





## Spring Boot for Microservices: New Microservices & Resilience

In this lesson, we'll be talking about microservice resilience and the creation of new microservices with Spring.

## We'll cover the following

- New microservices
- Resilience

## New microservices #

Creating a new microservice is very easy with Spring Boot. A **build script** and a **main class** are enough, as shown in the example simplest-spring-boot (https://github.com/ewolff/spring-boot-demos/tree/master/simplest-spring-boot).

To further simplify the creation of a new microservice, a **template** can be created. The template only needs to be adapted for a new microservice.

**Settings** for the configuration of the microservices or for logging can be defined in the template.

Thus, a template simplifies the creation of new microservices and facilitates compliance with macro architecture rules.

A particularly easy way to create a new Spring Boot project is to use

http://start.spring.io/ (http://start.spring.io/).



The developer must select the build tool, the programming language, and a Spring Boot version.

In addition, they can select different starters. Based on this, the website then creates a project that can be the basis for the implementation of a microservice.

## Resilience #

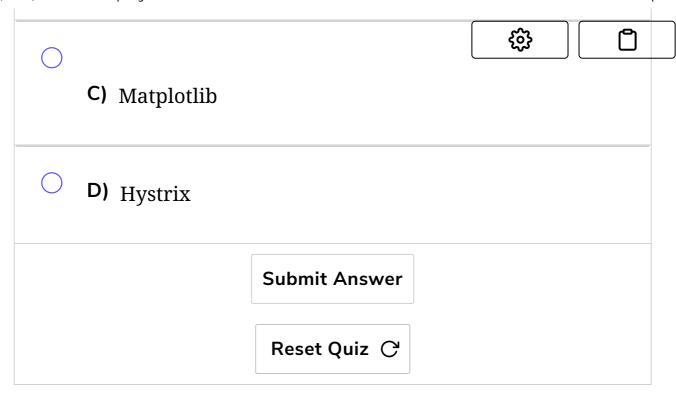
For resilience, a library like **Hystrix** can be useful.

Hystrix implements typical **resilience patterns** such as **timeouts in Java**. Spring Cloud offers an integration and further simplification for Hystrix.

 $\mathbf{Q} \mathbf{U} \mathbf{I}$ 

Z

- Name a library that can be used for resilience in Java.
- A) REST
- O B) JAX



In the next lesson, we'll begin our discussion on Go.

