Web programiranje – Vežbe 10
REST

Primer – upravljanje proizvodima

HTML stranica **webshop.html** sadrži tabelu sa listom proizvoda. Proizvodi se mogu dodavati, brisati i ažurirati. HTML i JS fajlovi se nalaze u **WebContent** folderu **WebShopREST** projekta. U **services** paketu ovog projekta se nalazi klasa **ProductService** koja nudi web interfejs za upravljanje proizvodima.

REST (**Re**presentational **S**tate **T**ransfer) je arhitekturalni stil za pisanje web servisa. Definiše pravila pristupanja i upravljanja resursima servisa na sledeći način (kroz primer entiteta User):

- Listanje korisnika: **GET /users**
- Dodavanje novog korisnika: **POST /users**
- Ažuriranje postojećeg korisnika: **PUT /users/123** (gde je 123 identifikator korisnika)
- Brisanje postojećeg korisnika: **DELETE /users/123** (gde je 123 identifikator korisnika)

REST servisi se mogu pisati za Apache Tomcat uz pomoć **Jersey** biblioteke u okviru **JAX-RS** specifikacije. Da bi projekat bio RESTful, moraju se uključiti potrebne biblioteke (videti **/WebContent/WEB-INF/lib** folder projekta) i dodati specifikacija posebnog Jersey servleta u **web.xml** fajl:

```
<!-- Obavezan Jersey REST Service servlet -->
 <servlet>
      <servlet-name>Jersey REST Service</servlet-name>
      <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
      <init-param>
            <param-name>jersey.config.server.provider.packages</param-name>
            <param-value>services,org.codehaus.jackson.jaxrs</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>Jersey REST Service</servlet-name>
      <!-- Bazna putanja do REST servisa. Može biti bilo šta (pa i prazan
string) -->
      <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Servisi za obrađivanje klijentskih zahteva se definišu kao obični Java objekti (**POJO – P**lain **O**ld **J**ava **O**bject). Ono što razlikuje servise od običnih klasa je korišćenje JAX-RS anotacija za definisanje servisa:

- **@Path:** Putanja do servisa / metode. Ako je definisana za klasu, to je bazna putanja do metoda klase. Ako je definisana za metodu, putanja do nje se dobija kao **putanjaDoKlase + putanjaDoMetode.**

- **@GET, @POST, @PUT, @DELETE:** Definiše za koju HTTP metodu je vezana anotirana metoda.
- **@Produces:** Tip odgovora koji vraća metoda. Odgovara **ContentType** zaglavlju HTTP odgovora.
- **@Consumes:** Tip zahteva koji traži metoda. Odgovara **ContentType** zaglavlju HTTP zahteva.

Anotirati klasu **ProductService** sledećom anotacijom:

```
@Path("/products")
```

Ovim je definisana bazna putanja do metoda klase ProductService. Pošto je bazna putanja do REST servisa "/rest" (**url-mapping** u **web.xml** fajlu gore), putanja do ove klase je **/rest/products**. Dodati metode za listanje i dodavanje proizvoda u istu klasu:

```
@GET
@Path("/")
@Produces(MediaType.APPLICATION_JSON)
public Collection<Product> getProducts() {
	ProductDAO dao = (ProductDAO) ctx.getAttribute("productDAO");
	return dao.findAll();
}

@POST
@Path("/")
@Produces(MediaType.APPLICATION_JSON)
public Product getProducts(Product product) {
	ProductDAO dao = (ProductDAO) ctx.getAttribute("productDAO");
	return dao.save(product);
}
```

Vrednosti parametri ovih metoda dolaze iz HTTP zahteva. Sldećim anotacijama se može označiti koji parametar odgovara kom delu HTTP zahteva:

- **@PathParam**: Parametar dobijen iz putanje. Npr. iz putanje oblika **"/rest/users/{id}"** se za konkretan URL **"/users/123"** može izvući vrednost **id="123"**.
- **@QueryPath**: Parametri upita (iza upitnika). Npr. za **"/rest/books?num=5", @QueryParam("num") int num** će izvući vrednost **num=5**.
- **@FormParam**: Parametar forme izvučen iz name atributa input polja.
- **@DefaultValue:** Podrazumevana vrednost, ako nije prisutna.
- **@Context:** Anotira **HttpRequest** objekat, koji predstavlja HTTP zahtev kao kod servleta. Korisno za praćenje sesije.

Ako nema anotacije, vrednost parametra se formira na osnovu sadržaja tela HTTP zahteva.

Metode servisa mogu vratiti jedan bean objekat ili kolekcija bean objekata koji će se konvertovati u oblik specificiran **@Produces** anotacijom (npr. JSON). Takođe se mogu vratiti primitivni tipovi, kao i poseban **Response** objekat kojim se može precizno formirati odgovor (npr. može se vratiti status 400 ako zahtev nije ispravan).

Zadaci

1. Ažuriranje proizvoda:
    - 1.1 Dodati metodu za ažuriranje proizvoda u klase **ProductDAO** i **ProductServis**. URL je oblika **PUT /products/{id}**, gde je id identifikator proizvoda (ključ iz mape). U slučaju da proizvod sa tim id-em ne postoji, kreirati novi.
    - 1.2 Proširiti fajl **webshop.html i webshop.js** tako da se pri selekciji proizvoda iz tabele otvara forma za izmenu proizvoda.
    - 1.3 Klikom na submit dugme se šalje zahtev za ažuriranje. Pri uspešnom kreiranju se na 3 sekunde prikazuje poruka zelenim slovima “Proizvod uspešno ažuriran“.
    - 1.4 Ažurirati vrednost proizvoda u tabeli.
2. Ažuriranje proizvoda:
    - 2.1 Dodati metodu za brisanje proizvoda u klase **ProductDAO** i **ProductServis**. URL je oblika **DELETE /products/{id}**, gde je id identifikator proizvoda (ključ iz mape). U slučaju da proizvod sa tim id-em ne postoji, vratiti **null**.
    - 2.2 Proširiti fajl **webshop.html i webshop.js** tako da pored svakog reda u tabeli stoji dugme za prisanje proizvoda.
    - 2.3 Klikom na dugme za brisanje se šalje zahtev za brisanje odgovarajućeg proizvoda. Pri uspešnom brisanju se proizvod iz tabele i na 3 sekunde se prikazuje poruka zelenim slovima “Proizvod uspešno obrisan“ .