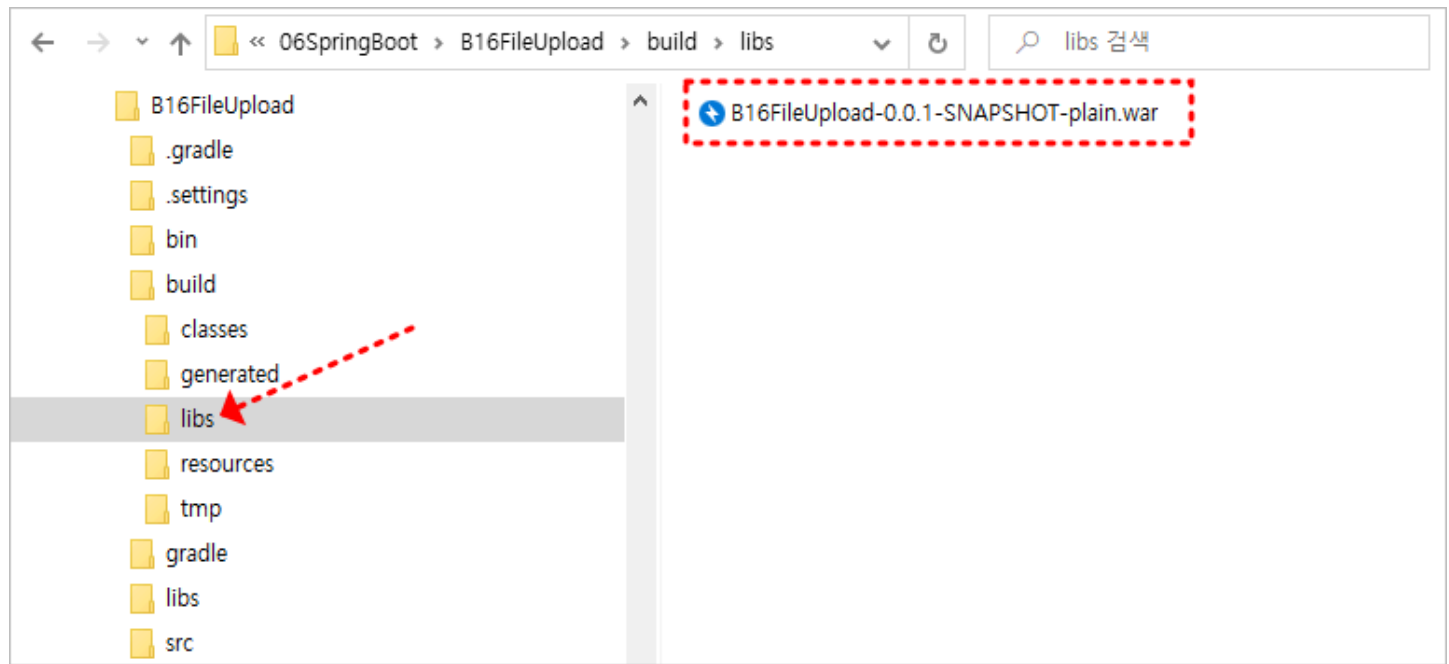
Gradle Tasks 에서 war를 선택한 후 Run Gradle Tasks를 클릭한다.



작업이 끝난 후 콘솔을 확인해보면 다음과 같은 출력메세지를 볼 수 있다.

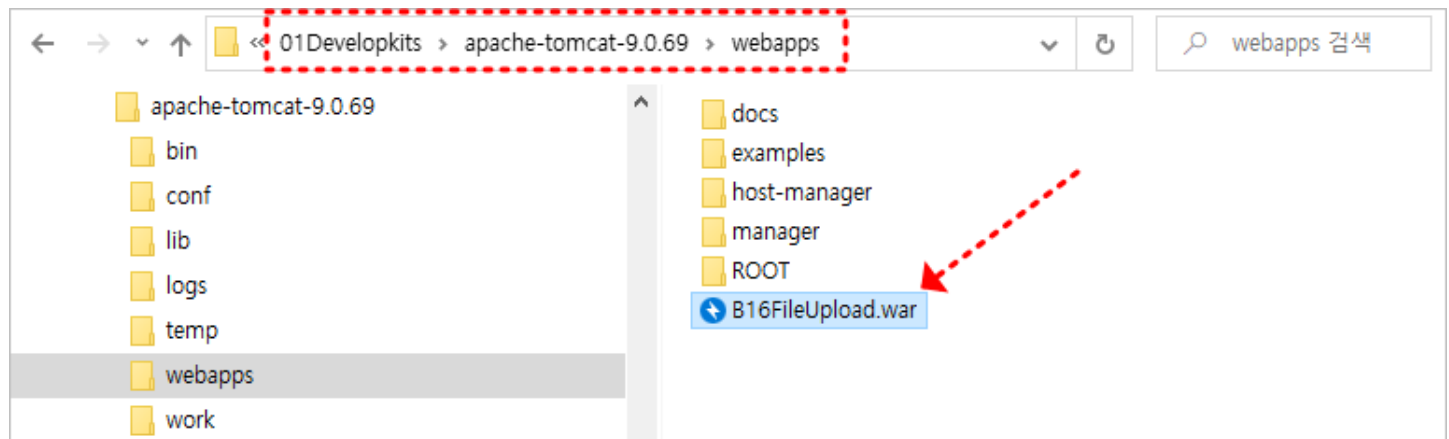


윈도우 탐색기에서 생성된 war 파일을 확인한다.



war 파일을 톰캣 폴더 하위의 webapps로 이동시킨다.

배포된 파일은 파일명이 너무 길기때문에 컨텍스트 루트명과 동일하게 수정한다.



명령프롬프트(cmd) 에서 톰캣의 bin 폴더로 이동한 후 startup.bat 를 실행한다.

이 명령을 통해 톰캣을 실행할 수 있다.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

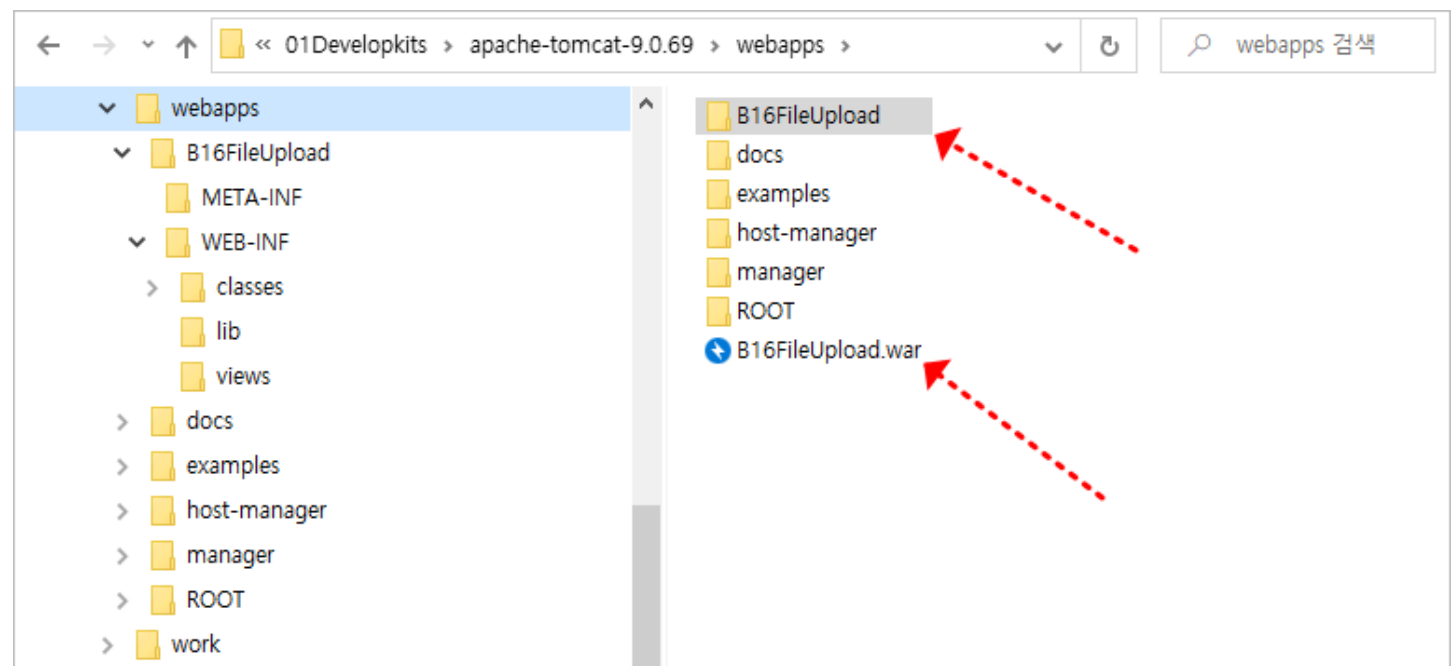
C:\Users\wtjoeun_jr>cd C:\W01Developkits\Wapache-tomcat-9.0.69\bin
C:\W01Developkits\Wapache-tomcat-9.0.69\bin>startup.bat
Using CATALINA_BASE:   "C:\W01Developkits\Wapache-tomcat-9.0.69"
Using CATALINA_HOME:   "C:\W01Developkits\Wapache-tomcat-9.0.69"
Using CATALINA_TMPDIR: "C:\W01Developkits\Wapache-tomcat-9.0.69\temp"
Using JRE_HOME:        "C:\W01Developkits\Wjdk-17"
Using CLASSPATH:       "C:\W01Developkits\Wapache-tomcat-9.0.69\bin\bootstrap.jar;C:\W01Developkits\Wapache-tomcat-9.0.69\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
C:\W01Developkits\Wapache-tomcat-9.0.69\bin>
```

톰캣이 실행되면 별도의 실행창이 하나 더 뜨게된다. 여기에서 웹 애플리케이션 실행시의 모든 로그를 확인할 수 있다.

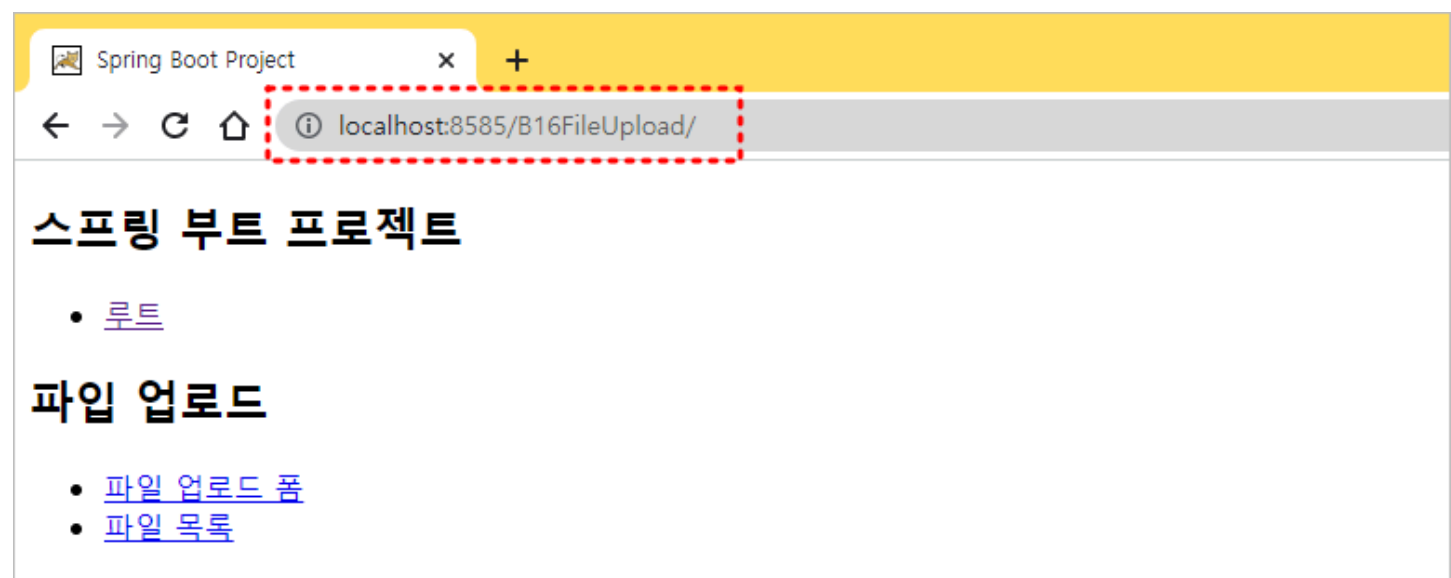
```
Tomcat
org.springframework.security.web.context.SecurityContextPersistenceFilter@6113d713, org.springframework.security.web.header.HeaderWriterFilter@74a0d8a, org.springframework.security.web.authentication.logout.LogoutFilter@2a97559f, org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter@47a75bc, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@44be009c, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@695295a9, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@aacb1f0, org.springframework.security.web.session.SessionManagementFilter@5931a306, org.springframework.security.web.access.ExceptionTranslationFilter@55b4ebbd, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@233cd973]
2023-03-08 14:45:04.560 INFO 29616 --- [main] com.edu.springboot.ServletInitializer : Started ServletInitializer in 4.731 seconds (JVM running for 20.719)
08-Mar-2023 14:45:04.577 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory ??????
?덱??? ????[C:\W01Developkits\Wapache-tomcat-9.0.69\webapps\ROOT]???[8,972]
08-Mar-2023 14:45:04.584 INFO [main] org.apache.coyote.AbstractProtocol.start ????["http-nio-8585"]??
08-Mar-2023 14:45:04.613 INFO [main] org.apache.catalina.startup.Catalina.start ????[18423]
???
```

출력된 로그에는 한글이 깨진것을 볼 수 있지만 우리가 print()문으로 출력한 로그는 정상적으로 나오니 걱정할 필요없다.

톰켓이 실행되면 우리가 배포했던 war파일은 자동으로 압축해제되어 아래와 같이 디렉토리로 생성된다.



정상적으로 실행되는지 웹브라우저에서 확인해보자.



실행시 두가지가 달라진것을 볼 수 있다.

1. 포트번호가 8586이 아니라 8585을 사용해야 한다.
2. 스프링 레거시와 같이 컨택스트 루트를 호스트 뒤에 붙여줘야 한다.

첫번째, 포트번호는 내장 톰켓이 아니라 외부 톰켓을 사용하므로 server.xml에 설정된 값을 따른다.

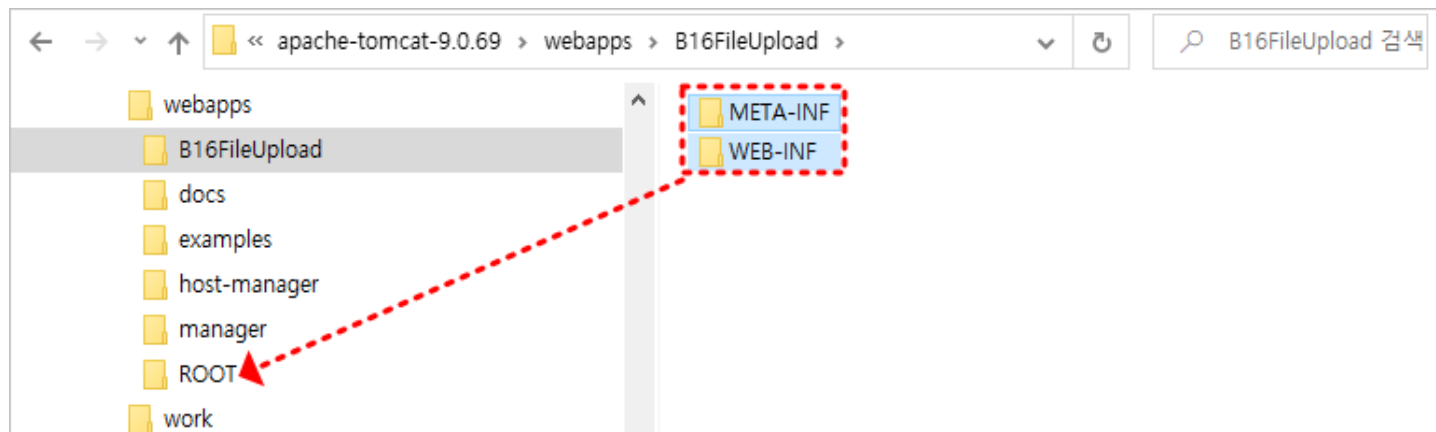
```
<Connector port="8585" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443" />
```

두번째, 컨텍스트 루트명을 제거해보겠다.

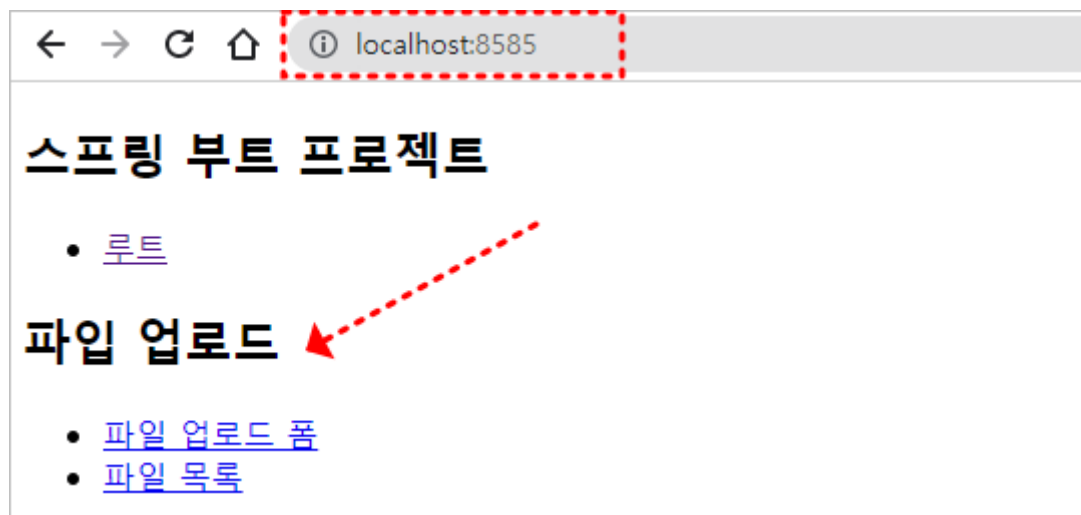
먼저 기존 실행된 실행창을 닫아 톰켓을 종료한다.

war 파일을 배포한 webapps 하위에는 ROOT 라는 폴더가 있다.

해당 폴더의 모든 파일을 삭제한 후 B16FileUpload 하위의 모든 파일을 이동시켜주면 된다.

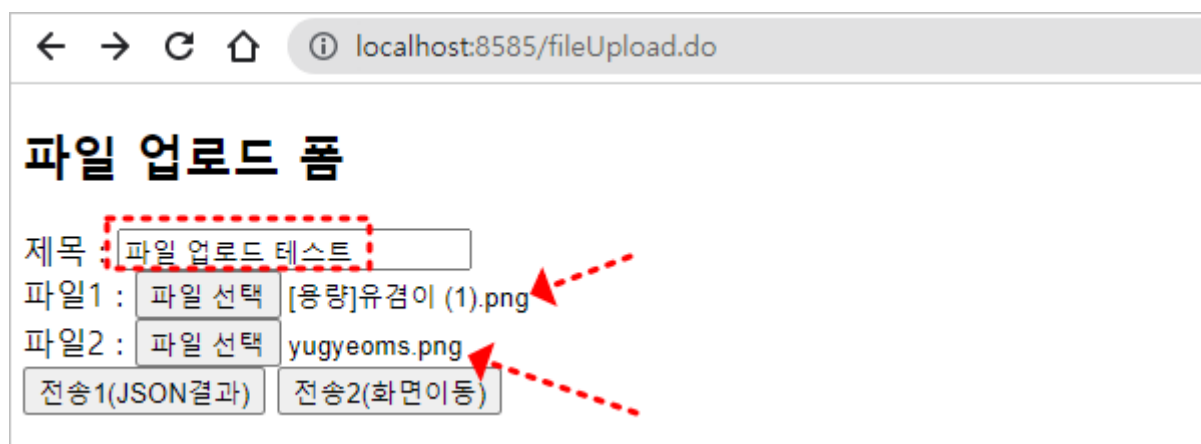


톰켓을 다시 시작한 후 웹브라우저에서 실행해본다. 컨텍스트 루트가 제거되었다.



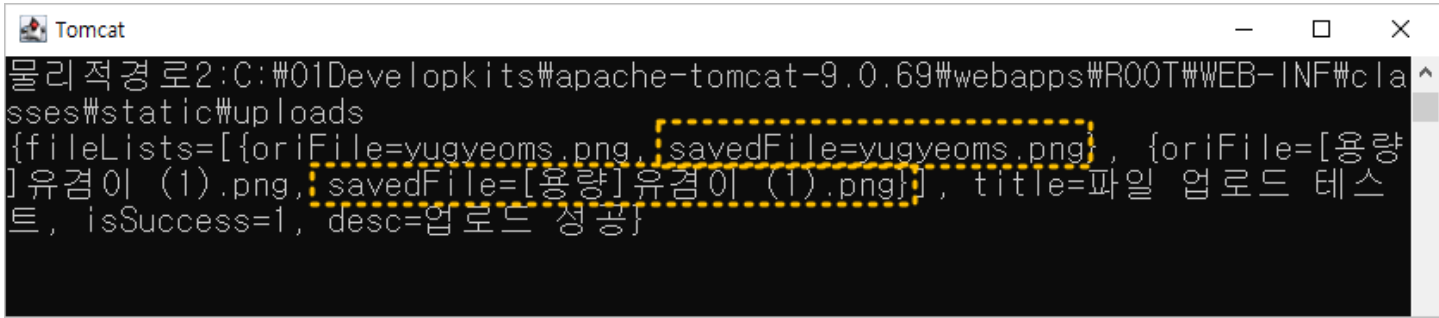
파일업로드 폼으로 이동한 후 파일을 업로드 해보겠다

제목과 2개의 파일을 입력한다. 단 파일명의 경우 영문과 한글 두가지를 모두 업로드 해보겠다.

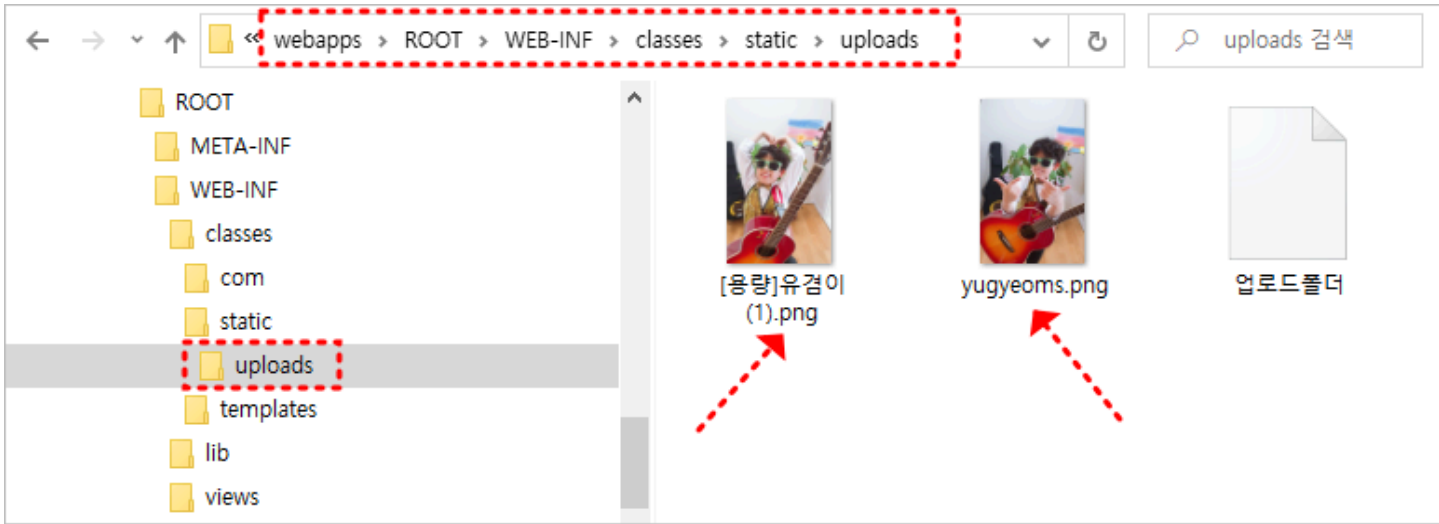


전송2를 누른다.

전송결과1 (콘솔)



전송결과2 (디렉토리)



전송결과3 (웹브라우저)

← → ↻ 🏠 ⓘ localhost:8585/fileList.do ⚙ ☆

파일목록 보기

이미지	파일명	파일크기	
	업로드폴더	0Kb	[다운로드]
	[용량]유겸이 (1).png	508Kb	[다운로드]
	yugyeoms.png	472Kb	[다운로드]

파일목록을 보면 파일명이 한글인 경우는 이미지가 제대로 표시되지 않는다. ㅌㅌ
이 때문에 불편하지만 저장된 파일명을 영문이나 숫자로 변경하는 것이다. ^^*

수업시간에 배웠던 물리적 경로가 bin이나 metadata가 아닌 우리가 지정한 디렉토리인것을 알 수 있다.

