Programski Sistemi

Web servisi i Java: JAX-RS

RESTful Web servisi

- Sa pojavom Weba 2.0, RESTful Web servisi su postali popularni
- Mnoge važne IT kompanije (Amazon, Google i Yahoo!) odbacile su SOAP servise u korist RESTful resursno-orijentisanih servisa
- Representational State Transfer (REST) je stil projektovanja zasnovan na načinu rada Weba
- U REST principu, svaka informacija je resurs, kome se pristupa preko URI-ja (Uniform Resource Identifier), što je obično link na Webu

Resursi i URI

- Resurs je bilo šta što bi klijent mogao da referencira, tj. bilo kakva informacija koja se može referencirati kao hiperlink
- Resurs se može čuvati u bazi, fajlu itd.
- Kao resurse treba izlagati jednostavne objekte, tj. treba izbegavati apstraktne pojmove
- Resursi se na Webu identifikuju pomoću jedinstvenih identifikatora, URI-ja
- URI čini ime resursa i strukturirana adresa koja označava gde će se pronaći resurs

URI

- Postoje različiti primeri URI-ja:
  - Web adrese (URL),
  - UDI (Universal Document Identifier),
  - Universal Resource Identifiers,
  - kombinacija URL i URN (Uniform Resource Names).
- URI-ji treba da budu što više opisni i da se odnose na jedinstveni resurs
- Pri tom, različiti URI-ji koji identifikuju različite resurse mogu da vode do istih podataka

URI: primeri

Resurs	URI
The catalog of Apress books	`http://www.apress.com/book/catalog`
The cover of the Java EE 6 book	`http://www.apress.com/book/catalog/beginning-javaee6.jpg`
Information about jobs at Apress	`http://www.apress.com/info/jobs`
The weather in Paris for 2008	`http://www.weather.com/weather/2008?location=Paris,France`
Interesting photos on Flickr for January 1st, 2009	`http://www.flickr.com/explore/interesting/2009/01/01`
Interesting photos on Flickr for the last 24 hours	`http://www.flickr.com/explore/interesting/24hours`
The list of adventure movies	`http://www.movies.com/categories/adventure`

Reprezentacija resursa

- Resurs može biti predstavljen kao tekst, XML, PDF dokument, JPG slika ili u nekom drugom formatu
- Klijent sa resursom uvek radi u nekom formatu, preko njegove reprezentacije; sam resurs ostaje na serveru
- Reprezentacija je bilo kakva korisna informacija o stanju resursa

Izbor reprezentacije resursa

- Postoje dva rešenja za pristup različitim reprezentacijama objekta
- Prvo je da se za svaku reprezentaciju definiše različit URI, npr.

  http://www.apress.com/book/catalog/java
  http://www.apress.com/book/catalog/java.csv
  http://www.apress.com/book/catalog/java.xml
- Drugo (bolje) rešenje je da se izloži isti URI za sve reprezentacije (http://www.apress.com/book/catalog/java), a da se zatim upotrebi mehanizam content negotiation

Content negotiation

- Postupak izbora najbolje reprezentacije za dati HTTP odgovor kada postoji više reprezentacija istog resursa
- Zasniva se na HTTP zaglavljima zahteva HTTP Accept, Accept-Charset, Accept-Encoding, Accept-Language i User-Agent
- Zaglavlje Accept definiše MIME tip; najčešće korišćeni tipovi su text/html, text/plain, image/gif, image/jpeg, image/gif, application/xml, application/json

Preslikavanje REST-a u HTTP Metode

- GET
  - Služi za čitanje reprezentacije resursa. GET mora biti idempotentan, tj. mora da ostavi resurs u istom stanju bez obzira da li se pozove jednom ili više puta.
- POST
  - Kreira nov resurs na osnovu URI-ja zahteva. POST nije idemotentan (ako se ponovi dvaput, biće kreirana dva nova resursa)
- PUT
  - Ažurira stanje resursa ako on postoji; ako ga nema, onda ga kreira; nije idempotentan.
- DELETE
  - Briše resurs zadat u URI-ju, idempotentan.

Od Weba do Web servisa (1)

- Za CRUD operacije koristimo HTTP metode, na primer:
  - POST zahtev sa podacima u nekom formatu (XML, JSON, ili tekstualnom) koristimo za pravljenje novog resursa na URI-ju http://www.apress.com/books/. Kada se napravi, kao odgovor dobijamo URI novog resursa: http://www.apress.com/books/123456.
  - GET zahtev koristimo za čitanje resursa preko URI-ja http://www.apress.com/books/123456.

Od Weba do Web servisa (2)

- PUT zahtev koristimo za ažuriranje resursa na URI-ju http://www.apress.com/books/123456.
- DELETE koristimo za brisanje resursa na URI-ju http://www.apress.com/books/123456.
- RESTful Web servisi imaju uniformni interfejs (HTTP metode i URI-je); kada znamo URI resursa, možemo da pozovemo odgovarajući HTTP metod
- Iako i SOAP koristi HTTP, da bismo koristili SOAP servis moramo da iz WSDL fajla otkrijemo semantiku servisa

Statelessness

- Važno svojstvo REST pristupa, koje znači da se svaki HTTP zahtev dešava potpuno izolovano, tj. server ne treba da prati zahteve koji se izvršavaju
- Treba razlikovati stanje resursa i stanje aplikacije; stanje resursa dele svi klijenti, a stanje aplikacije je u isključivoj nadležnosti klijenta
- Npr. ako klijent upotrebi GET da bi očitao stanje resursa, server ne zna ništa o tome
- Ovo važno svojstvo omogućuje skalabilnost, jer se ne prate sesije

REST i Java

- Da bi se napisao RESTful Web servis u Javi, potrebni su samo klijent i server koji podržavaju HTTP, ali bi takav kod bio težak za razumevanje i održavanje
- Razvoj RESTful servisa olakšava specifikacija JAX-RS; to je skup API-ja koji podržava REST stil projektovanja
- Počev od verzije1.1, JAX-RS je postao zvaničan deo Java EE 6

Jersey

- Jersey je referentna implementacija za JAX-RS
- To je open-source projekat koji nudi API-je za pisanje RESTful servisa i klijenata
- Postoje i druge implementacije za JAX-RS: CXF, RestEasy, Restlet

JAX-RS model (1)

- JAX-RS je po prirodi HTTP-centričan i ima skup jasno definisanih anotacija za HTTP metode i URI-je
- Pošto resurs može imati više reprezentacija, API podržava različite MIME tipove i koristi JAXB za maršaling i demaršaling XML i JSON reprezentacija u objekte i iz njih
- JAX-RS je takođe nezavisan od servlet kontejnera, pa se deploy resursa može obaviti pomoću GlassFisha ili bilo kog drugog kontejnera

JAX-RS model (2)

- Da bi klasa postala REST servis:
  - Mora biti anotirana sa **@javax.ws.rs.Path**
  - Da bi se omogućio rad sa bazom, tj. Enterprise Java Beans (EJB), treba dodati anotaciju **@javax.ejb.Stateless**
  - Klasa mora biti public, a ne sme biti final ili abstract.
  - Mora da ima javni default konstruktor
  - Ne sme da ima metod finalize().

Primer REST servisa (1)

```
@Path("books")
@Stateless
@Produces({"application/xml", "application/json"})
@Consumes({"application/xml", "application/json"})
public class BookResource {

    @Context
    private UriInfo uriInfo;
    @PersistenceContext(unitName = "chapter15PU")
    private EntityManager em;
    @GET
    public List<Book> getAllBooks() {
        Query query = em.createNamedQuery("findAllBooks");
        List<Book> books = query.getResultList();
        return books;
    }

    @POST
    public Response createNewBook(JAXBElement<Book> bookJaxb) {
        Book book = bookJaxb.getValue();
        em.persist(book);
        URI bookUri = uriInfo.getAbsolutePathBuilder().path(➥
                                book.getId().toString()).build();
        return Response.created(bookUri).build();
    }
```

Primer REST servisa (2)

- Kod REST servisa koji može da prima i pravi XML i JSON reprezentacije knjige
- Metoda getAllBooks() iz baze konstruiše listu knjiga i vraća XML ili JSON reprezentaciju ove liste (pomoću content negotiation), kojoj se pristupa metodom GET
- Metoda createNewBook() prihvata XML ili JSON reprezentaciju knjige i prezistira je u bazi podataka. Ovaj metod se poziva preko POST-a i vraća odgovor sa URI-jem nove knjige

Definicija URI-ja

- Vrednost anotacije **@Path** je relativna URI putanja; kada se ova anotacija koristi za klasu, odnosi se na korenski resurs, tj. koren stabla resursa
- Može se koristiti i u metodama korenskog resursa, a ne samo za klasu
- Može se koristiti i sa URI šablonima u kojima su ugrađene promenljive, npr. @Path("/books/{idbook}")

Primer anotacije @Path

```
@Path("/items")
public class ItemResource {

    // URI : /items
    @GET
    public List<Item> getListOfItems() {
        // ...
    }

    // URI : /items/1234
    @GET
    @Path("{itemid}")
    public Item getItem(@PathParam("itemid") String itemid) {
        // ...
    }

    // URI : /items/1234
    @DELETE
    @Path("{itemid}")
    public void deleteItem(@PathParam("itemid") String itemid) {
        // ...
    }

    // URI : /items/books
    @GET
    @Path("/books/")
    public List<Book> getListOfBooks() {
        // ...
    }

    // URI : /items/books/1234
    @GET
    @Path("/books/{bookid}")
    public Book getBook(@PathParam("bookid") String bookid) {
        // ...
    }
}
```

Konstruisanje URI-ja

- Ako anotacija @Path postoji i za klasu i za metodu, relativna putanja do metode resursa se dobija nadovezivanjem putanje klase i metode
- Na primer, za dobijanje knjige preko ID-ja, putanja bi bila/items/books/1234.
- Kada bi se tražio korenski resurs /items, bila bi izabrana jedina metoda bez anotacije @Path, tj. getListOfItems()
- Kada bi se tražio resurs /items/books, bila bi pozvana metoda getListOfBooks()

Prihvatanje i kreiranje MIME tipova

- JAX-RS podržava brojne Javine tipove za reprezentaciju resursa, npr. String, InputStream ili JAXB bin
- Anotacije **@javax.ws.rs.Consumes** i **@javax.ws.rs.Produces** mogu se primeniti na resurse u slučajevima kada može da postoji više reprezentacija
- One definišu MIME tipove reprezentacija koje se razmenjuju između klijenta i servera
- JAX-RS klasa **MediaType** služi kao apstrakcija za MIME tip

MIME tipovi definisani u klasi javax.ws.rs.core.MediaType

Constant name	MIME type
APPLICATION_ATOM_XML	"application/atom+xml"
APPLICATION_FORM_URLENCODED	"application/x-www-form-urlencoded"
APPLICATION_JSON	"application/json"
APPLICATION_OCTET_STREAM	"application/octet-stream"
APPLICATION_SVG_XML	"application/svg+xml"
APPLICATION_XHTML_XML	"application/xhtml+xml"
APPLICATION_XML	"application/xml"
MULTIPART_FORM_DATA	"multipart/form-data"
TEXT_HTML	"text/html"
TEXT_PLAIN	"text/plain"
TEXT_XML	"text/xml"
WILDCARD	"*/*"