ISS odgovori

Struktura HTTP zahteva

- Počinje redom: METOD putanja HTTP/verzija
- METOD je: GET, PUT, POST, DELETE...
- Putanja je: /nešto ili samo /
- Dodatni redovi sadrže atribute oblika: “ime : vrednost”
- Prazan red
- Opciono telo zahteva

Metod
GET = zahteva resurs od web servera
POST = šalje parametre forme i traži odgovor
HEAD = zahteva samo HTTP odgovor (response), bez slanja samog resursa
PUT = omogućava klijentu da pošalje resurs na web server
OPTIONS = od web servera se traži spisak metoda koje podržava
DELETE = omogućava klijentu da obriše resurs sa web servera

GET, POST, PUT i DELETE su dobili dodatne funkcionalnosti kod REST-a

Atributi u HTTP zahtevu
User-Agent - identifikuje web browser
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.2) Gecko/20070219 Firefox/2.0.0.2
Accept - definiše koje tipove resursa navigator prihvata kao odgovor na ovaj zahtev
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language - definiše koji jezik očekuje kao odgovor
Accept-Language: en-us,en;q=0.5
Accept-Encoding - definiše koje kodiranje očekuje kao odgovor
Accept-Encoding: gzip,deflate
Accept-Charset – definiše koju kodnu stranu očekuje

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Cookie – definiše mehanizam praćenja sesije
Cookie: id1172566682241_1=1172566682241_1
Referer – definiše URL sa kojeg se došlo na ovu stranicu
koristi se za statistiku
hotlinking
Referer: http://localhost/
Connection – HTTP1.1 "kaže" serveru da ne zatvara konekciju po isporuci resursa
Connection: Keep-Alive
q=broj definiše qvalue, odn. floating point vrednost "težine" parametra

Primer HTTP zahteva
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*
Accept-Language: sr
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Host: localhost
Connection: Keep-Alive (ako je close, konekcija se zatvara)

Struktura HTTP odgovora

- Počinje redom: **HTTP/verzija kod tekstualni_opis**
- odatni redovi sadrže atribute: "ime: vrednost"
- prazan red
- sledi sadržaj datoteke

Kod:
"101" ; Protocol upgrade
"200" ; OK
"201" ; Created
"202" ; Accepted
"204" ; No Content
"301" ; Moved Permanently
"302" ; Moved Temporarily

"304" ; Not Modified
"400" ; Bad Request
"401" ; Unauthorized
"403" ; Forbidden
"404" ; Not Found
"500" ; Internal Server Error
"501" ; Not Implemented
"502" ; Bad Gateway
"503" ; Service Unavailable

Atributi u HTTP odgovoru

Content-type – definiše tip odgovora
Content-Type: text/html
Cache-Control – definiše kako se keš na klijentu ažurira
koristi se i Pragma: no-cache
Cache-Control: no-cache
Location – definiše novu adrese kod redirekcije
Location: new.html
Connection –potvrda klijentu da li da zatvori konekciju ili da je ostavi otvorenu
Connection: Keep-Alive

Primer HTTP odgovora

HTTP/1.0 200 OK
Date: Tue, 04 May 02004 08:55:09 GMT
Status: 200
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.4.2_02; Windows XP 5.1 x86; java.vendor=Sun Microsystems Inc.)
Content-Type: text/html
Last-Modified: Fri, 24 Oct 02003 16:07:24 GMT
Content-Length: 2524
Content-Language: en

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <meta name="GENERATOR" content="Mozilla/4.72 [en] (WinNT; U) [Netscape]">
  <meta name="Author" content="Anil K. Vijendran">
  <title>Tomcat v3.1</title>
</head>
<body></body>
</html>

Primer HTTP odgovora sa redirekcijom

HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.1
Date: Mon, 26 Apr 2004 17:50:55 GMT
X-Powered-By: ASP.NET
Location: localstart.asp
Connection: **Keep-Alive (ako je close, konekcija se zatvara)**
Content-Length: 135
Content-Type: text/html
Set-Cookie: ASPSESSIONIDGGQQQAEK=JKKPPFKCMNDNMEEOHAADJKPM; path=/
Cache-control: private

<html>
<head>
	<title>Object moved</title>
</head>
<body><h1>Object Moved</h1>This object may be found <a HREF="localstart.asp">here.
</body>
<html>

Cookie

Praćenje sesije korisnika

HTTP protokol ne prati sesiju - veza klijenta i servera se zatvara po isporuci resursa.

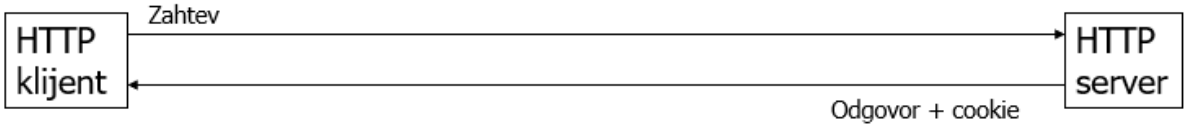
Koristi se **cookie** mehanizam:

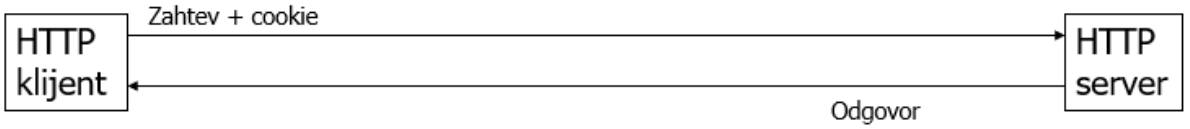
- server šalje cookie klijentu u okviru http response
- klijent čuva primljeni cookie i šalje ga uz svaki http request.

Ako navigator ne prihvata cookie-je, koristi se URL Rewriting mehanizam:

<a
href="http://www.myserver.com/catalog/index.html;jsessionid=1234">
link

- *cookie* mehanizam





Options

OPTIONS – od web servera se traži spisak metoda koje podržava

Osnovna svrha OPTIONS metode je omogućiti klijentu da utvrdi koje HTTP metode, zaglavlja i druge mogućnosti podržava server za određeni resurs. OPTIONS zahtev ne izvršava nikakvu akciju na samom resursu. Umesto toga, on dobavlja metapodatke ili informacije o resursu, kao što su dozvoljene metode i podržana zaglavlja.

JWR = JSON Web Token

Token koji se generiše prvi put na serveru, vraća klijentu i koji klijent šalje u zahtevu svaki put kada se obraća serveru
Sve se radi ručno, za razliku od kukija
Sastoji se iz Base64 enkodiranog JSON stringa:
Header,
Payload,
Signature.

Prilikom prijave na sistem, ako su kredencijali OK, u HTTP odgovoru se šalje upravo izgenerisani JWT token.
Svaki naredni REST zahtev sadrži i JWT token, server proverava validnost tokena i identifikuje korisnika na osnovu tokena.

kako da korisnik ostane ulogovan kada ugasi pa upali racunar

napisati doGet za servlet koji ispisuje prvih 100 neparnih brojeva

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    PrintWriter out = response.getWriter();
    for (int i = 1; i < 200; i += 2) {
        out.println(i);
    }
}
```

razlika izmedju HEAD i GET

GET se koristi za zahtevanje podataka sa servera, dok HEAD zahteva samo informacije o resursu bez samih podataka.
GET se koristi za dobavljanje resursa, dok HEAD koristi iste metode, ali ne prenosi telo odgovora.

Kako se prosleđuju parametri forme GET zahtevom (primer zahteva za 2 parametra a i b sa vrednostima 5 i 6)

Parametri se dodaju URL-u kao deo query stringa. Primer: http://example.com/page?a=5&b=6

vrste veza (relacija)

Četiri vrste:

- OneToOne = atribut koji ukazuje na drugi entity, označava se anotacijom @OneToOne
    - Radnik - Polisa osiguranja
    - Person - Home address
- OneToMany = Kolekcija koja ukazuje na druge entity-je, označava se anotacijom @OneToMany
    - Person – Bank account
    - Ako u entitiju imamo jednu kolekciju, ona se deklariše kao List ili Collection, a implementira kao ArrayList
    - Ako u entitiju imamo više kolekcija, one se deklarišu kao Set, a implementiraju kao HashSet
    - Radno mesto - Radnik
    - Manager - Employees
- ManyToOne = atribut koji ukazuje na drugi entity, označava se anotacijom @ManyToOne
    - Radnik - Radno mesto
    - Bank account - Person
- ManyToMany
    - Nastavnik - Predmet
    - Email list - Subscriber

REST metode

REST se u Spring Boot tehnologiji realizuje POJO Java klasama koje su anotirane @RestController anotacijom.
Svaka metoda u kontroleru se anotira @RequestMapping anotacijom, čime se definiše:
kako se parametri prenose,
kojom metodom iz HTTP zahteva se aktivira,
kako se povratna vrednost mapira u HTTP odgovor.
Alternativno imamo: @GetMapping, @PostMapping, @PutMapping, itd.

Redirekcija

Redirekcija je proces preusmeravanja korisnika sa jedne URL adrese na drugu. Može biti privremena (HTTP status 302) ili trajna (HTTP status 301).

Joinpoint i Pointcut

Joinpoint je tačka u izvršenju koda gde se aspekt može primeniti:

1. metode
2. atributi
3. izuzeci
4. konstruktori
5. statička inicijalizacija (static blok)

Pointcut je skup joinpoint-ova gde se primenjuje aspekt
Poincut definiše na koji/koje od joinpoint-ova će biti primenjen aspekt
Primer: pointcut call() koji obeležava poziv bilo koje metode klase C, bez obzira na parametre:
call(void C.*(..))

hashset i arraylist

- HashSet je implementacija Set interfejsa, ne dozvoljava duplikate.
- ArrayList je implementacija List interfejsa, omogućava dinamičko proširivanje niza.

Koriste se u one to many relacijama:

- Ako u entitiju imamo jednu kolekciju, ona se deklariše kao List ili Collection, a implementira kao ArrayList
- Ako u entitiju imamo više kolekcija, one se deklarišu kao Set, a implementiraju kao HashSet

Referencijalni integritet

Referencijalni integritet održava konzistentnost podataka u bazi tako što osigurava da se ne mogu uneti referencijalno nevalidne vrednosti.
U okviru anotacija stavlja se atribut cascade sa skupom vrednosti:
CascadeType.PERSIST

poziva persist operaciju nad povezanim objektom prilikom perzistiranja ovog objekta

CascadeType.MERGE

poziva merge operaciju nad povezanim objektom prilikom pozivanja merge nad ovim objektom

CascadeType.REMOVE

poziva remove operaciju nad povezanim objektom prilikom pozivanja remove nad ovim objektom

CascadeType.REFRESH

poziva refresh operaciju nad povezanim objektom prilikom pozivanja refresh nad ovim objektom

CascadeType.ALL

sve četiri varijante zajedno

dobavljanje veza sa fetchType.LAZY

FetchType.LAZY

- default za LOB i veze tipa kolekcija (kardinalitet n)
- ne dovlači odmah objekte iz veze
- dovlači ih rekurzivno, ako je potrebno, samo ako smo u istoj sesiji i zatražimo te objekte iz veze
- ako nismo u istoj sesiji, pokušaj pristupa takvom objektima daje izuzetak
- gde je granica sesije?

FetchType.EAGER

- default za primitivne tipove i veze tipa referenca (kardnalitet 1)
- dovlači sve objekte iz veze, rekurzivno, odmah
  - Ovo je potencijalno opasno!
  - manje efikasno

Može se podesiti na nivou upita koji atribut se dovlači odmah, a koji kasnije – fetch join

`FetchType.LAZY` je strategija dovlačenja podataka u JPA (Java Persistence API). Kada se koristi `FetchType.LAZY` za određeno polje koje predstavlja vezu (kolekciju ili LOB - Large Object), JPA će pokušati odložiti dovlačenje tih podataka do trenutka kada su stvarno potrebni.

Granica sesije u ovom kontekstu se odnosi na trenutak kada se sesija s bazom podataka zatvori. Sesija može biti vezana za transakciju, pa granica sesije može biti kraj transakcije. Ako pokušate pristupiti objektima iz odloženih veza nakon

zatvaranja sesije ili transakcije, može se dobiti izuzetak jer JPA više nema aktivnu sesiju i ne može dohvatiti podatke iz baze.

Što se tiče LOB (Large Object) u ovom kontekstu, LOB se odnosi na tip podataka u bazi podataka koji može sadržavati velike količine podataka, kao što su tekstualni dokumenti, slike ili binarni podaci. Kada se koristi `FetchType.LAZY` za LOB, to znači da se ti veliki podaci neće odmah dohvatiti prilikom dohvaćanja entiteta, već će se dohvatiti samo ako se izričito zatraže. Ovo može biti korisno ako želite smanjiti opterećenje baze podataka i ubrzati dohvaćanje osnovnih informacija iz entiteta koji sadrži LOB podatke.

Mapiranje parametara funkcija

Preko putanje (@PathVariable anotacija):

```
@RequestMapping(
        value = "/rest/findByImeAndPrezime/{ime}/{prezime}",
        method = RequestMethod.GET)
public List<Student> getStudents(@PathVariable String ime, @PathVariable String prezime) {
    return srv.getStudentsByImeAndPrezime(ime, prezime);
}
```

GET /rest/findByImeAndPrezime/p/p HTTP/1.1

Preko putanje (@PathVariable anotacija):

```
@GetMapping("/rest/findByImeAndPrezime/{ime}/{prezime}")
public List<Student> getStudents(@PathVariable String ime, @PathVariable String prezime) {
    return srv.getStudentsByImeAndPrezime(ime, prezime);
}
```

GET /rest/findByImeAndPrezime/p/p HTTP/1.1

Preko parametara forme u URL-u (@RequestParam anotacija):

```
@RequestMapping(
        value = "/rest/findByDatumRodjenjaBetween",
        method = RequestMethod.GET)
public List<Student> findByDatumRodjenjaBetween(
        @RequestParam("begin") @DateTimeFormat(pattern="yyyy-MM-dd") Date begin,
        @RequestParam("end") @DateTimeFormat(pattern="yyyy-MM-dd") Date end) {
    return findByDatumRodjenjaBetween(begin, end);
}
```

GET /rest/findByDatumRodjenjaBetween?begin=2000-01-01&end=2002-01-01 HTTP/1.1

Kao JSON string u telu POST zahteva

```
@RequestMapping(
        value = "/rest/updateStudent",
        method = RequestMethod.POST)
public Student updateStudent(@RequestBody Student s) {
    s = srv.save(s);
    return s;
}
```

Advice

Savet – kod koji se ubacuje u postojeću klasu/metodu
Definiše ponašanje aspekta

- kada će se aktivirati i šta će raditi
- tu se zapravo piše programski kod aspekta

Tri vrste:

- before – advice code se izvrši pre joinpoint-a
- after – advice code se izvrši posle joinpoint-a
    - ako se ne stavi after() returning, onda se svakako izvršava, bez obzira da li je bio izuzetak ili ne
- around – advice code se izvrši pre i posle joinpoint-a; postoji ključna reč proceed kojom se nastavlja izvršenje normalnog koda

Vrste saveta

1. before()
2. after() returning
3. after() throwing
4. after()
5. around()

- after() returning [(type expr)]
    - posle "normalnog" izvršenja
- after() throwing [(exception expr)]
    - ako dođe do izuzetka navedene vrste

- after()
    - poziva se uvek (nešto kao finally)

Refleksija Class

Mehanizam za ispitivanje osobina proizvoljne klase (pregled atributa, metoda i konstruktora) u toku izvršenja programa (run-time).
Paket java.lang.reflect

Odakle krenuti?

Klasa Class (iz paketa java.lang)!

- reprezentuje klase i interfejse u Java aplikaciji (run-time)

Kako doći do Class klase?

- metodom getClass svakog objekta,
- statičkom metodom forName klase Class i
- statičkim atributom class svake klase.

Šta možemo da uradimo sa Class klasom?

- metoda *getName* daje pun naziv klase
- metoda *newInstance* poziva default konstruktor
- metoda *getDeclaredFields/getFields* daje spisak atributa
- metoda *getDeclaredMethods/getMethods* daje spisak metoda
- metoda *getDeclaredConstructors/getConstructors* daje spisak konstruktora