FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Labo frameworks voor serverapplicaties
Spring Reactive

6 november 2024

1 Inleiding

In de vorige labo's hebben we een volledig blog platform ontwikkeld met Java en Spring Boot bestaande uit volgende onderdelen:

- In-memory database met H2
- Datalaag met JDBC en JPA
- REST API (JSON & XML) met Spring MVC
- Statische webpagina (HTML, JS & CSS) met Spring MVC
- Authenticatie en authorisatie met Spring Security

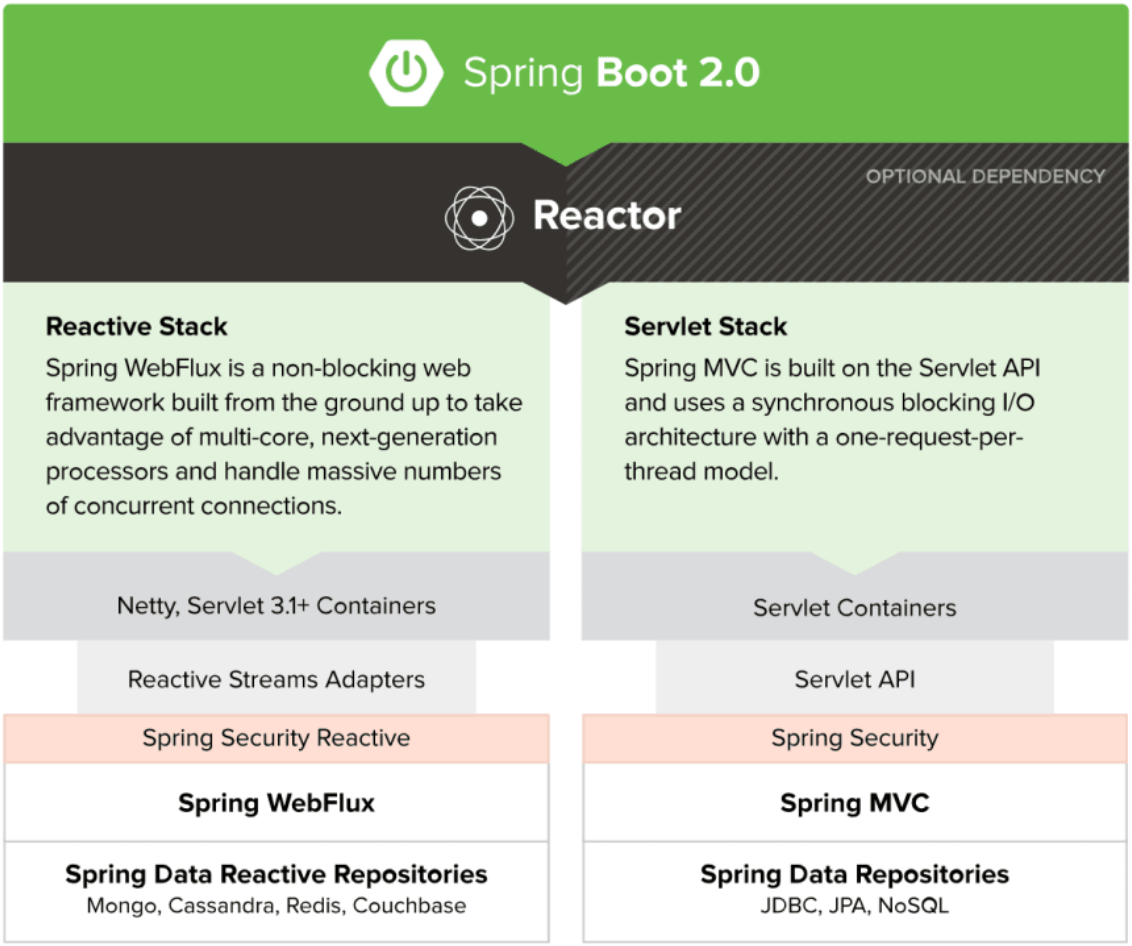
In dit labo maken we opnieuw hetzelfde blog platform maar nu met behulp van de reactive stack in Spring Boot in plaats van de servlet stack. Als ontwikkelaar zal je veel gelijkenissen ervaren tussen beide stacks maar onderliggend zijn ze volledig onafhankelijk.

2 Spring Reactive Programming

Reactive programming is een paradigma waarmee ontwikkelaars niet-blokkerende, asynchrone applicaties kunnen bouwen die back-pressure aankunnen (flow control). Reactieve systemen maken beter gebruik van moderne processors. Ook zorgt de opname van back-pressure in reactieve programmering voor een betere veerkracht tussen ontkoppelde componenten.

De Spring-portfolio biedt twee parallelle stacks, zie figuur 1. De ene is gebaseerd op een Servlet-API met Spring MVC- en Spring Data-constructies. De andere is een volledig Reactive stack die gebruikmaakt van Spring WebFlux en de reactieve repositories van Spring Data. Spring Security heeft native ondersteuning voor beide stacks.

In Spring Reactive zijn 'Mono' en 'Flux' twee fundamentele concepten voor het omgaan met asynchrone datastromen. 'Mono' vertegenwoordigt een asynchrone datastroom die 0 of 1 element kan uitzenden, perfect voor enkelvoudige resultaten. Aan de andere kant staat 'Flux', ontworpen voor het verwerken van 0 tot vele elementen, ideaal voor het werken met reeksen van data. https://www.baeldung.com/reactor-core



Figuur 1: Overzicht van de Reactive en Servlet Stack in Spring.

3 Opzetten NoSQL Database

Om de capaciteiten van Spring Reactive goed te benutten maken we gebruik van MongoDB als NoSQL database. MongoDB is document gebaseerd en een van de meest gebruikte NoSQL databases. Kies hieronder om MongoDB te gebruiken via Docker of te installeren op je computer.

- Starten MongoDB
  - MongoDB via Docker

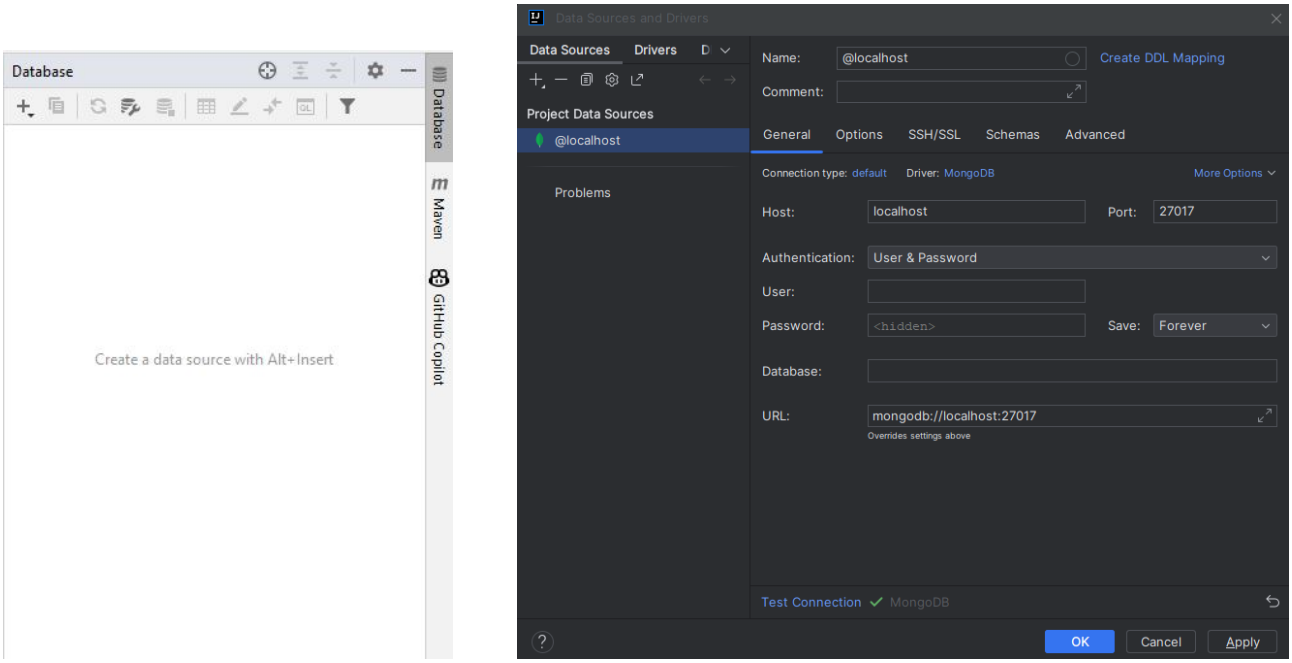
    ```
    docker run --rm -d --name my-mongo -p 27017:27017 \
    mongo mongod --replSet myReplicaSet

    docker exec my-mongo mongosh --eval "rs.initiate();"
    ```

  - MongoDB installeren op Windows

    Volg de instructies op volgende link om mongodb te installeren op je windows computer: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

- Test nu of je kan verbinden met de databank door mongoDB toe te voegen in het Database Tool Window van Intellij. Klik op '+' om mongoDB toe te voegen als datasource. Pas vervolgens de credentials aan indien nodig en verbind, zie figuur 2. Ga pas verder als dit lukt.



Figuur 2: Opzetten van een connectie met MongoDB in Intellij.

4 Reactive Blog Platform

4.1 Nieuw Project

In plaats van het bestaande blog project in de servlet stack om te migreren naar de reactive stack, starten we vanaf een nieuw project. Op deze manier kunnen we stapgewijs opbouwen en voorkomen we fouten door dependencies uit beide stacks.

- Maak een nieuw Spring Boot project aan met Spring Initializr en volgende dependencies:
  - Spring Boot DevTools
  - Spring Reactive Web (tegenhanger Spring Web)
  - Spring Data Reactive MongoDB (tegenhanger Spring Data JPA voor MongoDB)
  - Spring Boot Actuator
- Start de applicatie en bekijk de logs aandachtig. Wat vind je wel en niet terug?

  ➪ Indien er nog geen connectie met MongoDB wordt gelogd, zal je zelf nog de juiste properties moeten toevoegen, zie hier voor de configuratie.

4.2 Datalaag

- Nu voegen we het model toe, kopieer de BlogPost klasse naar het reactive project en pas aan waar nodig. TIP: MongoDB is een *document* store en slaat ID's op als ObjectId.
- De volgende stap is het toevoegen van een repository die de communicatie met de database uitvoerd. Aangezien de JpaRepository deel uit maakt van de servlet stack kunnen we deze niet meer gebruiken. Er zijn meerdere alternatieven in de reactive stack. HINT: https://www.baeldung.com/spring-data-mongodb-reactive

- Als de vorige stappen zijn uitgevoerd, kunnen we nu blogposts opslaan, zoeken, verwijderen en aanpassen in MongoDB via de repository.
  - We voegen een property toe aan de application.properties file zodat alle uitgevoerde MongoDB queries gelogd worden in de console.

```
logging.level.org.springframework.data.mongodb.core.ReactiveMongoTemplate=DEBUG
```

  - Om te testen of alles correct werkt, voeg je onderstaande bean toe aan de klasse met de '@SpringBootApplication' annotatie. De bean wordt uitgevoerd na het opstarten van de applicatie en plaatst dummy data in de database. Welke queries vind je terug in de logs?

```
@Bean
CommandLineRunner test(BlogPostRepository repository) {
  return args -> {
    repository.deleteAll()
      .thenMany(
        Flux.just("Reactive Spring Boot", "Reactive Spring Data",
                  "Reactive MongoDB")
          .map(title -> new BlogPost(null, title, "Some content ..."))
            .flatMap(repository::save)
      )
      .thenMany(repository.findAll())
      .subscribe(System.out::println);
  };
}
```

4.3 REST API

Via een REST API kunnen eenvoudige CRUD-operaties (Create, Read, Update en Delete) uitgevoerd worden op een resource, in ons geval de blogposts. Doordat we nu gebruik maken van de reactive stack met non-blocking I/O returnen de methodes van de controller geen objecten meer maar asynchrone resultaten als 'Mono' of 'Flux'.

- Voorzie in een controller dezelfde api endpoints als het originele blog platform (zonder beveiliging) maar nu met 'Mono' of 'Flux' als return type. Gebruik ook opnieuw een service met bijhorende interface als abstractie tussen de repository en de rest-controller.
  - Alle blogposts ophalen.
  - Eén specifieke blogpost ophalen a.d.h.v een id. Indien een onbestaande blogpost wordt opgehaald, wordt er een `BlogPostNotFoundException` gegooid
  - Een blogpost toevoegen. De HTTP-response bevat de locatie header en HTTP-status 201: created.
  - Een blogpost verwijderen, resulteert in een 204 HTTP-status.
  - Een bericht aanpassen, resulteert in een 204 HTTP-status. Indien het id in het path niet overeenkomt met het id in de body wordt een HTTP-status 409 - conflict terug gestuurd.
  - Zoeken naar blogposts op basis van een zoekterm.
- Test elk endpoint manueel uit met postman.

4.4 Streaming REST API

Functioneel werkt de REST API analoog aan de originele versie in de servlet stack. Reactive programming komt pas goed tot zijn recht in een streaming scenario of wanneer threads vaak wachten op een blocking I/O. Dit gaan we nu simuleren door blogposts te streamen naar de browser.

- Voeg een nieuw endpoint ("/stream/posts") toe die alle blogposts terug geeft maar met 1 seconde vertraging tussen elke blogpost. TIP: delayElements
- Wanneer we het nieuwe endpoint testen met postman merken we op dat er niet veel veranderd is behalve de langere antwoord tijd. Dit komt doordat de webserver standaard een json bericht wil terug sturen, dus pas een antwoord stuurt eens alle blogpost beschikbaar zijn. Test nu opnieuw met onderstaande waardes in de "Accept" header. Wat merk je op?
  - application/json
  - text/event-stream
  - application/x-ndjson
- Herhaal de vorige stap ook met de commandline tool *curl* om de streaming endpoint te testen. Probeer zeker ook de verschillende waardes in de "Accept" header.
- Het is ook mogelijk om via de browser een streaming endpoint te consumeren als een event stream. Er zijn twee mogelijkheden:
  - We dupliceren het endpoint dat de blogpost als een stream teruggeeft maar ondersteunen nu enkel 'text/event-stream'. TIP: zet het produces attribuut van GetMapping gelijk aan het juiste MediaType, meer info.
  - Een tweede manier is om niet op de server het media type te bepalen, maar om de client een request te laten sturen met een specifieke header. Test dit opnieuw uit via Postman.
- Voeg een laatste endpoint toe dat een gebruiker toelaat om te subscriben op een event stream die een nieuw event verstuurd voor elke aanpassing aan de tabel met blogposts in de database. TIP: change streams Spring en MongoDB

4.5 Web Applicatie

- Plaats de startcode (JS & HTML) onder **/src/main/resources/static**. Zoals je reeds weet is alles in deze map statisch beschikbaar via de browser.
- Browse naar http://localhost:8080 en test de functionaliteit uit in de browser. Waar wordt de change stream gebruikt?

4.6 Snel klaar?

- Voor zie extra integration testen voor de nieuwe streaming endpoints.
- Voeg de nodige beveiliging toe met Spring Security zodat enkel een admin account blog posts kan toevoegen of aanpassen.