Chapter 01.코틀린 소개 - 04.스프링의 코틀린 지원

스프링의 코틀린 지원

1. 공식 문서

- https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#mvc-servlet
- 공식 문서의 샘플 코드를 보면 Kotlin 탭을 제공

Java | Kotlin

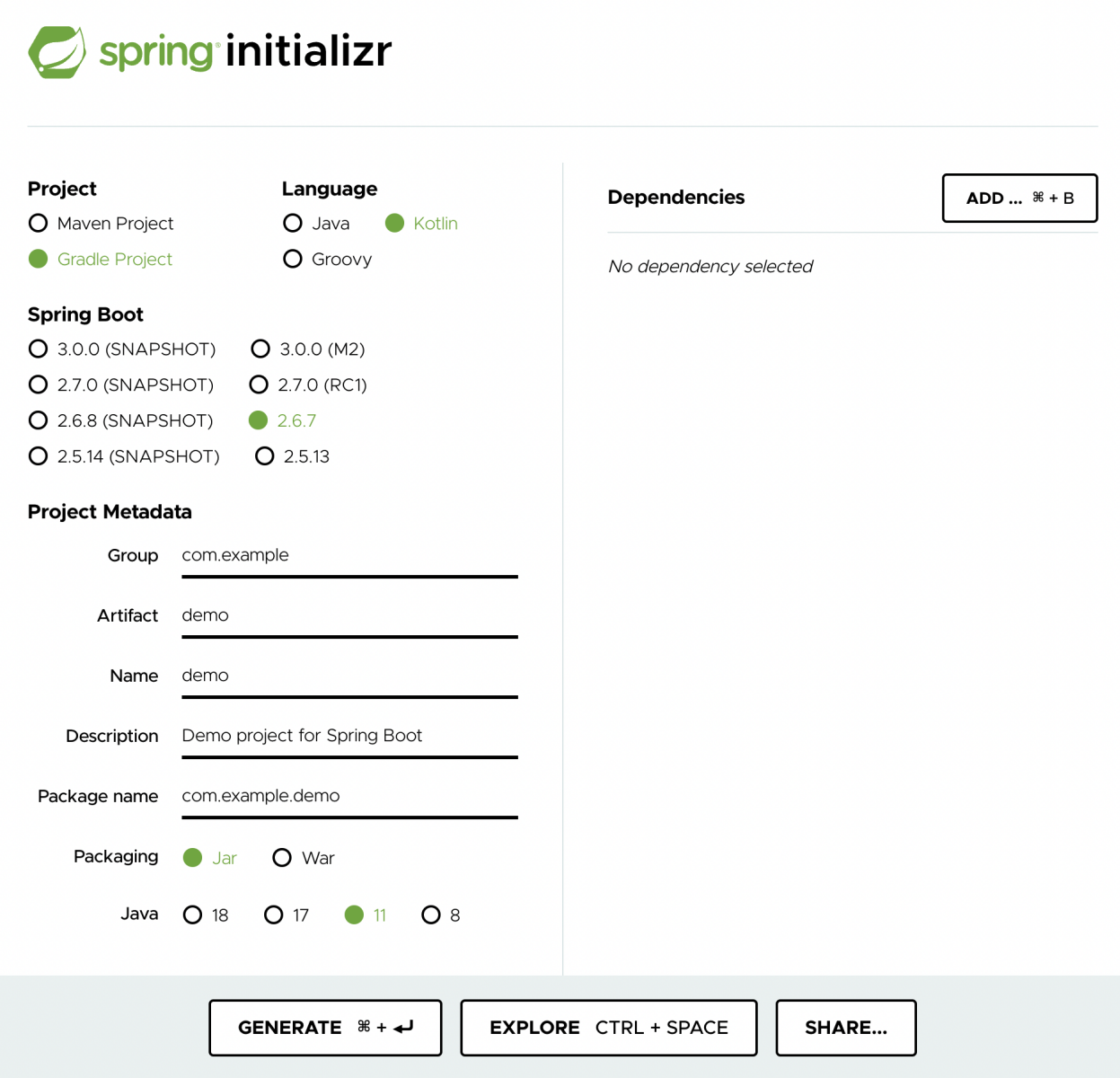
```kotlin
import org.springframework.ui.set

@Controller
class HelloController {

    @GetMapping("/hello")
    fun handle(model: Model): String {
        model["message"] = "Hello World!"
        return "index"
    }
}
```

2. Spring initializr

- 기본 언어로 코틀린을 선택할 수 있고 코틀린인 경우 Gradle Project를 선택하면 빌드 설정을 `Kotlin DSL` 기반으로 생성해준다



- Spring initialzr를 통해 생성된 build.gralde.kts

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

plugins {
  id("org.springframework.boot") version "2.6.7"
  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.example"
version = "0.0.1-SNAPSHOT"
java.sourceCompatibility = JavaVersion.VERSION_11

repositories {
  mavenCentral()
}

dependencies {
  implementation("org.springframework.boot:spring-boot-starter")
  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 = "11"
  }
}

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

- 코틀린 스프링 프로젝트에서 필수적인 플러그인
  - `kotlin("plugin.spring")`
- 코틀린 스프링 프로젝트에서 필수적인 의존성
  - `org.jetbrains.kotlin:kotlin-reflect`
  - `org.jetbrains.kotlin:kotlin-stdlib`
- 이외에도 `plugin.jpa`, `jackson-module-kotlin` 등 프로젝트를 구성하면서 필요한 플러그인과 코틀린 의존성이 있고 Spring initialzr에서 프로젝트를 구성할 경우 자동으로 세팅해준다

---

3. 스프링 부트

- 아래 예제와 같은 방식으로 스프링 부트 애플리케이션을 실행할 수 있다

```
@SpringBootApplication
class DemoApplication

// 탑-레벨 함수이므로 클래스 바깥에서 호출
fun main(args: Array<String>) {
    runApplication<DemoApplication>(*args)
}
```

---

4. @ConfigurationProperties

- 스프링 애플리케이션에 지정한 설정을 기반으로 설정 클래스를 만들때 `@ConstructorBinding` 을 사용하면 **setter가 아닌 생성자를 통해 바인딩 하므로** 불변 객체를 쉽게 생성할 수 있다

```
@ConstructorBinding
@ConfigurationProperties("example.kotlin")
data class KotlinExampleProperties(
        val name: String,
        val description: String,
        val myService: MyService) {

    data class MyService(
            val apiToken: String,
            val uri: URI
    )
}
```

---

5. 테스트 지원

- 기본 제공되는 Junit5 기반의 테스트를 특별한 설정 없이 그대로 사용이 가능하다
- 모의 객체를 만들어 테스트하려면 Mockito 대신 MockK를 사용할 수 있다

- Mockito에서 제공하는 `@MockBean` `@SpyBean`을 대체하는 SpringMockK의 `@MockkBean`, `@SpykBean`

```
@ExtendWith(SpringExtension::class)
@WebMvcTest
class GreetingControllerTest {

    @MockkBean
    private lateinit var greetingService: GreetingService

    @Autowired
    private lateinit var controller: GreetingController

    @Test
    fun `should greet by delegating to the greeting service`() {
        // Given
        every { greetingService.greet("John") } returns "Hi John"

        // When
        assertThat(controller.greet("John")).isEqualTo("Hi John")

        // Then
        verify { greetingService.greet("John") }
    }
}
```

---

6. 확장함수

- 스프링에서 지원하는 코틀린 API의 대부분은 이러한 확장 기능을 사용해 기존 API에 코틀린 API를 추가

```
// CrudRepositoryExtensions.kt

package org.springframework.data.repository

fun <T, ID> CrudRepository<T, ID>.findByIdOrNull(id: ID): T? {
    return findById(id).orElse(null)
}


//MyService.kt
class MyService(
```

```
  private val myRepository: MyRepository,
) {

  fun findById(id: Long): My? = myRepository.findByIdOrNull(id)
}
```

- 스프링 프로젝트에선 확장함수를 통해 기존 자바 API를 건드리지 않고 쉽게 코틀린 확장 기능을 추가하고 있다

---

7. 코루틴

- 비동기-논블로킹 방식을 선언형으로 구현하기 위한 코틀린 기능
- 스프링 MVC, 스프링 WebFlux 모두 코루틴을 지원하여 의존성만 추가하면 바로 사용 가능

```
// build.gradle.kts

dependencies {
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-reactor")
}
```

- 코루틴이 지원되어 비동기-논블로킹 스타일의 구현을 쉽게 할 수 있다

```
@RestController
class UserController(
  private val userService : UserService,
  private val userDetailService: UserDetailService
) {

  @GetMapping("/{id}")
  suspend fun get(@PathVariable id: Long) : User {
    return userService.getById(id)
  }

  @GetMapping("/users")
  suspend fun gets() = withContext(Dispatchers.IO) {
      val usersDeffered = async { userService.gets() }
      val userDetailsDeffered = async { userDetailService.gets() }

      return UserList(usersDeffered.await(), userDetailsDeffered.await())
```

```
  }

}
```

8. 스프링 Fu

- https://github.com/spring-projects-experimental/spring-fu
- 스프링 부트 설정을 Java DSL 혹은 Kotlin DSL 방식으로 작성 가능한 아직은 `실험적(experimental)` 프로젝트
- Java DSL은 JaFu, Kotlin DSL은 KoFu로 부른다

```
val app = webApplication {
  logging {
    level = LogLevel.DEBUG
  }
  beans {
    bean<SampleService>()
  }
  webMvc {
    port = if (profiles.contains("test")) 8181 else 8080
    router {
      val service = ref<SampleService>()
      GET("/") {
        ok().body(service.generateMessage())
      }
      GET("/api") {
        ok().body(Sample(service.generateMessage()))
      }
    }
    converters {
      string()
      jackson {
        indentOutput = true
      }
    }
  }
}
```

```
data class Sample(val message: String)

class SampleService {
  fun generateMessage() = "Hello world!"
}

fun main() {
  app.run()
}
```