KI204 – TESTOVI

TEST 1

1. Šta je Servlet?

- **Servlet** je server-side komponenta koja se izvršava isključivo unutar Java virtualne mašine. On je zapravo Java klasa napisana na osnovu određenih pravila i biće isporučena u Java EE kontejner koji se odabere.

2. Koje su prednosti primene Servleta?

- **Prednost** je u tome što servleti obezbeđuju sofisticirani način kreiranja serverske strane prateći standardno J2EE okuženje i koristeći programski jezik Java.

3. Za šta se obično koriste HTTP servleti?

- Obično se koriste da:
  = Obezbede dinamički sadržaj kao što je uzimanje rezultata upita i vraćanje istih do klijenta.
  = Obrada i čuvanje podataka koji se nalaze na HTML strani,
  = Upravljanje informacijama koje se odnose na stanje HTTP-a.

4. Objasnite tokove podataka između klijenata i severa?

- Od web browsera se šalje HTTP zahtev ka web serveru, a od njega se šalje HTTP servlet zahtev ka Servletskom kontejneru u kome se nalazi kreirani Servlet. Ovaj kontejner se snabdeva podacima iz baze podataka i na primljeni HTTP Servlet zahtev, kontejner/servlet šalje HTTPServlet odgovor ka web serveru, a od njega se šalje HTTP odgovor ka web browseru.

5. Navedite pravila koja bi trebala da zadovolji servlet aplikacija?

- **Pravila** su:
  = Klijent-računar koji se obraća servletu mora da ima instaliran web brovser.
  = Računar koji hostuje stranicu kreiranu pomoću servleta mora da pokrene web server, obično na portu 8080.
  = Web server ima zadatak da osluškuje zahteve korisnika.
  = Stranica koja je kreirana pomoću servleta, izvršava kontejner servelta zajedno sa isporučenim servletom.
  = Servlet angažuje Java kod čiji je zadatak realizovanje pretrage.
  = Web server izoluje sadržaj iz HttpServletResponse objekta, pakuje ga u HttpResponse objekat i šalje u izabrani browser.
  = Web browser prikazuje korisniku primljenu stranicu.

6. Kako se kreira servlet?

- Servlet predstavlja Java klasu koja nalseđuje osnovnu klasu HttpServlet i koja je obeležena anotacijom @WebServlet.
  Koraci u **kreiranju** su:
  = desni klik na **Source Packages → New/Servlet →** popunim podatke → klik na **Finish**

7. Objasnite životni ciklus servleta?

- Životni ciklus servleta, obuhvata sledeće faze:
  = Učitavanje – serveltski kontejner učitava servelt za vreme pokretanja (kada se pošalje 1 zahtev)
  = Inicijalizacija – servletski kontejner pokreće metodu init() i prosleđuje inicijalne parametre servleta istoj.
  = Obrada zahteva – pozivaju se metode service(), doGet(), doPost().
  = Uništavanje servelta – poziva se destroy() metoda, ako servlet duže vreme nije potreban.

8. Objasnite HTTP Get i Post zahteve u radu sa servletima?

- Get i Post su najčešće korišćene metode za razmenu podataka na web-u.
  = **Get** = Ova metoda kopira ili označava URL adresu koja sadrži parametre. Get metoda se koristi za čitanje podataka.
  = **Post** = ovom metodom je moguće slanje binarnih (npr: slike i muzički fajlovi) i tekstualnih podataka do servera. Post metoda se koristi za slanje podataka ka serveru.

9. Šta je sesija?

- **Sesija** je logički zadatak, koji korisnik pokušava da izvrši pristupanjem web prezentaciji.

10. Objasnite najpoznatije načine praćenja sesija?

- Informacije o sesiji mogu biti čuvane na dva načina:
  = **Čuvanje na strani klijenta** – gde se korisnički podaci o sesiji beleže smeštanjem u „kolačićima“ ili modifikovanjem URL adrese.
  = **Čuvanje na serverskoj strani** – se vrši primenom API-ja za praćenje sesija.

11. Kako objekat HttpSession čuva podatke o sesiji?

- U ovom slučaju, podaci o sesijama se isključivo čuvaju na serveru a klijent dobija samo ID broj sesije, da bi bila identifikovana serija HTTP zahteva koje šalje isti korisnik.

12. Šta je GlassFish server i koja mu je namena?

- **GlassFish** server je opn-source server baziran na JavaEE implementaciji. Razvijen je kao specifična podrška razvoju Java web aplikacija.

13. Navesti neke prednosti GlassFish servera?

- **Prednosti** su: besplatan je, ima veliku brzinu rada, može da upravlja velikim brojem instanci, jednostavan je za upotrebu, … i slično.

14. Servelt rukuje zahtevima i odgovorima primenom klasa? ***

- **HttpServeltRequest** i **HttpServletResponse**.

15. Primenom GlassFish servera proces razvoja web aplikacija se odvija u 3 koraka: editovanje, snimanje i osvežavanje web pregledača? ***

- TRUE

16. Za komunikaciju sa serveltima, korsite se web pregledači? ***

- TRUE

17. Praćenje sesije primenom servelta je moguće? ***

- na klijentu modifikovanje URL-a,
- na serveru primenom metoda HttpSession interfejsa,
- na klijentu primenom kolačića.

18. Šta od navedenog je tačno? ***

- Oracle GlassFish aplikacioni server je baziran na JavaEE implementaciji.
- Nove verzije Jave su unazad kompatibilne a to znači da sve postojeće web aplikacije mogu bez ikakvih problema da se izvršavaju na najnovijoj platformi.
- Jedan dopunski GlassFish server može da upravlja većim brojem instanci koje su simultano pokrenute od strane više softverskih komponenata.

TEST 2

1. Objasnite kako web server kreira web stranicu koristeći JSP?

- Koraci su:
  1) Veb pregledač šalje HTTP zahtev serveru,
  2) Veb server prepoznaje da je HTTP zahtev za web stranicu i prosleđuje ga JSP kontejneru.
  3) JSP mehanizam učitava JSP stranicu sa diska i konvertuje je u srevlet sadržaj
  4) JSP kontejner prevodi servlet u izvršnu klasu I prosleđuje originalni zahtev serevlet kontejneru
  5) Veb server učitava servlet klasu I izrvšava je
  6) Veb server prosleđuje HTTP odgovor ka web pregledaču
  7) Veb pregledač rukuje dinamički generisanim HTML sadržajem.

2. Koji zadatak obavlja JSP kontejner?

- **JSP kontejner** prevodi servlet u izvršivu klasu i prosleđuje originalni zahtev servlet kontejneru.

3. Navesti i objasniti faze životnog ciklusa JSP stranice?

- Faze su:
  = **Kompajliranje** – JSP kontejner vrši: Parsiranje JSP stranice, Prevođenje JSP atranice u servlet i potom kompajliranje Servleta.
  = **Inicijalizacija** – kada kontejner učita JSP stranicu on poziva jspInt() metodu pre servisiranja bilo kog zahteva. Za izvršenje zahteva, potrebno je redefinisati ispInit() metodu.
  = **Izvršavanje** – ovde se dešavaju sve interakcije sa zahtevima, do konačnog uništenja JSP stranice.
  = **Čišćenje** – ovo je faza destrukcije JSP stranice koja se odnosi na uklanjanje JSP stranice iz upotreba od strane kontejnera koristi se redefinisanje jspDestroy() metode.

4. Šta je skriptlet?

- **Skriptlet** je osnovni element JSP sintakse <% kod %>. Njime može biti obuhvaćen bilo koji broj Java iskaza, promenljivih ili deklaracija metoda ili izraza koji su validni.

5. Šta su JSP direktive?

- JSP direktive predstavljaju instrukcije koje imaju implikacije na celokupnu strukturu servlet klase. Postoje 3 direktive: **page**, **include** i **taglib**.

6. Koji zadatak imaju JSP akcije?

- **JSP akcije** su elmenti XML sintakse za kontrolu ponašanja servlet kontejnera.

7. Šta su implicitni objekti? Navedite ih i objasnite.

- JSP implicitnih objekata, ima 9 i to su:
  - ***request*** - Predstavlja ***HttpServletRequest*** objekat pridružen zahtevu;
  - ***response*** - Predstavlja ***HttpServletResponse*** objekat pridružen odgovoru klijentu;
  - ***out*** - Predstavlja ***PrintWriter*** objekat zadužen da pošalje izlaz ka klijentu;
  - ***session*** - Predstavlja ***HttpSession*** objekat pridružen zahtevu;
  - ***application*** - Predstavlja ***ServletContext*** objekat pridružen kontekstu aplikacije;
  - ***config*** - Predstavlja ***ServletConfig*** objekat pridružen stranici;
  - ***pageContext*** - Enkapsulira primenu specifičnih elemenata servera;
  - ***page*** - Predstavlja sinonim za ***this*** i koristi se za pozivanje metoda definisanih prevedenom servlet klasom;
  - ***Exception*** - Ovaj objekat dozvoljava pristup podacima izuzetaka odgovarajućom JSP stranicom.

8. Objasnite razliku između GET i POST metoda?

- **GET** je podrazumevana metoda za prosleđivanje informacija od web pregledača ka web serveru, informacije se prosleđuju kao dugačak string. Ovu metodu nikada ne treba koristiti kada se prosleđuju informacije poput korisničkog imena i lozinke. Ova metoda ima ograničen broj karaktera u jednom GET stringu na 1024 karaktera.
- **POST** metoda pakuje informaciju na isti način kao i GET metoda, ali umesto upotrebe stringa nakon specijalnog znaka „?“ u URL-u, informacija se šalje kao odvojena poruka., koja stiže do backend programa.

9. Kada zahtevom šaljete informacije poput korisničkog imena i lozinke za koju biste se metodu opredelili? Zašto?

- Ove informacije šaljemo POST metodom, zato što GET metodu ne treba korisiti kada se prosleđuju infromacije poput korisničkog imena i lozinke, zbog pouzdanosti podataka.

10. Instrukcije koje imaju implikacije na celokupnu strukturu servlet klase su? ***

- JSP direktive

11. JSP kontejner prevodi servlet u izvršivu klasu I prosleđuje originalni zahtev servelt kontejneru?***

- TRUE

12. Šta je od navedenog tačno za JSP stranice? ***

- JSP mehanizam učitava JSP stranicu sa diska i konvertuje je u servelt sadržaj.
- Veb server učitava servlet klasu I izvršava je.

13. Šta je od navedenog tačno za JSP stranice? ***

- Skriptlet može da sadrži petlje
- Od naredbi grananja skriptlet može da sadrži if I switch naredbe.

14. Ukoliko je vreme modifikacije JSP stranice starije od serveleta, JSP kontejner će smatrati da je JSP stranica promenjena i da se generisani servlet još uvek podudara sa JSP sadržajem? ***

- TRUE

TEST 3

1. Zašto se koriste servelti u kombinaciji sa JSP filterima?

- Servleti i JSP filteri se koriste za:
  = Presretanje zahteva klijenata pre nego što pristupe resursima servleta.
  = Manipulisanje odgovorima servera pre nego što se proslede klijentu.

2. Koje tipove JSP filtera poznajete?

- Postoje: filteri autentikacije, filteri kompresovanja podataka, filteri enkripcije, filteri okidači za pristupanje događajima, filteri konverzije slike, filteri logovanja i provere, filteri lanca MIME-TYPE, filteri tokenizacije, XSL/T filteri za transformisanje XML sadržaja.

3. Navesti i objasniti korake koji se izvode kada se koriste kolačići za identifikovanje korisnika?

- Tri koraka:
  1) Server skript šalje skup kolačića ka veb pregledaču,
  2) Veb pregledač snima kolačiće na lokalnom računaru za buduću upotrebu,
  3) Sledeći put kada pregledač pošalje bilo kakav zahtev ka veb serveru, on šalje i podatke kolačića serveru.

4. Objasniti kako se podešavaju i koriste kolačići u JSP stranicama?

- Podešavanje kolačića se vrši u 3 koraka:
  1) Kreiranje Cookie objekta – poziva se konstruktor sa nazivom i vrednošću kolačića.
  2) Vrši se podešavanje maksimalnog vremena validnosti kolačića – u sekundama.
  3) Slanje kolačića u zaglavlje HTTP odgovora pomoću instrukcije **response.addCookie()**.

5. Ako klase za rad na veb aplikaciji nisu direktno dostupne, kako mogu da se učine dostupnim?

- Neophodno ih je dodati odgovarajućim bibliotekama u okviru foldera Libraries kreiranog veb projekta.

6. Objasniti i demonstrirati različite načine primene Java klase Date u JSP stranicama?

- Klasa Date se nalazi u paketu java.util i enkapsulira tekući datum i vreme. Ona poseduje podrazumevani konstruktor Date() i drugi konstruktor koji uzima kao argument broj milisekundi od 1 Januara 1970 god., čime računar računa tekuće vreme (npr: Date(long milisec.)).

7. Kako se formiraju datum i vreme u JSP stranicama?

- Kreira se objekat klase Date i poziva se metoda **toString()**, koja konvertuje objekat datuma u String i vraća rezultat. Pomoću klase **SimpleDateFormat**, koja formatira i prosleđuje datum u zavisnosti od standarda i regiona.

8. Objasniti kako je moguće na dva načina realizovati preusmeravanje JSP stranica?

- Preusmeravanje (redirect) se koristi da bi klijent bio upućen na novu lokaciju. Vrši se pozivom **sendRedirect()** metode. Takođe, moguće je koristiti metode **setStatus()** i **setHeader()** u kombinaciji sa objektom odgovora.

9. Kako su klasifikovani JSTL tagovi?

- JSTL tagovi mogu biti klasifikovani po nameni u sledeće grupe:
  = Osnovni tagovi,
  = Tagovi za formatiranje,
  = SQL tagovi
  = XML tagovi
  = JSTL funkcije

10. Objasnite pojedinačno kategorije JSTL tagova?

- **Osnovni tagovi**, su u najfrekventnijoj upotrebi i zahtevaju uključivanje Core JSTL biblioteke da bi mogli da budu korišćeni.
- **Tagovi za formatiranje**, se koriste za formatiranje i prikazivanje teksta, vremena, datuma, i brojeva za internacionalizovane veb sajtove.
- **SQL tagovima** je omogućeno interreagovanje sa relacionim bazama podataka u JSP stranicama.
- **XML tagovima** je omogućeno kreiranje i manipulisanje XML dokumentima iz JSP stranica.
- **JSTL funkcije** su zadužene za manipulisanje sa stringovima (uglavnom).

11. Šta je JavaBean i koje ima specifičnosti?

- JavaBean je specijalno konstruisana Java klasa kreirana u skladu sa JavaBeans API specifikacijama. **Specifičnosti** su:
  = Obezbeđuju podrazumevane argumente bez konstruktora,
  = Treba da implementiraju Serializable interfejs.
  = Sadrže brojne osobine koje je moguće učitati i upisivati.
  = Za osobine sadrže brojne setter i getter metode.

12. Objasnite primenu akcije useBean u JSP stranicama?

- Akcija **useBean** omogućava primenu zrna u JSP stranici. Kada je jednom zrno deklarisano, ono postaje skripting promenljiva kojoj je moguće pristupiti odgovarajućim elementima u JSP stranici.

13. U kojim slučajevima u JSP stranicama je moguće koristiti JSP EL izraz?

- **EL** – Expression Language, omogućava jednostavan pristup aplikativnim podacima koji se čuvaju u komponentama koje odgovaraju Java zrnima. On dozvoljava kreiranje izraza koji mogu biti aritmetički i logički. U okviru JSP EL izraza, moguće je koristiti podatke reznih tipova: numeričke (celobrojne i realne), stringove, ugrađene konstante, logičke konstante, … i sl.

14. Koja Java klasa je zadužena za korišćenje lokalnog jezika, pisma i podešavanja u JSP stranicama?

- Klasa Locale.

15. Brisanje kolačića se vrši podešavanjem starosti kolačića na nulu pozivom metode **setMaxAge()**? ***

- TRUE

16. Filter je Java klasa koja implementira interfejs: koji? ***

- javax.servlet.Filter

17. JSP ne podržavaju lokalizaciju? ***

- FALSE

18. Šta od navedenog je tačno? ***

- Moguće je koristiti metode **setStatus()** i **setHeader()** u kombinaciji sa objektom odgovora za preusmeravanje stranica?
- Preusmeravanje zahteva na novu stranicu može veoma jednostavno da bude obavljeno pozivom metode **sendRedirect()** objektom odgovora.

TEST 4

1. Šta čini jednu JSF aplikaciju?

- **JSF aplikacija** se sastoji iz jedne ili više XHTML stranica koje sadrže JSF tagove, jedan ili više CDI zrna i opciono konfiguracione datoteke pod nazivom faces-config.xml.

2. Navedite i objasnite faze razvoja JSF aplikacije?

- **Faze** su:
  1) **Developmnet** – ovde se dodaju inf. koje pomažu prilikom debagovanja,
  2) **Production** – se bavi performansama,
  3) **SystemTest** – testiranje celog sistema,
  4) **UnitTest** – jedinično testiranje.

3. Šta predstavlja facelet?

- **facelet** je XHTML datoteka sa spacifičnim JSF prostorom naziva (namespace). On je automatski generisan.

4. Objasnite JSF menadžer rasporeda?

- On definiše raspored elemenata na Veb stranici.

5. Kako omogućavamo primenu CSS stilova?

- Tako što u JSF stranicu uključimo datoteku **styles.css** pomoću elementa:

<h:outputStylesheet name=”styles.css”>

Tag za ukljucivanje CSS datoteke u JSF stranice dostupan je od JSF verzije 2.0

6. Objasnite vezu između **for** i **id** atributa?

- Prilikom korišćenja taga **<h:label>** za definisanje polja za unos teksta ili taga **<h:message>** za definisanje greške validacije, neophodno je eksplicitno definisati **id** atribut polja za unos koji mora da se podudara sa atributima **for** navedenih tagova. Ukoliko programer nije specificirao vrednost za atribut **id**, ona će automatski biti generisana.

7. Šta je CDI zrno?

- **CDI** (Context and Dependency) **zrna** su klasična Java zrna čiji je zadatak prihvatanje podataka koje je korisnik uneo u JSF aplikaciji.

8. Kako se obeležava klasa CDI zrna?

- anotacijom **@Named**

9. Koje oblasti moze da pokrije CDI zrno? Kojim anotacijama su odredjene oblasti?

- Anotacijama:
  = Request - @RequestScoped
  = Session - @SessionScoped
  = Converstion - @ConversationScoped
  = Application - @ApplicationScoped
  = Dependent - @DependentScoped
  = Flow - @FlowScoped

10. Šta je klasa Validator?

- To je klasa koja implementira interfejs **javax.faces.validator**. I omogućava proveru podataka unetih putem forme na veb aplikaciju.

11. Koji zadatak obavlja klasa validator?

- Obavlja preuzimanje i proveru ispravnosti podataka unetih od strane korisnika.

12. Objasnite primenu atributa required?

- Atribut **required** označava polja koja su obavezna za unos i ukoliko neko od njih nije popunjeno, javiće se greška.

13. Navedite prednosti primene JSF šablona?

- **Šabloni** omogućavaju da se izgled i raspored JSF stranice specificira na jednom mestu, postiže se veći stepen održivosti kreirane veb aplikacije, jedna promena reflektuje se na sve klijente šablona.

14. Kako se kreira datoteka šablona?

- Koraci su: **File → New File → JavaServerFaces → Facelets Templates**

15. Šta je klijent šablona?

- To je implementacija koja omogućava upotrebu šablona „**Facelets template client**“.

16. Šta predstavlja Resource Library Contract?

- To je podrška za kreiranje Facelet šablona koji se koriste za kreiranje tematskih veb aplikacija.

17. Objasnite kako se uključuju PrimeFaces komponente u projekat veb aplikacije?

- Pri kreiranju projekta odabira se **JavaServer Faces** okvir a zatim klikom na tab **Components** bira se opcija **PrimeFaces** i na kraju se klikne **Finish**.

18. Objasnite primenu i organizaciju komponente?

- Pomoću atributa position kojima se dodeljuju vrednosti: **north**, **west**, **east**, **south**, **center** se vrši organizacija komponenti na posmatranoj stranici.

19. Objasnite specifičnosti PrimeFaces stranica u odnosu na standardne JSF stranice?

- Većina specifičnih JSF tagova poseduje odgovarajuću PrimeFaces alternativu.

20. Objasnite kako se uključuju ICEFaces komponente u projekat veb aplikacije?

- Kreiramo nov **Java Web** projekat, u prozoru u kome se bira korišćenje **JavaServer Faces** okvira, klikom na tab **Components**, odabira se **IceFaces biblioteka** čekiranjem.

21. Koji skup ICEFaces komponenata smo koristili i zašto?

- Postoje 2 skupa komponenata:
  1) **ICE** komponente sa server-side funkcionalnošću (garantuje kompatibilnost unazad).
  2) **ACE** komponenta sa kombinacijom server-side i client-side koda. Ova druga komponenta se više koristi jer je okrenuta ka novijim verzijama brovsera.

22. Objasnite kako se uključuju RichFaces komponente u projekat veb aplikacije?

- Prilikom kreiranja **Java Web** projekta, neophodno je izabrati **JavaServer Faces** okvir a u tabu **Components** selektovati **RichFaces** opciju, čekiranjem.

23. Objasnite i demostrirajte upotrebu komponente ??????????????

- Funkcioniše slično kao Combobox.

24. Šta od navedenog je tačno? ***

- Svaka JSF forma se realizuje tagom koji je ekvivalentan tagu standardne HTML stranice.

25. Podešavanjem JSF faze projekta na vrednost Production, odgovarajuće validacije biće automatski dodate u stranicu, bez potrebe da ih programer eksplicitno dodaje u kod odgovarajućim tagovima? ***

- TRUE

26. CDI (Context and Dependency Injection) zrna su klasična Java zrna čiji je zadatak prihvatanje podataka koje je korisnik uneo u JSF aplikaciji? ***

- TRUE

27. CDI zrna mogu pokrivati različite oblasti aplikacije (npr: Request, Session, Conversation, …)? ***

- TRUE

28. Šta je od navedenog tačno? ***

- Iz ugla aplikativnog programera, JSF aplikacija se sastoji iz jedne ili više XHTML stranica koje sadrže JSF tagove, jedan ili više CDI zrna i opciono konfiguracione datoteke pod nazivom faces-config.xml?

TEST 5

1. Šta je REST?

- **REST** (Representational State Transfer) predstavlja arhitekturni stil u kome su veb servisi reprezentovani kao resursi i mogu biti identifikovani preko jedinstvenih identifikatora resursa (URI-Uniform Resource Identifier).

2. Koja je namena RESTFull servisa?

- **RESTFull veb servisi** se koriste za izvođenje CRUD (Create, Read, Update i Delete) operacija nad bazom podataka.

3. Kako se kreira datoteka RESTFull servisa nad tabelom baze podataka u razvojnom okruženju NetBeans IDE?

- Automatski pomoću NetBenasIDE-a, gde se pri kreiranju projekta bira: **Web Services** → **RESTFull Web Services From Database** → Zatim se izabere tabela iz baze podataka.

4. Šta predstavlja Fasadna klasa?

- **Fasadna klasa** je omotač odgovarajuće klase JPA entiteta i svaka fasadna klasa nasleđuje apstraktnu klasu **AbstractFacade.java.**

5. Šta se testira opcijom Test Resource URI?

- Testiraju se i druge metode aktuelnog veb servisa.

6. Objasnite postupak testiranja RESTFull servisa opcijom Test RESTFull Services?

- Koraci su: Desni klik na projekat → **Test RESTFull Web Services** → prihvati se veb klijent i vraca se XML i JSON odgovor.

7. Koji zadatak ima klasa ApplicationConfig.java?

- Osnovna namena ove klase je da konfiguriše JAX-RS. A jedini zahtev koji klasa ApplicationConfig.java mora da ispuni je nasleđivanje osnovne klase javax.ws.rs.ApplicationPath(„webservices“) i njeno obelezavanje anotacijom **@javax.ws.rs.ApplicationPath(“webservices”).**

8. Kako se generisu datoteke Java klijenata RESTFull veb servisa?

- Pomocu NetBeansIDE wizarda-a. On ima zadatak da poziva metode RESTfull servisa na osnovu odgovarajuceg HTTP zahteva. Izabira se opcija **File → New File** → zatim iz kategorije **Web Services** se kao tip datoteke bira **RESTfull Java Client**.

9. Kako se generisu datoteke JS klijenata RESTfull veb servisa?

- Kreira se projekat iz kategorije **HTML5/JavaScript** → definise se ime projekta i klikne se na **Finish**. Zatim se bira **File → New File**, zatim se , … ???????

10. Kako se angazuje generisani JS kod?

- Prevodjenjem oba projekta i izborom opcije Deploy za projekat RESTFullDemo, moguce je pristupiti testiranju kreiranog JavaScript klijenta RESTFull servisa.

11. Klijent RESTful servisa koristi RESTfull veb servis za izvodjenje CRUD (Create, Read, Update, i Delete) operacija nad bazom podataka? ***

- TRUE

12. RESTfull veb servisi se ponasaju kao frontend za bazu podataka da ili ne? ***

- TRUE

13. Pre testiranja RESTFull servisa neophodno je izvrsiti desni klik na koren projekta i izabrati opciju deploy? ***

- TRUE

14. Kako se naziva arhitekturni stil u kojem su veb servisi reprezentovani kao resursi i mogu biti identifikovani preko jedinstvenih identifikatora resursa? ***

- REST

15. Sta od navedenog je tacno? ***

- Anotacija **@Path** je upotrebljena sa ciljem identifikovanj URI (Uniform Resource Identifier) identifikatora za koji ce kreirana fasadna klasa slati zahteve.

TEST 6

1. Koja je razlika između standardnih JSF i CDI aplikacija?

   - Razlika se ogleda u primeni CDI obleženih zrna umesto JSF upravljanih zrna za model i kontrolere.

2. Objasnite ulogu anotacije @Named?

   - Klasa koja je obelezena ovom anotacijom predstavlja klasu CDI zrna.

3. Objasnite razliku izmedju JSF i CDI oblasti zrna?

   - CDI uvodi dve dodatne oblasti: Conversation i Dependent.

4. Objasnite primenu anotacije @Inject?

   - Ova anotacija omogucava jednostavnu primenu umetanja zavisnosti u CDI aplikacijama.

5. Sta je kvalifikator?

   - Kvalifikatori se koriste za ukazivanje specificnog tipa kojeg je neophodno umetnuti u kod. To je specificna anotacija.

6. Kako se kreira kvalifikator?

   - **File → New File → Context and Dependency Injection → Qualifier Type**.

7. Kojom anotacijom se obelezava kvalifikator?

   - **@Qualifier**

8. Opisite kako se primenjuje kvalifikator na potklasu ili implementaciju interfejsa?

   - Da bi ovo bilo obavljeno neophodno je kreirati potklasu koja ce odgovarati potrebi i koja ce biti obelezena kvalifikatorom **@Premium**.

9. Sta je stereotip u JavaEE aplikacijama?

- **CDI stereotipovi** dozvoljavaju kreiranje novih anotacija kojima je moguce zameniti grupu od nekoliko CDI anotacija.

10. Kako se kreiraju stereotipovi?

- U aktuelnom projektu, kreira se novi fajl iz kategorije **Context nad Dependency Injection**, tipa **Steretype**. Nakon toga vrse se dopunska podesavanja stereotipa, kao sto su odredjivanje naziva datoteke, paketa kojem datoteka pripada i slicno.

11. Koje datoteke su neophodne da bi tipovi povezivanja presretaca imali punu funkcionalnost?

- Potrebno je kreiranje CDI konfiguracione datoteke koja ce nositi zvanican naziv **beans.xml**.

12. Sta su tipovi povezivanja presretaca?

- Tipovi presretaca predstavljaju anotacije predstavljene pomocu rezervisane reci **@InterceptorBinding**.

13. Objasnite kako se kreiraju tipovi povezivanja presretaca?

- U aktuelnom projektu, bira se opcija za kreiranje nove datoteke, zatim se iz **Context and Dependency Injection** bira tip datoteke **Interceptor Binding Type** i nakon toga se izvrse i dodatna podesavanja, naziva datoteke, paketa kome datoteka pripada, … i slicno.

14. Kako se kreiraju vlastite CDI oblasti?

- Pomocu carobnjaka (wizard-a) za kreiranje vlastitih prilagodjenih oblasti CDI zrna.

15. Koja vrsta programera ce koristiti pristup kreiranju vlastitih CDI oblasti?

- Programeri koji kreiraju okvir (framework) nad CDI zrnima.

16. Sta od navedenog je tacno? ***

- CDI aplikacije koriste fejslete (facelets) kao vlastitu tehnologiju pogleda (views).

17. U odnosu na obicna JSF zrna CDI ima dve oblasti manje: Conversation i Dependent? ***

- FALSE

18. Sta od navedenog je tacno? ***

- CDI kvalifikator je specificna anotacija predstavljena rezervisanom recju @Qualifier.
- Za umetanje potklase ili klase koja implementira neki interfejs zaduzen je kvalifikator.

19. U odnosu na obicna JSF zrna, CDI uvodi i dve dodatne oblasti: Conversation i Dependent? ***

- TRUE

20. Sta od navedenog je tacno? ***

- Ukoliko se zeli da zrno nosi drugaciji naziv, taj naziv je moguce proslediti kao atribut value anotacije **@Named**.

TEST 7

1. Šta predstavlja JMS?

- **JMS** (Java Message Service) je standardni JavaEE API za upravljanje porukama koji omogućava u formi labave povezanosti komponenata asinhronu komunikaciju između JavaEE komponenata.

2. Koje destinacije JMS poruka poznajete?

- Postoje 2 **destinacije**:
  = Redovi sa porukama – JMS queve
  = Teme - topic

3. Koje domene JMS poruka poznajete?

- Postoje 2 **domena poruka** (messaging domans):
  = point-to-point (PTP) messaging,
  = publish/subscribe (pub/sub) messaging

4. Koje JMS destinacije su karakteristicne za različite domene JMS poruka?

- Ukoliko se koristi PTP domen poruka, kao JMS destinacija, javljaju se redovi za poruke (message queves).
- Ukoliko se koristi pub/sub domen poruka kao JMS destinacija, javlja se tema poruka (message topic).

5. Objasnite kako se dodaje red za poruke na aplikativni server za kreirani JavaEE projekat?

- Da bi red bio kreiran, potrebno je iz menija izabrati **File → New File →** a zatim iz kategorije **GlassFish** servera izabrati tip datoteke **JMS Resource**, zatim klik na **Next →** uneti **JNDI** (npr> jms/mojRed), pa **→ javax.jms.Queve →** pa klik **Finsh**.

6. Kako se koristi GlassFish konzola za proveru kreiranih JMS resursa?

- Na projektu klik na **Services+ →** desni klik na **GlassFish 4.1.1 →** odaberem **View Domain Admin. Console →** desni klik na projekat **→ Deploy**.
  Zatim u **JMS Resources/Destintion Resources** nadjem **jms/mojRed.**

7. Kako se naziva XML datoteka u kojoj je upisan kreirani red za poruke?

- **glassfish-resources.xml**

8. Šta je red za poruke?

- JMS destinacija je red za poruke za aplikativni server.

9. Objasnite kako JMS proizvođač postavlja poruke na red za poruke?

- Potrebno je iz menija izabrati **File → New File** → Iz **Categories → GlassFish** → tip datoteke **JMS Resource → Next** → unesemo **JNDI** → unesemo naziv reda i odaberemo **javax.jms.Queve**.

10. Objasnite proces automatskog generisanja JMS koda. Objasnite detalje koda koji je generisan I koji je kasnije dodat automatski generisanom kodu?

- //?????????????

11. Šta je Message Driven Bean?

- **Message Driven Bean** (zrna vođena porukama) su vrsta EJB zrna čija je namena osluškivanje JMS poruka postavljenih u red za poruke ili temu. Ono nam omogućava da primamo poruke.

12. Koje osobine može da poseduje zrno vođeno porukama?

- Osobine su: **acknowledgeMode** (auto_acknowledge ili dups_ok_acknowledge), clientId, **connectionFactoryLookup**, **destinationType**, **destinationLookup**, **messageSelector**, **subscriptionDurability**, **subscriptionName**.

13. Koji interfejs implementria zrno vođeno porukama?

- Interfejs **javax.jms.MessagListener.**

14. Koji zadatak ima metoda **onMessage()** zrna vođenog porukama?

- Ona se automatski poziva kada je poruka primljena na JMS destinaciji koje kreirano zrno vođeno porukama osluškuje. U njoj definišemo putanju na kojoj će poruka završiti (u našem slučaju, to je na konzoli NetBenas-a).

15. Oblasti koje su specifične samo za CDI zrna su? ***

- Dependent i Conversation

16. EJB Module preojekat sadrži samo EJB zrna dok Enterprise Application projekat sadrži EJB zrna zajedno sa njihovim klijentima u formi Java Web aplikacije ili *standalone* Java aplikacije, da ili ne? ***

- TRUE

17. U aplikaciji može da postoji samo jedna instanca zrna? ***

- singlton zrno sesije

18. Šta od navedenog je tačno? ***

- Krajnja tačka WebSocket servera je klasa obeležena anotacijom **@ServerEndpoint**

19. Java Message Service (JMS) predstavlja standardni JavaEE API (Application Programming Interface) za upravljanje porukama koji omogućavaju u formi krute povezanosti komponenata, sinhronu komunikaciju između Java EE komponenata? ***

- TRUE

20. Vrednost koja je istaknuta u okviru anotacije @ServerEndpoint, ukazije na uniformni identifikator resursa (URI-Uniform Resource Identifier) krajne tacke servera I klijente koji pristupaju krajnjoj tacki servera preko navedenog URI, da ili ne? ***

- TRUE

21. EJB zrno koje se koristi za implementiranje JMS potrosaca poruka je? ***

- zrno vodjeno porukama (message driven bean)