Diplomski studij

Informacijska i komunikacijska tehnologija:

Telekomunikacije i informatika

Računarstvo:

Programsko inženjerstvo i informacijski sustavi

Računarska znanost

Ak.god. 2019./2020.

Raspodijeljeni sustavi

3a.

Arhitekture web-aplikacija, tehnologije weba - REST klijenti

Creative Commons

Zavod za telekomunikacije

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod sljedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s http://creativecommons.org/.

REST klijent

Kreiranje projekta

- File -> New -> Other...
  - Gradle Project
  - Project Name: billings-client
- Obrisati klase Library i LibraryTest

- Ovaj projekt se nalazi na https://gitlab.tel.fer.hr/spring/billings-client

Klijenti

- **URLConnection** - iz standardne Jave
- HttpClient - Java 11

- dodatne knjižnice:
  - Apache HTTP Client
  - **RestTemplete - iz Springa**
  - **Square Retrofit**
    - verzija 1.x
    - **verzija 2.x**

REST API

Primjer upita i odgovora - lista osoba

- GET /persons
- odgovor:

200 OK

```
[
   {
      "id"    : 1,
      "name"  : "Ignac Lovrek",
   },
   {
      "id"    : 2,
      "name"  : "Ivana Podnar Žarko",
   },
   {
      "id"    : 3,
      "name"  : "Mario Kušek",
   },
   {
      "id"    : 4,
      "name"  : "Krešimir Pripužić",
   },
...
]
```

Primjer upita i odgovora - stvaranje resursa

- POST /persons
- sadržaj zahtjeva:

```
{
   "firstName": "Jura",
   "lastName": "Jurić",
   "address": "Unska 3, 10000 Zagreb",
   "phone": "+385 1 6129 999",
   "email: "jura.juric@fer.hr"
}
```

- odgovor:

201 Created
Location: http://localhost:8080/persons/4

Primjer upita i odgovora - stvaranje resursa

- POST /persons/2/bills
- sadržaj zahtjeva:

```
{
   "month": 3,
   "year": 2019,
   "amount": 250
}
```

- odgovor:

201 Created
Location: http://localhost:8080/bills/1

Primjer upita i odgovora - lista računa

- GET /persons/2/bills
- odgovor:

200 OK


```
[
  {
    "id": 1,
    "period": "2019-3"
  },
  {
    "id": 3,
    "period": "2019-4"
  },
...
]
```

Primjer upita i odgovora - jedan račun

- GET /bills/1
- odgovor:

200 OK

```
{
  "month": 3,
  "year": 2019,
  "amount": 250
}
```

Scenarij

- Ispisati popis osoba
- Napraviti novu osobu
- Ispisati popis osoba
- Novoj osobi dodati račune
- Ispisati popis računa nove osobe
- Ispisati jedan račun

Sučelje


```
public interface RestInterface {
    List<ShortPerson> getListOfPersons();
    int newPerson(Person person);

    int newPersonBill(int personId, Bill bill);
    List<ShortBill> getPersonBills(int personId);
    Bill getBill(int billId);
}
```

Tvornica


```
public class RestFactory {

    public static RestInterface getUrlConnectionImplementation(String url) {
        return new UrlConnectionImplementation(url);
    }

    public static RestInterface getRestTemplateImplementation(String url) {
        return new RestTemplateImplementation(url);
    }

    public static RestInterface getRetrofitImplementation(String url) {
        return new RetrofitImplementation(url);
    }
}
```

Scenarij


```
public static void main(String[] args) {
    String url = "http://localhost:8080";

    runScript(RestFactory.getUrlConnectionImplementation(url));
    runScript(RestFactory.getRestTemplateImplementation(url));
    runScript(RestFactory.getRetrofitImplementation(url));
}

private static void runScript(RestInterface rest) {
    System.out.println("=============== " + rest.getClass().getName());
    System.out.println("== Person List");
    System.out.println(rest.getListOfPersons());
    System.out.println("== Create Person");
    System.out.print("personId=");
    int personId = rest.newPerson(new Person("Pero", "Kvržica", "Unska 3", "nema", "pk@fer.hr"));
    System.out.println(personId);
    System.out.println("== Person List");
    System.out.println(rest.getListOfPersons());
    System.out.println("== Create Bills");
    System.out.print("billId=");
    System.out.println(rest.newPersonBill(personId, new Bill(9, 2019, 245)));
    System.out.print("billId=");
    System.out.println(rest.newPersonBill(personId, new Bill(10, 2019, 189)));
    System.out.println("== Person Bills");
    List<ShortBill> personBills = rest.getPersonBills(personId);
    System.out.println(personBills);
    System.out.println("== First Bill");
    System.out.println(rest.getBill(personBills.get(0).getId()));
}
```

URLConnection

build.gradle


```
apply plugin: 'java-library'

repositories {
    jcenter()
}

dependencies {
   compile 'com.fasterxml.jackson.core:jackson-core:2.10.0'
    compile 'com.fasterxml.jackson.core:jackson-databind:2.10.0'
}
```

Klasa Person


```
package hr.fer.spring.client;

public class Person {
  private String firstName, lastName, address, phone, email;

  // konstruktori, setteri i getteri, toString
}
```

- Isto napraviti i za ShortPerson, Bill i ShortBill
- Umjesto ručnog generiranja klasa za JSON možemo koristiti uslugu:
  - http://www.jsonschema2pojo.org ili
  - http://pojo.sodhanalibrary.com

Klasa UrlConnectionImplementation (1)


```
public class UrlConnectionImplementation implements RestInterface {

    private String baseUrl;
    private ObjectMapper mapper;

    public UrlConnectionImplementation(String url) {
        this.baseUrl = url;
        mapper = new ObjectMapper();
}

    private String loadData(String textURL) {
        try {
            URL url = new URL(textURL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Accept", "application/json");
            if (connection.getResponseCode() != 200) {
                return "" + connection.getResponseCode();
            }

            InputStream contentStream = connection.getInputStream();
            String text = readTextFromStream(contentStream);
            connection.disconnect();
            System.out.println("URL: učitao sam: " + text);
            return text;
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

Klasa UrlConnectionImplementation (2)


```
private String readTextFromStream(InputStream contentStream) throws IOException {
    BufferedReader reader = new BufferedReader(new InputStreamReader(contentStream));

    StringBuffer sb = new StringBuffer();
    String line = reader.readLine();
    while (line != null) {
        sb.append(line);
        sb.append("\n");
        line = reader.readLine();
    }
    String text = sb.toString();
    return text;
}
```

Klasa UrlConnectionImplementation (3)


```
private HttpURLConnection writeData(String textURL, String method, String json) {
    try {
        URL url = new URL(textURL);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod(method);
        connection.setRequestProperty("Accept", "application/json");
        connection.setRequestProperty("Content-type", "application/json");
        connection.setDoOutput(true);
        OutputStream output = connection.getOutputStream();
        output.write(json.getBytes(StandardCharsets.UTF_8));
        output.close();

        connection.getResponseCode();
        return connection;
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

Klasa UrlConnectionImplementation (4)


```
@Override
public List<ShortPerson> getListOfPersons() {
    try {
        String jsonText = loadData(baseUrl + "/persons");
        return mapper.readValue(jsonText, new TypeReference<List<ShortPerson>>(){});
    } catch (JsonProcessingException e) {
        throw new RuntimeException(e);
    }
}

@Override
public int newPerson(Person person) {
    try {
        String personJson = mapper.writeValueAsString(person);
        HttpURLConnection connection = writeData(baseUrl + "/persons", "POST", personJson);
        String location =  connection.getHeaderField("Location");
        return Integer.parseInt(location.substring(location.lastIndexOf('/')+1));
    } catch (JsonProcessingException e) {
        throw new RuntimeException(e);
    }
}

...
```

Spring RestTemplate

build.gradle - popravljeni


```
apply plugin: 'java-library'

repositories {
    jcenter()
}

dependencies {
    compile 'com.fasterxml.jackson.core:jackson-core:2.10.0'
    compile 'com.fasterxml.jackson.core:jackson-databind:2.10.0'

    // RestTemplate
    compile 'org.springframework:spring-web:5.2.0.RELEASE'
}
```

Klasa RestTemplateImplementation (1)


```
public class RestTemplateImplementation implements RestInterface {

    private String baseURL;
    private RestTemplate restTemplate;

    public RestTemplateImplementation(String url) {
        this.baseURL = url;

            restTemplate = new RestTemplate();
            restTemplate.getMessageConverters()
                .add(new MappingJackson2HttpMessageConverter());
    }
```

Klasa RestTemplateImplementation (2)


```
@Override
public List<ShortPerson> getListOfPersons() {
    List<ShortPerson> courseList = restTemplate.getForObject(
          baseURL + "/persons",
          ShortPersonList.class);
    return courseList;
}

public static class ShortPersonList extends LinkedList<ShortPerson> {}
```

```
@Override
public int newPerson(Person person) {
    ResponseEntity<String> response = restTemplate.postForEntity(baseURL + "/persons",
      person, String.class);

    String locationPath = response.getHeaders().getLocation().getRawPath();
    return Integer.parseInt(locationPath.substring(locationPath.lastIndexOf('/') + 1));
}
```

Retrofit

http://square.github.io/retrofit/

build.gradle


```
apply plugin: 'java'

repositories {
    jcenter()
}

dependencies {
  // osnovno za retrofit
  compile 'com.squareup.retrofit2:retrofit:2.6.2'
  compile 'com.squareup.retrofit2:converter-jackson:2.6.2'

  // retrofit za Swing UI
  // compile 'io.reactivex:rxswing:0.27.0'
}
```

Klase PersonApi, BillApi


```
public interface PersonApi {
    @GET("/persons")
    Call<List<ShortPerson>> getPersons();

    @POST("/persons")
    Call<Void> newPerson(@Body Person person);

    @GET("/persons/{id}")
    Call<Person> getPerson(@Path("id") int personId);

    @POST("/persons/{pid}/bills")
    Call<Void> newBill(@Path("pid") int personId, @Body Bill bill);

    @GET("/persons/{pid}/bills")
    Call<List<ShortBill>> getPersonBills(@Path("pid") int personId);
}


public interface BillApi {
    @GET("/bills/{id}")
    Call<Bill> getBill(@Path("id") int billId);
}
```

Klasa RetrofitImplementation (1)


```
public class RetrofitImplementation implements RestInterface {

    private String baseURL;
    private PersonApi personApi;
    private BillApi billApi;

    public RetrofitImplementation(String url) {
        this.baseURL = url;
        Retrofit retrofit = new Retrofit.Builder().baseUrl(url)
          .addConverterFactory(JacksonConverterFactory.create())
          .build();

        personApi = retrofit.create(PersonApi.class);
        billApi = retrofit.create(BillApi.class);
    }
```

Klasa RetrofitImplementation (2)

Zavod za telekomunikacije

```
@Override
public List<ShortPerson> getListOfPersons() {
    try {
        return personApi.getPersons().execute().body();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

```
@Override
public int newPerson(Person person) {
    try {
        Response<Void> response = personApi.newPerson(person).execute();
        String location = response.headers().get("Location");
        return Integer.parseInt(location.substring(
          location.lastIndexOf('/')+1));
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
...
```