UNIVERZITET METROPOLITAN BEOGRAD

CS230 - DISTRIBUIRANI SISTEMI

Uvod u distribuirane sisteme i Java EE

Lekcija 01

PRIRUČNIK ZA STUDENTE

CS230 - DISTRIBUIRANI SISTEMI

Lekcija 01

UVOD U DISTRIBUIRANE SISTEME I JAVA EE

- Uvod u distribuirane sisteme i Java EE
- Poglavlje 1: Uvod u distribuirane sisteme
- Poglavlje 2: Oblikovanje distribuiranih sistema
- Poglavlje 3: Opravdanost i primeri distribuiranih sistema
- Poglavlje 4: Distribuirani sistemi - platforme i tehnologije
- Poglavlje 5: Podešavanje okruženja za razvoj JEE programa
- Poglavlje 6: Individualna vežba 1
- Poglavlje 7: Domaći zadatak 1
- Zaključak

UVOD

Lekcija će dati studentima sliku o distribuiranim sistemima i tehnologijama za njihovu implementaciju.

U današnjem vremenu sve više se javlja potreba za distribuiranim, transakcionalnim i prenosivim aplikacijama koje povećavaju brzinu, sigurnost i pouzdanost server-side tehnologije.
U prvom delu rada opisane su karakteristike i princip rada Distribuiranog Sistema. Razvoj distribuiranih sistema je sledio pojavu brzih lokalnim mreža početkom sedamdesetih godina. Pojavom jakih osobnih računara, radnih stanica i servera došlo je do napuštanja centraliziranih i višekorisničkih računara, čemu je pridonio i razvoj distribuirane programske podrške, kao i distribuiranih aplikacija.
U drugom delu rada je opisana Java Enterprise Edition tehnologija koja predstavlja razvojno okruženje za složene aplikacije. Glavni cilj Java EE platforme je da obezbedi programerima moćan aplikacioni programski interfejs (API), dok sa druge strane smanjuje vreme razvoja i složenost aplikacije, a poboljšava aplikacione performanse.
Posebno će, u ovoj lekciji, biti navedene komponente Java EE platforme koje će u narednim lekcijama biti detaljno analizirane i demonstrirane.

Savladavanjem ove lekcije student će biti upoznat sa konceptom distribuiranog sistema i osnovnim elementima koji čine Java Enterprise Edition platformu.

Poglavlje 1

Uvod u distribuirane sisteme

UVODNA RAZMATRANJA

Distribuirani sistem je kolekcija procesora koji komuniciraju slanjem poruka preko komunikacione mreže.

Distribuirani sistem je kolekcija procesora koji komuniciraju slanjem poruka preko komunikacione mreže. Razlozi korišćenja distribuiranih sistema su brojni. Kao prvo, veći broj procesora omogućava da se reši problem brže u odnosu na jednoprocesorski sistem. Kao drugo, podaci su distribuirani na veći broj lokacija što smanjuje uska-grla u komuniciranju. Kao treće, različiti procesori mogu sada da dele resurse kakvi su štampači, diskovi, i drugo.
Veliki broj aplikacija distribuiranih sistema zahteva visok nivo pouzdanosti u radu, kakvi su na primer, sistemi za kontrolu leta u avionskom saobraćaju, sistemi za upravljanje radom satelita, i brojni drugi. Kako se obim ovih sistema povećava potreba za visokom - pouzdanošću u radu (fault tolerance- FT) postaje imperativ. Algoritmi koji se ugrađuju kod FT sistema treba da obezbede produžetak njihovog rada i u slučajevima kada je ograničeni broj komponenata tih sistema u kvaru.

Kada se navode osobine distribuiranih sistema postoji nekoliko pristupa, međutim, generalno gledano, moguće je istaći sledećih šest osnovnih karakteristika distribuiranih sistema. To su:

- deljenje resursa (resource sharing),
- otvorenost (openess),
- istovremenost (concurrency),
- skalabilnost (scalability),
- otpornost na greške (fault tolerance)
- transparentnost (transparency)

Nijedna od tih karakteristika nije posledica distribuiranih sistema. Distribuirani sistemi moraju biti pažljivo planirani i izvedeni na taj način da osiguraju svaku od navedenih karakteristika.

DELJENJE RESURSA

Resurs pokazuje šta je sve moguće deliti u distribuiranim sistemima.

Iako je pojam resursa apstraktan, on najbolje opisuje šta je sve moguće deliti u distribuiranim sistemima. Prednosti deljenja resursa kao što su baze podataka, programi, dokumentacija i ostale informacije su se prvi puta pokazale pojavom višekorisničkih sistema (multi-user) te sistema sa podelom vremena (time sharing) početkom 1960-ih, te pojavom višekorisničkih UNIX operativnih sistema 1970-ih godina.

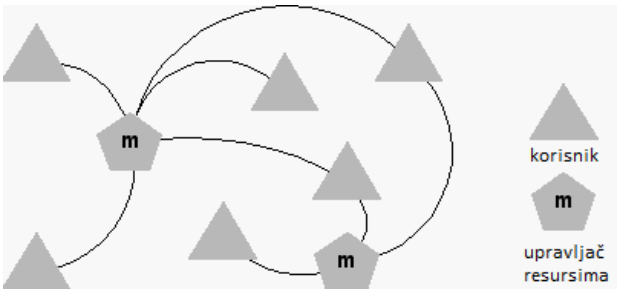
U samom početku akcenat je bio na deljenju uređaja kao što su štampači, diskovi velikog kapaciteta, kao i ostali uređaji koji mogu biti deljeni zbog lakoće upotrebe i pristupačnosti.

Poseban cilj je bio deljenje podataka. Deljenje podatakaje jedan od osnovnih zahteva u mnogim računarskim primenama. Posebno je neophodno uzeti u obzir sledeća razmatranja:

- Timovi za razvoj programske podrške mogu pristupati dosada razvijenim delovima programa i deliti iste alate za razvoj i na taj način koristiti samo jednu kopiju prevodioca, biblioteka i editora. Na taj način nova verzija alata postaje odmah dostupna svim korisnicima;
- Mnoge komercijalne aplikacije omogućavaju korisnicima korišćenje objekata u jednoj zajedničkoj bazi podataka;
- Najperspektivnije područje primene distribuiranih sistema je podrška grupama korisnika u timskom i kooperativnom radu koje uveliko zavisi od deljenja podataka između programa na različitim radnim stanicama.

Od posebnog značaja su mehanizmi upravljanja resursima u distribuiranim sistemima. Izraz upravljač resursa opisuje programski modul koji upravlja određenim resursom.

Sledeća slika prikazuje distribuirani sistem sastavljen od upravljača resursa i korisnika resursa. Korisnici resursa komuniciraju sa upravljačima resursa da bi dobili mogućnost korišćenja deljenog resursa.



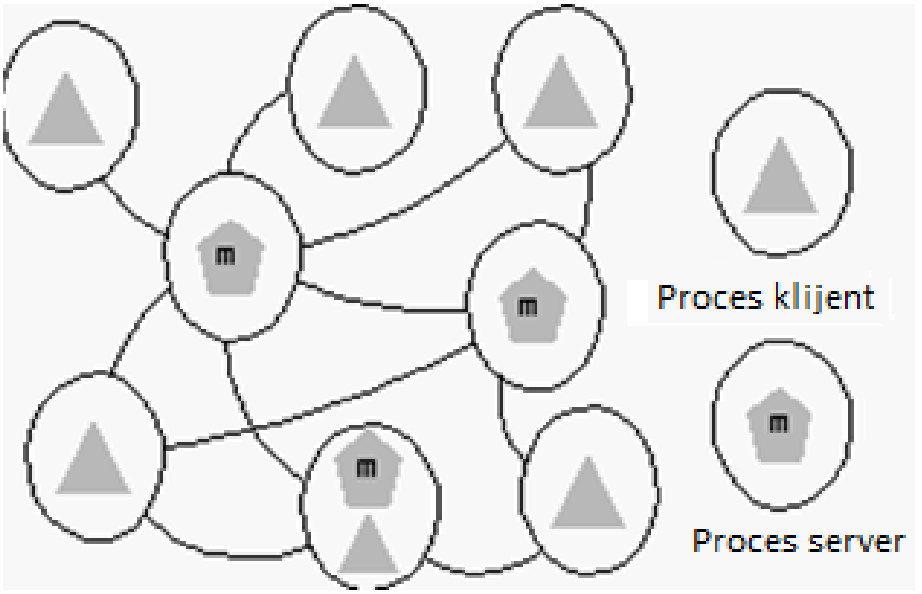
Slika 1.1.1 Upravljači i korisnici resursa [izvor: autor]

Iz prezentovanog načina razmatranja razvijaju se dva osnovna modela: klijent-server model i model zasnovan na objektima.

DELJENJE RESURSA - KLIJENT - SERVER MODEL

Klijent - server je najviše primenjivan model distribuiranih sistema.

U ovom delu lekcije će akcenat biti na razmatranju klijent - server modela. Istorijski gledano, to je najviše primenjivan model distribuiranih sistema. Sastoji se od procesa servera koji su upravljači resursa određenog tipa i od procesa klijenta koji za izvršavanje svog zadatka zahtevaju pristup deljenim resursima. Sami serveri mogu trebati neke od deljenih resursa, i neki serveri mogu istovremeno biti klijenti nekim drugim serverima.



Slika 1.1.2 Procesi klijenti i server [izvor: autor]

Proces se u ovom kontekstu shvata jednako kao u operacijskim sistemima: program u izvođenju.

Pojednostavljeni pogled na klijent-server model može biti centralizirani server resursa. Međutim, centralizirano posluživanje nije poželjeno u distribuiranim sistemima. Zbog toga se pravi razlika između servera (servers) i usluga koje oni pružaju (services). Usluga je apstraktni entitet koji se može pružati preko nekoliko servera na različitim računarima koji sarađuju preko mreže.

Klijent - server model je uspešno primenjen kod deljenja različitih resursa: elektronske pošte, mrežnih novosti, kod deljenja datoteka, sinhronizacija satova računara u mreži, deljenja prostora na diskovima i brojnim drugim. Međutim, nije moguće sve resurse deliti na taj način. Neki resursi moraju ostati lokalni za svaki računar, kao na primer: radna memorija (RAM), centralna upravljačka jedinca (CPU) i mrežni interfejs se smatraju najmanjim skupom resursa koji moraju ostati lokalni. Ti se ključni resursi mogu deliti samo među procesima koji se izvode i funkcionišu na istom računaru.

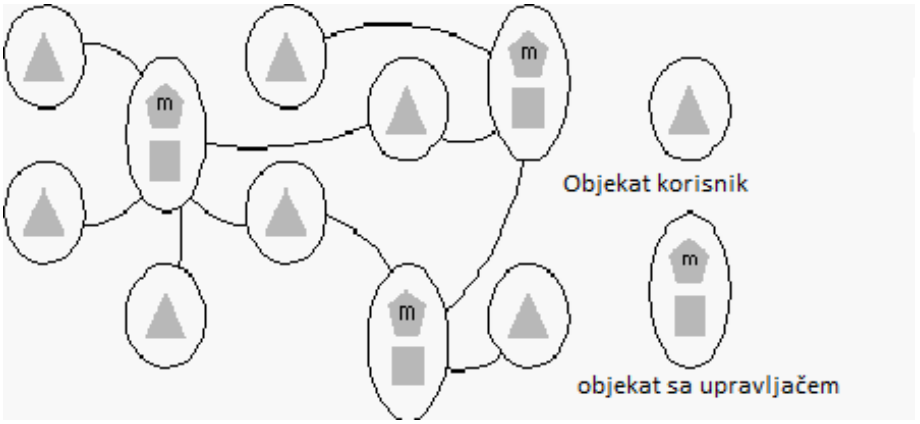
Klijent - server model ne zadovoljava sve uslove - neke primene zahtevaju čvršću povezanost između klijenta nego što je to moguće u klijent - server modelu, međutim, on je pogodan za veliki broj primena i kao baza za opšte distribuirane sisteme.

DELJENJE RESURSA - MODEL ZASNOVAN NA OBJEKTIMA

Ovaj model je sličan tradicionalnom objektnom modelu koji se koristi kod programiranja.

Ovaj model je sličan tradicionalnom objektnom modelu koji se koristi kod programiranja. On svaki izvršni deo programa posmatra kao objekat koji poseduje sistem za razmenu poruka

pomoću kojeg se pristupa prema mogućnosti objekta. U objektnom distribuiranom modelu, svaki deljeni resurs posmatra se kao objekat. Objekti su jednako određeni i mogu menjati položaj na mreži bez menjanja identiteta. Kada program pokaže potrebu za resursom, on šalje poruku koja sadrži zahtev za objektom. Poruka se prosleđuje odgovarajućoj proceduri ili procesu koji izvodi zahevanu operaciju i šalje odgovor ukoliko je to potrebno.



Slika 1.1.3 Objekti i upravljači [izvor: autor]

Slika pokazuje poznatu jednostavnost takvog modela. On omogućava na ujednačen način pristup svim deljivim resursima od strane objekata korisnika. Kao i kod klijent - server modela, objekti mogu biti istovremeno i upravljači i korisnici. Dok kod klijent - server modela način imenovanja zavisi od servera koji je pružao uslugu, kod objektnog modela je imenovanje svih resursa uvek ostvareno na isti način.
Kod primene objektnog modela pojavljuju se, takođe, i određeni problemi. On zahteva da se upravljači objekata nalaze na istom mestu kao i objekti kojima oni upravljaju zbog toga što oni poseduju reprezentaciju stanja upravljanog objekta. To je jednostavno kod sistema kod kojih se objekti ne mogu pomerati i taj pristup je primenjen u većini današnjih objektnih distribuiranih sistema.

OTVORENOST

Otvorenost je karakteristika distribuiranih sistema koja definiše mogućnost proširivanja na različite načine.

Otvorenost je karakteristika distribuiranih sistema koja definiše mogućnost proširivanja na različite načine. Sistem može biti otvoren na sklopovskom nivou - na primer za dodavanje dodatnih spoljnih uređaja, memorije ili komunikacijskih uređaja, ili na programskom nivou - za dodavanje novih funkcija distribuiranom sistemu, novih komunikacijskih protokola ili servisa za deljenje resursa. Otvorenost distribuiranih sistema se većinom odnosi na mogućnost dodavanja servisa za deljenje resursa bez uznemiravanja ili udvajanja postojećih resursa.

Ako bi ovde bilo neophodno u najkraćim crtama definisati otvorenost. to bi mogle da se svede na sledeće:

- Karakteristika otvorenih sistema da svoj interfejs publikuju;

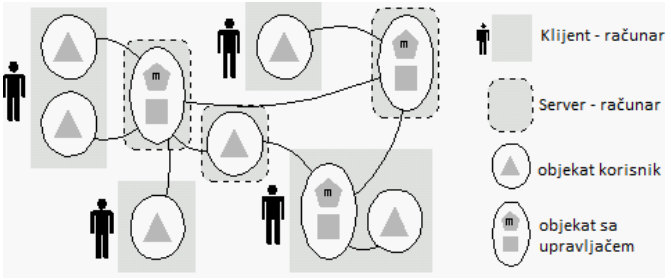
- Otvoreni distribuirani sistemi se bazira na pružanju ujednačenih mehanizama komunikacije među procesima i predstavljanju interfejsa da bi se omogućio pristup do deljenih resursa;
- Otvoreni distribuirani sistemi se mogu graditi od delova i programske podrške različitih proizvođača;
- Međutim, da bi se korisnici zaštitili od neusklađenosti, svi delovi se moraju brižljivo testirati s obzirom na predstavljeni interfejs.

ISTOVREMENOST

U distribuiranom sistemu se nalazi više računara, svaki sa jednim ili više centralnih upravljačkih jedinica.

Kada na istom računaru postoji nekoliko procesa, oni se odvijaju istovremeno. Ako je računar opremljen samo sa jednom centralnom upravljačkom jedinicom, onda se to postiže naizmeničnim izvršavanjem procesa. Ukoliko računar ima N centralnih upravljačkih jedinica, do N procesa se može izvršavati paralelno što takođe rezultuje višestrukim poboljšanjem reda u direktnoj srazmeri sa brojem N.
U distribuiranom sistemu se nalazi više računara, svaki sa jednom ili više centralnih upravljačkih jedinica. Ako se radi o N računara, onda se istovremeno može izvršavati N procesa. Istovremenost se postiže u dva slučaja (kao što je to prikazano na narednoj slici):

1. Kada više korisnika istovremeno koriste program.
2. Kada više procesa servera istovremeno poslužuju zahteve sa više procesa korisnika.



Slika 1.1.4 Istovremenost kod distribuiranih sistema [izvor: autor]

Istovremenost i paralelnost se, dakle, pojavljuju prirodno kod distribuiranih sistema kao posledica smeštanja procesa servera na različite računare. Takva podela procesa rezultira paralelnošću izvršavanja na različitim računarima. Istovremeni pristup i menjanje deljenih resursa mora biti usklađeno.

TRANSPARENTNOST

Transparentnost se definiše kao stepen skrivanja pojedinih komponenata distribuiranog sistema .

Transparentnost se definiše kao stepen skrivanja pojedinih komponenata distribuiranog sistema od krajnjeg korisnika i programera aplikacija na način da se sistem shvata kao celina, a ne kao skup nezavisnih delova.
Deljenje sistema na delove je osnovna karakteristika distribuiranih sistema. Posledice toga uključuju potrebu za komunikacijom među delovima i potrebu za upravljanjem i integracijom delova. Podela omogućava stvarni paralelizam u izvođenju programa, prikrivanje ispada opreme i oporavljanje od grešaka bez uznemirivanja celog sistema, izolaciju i kontrolu komunikacijskih kanala kao meru zaštite sadržaja kod prenosa, te rast ili smanjivanje sistema dodavanjem ili oduzimanjem komponenti.
ANSA Reference Manual i International Standards Organization's Reference Model for Open Distributed Processing (RM-ODP) određuju osam tipova transparentnosti. To su i osnovni motivi za izgradnju distribuiranih sistema:

- Transparentnost pristupa - omogućava da se lokalni i udaljeni resursi predstavljaju na jednak način.
- Transparentnost pozicije - omogućava da informacije o objektima ne ovise o fizičkom smeštaju.
- Transparentnost istovremenosti - pruža mogućnost da više procesa istovremeno pristupa podacima bez problema koji se pri tome mogu pojaviti.

- Transparentnost udvajanja - služi da se poveća pouzdanost i kvaliteta pristupa udvajanjem objekata.
- Transparentnost na greške - omogućava izvršavanje korisničkih zadataka bez obzira na ispade opreme.
- Transparentnost na promene lokacije sadržaja bez uticaja na korisnike.
- Transparentnost performansi - omogućava prilagođivanje sistema za veća opterećenja bez uznemirivanja korisnika.
- Transparentnost na rast - omogućuje rast bez promena strukture sistema ili algoritama.
- Dve najvažnije transparentnosti su transparentnost na pristup i transparentnost pozicije. Njihova prisutnost ili nedostatak najjače utiču na upotrebu distribuiranih resursa. Jednim imenom se nazivaju mrežna transparentnost.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

1.1 Rukovanje greškama distribuiranih sistema

SKALABILNOST I OTPORNOST NA GREŠKE

Skalabilnost i otpornost na greške su veoma bitne karakteristike distribuiranih sistema.

Distribuirani sistemi imaju jednu bitnu karakteristiku - moraju da funkcionišu efikasno bez obzira na veličinu. Najmanji praktičan distribuirani sistem sastoji se od dve radne stanice i

datotečnog servera. Distribuirani sistemi sagrađeni oko lokalne mreže mogu imati više stotina radnih stanica i više servera različitih namena. Više lokalnih mreža može biti međusobno povezano tako da više hiljada računara zajedno čine jedan distribuiran sistem koji omogućava deljenje resursa među njima.
Potreba za skalabilnošću nije samo problem opreme ili mrežnih karakteristika. Taj problem prožima sve delove distribuiranih sistema. Za razliku od centraliziranih sistema gde su resursi kao što su memorija, upravljačke jedinice i ulazno-izlazni kanali ograničeni i ne mogu se dodavati proizvoljno, distribuirani sistemi omogućavaju dodavanje takvih resursa bez fiksnih ograničenja. Međutim, ukoliko sistem nije bio planiran na adekvatan način, imajući u vidu skalabilnost, mogu se pojaviti izvesna ograničenja.
Ne postoji savršen računarski sistem. Čak i najnapredniji i najskuplji računarski sistemi ponekad zataje. Kada se greška pojavi u sklopovima ili programskoj podršci, mogu se dobiti pogrešne vrednosti ili nepotpuni rezultati.
Dizajn sistema otpornih na greške temelji se na dva osnovna načela, po jedno za svaki tip grešaka:

- udvajanje sklopova - korišćenjem više istih komponenti sklopova,
- otklanjanje programskih grešaka - oblikovanje programa na način da se sami mogu oporaviti od grešaka.

Da bi se stvorio sistem otporan na sklopovske greške, često se koriste dva međusobno povezana sistema za korišćenje samo jedne aplikacije, gde jedan služi kao rezerva ukoliko prvi sistem otkaže.
Kod distribuiranih sistema redundantnost može biti na mnogo finijem stepenu - pojedini servisi koji su kritični za funkcioniranje sistema mogu biti udvojeni. Udvojeni sklopovi se mogu koristiti za nekritične poslove kada oba sistema rade bez grešaka. Serveri mogu biti dizajnirani na taj način da otkrivaju greške kod obrade u udvojenom sistemu. Nakon greške, klijenti se preusmeravaju na drugi sistem. Zbog takvih karakteristika, otpornost na neke greške sklopova se može osigurati distribuiranim sistemima po relativno maloj ceni. Otklanjanje programskih grešaka uključuje vraćanje na prethodno stanje u slučaju pojave greške. Raspoloživost sklopova je takođe na visokom nivou, jer se u slučaju kvara radne stanice rad se može nastaviti na bilo kojoj drugoj, dok se kod ispada sklopova servera servisi mogu pokrenuti na drugom serveru. Mreže na kojima se temelje distribuirani sistemi najčešće nisu otporne na greške. Ispad mreže onemogućava dalje korišćenje distribuiranog sistema do popravka ispada. Mnogo se truda ulaže u oblikovanje mreža otpornih na ispade.

Poglavlje 2

Oblikovanje distribuiranih sistema

UVODNA RAZMATRANJA

Distribuirani sistem mora da ima formu kojom postiže dobro poznate ciljeve.

U ovom delu lekcije, akcenat će biti stavljen na analizi i predstavljanju oblikovanja distribuiranih sistema za postizanje konkretnih ciljeva koji su u direktnoj vezi sa kvalitetom posmatranih sistema. Osnovi ciljevi oblikovanja distribuiranih sistema su sledeći:

- performanse
- pouzdanost
- skalabilnost
- konzistentnost
- sigurnost

Iako oblikovanje distribuiranih sistema zahteva razmatranje i ostalih problema nevezanih sa distribucijom, ovde će posebno biti govora o sledećem:

- Imenovanje - imena bi trebalo da budu nezavisna od lokacije.
- Komunikacije - potrebno je optimalno koristiti komunikacijske resurse.
- Struktura programske podrške - mora osigurati otvorenost, što se postiže definisanjem interfejsa.
- Raspoređivanje opterećenja - postizanje dobrih svojstava sistema da bi se postigli najbolji rezultati bez obzira na opterećenje sistema.
- Ujednačenost sistema - problem ujednačenosti je jedan od najvećih problema distribuiranih sistema.

IMENOVANJE

Ime označava naziv koji ima značenje za korisnika ili sistem.

Proces koji se obraća zahtevanom resursu sa mora da zna njegovo ime ili identifikator. Ime označava naziv koji ima značenje za korisnika ili sistem, dok identifikator označava naziv koji ima smisla samo za sistem. Kaže se da je ime određeno (resolved), kada je pretvoreno u oblik pogodan za pokretanje akcije nad resursom ili objektom na koji se ime odnosi. U distribuiranim sistemima određeno ime (resolved name) je najčešće komunikacijski identifikator zajedno sa ostalim atributima koji su korisni za uspostavljanje veze.
Sam proces imenovanja je veoma bitan i uključuje nekoliko bitnih odluka:

- Određivanje prikladnog prostora imena (name space) za svaki tip resursa. Prostor imena može biti beskonačan ili konačan, strukturirani ili jednostavan. Resursi istog tipa moraju imati različita imena, bez obzira na njihovu lokaciju. Kod objektnog sistema, svi objekti dele isti prostor imena.
- Iz imena je neophodno imati mogućnost određivanja komunikacijskog identifikatora.
- Imena se često određuju u zavisnosti od konteksta. Zbog toga je potrebno navesti kako ime, tako i kontekst za to ime. Trebalo bi izbegavati uključivanje mrežne adrese ili druge lokacijske oznake u ime resursa. To direktno narušava transparentnost pozicije o kojoj je bilo govora u prethodnom izlaganju. Određivanje odgovarajućeg sistema imena je veoma važno za distribuirane sisteme.

KOMUNIKACIJA

Distribuirani sistemi su sastavljeni od delova koji su fizički i logički odvojeni i koji moraju komunicirati da bi mogli međusobno delovati.

Distribuirani sistemi su sastavljeni od delova koji su fizički i logički odvojeni i koji moraju komunicirati da bi mogli međusobno delovati. Ako se uzme pretpostavka da su komponente koje upravljaju ili trebaju resurse realizovane kao procesi. To je tačna pretpostavka za klijent - server model, dok kod objektnog modela praktična realizacija takođe može bazirati na tom principu.
Komunikacija između dva procesa uključuje slanje i primanje podataka što ima za posledicu:
a) prenos podataka iz procesa koji šalje, procesu koji prima i
b) kod nekih komunikacija sinhronizaciju slanja sa primanjem, tako da je proces koji šalje ili prima privremeno zaustavljen dok ne izvrši do kraja svoju akciju.

Kod slučaja (a) oba procesa dele isti komunikacijski kanal, dok je ponašanje opisano u slučaju (b) karakteristično za svako komuniciranje.

Osnovni način realizacije su šalji (send) i primi (receive) delovi koji zajedno čine akcije za prenos poruke između dva procesa. Akcijom prenosa poruke se određeni podaci (poruka) koju stvara proces koji šalje, prenose korišćenjem određenog komunikacijskog mehanizma (kanala ili porta), do procesa koji podatke prima. Taj mehanizam može biti sinhroni (engleski izraz je blocking) što znači da pošiljalac čeka dok primalac ne primi poruku ili asinhroni (non-blocking) kod kojeg se poruka stavlja u niz poruka koje čekaju na primanje, dok proces koji šalje može nastaviti svoje izvršavanje.

Dva osnovna načina komuniciranja su:

- klijent-server komuniciranje između dva procesa i
- grupno slanje (group multicast) za komuniciranje između grupe procesa koji međusobno sarađuju.

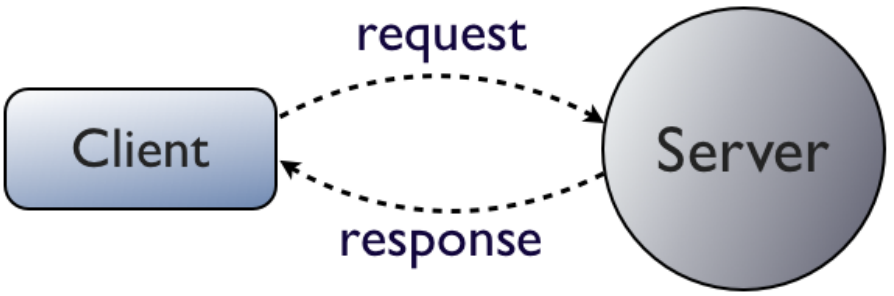
KLIJENT - SERVER KOMUNIKACIJA

Klijent - server komunikacija je orijentisana prema pružanju usluge.

Cilj ovog dela lekcije je kratko prezentovanje i osvrt na klijent server tip komunikacije. Klijent - server komunikacija je orijentisana prema pružanju usluge. Proces razmene podataka po ovom modelu je veoma jednostavan i poznat i sastoji od:

1. proces klijent šalje zahtev procesu server;
2. zahtev se obrađuje na serveru;
3. server šalje odgovor klijentu.

Ovakav proces razmene podataka može biti ilustrovan sledećom slikom.



Slika 2.1 Klijent - server komunikacija [izvor: autor]

Sledi kratko objašnjenje klijent - server tipa komuniciranja u distribuiranom sistemu. Takav način komuniciranja uključuje prenos dve poruke i sinhronizaciju klijenta i servera. Iako se ovaj način komunikacije može implementirati korišćenjem osnovnih operacija šalji i primi, najčešće se upotrebljava iz viših programskih jezika preko RPC-a (remote procedure calling) interfejsa koji skriva komunikaciju od korisnika.

GRUPNO SLANJE (GROUP MULTICAST)

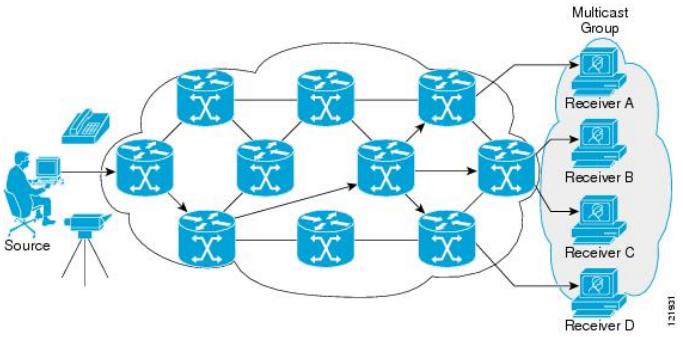
Kod grupnog komuniciranja procesi razmenjuju poruke na taj način da svaka poruka ide svim procesima u grupi.

U narednom izlaganju biće ukratko reči o sledećem tipu komunikacije poznatom kao grupno slanje (group multicast). Kod grupnog komuniciranja procesi razmenjuju poruke na taj način da svaka poruka ide svim procesima u grupi, a ne samo jednom. Jednom šalji pozivu odgovara više primi poziva, po jedan na svakom članu grupe. Takvo slanje naziva se multicast.

Grupno slanje ima sledeće dobre strane:

- Nezavisno je od lokacije objekta kojem je upućena poruka - poruka se šalje svim objektima, dok odgovara samo onaj kojem je namenjena.
- Otpornost na greške - poruka može biti poslata istovremeno na više servera, tako da na nju odgovara jedan ili više od njih. Kvar jednoga od servera klijent ne primećuje.
- Istovremeno usklađivanje svih članova grupe - jedan server može poslati vreme grupnim slanjem tako da se svi ostali serveri i klijenti sinhronizuju na to vreme.

Sledećom slikom može biti demonstriran ovaj tip komunikacije.



Slika 2.2 Grupno slanje podataka [izvor: autor]

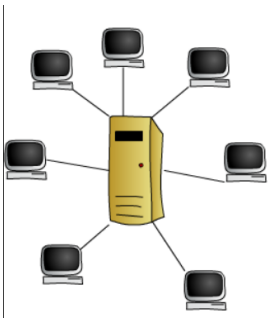
STRUKTURA PROGRAMSKE PODRŠKE

Paralela strukture programske podrške centralizovanih i distribuiranih sistema.

Kod centralizovanih sistema glavna komponenta je operativni sistem. On upravlja svim resursima i pruža usluge uslužnim programima i korisnicima koje uključuje:

- osnovno upravljanje resursima
- dodeljivanje i zaštita memorije
- stvaranje i određivanje redosleda izvršavanja procesa
- upravljanje spoljnim uređajima
- usluge programima i korisnicima
- provera korisnika i kontrola pristupa
- upravljanje datotekama i pravima pristupa
- usluge sata.

Sve te servise,kod centraliziranog sistema, obavlja kernel operativnog sistema.



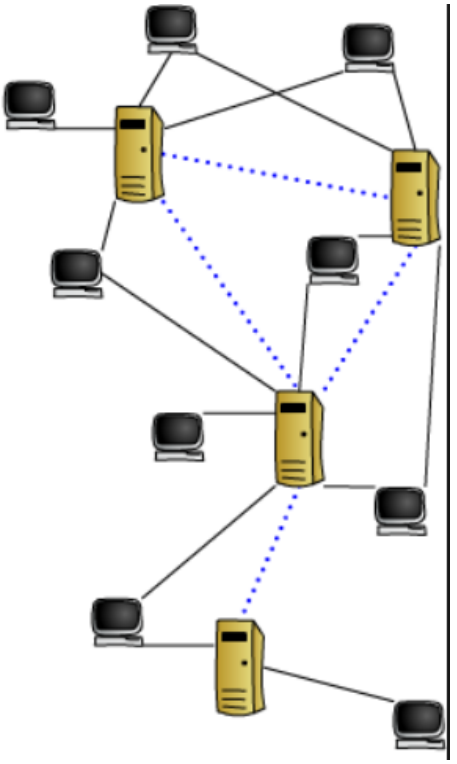
Slika 2.3 Centralizovana arhitektura [izvor: autor]

Distribuirani sistemi pružaju i druge usluge uslužnim programima i na taj način da dodavanje novih servisa ne predstavlja problem. Da bi to bilo moguće, kernel se ograničava na pružanje samo osnovnih usluga upravljanja resursima:

- dodeljivanje i zaštitu memorije
- stvaranje i određivanje redosleda izvršavanja procesa
- komunikaciju među procesima, i

- upravljanje spoljnim uređajima

Uvodi se novi tip servisa nazvan otvoreni servisi koji služi za pružanje usluga pristupa svim ostalim deljenim servisima i resursima. Svi servisi koji nemaju potrebu za pristupom kernel - ovim podacima ili sklopovima implementirani su na istom nivou kao i uslužni programi.



Slika 2.4 Distribuirana arhitektura [izvor: autor]

RASPOREĐIVANJE OPTEREĆENJA

Smeštanje resursa "blizu korisniku" ima mnoge prednosti naročito kod interaktivnih primena.

Kod klasičnih centralizovanih sistema, svi resursi centralne jedinice i memorije su na raspolaganju operativnom sistemu u skladu sa trenutnim opterećenjem. Kod najjednostavnijih distribuiranih sistema resursi centralne jedinice i memorije su ograničeni najvećim raspoloživim resursom na jednoj od radnih stanica. Takav jednostavan model naziva se radna stanica - server model. Smeštanje tih resursa "blizu korisniku" (na njegovu radnu stanicu) ima mnoge prednosti naročito kod interaktivnih primena i to je glavna prednost modela radna stanica - server. Međutim, iz toga slede i ograničenja: model ne koristi optimalno resurse niti omogućava korisniku sa velikim zahtevima pristup do dodatnih resursa.
Zbog toga su se razvile dve modifikacije tog modela:

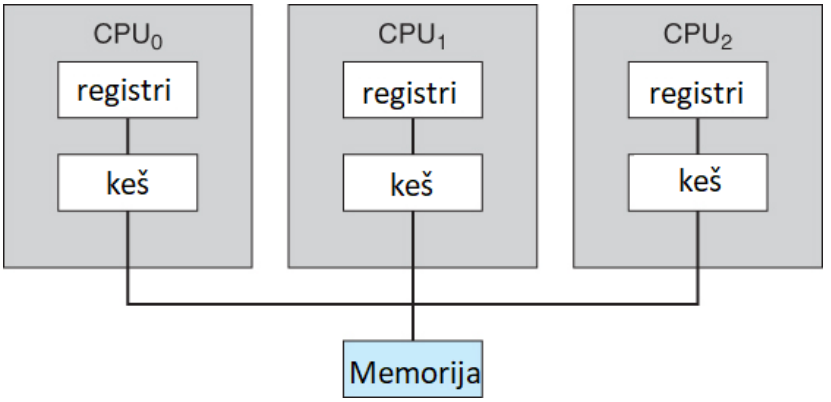
- Procesorski pool je prva od njih, i uključuje dinamičko dodjeljivanje procesorskih resursa korisnicima.
- Druga varijanta je upotreba neiskorišćenih radnih stanica koja koristi stanice koje se trenutno ne upotrebljavaju.

- Jedna od opcija su i višeprocesorski sistemi sa deljenom memorijom koji se mogu uspešno primeniti u oba slučaja, naročito kod velikih opterećenja koja se onda mogu podeliti na više procesora (sledeća slika).

Kod procesorskog pool - a procesori se dodeljuju procesima tokom njihovog izvršavanja, što rezultuje deljenjem "procesora po procesu". Korisnik sa više procesa može iskoristiti više procesorske snage nego što jedna radna stanica može ponuditi.

Procesorski pool sastoji se od većeg broja jeftinih računara, od kojih se svaki sastoji samo od procesora, memorije i mrežnih sklopova. Svaki procesor u pool - u ima svoj priključak na mrežu, kao i radne stanice i serveri. Procesori su smešteni na ploče, koje se montiraju u za to predviđene ormare (rack-ove). Procesori mogu biti različiti po arhitekturi da bi se omogućilo izvršavanje različite programske podrške.

Druga zanimljiva mogućnost je korišćenje radnih stanica koje su neiskorišćene ili malo iskorišćene kao dodatni izvori procesorskih resursa, slično procesorskom pool - u. Primeri takvih sistema su korišćenje crva (worms) koji putuju mrežom, traže neiskorišćene stanice za obradu, i premeštanje procesa (process migration) koje omogućava da se procesi vrate svom "izvorištu" u slučaju kada se stanica počne više iskorišćavati.



Slika 2.5 Više nezavisnih procesora koji izvršavaju procese i dele memoriju [izvor: autor]

KONZISTENTNOST SISTEMA

Konzistentnost je važan problem kod distribuiranih sistema zbog deljenja resursa i istovremenosti.

Konzistentnost ili održivost je važan problem kod distribuiranih sistema zbog deljenja resursa i istovremenosti. U ovom delu lekcije će akcenat biti na najvažnijim slučajevima:

- Konzistentnostkod obnavljanja sadržaja - Ovaj se problem javlja kada nekoliko procesa pristupa i menja podatke istovremeno. Menjanje podataka ne može biti trenutno, ali bi trebalo biti atomska (elementarna) operacija, tako da svim ostalim procesima izgleda kao da je trenutno. Problem nije isključivo vezan za distribuirane sisteme, već postoji svugde gde se dele podaci.
- Konzistentnost kod udvajanja - Kada se podaci sa jednog izvora udvoje na više različitih lokacija javlja se problem ukoliko se oni i promene na jednoj od lokacija. Sve ostale

lokacije moraju biti obaveštene o takvoj promeni, da bi sve kopije podataka bile identične.

- Konzistentnost privremenememorije (cache-a) - Problem ujednačenosti privremene memorije je specijalni slučaj problema kod udvajanja podataka i rešava se na slične načine.
- Konzistentnost kod neotklonjivih grešaka - Kada kod klasičnog sistema dođe do greške koja prekida izvršavanje obrade, svi procesi se zaustavljaju. Međutim, kod distribuiranih sistema verovatno je da će kod ispada jedne od komponenti sistema ostali procesi biti u različitim tačkama izvršavanja. Zbog toga je potrebno vratiti se na zadnje poznato i ispravno stanje (roll back) da bi se obrada mogla da bude nastavljena.
- Konzistentnost sata - Mnogi od algoritama distribuiranih sistema zavise od vremenskih oznaka (time stamps), tako da je usklađenost satova kod distribuiranih sistema veoma bitna.
- Konzistentnost korisničkog interfejsa - Korisnički interfejs bi trebalo da reaguje na komande korisnika dovoljno brzo da omogući praćenje promena koje se dešavaju radom korisnika. Vreme kašnjenja se sastoji od: vremena potrebnog da se korisnički unos primi, obradi, da se izračunaju potrebne promene na ekranu i vremena potrebnog da se promene prenesu do korisničkog sistema prozora i da se obnovi slika na ekranu.
  Po ergonomskim studijama, to vreme bi trebalo biti manje od 0.1 sekunde.

Poglavlje 3

Opravdanost i primeri distribuiranih sistema

PARALELE IZMEĐU CENTRALIZOVANIH I DISTRIBUIRANIH SISTEMA

U ovom delu lekcije vrši se poređenje posmatranih tipova sistema.

Pre bilo kakvog isticanja opravdanosti primene distribuiranih sistema biće urađena kraća analiza kojom će biti istaknute karakteristike sistema sa centralizovanom arhitekturom i distribuiranih sistema. Upravo, povlačenjem paralela između ovih sistema biće identifikovane prednosti, ali i uočeni nedostaci u teoriji i praksi primene distribuiranih sistema.

U prvom koraku, biće priložene opšte poznate karakteristike sistema čija organizacija počiva na centralizovanom upravljanju podacima i resursima:

- jedna komponenta ne sadrži neautonomne delove - softverski elementi ne mogu da funkcionišu samostalno kao programske celine;
- konstantno deljenje komponente između korisnika - više korisnika koristi istu softversku komponentu i upravlja njenim sadržajem;
- svi resursi sistema su dostupni;
- jednoprocesnost - softver se izvršava kroz jedan proces;
- jedinstvena kontrola programskog toka - programska kontrola je fiksirana na jednom mestu;
- jedna lokacija za upravljanje greškama.

Nakon isticanja karakteristika centralizovanih softverskih sistema, na red dolaze i karakteristike sistema sa distribuiranom arhitekturom. Distribuirani sistemi poseduju sledeće, opšte prihvaćene, karakteristike:

- više autonomnih komponenata - softverski elementi mogu da funkcionišu samostalno kao programske celine;
- korišćenje komponenata je ograničeno po korisnicima - komponentu može da koristi veći broj korisnika i da upravlja njenim sadržajem;
- resursi sistema ne moraju uvek da budu dostupni - dostupnost i prava korišćenja zavise od autentifikacija i autorizacije;
- softver se izvršava u paralelnim procesimana više procesora;
- paralelno izvođenje procesa - zahteva više programskih kontrola;
- više lokacija za lociranje i upravljanje greškama.

DISTRIBUIRANI SISTEMI PREDNOSTI I NEDOSTACI

Nakon analiziranih osobina, sledi isticanje prednosti i nedostataka distribuiranih sistema.

Distribuirani sistem, kao skup nezavisnih računara i odgovarajuće softverske i komunikacione podrške, predstavljeni su korisnicima kao jedinstvena celina. Ovi sistemi imaju široku praktičnu primenu, zato im se i opravdano posvećuje velika teorijska pažnja. Posebno je potrebno opravdati njihovo korišćenje na adekvatan način. Iz navedenog razloga su u prethodnom izlaganju navedene karakteristike ovih tipova sistema, ali i centralizovanih, sa ciljem uočavanja i isticanja prednosti, ali i nedostataka, u primeni distribuiranih računarskih sistema.

Sledi lista opšte prihvaćenih prednosti primene distribuiranih sistema:

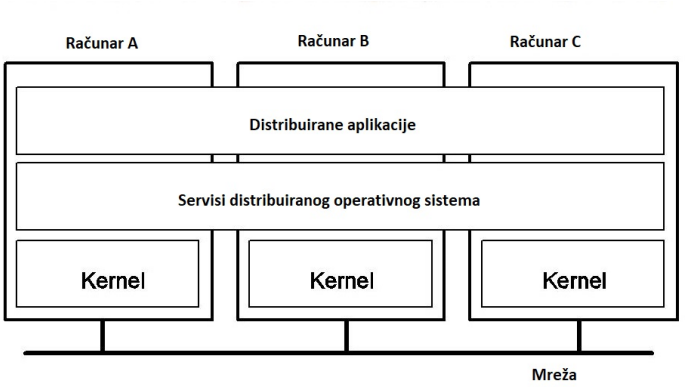
- ekonomičnost – smanjeni zahtevi za hardverom, smanjeni zahtevi za softverom;
- brzina – obrade se izvode na više računara;
- podjela posla – ravnomerno opterećenje računarske mreže
- pouzdanost – greška u obradi ponavlja se samo na jednom računaru
- proširivost - lako se dodaju nove komponente u sistem.

Budući da ne postoji savršen računarski sistem, po bilo kojoj komponenti (hardver, softver ili mreža), ni distribuirani sistemi nisu savršeni. Pored navedenih prednosti identifikovani su i određeni nedostaci sa kojima primena ovakvih sistema, u budućnosti, mora da posebno da se pozabavi.

Sistemi bazirani na distribuiranoj arhitekturi imaju sledeće opšte prihvaćene nedostatke:

- softver - kompleksniji i zahtevniji nego kod centralizovanih sistema;
- mreža - sistem obavezno zahteva mrežu za funkcionisanje;
- više komponenti može biti izvan funkcije;
- sigurnost - stavka o kojoj mnogo više mora da se vodi računa zbog mogućnosti pristupa podacima sa raznih lokacija;

Iako je mreža okarakterisana kao nedostatak, on se mora posmatrati kao jedna komponenta više u sistemu. Mrežni sistemi omogućavaju i realizovanje kontrolisanog pristupa drugim računarima i prenos podataka između njih, a to otvara brojne druge prednosti u primeni ovakvih sistema.



Slika 3.1 Mreža u distribuiranom sistemu [izvor: autor]

INTERNET KAO DISTRIBUIRANI SISTEM

Prezentacija dobro poznatih distribuiranih sistema iz prakse.

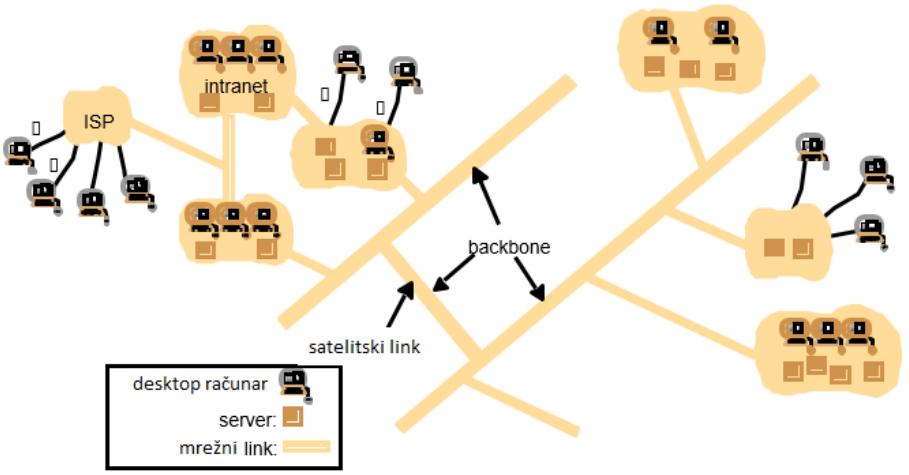
U sledećem izlaganju je neophodno predočiti studentima implementaciju nekih dobro poznatih distribuiranih sistema. Posebno će za svaki od njih biti navedene i relevantne karakteristike. Kao prvi primer primene distribuiranih sistema u praksi moguće je navesti Internet.

NAPOMENA: DS nije mreža ili skup računara. DS predstavlja organizaciju izgrađenu na mreži nezavisnih računara, pri čemu se svi računari prezentuju korisniku kao jedinsteni (lokalni) računar.

Internet kao distribuirani sistem ima sledeće karakteristike:

- Internet mreža svih mreža;
- Internet je globalna mreža - globalni pristup bilo kome kroz podatke, servise i druge vrste IKT sredstava;
- Internet je otvorenog tipa - nadogradiv i proširiv;
- Na Internetu ne postoji jedan autoritet - nije hijerarhijske strukture;
- Komunikacija se izvodi u koracima - proverava se izvor, predstavljanje i prenos podataka;
- Oblici komunikacije su u različitim formatima - audio, video, tekst i drugi oblici podataka.

Od posebnog značaja je vizuelno predstavljanje Interneta kao distribuiranog sistema i uklapanje u opšte poznatu sliku o distribuiranim sistemima. Navedeno će biti realizovano kroz sliku broj 2.



Slika 3.2 Internet kao distribuirani sistem [izvor: autor]

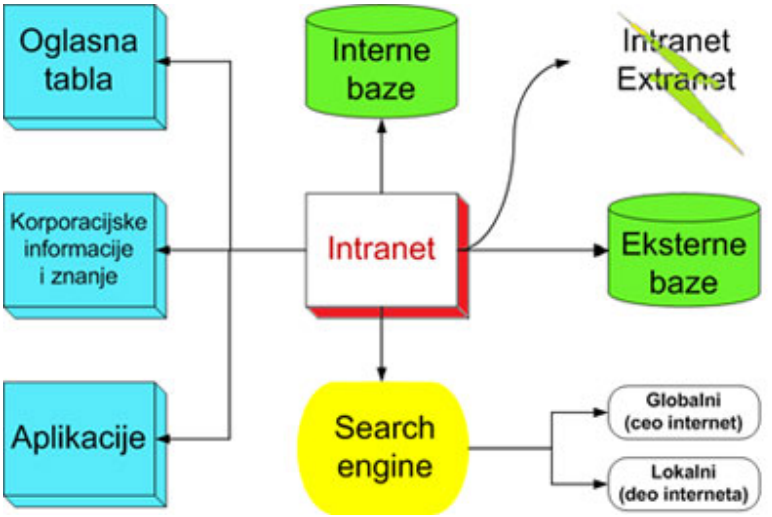
INTRANET KAO DISTRIBUIRANI SISTEM

Drugi primer distribuiranog sistema je intranet.

Sledeći oblik distribuiranog sistema, koji se veoma često sreće u praksi je intranet. Kao unutrašnja mreža neke organizacije ili sistema, intranet ima sledeće opšte prihvaćene karakteristike:

- Jedan autoritet - ponuđač usluge, davalac autentifikacije i autorizacije;
- Zaštićen pristup - najčešće firewall, moguća je potpuna izolacija;
- Nije vezan za fiksnu lokaciju - može biti šire lokacijski pozicioniran;
- Intranet servisi mogu biti : infrastrukturni (datotečni i komunikaciono - prezentacioni) i aplikacijski.

Od posebnog značaja je vizuelno predstavljanje intraneta kao distribuiranog sistema i uklapanje u opšte poznatu sliku o distribuiranim sistemima. Navedeno će biti realizovano kroz sliku broj 3.



Slika 3.3 Intranet kao distribuirani sistem [izvor: autor]

PRIMERI DISTRIBUIRANIH SISTEMA - DISTRIBUIRANI OPERATIVNI SISTEM

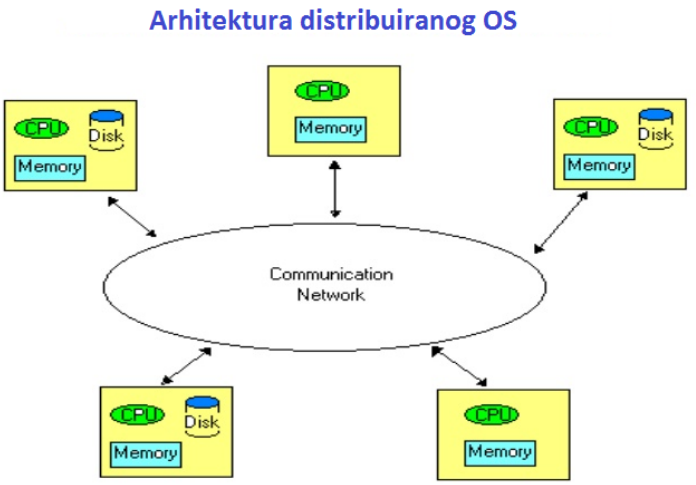
Poseban primer su distribuirani operativni sistemi.

Pod *distribuiranim operativnim sistemom* (DOS) može da se shvati zajednički operativni sistem koji upravlja mrežom računara i njihovim resursima. Iz vizure korisnika DOS izgleda kao običan centralizovani operativni sistem ali on obezbeđuje korisniku pristup resursima većeg broja računara.

> Distribuirani operativni sistemi upravljaju međusobno povezanim računarima, koji su prostorno udaljeni.

Ovakvi udaljeni računari se povezuju komunikacionim linijama (optički kablovi, mrežni kablovi) koje omogućuju razmenu podataka, a međusobna komunikacija ostvaruje se putem LAN ili WAN mreže. Računari ne dele zajedničku memoriju i sistemski časovnik. Umesto toga svaki računar ima sopstveni procesor, radnu memoriju i mrežni kontroler. Na ovaj način nastaje distribuirani računarski sistem (distributed computer system).

Od posebnog značaja je vizuelno predstavljanje DOS kao distribuiranog sistema i uklapanje u opšte poznatu sliku o distribuiranim sistemima. Navedeno će biti realizovano kroz sliku broj 4.



Slika 3.4 DOS kao primer distribuiranog sistema [izvor: autor]

Poglavlje 4

Distribuirani sistemi - platforme i tehnologije

JAVA ENTRERPRISE EDITION - UVODNO RAZMATRANJE O PLATFORMI

Java Enterprise Edition platforma predstavlja moćnu, razvijenu i široko korišćenu platformu za razvoj distribuiranih aplikacija

Sa pojavom Jave dolazi do velikih promena – ona sa sobom nosi veću mogućnost interakcije između odvojenih procesnih jedinica (klijenta i servera) čija je posledica pomeranje granice aktivnisti sa web servera na web klijenta, što nije bio slučaj sa predhodnim softverskim sistemima. Jedan od trendova i tendencija današnjice je da se web aplikacije koriste za rešavanje najrazličitijih problema iz sfere poslovanja kako malih tako i velikih preduzeća.
Kao jedno od najefikasnijih sredstava za rešavanje ovih problema samo po sebi se nameće Java Enterprise Editon tehnologija. Java EE platforma je projektovana da pomogne programerima pri kreiranju velikih, više - nivojskih, skalabilnih, pouzdanih i bezbednih web aplikacija. Ove karakteristike koje čine složene (enterprise) aplikacije veoma moćnim često ih čine i složenim. Java EE paltforma je projektovana tako da smanji cenu i složenost razvijanja aplikacija obezbeđujući moćni razvojni model i razvojno okruženje koje omogućuje programerima da se koncentrišu na funkcionalnost aplikacije. Java EE aplikacije se mogu brzo raspoređivati i lako nadograđivati što sa ostalim navedenim karakteristikama predstavlja odgovor Jave na takmičarski pritisak od strane drugih, konkurentnih programskih jezika.

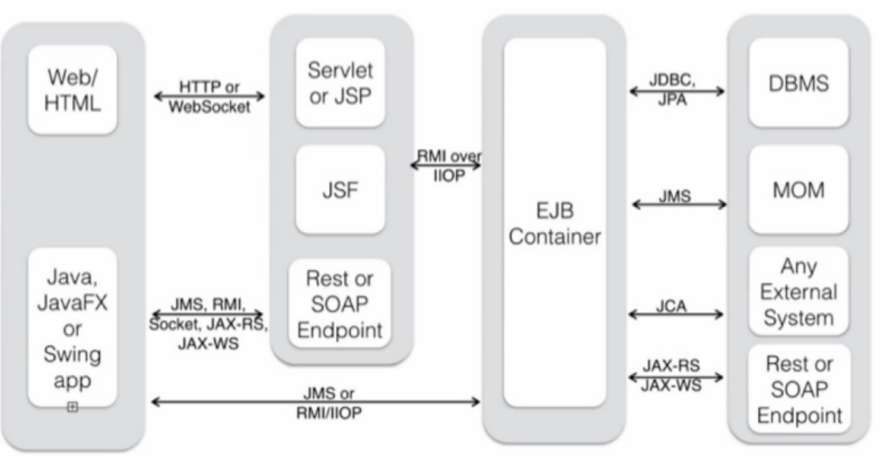
Java Enterprise Edition platforma predstavlja moćnu, razvijenu i široko korišćenu platformu za razvoj distribuiranih aplikacija. Reč Enterprise ne znači da se primenjuje samo na složene aplikacije. Java EE komponente se koriste za razvoj svih vrsta aplikacija,od aplikacije lokalnog prodavca pica, koja se izvršava na računaru od 500 dolara do moćne aplikacije za trgovanje na Volstritu, koja se izvršava na klasteru od nekoliko stotina međusobno povezanih servera (izv, Yakov Fain - Java 8 - programiranje).

Sama platforma je veoma kompleksna i u ovom delu materijala će biti istaknuta najosnovnija razmatranja koja će posebno dobiti širu analizu i diskusiju u narednim lekcijama. Svaka Java EE verzija sadrži skup specifikacija za različite tehnologije, kao sto su servleti, JavaServer Pages (JSP), Enterprise Java Beans (EJB) i Java Messaging Service (JMS). Svaka od ovih specifikacija je definisana u organizaciji koja se naziva Java Community Proces (JCP). Ako neka osoba ili grupa ljudi odluči da predloži specifikaciju određene nove tehnologije , ona kreira zahtev koji se naziva Java Specification Request (JSR) i formira grupu eksperata koji rade na specifikaciji.

SLOJEVI JAVA EE APLIKACIJA

JAVA EE platformu čini više slojeva.

Klijentski sloj se može implementirati na desktop računaru, prenosnom računaru, mobilnom telefinu ili bilo kom drugom uređaju koji ima ugrađeno JRE okruženje ili se može povezati pomoću Java koda sa web serverom. Aplikacije koje povezuju više uređaja su poznate pod nazivom *Internet of Things (IoT)*, one mogu da koriste male senzore sa ugrađenom podrškom za Java jezik i servere koji koriste Java tehologije. Ukoliko kreiramo web aplikaciju , web sloj je veoma važan. Možemo da izaberemo JSP, JavaServer Face (JSF),servlete ili web servise. Ove komponente su odgovorne za izgled klijenta. Java Swing i JavaFX se koriste za složene klijentske aplikacije. Klijent može da sadrži poslovnu logiku koja smanjuje opterećenje servera. Poslovna logika aplikacije razvijena je u sloju poslovne obrade, koji predstavlja EJB u Java EE svetu.
Korišćenje DBMS sistema, ostaje najpopularniji način za skladištenje podataka u poslovnim Java EE aplikacijama. Podaci mogu da se koriste na osnovu eksternih web servisa ili da stižu u realnom vremenu iz nekog sistema za obaveštavanje.
MOM predstavlja skraćenicu za „Message-oriented middleware“.



Slika 4.1 Slojevi Java EE [izvor: autor]

JAVA EE KONTEJNERI

Za svaku od komponenata višeslojne Java EE aplikacije obezbeđen je odgovarajući kontejner.

Za svaku od komponenata višeslojne Java EE aplikacije, Java EE server obezbeđuje odgovarajući kontejner. Kontejneri predstavljaju interfejs između komponenata i funkcionalnosti Java platforme neophodnih za funkcionisanje datih komponenti. Da bi određena komponenta bila spremna za izvršavanje, prethodno mora biti raspoređena u odgovarajući kontejner shodno sloju aplikacije u kojem se nalazi.

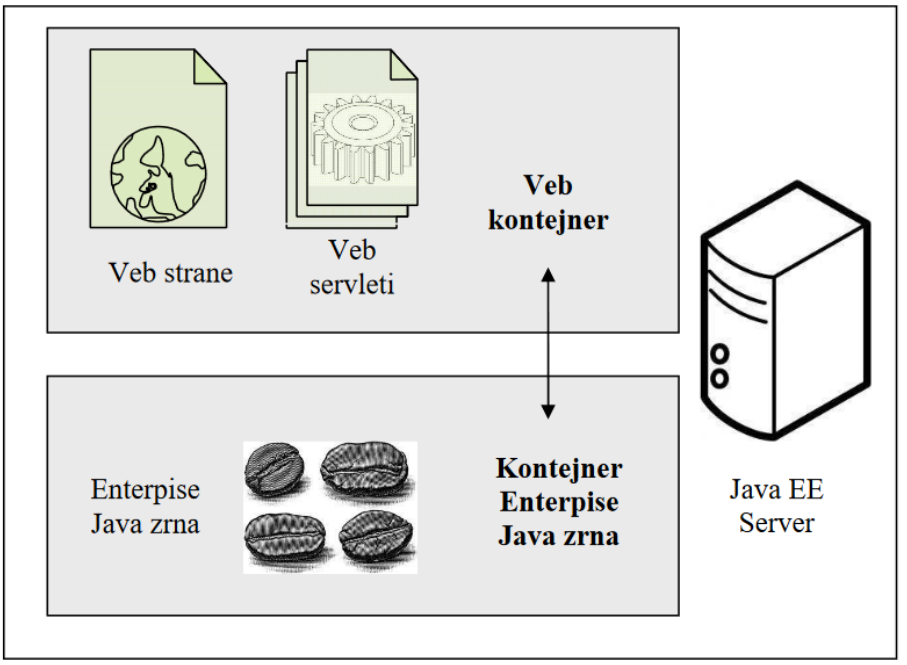
Na Java EE serveru imamo dva osnovna tipa kotejnera i to su:

- Kontejner Enterprise Java zrna (EJB kontejner)
- Veb kontejner

Kontejner Enterpise Java zrna je zadužen za način izvršavanja Enterprise Java zrna, a Veb kontejer je zadužen za način izvršavanja Veb strana i servleta.

U zavisnosti od konfiguracije kontejnera, zavisi ponašanje komponete koja je u njega raspoređena. Drugim rečima, komponente iste aplikacije se mogu ponašati različito u zavisnosti od toga kako su konfigurisani kontejneri u koje su raspoređene. Tako, na primer, u jednom okruženju Enterprise Java zrna mogu imati određena prava pristupa podacima baze, dok u drugom okruženju ta prava pristupa mogu biti potpuno drugačija.

Kontejneri Java EE servera prikazani su sledećom slikom.



Slika 4.2 Java EE kontejneri [izvor: autor]

Enterprise Java zrna se izvršavaju u kontejneru za Enterprise Java zrna te on predstavlja njihovo izvršno okruženje na Java EE serveru. Kontejner pruža zrnima funkcionalnosti platforme kao što su bezbednost i obavljanje transakcija.

ENTERPRISE JAVA ZRNA

Enterprise Java zrna pojednostavljuju razvoj većih, distribuisanih aplikacija.

Enterprise Java zrna pojednostavljuju razvoj većih, distribuisanih aplikacija iz više razloga:

- Prvo, kontejner Enterprise Java zrna pruža sistemske funkcionalnosti kontrolišući instance zrna, transakcije, bezbednosne autorizacije, niti itd. pa osoba koja razvija softver ne mora mnogo da brine o pomenutim funkcionalnostima već može da se posveti rešavanju

problema same poslovne logike. Inače, niti predstavljaju tok programa. Svaki java program ima barem jednu nit koja se kreira prilikom pokretanja aplikacije, ali za izvršavanje relativno nezavisnih delova programa može se koristiti više niti.

- Drugo, pošto zrna sadrže poslovnu logiku aplikacije, ona je jasno razdvojena od prezentacije. To dovodi do takozvanog *laganog klijenta* što je naročito bitno kod manjih uređaja kod kojih bi svaka rutina poslovne logike ili pristup bazi iz sloja klijenta opterećivala uređaj. Takođe, razgraničavanjem komponenti na ovaj način, Veb dizajner ne mora da zna na koji način se implementira određena poslovna logika, već može da se skoncentriše na svoj deo posla.
- Treće, pošto su Enterpirse Java zrna prenosiva, moguće je već postojeća zrna koristiti u razvoju novih aplikacija i na drugim Java EE serverima. Zrna se pakuju u EJB JAR datoteke koje predstavljaju nezavisne komponente. Dakle, zrna se mogu posmatrati kao komponente, koja su, kada su jednom nepravljena, na raspolaganju u novim projektima. Pri pravljenju nove aplikacije mogu se koristiti već gotovi moduli, kao što su EJB JAR datoteke, koje se postavljaju u EAR datoteku koja sadrži aplikaciju.

DODATNO RAZMATRANJE ZA EJB

Neophodno je istaći kriterijume po kojima se koriste Java EE zrna.

Tokom razvoja Java aplikacije moguće je sresti se sa problemom koji podrazumeva donošenje odluke u vezi sa korišćenjem Enterprise Java zrna. Ako aplikacija koja je predmet razvoja zadovoljava barem jedan od sledećih kriterijuma, onda treba razmisliti o upotrebi Enterprise Java zrna:

- Aplikacija bi trebalo da bude skalabilna - Skalabilna aplikacija ima karakteristiku da prilikom povećanja radnog opterećenja mašine ili mašina na kojoj se aplikacija izvršava, na primer zbog povećanja broja korisnika aplikacije, nije potrebno praviti veće izmene same aplikacije već se problem može rešiti povećanjem resursa mašine, odnosno mašina na kojima se aplikacija izvršava. Da bi bio omogućen rastući broj korisnika aplikacije, moguće je izvršiti distribuciju njenih komponenti na više mašina. Ne samo da Enterprise Java zrna aplikacije mogu da se izvršavaju na različitim mašinama, već će njihova lokacija ostati transparentna odgovarajućem kontejneru.
- Transakcije moraju osigurati integritet podataka - Kontejner Eterprise Java zrna brine se o transakcijama, mehanizmu koji upravlja konkurentnom pristupu zajedničkim objektima.
- Aplikacija može da ima raznovrsne korisnike - Ovde se pre svega misli na takozvane lagane klijente, najčešće manje uređaje preko kojih se može pozivati aplikacija. Oni mogu biti raznovrsni i brojni. Na primer, mobilni telefon, tableti i ostali prenosivi uređaji.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VRSTE ENTERPRISE JAVA ZRNA

U ovom delu lekcije će biti predstavljene različite vrste Java EE zrna.

Java EE platforma podržava različite tipove Java EE zrna. Enterprise java zrna mogu biti:

1. Zrna sesije (Session bean) - Zrna sesije izvršavaju zadatake koje im klijent zada, tačnije, klijent poziva metode zrna čijim izvršavanjem zrno pruža traženu uslugu klijentu. Zrna sesije takođe mogu biti implementirana kao Veb servisi. Veb servisi su programske komponente koje komuniciraju sa klijentima koristeći otvorene XML standarde i Internet protokole, kao što je HTTP protokol.
2. Zrna vođena porukom (Message-driven bean) - Zrna voñena porukom služe za osluškivanje i reagovanje na odreñeni tip poruka.

Zrna sesije sadrže poslovnu logiku aplikacije i po potrebi mogu biti pozvana bilo lokalno, bilo sa neke druge mašine na serverskoj strani ili bilo putem Veb servisa. Zrna sesije se koriste pozivanjem metoda koje zrno sadrži. Po pozivanju odgovarajuće metode zrna izvršavaju se pridružene operacije i onom ko je pozvao zrno vraća se rezultat metode. Pozivač zrna ne zna na koji način se pozvana metoda izvršava jer poslovna logika ostaje spakovana u okviru samog zrna.

Postoje tri različita tipa zrna sesije, o kojima će u narednom izlaganju biti više reči, i to su:

1. Zrna stanja (Stateful beans)
2. Zrna bez stanja (Stateless beans)
3. Zrna Jedinci (Singleton beans)

TIPOVI ZRNA SESIJA

Postoje tri različita tipa zrna sesije.

U narednom izlaganju će prvo biti reši o zrnima stanja. Kod zrna stanja, kao što i samo ime sugeriše, zrno predstavlja stanje klijentove sesije. Sesija je jedinstvena, vremenski ograničena veza između identifikovanog korisnika i Veb servera, pomoću koje se obavlja komunikacija sa korisnikom od trenutka njegovog pristupa Veb aplikaciji do trenutka isteka vremenskog ograničenja ili odjave korisnika. Stanje zrna pedstavljeno je vrednostima atirbuta konkretne instance. Ovo stanje još se naziva i stanje konverzacije, pošto je korisnik u stalnoj komunikaciji sa svojom instancom zrna.

Jednoj instanci zrna stanja pridružuje se samo jedan korisnik, tj. ono ne moze biti deljeno između više korisnika istovremeno. Kada korisnik okonča sesiju, ili kada ona istekne, njemu pridružena instanca zrna stanja se uništava i svi podaci o trenutnom stanju se gube. Ovo ne predstavlja problem pošto obično nema potrebe da se podaci o stanju sesije čuvaju po njenom okončanju.

Sledeći tip zrna sesije je zrno bez stanja. Zrna bez stanja nisu u stalnoj komunikaciji sa korisnikom, već samo onda kada korisnik pozove neku od metoda zrna. Instance zrna bez stanja nalaze se u kontejneru Enterprise Java zrna gde čekaju da budu uposlena. Po

korisnikovom pozivu određene metode zrna, kontejner dodeljuje slobodnu instancu zrna korisniku.

Ta instanca će sadržati podatke vezane za konkretnog korisnika dok traje izvršavanje metode koju je korisnik pozvao i već pri sledećem pozivu iste instance od strane novog korisnika ti podaci će biti zamenjeni podacima novog korisnika. Po izvršavanju metoda zrno više nije u vezi sa korisnikom i spremno je da bude uposleno ponovo, kao što su to i ostale slobodne instance zrna u kontejneru. S obzirom da instance zrna bez stanja mogu biti ponovo uposlene kad su slobodne, u aplikacijama sa većim brojem korisnika one imaju bolje performanse od zrna stanja. Obično je potreban manji broj zrna bez stanja nego zrna stanja za pružanje podrške istom broju korisnika.

Zrno bez stanja može da implementira Veb servis dok zrno stanja to ne može.

Na kraju, razmatra se zrno jedinac koje, kada se kreira, predstavlja jedinstvenu instancu zrna za životni ciklus aplikacije. Ovoj instanci mogu konkurentno pristupati različiti korisnici, a instanca zadržava trenutno stanje između različitih poziva, osim ako ne dođe do pada ili spuštanja servera. Moguće je izvesti da se instanca zrna jedinca kreira pri pokretanju aplikacije i u tom slučaju ona može obaviti potrebne zadatke inicijalizacije prilikom pokretanja aplikacije. Takođe, instanca može obaviti poslove čišćenja pre gašenja aplikacije pošto postoji tokom čitavog životnog ciklusa aplikacije.

Kao i zrna bez stanja, zrna jedinci mogu implementirati Veb sevis.

PRISTUP ZRNIMA SESIJE

Pristup je omogućen na dva načina, kroz odgovarajuće interfejse.

Pristup zrnima sesije omogućen je kroz odgovarajuće interfejse. Interfejs zrna sesije može biti:

- lokalni (local);
- udaljeni (remote).

Jedno zrno sesije može imati više interfejsa. Koji tip interfejsa će biti izabran, kao pogodan, zavisi od više faktora, kao što je npr. distribucija komponenata. Ako će se komponente, koje pozivaju zrno, nalaziti na različitim mašinama, onda je potreban udaljeni interfejs, što daje veću fleksibilnost ali manju brzinu poziva. Sa druge strane, neka zrna su tesno povezana i često komuniciraju pa bi se lokalnim pozivima postigle bolje performanse.

Korisnici zrna sesije takođe mogu biti lokalni i udaljeni. Lokalni korisnik ima sledeće karakteristike:

- mora se nalaziti u istoj aplikaciji kao i zrna sesije kojima pristupa;
- može biti Veb komponenta (servlet, dinamička Veb strana) ili drugo zrno;
- lokacija zrna sesije čije metode koristi nije mu transparentna;

Java EE platforma podržava primenu anotacija za obeležavanje tipova zrna i odgovarajućeg tipa interfejsa.

Prilikom implementacije, klase zrna obeležavaju se anotacijom koja označava tip zrna (sledeća tabela), a u datoteci u kojoj se zrno implementira potrebno je uključiti odgovarajuću klasu koja tu vrstu zrna opisuje.

vrsta zrna	Java anotacija	uključivanje odgovarajuće klase
zrno bez stanja	*@Stateless*	*import javax.ejb.Stateless*
zrno stanja	*@Stateful*	*import javax.ejb.Stateful*
zrno jedinac	*@Singleton*	*import javax.ejb.Singleton*
zrno vođeno porukom	*@MessageDriven*	*import javax.ejb.MessageDriven*

Slika 4.3 Anotacije vrste zrna [izvor: autor]

Takođe, posebnim anotacijama se obeležavaju navedeni interfejsi.

tip interfejsa	Java anotacija	Uključivanje odgovarajućeg interfejsa
lokalni interfejs	*@Local*	*Import javax.ejb.local*
udaljeni interfejs	*@Remote*	*Import.javax.ejb.remote*

Slika 4.4 Anotacije za interfejse [izvor: autor]

U narednom izlaganju, biće demonstrirana primena navedenih anotacija kroz nekoliko linija koda. Ako se koristi neko zrno bez stanja, moguće ga je deklarisati na sledeći način:

```
@Stateless
public class NekoZrno implements LokalniInterfejs{...}
```

Deklaracija interfejsa može biti sledeća:

```
@Local
public interface LokalniInterfejs{...}
```

PRIMENA ZRNA SESIJE I ODGOVARAJUĆEG INTERFEJSA

Za korišćenje instance zrna sesije ili njegovog interfejsa, neophodna je referenca na tu instancu.

Da bi korisnik mogao da koristi instancu zrna sesije ili njegovog interfejsa, prvo mora da napravi referencu na tu instancu. To se može uraditi prekoumetanja zavisnosti (dependence injection) ili koristeći Java Naming and Directory Interface.
Java Naming and Directory Interface (JNDI) je deo Java platforme koji aplikacijama zasnovanim na Java tehnologiji obezbeđuje jedinstven interfejs za višestruko imenovanje i servise za rad sa direktorijumima. JNDI API pruža aplikacijama mogućnost pretrage objekta koristeći atribute objekta i njegovo preuzimanje i skladištenje.

Umetanje zavisnosti je funkcionalnost Java EE platforme koja omogućava olakšano korišćenje Enterprise Java zrna. Mogućnost bezbednog ubrizgavanja zavisnosti i izbora implementacije

interfejsa u fazi raspoređivanja aplikacije na server, naročito je značajna zbog zbližavanja Veb sloja aplikacije sa funkcionalnostima koje pruža kontejner Enterprise Java zrna, kao što su kontrola transakcija.

Jednostavnije i elegantnije je korišćenje umetanja zavisnosti. Da bi referencu bila kreirana umetanjem zavisnosti, moguće je koristiti anotaciju ***@EJB.*** Za primer NekoZrno, na mestu gde koriste metode lokalnog interfejsa, neophodno je ubaciti sledeću anotaciju:

```
@EJB
LokalniInterfejs lokalniInterfejs;
```

```
@EJB
LokalniInterfejs lokalniInterfejs;
```

Instanca lokalniInterfejs sada je spremna za poziv metode interfejsa.

Slično, ako zrno ima javne metode, može im se pristupiti direktno ubacivanjem sledeće anotacije:

```
@EJB
NekoZrno nekoZrno;
```

Instanca NekoZrno sada je spremna za poziv javne metode.

Neophodno je pomenuti i udaljenog korisnika koji ima sledeće karakteristike:

- može se nalaziti na drugoj mašini ili drugoj Java virtuelnoj mašini;
- može biti Veb komponenta, klijent-aplikacija (application client), ili drugo zrno;
- zrno kojem pristupa treba da ima udaljeni interfejs;
- Udaljeni interfejsi obeležavaju se anotacijom ***@Remote;***
- Udaljeni interfejsi referenciraju se pomoćuumetanja zavisnosti na isti način kao i lokalni interfejsi - anotacijom ***@EJB.***

Aplikacije koje se nalaze van Java EE server okruženja moraju koristiti JNDI za referenciranje udaljenog interfejsa. Na primer:

```
UdaljeniInterfejs udaljeniInterfejs = (UdaljeniInterfejs)
InitalContext.lookup(“java:global/ime_aplikacije/UdaljeniInterfejs”);
```

ZRNA VOĐENA PORUKOM

Neophodno je obraditi još jedan tip Java EE zrna.

Zrna vođena porukom omogućavaju Java EE aplikacijama da obrađuju poruke asihrono. Kod sinhronih operacija dolazi do blokiranja procesa sve dok se pozvana operacija ne završi, dok kod asihronih operacija po pozivanju nema blokiranja, nego se proces nastavlja. Zrna vođena porukom obično funkcionišu kao osluškivač Java Message Service (JMS) poruka. Java Message Service API je standard za slanje poruka koji pruža mogućnost komponentama Java EE aplikacija da kreiraju, šalju i primaju poruke, omogućavajući komunikaciju koja je

pouzdana i asinhrona. Dakle, zrna vođena porukom osluškuju i primaju JMS poruke. Poruka može biti poslata od strane bilo koje Java EE komponente, npr. od strane drugog Enterpise zrna. Po primanju JMS poruke, zrna vođena porukom preduzimaju određene akcije, obično obrađujući primljenu poruku.

Među karakteristikama zrna vođenih porukom, neophodno je istaći sledeće:

- izvršavaju se po prijemu poruke;
- pozivaju se asinhrono;
- ne predstavljaju podatke u bazi, ali se mogu koristiti za menjanje podataka baze;
- nemaju stanje, tj. ne pamte podatke o klijentu;

Komponente koje pozivaju zrno vođeno porukom ga ne lociraju, već koristeći JMS šalju poruku na lokaciju za koju zrno vođenom porukom ima osluškivač. Takva lokacija se može izabrati na serveru za svako zrno vođeno porukom. Kada poruka stigne, kontejner poziva metodu zrna onMessage() da obradi poruku. Poruka se dalje može obraditi u zrnu u skladu sa poslovnom logikom, može se pozvati neko zrno sesije da obradi podatke iz poruke ili da ih sačuva u bazu.

Moguće je uočiti nekoliko sličnosti između zrna vođenih porukom i zrna bez stanja:

- instanca zrna vođenog porukom ne pamti stanje klijenta, tj. podatke o klijentu;
- Sve instance zrna vođenih porukom u kontejneru su ekvivalentne;
- Jedna instanca zrna vođenih porukom može opsluživati više klijenata, jednog po jednog;

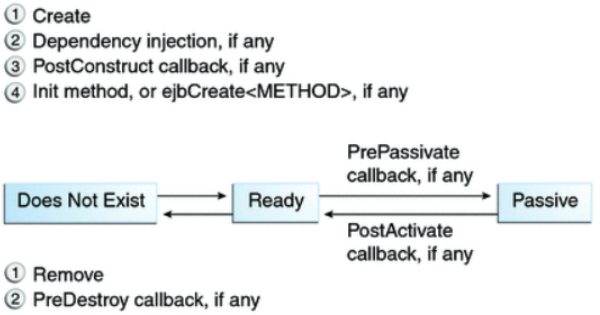
> Pomoću zrna sesije takođe je moguće slati JMS porkuke, ali nije moguće da ih primati ih asihrono. Za asihrono primanje poruka, neophodno je odlučiti se za zrna vođena porukom.

ŽIVOTNI CIKLUS ENTERPISE JAVA ZRNA ZA ZRNA SA I BEZ STANJA

Životni ciklus zrna je važna tema za ovaj deo izlaganja.

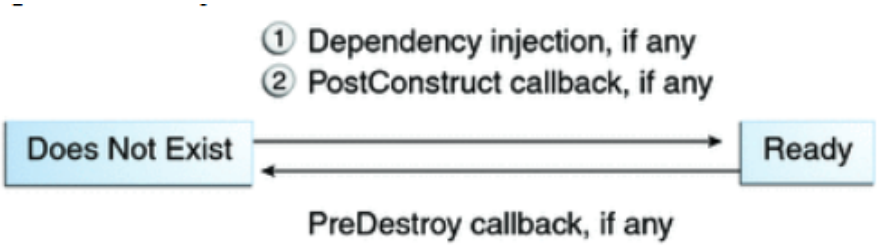
Prvo će biti diskutovano o zrnima stanja. Referenciranjem zrna stanja inicira se umetanje eventualnih zavisnosti ako ih ima. Ubrizgavanje se vrši od strane kontejnera, a zatim se poziva metoda obeležena anotacijom @PostConstruct., ako takva metoda postoji. Po izvršavanju init(), ili ejbCreate(), metode, ako postoji, instanca zrna postaje spremna za izvršavanje metoda ukoliko budu pozvane od strane klijenta.

Dok se nalazi u fazi Spremno, kontejner može odlučiti da izmesti instancu zrna iz radne memorije stavljajući je u pasivno stanje. Pri tome, prilikom izbora instance za stavljanje u pasivno stanje, kontejner obično bira onu koja najduže nije bila korišćena. Tom prilikom, ako postoji, poziva se metoda obeležena sa @PrePassivate. Ako klijent pozove neku od metoda zrna dok je njegova instanca u pasivnom stanju, kontejner ponovo aktivira instancu zrna, poziva metodu označenu sa @PostActivate, ako ona postoji i instanca je ponovo u fazi Spremno.



Slika 4.5 Životni ciklus zrna stanja [izvor: autor]

Životni ciklus zrna bez stanja ima samo dva stanja, instanca ili postoji i spremna je za pozivanje metoda zrna ili ne postoji. Kontejner zrna pravi i održava kolekciju instanci zrna bez stanja, ubrizgava zavisnosti i poziva metodu označenu sa @PostConstruct, ukoliko ona postoji. Kada se to obavi, instanca zrna je u fazi Spremno. Na kraju životnog ciklusa, kontejner poziva metodu označenu sa @PreDestroy ukoliko ona postoji i tu se život instance zrna bez stanja završava.

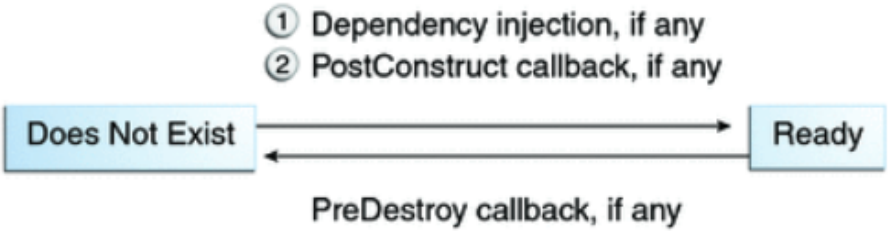


Slika 4.6 Životni ciklus zrna bez stanja [izvor: autor]

ŽIVOTNI CIKLUS ENTERPISE JAVA ZRNA ZA SINGLETONE I MDB ZRNA

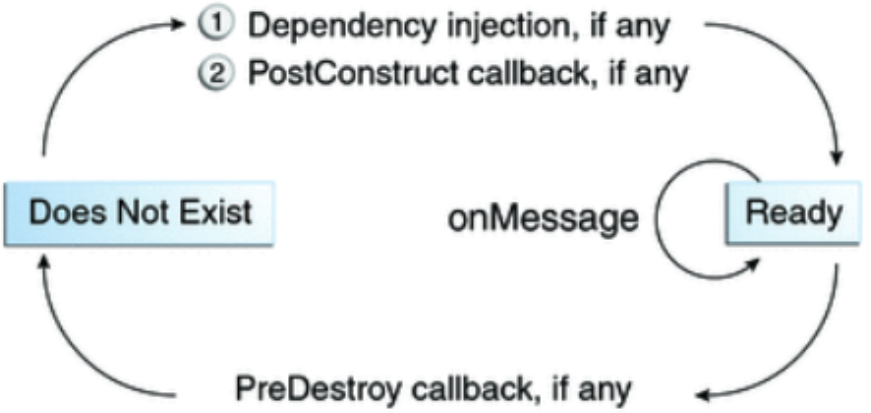
Za preostala dva tipa Java EE zrna, životni ciklus je, takođe, veoma važna tema.

Isto kao kod zrna bez stanja, zrno jedinac ima samo dve faze svog životnog ciklusa. Instanca ovog zrna ili ne postoji ili postoji i spremna je za pozive metoda zrna. Kontejner zrna započinje životni ciklus zrna jedinca praveći instancu ovog zrna. Ukoliko je zrno jedinac označeno deklaracijom @Startup, pravljenje ove instance događa se prilikom raspoređivanja(deployment) aplikacije na serveru. Po umetanju eventualnih zavisnosti i pozivu metode označene sa @PostConstruct od strane kontejnera, instanca prelazi u fazu Spremno. Životni ciklus zrna jedinca završava se kontejnerovim pozivom metode označene sa @PreDestroy, ukoliko ona postoji i to označava kraj instance ovog zrna.



Slika 4.7 Životni ciklus zrna jedinca [izvor: autor]

Na kraju, akcenat je stavljen na analizu životnog ciklusa zrna vođenih porukama. Kontejner zrna pravi kolekciju instanci zrna vođenih porukom. Za svaku od tih instanci kontejner obavlja zadatke umetanja zavisnosti i poziva metodu označenu sa @PostConstruct, ukoliko ona postoji. Time zrno prelazi u fazu Spremno.
Instanca zrna vođenog porukom, koja se nalazi u fazi Spremno, služi da nad njom bude pozvana metoda onMessage(). Pozivanjem ove metode instanca zrna opravdava svrhu svog postojanja.
Kao i kod zrna sesije bez stanja, na kraju životnog ciklusa kontejner poziva metodu označenu sa @PreDestroy i život instance zrna se završava.



Slika 4.8 Životni ciklus zrna vođenog porukom [izvor: autor]

NOVITETI U JAVA EE 8

Java Enterprise Edition 8 donosi nova unapređenja u svet veb i distribuiranih aplikacija.

Java Enterprise Edition 8 donosi nova unapređenja u svet veb i distribuiranih aplikacija, a najznačajnija su u sledećim oblastima:

- podrška za računarstvo u oblaku (*Cloud Computing*);
- *HTML5*;
- *HTTP/2*.

Konkretno gledano, sa Javom EE 8 dolazi do unapređenja postojećih i uvođenja novih specifikacija koje čine koncept koji danas nazivamo Java tehnologijama. Slede najznačajnije:

- *JSON-B (JavaScript Object Notation Binding)* - obezbeđuje povezujući nivo za konverziju Java objekata u / iz JSON poruka;
- Unapređen *JSON-P (JSON Processing API)* - unapređenja u objektnom modelu;
- *JAX-RS (Java API for RESTful Web Services) 2.1* reaktivni klijent API.
- *JAX-RS podrška za događaje tipa server-sent*: implementacija jednosmernog kanala od klijenta ka serveru;

- *HTTP/2* Java Servlet podrška - Java Servlet obezbeđuje programsku klasu za proširenje serverskih mogućnosti.
- *Java EE Security API* - podrška za Cloud i PaaS paradigme.
- *Bean Validation 2.0* - Omogućava iskazivanje ograničenja objektnog modela putem anotacija.
- *Java Server Faces 2.3* - za građenje "server-side" korisničkih interfejsa.
- *CDI (Contexts and Dependency Injection) 2.0* - za isticanje asinhronih događaja.

Slika 4.9 Java EE 8 - karakteristike u slici [izvor: Oracle]

VIDEO MATERIJAL

Java EE 8: What's New in the Java EE 8 Release - video materijal

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

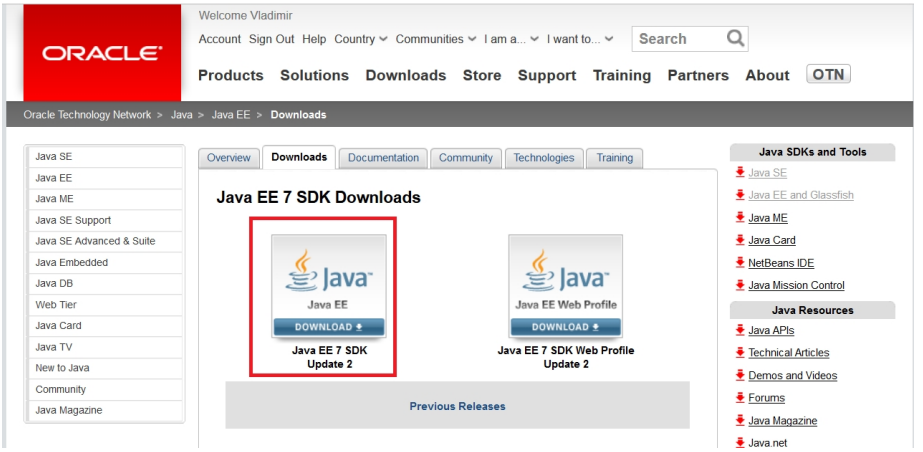
Podešavanje okruženja za razvoj JEE programa

ZADATAK 1 (25 MIN)

Preuzimanje i instaliranje neophodnih razvojnih alata i tehnologija.

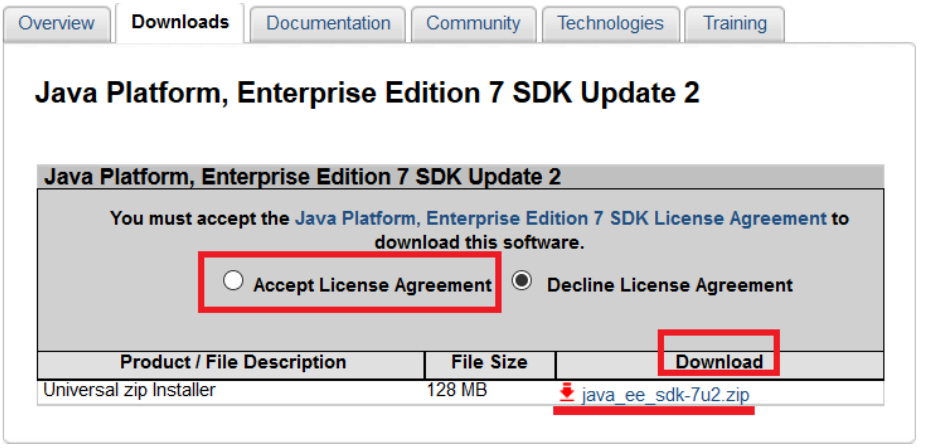
1. Instalirati Java EE poslednju JDK verziju i podesiti sistemske varijeble;
2. Instalirati razvojno okruženje NetBeans IDE - poslednju dostupnu verziju;
3. Instalirati GlassFish aplikativni server;
4. Instalirati MySQL server

Za početak je potrebno otići na link http://www.oracle.com/technetwork/java/javaee/downloads/index.html i izabrati preuzimanje Java EE 7 JDK instalacionog paketa.



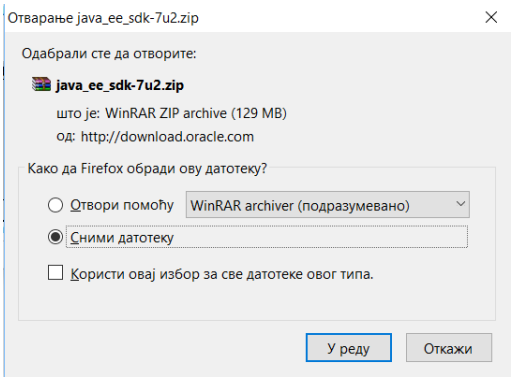
Slika 5.1 Stranica za preuzimanje Java EE 7 JDK instalacionog paketa. (izvor: autor)

Klikom na obeleženu sliku otvara se prozor u kojem se zahteva prihvatanje uslova korišćenja. Kada se uslovi prihvate vrši se klik na link za download.



Slika 5.2 Prihvatanje uslova korišćenja [izvor: autor]

Klikom na odgovarajući link preuzima se Java EE 7 JDK ZIP datoteka.

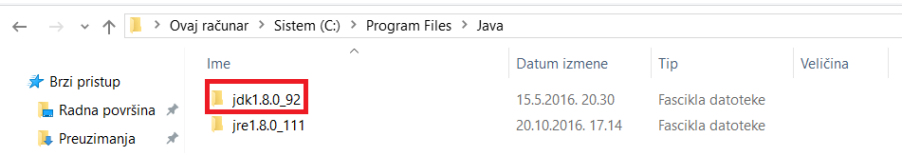


Slika 5.3 Preuzimanje Java EE 7 JDK instalacionog paketa [izvor: autor]

ISTALIRANJE JAVE I PODEŠAVANJE SISTEMSKIH VARIJABLI.

Istaliranje Jave i podešavanje sistemskih varijabli na lokalnom računaru.

JDK datoteka je preuzeta i sačuvana u folderu Downloads. U sledećem koraku je neophodno je izvršiti njeno raspakivanje i neka to bude na lokaciji: C:\Program Files\Java kao na sledećoj slici:

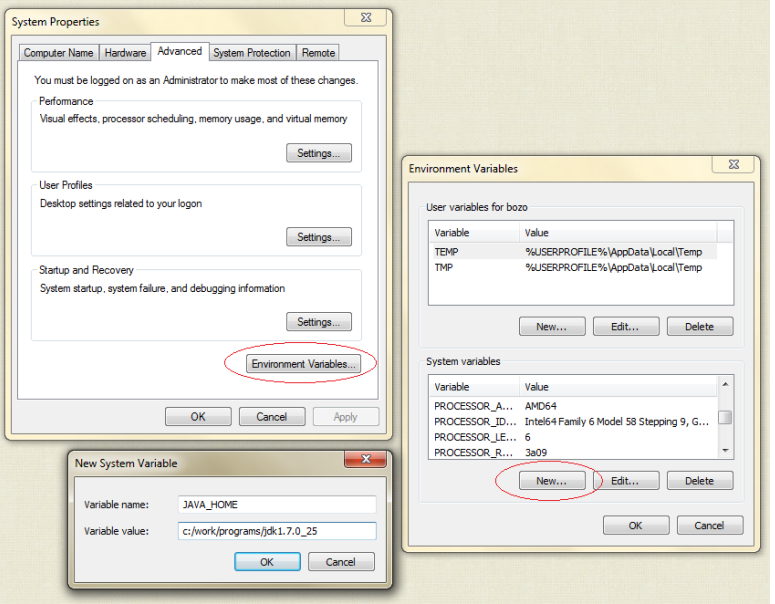


Slika 5.4 JDK je raspakovan [izvor: autor]

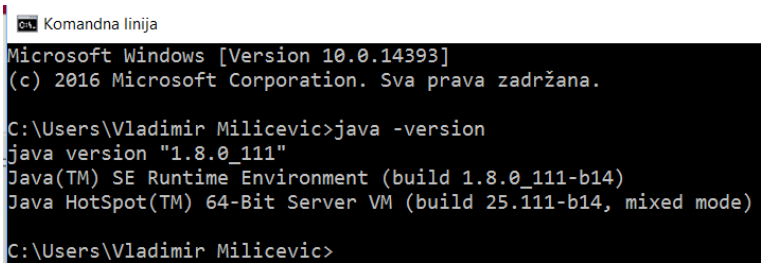
Sada je neophodno učiniti Javu "vidljivom" sa bilo koje lokacije na lokalnom računaru. To se omogućava podešavanjem sistemskih varijabli. Desnim klikomna My Computer, zatim Properties, pa Advanced System Settings i Enviroment Variables, otvara se prozor u kojem će

biti izvršena navedena podešavanja. Sada se definiše jedna sistemska promenljiva po imenu JAVA_HOME čija će vrednost biti putanja do JDK foldera. Nakon toga iz liste postojećih varijabli, bira se varijabla PATH u okviru koje je neophodno dodati sledeći string ;%JAVA_HOME%/bin kojim je određena putanja do JAVA prevodioca. Navedeno je prezentovano slikom 5.

Nakon podešavanja sistemskih varijabli, neophodno je proveriti da li je sve obavljeno kako treba. Otvaranjem MS DOS Command Prompt i kucanjem instrukcijejava –version vrši se provera. Ukoliko se dobije informacija kao na slici 6, posao instalacije Jave je uspešno završen.



Slika 5.5 Podešavanje sistemske varijable za Javu [izvor: autor]

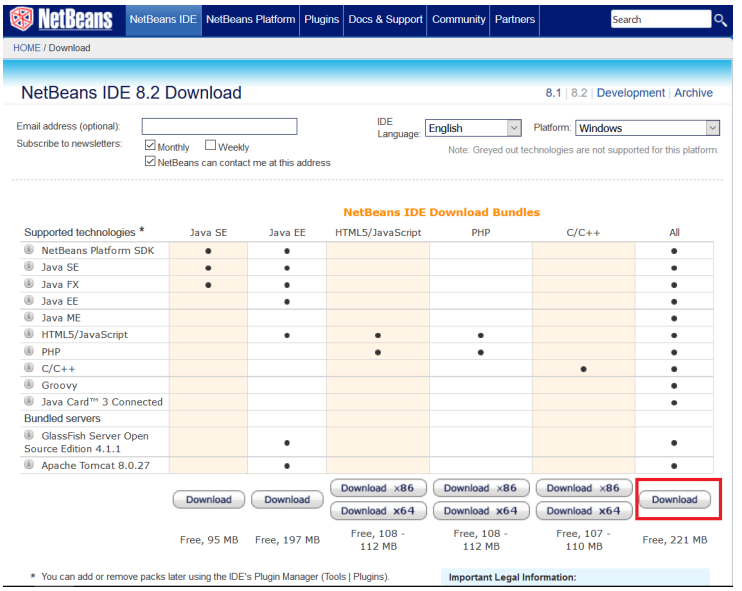


Slika 5.6 Provera instalacije [izvor: autor]

ISTALIRANJE RAZVOJNOG OKRUŽENJA

Preuzimanje razvojnog okruženja za razvoj Java veb aplikacija.

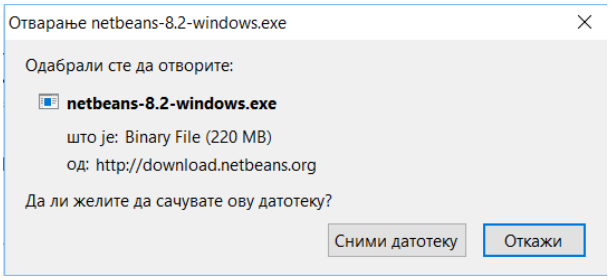
U nastavku je neophodno otići na link http://netbeans.org/downloads/index.html i preuzeti instalacionu datoteku za NetBeans IDE razvojno okruženje.



Slika 5.7 Stranica za preuzimanje instalacione datoteke [izvor: autor]

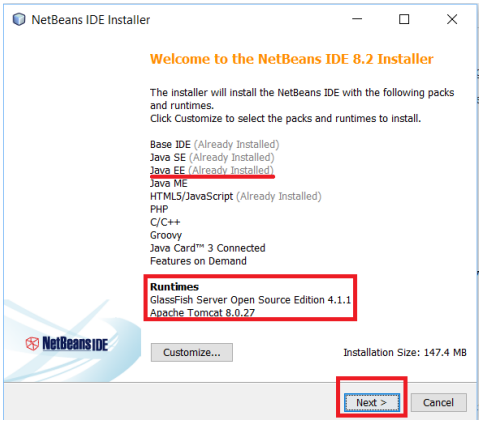
Sa slike je moguće primetiti da je Javu EE bilo moguće preuzeti zajedno, u paketu sa NetBeans.

Takođe, u ovom instalacionom paketu je moguće preuzeti i aplikacioni server GlassFish koji će u radu na razvoju veb Java aplikacija biti neizostavan u daljem radu.



Slika 5.8 Preuzimanje instalera [izvor: autor]

Pokretanjem instalera, bira se lokacija gde će razvojno okruženje biti instalirano, takođe i svi potrebni alati - GlassFish i Apache server. Praćenjem čarobnjaka za instalaciju, ceo proces se jednostavno i automatski obavlja.



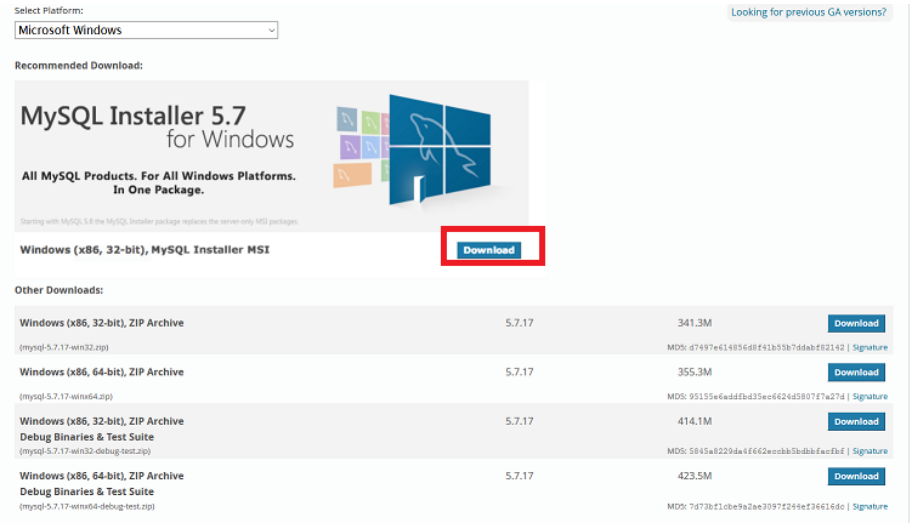
Slika 5.9 Izbor tehnologija i servera za instalaciju [izvor: autor]

ISTALIRANJE SERVERA BAZE PODATAKA

U narednom radu će biti korišćene i baze podataka pa je neophodno instalirati i odgovarajuću podršku.

U narednom radu će biti korišćene i baze podataka pa je neophodno instalirati i odgovarajuću podršku. Ovde će za to poslužiti MySQL server. Instalaciju je moguće preuzeti sa linka http://dev.mysql.com/downloads/.

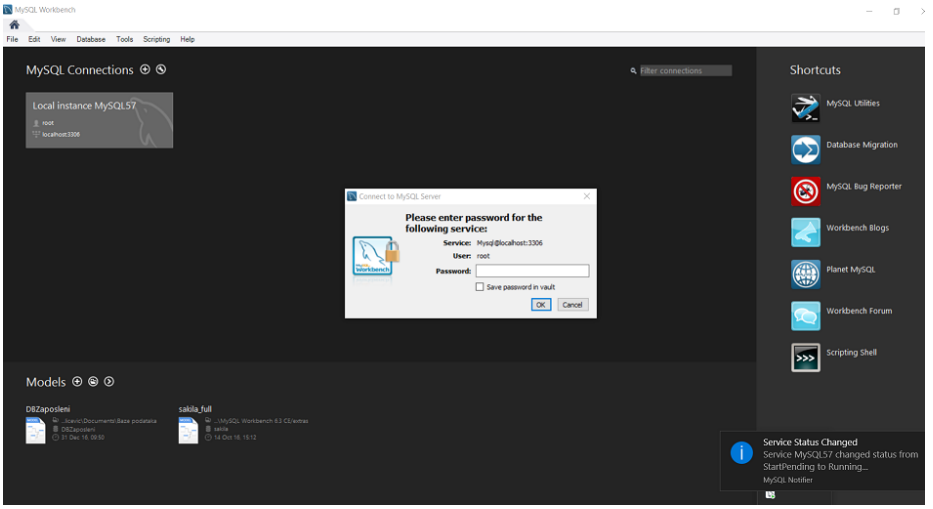
Izborom preuzimanja MySQL Installer 5.7 i njegovim pokretanjem sva neophodna MySQL podrška biće automatski instalirana ako se prate uputstva za instalaciju iz odgovarajućeg instalacionog čarobnjaka.



Slika 5.10 Istaliranje servera baze podataka [izvor: autor]

Bez obzira što je instaler 32 - bitni, on podržava instalaciju kao 32 - bitnih, tako i 64 - bitnih datoteka MySQL servera.

Nakon obavljene instalacije moguće je koristiti ovaj server za upravljanje bazama podataka (sledeća slika). Sada je kompletirana podrška za razvoj Java EE veb aplikacija.



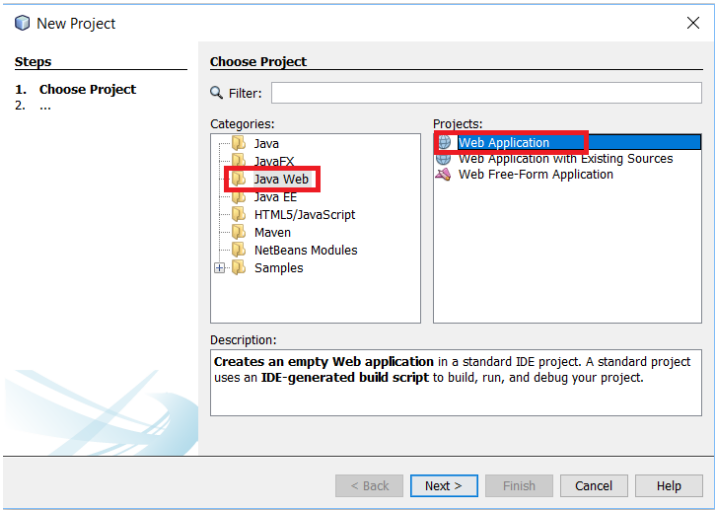
Slika 5.11 Instaliran MySQL server [izvor: autor]

ZADATAK 2 (20 MIN)

Provera instalirane podrške za razvoj veb aplikacija

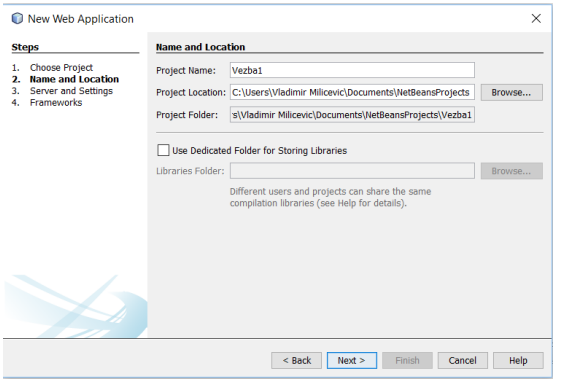
1. Kreirati nov Java veb projekat pod nazivom Primer1;
2. Modifikovati osnovnu indeks.html stranicu da ukazuje da se radi o našoj prvoj veb aplikaciji;
3. Pokrenuti kreiranu aplikaciju.

Pokreće se NetBeans razvojno okruženje i u meniju File bira se opcija New Project. Otvoriće se nov prozor u kojem je neophodno izabrati vrstu Java projekta. Ovde se sada bira kategorija Java Web i vrsta aplikacije Web Application (sledeća slika).



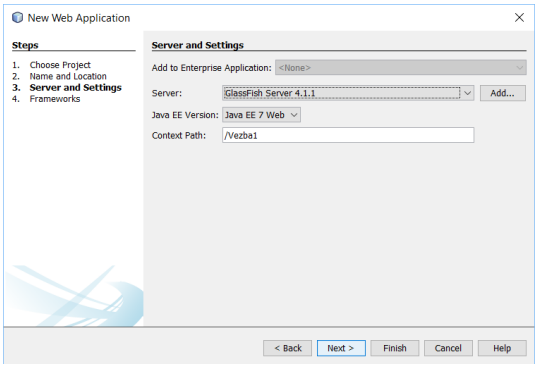
Slika 5.12 Kreiranje Java veb projekta [izvor: autor]

U nastavku se, u novom prozoru, zadaje naziv projekta - Vezba1.



Slika 5.13 Naziv projekta [izvor: autor]

Klikom na Next, o novom prozoru se bira aplikativni server i dodatna podešavanja. Klikom na Finish kreiran je prvi veb Java projekat.

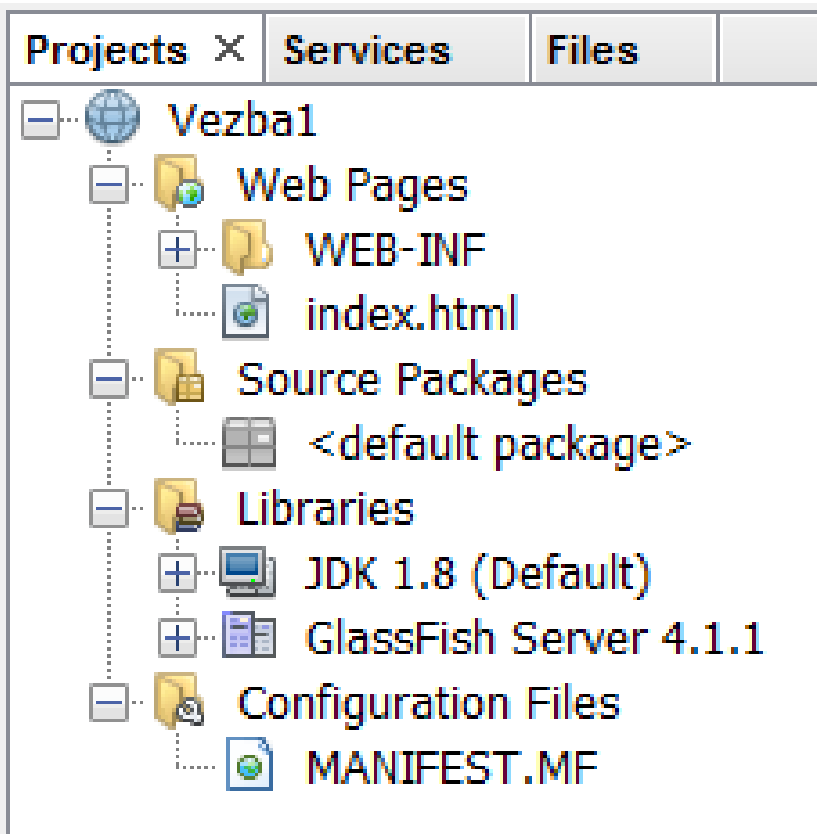


Slika 5.14 Poslednja podešavanja [izvor: autor]

MODIFIKACIJA PODRAZUMEVANE APLIKACIJE I TESTIRANJE

Sada je moguće testirati instaliranu podršku.

Kreiran je projekat Vezba1 i njegova hijerarhija je prikazana sledećom slikom.

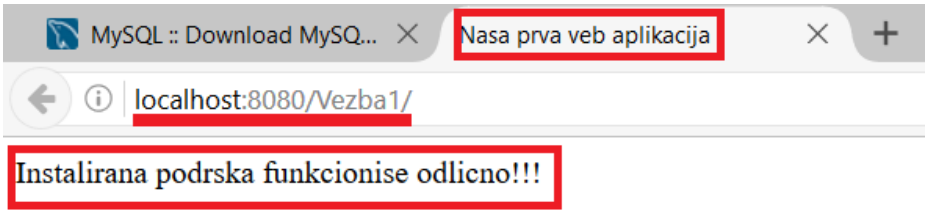


Slika 5.15 Folderi kreiranog projekta [izvor: autor]

Od posebnog je značaja datoteka index.html koju je moguće modifikovati na sledeći način:

```
<html>
    <head>
        <title>Nasa prva veb aplikacija</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>Instalirana podrska funkcionise odlicno!!!</div>
    </body>
</html>
```

Sada je tu najjednostavnija veb aplikacija, koju je moguće pokrenuti i testirati da li je prethodni posao uspešno obavljen.



Slika 5.16 Naša prva veb aplikacija [izvor: autor]

Zaključak: Sve funkcioniše i sve je spremno za razvoj složenijih Java veb aplikacija.

Poglavlje 6

Individualna vežba 1

INDIVIDUALNA VEŽBA (135 MIN)

Pokušajte sami

Kreirajte samostalno prvu veb aplikaciju koristeći znanja stečena na predavanjima;

1. Napravite korekcije u index.html stranici:
2. Dodajte nove stringove;
3. Pokušajte, koristeći Internet literaturu da dodate neke GUI kontrole.
4. Pokažite strukturu vlastite Java EE veb aplikacije;
5. Izvršite prevođenje i testiranje rezultata izvršavanja.
6. Dokumentujte rezultate i konsultujte se sa predmetnim asistentom

Koristan link za individualnu vežbu: https://www.w3schools.com/html/html_form_elements.asp

Poglavlje 7

Domaći zadatak 1

ZADATAK 1

Podešavanje vlastitog računara za razvoj veb aplikacija i testranje jednostavne apliakcije.

1. Instalirati i podesiti na vlastitom računaru Java EE JDK 7;
2. Instalirati razvojno okruženje NetBeans (može i drugo po izboru);
3. Instalirati aplikativni server;
4. Instalirati MySQL server;
5. Kreirati jednostavnu Java veb aplikaciju u razvojnom okruženju i testirati je na GlassFish aplikativnom serveru.
6. Obavljene zadatke dokumentovati u PDF ili DOC dokumentu.

Nakon urađenog obaveznog zadatka, studenti na mail od predmetnog asistenta dobijaju različite zadatke.

Poglavlje 8

Zaključak

ZAKLJUČAK

Lekcija je povezala distribuirane sisteme sa tehnologijom i alatima za njihovo razvijanje

U ovoj lekciji je cilj bio približavanje studentima koncepta distribuiranog sistema i predstavljanje alata i tehnologija za njihovo razvijanje.

Upravo iz navedenog razloga uvodne teme se bave predstavljanjem distribuiranih sistema i njihovih ključnih karakteristika: deljenje resursa, otvorenost, istovremenost, skalabilnost, otpornost na greške i transparentnost. Nakon toga, akcenat je stavljen na oblikovanje distribuiranih sistema.

Posebno je vođeno računa o tome da se da opravdanje za korišćenje ovakvih tipova informacionih sistema uz demonstriranje njihovih prednosti i nedostataka. Posebno su poređeni sa sistemima sa centralizovanom arhitekturom, a na kraju ove teme su navedeni primeri upotrebe distribuiranih sistema.

Lekcija se u posebnom delu bavila Java tehnologijama i alatima za podršku razvoja distribuiranih sistema.

Savladavanjem ove lekcije student razume koncept distribuiranog sistema i sposoban je da podesi razvojno okruženje za njihov razvoj

LITERATURA

Za pripremu materijala L01 korišćena je savremena pisana i elektronska literatura.

1. http://docs.oracle.com/javaee/6/tutorial/doc/giplj.html
2. Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Kim Haase, William Markito. 2014. Java Platform, Enterprise Edition The Java EE Tutorial, Release 7, ORACLE
3. David R. HeffelFinger. 2015. Java EE7 Develpomnet With NetBeans 8, PACK Publishing
4. Yakov Fain. 2015. Java 8 programiranje, Kombib (Wiley)
5. Josh JUneau. 2015. Java EE7 Recipes, Apress
6. Elazar FIlip. 2011. PRIMENA KOMPONENTNOG PROGRAMIRANJA ZASNOVANOG NA ENTERPRISE JAVA ZRNIMA U EDUKATIVNE SVRHE, Matematički fakultet u Beogradu