Beschreibung der REST-Services

Im Rahmen der Vorlesung „Webbasierte Datenbankanwendungen“ haben wir es uns zur Aufgabe gemacht eine Fortbildungsdatenbank zu entwerfen. Diese soll Mitarbeiter und Seminare sowohl anlegen als auch löschen können. Des Weiteren soll die Fortbildungsdatenbank einzelne Mitarbeiter zu den angelegten Seminaren zuordnen und die dazugehörigen Informationen speichern. Um diese Services bereitstellen zu können, wurde Javascript in Verbindung mit JAX-RS (JAVA API for RESTful WebServices) verwendet und für jede Entity sowohl eine eigene Java-Klasse, als auch eine REST-Klasse (die auch in Java geschrieben ist) erstellt. Innerhalb der REST-Klassen wurden die Methoden, die zur Bereitstellung der Funktionalitäten der Fortbildungsdatenbank verantwortlich sind, ausprogrammiert. Im Folgenden wird exemplarisch aufgezeigt, wie sich die Benutzung eines REST-Services darstellt.

Möchte der Endbenutzer beispielsweise einen Mitarbeiter anlegen, so füllt er die entsprechenden Felder im Formular aus, welches hierfür zuständig ist und betätigt die Schaltfläche „Mitarbeiter anlegen“. Daraufhin wird ein Javascript-Code ausgeführt, der die einzelnen Werte aus den Formularfeldern entnimmt, einen AJAX-Aufruf bewirkt und die Daten mit Hilfe der Methode „open“ des XMLHttpRequest-Objekts an die Adresse der REST-Klasse übergibt, die in der „open“-Methode definiert ist. Anschließend wird die Methode in der REST-Klasse ausgeführt, welche ebenfalls in der „open“-Methode des XMLHttpRequest-Objekts definiert ist. Die Ausführung der richtigen Methode innerhalb der REST-Klasse wird durch entsprechende Annotationen, die den Methoden vorangestellt sind, sichergestellt. Im Folgenden wird ein Beispiel eines solchen AJAX-Aufrufs in Verbindung mit der Bereitstellung eines REST-Services dargestellt.

Beispiel für den Javascript-Teil der zu Erstellung der REST-Schnittstelle benötigt wird:

```
function mitarbeiter_anlegen(){

      var   p_nummer    = document.getElementById("p_nummer").value;
      var   vorname     = document.getElementById("vorname").value;
      var   nachname    = document.getElementById("nachname").value;
      var   stelle      = document.getElementById("stelle").value;
      var   e_mail      = document.getElementById("e_mail").value;
      var   telefon     = document.getElementById("telefonnummer").value;
      var geburtsdatum  = document.getElementById("geburtsdatum").value;
```

//Formulardaten werden hier ausgelesen

```
      var ajax = new XMLHttpRequest();
      ajax.open("POST", "webresources/entities.mitarbeiter",true);
      ajax.responseType = "json";
      ajax.setRequestHeader("Content-Type", "application/json")

      ajax.addEventListener("load", function() {
      window.alert("Datensatz erfolgreich angelegt!");
      });
      ajax.addEventListener("error", function() {
      window.alert("Datensatz konnte nicht angelegt werden");
      });

      ajax.send(JSON.stringify({
      vorName: vorname,
      nachName: nachname,
      pNummer: p_nummer,
      stelle: stelle,
      telefon: telefon,
      email:e_mail,
      geburtsdatum: geburtsdatum

          }));
```

//JSON wird als Anwortformat erwartet

// XMLHttpRequest-Objekt übergibt Daten an REST-Klasse

//JSON-Parameter, die gesendet werden

Beispiel für die REST-Klasse, der für die Bereitstellung der REST-Services benötigt wird:

```
@Stateless
@Path("entities.mitarbeiter")

public class MitarbeiterFacadeREST {
     @PersistenceContext(unitName = "RESTful_FB_DatabasePU")
     private EntityManager em;
     public MitarbeiterFacadeREST() {

    }
     @POST            //Annotation, die benötigt wird um die Post-Methode auszuführen
     @Consumes(MediaType.APPLICATION_JSON)

     public String create(String json) {
        Gson gson = new GsonBuilder().create();
        Mitarbeiter mitarbeiter = gson.fromJson(json,
        Mitarbeiter.class);
        mitarbeiter = em.merge(mitarbeiter);
       //Speichern des Datensatzes eines Mitarbeiters in der Datenbank

     return gson.toJson(mitarbeiter); }
```

// JSON als Ergebnis zurückliefern