5 – Programska podrška Klijenta

S A D R Ž A J

5.1 Programska podrška klijenta

- Sistemi zasnovani na distribuiranim objektima, zbog svojih brojnih prednosti razvijaju se već duži niz godina, od 1990 godine.
- Na tržištu postoji veliki broj različitih platformi (tehnologija) za implementaciju višeslojne distribuirane arhitekture.
- Pri izboru distribuirane tehnologije treba obratiti pažnju na:
  - ✓ **prenosivost (portabilnost) klijentske i serverske platforme**
  - ✓ **prenosivost programskog jezika**
  - ✓ **performanse pri izvršavanju**
  - ✓ **jednostavnost razvoja**
  - ✓ **sigurnost**
- Rezultati testiranja sa različitim komunikacionim modelima za udaljeno pozivanje komponenata do danas su rezultirali sa nekoliko najpoznatijih tehnologija:
  - ❑ **COM, COM+, DCOM**
  - ❑ **JAVA, EJB**
  - ❑ **COBRA**

5.2 Distributed interNet Aplication architecture

- **COM** je specifikacija za objekte koja definiše interfejs preko koga različiti objekti mogu da komuniciraju.
- Nezavisan je od progr.jezika ukoliko on implementira COM interfejs
- Teoretski može da se implementira na različitim OS, međutim nijedan drugi OS nije prihvatio COM objekte osim Microsoft Windows-a.
- Da bi se omogućilo da COM objekti sa različitih sistema međusobno razmenjuju informacije, COM specifikacija je proširena u **DCOM**
- **DCOM** poseduje znatno kompleksniji model konfiguracije i sigurnosti
- Za potpuno razumevanje COM+ komponentnog modela prvo je potrebno razjasniti Windows DNA
- Windows DNA je platforma koja opisuje kako razvijati višeslojne, skalabilne distribuirane aplikacije visokih performansi za rad u mreži.
- Osnovni cilj DNA je omogućavanje izrade ***enterprise level*** rešenja.
- Srce DNA je integrisani program.model baziran na COM+ i sastoji se:
  - ***Prezentacionog sloja***
  - ***Sloja poslovne logike***
  - ***Sloja za pristup podacima***

5.2 Distributed interNet Aplication architecture

1. ***Prezentacioni sloj*** - odgovoran je za prikupljanje informacija od korisnika, kontrolu unetih podataka, njihovo slanje sloju poslovne logike, primanje rezultata od sloja poslovne logike i njihove prezentacije korisniku - VB, HTML, DHTML, java apleti, ActiveX kontrole
2. ***Sloj poslovne logike*** - vrši iterakciju sa slojem za pristup podacima radi njihove obrade, slanje dobijenih informacija prezentacionom sloju i obezbeđuje pravila i servise koji pomažu kod pisanja skalabilnih aplikacija a koji su čvrsto integrisani jedan sa drugim kao i sa OS:
   1. **Web servise** - Microsoft **I**nternet **I**nformation **S**erver-a (IIS)
   2. **Transakcionog i servisa komponenti** -**M**icrosoft **T**ransaction **S**erver
   3. **Asihronih i servisa redova** - **M**icrosoft **M**essage **Q**ueue **S**erver-a
   4. **Serverskog skriptovanja** - **A**ctive **S**erver **P**ages (ASP).
3. ***Sloj za pristup podacima*** - direktno interaguje sa podacima koji obično egzistiraju u bazi podataka. Odgovoran je za smeštanje, pronalaženje i održavanje podataka kao i njihovog integrititeta. Pristup podacima preko Windows DNA naziva se ***Universal Data ACCess*** (**UDA**). **UDA** je skup modela sistemskog i aplikacionog nivoa zvanih **OLE-DB, ADO i RDO**.

5.3 Component Object Model - COM+

- Početak razvoja COM+ objekata datira od 1992 god. kada je Microsoft razvio OLE koji je kasnije (1995) nazvao ***Component Object Model***.
- 1996 COM počinje sa podrškom distribuiranog procesiranja - **DCOM**
- Istovremeno Microsoft razvija novi transakcioni server koji dobija naziv ***Microsoft Distributed Transaction Coordinator*** (**MTDC**)
- 1997 on je unapređen u ***Microsoft Transaction Server*** (**MTS**).
- Iste godine Microsoft razvija još jedan server koji dobija ime ***Microsoft Message Queue Server*** (**MSMQ**).
- 1999 Microsoft kombinuje sve do tada razvijene servise u integrisano *runtime* okruženje koje naziva **COM+**.

COM+ je integrisano okruženje koje programerima obezbeđuje pristup COM, MTS, MSMQ i drugim servisima.

- COM+ je proširena verzija COM-a sa dve glavne razlike: on uključuje proširenu i nadgrađenu verziju MTS-a, **MTS 3**, i četiri glavna servisa uključujući **MSMQ**, ***Load Balancing***, ***Event Services*** i **IMDB**.
- Sve ove komponente i servisi rade zajedno radi obezbeđivanja integrisanih ***enterprise*** (poslovnih) rešenja.

5.3 Component Object Model - COM+

COM+ = COM + MTS (nadgrađena verzija) + Servisi

- **COM** objekti,
- **MTS** (*Microsoft Transaction Server*)
- **MSMQ** (*Microsoft Message Queue Server*): služi da bi se podržao ***queue*** servis, koji omogučuje klijentu da može automatski izvršiti pozive metoda (osim ako komponenta nije *ofline*). **MSMQ** pozive metoda automatski snima i stavlja u redove čekanja uvek kada je komponenta dostupna. Ovaj servis je naročito koristan za *online* aplikacije koje se moraju izvršiti do kraja: *online* bankarstvo ili rezervacija avionskih karata
- ***Load Balancing*** - mehanizam za distribuciju klijentskih poziva ka više servera. COM+ omogućuje korišćenje dve vrste *load balancing*-a – **dinamičko** i **statičko**. COM+ podržava *load balancing* na nivou komponenti - ***Clustering service***.

5.3 Component Object Model - COM+

- ***Event Services*** - obrađuje događaje između dva objekta. U COM-u, događaji su se mogli obrađivati na dva načina: korišćenje povratnog mehanizma samog interfejsa, gde klijent implementira neki interfejs koji koristi komponenta i ***Connectable Objects*** koji koristi standardni COM IconnectionPoint interfejs gde jedan prosleđuje informacije (***Publisher***) a drugi ih prima (***Subscriber***).
- **IMDB** (*In-Memory Database*) - robustan, kratkotrajni i izvršni keš koji povećava performanse distribuiranih aplikacija). IMDB koristi OLE_DB i ADO i obezbeđuje brz pristup podacima koji se nalaze u bazama podataka. Umesto učitavnja u runtime modu, IMDB čuva bazu u kešu.
- ***Just-In-Time Activation*** - kada klijent prosledi zahtev za kreiranje instance objekta, COM+ obezbeđuje da klijent **referencira kontekst objekta** umesto da referencira sam objekat. Klijent dobija referencu na objekat tek kada pozove metodu tog objekta (rešava problem *overhead*).

5.3 Component Object Model - COM+

- ***Object pooling*** - je vrsta "recikliranja" objekata. Kada klijent završi sa radom sa instancom komponente, umesto uništavanja instance COM+ je "**reciklira**". Kada neki drugi klijent uputi zahtev za instancom iste komponente, COM+ vraća "recikliranu" instancu.

- ***Role-Based Security*** - predstavlja način da pridružite različita prava pristupa komponenti različitim korisnicima ili grupama korisnika. Prava pristupa se mogu određivati korišćenjem COM+ kataloga ili programskim putem (programiranjem).

Problemi koji su rešeni u COM+:

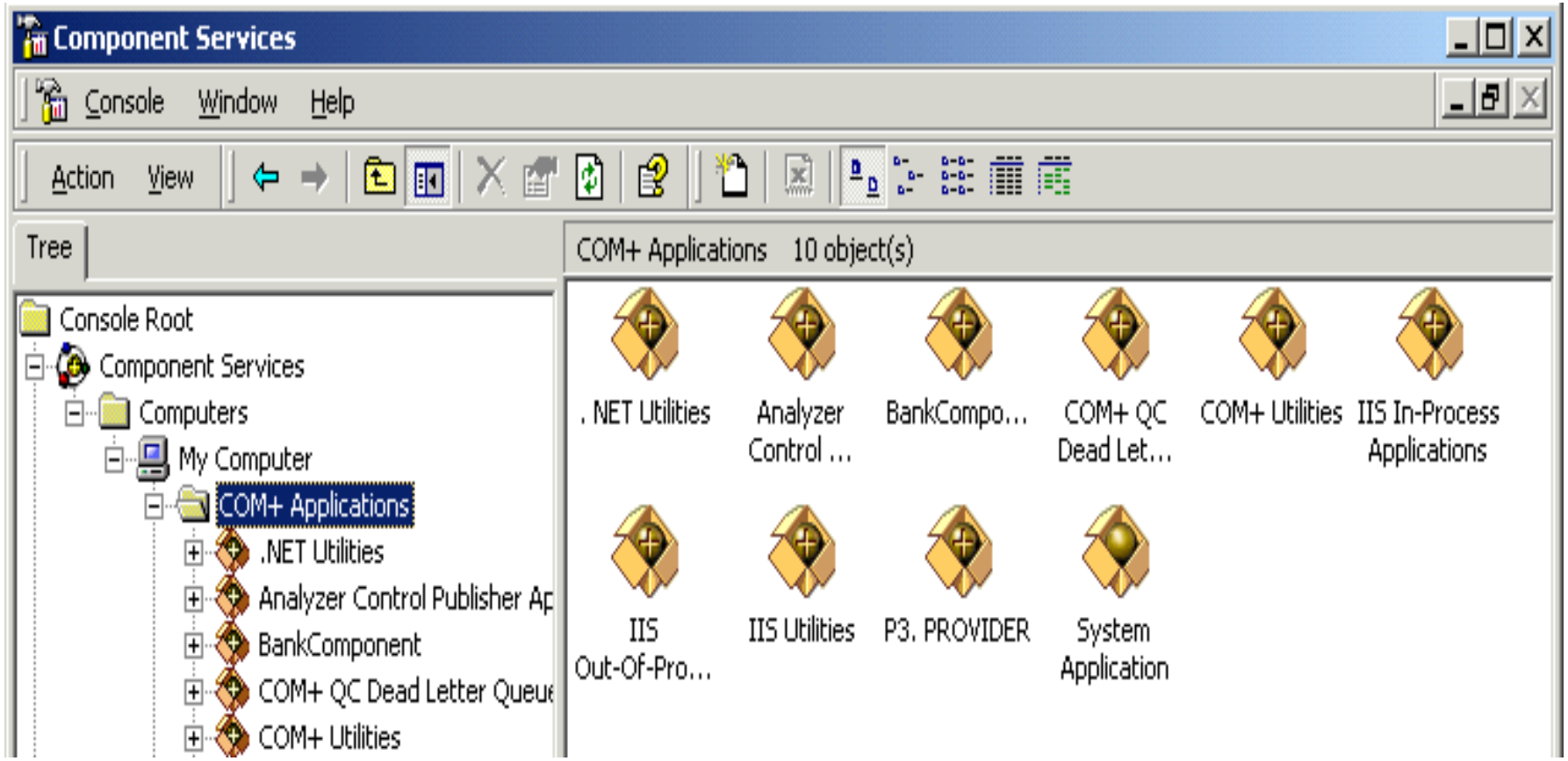
✓ *Publisher* i *Subscriber* su bili tesno povezani zbog potrebe za "poznavanjem" njihovih definicija interfejsa u vreme kompajliranja

✓ Dodavanje *Publisher*-u podrške za multi-kasting ili povećani broj izlaza zahtevalo je mnogo kodovanja.

✓ Arhitektura je samo opisivala set interfejsa a programeri su i dalje morali da implementiraju te interfejse.

✓ *Connection point* su kreirani bez podrške za distribuirana okruženja

5.3 COM+ katalog



5.4 Common Object Request Broker Architecture

- Velika raznolikost umreženih sistema kao što su različiti hardver, različiti OS, različita mrežna oprema kao i veliki broj mrežnih protokola doveo je do jako otežanog razvoja mrežnih aplikacija.
- **CORBA** je prvi, najvažniji i pre svega najambiciozniji *middleware* projekat pokrenut u istoriji namenjen da premosti ove razlike.
- Proizvod je konzorcijuma OMG (*Object Management Group*) koji broji preko 700 članova (pojedinci i velike softverske kompanije).
- CORBA je „papir“, tj. specifikacija na preko hiljadu strana.
- Prva specifikacija CORBA modela data je 1991 god.
- CORBA je bila veoma popularna sredinom devedesetih, ali sa pojavom Internet-a na koji je OMG sporo reagovao postaje deo istorije

Razlozi zbog kojih CORBA nije postala ono za šta je bila namenjena:

1. Skup razvoj aplikacija
2. CORBA je često suviše komplikovana
3. Mnoge implementacije su pune raznih propusta (bug-ovi, sigurnosne rupe, itd.)

5.4 Common Object Request Broker Architecture

- Glavni razlog polakog odustajanja od CORBA specifikacije je nedovoljno brzo praćenje zahteva tržišta.
- Tržištu je bio potreban otvoren standard koji bi omogućio komunikaciju između različitih aplikacija kroz otvoreni Internet.
- CORBA komunikacija koristi više portova i zato ne prolazi kroz zaštitne zidove (*firewall*).
- Bez obzira što su CORBA poruke bile binarne i iz tog razloga su se prenosile jako brzo, prva tehnologija koja je značajnije zapretila CORBA-i je bila **SOAP** (*Simple Object Access Protocol*), 1999 god.
- SOAP koristi XML poruke (tekstualne) koje se lako prenose Internet-om i čitljive su i za računare i za ljude.
- SOAP je brzo nadirao zbog lakoće upotrebe i lakog prolaženja kroz Internet (HTTP bez problema prolazi kroz većinu *firewall*-ova).
- Danas se CORBA jako malo koristi iako je ona nezavisna od jezika i implementirana je na većem broju platformi nego COM.
- Postoje velike nekompatibilnosti između implementacija različitih proizvođača što otežava njenu primenu.

5.4 Common Object Request Broker Architecture

- CORBA predstavlja *object bus* koji omogućava klijentu da poziva metode sa objekta na serveru uz nezavisnost program. jezika i lokacije
- Interakcija je omogućena preko **ORB** - *Object Request Brokers* komponenata na klijentu i na serveru (koji poseduju mehanizam RPC) a komunikacija se odvija preko **IIOP** (*Internet Inter-ORB Protocol*).
- Mogućnosti CORBA definisane su **IDL**-*Interface Definition Language*
- Ovaj standard definiše interfejs koji bi trebalo da omogući da komponente napisane u različitim jezicima međusobno sarađuju.
- To je ostvareno definisanjem seta metoda koje su vidljive za ostale komponente koje mogu da budu na različitim čvorovima u mreži.
- CORBA model u potpunosti podržava ideju troslojne arhitekture, koja po prirodi ima aplikaciju distribuiranu između klijenta, aplikativnog servera i servera baze podataka.
- ORB može biti implementiran kao proces na host-računaru kome klijent pristupa, kao proces na nekom centralnom računaru kroz koji klijent i server komuniciraju, ili kao servis samog operativnog sistema.
- IDL specifikacija je kompajlirana pomoću IDL kompajlera u svakom ciljnom jeziku i on treba da omogući impementaciju svih ovih metoda

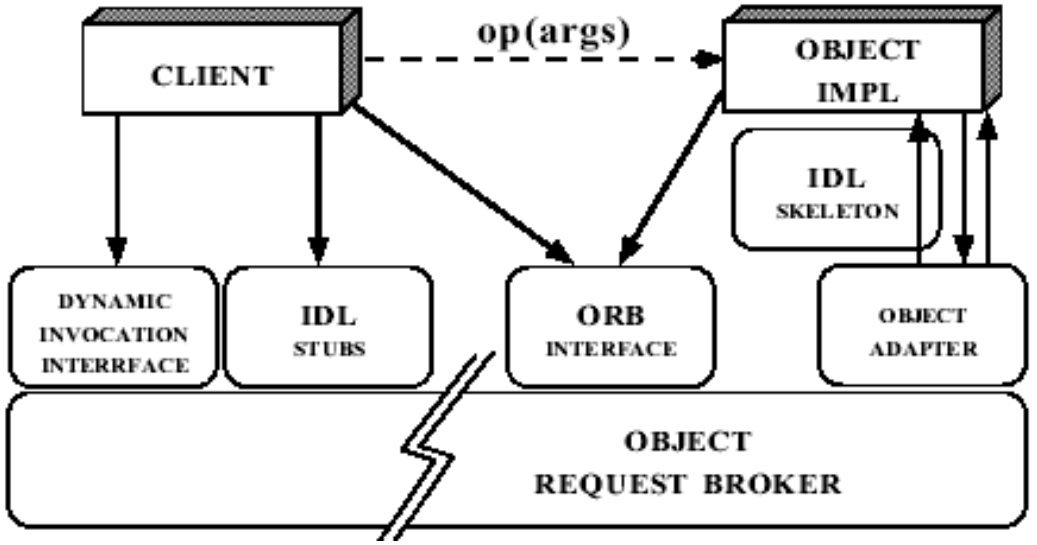
5.4 Primena CORBA modela

- CORBA je specifikacija koja teoretski dozvoljava da dva CORBA programa komuniciraju bez ikakvih promena iako su:
  - ✓ pisana na različitim programskim jezicima (C++ i Java)
  - ✓ izvršavaju se na različitim OS (Windows i UNIX)
  - ✓ koriste različite mrežne protokole (TCP/IP i Modbus)
  - ✓ izvršavaju se na različitom hardveru (Intel 32 bitni i Alpha CPU)
  - ✓ različite mrežne tehnologije (Ethernet i token ring mreže)
- Na ovako složene zahteve CORBA odgovara tako, što se „useli“ između OS i aplikacije i „sakrije“ složenost OS a samim tim i složenost mrežnih protokola, hardvera računara i mrežnog hardvera.
- Kod CORBA-e se podaci prenose binarno. Ovakav tip prenosa je puno brži od prenosa tekstualnih sadržaja (npr. XML poruke kod SOAP-a).
- CORBA danas jedino na tržištu *embedded* sistema ima porast tržišnog ucešća i primenjije se još kod:
  - rezervacije avio karata
  - komunikacija između *e-commerce* i finansijskim sistemima
  - telefonskim kompanijama (transakcije)

5.4 Primena CORBA modela

➢U poznatije CORBA specifikacije spadaju omniORB 2.6, TAO 3.0, ORBit2 2.4, MICO 3.0, IIOP.NET 2.3, Visibroker 2.6 i orbacus 2.6.

- ***Object impl*** – server.
- ***Client*** – klijent
- ***Object Request Broker*** – sloj u okviru OS koji sakriva svu složenost CORBA modela
- ***ORB Interface*** – API preko koga klijent i server pristupaju



- ***IDL Skeleton***-kod dobijen prevođenjem IDL-a. Mapiranje tipova (tzv. *marshaling*) i bazne klase koje serverski objekat treba da implementira
- ***IDL Stubs*** – poput IDL Skeleton-a, služi za mapiranje tipova
- ***Object adapter*** – povećava portabilnost serverske implementacije jer omogučava pisanje koda servera koji je nezavisan od korišćene CORBA implementacije. Postoje dve verzije *object adapter*-a: **BOA** (*Basic Object Adapter*) i **POA** (*Portable Object Adapter*)

5.5 Enterprice Java Beans - EJB

- Predstavlja distribuirani, transakcioni, serverski komponentni model.
- EJB nije proizvod već specifikacija koju izdaje Sun Microsys za Javu
- EJB je nezavisna od platforme ali ne i od jezika.
- Svi EJB objekti moraju biti napisani u jeziku Java.
- Za komunikaciju između različitih sistema, EJB koristi varijantu IIOP nazvanu **RMI** preko **IIOP** (***R**emote **M**ethod **I**nvocation over IIOP*).
- Kao što je RPC u klijent server modelu to je **RMI** u Java modelu
- Omogućava Java-Java komunikaciju, pozivanje metoda iz udaljenih Java aplikacija od strane Java apleta, Java IDL jezika (definišu se interfejsi po CORBA standardu) i preko ORB-a se pozivaju metode u programima napisanim u bilo kom programskom jeziku koji podržava CORBA standard i **JDBC** (*Java DB Connectivity*) koji predstavlja prvi standardizovan set objekata i metoda za interakciju sa bazama podataka i predstavlja sastavni deo **Java Entrprise API-a**.
- RMI mehanizam omogućava da pozovete metodu udaljenog objekta.
- RMI omogućije da se parametri metode proslede drugom računaru, obavesti server o metodi koju treba da izvrši i da vrati vrednost nazad.

5.5 Enterprice Java Beans - EJB

- U RMI terminologiji, objekat čija metoda pravi daljinski poziv se naziva klijentski objekat a daljinski objekat serverski objekat.
- Klijent/server terminologija odnosi se na samo jedan poziv metodi.
- Računar na kome se izvršava Java kod koji poziva udaljenu metodu je klijent, a računar koji sadrži objekat je server, samo za taj poziv.
- Sve ove tehnologije pružaju jaka sredstva za pisanje distribuiranih, višeslojnih/višekorisničkih klijent server aplikacija za pristup relacionim bazama podataka.
- Java programeri su dobili snažno oruđe koje im omogućava da primene koncept "**nezavisna platforma - nezavisna baza**".
- Java apleti i aplikacije sada mogu podjednako lako pristupati bilo kojoj bazi podataka na bilo kojoj platformi.
- JDBC predstavlja mnogo širi pojam od same baze podataka jer ona obuhvata i podatke kao što su oblik, lokacija ili organizacija podataka.
- Korišćenjem odgovarajućeg drajvera omogućeno je da se podaci koji se dobijaju iz klasične baze podataka ili iz audio-video signala sa satelita, potpuno isto tretiraju i obrađuju.

5.5 Enterprice Java Beans - EJB

- Za razliku od RMI tehnologije čija je implementacija sastavni deo Java jezika, EJB ne proširuje nijednu određenu implementaciju, već dozvoljava proizvođačima da sami prave sopstvene implementacije.
- Specifikacija je precizno definisana tako da se softver koji koristi EJB tehnologiju može instalirati u različita EJB okruženja bez modifikacije.
- Ovakav pristup omogućuje da se poslovna logika potrebna aplikaciji razvija bez potrebe da se brine o okruženju u kome će se ona izvršavati
- EJB specifikacija definiše model serverskih komponenti za razvoj višeslojnih arhitektura sa distribuiranim objektima.
- EJB definiše okruženje u kome žive EJB komponente.
- Arhitekturu čine EJB server i kontejner čije f-je nisu strogo razdvojene
- EJB server i kontejner treba da budu nezavisni jedan od drugog što bi omogućilo kombinovanje EJB komponenti različitih proizvođača.
- U praksi to nije moguće tako da svaki proizvođač nudi svoj par
- JDBC je otvorio velike mogučnosti komunikacije sa bazama podataka preko Intraneta i Interneta tako da se danas najviše traže programeri sa znanjem **COM+, JDBC, RMI i CORBA** tehnologija.

5.6 Simple Object Access Protocol

- Kreiran na postojećim, proverenim i široko prihvaćenim tehnologijama
- SOAP koristi XML za prenos podataka i izmenu aplikacija, a kako je XML standard, sve platforme mogu da pristupe i obrade informaciju.
- Pošto koristi HTTP, jednostavno prolazi kroz portove (*firewall*)
- Pristup različitim aplikacijama na raznim platformama sa SOAP-om postaje jednostavan, Java aplikacija na Unix-u jednostavno može da poziva metode COM objekta na Windows serveru.
- Sve ovo postaje transparentno i ne zahteva posebnu administraciju.
- SOAP je možda nekad bio jednostavan, ali je i on postao kompleksan, dok je prikupljao sve više osobina koje ima CORBA.
- XML protokol ima prednost u tome što je manje/više čitljiv za ljude.
- XML procesiranje predstavlja svojevrsno usko grlo za performanse.
- Sve u svemu, CORBA je mnogo efikasniji, premda je SOAP bolje prilagođen mrežnoj arhitekturi.
- Ako se oba objekta koji komuniciraju međusobno implementiraju u Javi onda je sva uopštenost i kompleksnost CORBA-e ili SOAP-a potpuno nepotrebna.

5.7 Hyper Text Markup Language

- HTML - jednostavan jezik za izradu hipermedijskih WEB dokumenata
- Temelji se na oznakama (*tags*) kojima se oblikuje izgled dokumenta koji se sastoji od teksta, slike, zvuka, filma i drugih objekata.
- Razvijen je početkom 90-tih god. na temeljima ***S**tandard **G**eneralized **M**arup **L**anguage*) - definisana sintaksa elemenata za oznake
- Razvijao se i proširivao, pa je prerastao u jezik pogodan za iterakciju korisnika i WEB servera i prikaz dinamičkih multimedijalnih objekata.
- Razvojem dinamičkog HTML (DHTML) postaje nezamislivo kreiranje stranica koje ne sadrže dinamičke objekte i naredbe skriptnih jezika.
- Podrazumeva korišćenje struktuiranih tekstualnih datoteka u kojima se nalaze podaci i oznake jezika koje prepoznaju klijentske aplikacije
- Moderni Web čitači naprednim osobinama diktiraju razvoj HTML i DHTML te olakšavaju interpretaciju i kvalitet stranica.
- Obogačen ugrađenim skript-programima koji se izvršavaju na strani klijenta ili servera, HTML predstavlja odličan jezik za prezentaciju
- Svaki HTML dokument se sastoji od određenih oznaka (*tag*) i tekstualnih podataka koji dele dokumenat na sekcije zaglavlja i tela.

5.8 eXtensible Markup Language

- Predstavlja jedinstvenu tehnologiju za modeliranje struktura podataka koje se razmenjuju između poslovnih softvera na Web-u.
- Može da se pokrene na bilo kojoj platformi, OS ili okruženju i dizajnerima pruža mehanizme za bolje opisivanje njihovog sadržaja.
- Prvobitno je razvijen za izdavačke kuće i njihove projekte, ali je kasnije razvijen za efikasnu i lakšu razmenu podataka na Webu.
- XML jezik predstavlja standardni način za modeliranje struktura podataka u elektronskom poslovanju.
- Format koji obezbeđuje XML za računarske elemente može se prilagoditi najrazličitijim oblastima, kao što su elektronska razmena podataka, čuvanje podataka, odvajanje podataka od prezentacije, vektorska grafika, sistemi glasovne pošte, izrada novih specijalizovanih jezika za označavanje.
- Osnovna svrha XML-a je da olakša deljenje podataka kroz različite informacione sisteme, posebno kroz one sisteme koji su na Internetu.
- XML je nastao iz potrebe da se same informacije sa HTML strana fizički odvoje od načina na koji se prikazuju unutar Web strana

5.8 eXtensible Markup Language

- To je proširivi jezik za označavanje, veoma prost i fleksibilan jezik zasnovan na SGML-u (*Standard Generalized Marup Language*).
- SGML definiše gramatiku za sve markup jezike dokumenata.
- SGML dokumenti nose svoju gramatičku definiciju sa sobom u obliku ***Document Type Definition***-a (**DTD**) datoteka.
- Omogućuje dizajnerima da pišu svoje definicije tipova dokumenata (***D****ocumet* ***T****ype* ***D****efinitions*) koje opisuju skupove oznaka i atributa koji mogu da se koriste za opisivanje specifične vrste sadržaja
- DTD su pravila vladanja jezika za označavanje koja definišu koji elementi označavanja mogu biti upotrebljeni za opisivanje dokumenata
- XML je dovoljno robustan i proširiv da može da opiše ne samo sadržaj već i ***metapodatke*** (informacije koje opisuju druge informacije).
- Mnogi poslovi vezani za pretraživanje ili razmenu informacija mogu se automatizovati pomoću XML-a
- Struktura XML datoteke je hijerarhijska: otvoreni i zatvoreni tagovi
- Za tumačenje XML strukture i transformaciju u određenu strukturu podataka,koriste se **parseri**-programi napisani u odgovarajućem jeziku

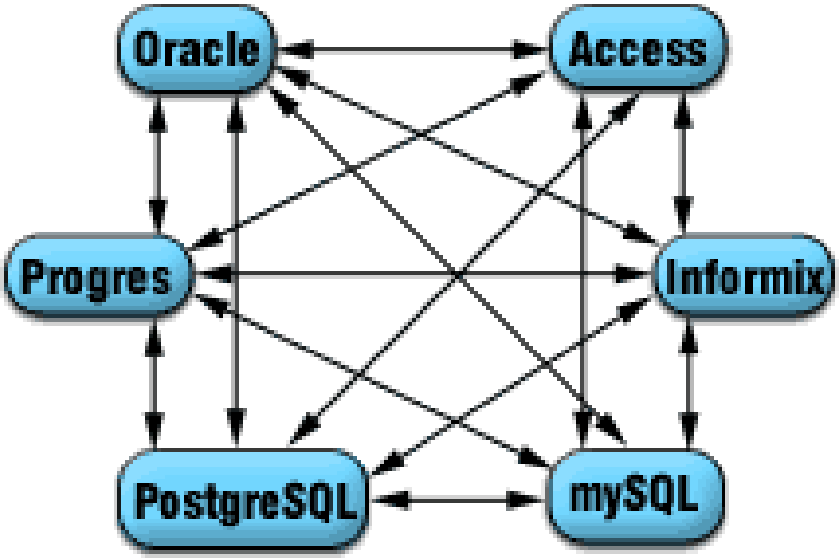
5.8 Primer XML dokumenta

```
<osoba>
  <ime_i_prezime>
    <ime>Miloš</ime>
    <prezime>Crnjanski</prezime>
  </ime i prezime>
  <zanimanje>književnik</zanimanje>
  <zanimanje>diplomata</zanimanje>
</osoba>
```

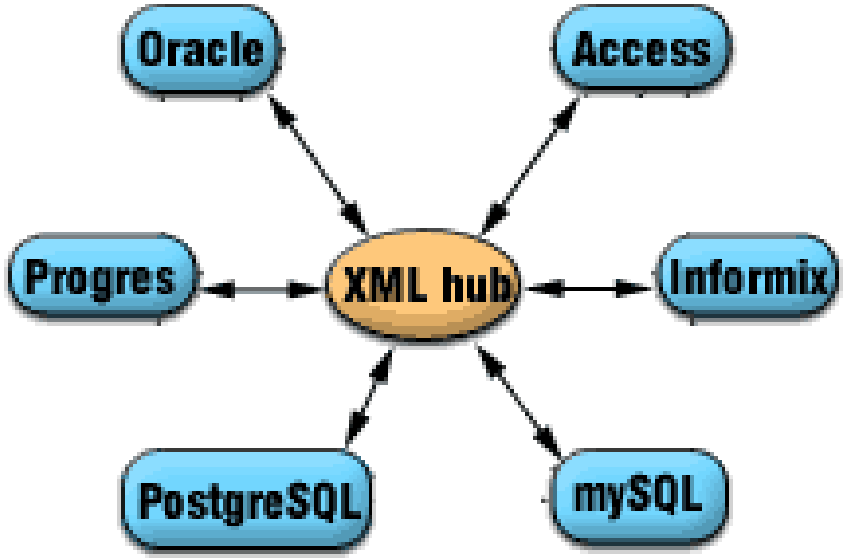
- Iako XML ne zahteva striktnu specifikaciju gramatike dokumenta, dobra je praksa da se ona formira, da bi odgovarajući **parseri** mogli da izvrše validaciju dokumenta opisanog XML datotekom.
- Najvažnije prednosti korišćenja XML standarda za modeliranje su:
  - ✓ **Samodokumentovanje**
  - ✓ **Veliki stepen konzistentnosti sa HTML jezikom – jezikom Web-a.**
  - ✓ **XML je danas usvojen kao de fakto i de jure standard za EDI (*Electronic Data Interchange*).**
  - ✓ **XML reprezentacija kompletne baze ili nekog njenog dela može poslužiti za backup, nezavistan od RDBMS (sistema za upravljanje relacionim bazama podataka)**, ili za migraciju podataka sa jednog sistema na drugi (npr. Oracle > SQL Server).

5.8 Način funkcionisanja XML-a

- Migracija se može izvesti na 2 načina:
  1. **pisanjem koda** koji će vršiti svaku pojedinačnu migraciju (sl.1)
  2. **centralnog XML hub-a** (sl.2)
- Prednost centralnog hub-a je da su za dodavanje novog formata baze potrebna samo 2 nova konvertora, dok je kod drugog načina za 6 prikazanih sistema potrebno ukupno 30 filtera.
- Najnovije verzije gotovo svih vodećih browser-a podržavaju XML.
- Na taj način podaci iz različitih baza podataka jednostavno se prezentuju klijentima putem različitih browser-a
- Najčešći način za prikaz XML-a na Internetu je putem **XSLT**-a (XML *Style Language Transformation*).



slika 1



slika 2

Hvala na pažnji !!!

Pitanja

? ? ?