

# Spring Boot Application

---

Jae Hyeon Kim

# — Contents

1. WAS
2. Spring Boot Application

***WAS***

# — Server

- 네트워크에서 다른 컴퓨터나 소프트웨어와 같은 클라이언트에게 서비스를 제공하는 컴퓨터
- 서버는 서비스를 제공하기 위한 소프트웨어인 서버 소프트웨어가 가지고 있는 기능 자체를 뜻한다

## 웹 서버 소프트웨어

Apache

IIS

nginx

웹 서버 소프트웨어의 경우 위 제품이 대표적입니다. 각각의 특징은 다르지만 '웹 서버'로서의 기능은 똑같습니다.

# — Web Server

- 웹 서버는 클라이언트가 요청한 정적인 콘텐츠를 HTTP 프로토콜을 통해 제공하는 서버
- 동적인 요청이 클라이언트로부터 들어왔을 때, 해당 요청을 웹 서버에서 처리할 수 없기 때문에 컨테이너로 보내줌

# — Static Page & Dynamic Page

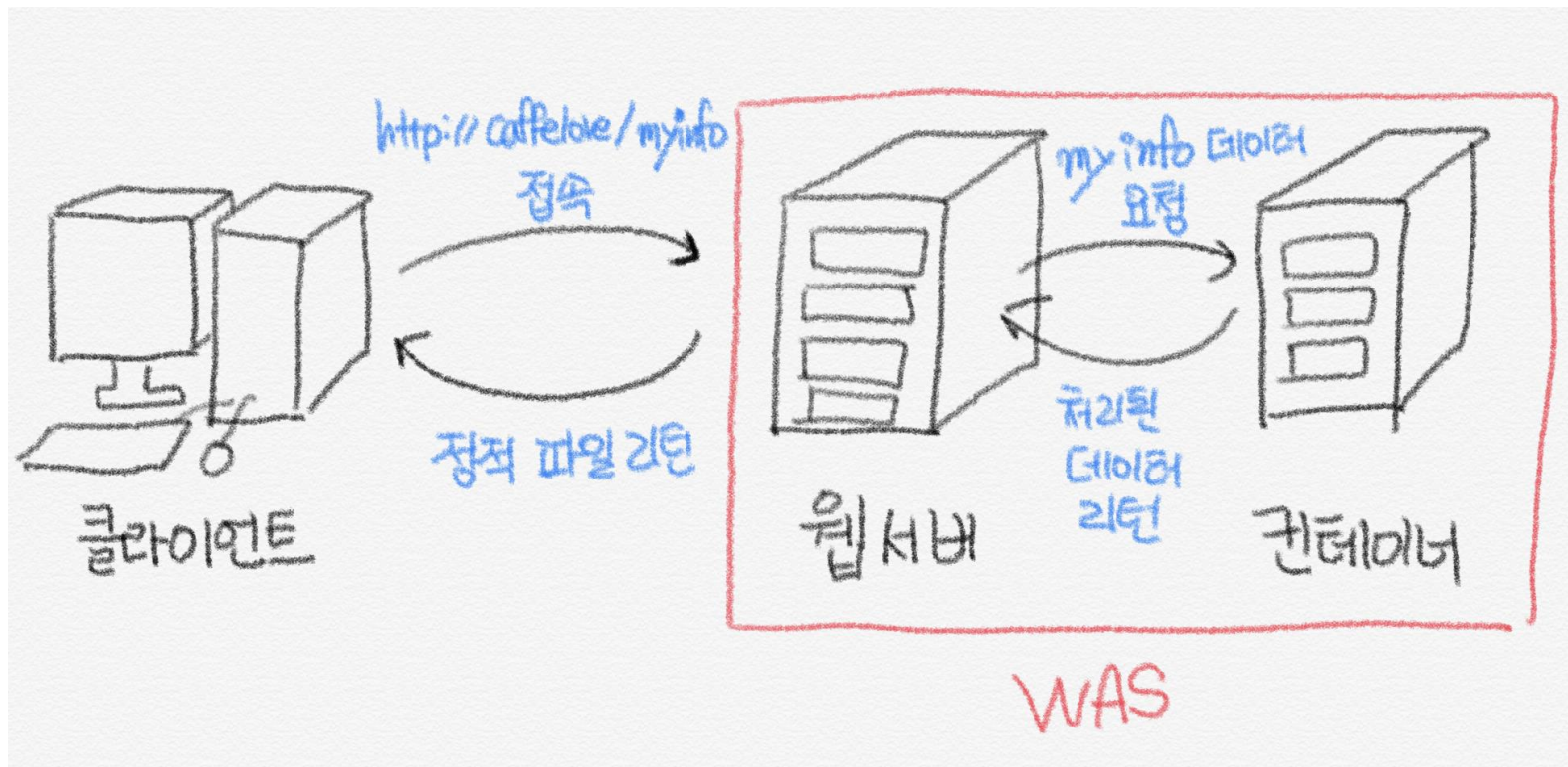
- 정적 페이지는 데이터베이스에서 정보를 가져오거나 별도의 서버에서의 처리가 없어도 사용자들에게 보여줄 수 있는 페이지
- 동적 페이지는 서버나 데이터베이스에서 정보를 가져와서 처리하는 것처럼, 어떠한 요청에 의하여 서버가 일을 수행하고 해당 결과가 포함된 파일을 보여주는 페이지

# — Container

- 컨테이너는 동적인 데이터들을 처리하여 정적인 페이지로 생성해주는 소프트웨어 모듈
- 컨테이너는 동적인 데이터를 처리해서 웹 서버에 정적인 파일로 만들어 보내준다

# — WAS

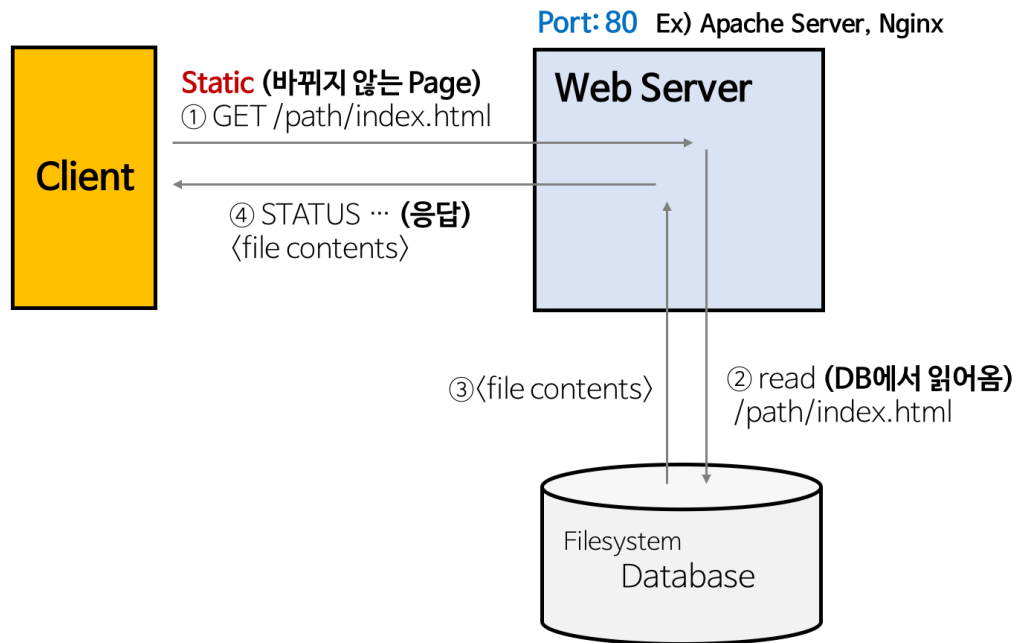
- 웹 서버와 컨테이너를 붙여 놓은 서버



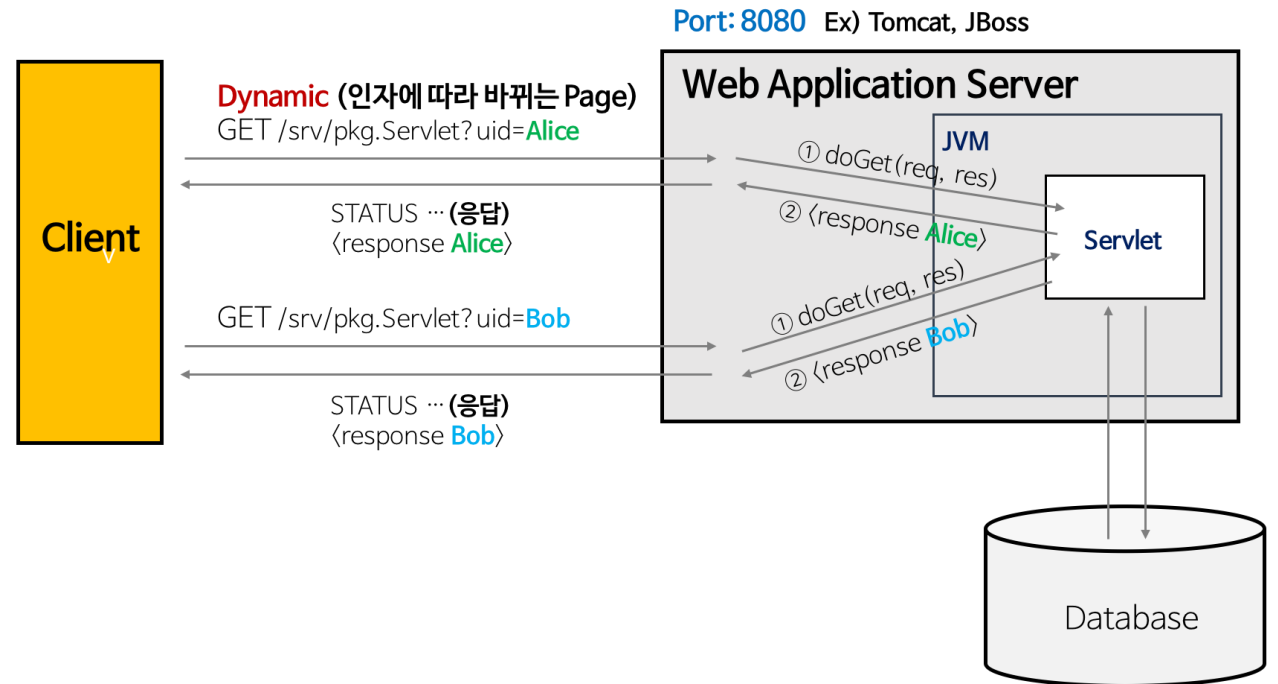


# WAS

## Static Pages

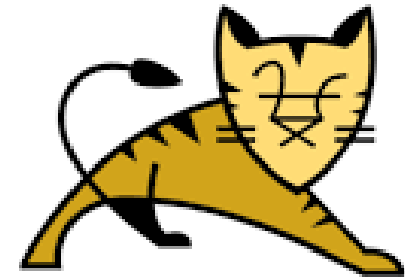


## Dynamic Pages



# — Apache Tomcat

- Apache Tomcat은 자바 servlet을 이용하여 데이터 요청에 대한 응답을 자바코드로 처리하고, 해당 내용을 유저에게 제공하는 구조를 가진 WAS
- 보통은 application sever를 tomcat으로 이용하는 경우 정적인 데이터는 웹 서버인 apache가 처리하도록 하기 때문에 apache tomcat이라고 부른다

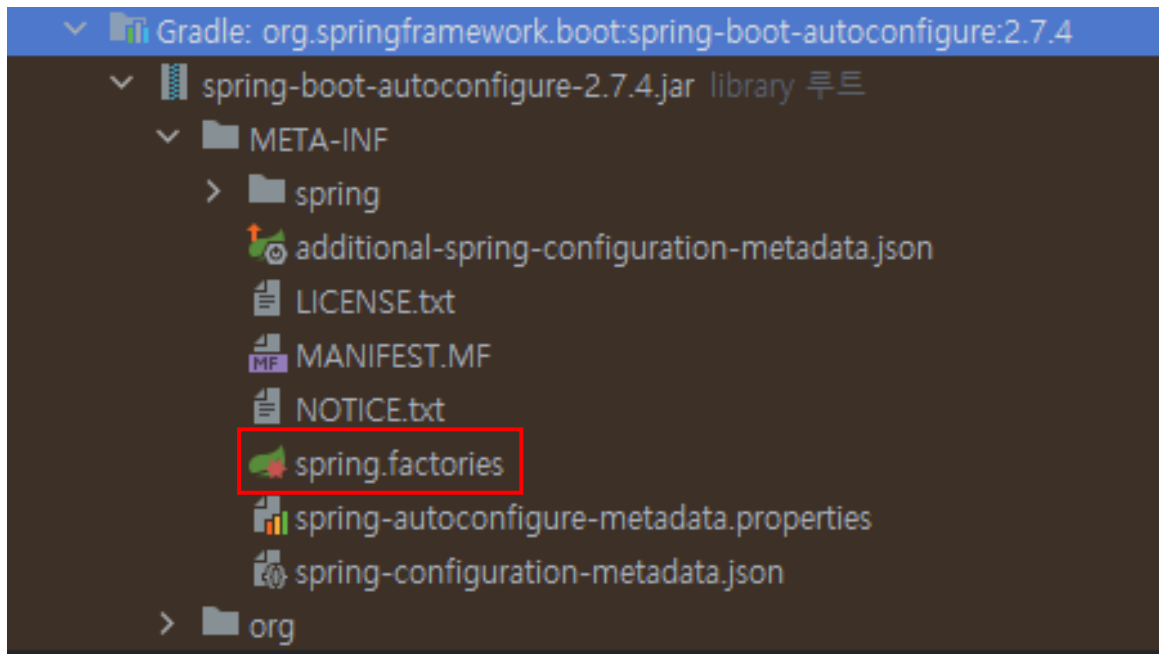


Apache Tomcat

# **Spring Boot Application**

# — Tomcat on Spring Boot

- ServletWebServerFactoryAutoConfiguration 클래스에서 자동적으로 Tomcat, jetty 같은 내장 WAS에 대한 설정을 자동적으로 처리



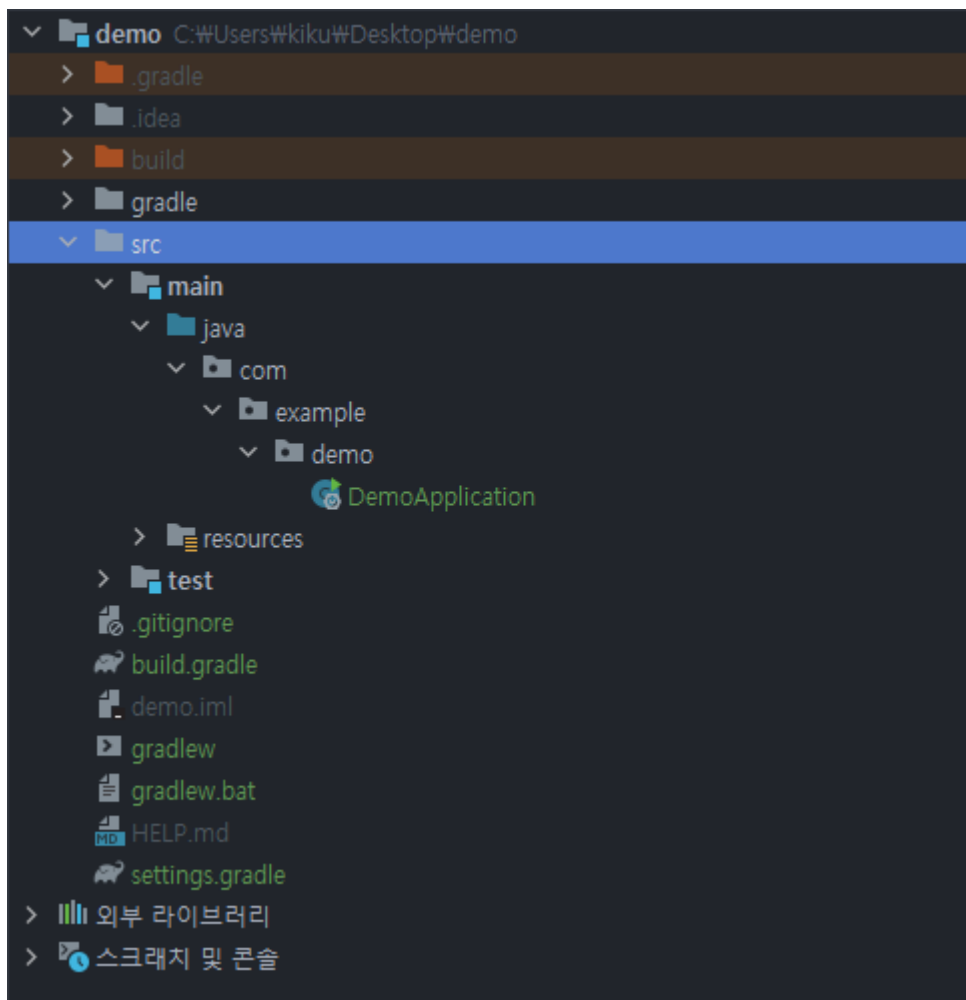
# — Tomcat on Spring Boot

```
@AutoConfiguration
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE)
@ConditionalOnClass(ServletRequest.class)
@ConditionalOnWebApplication(type = Type.SERVLET)
@EnableConfigurationProperties(ServerProperties.class)
@Import({ ServletWebServerFactoryAutoConfiguration.BeanPostProcessorsRegistrar.class,
    ServletWebServerFactoryConfiguration.EmbeddedTomcat.class,
    ServletWebServerFactoryConfiguration.EmbeddedJetty.class,
    ServletWebServerFactoryConfiguration.EmbeddedUndertow.class })
public class ServletWebServerFactoryAutoConfiguration {

    @Bean
    public ServletWebServerFactoryCustomizer servletWebServerFactoryCustomizer(ServerProperties serverProperties,
        ObjectProvider<WebListenerRegistrar> webListenerRegistrars,
        ObjectProvider<CookieSameSiteSupplier> cookieSameSiteSuppliers) {
        return new ServletWebServerFactoryCustomizer(serverProperties,
            webListenerRegistrars.orderedStream().collect(Collectors.toList()),
            cookieSameSiteSuppliers.orderedStream().collect(Collectors.toList()));
    }

    @Bean
    @ConditionalOnClass(name = "org.apache.catalina.startup.Tomcat")
    public TomcatServletWebServerFactoryCustomizer tomcatServletWebServerFactoryCustomizer(
        ServerProperties serverProperties) {
        return new TomcatServletWebServerFactoryCustomizer(serverProperties);
    }
}
```

# @SpringBootApplication



```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

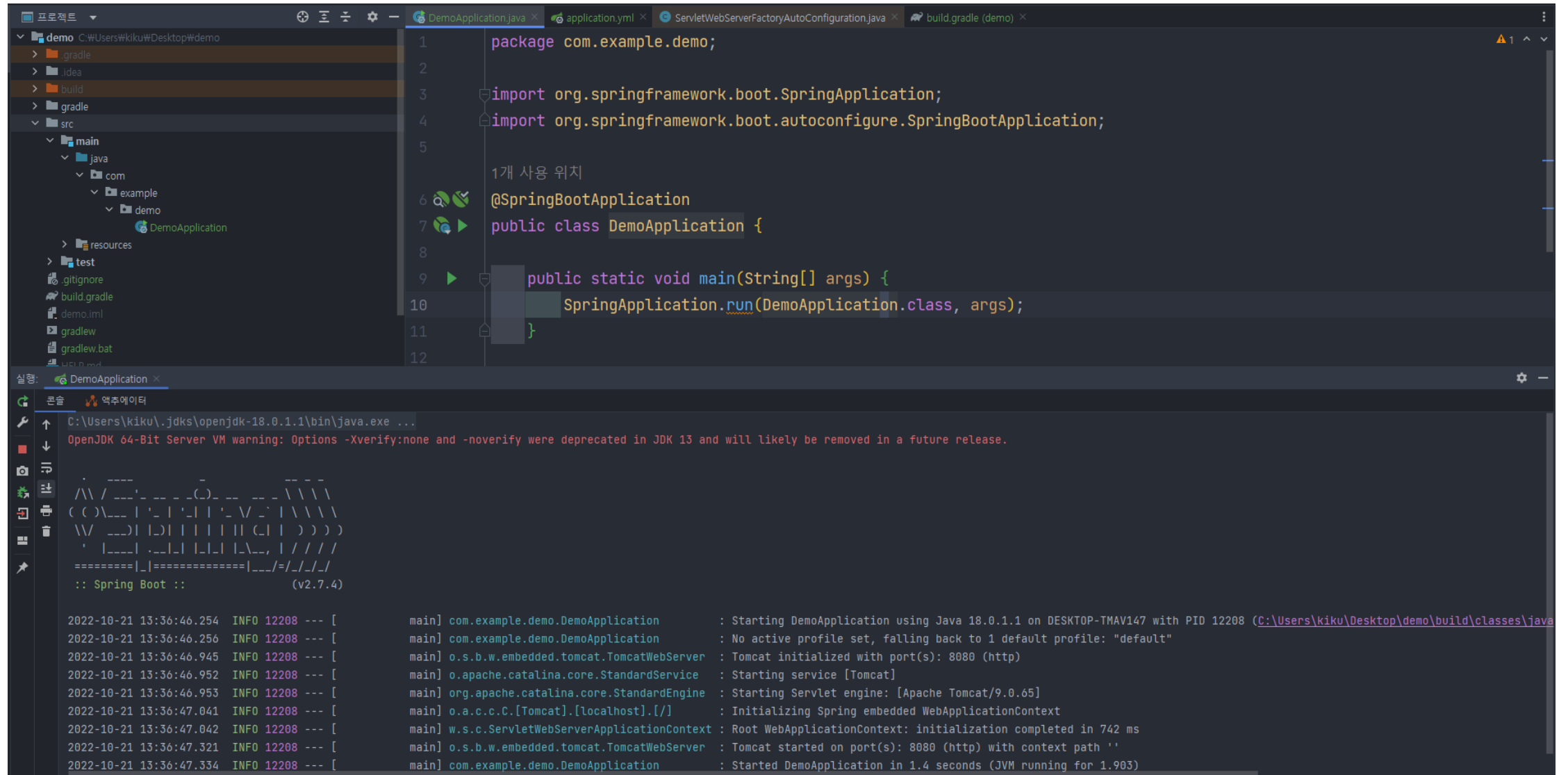
1개 사용 위치

@SpringBootApplication
public class DemoApplication {

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

}
```

# @SpringBootApplication



The screenshot displays an IDE with the following components:

- Project Explorer:** Shows a project named 'demo' with a directory structure including 'src/main/java/com/example/demo' and 'resources'.
- Code Editor:** Displays the content of 'DemoApplication.java'.
- Run Console:** Shows the execution output of the application.

**Code Editor Content:**

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 1개 사용 위치
7 @SpringBootApplication
8 public class DemoApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(DemoApplication.class, args);
12     }
13 }
```

**Run Console Output:**

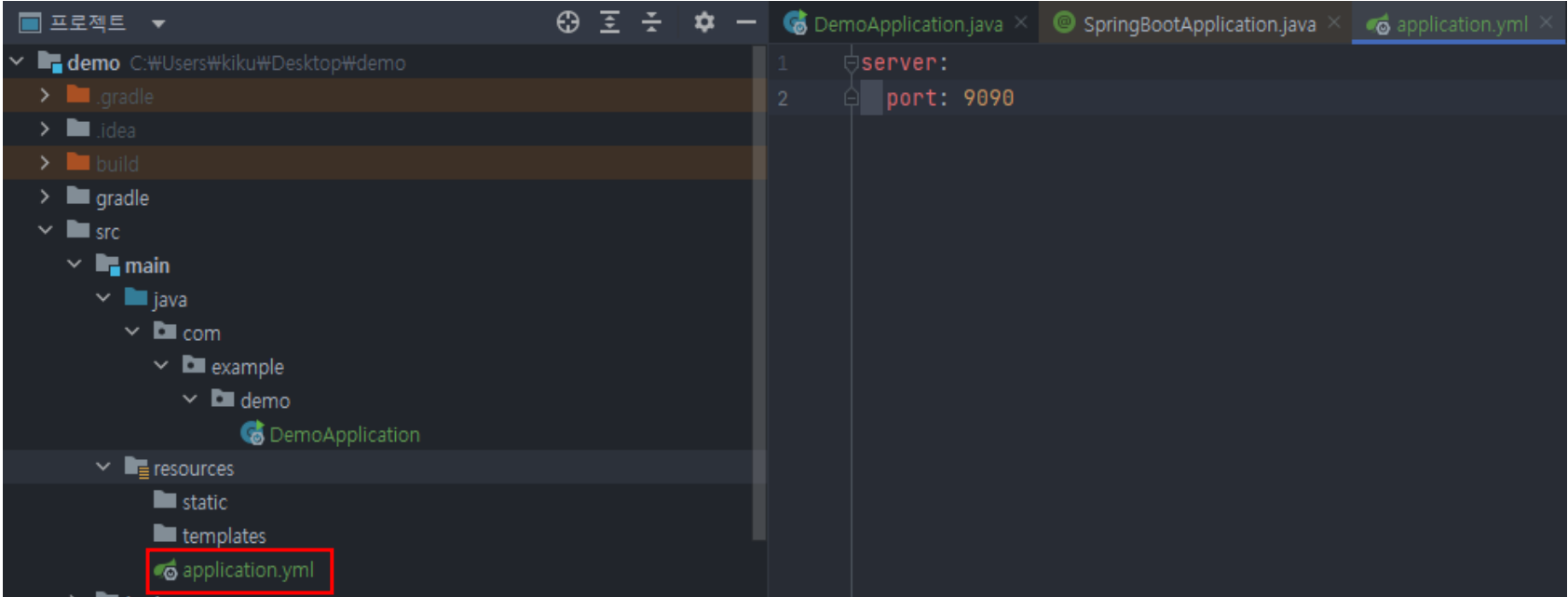
```
실행: DemoApplication
C:\Users\kiku\.jdk\openjdk-18.0.1.1\bin\java.exe ...
OpenJDK 64-Bit Server VM warning: Options -Xverify:none and -noverify were deprecated in JDK 13 and will likely be removed in a future release.

  ____ _
 / ___ \ | |
/ /___ \ | |
\ ___ \ | |
 \___ \_|_|
  =====|_|=====|_|/=/_/=/_/

:: Spring Boot ::      (v2.7.4)

2022-10-21 13:36:46.254 INFO 12208 --- [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 18.0.1.1 on DESKTOP-TMAV147 with PID 12208 (C:\Users\kiku\Desktop\demo\build\classes\java
2022-10-21 13:36:46.256 INFO 12208 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to 1 default profile: "default"
2022-10-21 13:36:46.945 INFO 12208 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-10-21 13:36:46.952 INFO 12208 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-21 13:36:46.953 INFO 12208 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-21 13:36:47.041 INFO 12208 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-21 13:36:47.042 INFO 12208 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 742 ms
2022-10-21 13:36:47.321 INFO 12208 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-21 13:36:47.334 INFO 12208 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 1.4 seconds (JVM running for 1.903)
```

\_\_\_\_\_



```
2022-10-21 13:47:15.768 INFO 19952 --- [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 18.0.1.1
2022-10-21 13:47:15.770 INFO 19952 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to 1 default
2022-10-21 13:47:16.433 INFO 19952 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 9090 (http)
```



# — @SpringBootApplication

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

1개 사용 위치

@SpringBootApplication

```
public class DemoApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(DemoApplication.class, args);
```

```
    }
```

```
}
```

# — @SpringBootApplication

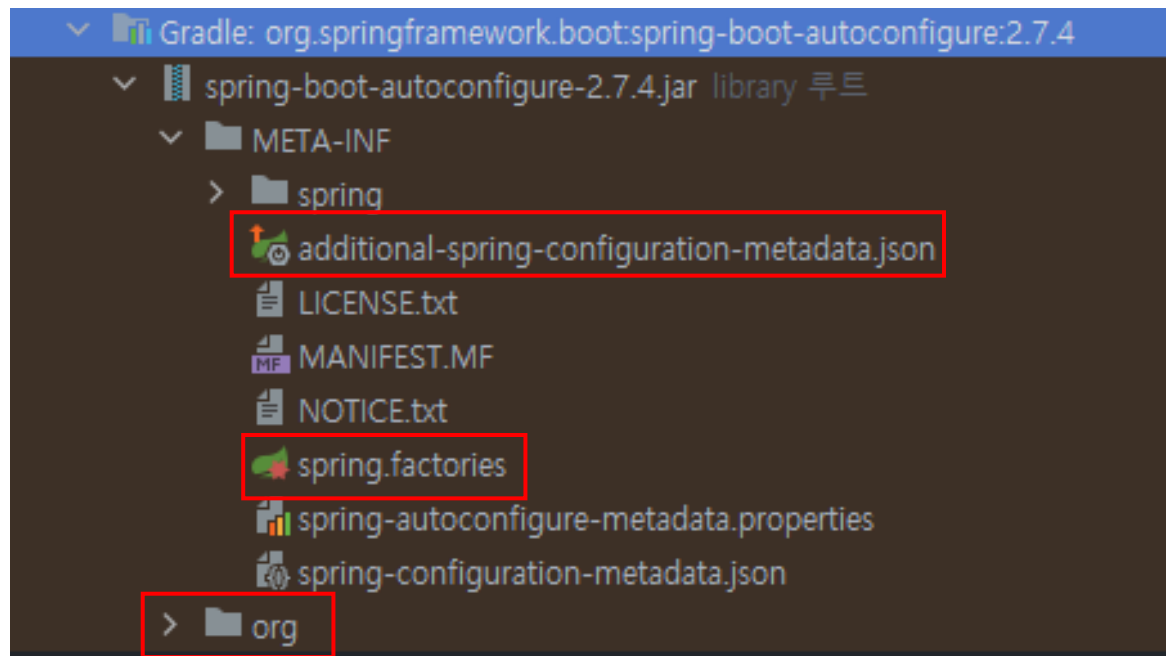
```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
    @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {
```

Exclude specific auto-configuration classes such that they will never be applied.

반환: the classes to exclude

```
@AliasFor(annotation = EnableAutoConfiguration.class)
Class<?>[] exclude() default {};
```

# – @SpringBootApplication



- META-INF/spring.factories – 자동 설정 대상 클래스 목록으로 @EnableAutoConfiguration 사용시, 해당 클래스들은 자동 설정 대상이 된다
- META-INF/spring.configuration-metadata.json – 자동 설정에 사용할 property 정의 파일
- Org/springframework/boot/autoconfigure – 미리 구현해 놓은 자동 설정 리스트

# — @SpringBootApplication

```
},  
{  
  "name": "server.port",  
  "defaultValue": 8080  
},  
{  
  "name": "server.reactive.session.cookie.domain",  
  "description": "Domain for the cookie."  
},
```