

Resilience4j bei Microservices – Gegenüberstellung Hystrix anhand einer Beispielanwendung

Studienarbeit

im Studiengang
Softwaretechnik und Medieninformatik

vorgelegt von

Heiko Fischer

Matr.-Nr.: 751209

Patrick Auer

Matr.-Nr.: 755350

am 14. November 2019
an der Hochschule Esslingen

Prüfer:

Matthias Häussler

Kurzfassung

Inhalt der hier vorgestellten Studienarbeit ist es anhand einer Beispielanwendung den Unterschied zwischen Resilience4j und Hystrix aufzuzeigen. Es handelt sich hierbei um sogenannte Circuit Breaker. Zu diesem Zweck wurden zwei unterschiedliche Backendanwendungen implementiert, eine für den Umgang mit Hystrix, die andere im Umgang mit Resilience4j.

Es wurde ein verteiltes System implementiert. Die Backendanwendung wurde in Java und Maven umgesetzt. Eine Frontendanwendung wurde mit React JS implementiert, um die Funktionalität der beiden Backendanwendungen zu prüfen. Mittels einer API (REST-Schnittstelle) werden die Daten übermittelt. Grundsätzlich wird hierbei der Nutzen von Resilience4j und Hystrix aufgezeigt, welche dabei helfen ein System performanter zu gestalten.

Mit Hilfe von Circuit Breaker wird eine Lösung angeboten, die Systemen bei der Behandlung von Fehlern unterstützt. Ein weiterer Vorteil von Circuit Breaker ist, dass das System bei der Wiederherstellung von Daten unterstützt wird.

Inhaltsverzeichnis

Kurzfassung	2
Inhaltsverzeichnis.....	3
1 Wahl der Programmiersprache	4
2 Circuit Breaker	5
2.1.1 Die Problematik:	5
2.1.2 Lösungsansätze durch Circuit Breaker:	5
2.1.3 Funktionsweise von Circuit Breaker:	6
2.1.4 Die REST-Schnittstellen.....	6
3 Der React Client	7
3.1 Vorstellung des Frameworks	7
3.2 Erstellen eines React Projekts	8
3.3 Aufbau der Frontendanzwendung.....	9
Ehrenwörtliche Erklärung	10
Ehrenwörtliche Erklärung	11

1 Wahl der Programmiersprache

Für die Umsetzung der Studienarbeit wurde die Programmiersprache Java für die Backendentwicklungen von Hystrix und Resilience4J gewählt, sowie für das Frontend die Sprachen JavaScript, HTML und CSS mit der Bibliothek React JS.

Natürlich wären hier auch andere Programmiersprachen oder Frameworks möglich gewesen. Aufgrund der Erfahrungen mit Java im Studium und die damit gelernten Vor- und Nachteile dieser Sprache, war dies die begründete Wahl zur Programmiersprache, sowie React JS aufgrund von Umsetzungen anderer Studienprojekten. Des Weiteren wurde Java gewählt, aufgrund der Verbindung von einzelnen Systemen über Schnittstellen. Weiter spielte auch die von Java angebotene Flexibilität eine große Rolle. Da wir mit Windows sowie Apple Geräten arbeiten wurde Wert auf Plattform unabhängiges Programmieren gelegt. Um eine gute Grundlage für das Testen der Circuit Breaker zu ermöglichen, empfiehlt sich Java, da hier Multithreading möglich ist.

Java liefert dabei eine etwas schlechtere Performance, als das beispielsweise bei C# der Fall ist, da Java mit einem Interpreter arbeiten muss. Dies spielt in unserem Fall allerdings keine große Rolle, da die Beispielanwendung sehr klein ist und nur die nötigsten Funktionalitäten zur Darstellung genutzt werden.

2 Circuit Breaker

2.1.1 Die Problematik:

Innerhalb eines Systems, beziehungsweise einer Software kann es immer zu Fehlern kommen.

Neben den „normalen“ Fehlern, die mit Hilfe von Fehlerbehandlungen (Exception Handling), gibt es noch zahlreiche andere Fehler, die auftreten können.

Hierzu zählen beispielsweise Fehler durch:

- Schlechte Netzwerkverbindung -> Connection Timeout
- Verlust der Netzwerkverbindung -> Connection Timeout
- Übertragungsfehler
- Falsche Datenstände durch Commits
- Nicht verfügbarer Ressourcen
- Defekte Hardware

2.1.2 Lösungsansätze durch Circuit Breaker:

Mit Circuit Breaker kann man oben genannte Fehler behandeln. So kann es beispielsweise sein, dass ein gewünschter Service für längere Zeit ausfällt. Hier würde es keinen Sinn machen, immer wieder einen neuen Verbindungsaufbau zu starten. Mit Hilfe von Circuit Breaker kann der erneute Verbindungsaufbau gesteuert und somit untersagt werden.

Ein weiteres Beispiel ist der Umgang mit Services, die aufgrund hoher Anfragen überlastet sind. Hier kann es dazu kommen, dass das Warten auf den Service kritische Systemressourcen blockiert und somit das gesamte System zum Abstürzen bringen kann. Mit Circuit Breaker kann eine Regelung getroffen werden, die die Ressourcen in so einem Fall dann rechtzeitig wieder frei gibt. Daher werden sich konkurrierende Prozesse gemäht. So nutzen dann nur noch Services die Ressourcen, die auch tatsächlich verfügbar sind. Services, die nicht verfügbar sind, werden automatisch geblockt und die notwendigen Ressourcen wieder dem System zur Verfügung gestellt.

2.1.3 Funktionsweise von Circuit Breaker:

Grundsätzlich dienen Circuit Breaker dazu, dass man die Anfragen von Usern reduziert. Wenn ein Client beispielsweise eine Ressource anfordert, die zum Zeitpunkt nicht zur Verfügung steht, so fragt der Client ständig erneut den Server nach dieser Ressource an. Dies erzeugt erheblichen Traffic und kann im schlimmsten Fall zum Serverausfall führen. Das ist kein großes Problem, so fern wenige User vorliegen. Allerdings kommt es vor Allem bei Streaming Portalen dazu, dass zu einem Zeitpunkt sehr viele User ein und dieselbe Ressource anfordern. Hier kommen die sogenannten Circuit Breaker zum Einsatz. Diese unterbrechen den „normalen“ Kreislauf. Dies kann zum Beispiel ein Timeout sein, der dem User erst wieder nach einer gewissen Zeit erlaubt, die gewünschte Ressource anzufragen.

2.1.4 Die REST-Schnittstellen

Das Projekt wurde dabei mit REST-Schnittstellen implementiert. Hierbei wurde ein Service implementiert, der dazu dient, REST-Requests durchzuführen. Dies dient der Kommunikation zwischen Client und Server. In der Regel werden hier GET-Requests ausgeführt die der Client an den Server sendet. Dieser antwortet dann auf den Request und sendet, sofern eine gültige Anfrage vorliegt, eine entsprechende Antwort an den Client. Andernfalls wird eine entsprechende Fehlermeldung gesendet.

3 Der React Client

3.1 Vorstellung des Frameworks

React ist eine deklarative, effiziente und flexible JavaScript-Bibliothek für den Aufbau von Benutzeroberflächen. Es ermöglicht das Zusammenstellen komplexer UIs aus kleinen und isolierten Codestücken, die "Komponenten" genannt werden. Facebook veröffentlichte 2013 React als OpenSource Projekt und seitdem beeinflusst es die gesamte JavaScript Frontend Landschaft. Facebook, Instagram, AirBnB und weitere andere bekannte Anwendungen nutzen React.

React arbeitet mit einem virtuellen DOM, was das System sehr performant macht. Es ist modular aufgebaut, was React zu einem leicht zu lesenden und übersichtlichen Framework macht. Außerdem fordert dies die Flexibilität. Außerdem ist React beliebt, da es wenige Vorschriften und Bedingungen gibt. Das macht es bei Entwicklern beliebt, da es einfach ist bestehenden Code zu integrieren, ohne ein bestimmtes Muster bzw. Format zu benutzen.

Zusammenfassend hat React folgende Bestandteile:

- Komponenten basierend
- Virtueller DOM
- Browserkompatibilität

3.2 Erstellen eines React Projekts

Um einen React Client zu starten wird NodeJS auf der lokalen Maschine benötigt. Über die Konsole der lokalen Maschine werden entsprechende Ausführungen durchgeführt:

- Als erstes installiert man global mit dem Node Package Manager die Anwendung mit `npm install -g create-react-app`
- Im nächsten Schritt wird der Generator im ausgewählten Verzeichnis ausgeführt: `create-react-app my-app`
- In dem neu erstellten Verzeichnis wird nun das Startscript ausgeführt: `npm start`

Im Folgenden wird die Beispielanwendung des Studienprojekts dargestellt. Anstelle von „my-app“ wurde das React Projekt als „list“ erstellt. Hierzu im nächsten Abschnitt mehr.

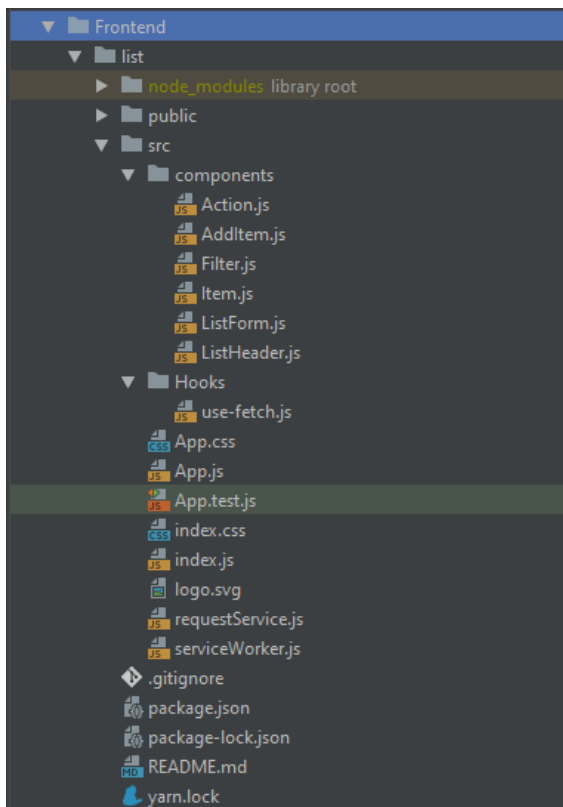


Abbildung 1: Projekt Struktur Frontend

3.3 Aufbau der Frontendanwendung

Für die Umsetzung der Anwendung haben wir uns eine Einkaufsliste ausgedacht. In der Anwendung wird eine Liste angezeigt, hier können Items hinzugefügt und wieder gelöscht werden. Die Anwendung erhält dabei die Daten aus der Hystrix oder Resilience4J Backendanwendung, hierzu müssen entsprechend die API Calls im Frontend angepasst werden. Hierzu wurde eine Funktion umgesetzt, mit der die Daten für das Frontend gefetchet werden. In der folgenden Darstellung wird eine Mock API eingebunden, alternativ ist der localhost auskommentiert dargestellt.

```
const itemApi = useFetch(  
  endpoint: "https://5cfabdcbf26e8c00146d0b0e.mockapi.io/tasks"  
/* const itemApi = useFetch(  
  "http://localhost:8080/" */  
);
```

Abbildung 2: Code Ausschnitt Frontend

Ehrenwörtliche Erklärung

Name:	Fischer	Vorname:	Heiko
Matrikel-Nr.:	751209	Studiengang:	SWB

Hiermit versichere ich, Heiko Fischer, dass ich die vorliegenden Studienarbeit mit dem Titel „Ein verteiltes System mit Angular, React, Spring Boot und Docker“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ort, Datum

Unterschrift

Ehrenwörtliche Erklärung

Name: Auer

Vorname: Patrick

Matrikel-Nr.: 755350

Studiengang: SWB

Hiermit versichere ich, Patrick Auer, dass ich die vorliegenden Studienarbeit mit dem Titel „Ein verteiltes System mit Angular, React, Spring Boot und Docker“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ort, Datum

Unterschrift