Hochschule Rosenheim, Fakultät für Informatik

Vorlesung: Verteilte Verarbeitung SS 2018 Prof. Dr. Gerd Beneken
Nachholprüfung WS 18/19 Lehrbeauftragter Thomas Mildner

Übungsblatt 2 von 3 – Praktikum als Grundlage für ein Kolloquium

Ziel dieses Praktikums

Dieses Praktikum hat zwei Ziele, erstens sollen Sie nach erfolgreichem Abschluss aller Praktikumsaufgaben verteilte Systeme besser verstehen und diese praktisch umsetzen können. Zweitens sollen Sie möglichst viele praktische Erfahrungen beim Programmieren sammeln, damit sie das Praxissemester möglichst erfolgreich überstehen.
Ziel ist es, dass Sie während des Programmierens möglichst umfangreich Feedback zu Ihren Ergebnissen erhalten und zwar von dem betreuenden Professor und auch von Ihren KommilitonInnen. Je mehr Feedback Sie erhalten, desto mehr und besser lernen Sie. Ich hoffe, dass dadurch auch Ihre Motivation steigt, sich in das Thema Programmierung zu vertiefen.
Alle Aufgaben sind jeweils über einen Zeitraum von etwa einem Monat zu bearbeiten. Das Semester wird in vier große Praktika unterteilt. Diese sind jeweils im Laufe des Semesters (weit vor dem Termin des Kolloquiums abzugeben).

Ziel der Aufgabe: RESTful WebServices und deren Absicherung

Wenn Sie diese Aufgabe erfolgreich abgeschlossen haben, können Sie

- Selbstständig RESTful-Webservices entwerfen und dokumentieren. Im Beispiel sogar mit OR-Mapper (Hibernate) und Profi-Applikationsserver (von Pivotal).
- Haben das Proxy-Pattern verstanden und können das Lost-Update Problem erklären
- Haben ein Grundverständnis für die Absicherung (TLS, Basic Auth, OAuth 2, JWT) von RESTful WebServices. Und Sie wissen, was Sie tun müssen um einen RESTful WebService **sicher** ins Internet zu hängen.

Achtung: Achten Sie bitte darauf, wenn Sie mit RESTful Webservices arbeiten, dass Sie das Architekturparadigma „Ressourcen Orientierte Architektur“ (R. Fielding) verstanden haben. Sie müssen die httpVerben auswendig kennen, wissen wozu Sie den http-Header verwenden und auch die Bedeutung von URIs in RESTful Architekturen sollten Sie erklären können. Nur laufender Code genügt hier nicht. Sie müssen die Theorie erklären können!
Schauen Sie sich bitte im Skript auch noch den Begriff des „Microservices“ an. Den in der Aufgabe erstellten Service könnte man als Microservice bezeichnen, eventuell müsste man die Verträge noch in einen eigenen Service auslagern.

Aufgabe:

1. Erstellen Sie einen RESTful WebService z.B. mit SpringBoot, wie in der Übung gezeigt. Wenn Sie technische Probleme mit der Erstellung haben, bitte fragen Sie noch mal nach einem Einzeltermin. Sie können auch andere Java-Frameworks für den Service verwenden z.B. Jersey mit einem einfachen Java EE-Container (Payara, TomEE). Oder eine andere Sprache (Python, JavaScript, …).
2. Der RESTful WebService verwaltet eine Datenstruktur bestehend aus Kunde (kundennummer, vorname, nachname, geburtstag), einer Adresse (strasse, postleitzahl, ort) sowie einer Liste von Verträgen des Kunden (vertragsnummer, vertragsart (Haftpflicht, Rechtsschutz, KFZ), jahresbeitrag). Damit haben Sie eine 1:1 Beziehung zwischen Kunde und Adresse und eine 1:n Beziehung zwischen Kunde und Versicherung. Hierzu implementieren Sie einen Service (anlegen, ändern, löschen, suchen) für Kunden und einen zweiten für Verträge (anlegen, ändern, löschen, suchen).
3. Bauen Sie einen Testtreiber-Client: der einige Kunden anlegt, ändert und löscht und entsprechendes für Verträge tut.
4. Dieser implementiert für die Kunden- wie für die Vertragsverwaltung einen **Proxy am Client**. Also eine am Client vorhandene Klasse „Vertragsverwaltung“ und eine am Client vorhandene Klasse

„Kundenverwaltung“, welche lokale Java-Aufrufe in entsprechende Aufrufe des RESTful Webserivces umsetzt.

5. Achten Sie beim Implementieren des Clients und der Datenstrukturen (Kunde, Vertrag etc.) drauf, dass es zu Lost-Updates kommen kann und überlegen Sie sich bitte eine Strategie, wie Sie diese entweder vermeiden (Pessimistische Strategie über Sperren) oder erkennen können (Optimistische Strategie z.B. über Versionszähler oder Zeitstempel). Lost-Update kann vorkommen, wenn zwei Clients denselben Kunden laden, denn beide haben eine Kopie des Datensatzes. Der zweite Client, der seine Änderungen dem Server mitteilt „gewinnt“ (vgl. DB1 Vorlesung bei Prof. Dr. Höfig).

Optionaler Teil:

6. Optionaler Teil: Implementieren Sie den Server so, dass sich der Client Authentisieren muss. Hierzu können Sie z.B. http-Basic-Auth verwenden. Dann müssen Sie auch eine Nutzerverwaltung mit integieren (vgl. Vorlesung).
7. Optionaler Teil: Machen Sie Ihren RESTful Webservice über https zugreifbar über ein (SelbstSigniertes) Zertifikat. Typischerweise würden Sie für diese Aufgabe einen Reverse-Proxy verwenden und das Zertifikat eher in einen vorgelagerten Dispatcher oder einen Webserver (z.B. Apache oder NGINX) einspielen.
8. Erzeugen Sie mithilfe eines Build-Skriptes einen Docker-Container. Dieser enthält im einfachsten Fall Ihre SpringBoot Implementierung. Sie müssen NICHT sicherstellen, dass beim Neustart des DockerContainers die Daten noch da sind.

Tipps:

- Erstellen Sie einen neuen Ordner im Repository mit dem Namen „02_RestfulWebservice“
- Erstellen Sie ein neues Projekt mit einer passenden .gitignore Datei
- Verwenden Sie Tools wie Maven / Gradle für die Abhängigkeiten
- Einige Tutorials für SpringBoot sind sehr umfangreich und decken mehr ab als in dieser Aufgabe gefordert, bitte nur das umsetzen was für diese Aufgabe benötigt wird.
- Es ist KEIN Frontend gefordert!