# Teorija iz web programiranja

**Napomena:**

Ova teorija nikako **nije dovoljna** za polaganje ispita, ona je samo enkapsulacija zanimljivih delova sa predavanja na jednom mestu kako ne bih stalno morao posećivati prezentacije.

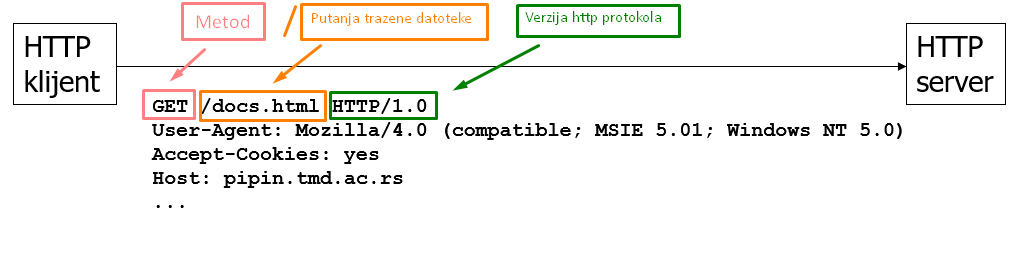
Deo koji je kompetentniji za polaganje je u sekciji **pitanja sa ispita** u kome se nalazi kulminacija pitanja koja su se pojavljivala na **prethodnim ispitima**.

## HTTP Protokol

Kada god naš browser zatraži neki sadržaj od web servera, on to radi tako što uputi **zahtev** i dobije **odgovor**. A taj zahtev i odgovor su formirani po **HTTP protokolu**.

Komunikacija je zasnovana na **zahtev/odgovor** principu koji su *nezavisni* od ostalih parova zahtev/odgovor.

##### Http zahtev



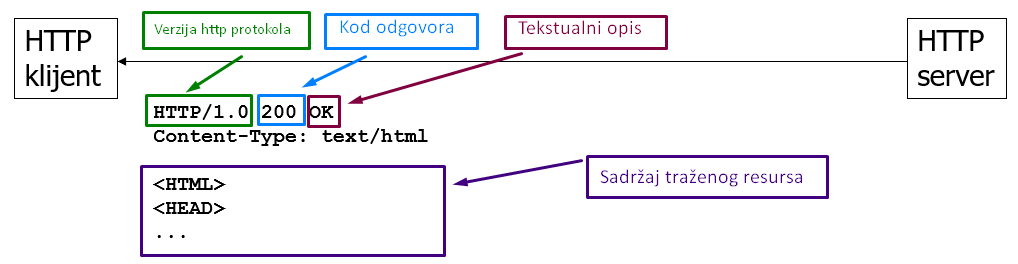
**Slika 1.** Primer http zahteva

**Prvi red** u zahtevu je **obavezan** i formata je **metod** /**putanja\_trazene\_datoteke** **verzija\_http** kao što vidimo na slici 1. Posle prvog reda idu opcioni atributi oblika **ime\_atributa**: **vrednost.** Posle tih opcionih atributa ide prazan red koji označava kraj http zahteva.

##### Http odgovor

Format odgovora u **prvom redu** je **obavezan** i oblika: **verzija\_http** **code** **txt\_opis** a u sledećim redovima idu opcioni atributi istog formata kao i opcioni atributi u http zahtevu (slika 1). Posle opcionih atributa ide prazan red, nakon kog ide **sadržaj traženog resursa**. Taj sadržaj je onakav kakva je ta datoteka, ako je tekstualni, ide txt ako je binarni idu binarni podaci.

**Kodovi** su podeljeni u opsege, **100** – unapredjenje protokola , **200** – uspešnost , **300** – redirekcija, **400** – greška na *frontu*, **500** – greška na *beku*



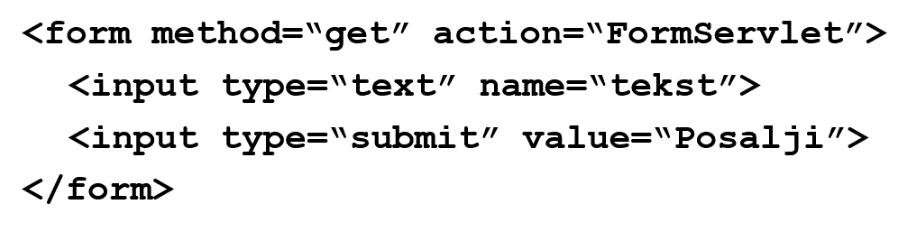
**Slika 2**. Primer http odgovora

**Sadržaj** traženog resursa može biti **statički**(unapred uskladišten) i **dinamički**(generisan po zahtevu).

##### Preuzimanje podataka sa formi

Na slici 3 vidimo primer kako mi preko forme preuzimamo podatke. Te podatke možemo poslati na 2 načina, preko metode **get** i metode **post**. Akcija ( ***action*** parametar forme) predstavlja url koda koji će pokupiti parametre forme i obraditi ih.

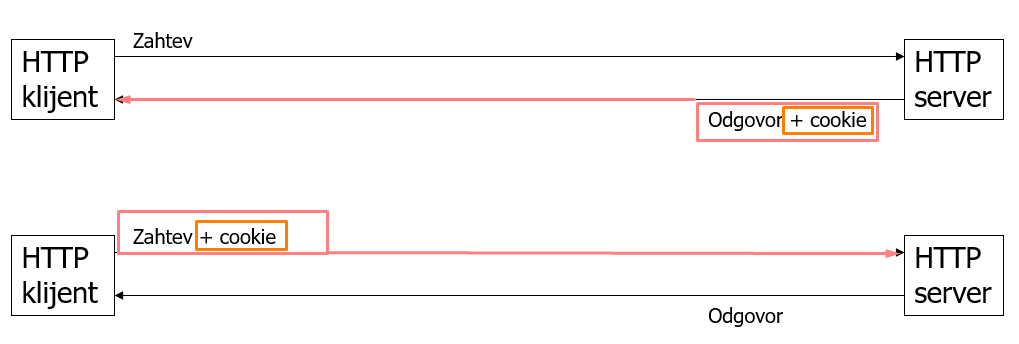
U slučaju metode **GET**, parametri forme se nalaze u ***adresnoj liniji*** i svima su vidljivi što nije bezbedan način. U slučaju metode **POST**, parametri forme se nalaze u ***telu zahteva***, što definitivno predstavlja bezbedniji način za slanje parametara.

**Slika 3**. Primer forme preko koje šaljemo podatke

##### Praćenje sesije korisnika

Http protocol ne prati sesiju, pošto se veza klijenta i servera zatvara po isporuci resursa. Alternativa koja rešava taj problem je ***cookie******mehanizam***.

**Cookie mehanizam** funkcioniše tako što *server* **šalje** ***cookie*** klijentu u okviru http **odgovora**, *klijent* **čuva** primljeni ***cookie*** i **šalje** ga uz svaki http **zahtev**.

**Slika 4**. Primer cookie mehanizma

## Servleti

Tehnologija za generisanje **dinamičkih sadržaja** koja kao rezultat izvršenja ima **dinamički kreiran sadržaj**.

Naziv resursa iz HTTP zaglavlja se koristi za pretragu postojećih servleta.

**Napomena:**

Dosta pitanja iz ove oblasti je vezano za **metode** i **klase** koje se koriste u **JAVI** za implementaciju servleta. A da bi teorija bila čistija, te stvari će biti obrađene dole u **pitanjima sa ispita** koja su vezana za **servlete**.

## JSP, JSTL, EL

##### Java Server Pages (JSP)

Ideja je u tome da imamo **HTM**L + **dinamičke elemente**. Vrste dinamičkih elemenata:

* **Izrazi** (expressions): **<%= java\_izraz %> ( <%= new Date() %> )**
* **Skripleti** (scriptlets): **<% java\_kod %> ( <% for(int i = 0; i< 10; i++) … %> )**
* **Deklaracije** (declarations): **<%! java\_deklaracija %> ( <%! int a; %> )**
* **Directive** (directives): **<%@ direktiva attr=”…” %> ( <%@ page contentType=”text/plain” %> )**

##### Expression Language (EL)

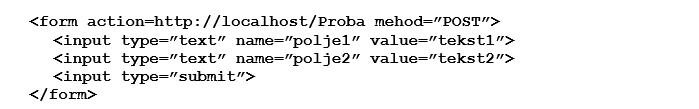
Uveden kako bi skratili dugačke zapise, opšti oblik : **${ izraz }.** Jednostavniji pristup kolekcijama i nizovima, automatska konverzija tipova.

##### JSP Standard Tag Library (JSTL)

Predstavlja specifikaciju **dodatnih tagova**, tagovi za ***iteraciju*, *grananja*, *pristup*** *bazi* podataka itd.

### Pitanja sa ispita

#### HTTP Protokol

**1. Napisati prvi i poslednji red HTTP zahteva (HTTP verzija 1.1) koji se dobija kada se klikne na submit forme (a ništa se ne unese u polja) u sledećoj formi:**

###### Odgovor:

Prvi: **POST** **/http://localhost/Proba** **HTTP/1.1**

Poslednji: polje1=tekst1**&**polje2=tekst2

###### Napomena:

Voditi računa da je u pitanju bio POST zahtev pa su zbog toga parametri forme zapravo činioci poslednjeg reda, što nije u slučaju GET zahteva.

U **POST** metodi postoji atribut **Content length: dužina** ( atribut koji nam govori koliko još **bajtova** treba da **učitamo** **nakon praznog reda**), dok kod **GET** metode kraj označava **prazan red**.

**2. Šta je to “Permanent connection” u HTTP protokolu verzije 1.1?**

Klijent od servera zahteva se konekcija ne zatvara odmah po slanju odgovora. U zaglavlju http zahteva i odgovora se stavlja atribut **Connection = Keep-Alive**

**3. Navesti nazive atributa u HTTP zahtevu i HTTP odgovoru, a koji omogućava praćenje sesije u HTTP protokolu.**

###### Odgovor:

Pošto ne postoji način za praćenje sesije, uvodi se mehanizam zvani ***cookie* mehanizam**. Kada se prvi put šalje zahtev na server, nemamo atribut vezan za praćenje sesije. Nakon prvog zahteva, u odgovoru servera se nalazi atribut **Set-Cookie: ime=vrednost** . Pošto je klijent sada dobio svoj cookie, u sledećem zahtevu on ima atribut **cookie: ime=vrednost** koji ga jedinstveno identifikuje. Na taj način je omogućeno praćenje sesije u http protokolu.

**4. Čime se označava kraj HTTP zahteva u GET metodi?**

###### Odgovor:

Praznim redom.

###### Napomena:

Ovo **ne važi** i za **POST** metod, za njega je kraj zahteva određen atributom **Content length**( atribut koji nam govori koliko još **bajtova** treba da **učitamo** **nakon praznog reda**) !

**5. Navesti osnovnu razliku između GET i POST zahteva.**

###### Odgovor:

U slučaju GET metode, podaci forme se nalaze u adresnoj liniji, dok se kod POST metode podaci forme nalaze u telu zahteva.

**GET /FormServlet?tekst=asdf HTTP/1.1**

**...**

**POST /FormServlet HTTP/1.1**

**...**

**Content-length: 10**

**...**

**tekst=asdf**

**6. Navesti ključne elemente HTTP odgovora koji se koriste kod redirekcije. Dati primer takvog HTTp odgovora.**

###### Odgovor:

Redirekcija se svodi na slanje poruke: **302 Object moved** i postavljanja atributa **location: nova\_adresa** u http odgovoru.

Primer:

**HTTP/1.1 302 Object moved**

**Location: neki\_novi\_url**

**7. Ukratko napiši sadržaj HTTP zahteva.**

###### Odgovor:

**METOD /putanja HTTP/verzija**

Dodatni redovi koji sadrže attribute u obliku: **ime\_atributa: vrednost**

**Prazan** **red** na kraju

**8. Ukratko napiši sadržaj HTTP odgovora.**

###### Odgovor:

**HTTP/verzija** **code** **txt\_opis**

Dodatni redovi koji sadrže attribute u obliku: **ime\_atributa: vrednost**

**Prazan** **red**

**Sadržaj**

**9. Napisati prvi red HTTP zahteva i odgovora.**

###### Odgovor:

Zahtev: **METOD /putanja HTTP/verzija**

Odgovor: **HTTP/verzija** **code** **txt\_opis**

**10. Napisati prvi red HTTP odgovora u HTTP protokolu verzije 1.0 za kod 404.**

###### Odgovor:

**HTTP/1.0** **404** **Not found**

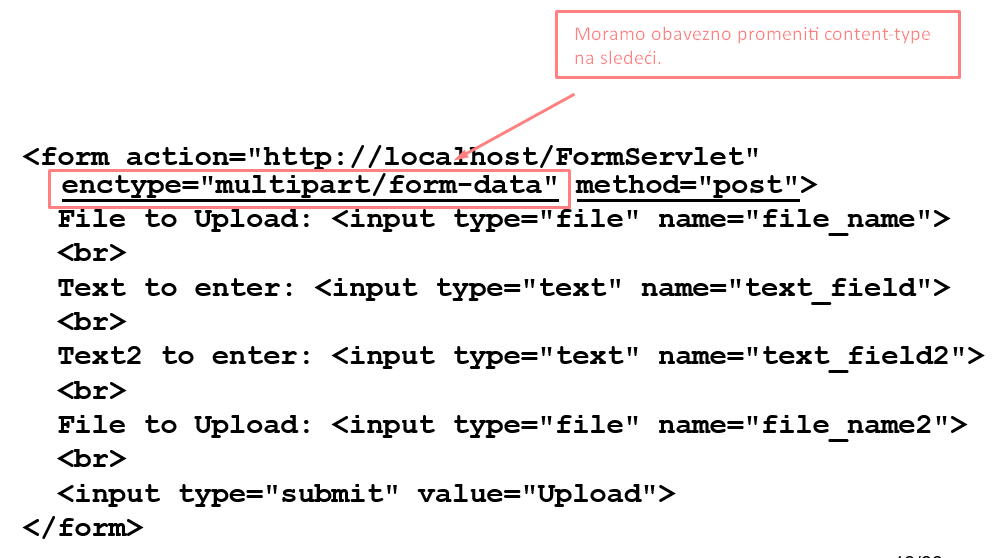
**11. Navesti metode HTTP zahteva i šta radi koji.**

###### Odgovor:

**GET** – **zahteva** resurs od web servera, **POST** - **šalje** parametre forme i traži odgovor, **HEAD** – **zahteva** samo HTTP odgovor, bez slanja samog resursa, **PUT** – omogućava klijentu da **pošalje** **datoteku** na web server, **OPTIONS** – od web servera se **traži** spisak **metoda** koje podržava, **DELETE** – omogućava klijentu da **obriše** resurs sa web servera.

**12. Kako izgleda HTTP zahtev u kom se prosleđuje fajl?**

###### Uvod:

****Običan POST zahtev ima atribut **Content-Type: application/x-www-form-urlencoded** ali kada je u pitanju POST zahtev koji prosleđuje fajl priča se komplikuje.

**Slika 1.** Primer HTML koda na klijentu pri slanju datoteke

Prvo u formi na frontu moramo imati podešen **enctype** na **multipart/form-data.** Tek kada smo enctype podesili, onda u **input** atributima možemo koristiti **type=”file”.** Zbog ovoga imamo totalno drugačiji donji deo HTTP zahteva ( onaj posle praznog reda, tj tamo gde dolaze parametri forme ).

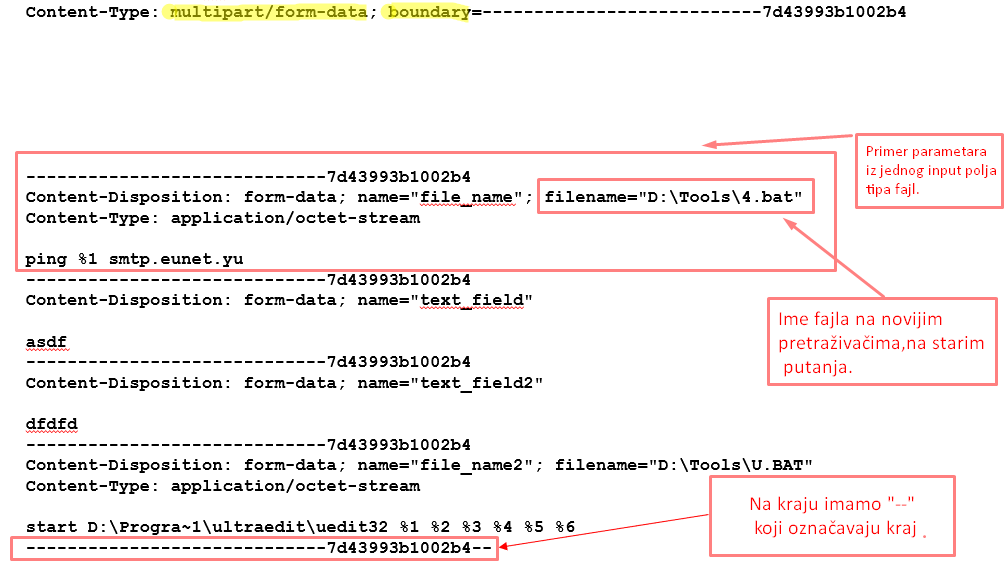
###### Odgovor:

U odnosu na klasični POST zahtev **Content-Type** atribut dobija i nastavak zvani **boundary** koji služi za definisanje granice između pojedinih elemenata forme.

Primer:

**Content-Type: multipart/form-data; boundary=---------------------------7d43993b1002b4**

Sada se menja i format parametara posle praznog reda. Na slici 2 vidimo da kada se prosleđuje fajl, imamo “***novo zaglavlje***” koje ima attribute **Content-Disposition** koji dobija još dodatni deo **filename** kada je input polje bilo tipa file. A takođe ima i **Content-Type** atribut u koliko je input polje bilo tipa fajl, to polje nam govori tačan tip fajla. Nakon toga ide prazan red i bajtovi fajla.

**Slika 2.** Primer HTTP zahteva u kom se prosleđuje fajl

**13. Navesti attribute i njihovu namenu u HTTP odgovoru.**

###### Odgovor:

* **Accept-Charset** – definiše koju kodnu stranu očekuje
* **Cookie**- definiše mehanizam praćenja sesije
* **Referer** – definiše URL sa kojeg se došlo na ovu stranicu
* **Connection** – podešavamo da li da se (ne)zatvori konekcija po isporuci resursa

**14. Navesti attribute i njihovu namenu u HTTP zahtevu.**

###### Odgovor:

* **Content-Type –** definiše tip odgovora
* **Cache-Control –** definiše kako se keš na klijentu ažurira
* **Location –** definiše novu adresu kod redirekcije
* **Connection –** potvrda klijentu da li da zatvori konekciju ili da je ostavi otvorenu

**15. Kako se ostvaruje trajna konekcija (permanent connection) u HTTP protokolu verzije 1.1 ?**

###### Odgovor:

Postavljanjem atributa ***Connection*** HTTP odgovora na ***Keep-Alive.***

**16. Koji atribut form taga omogućuje da forma služi za upload i slanje datoteka? Navesti ime I vrednost.**

###### Odgovor:

**enctype=”multipart/form-data” .**

**17. U kom atributu HTTP zahteva se podešava gornja granica (boundary) veličine fajla prilikom upload-a/slanja?**

###### Odgovor:

Content-Type .

**18. Koji je minimalan broj atributa u HTTP zahtevu ?**

###### Odgovor:

Tri jer zahtev obavezno mora da ima samo **prvi red**, ostali atributi nisu obavezni.

**19. Koji atributi HTTP zahteva i HTTP odgovora omogućavaju praćenje sesije ?**

###### Odgovor:

U HTTP zahtevu, atribut koji omogućava praćenje sesije, je atribut ***Cookie***. On čuva primljeni cookie i šalje ga uz svaki *zahtev*. U HTTP *odgovoru*, atribut koji omogućava praćenje sesije je atribut ***Set-Cookie***.

Ostali opcioni atributi u HTTP *odgovoru* su:

* **Domain** – domen u kome važi cookie
* **Path** – za koje URL-ove na sajtu važi cookie
* **Expires** – datum isticanja cookie-a

#### Servleti i praćenje sesije

**20. Navesti klasu koja se nasleđuje prilikom pravljenja servleta i navesti metode (sa parametrima) koje se redefinišu za POST i GET metode.**

###### Odgovor:

Klasa: **HttpServlet**

Metoda za **POST** metod:

*protected**void* **doPost(HttpServletRequest request, HttpServletResponse response)**

Metoda za **GET** metod:

*protected**void* **doGet(HttpServletRequest request, HttpServletResponse response)**

**21. Koliko objekata neke servletske klase se kreira tokom rada?**

###### Odgovor:

Jedan objekat.

**22. Kojom metodom, koje klase se kod servleta obezbeđuje redirekcija po HTTP protokolu?**

###### Odgovor:

Klasa: **HttpServletResponse**

Metoda: **sendRedirect(nova\_lokacija)** //kod u javi: sendRedirect(String location);

Atribut: **location**

**23. Čime se podešava kodna strana rezultujućeg teksta koji se prikazuje iz servleta? Dati primer za UTF-8.**

###### Odgovor:

Metodom **setContentType** se podešava character encoding. Parametar (atribut) **charset** definiše kodnu stranu kojom će biti kodirani svi stringovi na klijentu.

Primer: **response.setContentType(“text/html; charset=UTF-8”)**

**24. Kojom klasom je implementirano praćenje sesije u servletima? Koje metode ove klase se koriste?**

###### Odgovor:

Klasa: **HttpSesion**

Metode:

* **setAttribute(ime, obj), removeAttribute(ime), getAttribute(ime) -** čuva **objekte** vezane za sesiju
* **getId() –** čuva **cookie** ili ID sesije za URL redirection
* **invalidate() – invalidira** sesiju i razvezuje sve objekte vezane za nju
* **setMaxInactiveInterval(sekunde) –** podešava period neaktivnosti

**25. Navesti dva tipa kolačića i za šta se koriste ?**

###### Odgovor:

Postoje dve osnovne vrste cookie-a: ***session* cookie** i ***permanent* cookie**.

**Session cookies** se čuvaju u **privremenoj memoriji** **browsera** sve dok se ne raskine veza sa serverom. Oni obično **čuvaju ID sesije** koji ne identifikuje korisnika, omogućavajući im da se kreću od stranice do stranice bez potrebe da se stalno loguju.

**Permanent cookies** se čuvaju u **trajnoj memoriji browsera.** Brišu se tek kad isteknu(server podešava vreme važenja cookie-a). Obično sadrže podatke o korisniku određenog sajta. Omogućavaju da se ti podaci koriste u budućim sesijama.

**26. Kako se uništavaju svi objekti vezani za sesiju?**

###### Odgovor:

Pozivom metode: **invalidate()** . Ona je jedna od metoda klase HttpSession, za više metoda te klase, pogledati pitanje **24.**

**27. Kako obezbediti da browser zapamti korisnika i prilikom gašenja browser-a i ponovnog pokretanja?**

###### Odgovor:

Postavljanjem atributa **expires**, prilikom davanja kolačića.

**28. Kako se vrši praćenje sesije, ako navigator ne prihvata cookie-e?**

###### Odgovor:

U tom slučaju se koristi ***URL Rewriting*** mehanizam. U hyperlink (<a href=”…”>) koji “gađa” naš server **ugradi se ID sesije**: <a href=http://www.pero.com/index.html; **jsessionid=1234**>

**29. Koju klasu nasleđuju svi servleti i koja metoda te klase se koristi za inicijalizaciju/”uništavanje”.**

###### Odgovor:

Klasa: **HttpServlet**

Metoda za **inicijalizaciju**: **init()**

Metoda za “**uništavanje**”: **destroy()**

**30. Koja metoda klase HttpServlet se redefiniše za obradu GET zahteva i koje parametre ona ima?**

###### Odgovor:

Metoda: **doGet(HttpServletRequest request, HttpServletResponse response)**.

Ovde može pitati i za druge zahteve (**POST, PUT, HEAD**…), a **parametri** su uvek **isti**, jedino se razlikuje naziv metode **doGet, doPost, doPut**...

**31. Koje metode sadrži klasa HttpServletRequest/HttpServletResponse i čemu one služe?**

###### Odgovor:

Metode HttpServletRequest:

* **getParameter(ime), getParamerNames(), getParameterMap() –** služe za izdvajanje parametara GET/POST metode iz forme, smeštanje istih u asocijativnu listu
* **getHeader(ime), getHeaderNames(), getHeaders(ime) –** služe za izdvajanje parametara iz zaglavlja HTTP zahteva i smeštanje istih u asocijativnu listu

Metode HttpServletResponse:

* **setContentType(vrednost) –** čuva tip odgovora
* **addCookie(cookie) –** čuva cookie
* **sendRedirect(nova\_lokacija) –** omogućava redirekciju
* **setHeader(naziv, vrednost) –** podešava proizvoljan atribut zaglavlja
* **encodeURL(url) i encodeRedirectURL(url) –** ugrađuje ID sesije ako cookies nisu uključeni

**32. Napisati attribute HTTP Odgovora/Zahteva koji omogućavaju praćenje sesije u HTTP protokolu.**

###### Odgovor:

**Cookie**, **Set-Cookie** a opcioni su ***domain***, ***path*** i ***expires***.

**33. Napisati kod koji u servletu zakači promenljivu koja se zove “pera” i njegova je vrednost “abc”, na zahtev u POST metodi.**

###### Odgovor:

protected void **doPost**(HttpServletRequest request, HttpServletResponse response) {

String **pera** = **“abc”;**

Request.**setAttribute**(**“pera”,** **pera**);

…

}

**34. Napisati redirekciju na stranicu pera.jsp po HTTP protokolu.**

###### Odgovor:

response.**sendRedirect(“pera.jsp”)**

**35. Ako 100 klijenata gađa servlet, koliko objekata te klase će se kreirati ?**

###### Odgovor:

Jedan**.**

**36. Napisati programski kod u GET metodi u servletu, takav da ispiše na ekran brojeve od 1 do 100.**

###### Odgovor:

protected void **doPost(HttpServletRequest request, HttpServletResponse response)** {

response.setContentType(“text/html; charset=UTF-8”);

PrintWriter **pout** = response.**getWriter**();

// ispis tagove html, head, body-a

for(int i = 1; I <=100; i++){

**pout.println(i);**

}

// zatvoriti tagove html i body-a

**pout.close();**

}

**37. Da li servlet podržava rad sa upload-ovanim fajlovima ?**

###### Odgovor:

Da, od verzije 3.0.

**38. Redirekcija u MVC modelu 2.**

###### Odgovor:

Za redirekciju se koristi klasa **RequestDispatcher**(kontrola toka ).

Klasa ima dve metode:

* **forward** – koristi se kada **kompletno prebacujemo** kontrolu na **odredišnu** stranicu
* **include** – koristi se kada **ubacujemo** **sadržaj** **odredišne** stranice i nastavljamo sa tekućom

**RequestDispatcher** omogućuje da na odredištu vidimo request i response objekte iz polazne stranice(servleta).

RequestDispatcher disp = *request*.**getRequestDispatcher**(***relativanURL***);

RequestDispatcher disp = *getServletContext*().**getRequestDispatcher**(***apsolutniURL***);

#### JSP, EL, JSTL

**39. Navesti vrste dinamičkih elemenata u JSP.**

###### Odgovor:

Vrste dinamičkih elemenata:

* **Izrazi** (expressions): **<%= java\_izraz %> ( <%= new Date() %> )**
* **Skripleti** (scriptlets): **<% java\_kod %> ( <% for(int i = 0; i< 10; i++) … %> )**
* **Deklaracije** (declarations): **<%! java\_deklaracija %> ( <%! int a; %> )**
* **Directive** (directives): **<%@ direktiva attr=”…” %> ( <%@ page contentType=”text/plain” %> )**

**40. Navesti i opisati opsege vidljivosti komponenti, te dati primer za jednu od njih.**

###### Odgovor:

* **application** – *istu* instance beana **dele** **svi** **korisnici** sajta
* **session** – svaki **korisnik** sajta ima *svoju* instance
* **request** – svaki **zahtev** za stranicom ima *svoju* instance
* **page** (**default**) – svaka **stranica** ima *svoju* instance

**<jsp:useBean id=”user” class”somepackage.User” scope=”session”/>**

**41. Gde se podešava opseg vidljivosti komponenti ?**

###### Odgovor:

U **<jsp:useBean>** element u tagu **scope**.

**42. Napisati deo koda, koji služi za prenos parametara iz servleta u JSP stranicu u zahtevu/sesiji/aplikaciji.**

###### Odgovor:

Servlet strana:

BeanClass value = new BeanClass(...);

Onda u zavisnosti da li je **request**/**session**/**application** opseg koji podešavamo:

* **request**: request.setAttribute(“bean”, value);
* **session**: request.**getSession**(true).setAttribute(“bean”, value);
* **application**: **getServletContext**().setAttribute(“bean”, value);

JSP strana:

<jsp:**useBean** id=”bean” type=”BeanClass” **scope**=”**request**/**session**/**application**”>

<jsp:**getProperty** name=”bean” **property**=”someProperty” />

###### Napomena:

Voditi računa da na **Servlet** strani atribut setujemo nad opsegom koji podešavamo, a na **JSP** strani da atribut **scope** podesimo na opseg koji smo prethodno odredili (**request**/**session**/**application** ).

**43. Napisati kako se preuzimaju parametri u servletu.**

###### Odgovor:

* **Tekući zahtev** – request.getAttribute(“bla”);
* **Tekući zahtev** (**request**-scoped JavaBean) – request.getAttribute(“bla”);
* **Tekuća sesija** ( **session**-scoped JavaBean) – request.**getSession**().getAttribute(“bla”);
* **Globalni nivo** (**application**-scoped JavaBean) – **getServletContext**().getAttribute(“bla”);

**44. U servletu je kreiran niz messages na sledeći način:**

String [] massages = new String [] {”ok”,”error”, ”warning”}

I dodat u session scope. Dat je deo JSP stranice na koju se servlet redirektuje

<c:forEach var=”message” items=”${messages}”>

<c:out value=”${message}”/>

</c:forEach>

**Šta će biti ispisano nakon poziva JSP stranice u browseru ?**

###### Odgovor:

Biće ispisano: **ok error warning**

**45. Navesti neke od problema koji se javljaju kod servleta, a koji jsp popravlja.**

###### Odgovor:

* Dizajn stranica i programska obrada su pomešani u istim datotekama
* Teško razdvojiti funkciju dizajnera i programera ( nedostatak disjunkcije fronta i beka)
* Svaka promena izgleda stranice zahteva kompajliranje servleta

**46. Šta se dogodi kada klijent zatraži stranicu pera.jsp?**

###### Odgovor:

Kod **prvog** poziva JSP stranice, **JSP** **kod** se pretvara u **servlet**, koji generiše odgovarajući kod. Taj kod se **kompajlira** i klijentu se šalje odgovor. Prilikom sledećih poziva iste JSP stranice, **ne** radi se **ponovo** pretvaranje u kod i kompajliranje.

**47. Korištenjem JSP/JSTL/EL ispisati promenljivu pera zakačenu na zahtev.**

###### Odgovor:

* **JSP: <%=** pera **%>**
* **EL: ${** pera **}**
* **JSTL: <c:out value =” ${** pera **} “ />**

**48. Objasniti MVC i MVC 2 šablon. ( legenda kaže da nije ispit ako se ovo pitanje ne spomene bar jednom ).**

###### Odgovor:

**MVC**

Sastoji se iz povezanih JSP stranica. **Prezentacija** i **kontrola** su **ugrađeni** u **JSP** stranice. **Podaci** su modelirani upotrebom Java **beans**. Prelazak na sledeću stranicu je određen klikom na hyperlink i pritiskom na dugme submit.

Browser šalje request direktno JSP stranici, ona komunicira sa bean-ovima i vraća odgovor.

**MVC 2**

Razdvaja model, prezentaciju i kontrolu. Bolje upravlja tokom prezentacije. Omogućuje lakši prikaz različitih JSP stranica u zavisnosti od ulaznih podataka. **Servleti** obavljaju posao **kontrolera**. Upravljaju stanjem web applikacije a **stranicu** za prikaz određuju **redirekcijom**. **JSP** stranice su **prezentacioni** sloj dok Java **beans** modeliraju **podatke**.

Browser se obraća servletu, koji zatim komunicira sa bean-ovima, zatim JSP stranice koriste bean-ove da bi formirale konačni response.

**49. Razlika između metode *sendRedirect* i metode *forward* klase RequestDispatcher?**

###### Odgovor:

Upotrebom **sendRedirect**-a kreira se **novi** **objekat** request, a upotrebom metode **forward** se prosleđuje **postojeći**, zajedno sa svim atributima zakačenim na njega. Kod **sendRedirect** klijent je **informisan** da je urađena **redirekcija** ( promeni se url adresa u browser-u), a kod **forward** klijent **nije** **svestan** redirekcije.

Korišćenjem **sendRedirect**, omogućena je **redirekcija** **bilo gde**, a korišćenjem **forward** samo **u okviru istog servera** !

**50. Navesti ekvivalenti kod u JAVI za <jsp:useBean id=”pera” class=”beans.User” scope=”sesion”>.**

###### Odgovor:

User peraUser = request.getSession().getAttribute(“pera”);

If(peraUser == null){

peraUser = new User();

request.getSession().setAttribute(“pera”, peraUser);

}

**51. Napisati doGet metodu klase HttpServlet koja će ispisati 100 puta slovo A.**

###### Odgovor:

protected void doGet(HttpServletRequest request, HttpServletResponse response){

response.setContentType(“text/html; charset=UTF-8”);

**PrintWriter** **pout** = response.**getWriter**();

// ispis tagova html, head i body-a

**for(int i = 0; i < 100; i++){**

**pout.println(“A”);**

**}**

// zatvaranje tagova

pout.**close**();

}

**52. Napisati JSP kod koji će 100 puta ispisati slovo A.**

###### Odgovor:

**<%** for(int i = 1; i <= 100; i++) **%>**

**<%=** A **%>**

**53. Kako se podacima forme pristupa uz pomoć EL tehnologije ?**

###### Odgovor:

**${** model.atribut **}**

#### JavaScript

**54. Napisati JS kod koji će 100 puta ispisati slovo A.**

###### Odgovor:

for(let i = 0; i < 100; i++)

console.log(“A”);

**55. U JS-u napraviti jedan objekat sa atributom id i vrednošću tog atributa "pera".**

###### Odgovor:

let objectSample = { id: “pera” };

**56. Napisati JSON za objekat klase Student koji ima id=10, ime="pera", prezime="peric".**

###### Odgovor:

{ “id”: 10, “ime”: “pera”, “prezime”: “peric” }

###### Napomena:

Za razliku od JS objekta, **JSON** kod ***ključa*** ima **navodnike**, dok **JS** **objekat** **nema** !

**57. Navesti barem 2 oblika u kome se podaci mogu prenositi AJAX pozivom.**

###### Odgovor:

JSON, XML, text HTML .

**58. Navesti barem jedan način da se REST metodi prenesu parametri.**

###### Odgovor:

PathParam, QueryParam …

**59. Napisati anotacije i deklaraciju metode “izračunaj” koja kao parameter putanje prima “int visina” a vraća String.**

###### Odgovor:

@GET

@Path(“neka\_putanja/**{ visina }**”)

@Produces(MediaType.TEXT\_PLAIN)

Public String izracunaj(@**PathParam**(“**visina**”) int visina){ … }

**Čuvajte se i srećno, Vladislav.**