TH BINGEN
University of Applied Sciences

Dokumentation

OpenServiceBroker - Redis

Inhalt

1 Einleitung

Bei dem Projekt handelt es sich um eine Implementierung der OpenServiceBroker Spezifikation, die unter https://github.com/openservicebrokerapi/servicebroker/blob/v2.15/spec.md einsehbar ist. Die Applikation ist eine auf Spring Boot basierende Java-App, die mittels Helm Redis-Instanzen auf Kubernetes deployen kann. Dabei sind drei verschiedene Größen („small“, „standard“ und „cluster“) auswählbar.

Zusätzlich zu diesem Dokument wurde mit Hilfe des Javadoc Plugins für Maven eine Javadoc Dokumentation generiert.

2 Abhängigkeiten

2.1 Dependencies

Im Projekt werden die folgenden Dependencies genutzt:

Dependency	**Version**
Spring Boot Spring Boot Starter Parent Spring Boot Starter Web Spring Boot Starter Security Spring Boot Starter Test	2.2.4.RELEASE
Microbean Helm	2.8.2.1.1.1
JSON Schema	1.9.2
Javadoc	3.1.1
Jackson Core	2.10.2

Der genutzte Maven Compiler wird per Anweisung in der ‚pom.xml‘ des Projekts auf Version 1.8 gesetzt.

2.2 Lokale Abhängigkeiten

Um die Applikation lokal starten und testen zu können, wird eine Installation von Minikube und Helm (Version 2.14.0) benötigt. Eine Nutzung von Helm ab Version 3 ist nicht möglich, da ab dieser kein Tiller mehr verwendet wird. Tiller stellt eine serverseitige Komponente bereit, um Helm Charts in Kubernetes zu deployen. Die Microbean API baut komplett auf Tiller auf, wodurch ein Fehlen von Tiller die Nutzung dieser nicht mehr mölgich macht.

3 Aufbau

3.1 Projektstruktur

Das Projekt ist als Maven-Build in einer modularen Struktur aufgebaut. Es enthält die Module Redis, Core, Model, Security und Helm, welche im Folgenden näher beschrieben werden sollen. Jedes Modul ist zudem als eigenes Subpackage definiert.

Der zentrale Einstiegspunkt der Applikation befindet sich in der Datei Application.java (Modul: Redis) welche gleichzeitig als ApplicationContext für Spring fungiert.

3.2 Core

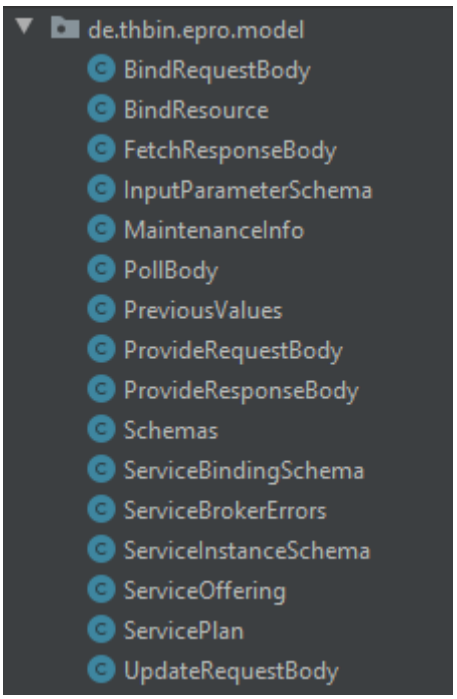
Im ‚Core‘-Modul ist die Implementierung der OpenServiceBroker Spezifikation enthalten. Die Klasse ‚ServiceBrokerImpl‘ stellt hier die geforderten Rest-Endpoints zur Verfügung und ist zuständig, die

Request-Anfragen zu bearbeiten und entsprechende Responses zurückzugeben. Sobald eine Anfrage zum Erzeugen einer Service Instanz, eines Bindings, eine Anfrage zum Update einer Service Instanz oder eine Anfrage zum Entfernen einer Service Instanz oder Bindings stattfindet, wird die entsprechende Methode der Klasse HelmDeployer aufgerufen. Des weiteren werden im Broker die instance_ids und binding_ids verwaltet.

Der Broker läuft auf der Version 2.14 und lehnt Anfragen mit einer falschen Versionsnummer ab.

3.3 Model

Im Modul ‚Model' befinden sich alle Klassen, um die von der OpenServiceBroker Definition geforderten Datenmodelle abzubilden. Um Servicepläne und alle nötigen Informationen zu laden befindet sich hier auch das ServiceSchema.json. Dieses wird beim Erstellen des ServiceCatalog gelesen und somit ein ServiceOffering und 3 ServicePläne erstellt.
In jedem ServicePlan befindet sich ein Schema, welches einen ServiceInstanceSchema mit 2 InputParameterSchemas zum Erstellen einer Service Instanz und zum Updaten einer Instanz sowie ein ServiceBindingSchema mit einem InputParameterSchema um die Service Instanz an die Applikation zu binden.



3.4 Helm

Die Klasse HelmDeployer stellt eine Schnittstelle zwischen der Java-Anwendung und dem Kubernetes-Package-Manager Helm dar. Über die Methode „DeployRedis" kann via Helm eine Redis-Instanz auf ein Kubernetes-Cluster deployed werden. Hierbei stehen die Größen „Small", „Standard" und „Cluster" zur Verfügung. Außerdem muss eine ID als String übergeben werden, um die ausgelieferte Service-Instanz identifizieren zu können. Die ID wird auch zum Deinstallieren eines Release benötigt.

Hierbei bedient sich die Klasse der „Microbean Helm"-API.

Die Enumeration DeploymentSize stellt die drei verfügbaren Deploymentgrößen zur Auswahl.

In der Klasse DeploymentData wird ein Release zusammen mit seinem DeploymentName gespeichert.

Folgende Versionen wurden für die Bestandteile verwendet:

microBean Helm: 2.8.2.1.1.1

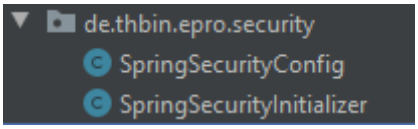
Helm: 2.14.0

Kubernetes: 1.15.5

Minikube: 1.5.1

3.5 Security

Im Modul ‚Security' wird in der Klasse ‚SpringSecurityConfig' die Standardkonfiguration, die mittels der Abhängigkeit ‚Spring Boot Starter Security' im Projekt aktiviert wurde, überschrieben. Es wird ein globaler Administrator angelegt, der ein zufällig generiertes Passwort erhält. Dieses wird bei Start des Programms auf der Konsole ausgegeben und kann zum Erreichen aller Rest-Endpoints in Kombination mit dem Nutzernamen ‚admin' verwendet werden.



4 Tests

Getestet wurde die Applikation mit Hilfe des Frameworks ‚osb-checker-kotlin' der Firma ‚evoila', welches unter https://github.com/evoila/osb-checker-kotlin zu finden ist. Um dieses zu nutzen, muss die application.yml bei jedem Start des ServiceBroker Application mit dem zufällig generierten Passwort angepasst werden, da das Framework sonst keinen Zugriff auf die Rest-Endpoints erhält.

Neben den Framework Tests existieren einige Integrationstests, um die Basisfunktionalitäten der Rest-Endpoints sicherzustellen.