

## Microservices met Spring Boot

### Doelgroep Cursus Microservices met Spring Boot

De cursus Microservices met Spring Boot is bestemd voor ervaren Spring Java Developers die Spring Boot willen gebruiken voor de ontwikkeling van Microservices.

### Voorkennis Cursus Microservices met Spring Boot

Ervaring met programmeren in Java en Spring is vereist om deel te kunnen nemen aan deze cursus. Basis kennis van een Microservice Architectuur is bevorderlijk voor een goede begripsvorming.

### Uitvoering Training Microservices met Spring Boot

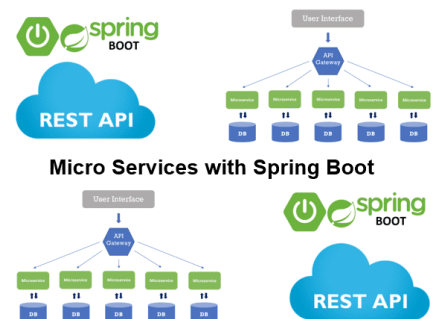
De concepten worden behandeld aan de hand van presentaties en demo's. De theorie wordt afgewisseld met oefeningen. De cursustijden zijn van 9.30 tot 16.30.

### Certificering Microservices met Spring Boot

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Microservices met Spring Boot.

Duur: 3 dagen

Prijs: € 1999

[Open Rooster](#)


## Inhoud Microservices met Spring Boot

In de cursus Microservices met Spring Boot leren de deelnemers hoe je met Spring Boot snel en efficient microservices kunt ontwikkelen in de vorm van fat jars met een embedded server. Deze Spring Boot microservices kunnen onafhankelijk van elkaar worden gedeployed en als processen worden gestart.

### Intro Microservices

De cursus gaat van start met een overzicht van het hoe en waarom van microservices. Microservices zijn ontwikkeld in reactie op problemen met monolithische applicaties die op den duur moeilijk onderhoudbaar en uitbreidbaar zijn gebleken. Bij een Microservices Architectuur wordt de totale functionaliteit gerealiseerd door samenwerkende microservices die ieder onder verantwoordelijkheid van een team vallen.

### Spring Boot

Het Spring Boot Framework is bij uitstek geschikt voor de ontwikkeling van microservices omdat bij Spring Boot applicaties alle dependencies in een jar worden opgenomen. Ook kunnen Spring Boot applicaties eenvoudig worden voorzien van een embedded server zodat de microservices via HTTP kunnen communiceren.

### Inter Process Communicatie

In de cursus Microservices met Spring Boot komen diverse inter process communicatie mechanismen tussen de microservices aan de orde zoals synchrone communicatie via een REST API en asynchrone communicatie via messaging. En ook communicatie via een binary protocol staat op het programma.

### Discovery Patterns

Voorts is er aandacht voor de manieren waarop microservices elkaar vinden. Zowel client side discovery als server side discovery wordt besproken. En eveneens wordt ingegaan op de inzet van tooling en utilities zoals Netflix Eureka en Apache Zookeeper.

### Data Management

Ook het omgaan met Data in Spring Boot Microservices is onderdeel van het programma van de cursus. Microservices hebben veelal een eigen database en aan de orde komt hoe in dat geval transacties kunnen worden afgehandeld die verschillende microservices betreffen. Zowel een event driven architectuur als local en compensating transacties worden besproken.

### Deployment

Voor de deployment van Microservices zijn diverse opties beschikbaar waaronder virtual machines en containers. Met name wordt de deployment van de Spring Boot Microservices in docker containers besproken. En ook container orkestratie met Kubernetes staat op het programma. Tenslotte wordt ingegaan op de strategieën om een monolithische architectuur om te vormen naar een microservices architectuur.

## Modules Microservices met Spring Boot

| Module 1 : Microservices Intro                                                                                                                                                                                                                                    | Module 2 : Using an API Gateway                                                                                                                                                                                                                                         | Module 3 : Spring Boot                                                                                                                                                                                                                                                    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| What are MicroServices?<br>Components and Services<br>Loose coupling<br>Passing Messages<br>Design Characteristics<br>Simplicity and Transparency<br>Reproduceability<br>Asynchronous calls<br>Mocking Components<br>Testing Components<br>Debugging Components   | REST Web Services<br>GET, POST, PUT, DELETE<br>@RestController<br>Default Content Types<br>@ResponseStatus and HttpStatus<br>Working with XML and JSON<br>Multiple Representations<br>Filtering with @JsonView<br>REST Clients<br>RestTemplate<br>Sending HTTP Requests | Convention over Configuration<br>No XML<br>Spring Boot CLI<br>Building and Deploying<br>Using Templates<br>Gathering Metrics<br>Using Java With start.spring.io<br>Spring Boot Starters<br>Building as a Runnable JAR<br>Data Access with Spring Data<br>Property Support |
| Module 4 : Interprocess Communication                                                                                                                                                                                                                             | Module 5 : Discovery Patterns                                                                                                                                                                                                                                           | Module 6 : Data Management                                                                                                                                                                                                                                                |
| Interaction Styles<br>Request/response<br>Notification<br>Publish/Subscribe<br>Synchronous vs Asynchronous<br>Messaging<br>Rest<br>Synchronous IPC<br>Apache Thrift<br>Message Formats                                                                            | Client Side Discovery<br>Load Balancing<br>Service Registry<br>Netflix Eureka Example<br>Client Side Drawbacks<br>Server Side Discovery<br>Request Routing<br>Kubernetes<br>Apache Zookeeper<br>Self Registration Pattern                                               | Distributed Data Problems<br>ACID Transactions<br>Distributed Transactions<br>Polyglot Persistence<br>Event Driven Architecture<br>Eventual Consistency<br>Achieving Atomicity<br>Local Transactions<br>Compensating Transactions<br>Mining Transaction logs              |
| Module 7 : Deployment Strategy                                                                                                                                                                                                                                    | Module 8 : Refactoring to Microservices                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                           |
| Multiple Services Pattern<br>Process or Process Group<br>Multiple Service Instances per Host<br>No isolation drawback<br>Service Instance Per Host<br>Service Instance per VM<br>Service Instance per Container<br>Docker and Kubernetes<br>Serverless Deployment | Monolithic Applications<br>Application Modernization<br>Big Bang Rewrite<br>Glue code<br>Split Frontend and Backend<br>Extract Services<br>Prioritizing<br>Extract Modules<br>Shrinking the Monolith                                                                    |                                                                                                                                                                                                                                                                           |