Variations

In this lesson, we'll see how implementation of Microservices can vary.

WE'LL COVER THE FOLLOWING

- Alternatives to Spring Boot

The technical micro architecture decisions can be made differently for each microservice. But there is a connection with the macro architecture.

> The **uniformity** of the operational aspects can be enforced by the **macro architecture**.

If you want to implement a microservice with other technologies in a **Spring Boot** microservices architecture, this can lead to a lot of effort.

A macro architecture decision could be to read out configurations from an `application.properties` file.

This decision does not restrict the choice of implementation technologies. But for a Spring Boot application, the implementation is very simple because this mechanism is built into Spring Boot and the default for Spring Boot applications.

A **Go** application, on the other hand, would have to be adapted to this requirement.

This effect supports a uniform choice of technology for the microservices because implementing a microservice with Spring Boot is easier, therefore, developers would prefer Spring Boot.

A uniform choice of technology has further advantages. For example, developers are more likely to find their way around in other microservices,

and developers of different microservices can help each other out with technology issues.

In order to really treat other technologies as equal, a different macro architecture decision should be made.

Spring Boot offers many more options.

For example, the configuration can be stored in environment variables, transferred via the command line or read from a configuration server.

Alternatives to Spring Boot

In the Java area, there are some alternatives to Spring Boot.

- A classic **Java EE application** with an application server or a web server is also conceivable as an implementation for a microservice. However, in this case, deployment is more complex because the application server has to be installed additionally. Also, application servers and applications must be configured, in some cases even with two different technologies. There are many doubts about the usefulness of application servers.

- thorntail provides a simple JAR deployment. However, instead of Spring APIs it implements the standardized Java EE APIs and supplements them with technologies from the microservices area such as Hystrix.
- Dropwizard has long been offering the possibility of developing Java REST services and deploying them as JARs.

Of course, there are many other possible choices for the programming language apart from **Java** or **Go**.

It is impossible to even list them in this course.

Actually, the point this course makes is that the technologies for the implementation of each microservice are not that important.

It is easily possible to implement each microservice with a different programming language and framework, so the decision can easily be changed.

However, it is much harder to change the technologies for communication, integration, and operations that this course focuses on.

The criteria from lesson 2 of this chapter are a yardstick to check the technologies for their suitability for microservices, as lesson 8 does for Spring Boot and lesson 12 for Go. Such an assessment is recommended for each technology used.

In the *next lesson*, we'll discuss the advantages of microservices that arise from our discussion in the previous lessons, and then formally conclude this course!