Chapter 04.스프링 부트 이해하기 - 02.스프링 이니셜라이저

스프링 이니셜라이저

1. 스프링 이니셜라이저란?

- `스프링 이니셜라이저(Spring initializr)`는 스프링 부트 기반의 애플리케이션을 쉽게 만들 수 있게 도와주는 웹 애플리케이션
- 스프링 이니셜라이저를 사용해서 애플리케이션을 구성하면 초기 프로젝트 구성 시에 들어가는 시간과 노력을 줄여준다

---

2. 스프링 이니셜라이저를 이용하여 프로젝트 생성하기

- 스프링 이니셜라이저를 사용해 프로젝트를 만드는 법은 여러가지이다

2.1. CLI 방식

- 터미널에서 아래와 같이 입력한다

```
$ curl https://start.spring.io
```

- 아래와 같이 터미널에 표시된다

```
(base) ➜  ~ curl https://start.spring.io
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
:: Spring Initializr ::  https://start.spring.io

This service generates quickstart projects that can be easily customized.
Possible customizations include a project's dependencies, Java version, and
build system or build structure. See below for further details.

The services uses a HAL based hypermedia format to expose a set of resources
to interact with. If you access this root resource requesting application/json
as media type the response will contain the following links:
+-----------------+-----------------------------------------+
| Rel             | Description                             |
+-----------------+-----------------------------------------+
| gradle-build    | Generate a Gradle build file.           |
| gradle-project  | Generate a Gradle based project archive. |
| maven-build     | Generate a Maven pom.xml.               |
| maven-project * | Generate a Maven based project archive.  |
+-----------------+-----------------------------------------+
```

- web-mvc 의존성을 가진 자바 11 기반의 스프링 부트 애플리케이션 다운 방법

```
curl -G https://start.spring.io/starter.zip -d dependencies=web \
      -d javaVersion=11 -o demo.zip
```

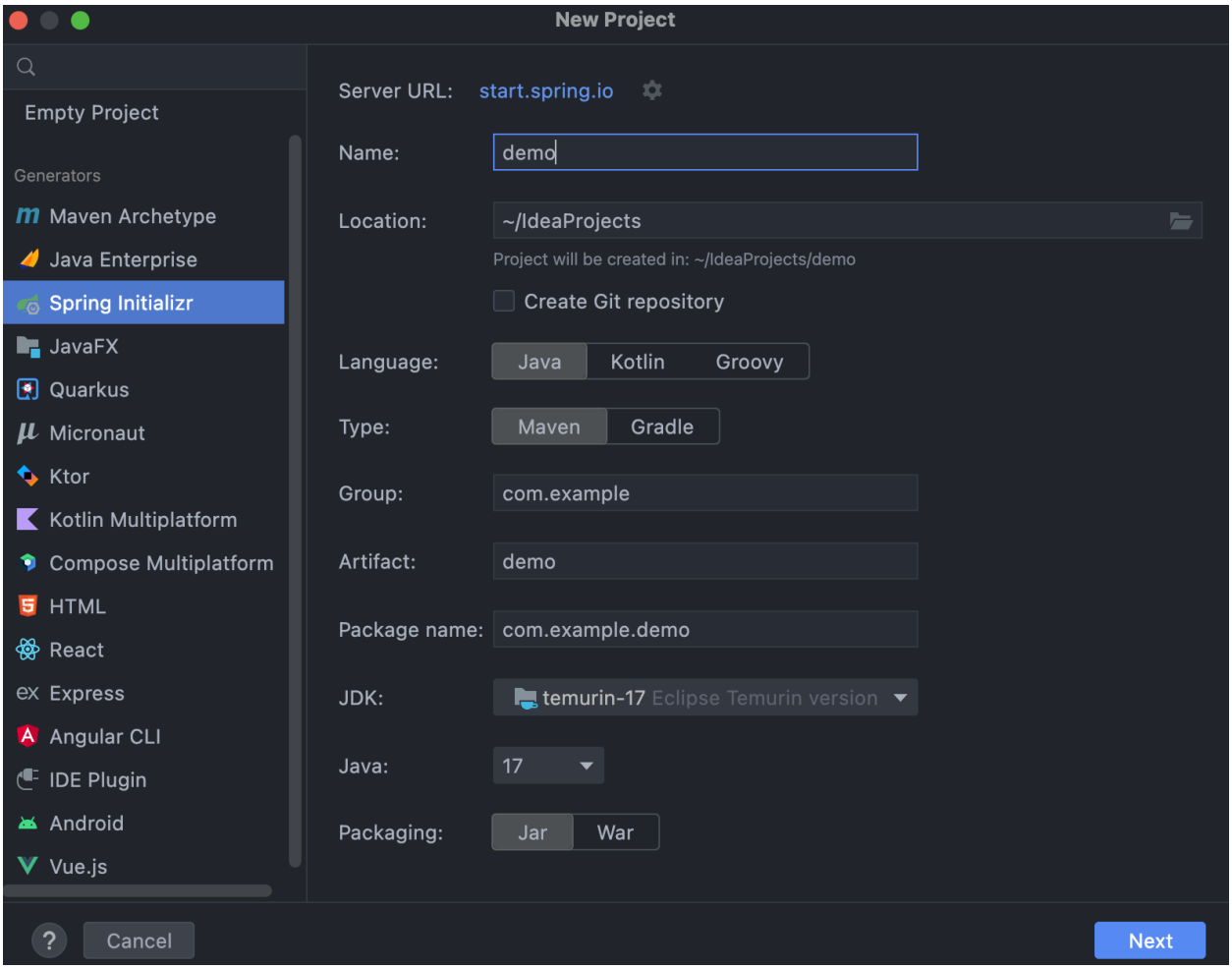
- web-mvc, data-jpa 의존성을 가진 스프링 부트 애플리케이션 다운 방법

```
curl -G https://start.spring.io/starter.zip -d dependencies=web,data-jpa \
      -d javaVersion=11 -o demo.zip
```

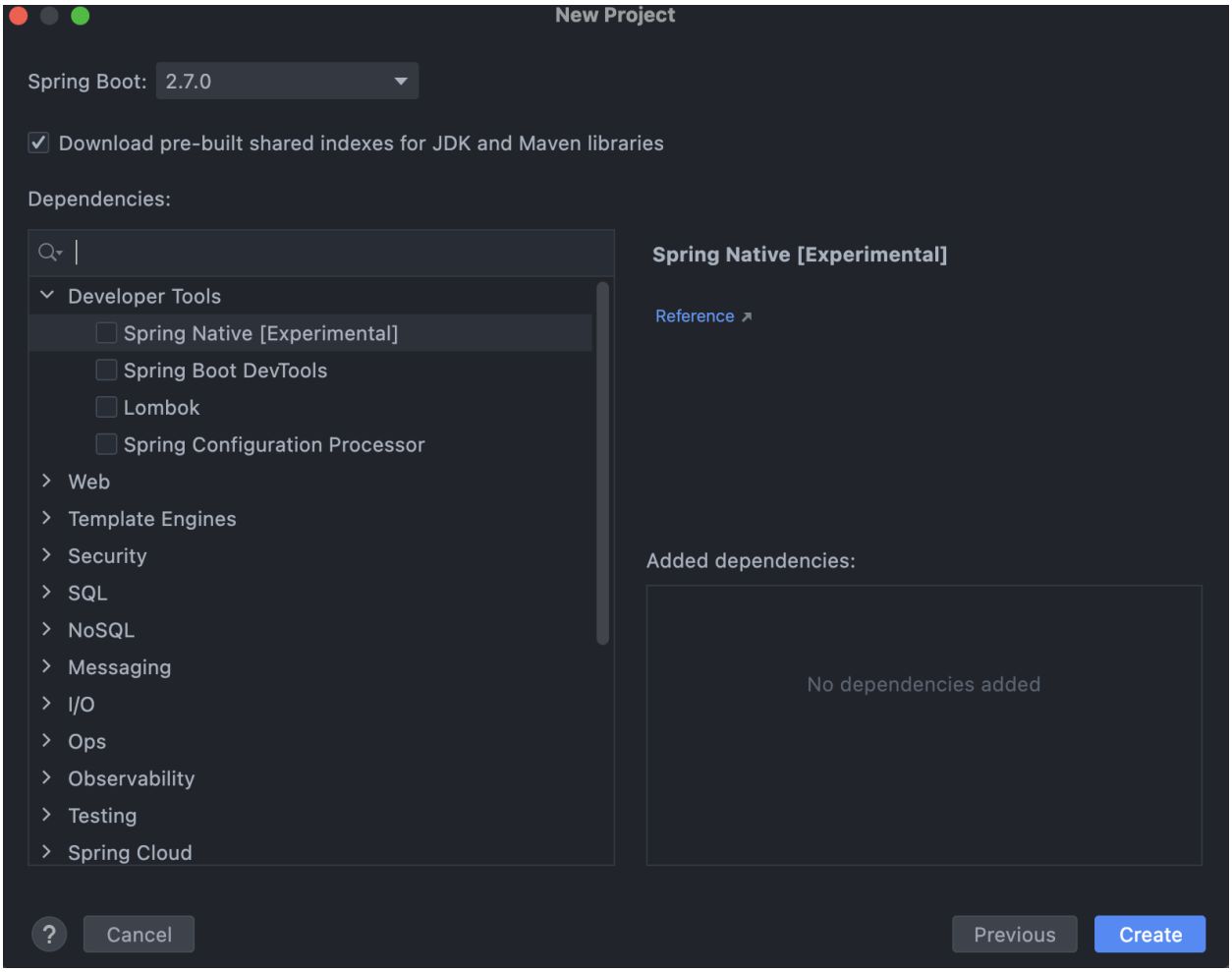
2.2. IDE에서 스프링 이니셜라이저 사용하기

- 이클립스 기반의 무료 IDE인 STS(Spring Tool Suite)나 유료 제품인 IntelliJ Ultimate 버전에서 스프링 이니셜라이저 연동을 제공

- IntelliJ의 스프링 이니셜라이저 1



- IntelliJ의 스프링 이니셜라이저 2



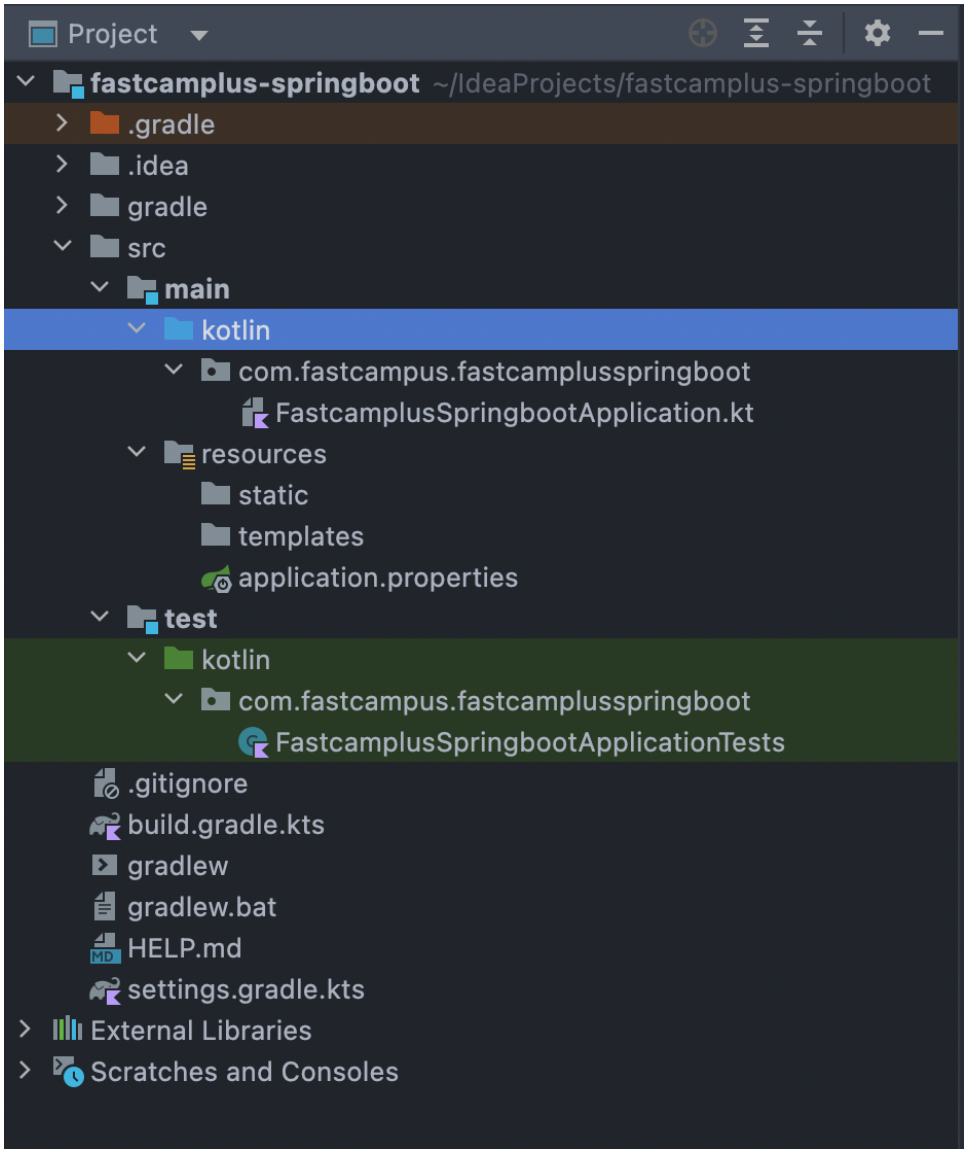
2.3. 스프링 이니셜라이저 웹

- 웹브라우저에서 `https://start.spring.io` 접속
- 프로젝트는 그레이들(Gradle), 언어는 코틀린(Kotlin), 스프링 부트 버전은 기본값을 사용하고 의존성(Dependencies)은 스프링 웹(Spring Web)을 추가한 뒤 하단의 GENERATE

버튼을 눌러서 스프링 부트 프로젝트를 다운로드 후 IntelliJ에서 세팅

프로젝트 구조 설명

- 스프링 이니셜라이저로 생성한 프로젝트의 구조는 그림과 같다



- 스프링 이니셜라이저로 만든 프로젝트의 핵심 패키지와 파일에 대한 설명

패키지 및 파일명	**설명**

gradle/wrapper	프로젝트에서 사용하는 그레이들 래퍼에 대한 설정 정보와 빌드를 위한 래퍼 jar를 포함
src/main/kotlin	.kt 확장자를 가지는 코틀린 소스 코드들을 포함
src/main/resources	프로젝트내의 리소스 파일들을 포함. 여기서 말하는 리소스는 html, css, xml, yml 등이 해당함
application.properties	스프링 부트 애플리케이션 각 설정에 해당하는 값을 주입하는 설정 파일
src/main/test	애플리케이션 테스트에 관련된 소스 코드들을 포함합니다. 일반적으로 하위 패키지는 src/main/kotlin과 동일하게 생성함
build.gradle.kts	코틀린 DSL로 작성된 그레이들 빌드 설정 파일이며 빌드시 필요한 의존성과 플러그인, 리파지토리 등 빌드 전반에 대한 설정을 관리
gradlew	유닉스 환경용 그레이들 래퍼 스크립트입니다. 실행시 gradle/wrapper에 포함된 jar 파일을 사용하여 그레이들 태스크를 실행
gradlew.bat	gradlew와 동일하며 윈도우 환경용 배치 스크립트
settings.gradle.kts	코틀린 DSL로 작성되었으며 프로젝트에 대한 구조를 설정하는 파일 입니다. 설정을 통해 멀티 모듈 프로젝트를 설정할 수 있음

Hello World 애플리케이션 만들기

- build.gradle.kts 열어보기

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

plugins {
  id("org.springframework.boot") version "2.7.0"
  id("io.spring.dependency-management") version "1.0.11.RELEASE"
  kotlin("jvm") version "1.6.21"
  kotlin("plugin.spring") version "1.6.21"
}

group = "com.fastcampus"
version = "0.0.1-SNAPSHOT"
java.sourceCompatibility = JavaVersion.VERSION_17

repositories {
  mavenCentral()
}

dependencies {
```

```
  implementation("org.springframework.boot:spring-boot-starter-web")
  implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
  implementation("org.jetbrains.kotlin:kotlin-reflect")
  implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
  testImplementation("org.springframework.boot:spring-boot-starter-test")
}

tasks.withType<KotlinCompile> {
  kotlinOptions {
    freeCompilerArgs = listOf("-Xjsr305=strict")
    jvmTarget = "17"
  }
}

tasks.withType<Test> {
  useJUnitPlatform()
}
```

- **build.gradle.kts** 는 스프링 이니셜라이저에서 코틀린과 그레이들을 선택하면 자동으로 생성해주는 파일입니다. build.gradle.kts에서는 현재 프로젝트의 의존성과 태스크 그리고 플러그인 등을 관리합니다. 내부 코드는 코틀린으로 작성되어 있기 때문에 자동 완성 기능과 컴파일 오류 체크 같은 IDE의 기능을 사용할 수 있다는것이 코틀린 DSL의 장점 입니다

- HelloWorldApplication.kt

```
package com.fastcampus.fastcamplusspringboot

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class HelloWorldApplication

fun main(args: Array<String>) {
  runApplication<HelloWorldApplication>(*args)
}
```

- 가장 먼저 보실 **@SpringBootApplication**은 현재 애플리케이션이 스프링 부트 애플리케이션이라는 것을 나타냅니다. 또한 다른 애노테이션을 내부에 포함하는데 이를 `메타-애노테이션`이라고 합니다. 그 중에서 핵심 애노테이션은 @EnableAutoConfiguration, @SpringBootConfiguration, @ComponentScan 입니다

- 세가지 애노테이션들이 하는 일

애노테이션 이름	설명
@EnableAutoConfiguration	스프링 부트의 자동 설정을 활성화 하는 애노테이션. 자동 설정에 대한 자세한 설명은 스프링 부트의 자동 설정 시간 참고
@ComponentScan	@Component를 선언한 오브젝트가 있는 패키지를 스캔하도록 활성화함
@SpringBootConfiguration 혹은 @Configuration	추가 적인 설정 클래스를 가져오거나 정의된 빈(Bean)을 스프링 컨텍스트에 추가한다

- 그다음 메인 함수 내부의 `runApplication` *함수는* 스프링 부트 애플리케이션의 시작점 입니다.
- runApplication은 내부적으로 `SpringApplication.run`이라는 자바 메서드를 대체 합니다

- 정상적으로 애플리케이션이 실행되는지 확인하기 위해 웹 브라우저를 띄워서 아래의 url로 접속합니다.

```
https://localhost:8080
```

- 정상적으로 Hello, World를 확인했다면 성공입니다.