Microservices

Blog Post App verarbeiten

Beschreibung

Die Aufgaben in diesem Dokument beinhalten die Inbetriebnahme der App, die im Blog-Post vorgestellt wird.

Inhaltsverzeichnis

1 Einstieg in Microservices

1.1 Der Blog-Post

Der folgende Artikel beschreibt den Aufbau einer kleinen Microservices Architektur und nutzt einige der meistverbreiteten Komponenten. Der Autor baut einen Car-Service mit einem Car-Client der weniger interessante Fahrzeuge aus der Gesamtliste des Car-Services herausfiltert.

Die Anwendung steht hinter einem Gateway als «Einheit» zur Verfügung.
https://developer.okta.com/blog/2019/05/22/java-microservices-spring-boot-spring-cloud

2 Aufgaben

2.1 Aufgabe 1 – Projekt herunterladen

App Clonen
Im Artikel findet sich auch ein Link, über welchen Sie das Projekt klonen können.

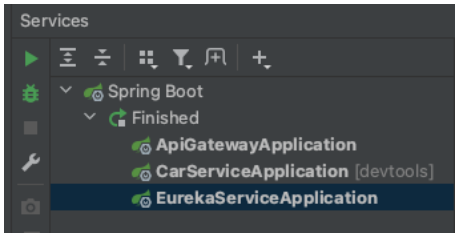
https://github.com/oktadev/java-microservices-examples

Übernehmen Sie den Ordner «spring-boot+cloud» und löschen Sie die restlichen Ordner.

2.2 Aufgabe 2 – Projekt in der Entwicklungsumgebung laden

Projekt mit den Unterprojekten in Intellij öffnen.
Wichtig dabei ist es, den übergeordneten Ordner zu öffnen und die Unterprojekte als «Services» zu laden. Zudem müssen die Unterprojekte als Maven Projekte geladen werden

Wählen Sie im Footer-Menü der Entwicklungsumgebung den Menüpunkt «Services» und kontrollieren Sie, ob die drei Services richtig geladen wurden.



2.3 Aufgabe 3 – Authentisierung entfernen

Einige Services beinhalten eine Authentisierung mit Spring Security und einem externen Dienst «Okta».

Entfernen Sie alle Methoden und Einstellungen, welche die Authentisierung betreffen. Dies betrifft insbesondere das «API-Gateway», sowie der «Car-Service».

Achten Sie auch Einträge in den `application.properties` Dateien.

2.4 Aufgabe 4 – Microservices starten

Nun können wir die Anwendung hochfahren, indem wir die einzelnen Microservices starten.

Starten Sie die Microservices in der folgenden Reihenfolge:

- DiscoveryService
- CarService
- API-Gateway

Beheben Sie allfällige Fehler. Suchen Sie nach den Ursachen.

Funktionalität überprüfen:

- Eureka Server und Registrierungen überprüfen:
  http://localhost:8761
  Das Eureka-Dashboard mit den registrierten Services sollten angezeigt werden.
- Car-Service, Rest-Services überprüfen:
  http://localhost:8090/home und http://localhost:8090/cars
- API-Gateway
  http://localhost:8080/cars, etc.

3 Untersuchung und Wissenserlangung

3.1 Aufgabe 5 – Annotationen untersuchen

Die Anwendung mit ihren Services beinhalte viele Annotationen die grosse Auswirkung auf die Anwendung haben. Jeder Dienst und jedes Tool löst mit seinen Annotationen irgendwelche Aufgaben. Dies kann auf den ersten Blick sehr verwirrend sein, da die damit verbundenen Implementationen nicht sichtbar sind.

Dies bedingt ein implizites Wissen über die Annotationen und deren Auswirkungen.

Erstellen Sie eine Tabelle mit den genutzten Annotationen. Führen Sie darin den Namen der Annotation, die Library oder Tool, zu welchem die Annotation gehören und schreiben Sie in einer dritten Spalte die Funktionalität, welche die Annotation mit sich bringt.

Gruppieren Sie die Annotationen nach Library/Tool.

Beispieltabelle:

Annotation	Library/Tool	
@SpringBootApplication	Spring Boot	Markiert eine Klasse als Spring Boot App Startklasse.
@EnableEurekaServer	Netflix Eureka (Spring Cloud)	Markiert eine Klasse/App als Eureka Server.
…		

3.2 Aufgabe 6 – Anwendung verändern

Die einzelnen Fahrzeuge sollen mit einer Leistung erweitert werden.

a) Erweitern Sie die Klasse «Car» im Microservice «Car-Service» mit einem Attribut «power» und dem Datentyp int.
b) Erweitern Sie den «ApplicationRunner», sodass zu jedem Fahrzeug eine Leistung angegeben werden kann. Dabei spielt es keine Rolle wie Sie das erreichen.
(Im ApplicationRunner haben Sie Zugriff auf das Repository und dessen Methoden).