

2장. 테스트와 로깅, 빌딩



JUnit 테스트



Spring Boot

스프링 부트에서 테스트하기

- 스프링 부트 테스트

대부분의 프로젝트에서는 Junit을 기반으로 일관성 있는 단위 테스트를 진행하는데, 단위 테스트란 자신이 작성한 클래스에 대한 테스트로서 테스트 중에 가장 기본이라 할 수 있다.

단위 테스트가 효율적으로 이루어지기 위해서는 테스트할 객체가 최대한 단순해야 한다. 따라서 서버를 구동하지 않고도 컨트롤러만 단독으로 테스트하거나 컨트롤러와 연관된 비즈니스 컴포넌트를 실행하지 않고 컨트롤러만 독립적으로 테스트할 수 있는 환경이 필요하다.

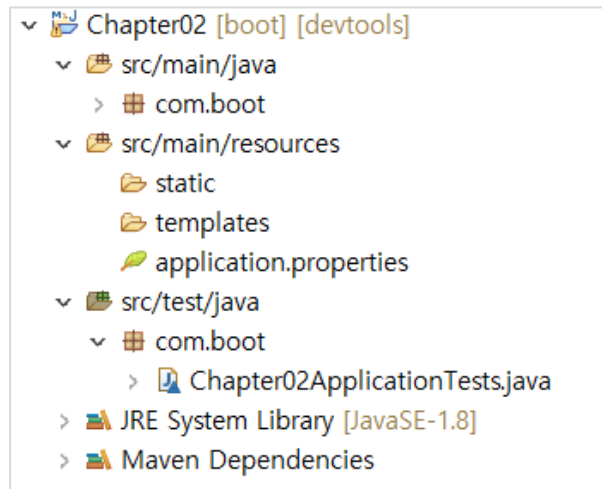


스프링 부트에서 테스트하기

■ 테스트 스타터 환경

실습을 위해 Jar, 8 버전 선택하고,

DevTools, Lombok, Web을 추가한 Chapter02 프로젝트 생성



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```



스프링 부트에서 테스트하기

- 테스트 스타터 환경 - 스프링 부트 제공 테스트 케이스

- (1) 기본 테스트 클래스 이해하기

```
package com.boot;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class Chapter02ApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

@SpringBootTest 어노테이션은 테스트 케이스가 실행될때 테스트에 필요한 모든 설정과 빈들을 자동으로 초기화 하는 역할 수행함

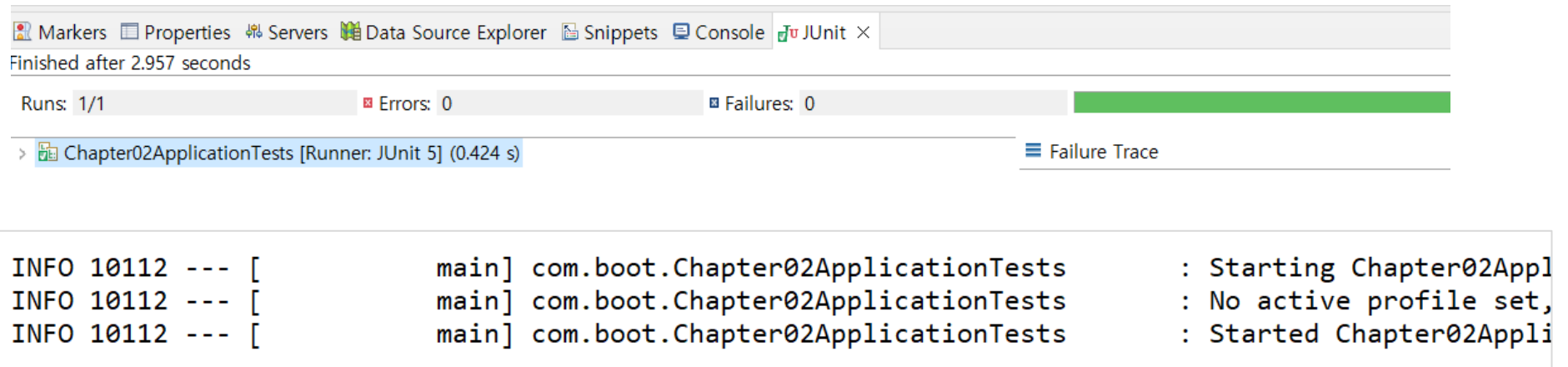


스프링 부트에서 테스트하기

- 테스트 스타터 환경 - 스프링 부트 제공 테스트 케이스

(2) 테스트 케이스 실행하기

마우스 오른쪽 > [Run As] -> [JUnit Test]



Markers Properties Servers Data Source Explorer Snippets Console JUnit x

Finished after 2.957 seconds

Runs: 1/1 Errors: 0 Failures: 0

> Chapter02ApplicationTests [Runner: JUnit 5] (0.424 s) Failure Trace

```
INFO 10112 --- [main] com.boot.Chapter02ApplicationTests : Starting Chapter02App1
INFO 10112 --- [main] com.boot.Chapter02ApplicationTests : No active profile set,
INFO 10112 --- [main] com.boot.Chapter02ApplicationTests : Started Chapter02Appli
```



스프링 부트에서 테스트하기

- 테스트 케이스 만들기

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
public class PropertiesTest {

    @Test
    public void testMethod() {

    }

}
```

```
2022-09-15 00:25:51.149 INFO 1784 --- [           main] com.boot.PropertiesTest
2022-09-15 00:25:51.149 INFO 1784 --- [           main] com.boot.PropertiesTest
==> BoardController 생성
2022-09-15 00:25:52.691 INFO 1784 --- [           main] com.boot.PropertiesTest
```



스프링 부트에서 테스트하기

- 프로퍼티 사용하기 – src/main/resources/application.properties

Run As > Spring Boot App으로 실행하기

```
@SpringBootApplication
public class Chapter02Application {

    public static void main(String[] args) {
        SpringApplication.run(Chapter02Application.class, args);
    }

}
```

application.properties

src/main/resources

- static
- templates
- application.properties

```
## WebApplication Type Setting
##spring.main.web-application-type=none
spring.main.web-application-type=servlet

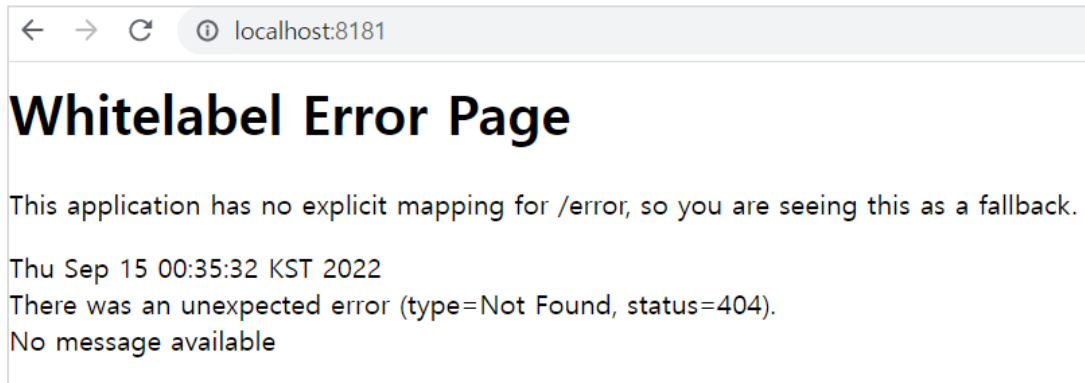
## Server Setting(포트번호 변경)
server.port = 8181
```



스프링 부트에서 테스트하기

- 프로퍼티 사용하기

```
o.s.b.d.a.OptionalLiveReloadServer      : LiveReload server is running on port 35729
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8181 (http) with co
com.boot.Chapter02Application           : Started Chapter02Application in 0.212 seconds
.ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
```



스프링 부트에서 테스트하기

- 프로퍼티 사용하기

```
## Server Setting(포트번호 변경)
server.port = 8181

## Test Property Setting
author.name=TESTER
author.age=29
```

```
@SpringBootTest
public class PropertiesTest {

    @Autowired
    Environment environment;

    @Test
    public void testMethod() {
        System.out.println("이름: " + environment.getProperty("author.name"));
        System.out.println("나이: " + environment.getProperty("author.age"));
    }
}
```

```
==> BoardController 생성
2022-09-15 00:40:40.891 INFO 13548 --- [
이름: TESTER
나이: 29
```



스프링 부트에서 테스트하기

- 프로퍼티 재정의하기

```
@SpringBootTest(properties= {"author.name=Gurum",
                              "author.age=45",
                              "author.nation=Korea"})
public class PropertiesTest {

    @Autowired
    Environment environment;

    @Test
    public void testMethod() {
        System.out.println("이름: " + environment.getProperty("author.name"));
        System.out.println("나이: " + environment.getProperty("author.age"));
        System.out.println("국적: " + environment.getProperty("author.nation"));
    }
}
```

```
2022-09-15 00:54:55.879 INFO 4124 --- [
이름: Gurum
나이: 45
국적: Korea
```

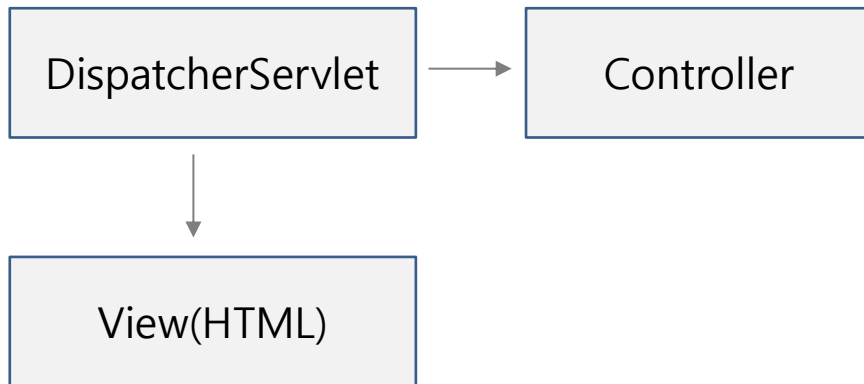


스프링 부트에서 테스트하기

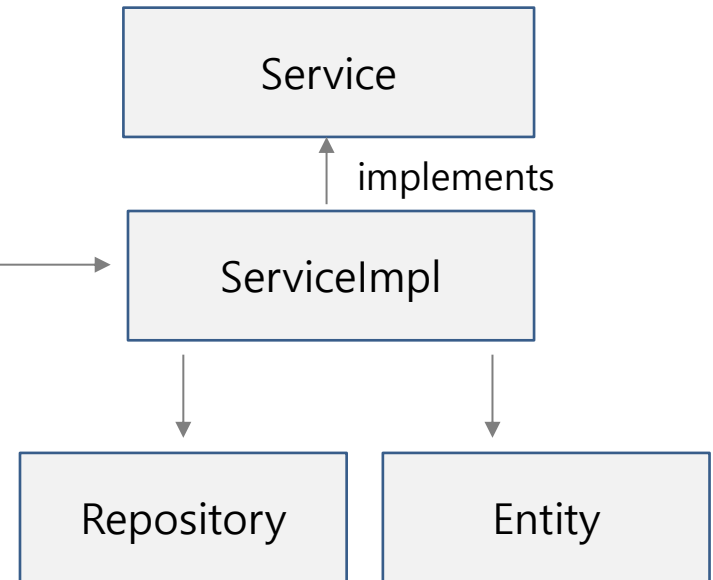
- 서비스 계층을 연동하는 컨트롤러 테스트하기

실제 프로젝트에서는 Controller는 단순히 사용자의 요청을 받아들이는 역할만 한다. 컨트롤러에서 실제 사용자가 원하는 비즈니스 로직을 처리하기 위해서 비즈니스 컴포넌트를 호출해야 한다.

프레젠테이션 레이어(MVC)



비즈니스 레이어(MVC)



스프링 부트에서 테스트하기

- 서비스 계층을 연동하는 컨트롤러 테스트하기

Chapter01에서 만든 BoardVO, BoardController는 복사해 옴.

BoardService와 BoardServiceImpl을 생성함.

```
src/main/java
├── com.boot
│   ├── Chapter02Application.java
│   ├── com.boot.controller
│   │   └── BoardController.java
│   ├── com.boot.domain
│   │   └── BoardVO.java
│   └── com.boot.service
│       ├── BoardService.java
│       └── BoardServiceImpl.java
```

```
public interface BoardService {

    String hello(String name);

    BoardVO getBoard();

    List<BoardVO> getBoardList();

}
```



스프링 부트에서 테스트하기

- 서비스 계층을 연동하는 컨트롤러 테스트하기

```
@Service
public class BoardServiceImpl implements BoardService{

    @Override
    public String hello(String name) {
        return "hello: " + name;
    }

    @Override
    public BoardVO getBoard() {
        BoardVO board = new BoardVO();
        board.setSeq(1);
        board.setTitle("테스트 제목...");
        board.setWriter("테스터");
        board.setContent("테스트 내용입니다...");
        board.setCreateDate(new Date());
        board.setCnt(0);
        return board;
    }
}
```



스프링 부트에서 테스트하기

- 서비스 계층을 연동하는 컨트롤러 테스트하기

```
@Override
public List<BoardVO> getBoardList() {
    List<BoardVO> boardList = new ArrayList<>();
    for(int i=1; i<=10; i++) {
        BoardVO board = new BoardVO();
        board.setSeq(i);
        board.setTitle("제목" + i);
        board.setWriter("테스터");
        board.setContent(i + "번 내용입니다.");
        board.setCreateDate(new Date());
        board.setCnt(0);
        boardList.add(board);
    }
    return boardList;
}
```



스프링 부트에서 테스트하기

- BoardController는 단순히 BoardService가 제공하는 메소드를 호출하여 사용자 요청을 처리하도록 수정한다.

```
@RestController
public class BoardController {

    @Autowired
    private BoardService service; //의존성 주입(DI)

    //문자열 리턴
    @GetMapping("/hello")
    public String hello(String name) {
        return service.hello(name);
    }
}
```



스프링 부트에서 테스트하기

```
//vo 객체 리턴
@GetMapping("/getBoard")
public BoardVO getBoard() {
    BoardVO board = service.getBoard();
    return board;
}

//컬렉션(리스트) 리턴
@GetMapping("/getBoardList")
public List<BoardVO> getBoardList(){
    List<BoardVO> boardList = service.getBoardList();
    return boardList;
}
}
```



스프링 부트에서 테스트하기

- 프로젝트 실행하기

```
@SpringBootApplication
public class Chapter02Application {

    public static void main(String[] args) {
        SpringApplication.run(Chapter02Application.class, args);
    }

}
```

← → ↻ ⓘ localhost:8181/hello?name=장그래

Hello : 장그래

← → ↻ ⓘ localhost:8181/getBoard

{"seq":1,"title":"테스트 제목...","writer":"테스터","content":"테스트 내용입니다."}



스프링 부트 로깅(Logging)

■ 스프링 부트 로깅

스프링 부트는 SLF4J(Simple Logging Façade for Java)를 이용하여 로그를 관리한다
퍼사드는 GoF 디자인 패턴 중 하나로서 복잡한 서브 시스템을 쉽게 사용할 수 있도록
간단하고 통일된 인터페이스를 제공한다.

SLF4J라는 퍼사드를 통해 궁극적으로는 LogBack을 사용한다.

LogBack은 기존의 Log4j에 비해 성능이나 기능면에서 많은 장점을 제공한다.

```
> spring-boot-starter-logging-2.7.3.jar - C:\Users\Wkiyon\m2\repository\org\springframework
> logback-classic-1.2.11.jar - C:\Users\Wkiyon\m2\repository\ch\qos\logback\logback-classic
> logback-core-1.2.11.jar - C:\Users\Wkiyon\m2\repository\ch\qos\logback\logback-core\1.
> log4j-to-slf4j-2.17.2.jar - C:\Users\Wkiyon\m2\repository\org\apache\logging\log4j\log4j-
> log4j-api-2.17.2.jar - C:\Users\Wkiyon\m2\repository\org\apache\logging\log4j\log4j-api
```

```
INFO 6032 --- [ restartedMain] com.boot.Chapter02Application
INFO 6032 --- [ restartedMain] com.boot.Chapter02Application
INFO 6032 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
INFO 6032 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor
INFO 6032 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
INFO 6032 --- [ restartedMain] o.apache.catalina.core.StandardService
INFO 6032 --- [ restartedMain] org.apache.catalina.core.StandardEngine
INFO 6032 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]
```



스프링 부트 빌드(Build)

- **스프링 부트 빌드**

스프링 부트는 메이븐을 이용해서 프로젝트를 생성하고, 완성된 프로젝트의 빌드 역시 메이븐으로 처리한다.

빌드라는 것은 완성된 프로젝트에서 소스를 컴파일하고 컴파일된 소스들을 적절한 폴더에 취합하고 JAR나 WAR 파일로 패키징하여 운영 서버에 배포하는 일련의 과정을 의미한다.

스프링부트는 독립적으로 실행 가능한 애플리케이션을 빠르게 개발하는 것을 목표로 한다. 그래서 웹 애플리케이션도 WAR가 아닌 JAR 파일로 패키징하여 사용할 수 있도록 지원한다.

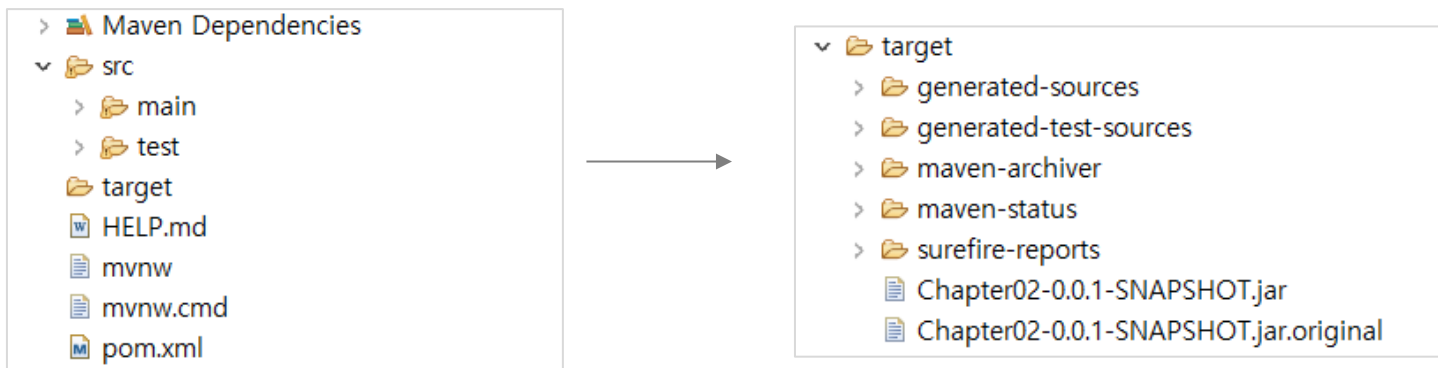


스프링 부트 빌드(Build)

■ 스프링 부트 빌드

메이븐이 프로젝트를 빌드할 때 기본적으로 사용하는 폴더가 target이다. 따라서 현재 프로젝트를 빌드하면 target 폴더에 빌드 과정에서 생성한 임시 폴더와 파일들이 생기고 패키징 결과 파일도 만들어진다.

프로젝트 우측 메뉴 -> [Run As] -> [Maven install]



```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 13.904 s  
[INFO] Finished at: 2022-09-17T06:16:10+09:00  
[INFO] -----  
[WARNING] The requested profile "pom.xml" could not be activated
```



스프링 부트 빌드(Build)

- 스프링 부트 빌드

Chapter02-0.0.1-SNAPSHOT.jar 파일 생성됨

pom.xml 설정에서 <artifactId> + <version> + jar 형태의 패키징 파일이 생성됨

```
<groupId>com.boot</groupId>
<artifactId>Chapter02</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Chapter02</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>1.8</java.version>
</properties>
```



스프링 부트 빌드(Build)

- JAR 파일 압축풀기

| Dev > jwspring > jar > Chapter02-0.0.1-SNAPSHOT > META-INF | | |
|--|--------------------|-------|
| 이름 | 수정한 날짜 | 유형 |
| maven | 2022-09-17 오전 6:27 | 파일 폴더 |
| MANIFEST.MF | 2022-09-17 오전 6:16 | MF 파일 |

| | |
|---|--|
| MANIFEST.MF - Windows 메모장 | |
| 파일(F) 편집(E) 서식(O) 보기(V) 도움말(H) | |
| Manifest-Version: 1.0 | |
| Created-By: Maven JAR Plugin 3.2.2 | |
| Build-Jdk-Spec: 17 | |
| Implementation-Title: Chapter02 | |
| Implementation-Version: 0.0.1-SNAPSHOT | |
| Main-Class: org.springframework.boot.loader.JarLauncher | |
| Start-Class: com.boot.Chapter02Application | |
| Spring-Boot-Version: 2.7.3 | |
| Spring-Boot-Classes: BOOT-INF/classes/ | |
| Spring-Boot-Lib: BOOT-INF/lib/ | |
| Spring-Boot-Classpath-Index: BOOT-INF/classpath.idx | |
| Spring-Boot-Layers-Index: BOOT-INF/layers.idx | |

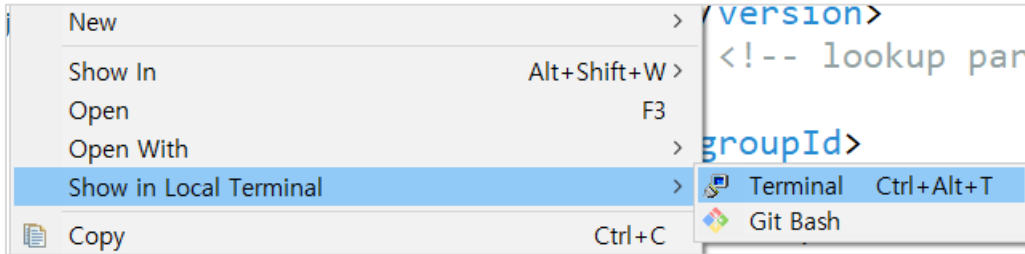
Main-Class는 JavaLancher 클래스라는 의미이며, Start-Class 정보는 Chapter02Application 클래스를 시작으로 실행한다는 의미이다.



스프링 부트 빌드(Build)

- Runnable JAR 실행하기

Chapter02-0.0.1-SNAPSHOT.jar > 우측메뉴 > Show in Local Terminal > Terminal



java -jar Chapter02-0.0.1-SNAPSHOT.jar 명령 입력

[illegible]

스프링 부트 빌드(Build)

- Runnable JAR 실행하기

localhost:8181/getBoardList 요청

```
← → ↺ ⓘ localhost:8181/getBoardList

[{"seq":1,"title":"제목1","writer":"테스터","content":"1번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":3,"title":"제목3","writer":"테스터","content":"3번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":4,"writer":"테스터","content":"4번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":6,"title":"제목6","writer":"테스터","content":"6번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":7,"writer":"테스터","content":"7번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":9,"title":"제목9","writer":"테스터","content":"9번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}, {"seq":10,"writer":"테스터","content":"10번 내용입니다.","createDate":"2022-09-16T21:43:20.783+00:00","cnt":0}]
```

| | | | |
|------------------------------|---------------------------------------|-------|--------|
| Chapter02-0.0.1-SNAPSHOT.jar | .. | | |
| BOOT-INF | archive | | |
| META-INF | data | | |
| org | jar | | |
| springframework | jarmode | | |
| boot | util | | |
| loader | ClassPathIndexFile.class | 2,675 | 5,871 |
| archive | ExecutableArchiveLauncher.class | 3,025 | 7,675 |
| data | JarLauncher.class | 1,098 | 2,551 |
| jar | LaunchedURLClassLoader\$DefineP... | 655 | 1,483 |
| jarmode | LaunchedURLClassLoader\$UseFast... | 723 | 1,535 |
| util | LaunchedURLClassLoader.class | 4,850 | 11,154 |
| | Launcher.class | 2,478 | 5,932 |
| | MainMethodRunner.class | 780 | 1,536 |
| | PropertiesLauncher\$1.class | 181 | 266 |
| | PropertiesLauncher\$ArchiveEntryFi... | 637 | 1,484 |
| | PropertiesLauncher\$ClassPathArch... | 3,384 | 8,128 |
| | PropertiesLauncher\$PrefixMatchin... | 783 | 1,953 |

JarLauncher 클래스는
org/springframework.boot/loader
폴더에 위치함



스프링 부트 빌드(Build)

- Runnable JAR 실행하기

명령프롬프트(CMD) 에서 실행하기

[illegible]