Übungsblatt 9
Data-Grid und MapReduce mit Hazelcast

Aufgabe 1: Repliziertes DataGrid mit Hazelcast

In Teilaufgabe 1.1 aus Übungsblatt 8 lesen Sie innerhalb der Methode `getStudentById` `Student`-Objekte aus der Datenbank aus.

Nutzen Sie die Projektvorlage `uebung09_vorlage.zip` aus GRIPS oder erweitern Sie Ihr bisheriges Projekt und implementieren Sie darin folgende Funktionalität:

- Sichern Sie das neu erzeugte `Student`-Objekt nach Auslesen aus der Datenbank zusätzlich in einer Instanz einer `ReplicatedMap` namens `students` (Parameter bzw. Name, unter dem diese Map im Hazelcast-Data-Grid verteilt wird)
    - Die `ReplicatedMap` soll dieses Objekt für 5 Minuten vorhalten, anschließend gilt es als veraltet (vgl. „soft state“, dies wird in der `put`-Methode individuell angegeben)
- Prüfen Sie nun zu Beginn jeder Ausführung der Methode, ob zur angeforderten Matrikelnummer bereits ein Objekt in der `ReplicatedMap` vorliegt
- Sofern ein Objekt vorliegt, geben Sie dieses zurück, eine Abfrage der Datenbank ist in diesem Fall nicht notwendig

Aufgabe 2: MapReduce

Vorbemerkungen:
Das Netbeans-Projekt aus `uebung09_vorlage.zip` enthält weitere Klassen, die für diese Aufgabe notwendig sind:

Im Package `de.oth.vs.xml` finden Sie die Klasse `Veranstaltung`. Diese repräsentiert eine Veranstaltung bzw. einen Veranstaltungstipp:

```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class Veranstaltung implements Serializable {
    private static final long serialVersionUID = 1L;
    private String id;
    private String titel;
    private String beschreibung;
    private Date start;
    private Date ende;
    private String owner;
    // ...
}
```

Im Package `de.oth.vs.rest` finden Sie die Klasse `VeranstaltungResource`. Hierin sind bereits zwei Methoden zum Erzeugen und Lesen einer Veranstaltung implementiert. Die `Veranstaltung`-Objekte werden darin in einer `IMap` namens `"veranstaltungen"` im Hazelcast-Data-Grid gespeichert bzw. daraus gelesen.

Beispielaufrufe siehe Folgeseiten!

Ihre Aufgabe:
Der Rumpf der Methode `getVeranstaltungsTipps` (Klasse `VeranstaltungResource`) ist bereits vorgegeben. Erweitern Sie diese Methode um die Funktionalität, das Data-Grid mit Hilfe des Hazelcast-MapReduce-Algorithmus nach Veranstaltungen zu durchsuchen, deren Titel oder Beschreibung mindestens eines der übergebenen Suchworte enthalten. Das Ergebnis des Algorithmus ist eine `List<Veranstaltung>` mit allen Veranstaltungen, die zu den Suchwörtern passen.

Die Ergebnisliste sollte nach Relevanz sortiert werden, d. h. je mehr Suchwörter auf eine Veranstaltung zutreffen, um weiter oben in der Liste sollte diese Veranstaltung stehen. Es sollen grundsätzlich nur Veranstaltung berücksichtigt werden, die noch nicht abgeschlossen sind.

Lösen Sie diese Aufgabe mit Hilfe von MapReduce innerhalb von Hazelcast.

Implementieren Sie hierzu auch folgende Klassen im Pakage `de.oth.vs.mapreduce`:

Veranstaltungs-ID ↘ *Ein Suchwort* ↘

- `TippMapper implements Mapper<String, Veranstaltung, String, Veranstaltung>`

Ein Suchwort ↘

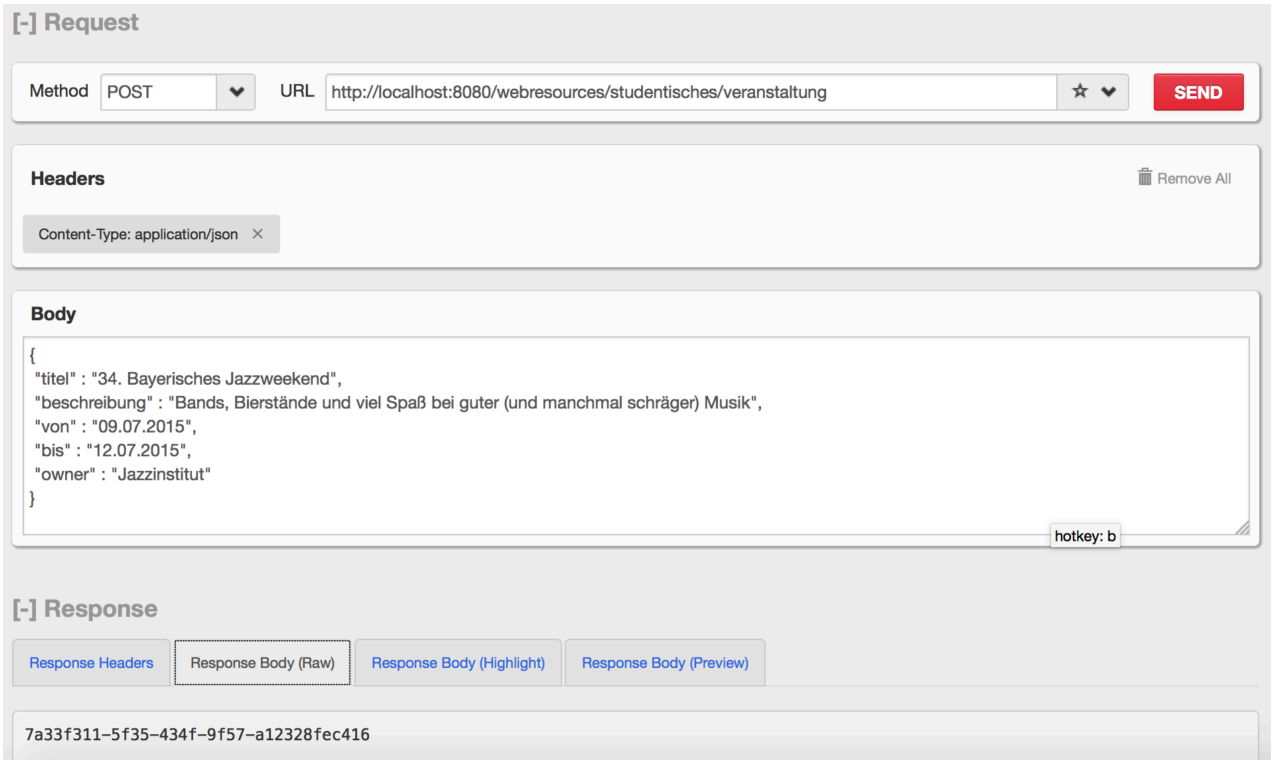
- `TippReducerFactory implements ReducerFactory<String, Veranstaltung, List<Veranstaltung>>`

Entries aus (Suchwort und Liste je Suchwort) ↘ ↘ *Gesamtliste für alle Suchworte* ↘

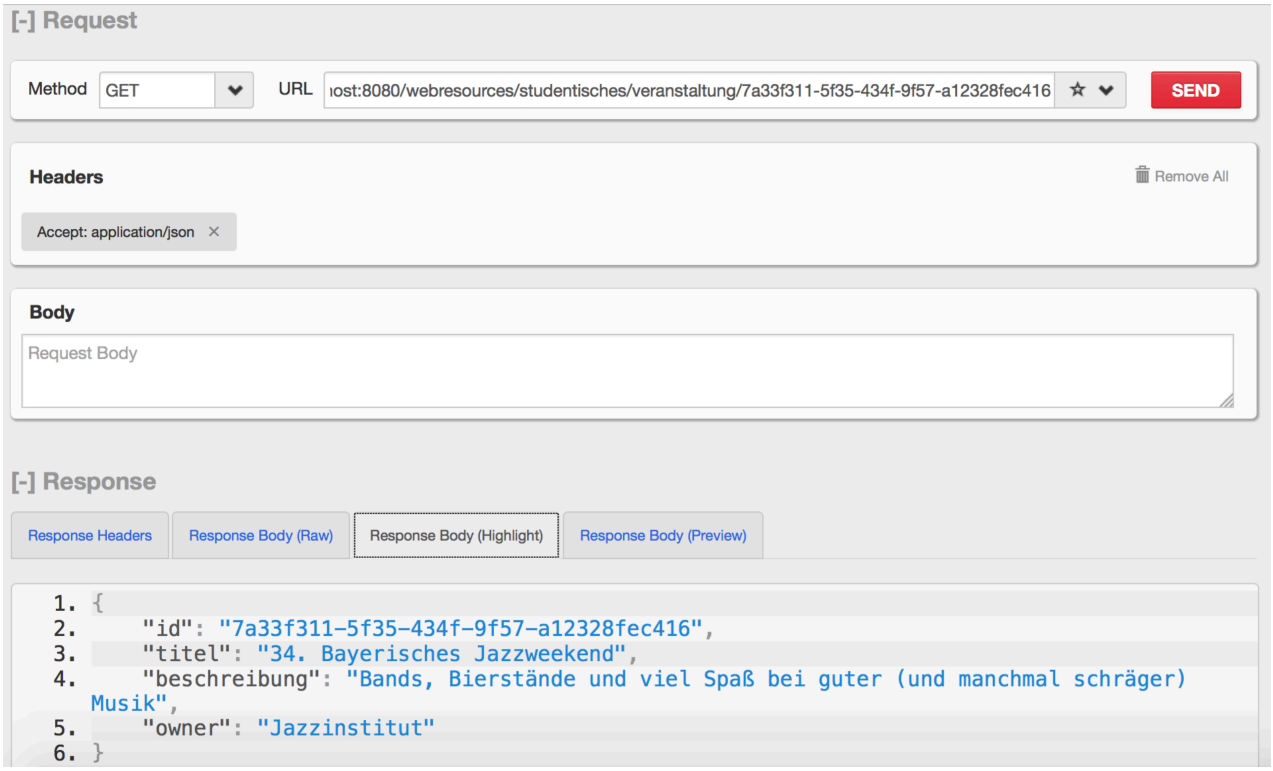
- `TippCollator implements Collator<Map.Entry<String, List<Veranstaltung>>, List<Veranstaltung>>`

Die Klasse `TippMapper` benötigt zusätzlich noch folgenden Konstruktor zur Übergabe der Suchwörter: `public TippMapper(String[] suchwoerter)`

Hinzufügen einer neuen Veranstaltung:



Abfragen der Details zu einer Veranstaltung (über dessen kryptische UUID):



Start einer MapReduce-Anfrage für Veranstaltungstipps zu den Suchwörtern `Musik` und `Regensburg`:
`http://localhost:8080/webresources/studentisches/veranstaltung/tipps/Musik+Regensburg`