LINKgroup

Distance Learning System

Java XML i web servisi

Uvod u web servise

Šta je REST?

- Representational State Transfer ili skraćeno REST je koncept koji je zasnovan na veb-standardnima i HTTP protokolu. REST je prvi put definisao 2000. godine Roj Filding. Centralni deo REST arhitekture zauzima resurs. Resursu se pristupa preko zajedničkog interfejsa uz korišćenje HTTP metoda. Stoga se u ovakvoj arhitekturi ne koristi XML, a postiže se identična platformska i jezička nezavisnost i otpornost na firewall. REST omogućava resursima da imaju različitu reprezentaciju, kao što je na primer tekst, XML, JSON i tako dalje

Arhitektura REST-a

- REST koncept koristi URL-ove za reprezentaciju objekata i HTTP metod za rukovanje njima. Najpoznatije metode su **GET** za preuzimanje i **POST** za kreiranje novih podataka
- Za ažuriranje podataka koristi se HTTP metod **PUT**
- **DELETE** HTTP metod možemo da koristimo za brisanje. Parametre prosleđujemo kroz URL String, pa je jedino važno prilikom korišćenja proveriti HTTP metod. Ukoliko je metod DELETE, brišemo po ključu prosleđenom kroz URL. Ukoliko nije, biće izvršena neka druga operacija (najverovatnije preuzimanje podataka za zadati parametar)

Metoda	Opis
GET	čita resurs
PUT	kreira resurs
DELETE	uklanja resurs
POST	ažurira postojeći ili kreira novi resurs

RESTFul veb-servisi

- Veb-servisi koji se baziraju na upravo opisanoj REST arhitekturi mogu se nazvati **RESTFul** veb-ervisima. RESTFul veb-servisi uglavnom definišu URI na kojem se servis nalazi, podržane tipove, koji mogu biti XML, text, JSON ili pak korisnički definisani i na kraju set operacija koje će podržavati servis.

RESTFul i Java

- Java definiše podršku za REST arhitekturu kroz Java Specification Request 311.
- Ova specifikacija se drugačije naziva **JAX-RS** ili The Java API for RESTful Web Services. Konkretna implementacija ove specifikacije se naziva ***Jersey*** i ona definiše biblioteke koje implementiraju pomenutu specifikaciju u Java servlet kontejneru.
- Jednostavno, ova implementacija postoji na serveru i zadužena je za to da zahtev upućen određenom metodom mapira na odgovarajuću metodu same klase. Da bi nešto ovako bilo moguće, upotrebljavaju se JAX-RS anotacije.

Anotacije

- Primarni element u realizaciji RESTful servisa u Javi čine JAX-RS anotacije. Ove anotacije rukovode ponašanjem servisa, odnosno Java metoda i klasa koje dekorišu i ponašaju se slično JAX-WS anotacijama, koje smo koristili za SOAP servise u prethodnim lekcijama.
- Java set anotacija neophodnih za dekorisanje klase za RESTful servis nalazi se u javax.ws.rs paketu. Jednom upoznate i savladane, ove anotacije omogućavaju brzo i jednostavno kreiranje efikasnih, portabilnih i sistemski nezavisnih veb-servisa.
- JAX-RS između ostalog, mapira različite anotacije na različite HTTP metode: POST, GET, PUT, DELETE, OPTION i HEAD. Iako ove metode možemo hipotetički iskoristiti kako god želimo, postoje određena pravila koja su podrazumevana ukoliko hoćemo da servis koji kreiramo bude u skladu sa REST standardima.

Tabela Rest anotacija

Anotacija	Opis
@PATH(your_path)	postavlja putanju
@POST	ukazuje na metodu koja će odgovarati na HTTP POST zahtev
@GET	ukazuje na metodu koja će odgovarati na HTTP GET zahtev
@PUT	ukazuje na metodu koja će odgovarati na HTTP put zahtev
@DELETE	ukazuje na metodu koja će odgovarati na HTTP DELETE zahtev
@Produces(MediaType.TEXT_PLAIN[, more-types])	definiše MIME tip koji će biti vraćen od metode dekorisane sa @GET
@Consumes(type[, more-types])	definiše koji MIME tip će metoda koristiti
@PathParam	@PathParam koristi se za ubacivanje vrednosti iz URL u parametre metode

Kreiranje REST klijenta

- Jax-rs sadrži klase i metode za dobavljanje resursa
- Resursi se direktno dobavljaju klasom **WebTarget**

```
Client client = ClientBuilder.newClient();
WebTarget wt = client.target("https://global.api.pvp.net/api/lol/static-data/euw/v1.2");
WebTarget wtwres = wt.path("champion");
WebTarget wtwpar = wtwres.queryParam("api_key", "a9b12261-f6a6-4a43-8efa-208f8db976d0");
Invocation.Builder req = wtwpar.request();
Response res = req.get();

String output = res.readEntity(String.class);

JSONObject json = new JSONObject();
JSONParser parser = new JSONParser();
json = (JSONObject) parser.parse(output);

JSONObject data = (JSONObject) json.get("data");

for(Object key : data.keySet()){
    JSONObject heroJson = (JSONObject) data.get(key);
    out.print(heroJson.get("name"));
}
```