THE DEVELOPER'S CONFERENCE

Trilha – Kotlin

André de Fontana Ignacio
Software Architect

THE DEVELOPER'S
CONFERENCE

Renovando sua stack Spring com Kotlin

Sobre mim

- Java desde 2006 (1.4)
- Spring Framework desde 2008 (2.5)
- Ritmista da Mocidade Alegre desde 2000

@aignacio83

andrefontanaignacio

ignacio83

Agenda

THE DEVELOPER'S CONFERENCE

- História
- Live Coding
- Perguntas

Um pouco de história

Kotlin é anunciado

jul/2011

fev/2012

Kotlin - Open Source

Kotlin 1.0

Spring + Kotlin

fev/2016

mai/2017

Android + Kotlin

Kotlin 1.2

nov/2017

out/2018

Kotlin 1.3

Em Maio de 2019 - Google anuncia que Kotlin é a linguagem preferida para desenvolvimento no Android

Projeto - Stack Java

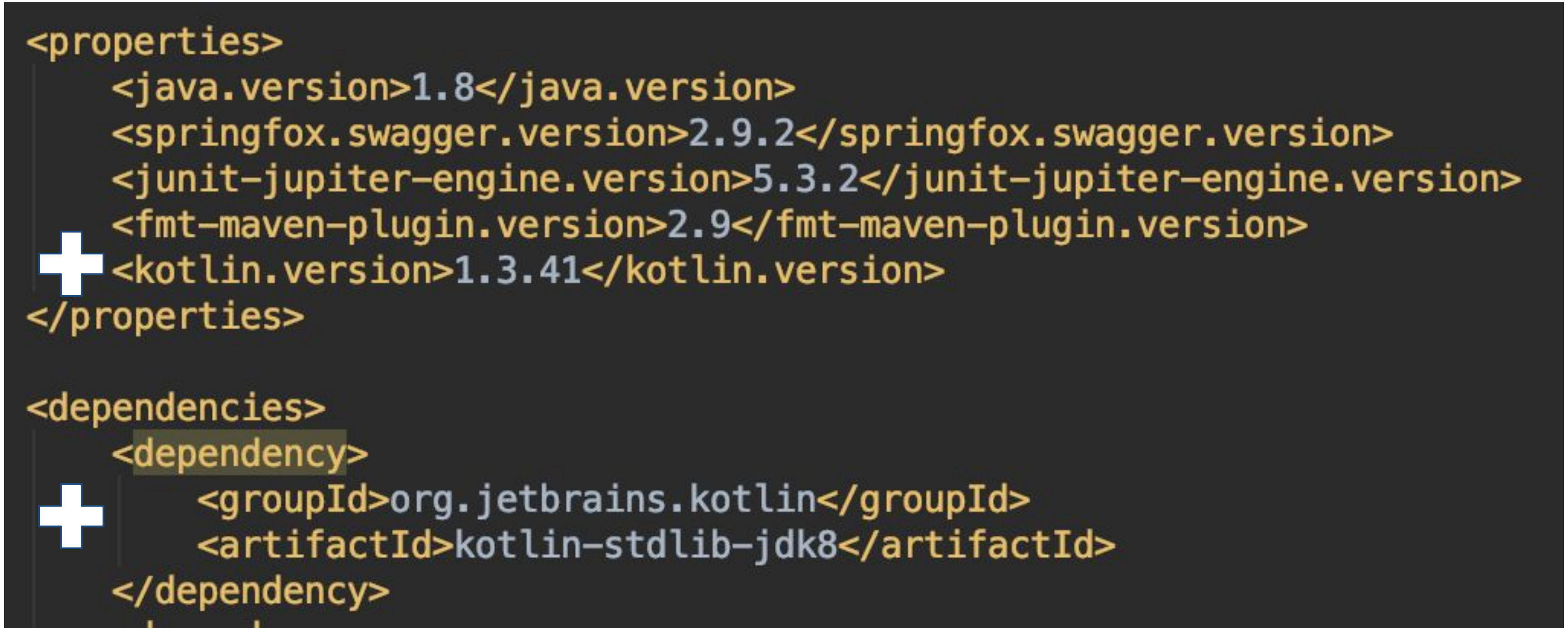
- Java 8
- Maven
- Spring Boot 2.1.6
- Spring Data JPA
- Spring MVC
- Lombok
- Spring Fox (Swagger)
- H2
- JUnit 5
- Mockito
- AssertJ
- Google Java Format

- Controllers
- Contracts/Resources
- Domain
- Services
- Repositories

THE DEVELOPER'S CONFERENCE

Live Coding

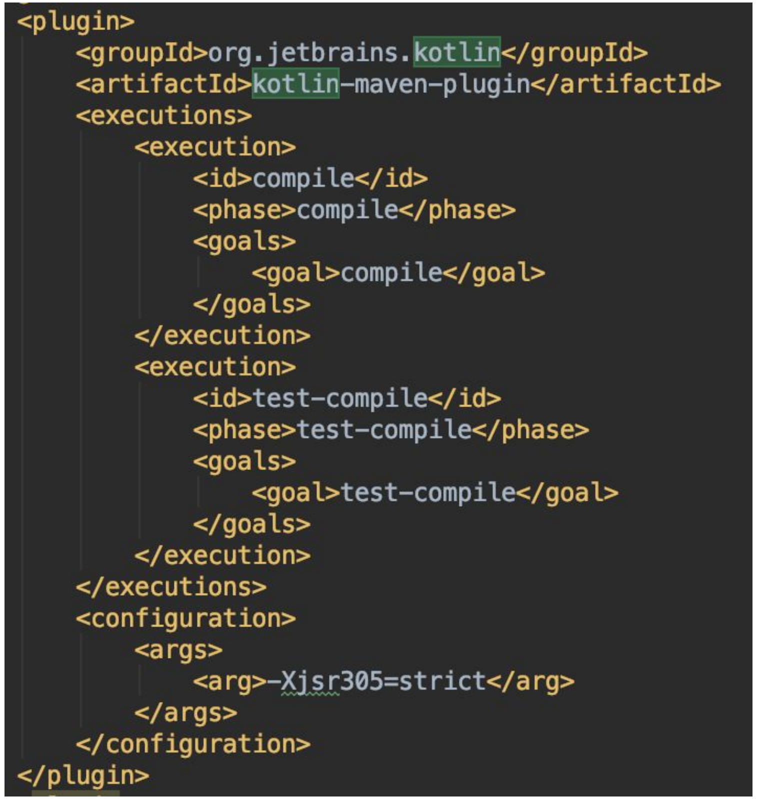
Kotlin libs



```
<properties>
    <java.version>1.8</java.version>
    <springfox.swagger.version>2.9.2</springfox.swagger.version>
    <junit-jupiter-engine.version>5.3.2</junit-jupiter-engine.version>
    <fmt-maven-plugin.version>2.9</fmt-maven-plugin.version>
 +  <kotlin.version>1.3.41</kotlin.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
 +      <artifactId>kotlin-stdlib-jdk8</artifactId>
    </dependency>
```

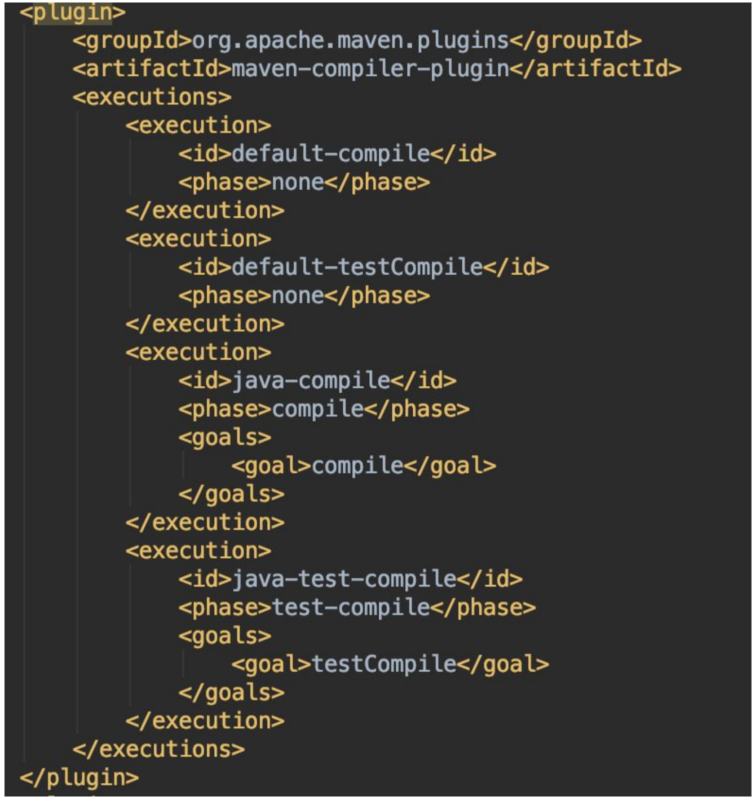
Compilando Kotlin

THE DEVELOPER'S CONFERENCE



```
<plugin>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>compile</id>
            <phase>compile</phase>
            <goals>
                <goal>compile</goal>
            </goals>
        </execution>
        <execution>
            <id>test-compile</id>
            <phase>test-compile</phase>
            <goals>
                <goal>test-compile</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <args>
            <arg>-Xjsr305=strict</arg>
        </args>
    </configuration>
</plugin>
```

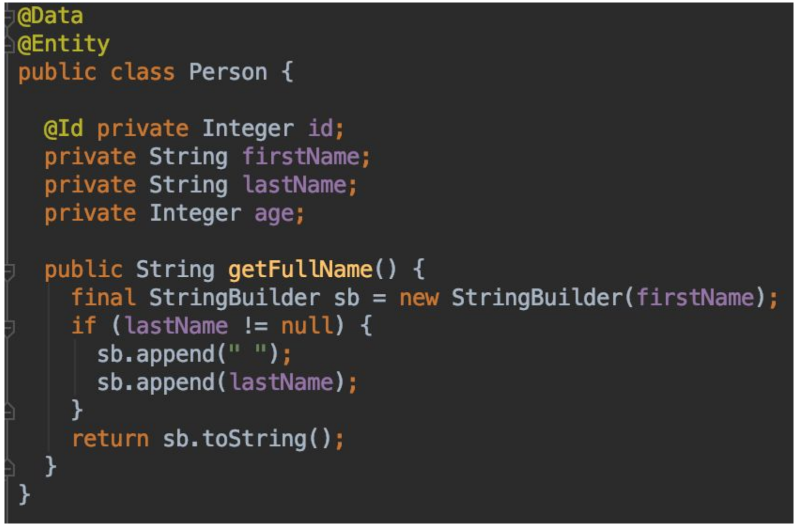
+



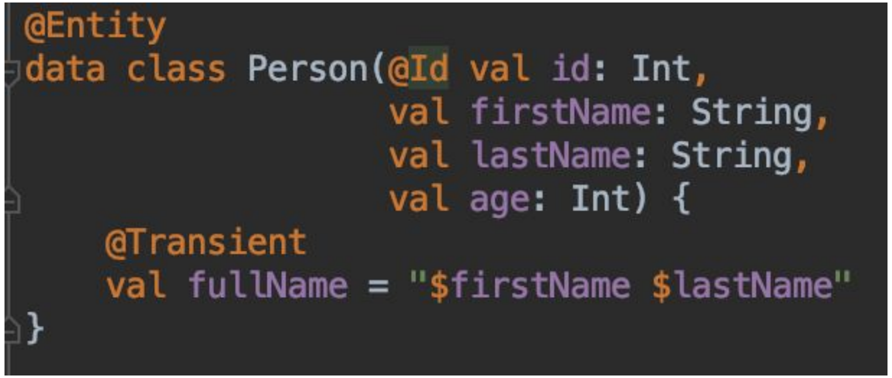
```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <executions>
        <execution>
            <id>default-compile</id>
            <phase>none</phase>
        </execution>
        <execution>
            <id>default-testCompile</id>
            <phase>none</phase>
        </execution>
        <execution>
            <id>java-compile</id>
            <phase>compile</phase>
            <goals>
                <goal>compile</goal>
            </goals>
        </execution>
        <execution>
            <id>java-test-compile</id>
            <phase>test-compile</phase>
            <goals>
                <goal>testCompile</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

+

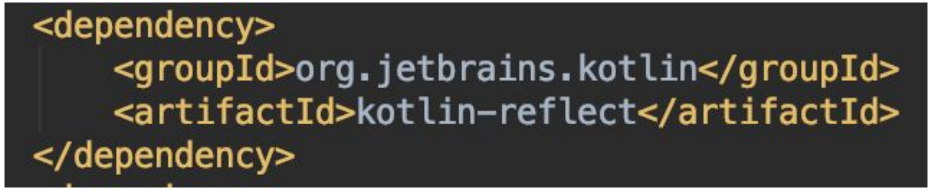
Domain/Entity



```java
@Data
@Entity
public class Person {

  @Id private Integer id;
  private String firstName;
  private String lastName;
  private Integer age;

  public String getFullName() {
    final StringBuilder sb = new StringBuilder(firstName);
    if (lastName != null) {
      sb.append(" ");
      sb.append(lastName);
    }
    return sb.toString();
  }
}
```



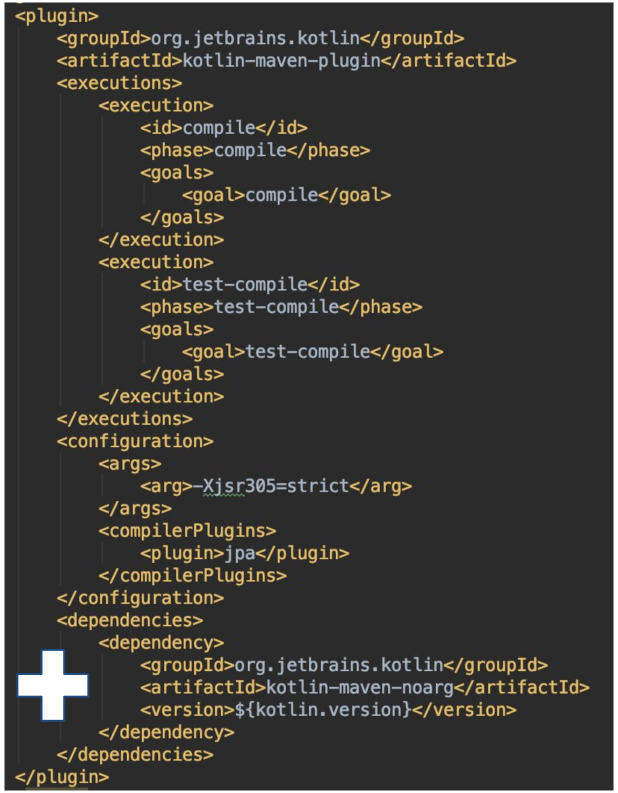
```kotlin
@Entity
data class Person(@Id val id: Int,
                  val firstName: String,
                  val lastName: String,
                  val age: Int) {
    @Transient
    val fullName = "$firstName $lastName"
}
```



```xml
<dependency>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-reflect</artifactId>
</dependency>
```

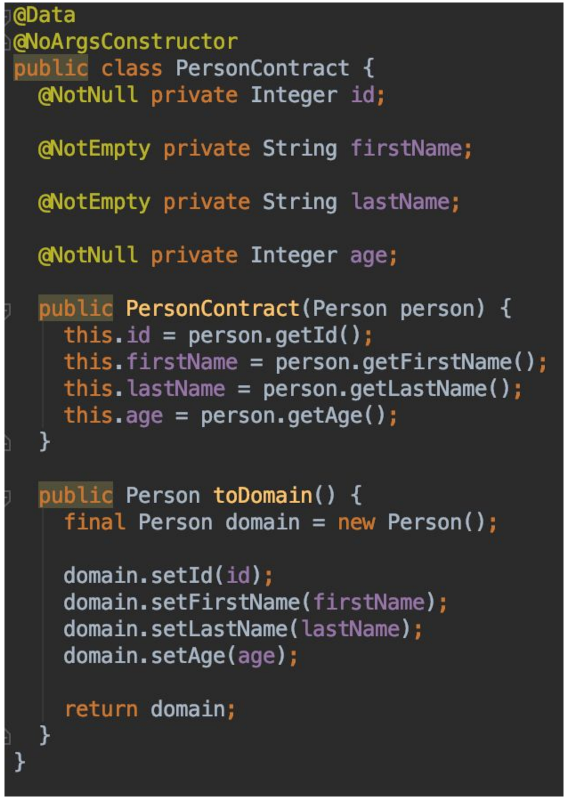
Domain/Entity - JPA plugin

THE DEVELOPER'S CONFERENCE

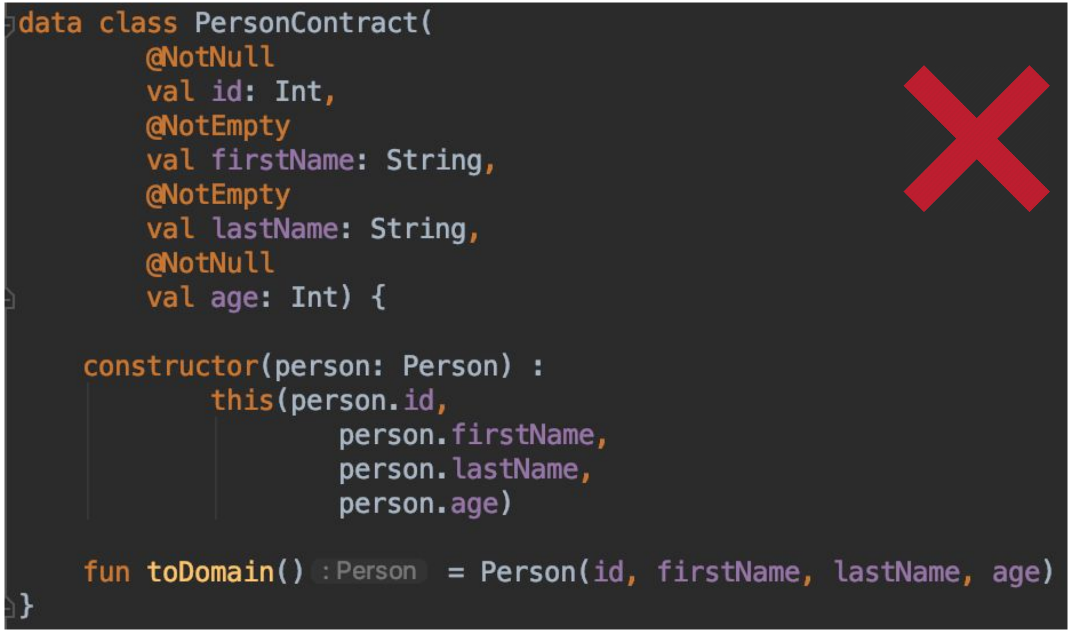


```xml
<plugin>
    <groupId>org.jetbrains.kotlin</groupId>
    <artifactId>kotlin-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>compile</id>
            <phase>compile</phase>
            <goals>
                <goal>compile</goal>
            </goals>
        </execution>
        <execution>
            <id>test-compile</id>
            <phase>test-compile</phase>
            <goals>
                <goal>test-compile</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <args>
            <arg>-Xjsr305=strict</arg>
        </args>
        <compilerPlugins>
            <plugin>jpa</plugin>
        </compilerPlugins>
    </configuration>
    <dependencies>
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-maven-noarg</artifactId>
            <version>${kotlin.version}</version>
        </dependency>
    </dependencies>
</plugin>
```

Contract/Resource



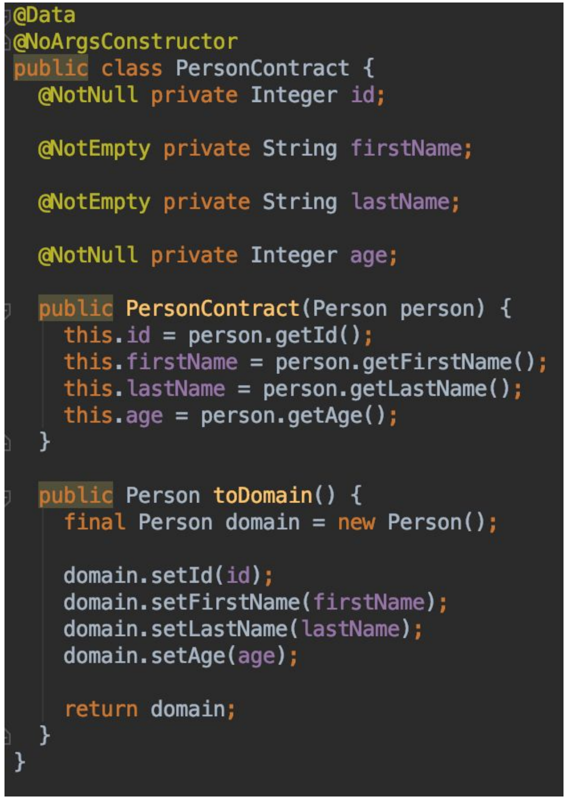
```java
@Data
@NoArgsConstructor
public class PersonContract {
  @NotNull private Integer id;

  @NotEmpty private String firstName;

  @NotEmpty private String lastName;

  @NotNull private Integer age;

  public PersonContract(Person person) {
    this.id = person.getId();
    this.firstName = person.getFirstName();
    this.lastName = person.getLastName();
    this.age = person.getAge();
  }

  public Person toDomain() {
    final Person domain = new Person();

    domain.setId(id);
    domain.setFirstName(firstName);
    domain.setLastName(lastName);
    domain.setAge(age);

    return domain;
  }
}
```



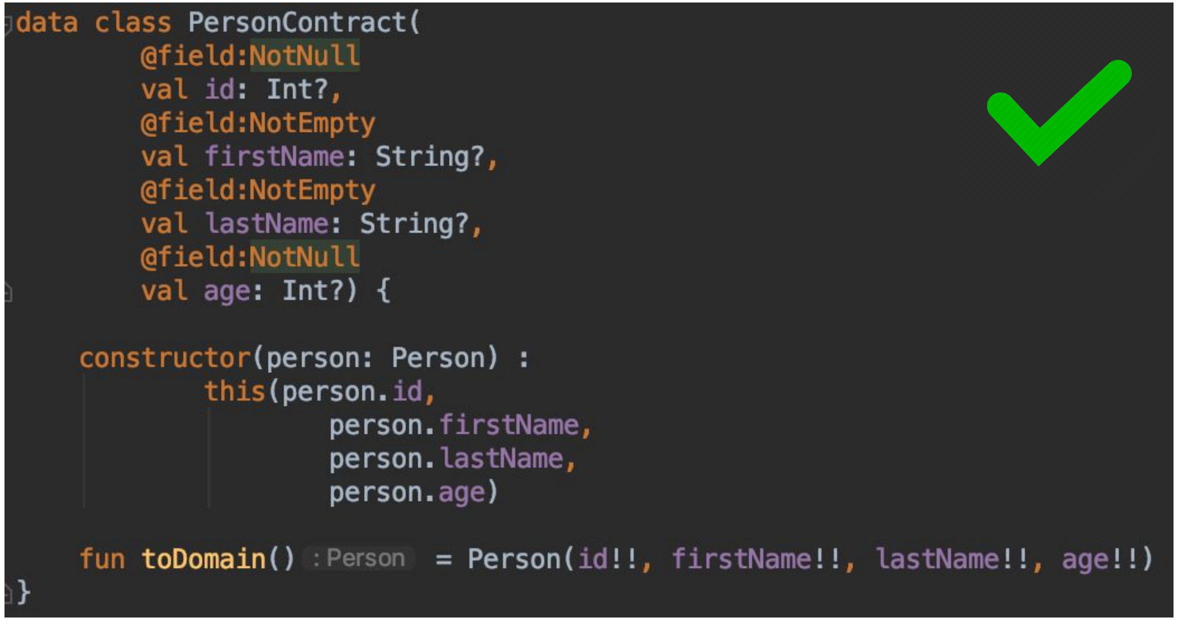
```kotlin
data class PersonContract(
        @NotNull
        val id: Int,
        @NotEmpty
        val firstName: String,
        @NotEmpty
        val lastName: String,
        @NotNull
        val age: Int) {

    constructor(person: Person) :
            this(person.id,
                    person.firstName,
                    person.lastName,
                    person.age)

    fun toDomain() : Person  = Person(id, firstName, lastName, age)
}
```

Contract/Resource

THE DEVELOPER'S CONFERENCE



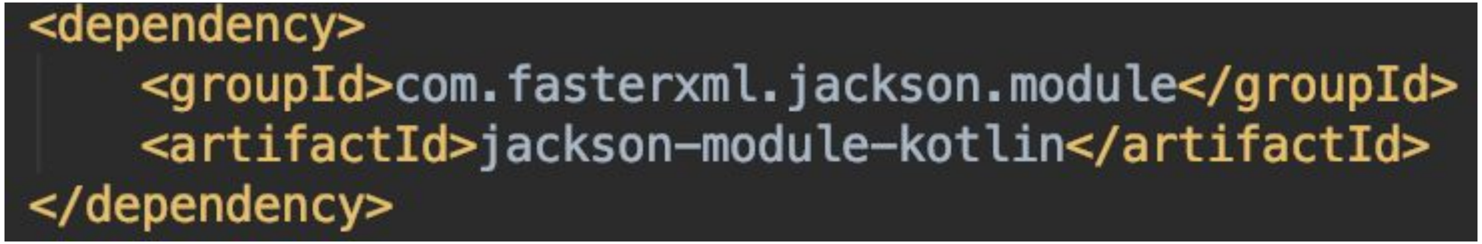
```java
@Data
@NoArgsConstructor
public class PersonContract {
  @NotNull private Integer id;

  @NotEmpty private String firstName;

  @NotEmpty private String lastName;

  @NotNull private Integer age;

  public PersonContract(Person person) {
    this.id = person.getId();
    this.firstName = person.getFirstName();
    this.lastName = person.getLastName();
    this.age = person.getAge();
  }

  public Person toDomain() {
    final Person domain = new Person();

    domain.setId(id);
    domain.setFirstName(firstName);
    domain.setLastName(lastName);
    domain.setAge(age);

    return domain;
  }
}
```



```kotlin
data class PersonContract(
        @field:NotNull
        val id: Int?,
        @field:NotEmpty
        val firstName: String?,
        @field:NotEmpty
        val lastName: String?,
        @field:NotNull
        val age: Int?) {

    constructor(person: Person) :
            this(person.id,
                    person.firstName,
                    person.lastName,
                    person.age)

    fun toDomain() : Person  = Person(id!!, firstName!!, lastName!!, age!!)
}
```

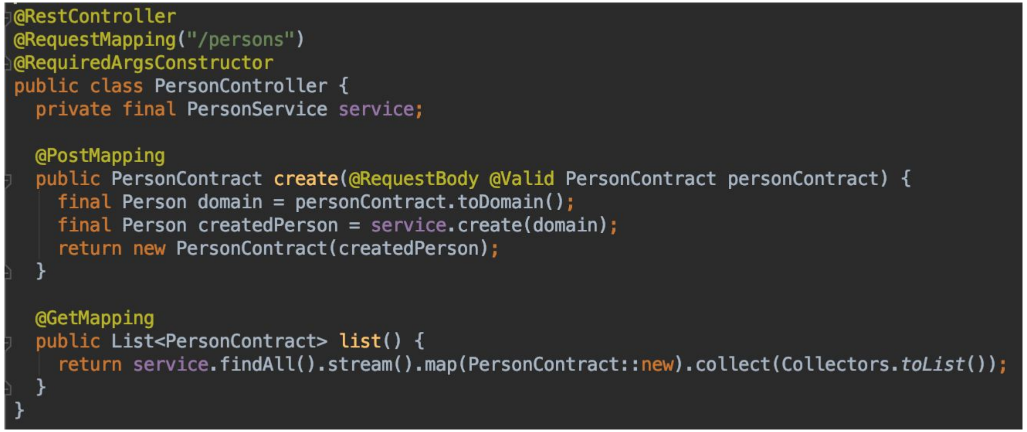
Recomendado

+

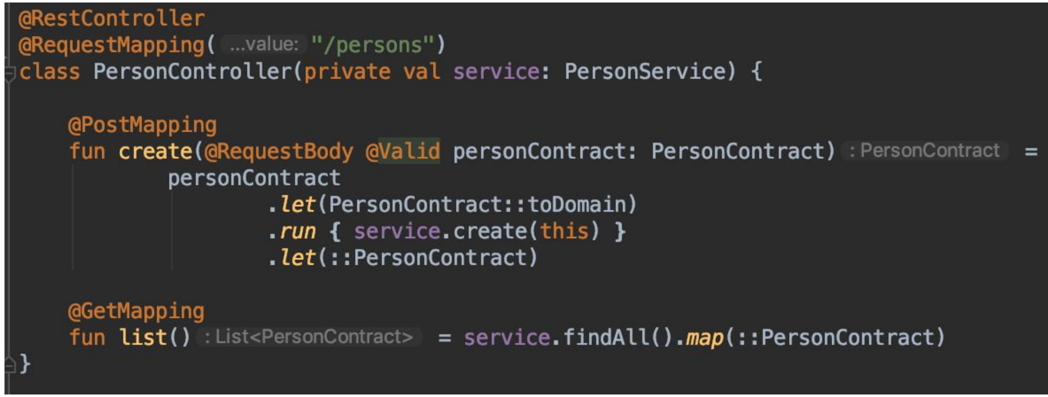


```xml
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-kotlin</artifactId>
</dependency>
```

Controller

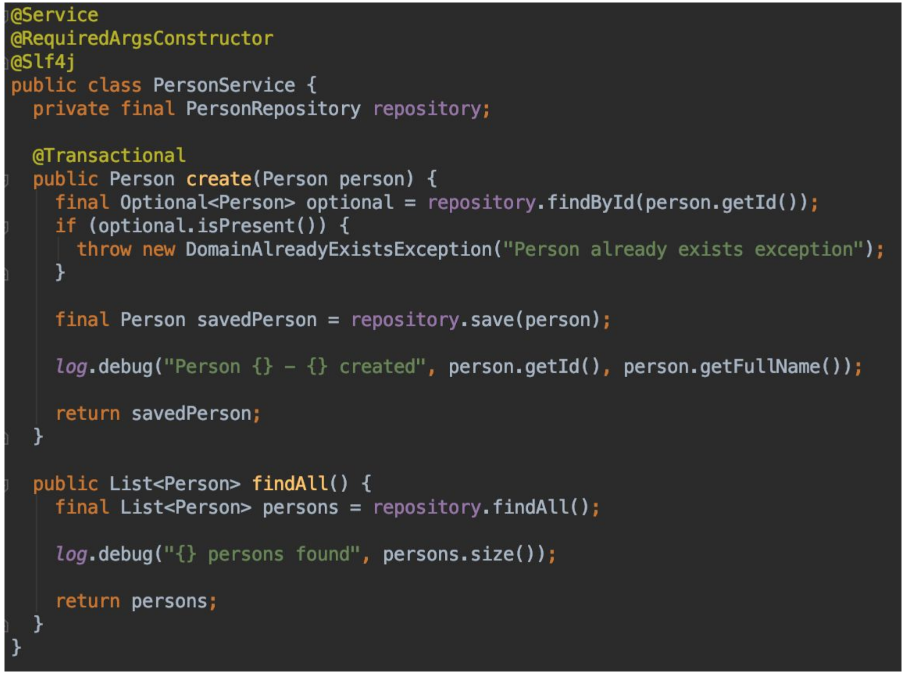


```java
@RestController
@RequestMapping("/persons")
@RequiredArgsConstructor
public class PersonController {
  private final PersonService service;

  @PostMapping
  public PersonContract create(@RequestBody @Valid PersonContract personContract) {
    final Person domain = personContract.toDomain();
    final Person createdPerson = service.create(domain);
    return new PersonContract(createdPerson);
  }

  @GetMapping
  public List<PersonContract> list() {
    return service.findAll().stream().map(PersonContract::new).collect(Collectors.toList());
  }
}
```

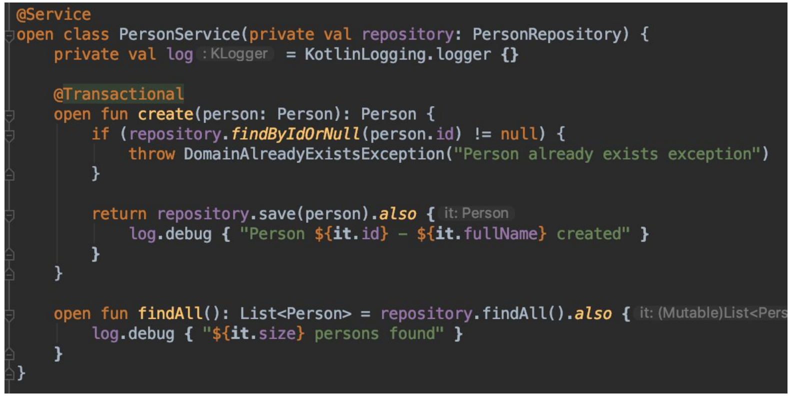


```kotlin
@RestController
@RequestMapping( ...value: "/persons")
class PersonController(private val service: PersonService) {

    @PostMapping
    fun create(@RequestBody @Valid personContract: PersonContract) : PersonContract =
            personContract
                    .let(PersonContract::toDomain)
                    .run { service.create(this) }
                    .let(::PersonContract)

    @GetMapping
    fun list() : List<PersonContract> = service.findAll().map(::PersonContract)
}
```

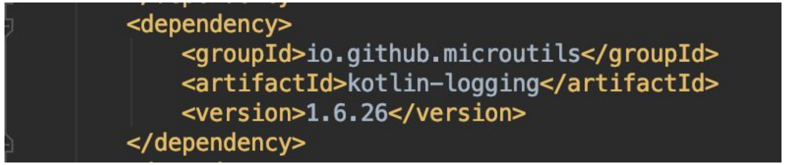
Service



```java
@Service
@RequiredArgsConstructor
@Slf4j
public class PersonService {
  private final PersonRepository repository;

  @Transactional
  public Person create(Person person) {
    final Optional<Person> optional = repository.findById(person.getId());
    if (optional.isPresent()) {
      throw new DomainAlreadyExistsException("Person already exists exception");
    }

    final Person savedPerson = repository.save(person);

    log.debug("Person {} - {} created", person.getId(), person.getFullName());

    return savedPerson;
  }

  public List<Person> findAll() {
    final List<Person> persons = repository.findAll();

    log.debug("{} persons found", persons.size());

    return persons;
  }
}
```



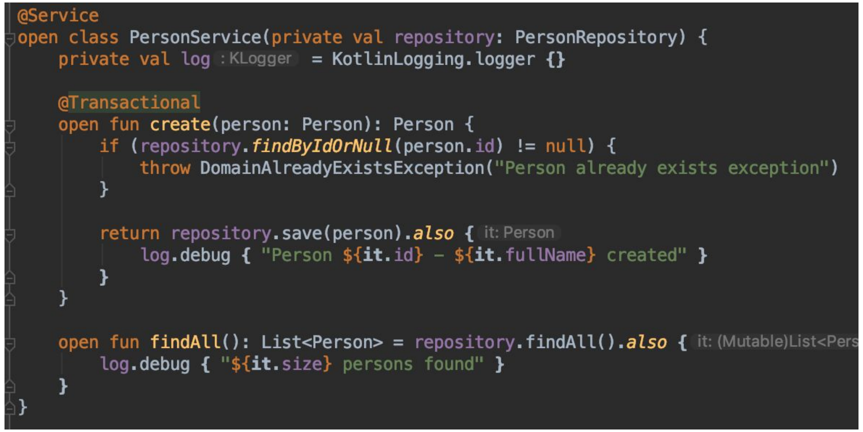
```kotlin
@Service
open class PersonService(private val repository: PersonRepository) {
    private val log :KLogger = KotlinLogging.logger {}

    @Transactional
    open fun create(person: Person): Person {
        if (repository.findByIdOrNull(person.id) != null) {
            throw DomainAlreadyExistsException("Person already exists exception")
        }

        return repository.save(person).also { it: Person
            log.debug { "Person ${it.id} - ${it.fullName} created" }
        }
    }

    open fun findAll(): List<Person> = repository.findAll().also { it: (Mutable)List<Pers
        log.debug { "${it.size} persons found" }
    }
}
```



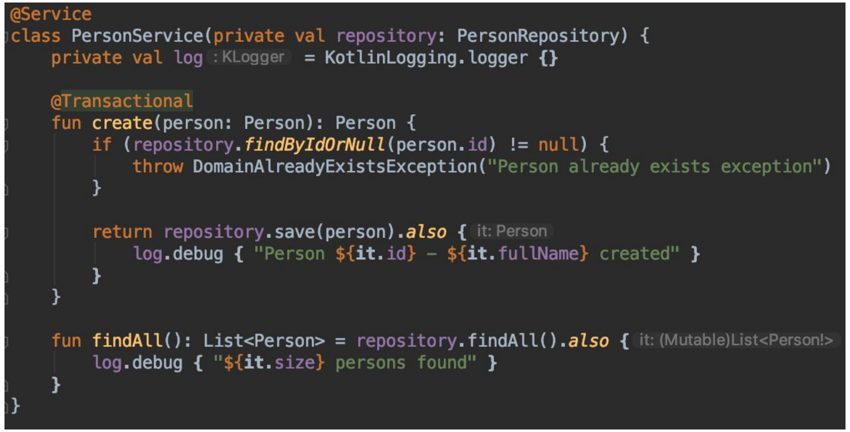
```xml
<dependency>
    <groupId>io.github.microutils</groupId>
    <artifactId>kotlin-logging</artifactId>
    <version>1.6.26</version>
</dependency>
```

Plugin allopen

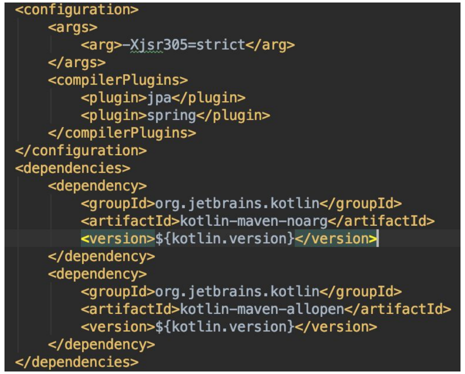
THE DEVELOPER'S CONFERENCE



```kotlin
@Service
open class PersonService(private val repository: PersonRepository) {
    private val log : KLogger  = KotlinLogging.logger {}

    @Transactional
    open fun create(person: Person): Person {
        if (repository.findByIdOrNull(person.id) != null) {
            throw DomainAlreadyExistsException("Person already exists exception")
        }

        return repository.save(person).also {  it: Person
            log.debug { "Person ${it.id} - ${it.fullName} created" }
        }
    }

    open fun findAll(): List<Person> = repository.findAll().also {  it: (Mutable)List<Pers
        log.debug { "${it.size} persons found" }
    }
}
```

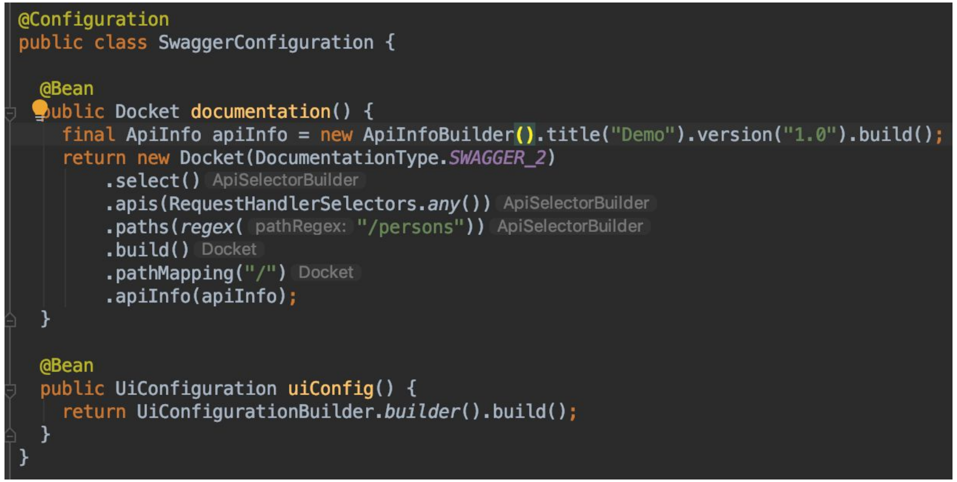


```kotlin
@Service
class PersonService(private val repository: PersonRepository) {
    private val log : KLogger  = KotlinLogging.logger {}

    @Transactional
    fun create(person: Person): Person {
        if (repository.findByIdOrNull(person.id) != null) {
            throw DomainAlreadyExistsException("Person already exists exception")
        }

        return repository.save(person).also {  it: Person
            log.debug { "Person ${it.id} - ${it.fullName} created" }
        }
    }

    fun findAll(): List<Person> = repository.findAll().also {  it: (Mutable)List<Person!>
        log.debug { "${it.size} persons found" }
    }
}
```

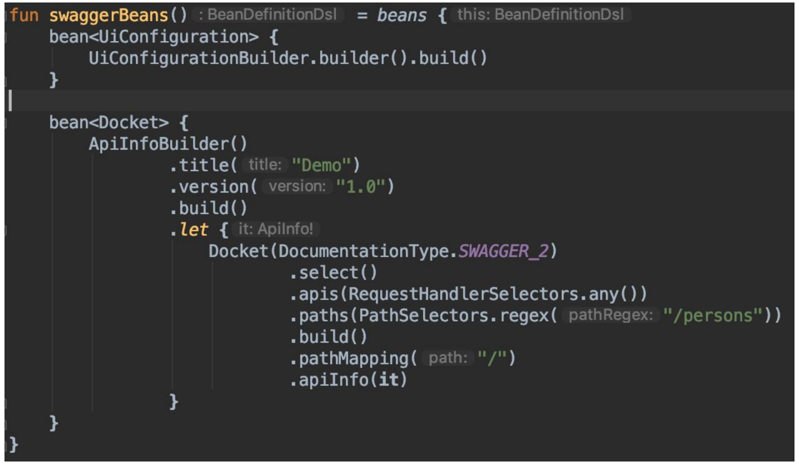


```xml
<configuration>
    <args>
        <arg>-Xjsr305=strict</arg>
    </args>
    <compilerPlugins>
        <plugin>jpa</plugin>
        <plugin>spring</plugin>
    </compilerPlugins>
</configuration>
<dependencies>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-maven-noarg</artifactId>
        <version>${kotlin.version}</version>
    </dependency>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-maven-allopen</artifactId>
        <version>${kotlin.version}</version>
    </dependency>
</dependencies>
```

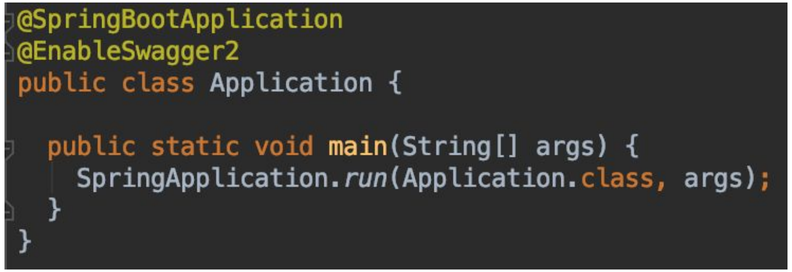
Configuration



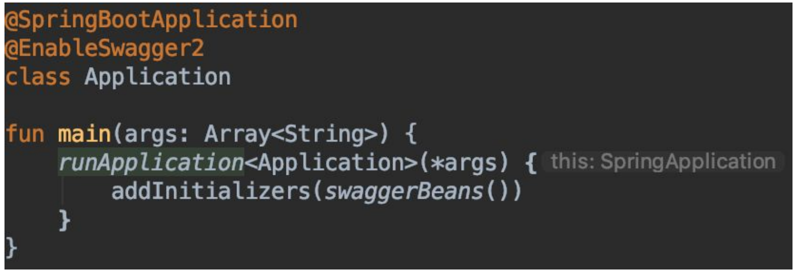
```
@Configuration
public class SwaggerConfiguration {

  @Bean
  public Docket documentation() {
    final ApiInfo apiInfo = new ApiInfoBuilder().title("Demo").version("1.0").build();
    return new Docket(DocumentationType.SWAGGER_2)
        .select() ApiSelectorBuilder
        .apis(RequestHandlerSelectors.any()) ApiSelectorBuilder
        .paths(regex( pathRegex: "/persons")) ApiSelectorBuilder
        .build() Docket
        .pathMapping("/") Docket
        .apiInfo(apiInfo);
  }

  @Bean
  public UiConfiguration uiConfig() {
    return UiConfigurationBuilder.builder().build();
  }
}
```



```
fun swaggerBeans() : BeanDefinitionDsl = beans { this: BeanDefinitionDsl
    bean<UiConfiguration> {
        UiConfigurationBuilder.builder().build()
    }

    bean<Docket> {
        ApiInfoBuilder()
                .title( title: "Demo")
                .version( version: "1.0")
                .build()
                .let { it: ApiInfo!
                    Docket(DocumentationType.SWAGGER_2)
                            .select()
                            .apis(RequestHandlerSelectors.any())
                            .paths(PathSelectors.regex( pathRegex: "/persons"))
                            .build()
                            .pathMapping( path: "/")
                            .apiInfo(it)
                }
    }
}
```

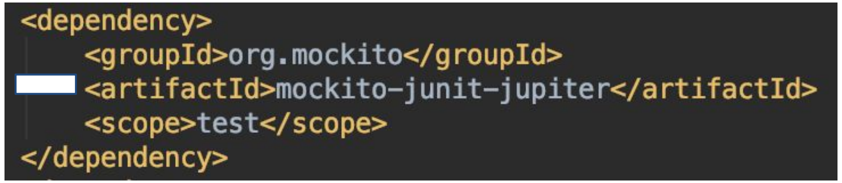


```
@SpringBootApplication
@EnableSwagger2
public class Application {

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

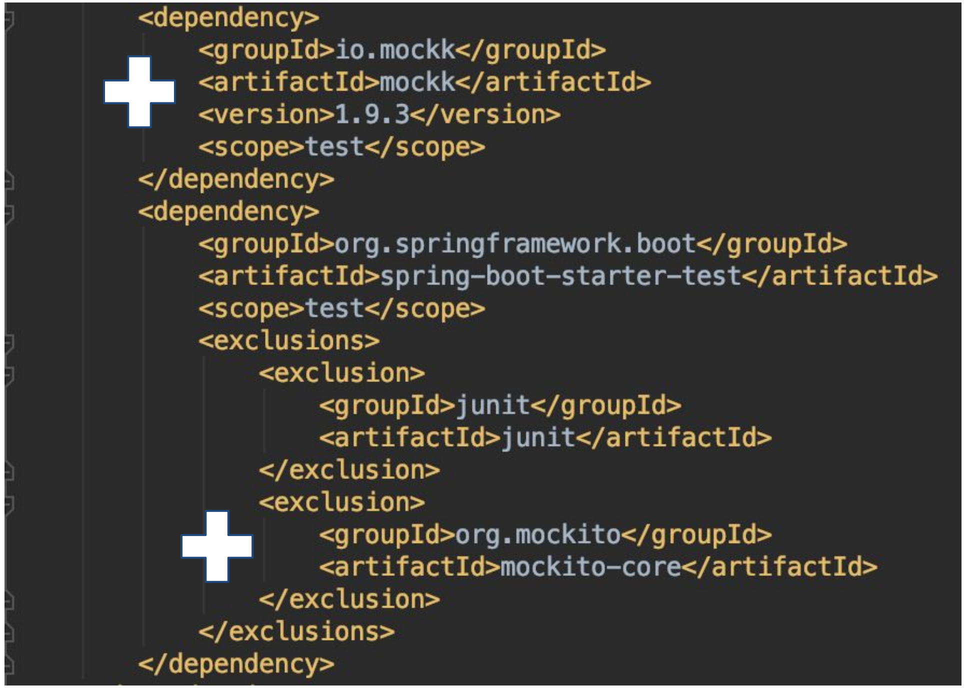


```
@SpringBootApplication
@EnableSwagger2
class Application

fun main(args: Array<String>) {
    runApplication<Application>(*args) { this: SpringApplication
        addInitializers(swaggerBeans())
    }
}
```

Mockk

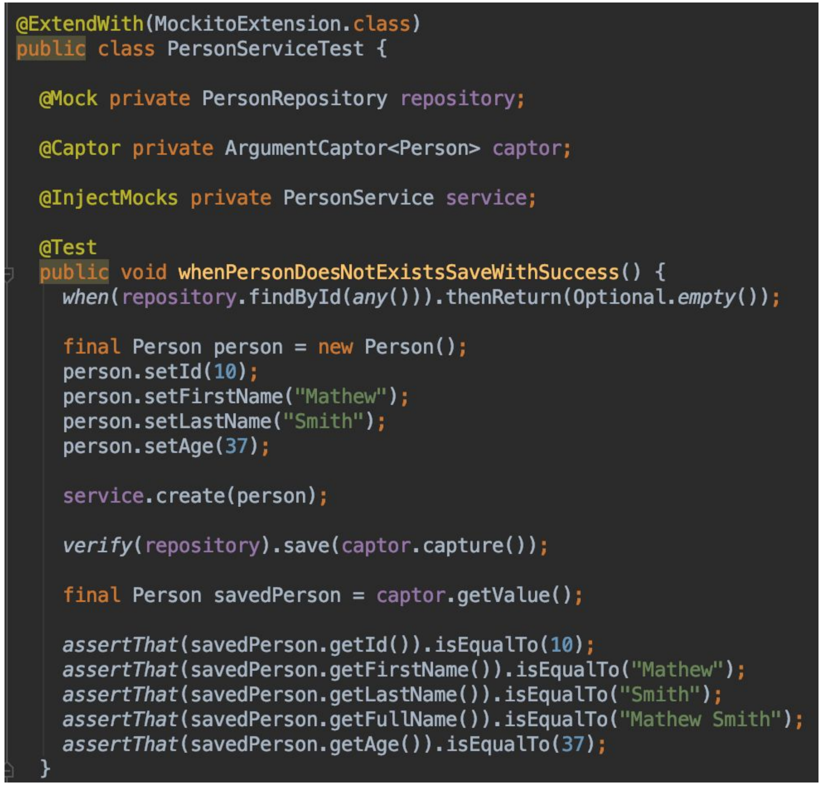


```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <scope>test</scope>
</dependency>
```

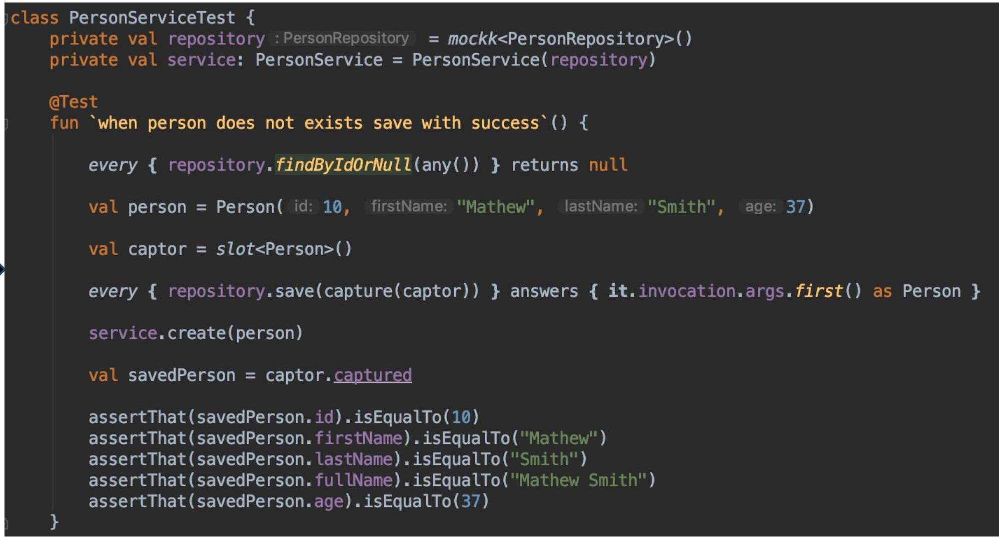


```
<dependency>
    <groupId>io.mockk</groupId>
    <artifactId>mockk</artifactId>
    <version>1.9.3</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
        <exclusion>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Unit Tests

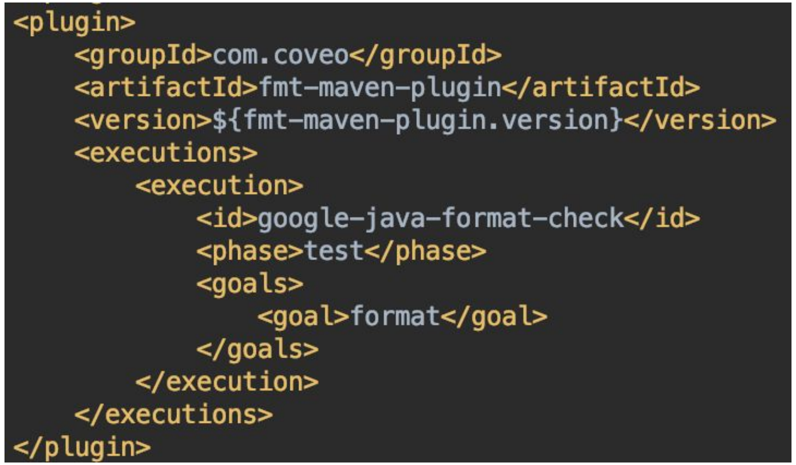


```java
@ExtendWith(MockitoExtension.class)
public class PersonServiceTest {

  @Mock private PersonRepository repository;

  @Captor private ArgumentCaptor<Person> captor;

  @InjectMocks private PersonService service;

  @Test
  public void whenPersonDoesNotExistsSaveWithSuccess() {
    when(repository.findById(any())).thenReturn(Optional.empty());

    final Person person = new Person();
    person.setId(10);
    person.setFirstName("Mathew");
    person.setLastName("Smith");
    person.setAge(37);

    service.create(person);

    verify(repository).save(captor.capture());

    final Person savedPerson = captor.getValue();

    assertThat(savedPerson.getId()).isEqualTo(10);
    assertThat(savedPerson.getFirstName()).isEqualTo("Mathew");
    assertThat(savedPerson.getLastName()).isEqualTo("Smith");
    assertThat(savedPerson.getFullName()).isEqualTo("Mathew Smith");
    assertThat(savedPerson.getAge()).isEqualTo(37);
  }
```

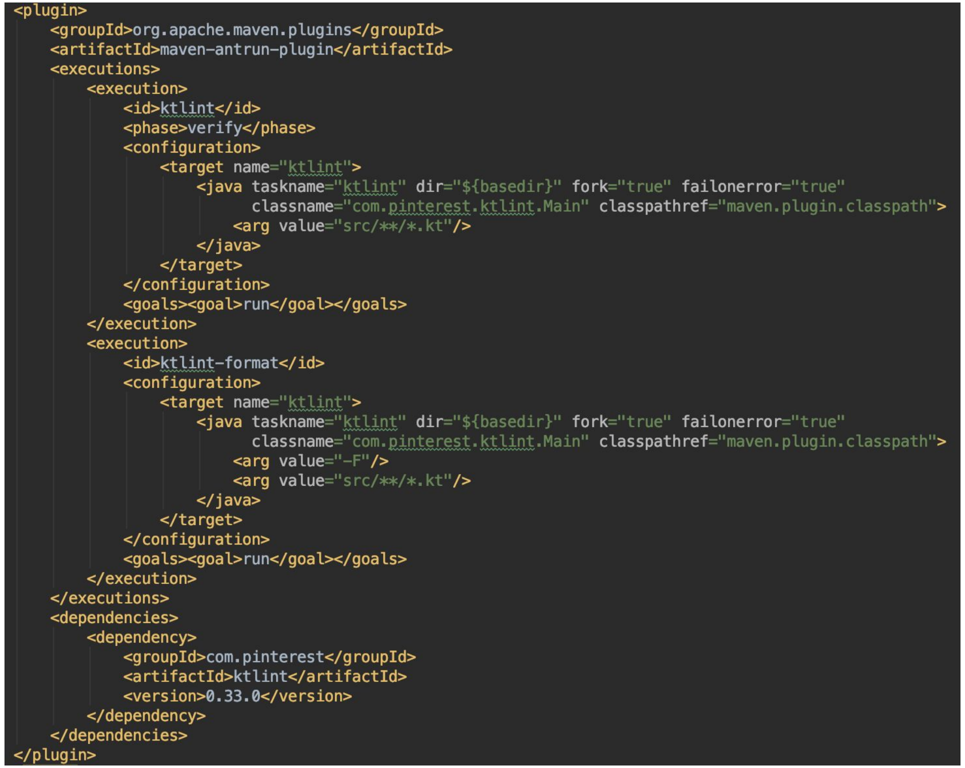


```kotlin
class PersonServiceTest {
    private val repository : PersonRepository  = mockk<PersonRepository>()
    private val service: PersonService = PersonService(repository)

    @Test
    fun `when person does not exists save with success`() {

        every { repository.findByIdOrNull(any()) } returns null

        val person = Person( id: 10,  firstName: "Mathew",  lastName: "Smith",  age: 37)

        val captor = slot<Person>()

        every { repository.save(capture(captor)) } answers { it.invocation.args.first() as Person }

        service.create(person)

        val savedPerson = captor.captured

        assertThat(savedPerson.id).isEqualTo(10)
        assertThat(savedPerson.firstName).isEqualTo("Mathew")
        assertThat(savedPerson.lastName).isEqualTo("Smith")
        assertThat(savedPerson.fullName).isEqualTo("Mathew Smith")
        assertThat(savedPerson.age).isEqualTo(37)
    }
```

Formating



```xml
<plugin>
    <groupId>com.coveo</groupId>
    <artifactId>fmt-maven-plugin</artifactId>
    <version>${fmt-maven-plugin.version}</version>
    <executions>
        <execution>
            <id>google-java-format-check</id>
            <phase>test</phase>
            <goals>
                <goal>format</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```



```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-antrun-plugin</artifactId>
    <executions>
        <execution>
            <id>ktlint</id>
            <phase>verify</phase>
            <configuration>
                <target name="ktlint">
                    <java taskname="ktlint" dir="${basedir}" fork="true" failonerror="true"
                          classname="com.pinterest.ktlint.Main" classpathref="maven.plugin.classpath">
                        <arg value="src/**/*.kt"/>
                    </java>
                </target>
            </configuration>
            <goals><goal>run</goal></goals>
        </execution>
        <execution>
            <id>ktlint-format</id>
            <configuration>
                <target name="ktlint">
                    <java taskname="ktlint" dir="${basedir}" fork="true" failonerror="true"
                          classname="com.pinterest.ktlint.Main" classpathref="maven.plugin.classpath">
                        <arg value="-F"/>
                        <arg value="src/**/*.kt"/>
                    </java>
                </target>
            </configuration>
            <goals><goal>run</goal></goals>
        </execution>
    </executions>
    <dependencies>
        <dependency>
            <groupId>com.pinterest</groupId>
            <artifactId>ktlint</artifactId>
            <version>0.33.0</version>
        </dependency>
    </dependencies>
</plugin>
```

THE DEVELOPER'S CONFERENCE

Obrigado

Perguntas?

andrefontanaignacio

@aignacio83

ignacio83

Links

THE DEVELOPER'S CONFERENCE

- https://github.com/ignacio83/spring-stack-j2k-demo
- https://kotlinlang.org/docs/reference/
- https://spring.io/guides/tutorials/spring-boot-kotlin/
- https://spring.io/blog/2017/01/04/introducing-kotlin-support-in-spring-framework-5-0
- https://spring.io/blog/2017/08/01/spring-framework-5-kotlin-apis-the-functional-way
- https://docs.spring.io/spring-framework/docs/5.1.8.RELEASE/kdoc-api/spring-framework/
- https://github.com/mockk/mockk
- https://github.com/Ninja-Squad/springmockk
- https://github.com/pinterest/ktlint

THE DEVELOPER’S CONFERENCE