Spring Boot

by Andrés Arcia

What is Spring Boot?

- "Opinionated view of the Spring Framework and third-party libraries to get started with minimum fuss"
- Most of Spring Boot applications need minimal Spring configuration

Spring Boot features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no more WAR files needed)
- Provide opinionated ‘starter’ dependencies to simplify configuration

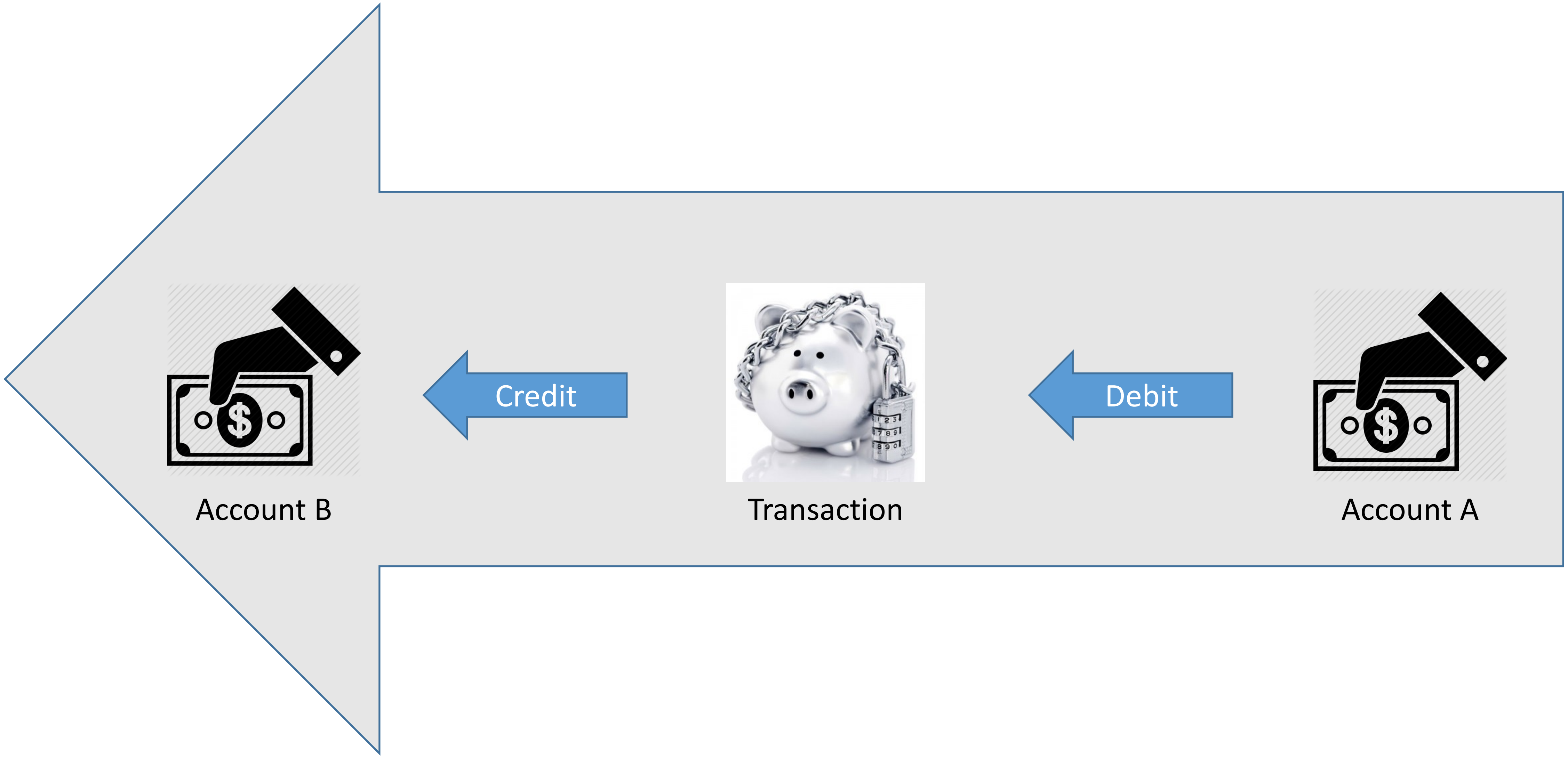
Spring Boot features

- Automatically configure Spring and 3rd party libraries whenever as possible

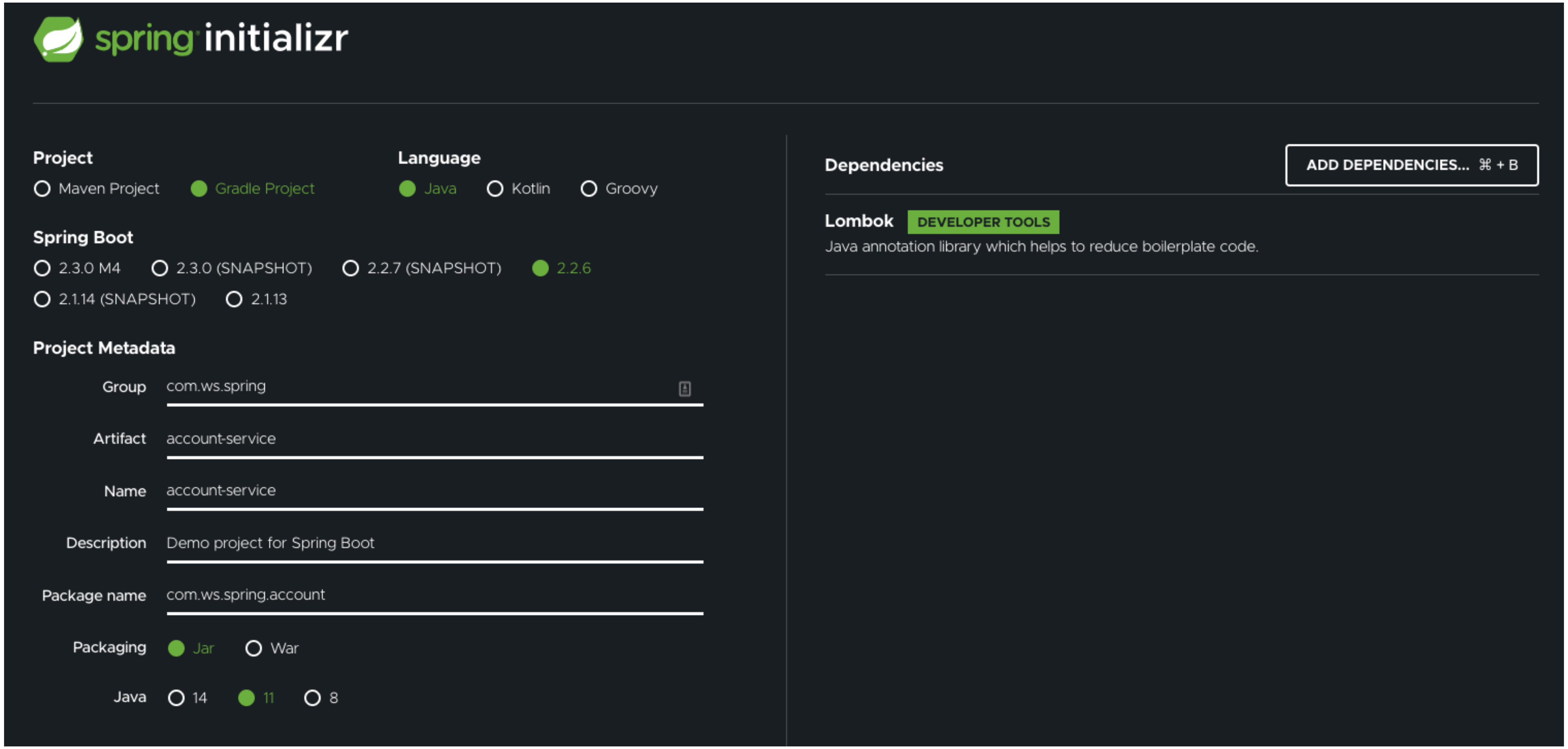
- Provide production-ready features such as metrics, health checks, and externalized configuration

- Absolutely no code generation and no requirement for XML configuration

Our project…



Our project…


spring initializr

Project
Maven Project [✓ Gradle Project]

Language
[✓ Java] Kotlin Groovy

Spring Boot
2.3.0 M4 2.3.0 (SNAPSHOT) 2.2.7 (SNAPSHOT) [✓ 2.2.6]
2.1.14 (SNAPSHOT) 2.1.13

Project Metadata

Group	com.ws.spring
Artifact	account-service
Name	account-service
Description	Demo project for Spring Boot
Package name	com.ws.spring.account
Packaging	[✓ Jar] War
Java	14 [✓ 11] 8

Dependencies

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

@Value("…")

- **@Value("…")** annotation lets Spring find properties from externalized configuration (.properties or .yml files) with the use of "${property.name.here}".

- Could also be used to set default values like **@Value("true")** or **@Value("10"),** or expression evalution (SpEL), or event system environment properties.

SpEL reference -> https://docs.spring.io/spring/docs/4.3.10.RELEASE/spring-framework-reference/html/expressions.html

@ConfigurationProperties("…") + @EnableConfigurationProperties({xxx.class})

- **@ConfigurationProperties("…")** is another way to obtain properties from an externalized configuration, it must be annotated with **@Component** to be found by the AutoScan.

- It works together with annotation **@EnableConfigurationProperties({xxx.class})**, which takes as a parameter an array of classes to scan and look for properties.

When to use what?

- If you have a single or couple of properties to inject, use **@Value("…"),** if you have multiple of complex structure, use **@ConfigurationProperties("…")** + **@EnableConfigurationProperties({xxx.class})**.

- Watch out with **@Value("…")** used as field injection.

@SpringBootApplication

- Is the Spring opinionated way to do 3 main things:
    - **@EnableAutoConfiguration**: enable Spring Boot’s auto-configuration mechanism.

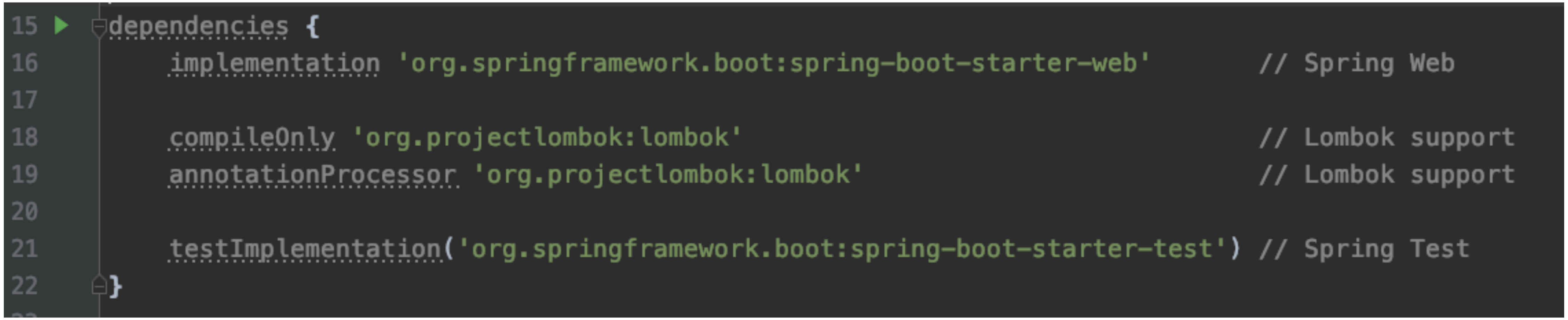
    - **@ComponentScan**: enable @Component scan on the package where the application is located (Remember the best practices)

    - **@Configuration**: allows to register extra beans in the context of import additional configuration classes.

- Is practically the same doing all together as: **@Configuration @EnableAutoConfiguration @ComponentScan**

Let’s add WEB layer to our project

- implementation 'org.springframework.boot:spring-boot-starter-web'   // Spring Web



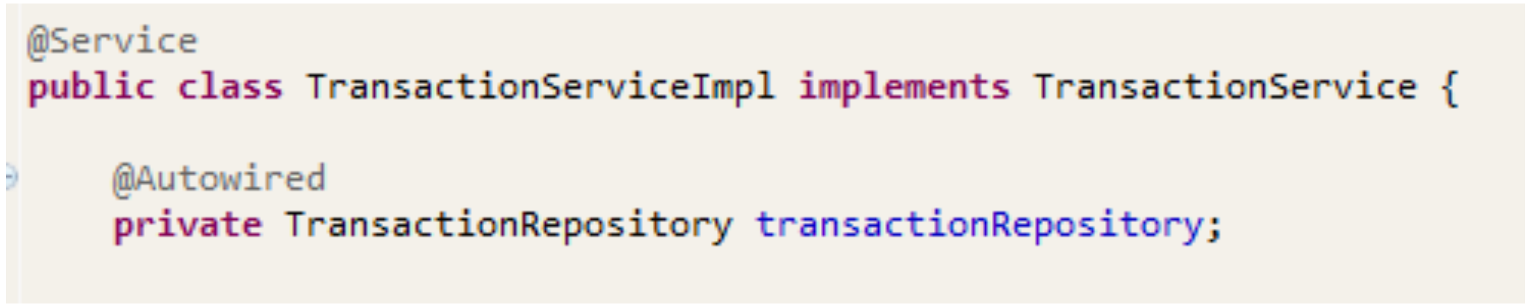
```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'        // Spring Web

    compileOnly 'org.projectlombok:lombok'                                   // Lombok support
    annotationProcessor 'org.projectlombok:lombok'                           // Lombok support

    testImplementation('org.springframework.boot:spring-boot-starter-test') // Spring Test
}
```

Spring Repo: https://github.com/spring-projects/spring-framework

Started Web: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web/2.2.6.RELEASE

Lombok: https://projectlombok.org/features/all

@Service


```
@Service
public class TransactionServiceImpl implements TransactionService {

    @Autowired
    private TransactionRepository transactionRepository;
```

- **@Service** annotation lets Spring know is a resource that is going to be used by Spring somewhere else by injection. **Normally for the service layer in MVC architecture**.

- **@Service("…")** might receive a String as parameter, that defines the ID of the resource in the Spring Application Context.

@Controller

- **@Controller** annotation lets Spring know is a resource that is going to be used by Spring somewhere else by injection. **Normally for the controller layer in MVC architecture**.

- **@Controller("…")** might receive a String as parameter, that defines the ID of the resource in the Spring Application Context.