Hochschule Rosenheim, Fakultät für Informatik

Vorlesung: Verteilte Verarbeitung SS 2019 — Prof. Dr. Gerd Beneken, MSc. 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 http-Verben 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!

Aufgabe RosenheimerKleiderKreisel[1]

1. Erstellen Sie einen RESTful WebService mit SpringBoot in der Sprache Java, wie in der Übung gezeigt. Wenn Sie technische Probleme mit der Erstellung haben, bitte fragen Sie noch mal nach einem Einzeltermin.
2. Der RESTful WebService ist das Backend zum RosenheimerKleiderKreisel.
    a. Ein **Kleindungsstück (Klasse Kleidung)** hat einen Neupreis (in Euro) und einen Tauschwert (in Euro), der zwischen 10% und 50% des Neupreises liegen muss. Es hat zusätzlich eine Kleidergröße, ein Geschlecht (Damen, Herren), einen Typ (Hose, Kleid, Hemd, Bluse, …) und einen Hersteller (z.B. H&M, Levis) sowie ein Foto (optional).
    b. Ein **Mitglied** kann bis zu zehn Kleidungsstücke in die Tauschbörse einstellen.

[1] Die Originalaufgabe stammt von Stefan Priebsch (https://priebsch.de/, https://thephp.cc) aus der Vorlesung Professionelle Webprogrammierung.

c. Ein **Mitglied** hat Vor- und Nachnamen, eine Mail-Adresse, eine Post-Adresse, einen Nick (Loginname, darüber wird der Benutzer eindeutig identifiziert) und ein Foto (optional). Sowie natürlich ein verschlüsselt gespeichertes Passwort. Jedes Mitglied hat zusätzlich einen Kontostand (in Euro).

d. Die Plattform erhält für jeden **Tauschvorgang** 1 EUR. Dieser ist zu beiden Teilen von den Tauschpartnern (= Mitgliedern) zu tragen. Für den Tauschvorgang wird eine ID generiert und das Tauschdatum und beide Taschpartner gespeichert.

e. Bieten sie geeignete Such-Funktionen (GET-Requests) für Kleidung, Mitglieder und Tauschvorgänge an. Sodass sie einen Smartphone- oder Web-Client schreiben könnten, der die Tauschbörse umsetzt.

3. Bauen Sie einen Testtreiber-Client in Java und JUnit: Der einige Benutzer, Kleidungsstücke und Tauschvorgänge anlegt, ändert und löscht.
4. Testen Sie ihren Server mit Postman.
5. Erstellen sie ein neues Package (ggf. auch ein zweites Projekt), in dem sie den Client Code implementieren. Der Client implementiert für die Mitglieder, Kleidung und Tauschvorgänge einen **Proxy am Client**. Also eine am Client vorhandene Klasse „Mitgliederverwaltung“ und eine am Client vorhandene Klasse „Kleiderverwaltung“, welche lokale Java-Aufrufe in entsprechende Aufrufe des RESTful WebServices umsetzt.
6. Achten Sie beim Implementieren des Clients und der Datenstrukturen (Mitglied, Kleidung, Tauschvorgang 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 dasselbe Mitglied, denn beide haben eine Kopie des Datensatzes. Der zweite Client, der seine Änderungen dem Server mitteilt „gewinnt“ (vgl. DB1 Vorlesung bei Prof. Dr. Höfig bzw. Prof. Dr. Breunig).
7. 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 integrieren (vgl. Vorlesung).
8. Versionieren sie ihr REST-Interface!
9. Dokumentieren Sie ihr REST-Interface mit Swagger

Optionaler Teil (für die 1.0, ohne die optionalen Teile nur 1.3 oder schlechter)

10. Machen Sie Ihren RESTful Webservice über https zugreifbar über ein (Selbst-Signiertes) 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.
11. 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 Docker-Containers die Daten noch da sind.