RESTful web servisi

Osnovni principi

- Sve može biti identifikovano kao resurs i svaki resurs se može jedinstveno identifikovati preko URI (Uniform Resource Identifier)
- Resurs može biti prikazan u različitim formatima, i format se definiše tipom medija (XML, JSON)
- Komunikacija između klijenta i servera je bez stanja. Svako stanje u komunikaciji mora se slati eksplicitno prilikom svake interakcije.

Klijent server komunikacija

- Klijenti koji koriste RESTful web servise komuniciraju sa serverom koristeći HTTP protokol
- Prilikom komunikacije koriste URI da identifikuju resurs
  - http://localhost:8080/resursi/resurs
- Prilikom komunikacije učesnici u komunikaciji koriste određenu HTTP metodu radi izvršavanja određene akcije nad određenim servisom (GET, POST, PUT, DELETE,...)
- Prilikom slanja zahteva klijent od servera očekuje odgovor

HTTP metode

- GET – zahtev za dobijanje resursa
- POST – zahtev za kreiranjem/ažuriranjem resursa
- PUT – zahtev za kreiranjem/ažuriranjem resursa
- DELETE – zahtev za brisanjem resursa

HTTP metode

- GET messages/10 – klijent sa servera uzima poruku sa identifikatorom 10
- PUT messages/10 – klijent menja na serveru poruku sa identifikatorom 10
- DELETE messages/10 – klijent briše sa servera poruku sa identifikatorom 10
- POST messages – na serveru se kreira nova poruka čiji će se identifikator automatski odrediti na osnovu zauzetih identifikatora

HTTP metode

- GET messages - klijent sa servera uzima sve poruke
- DELETE messages/10/comments – sa servera se brišu svi komentari za poruku sa identifikatorom 10
- POST messages/10/comments – kreira novi komentar za poruku sa identifikatorom 10
- PUT messages/10/comments – ažurira sve komentare za poruku sa identifikatorom 10 novom listom komentarima koji se nalaze u HTTP zahtevu

REST odgovor

- Nakon što klijent pošalje HTTP zahtev serveru, server klijentu šalje HTTP odgovor koji sadrži reprezentaciju resursa u različitim formatima (XML, JSON...).
- REST (Representational state transfer) šalje se samo reprezentacija resursa

Struktura HTTP odgovora

- Message Start-Line
- Headers (meta podaci)
  - Message Length
  - Date
  - Content Type – definiše šta se nalazi u Body segmentu
- Body (HTML, XML, JSON resurs)

HTTP struktura

- Message Start-Line je prva linija HTTP odgovora koja daje informacije o statusu:
- Grupe statusnih kodova opisuju:
  - 1XX – Informaciju (zahtev je pristigao i proces se nastavlja)
  - 2XX - Uspešnost (akcija je primljena, razumljiva i prihvaćena)
  - 3XX – Redirekciju (radi kompletiranja zahteva mora se uraditi još neka akcija)
  - 4XX – Klijentsku grešku (klijent je poslao loš zahtev, tj. zahtev sintaksno nije dobar)
  - 5XX – Serversku grešku (server ne može da ispuni ispravan klijentski zahtev)

Statusni kodovi

- 2XX
  - 200 OK – uspešan odgovor, generalno se može koristiti za sve metode
  - 201 Created – uspešno kreiran resurs, za POST metodu
  - 204 No content – server nema resurs koji može da vrati, za DELETE metodu
- 3XX
  - 302 Found, 307 Temporary Redirect – zatraženi resurs se nalazi na drugom URI-u
  - 304 Not Modified – zatraženi resurs nije menjan od poslednje potražnje, šalje se neizmenjen resurs

Statusni kodovi

- 4XX
  - 400 Bad Request – server ne razume zahtev
  - 401 Unauthorized – klijent nije uneo svoje korisničke podatke koji su serveru potrebni
  - 403 Forbidden – pristup određenom resursu je zabranjen tom korisniku
  - 404 Not Found – resurs nije pronađen
  - 415 Unsupported Media Type – server ne podržava zatražen tip formata za reprezentovanje resursa
- 5XX
  - 500 Internal Server Error – server je razumeo zahtev, pronašao resurs, ali nije uspeo, iz nekog razloga, da generiše odgovor

Statusni kodovi

- Preporučljivo je koristiti sledeće statusne kodove u većini slučajeva:
  - GET
    - uspeh: 200
    - neuspeh: 404
  - POST
    - uspeh: 200, 201
    - neuspeh: 409
  - PUT
    - uspeh: 200
    - neuspeh: 404
  - DELETE
    - uspeh: 200, 204
    - neuspeh: 404

JAX-RS

- Java API za RESTful web servise (JAX-RS) definiše standardni, anotacijom vođen API, koji pomaže pri konstrukciji RESTful web servisa i pri njigovom korišćenju.
- Jersey – java implementacija JAX-RS API-a.

JAX-RS – Server API

- RESTful resurse je posebno identifikovati na internetu.
- Identifikacija se vrši proširivanjem klase Application i njenim anotiranjem anotacijom @ApplicationPath ("appPath")
- URI_resursa = URI_aplikativnog_servera + "/" + appPath + "/" + resourcePath

```
@ApplicationPath("webresources")
class KompanijaZapPrevoz extends Application {
    :
}
```

http://localhost:8080/app/webresources

JAX-RS - Server API

- Jednostavan RESTful web servis može biti identifikovan koristeći anotaciju @Path
- Metoda servisa takođe može biti antoirana anotacijom @Path tako da će biti pozvana samo kada je identifikovana adresom koja je definisana anotacijom @Path

```
@Path("orders")
public class OrderResource {
    @GET
    public List<Order> getAll() {
        :
    }

    @GET
    @Path("{old}")
    public Order getOrder(@PathParam("old") int id) {
        :
    }
}
```

JAX-RS - Server API

- URI može proslediti upitne parametre koristeći ime/vrednost parove. Oni mogu biti mapirani u parametre ili polja koristeći @QueryParam anotaciju.

```
@GET
public List<Order> getAll(@QueryParam("start") int from,
                          @QueryParam("page") int page) {
    :
}
```

- Resursu se može pristupiti sa:

http://localhost:8080/app/webresources/orders?start=10&page=20

JAX-RS - Server API

- Za vezivanje metoda sa HTTP zahtevima koriste se sledeće anotacije:

HTTP method	JAX-RS annotation
GET	@GET
POST	@POST
PUT	@PUT
DELETE	@DELETE
HEAD	@HEAD
OPTIONS	@OPTIONS

JAX-RS - Server API

- GET

```
@Path("orders")
public class OrderResource {
    @GET
    public List<Order> getAll() {
        :
    }

    @GET
    @Path("{oid}")
    public Order getOrder(@PathParam("oid") int
id) {
        :
    }
}
```

http://localhost:8080/app/webresources/orders
http://localhost:8080/app/webresources/orders/1

JAX-RS - Server API

- POST

  U primeru sa desne strane dat je jedan od načina upotrebe POST metode koja se obično koristi na web stranicama radi popunjavanja forme. U opstem slučaju se POST kao i ostalim HTTP metodama može na server slati bilo koji tip podataka (ne samo podaci sa forme).

  @FormParam se koristi da veže polje u formi sa parametrom u metodi

```
@POST
@Path("create")
public Order createOrder(@FormParam("id") int id,
            @FormParam("name") String name) {
    return new Order(id, name);
}

Form form = new Form();
System.out.println("Id:");
form.param("id", br.readLine());
System.out.println(”Name:");
form.param("name", br.readLine());

<form method="post"
            action="websources/orders/create">
Id: <input type="text" name="id"/><br/>
Id: <input type="text" name=”name"/><br/>
</form>
```

JAX-RS - Server API

- PUT

```
@PUT
@Path("{id}")
@Consumes ("*/xml")
public Order putOrder(@PathParam("id") int id, String context) {
    Order order = findorder(id);
    //update order
    :
    return  order;
}
```

JAX-RS - Server API

- DELETE

```
@DELETE
@Path("{id}")
public Order deleteOrder(@PathParam("id") int id) {
    Order order = findorder(id);
    //delete order
}
```

JAX-RS - Server API

- @Produces anotacija definiše koje tipove reprezentacije će sve metoda moći da proizvede u HTTP odgovoru
- HTTP Accept zaglavlje u HTTP zahtevu definiše tip koji se traži

```
@GET
@Path("{id}")
@Produces({"application/xml", "application/json"})
public Order getOrder(@PathParam("id") int id) {
      :
}
```

JAX-RS - Server API

- Mogu se koristiti i džoker obrasci kao što je application/*. U konkretnom slučaju getOrder metoda će moći da pruža odgovor u application/xml, application/json ili nekom drugom formatu.

```
@GET
@Path("{id}")
@Produces({"application/*"})
public Order getOrder(@PathParam("id") int id) {
      :
}
```

JAX-RS - Server API

- @Consumes anotacija defineše koje tipove reprezentacije će sve metoda moći da dohvati kroz HTTP zahtev
- HTTP Content-Type zaglavlje u HTTP zahtevu definiše koji tip reprezentacije klijent šalje.
- U sledećem primeru metoda getOrder prihvata dva tipa reprezentacije: application/xml, application/json.

```
@POST
@Consumes({"application/xml", "application/json"})
public Response addOrder(Order order) {
      :
}
```

JAX-RS - Server API

- JAX-RS 2.0 omogućava da na serveru definišemo prednost za određene tipove za reprezentaciju koristeći qs (quality on server) parametar.
- U primeru ispod, ukoliko klijent nije naveo vrednost HTTP Content-Type biće izabran onaj sa većom vrednošću qs-a

```
@POST
@Consumes({"application/xml; qs=0.75", "application/json; qs=1"})
public Response addOrder(Order order) {
      :
}
```

- Parametar qs se može koristiti i kod @Produces anotacije kada ukoliko klijent u HTTP Accept zaglavlju definisao tip koji želi da dobije, ili je tražio application/* bira se onaj sa najvećim vrednošću qs-a.

JAX-RS – Client API

- JAX-RS 2 poseduje novi Client API koji se može koristiti za pristup web resursima.
- GET

```
Client client = ClientBuilder.newClient();
Order order = client
    .target("htttp://localhost:8080/app/webresources/orders")
    .path("{oid}")
    .resolvetemplate(("oid",1)
    .request()
    .get(Order.class);
```

JAX-RS – Client API

- POST

```
Client client = ClientBuilder.newClient();
Order order = client
    .target("htttp://localhost:8080/app/webresources/orders")
    .request()
    .post(Entity.entity(new Order(1), "application/json“, Order.class);
```

JAX-RS – Client API

- DELETE

```
Client client = ClientBuilder.newClient();
Order order = client
    .target("htttp://localhost:8080/app/webresources/orders")
    .path("{oid}")
    .resolvetemplate(("oid",1)
    .request()
    . delete();
```