ⰉⰕⰞⰑⰉⰎⰅⰐⰉ ⰎⰀⰈⰆⰐⰀⰎⰂⰕⰞⰑⰉ

ⰕⰑⰅⰈⰀⰐⰉⰏⰉⰍⰋⰆⰐⰅ ⰟⰄⰀ ⰋⰟⰄⰎⰀⰕⰎⰆ ⰅⰐⰕⰅⰎⰕⰎⰋⰂⰅ ⰀⰕⰏⰋⰔⰀⰜⰋⰆⰐⰀ

ⰒⰀⰎⰋⰉ ⰛⰀⰍⰀⰎ

Otvoreno računarstvo

- Tehnologije za izradu *enterprise* aplikacija
  - Uvod
  - *Enterprise JavaBeans*
  - *Middleware* i komunikacija porukama

Mario Žagar

Uvod

Enterprise Software

- *Software used to run a company:*
  - *computer software designed to integrate and automate all of a company's functions**

- Naglasak je na programskoj podršci namijenjenoj poslovanju (svim segmentima) velikih tvrtki

Što sve treba *enterprise* aplikaciji?

- Upravljanje transakcijama
  - Grupa zavisnih operacija se izvršava kao jedinstvena transakcija
- Sigurnosne usluge
  - Sprječavanje neautoriziranog pristupa resursima
- Upravljanje postojanošću stanja (*persistence*)
  - Automatizirana pohrana podataka
- Upravljanje resursima
  - Skalabilnost, učinkovitost
- Višenitnost (*multithreading*)
  - Konkurentni pristup poslovnoj logici i podacima

Koristi uobičajenih zajedničkih usluga

- Izvršno okruženje treba omogućiti korištenje uobičajenih (*common*) zajedničkih usluga
  - Ponovno korištenje (*reusability*)
  - Bogata i reprezentativne ponuda usluga
  - Nepromjenjiv (prenosivi) API
- Normiranje:
  - Ubrzava razvoj
  - Mogućnost korištenja različitih alata
  - Vještine su prenosive između platformi
  - Korištenje standardnih usluga
- Tehnologije se prilagođavaju normama
  - Jednom razvijati, više puta iskoristiti, izvršiti
  - Pružanje usluga na normirani način

Komponentni modeli

- Arhitektura i API koji omogućuje razvojnim inženjerima definiciju ponovno iskoristivih segmenata kôda
  - Koji mogu sagraditi aplikaciju
- Komponentni model ubrzava razvoj
  - Pojednostavljenje i brža integracija
  - Jednostavnije automatsko testiranje
  - Mogućnost jednostavnijeg ponovnog korištenja objekata
  - Standardni životni ciklus objekata i upravljanje objektima
- Danas često korišten objektno-orijentirani komponentni model

Mogućnost zajedničkog rada

- Komponente u načelu koriste niz raznih resursa
  - Resursi nisu nužno u istom komponentnom modelu
  - Resursi nisu nužno na istom poslužitelju
  - Sve komponente nisu nužno pisane istim jezikom

- Mogućnost zajedničkog rada
  - Jednostavnost i pouzdanost
  - Kompatibilnost tehnologije neovisno o platformi
  - Mogućnost komunikacije komponenata različitih platforma
  - Mogućnost korištenja resursa drugih platforma
  - Podrška za druge komponentne modele

Raspodijeljenost aplikacija

- Enterprise aplikacije su često raspodijeljene
  - Postojeće usluge se nalaze na različitim poslužiteljima
  - Potreba za redundancijom
  - Potreba za raspodjelom opterećenja (*load-balancing*)
  - Grupiranje nezavisnih sustava tako da rade kao jedan
  - Stvaranje grozdova poslužitelja (*clustering*)

- Distribucijski mehanizam
  - Dostupan svim aplikacijama sustava
  - Jednostavan
  - Automatska redundancija i raspodjela opterećenja

Kako sve to zadovoljiti?

- Rješenje:
  - Komponentni model temeljen na normi/normama
  - Razvoj raspodijeljenih aplikacija

- Moguće implementacije rješenja:
  - COM+, DCOM
  - CORBA
  - Java RMI (Remote Method Invocation)
  - Enterprise JavaBeans

DCOM

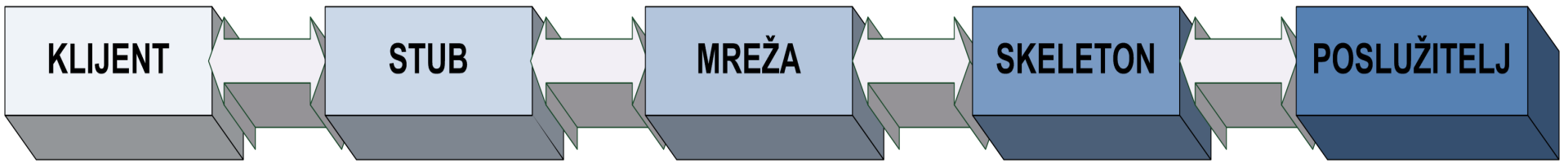
- COM+, DCOM
  - Vlasnička (*proprietary)* Microsoft tehnologija
  - Temeljena na COM (Component Object Model)
  - COM+ kombinira COM komponente i MTS (Microsoft Transaction Server)
- Koristi:
  - Dobavljanje resursa iz skupa (*resource pooling*)
  - Nezavisnost aplikacija
  - Raspodijeljene transakcije
- Tehnologija na kojoj se temelji .NET framework

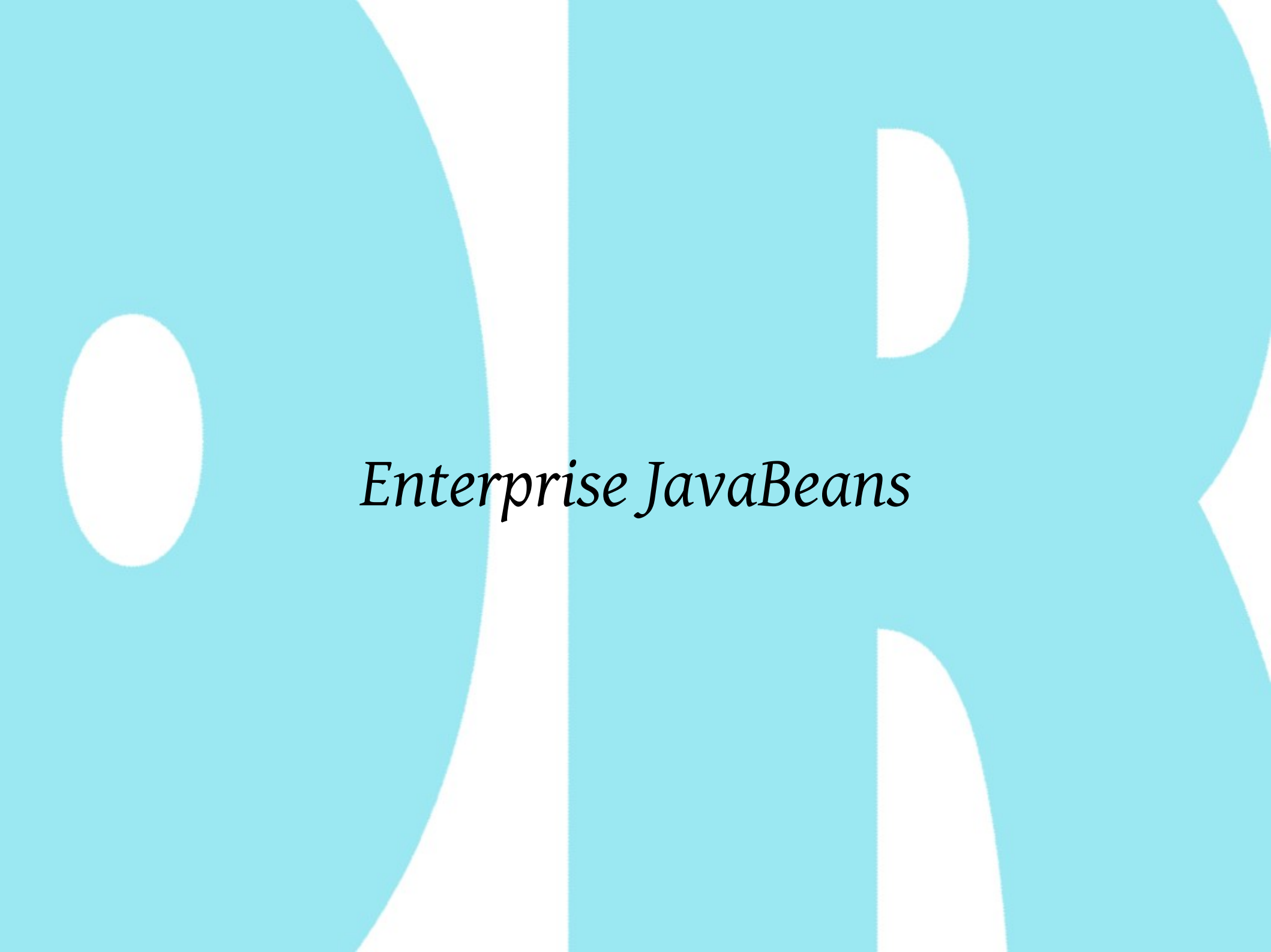
CORBA

- CORBA – Common Object Request Broker Architecture
- Arhitektura posrednika za pozive objektima
  - Ideja raspodijeljene objektno-orijentirane arhitekture
  - Odvaja klijenta i poslužitelja putem formalne definicije sučelja
  - Norma OMG (Object Management Group)
  - Neovisnost o programskom jeziku
  - Koriste se ORB-ovi (Object Request Broker) za komunikaciju s udaljenim objektima
  - Komunikacija putem IIOP (Internet Inter-Orb Protocol)
  - Skup standardnih usluga putem API

Java RMI

- Java RMI (Remote Method Invocation)
- Udaljeni pozivi procedura
  - Pozivi metoda udaljenih objekata
  - Java API - dio Java SE
  - Pozivi udaljenih Java objekata
  - Pozivi iz jednog JVM u drugi JVM (Java-to-Java)




Enterprise JavaBeans

Java EE standardne usluge/dijelovi

- Java EE standardne usluge uključuju:
  - **HTTP**
  - **RMI-IIOP** (RMI over Internet Inter-ORB Protocol) – udaljeni pristup komponentama
  - **Java IDL** (Interface Definition Language) – jezik za opis sučelja
  - **JAAS** (Java Authentication and Authorization Service) – sigurni pristup podacima i obradi
  - **JAXP** (Java API for XMP Parsing) – usluge za manipulaciju XML-om
  - **JDBC** (Java Database Connectivity) – pristup bazama podataka putem SQL-a
  - **JTA** (Java Transaction API) – podrška za transakcije
  - **JavaMail** – usluga slanja elektroničke pošte
  - **JNDI** (Java Naming and Directory Interface) – imenici podataka
  - **JMS** (Java Messaging Service) – razmjena podataka putem poruka
  - **JAF** (JavaBeans Activation Framework) – okružje za rad JavaBeans komponenti
  - **Web usluge** – JAX-RPC, SAAJ
  - **JCA** (Java EE Connector Architecture) – arhitektura spojnih sučelja
  - **JACC** (Java Authorization Service Provider Contract for Containers)

JNDI, JAXP

- Java Naming and Directory Interface – JNDI
  - Sučelje za **imenovanje** (i pronalaženje) resursa
  - Omogućava aplikacijama **pronalaženje i dohvat** objekata bilo kojeg tipa
  - Služi pristup poslužiteljskim objektima (npr. EJB)
    - Izvorima podataka – DataSources
    - Redovima poruka – JMS Queue
- Java API for XML Parsing – JAXP
  - Usluge za parsiranje i transformacije XML podataka
  - Neovisna o implementaciji XML procesora
  - Sadrži industrijski standardizirane komponente
    - Document Object Model – **DOM**
    - Simple API for XML Processing – **SAX**
    - XML Style Language Transformation – **XSLT**

JAAS, JTA, JCA

- Java Authentication and Authorization Service - JAAS
  - **Autentikacija** korisnika - pouzdano i sigurno definiranje TKO
    - JAAS autentikacija neovisna od tehnologija
  - **Autorizacija** korisnika - akcije se izvode uz odgovarajuće dozvole
    - JAAS autorizacija je nadogradnja sigurnosne Java arhitekture - kontrola pristupa temeljena ulogama
- Java Transaction API – JTA
  - Specificira sučelja između upravitelja **transakcijama** i drugih u raspodijeljenom transakcijskom sustavu
  - Implementacijski neutralno
- Java EE Connector Architecture – JCA (nekad J2C)
  - Omogućava resursne adaptere drugim resursima
    - Definira “ugovor” upravljanja spojevima

Enterprise JavaBeans

- Enterprise JavaBeans
  - Komponente koje služe za enkapsulaciju:
    - **Poslovne logike**
    - **Aplikacijsko-specifične logike**
    - **Pristupa podacima** (najčešće prema bazi podataka)
  - Omogućen pristup od strane više aplikacija koje koriste dijelove istih (ili sličnih) poslovnih procesa
- Rješavaju problem skalabilnosti
  - Dodavanje novih klonova komponenti po potrebi
  - Koriste ih svi – po potrebi
  - Rješavaju:
    - *Load-balancing, high-availabilty. redundancy, reusability, ...*

EJB sadržnik (*container*)

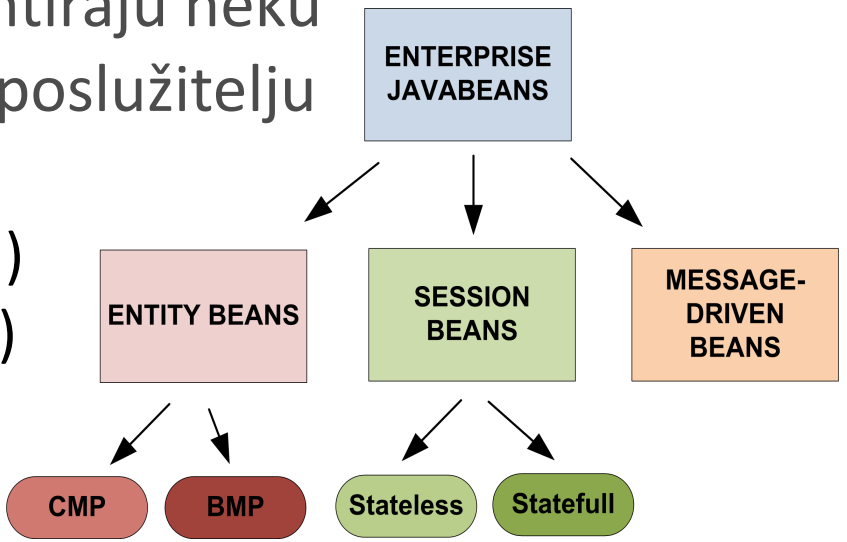
- Aplikacijski poslužitelj upravlja nizom EJB sadržnika
  - Omogućava **pristup uslugama sustava**
    - Usluge za **imenovanje**
    - Usluge za podršku **transakcijama**
    - **Sigurnosne** usluge
- EJB sadržnik (*container*)
  - Upravlja EJB razredima, instancama objekata i pristupom Enterprise JavaBean-ovima
    - Isto kao što to radi Web sadržnik za Servlete i JSP-ove
  - Nameće transakcijsku okolinu
  - Upravlja životnim ciklusom Enterprise JavaBeana
  - Može upravljati postojanošću (kod Entity beanova)

EJB klijent

- Što sve može biti **EJB klijent** (korisnik, pozivatelj):
  - Drugi EJB
    - *Client Bean* unutar istog ili drugog EJB sadržnika
  - Java Servlet ili JavaServer Page
    - Kao korisničko sučelje prema EJB poslovnoj logici
  - Java aplikacija
    - Samostalna (*standalone*) Java aplikacija koja koristi EJB poslovnu logiku
      - Često kod aplikacija unutar iste zone sigurnosti (vatrozid)
  - Java Applet
    - Mora biti preuzet s Web poslužitelja koji ima istu IP adresu kao i Java EE poslužitelj s EJB sadržnikom
  - Neki drugi (ne-Java) klijenti
    - Npr. CORBA klijent

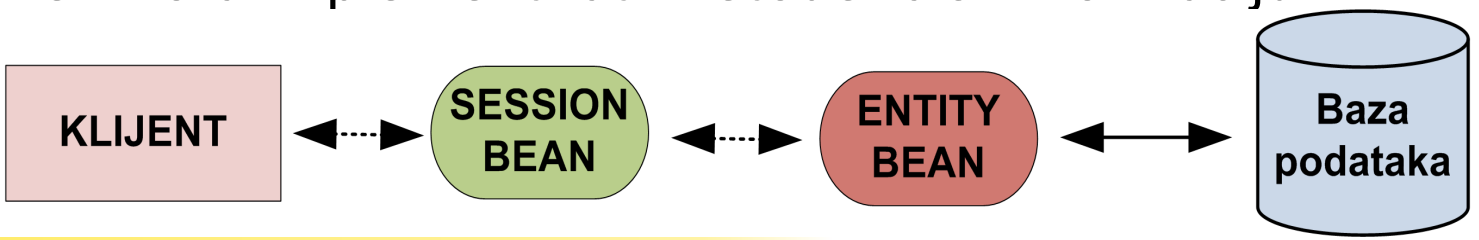
Tipovi Enterprise JavaBeanova

- Entity Beans
  - Objektno-orijentirani pogledi na trajno pohranjene podatke (npr. baza podataka)
  - Dva tipa:
    - **Container-managed persistence** (CMP)
    - **Bean-managed persistence** (BMP)
- Session Beans
  - Nepostojani objekti koji implementiraju neku poslovnu logiku i izvršavaju se na poslužitelju
  - Dva tipa:
    - **Sa zadržavanjem stanja** (statefull)
    - **Bez zadržavanja stanja** (stateless)
- Message-Driven Beans
  - Asinkroni sakupljači poruka



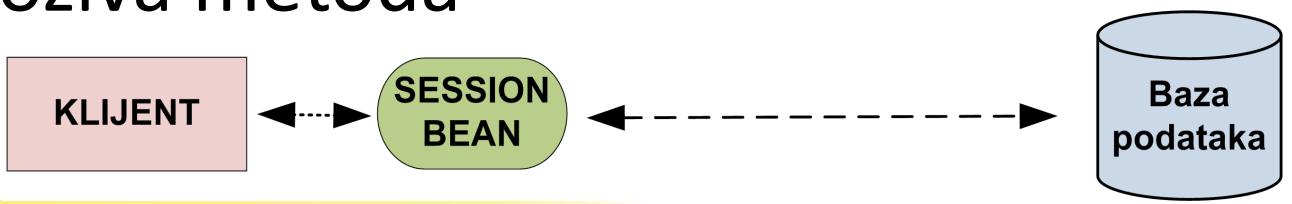
EJB – *Entity Bean*

- Instanca Entity Beana
  - predstavlja **objektno-orijentirani pogled na trajno pohranjene podatke**
  - na primjer u bazi podataka
- Postojanošću može upravljati:
  - Sadržnik (EJB container) - **Container-managed persistence – CMP**
    - EJB sadržnik brine o sinkronizaciji objekta i stvarnih podataka u bazi
  - Sam Bean - **Bean-managed persistence – BMP**
    - Programer mora implementirati metode za sinkronizaciju Bean o



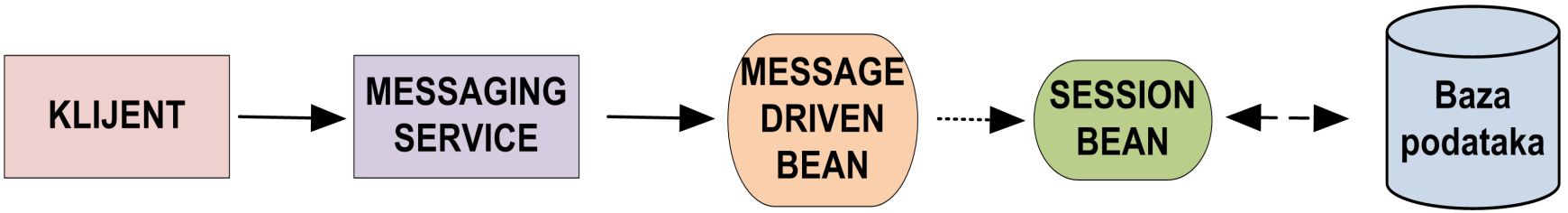
EJB - *Session Bean*

- Instanca Session Beana je **nepostojani** (*non-persistent*) **objekt koji implementira neku poslovnu logiku** i izvršava se na poslužitelju
- Mogu se ponovno koristiti od više klijenata
- Ne mogu dva klijenta konkurentno koristiti isti Session Bean
- **Stateless Session Bean** ne zadržava konverzacijsko stanje između poziva metoda
- **Stateful Session Bean** zadržava konverzacijsko stanje između poziva metoda



EJB – *Message-Driven Bean*

- Instanca Message-Driven Bean-a (MDB) je **asinkroni sakupljač poruka**
- Ne vide se izravno od strane klijenta
  - Klijent šalje poruke redu poruka
  - MDB preuzima poruke iz reda poruka
- Najnovija specifikacije podržava MDB za bilo koji tip poruka
  - Java Message Service (JMS) tip poruka je prirodni odabir



Komunikacija s EJB-ovima

- EJB-ovima se pristupa putem udaljenog poziva metoda u Javi – RMI
  - Mora postojati EJB udaljeno sučelje RMI oblika
- Distribucijski protokol je RMI-IIOP
  - Remote Method Invocation over Internet Inter-Orb Protocol
  - Java RMI protokol koristi dijelove CORBA infrastrukture
  - IIOP omogućava jednostavniju komunikaciju sa starijim (*legacy*) aplikacijama u drugim programskim jezicima
    - Jezici koji podržavaju normu CORBA: C++, Smalltalk, ...
- Iznimno, mogu se koristiti i drugi distribucijski protokoli

Middleware i komunikacija porukama

Sinkrona i asinkrona komunikacija

- Sinkrona komunikacija
  - Strane komuniciraju izravno
  - Jedna strana čeka na odgovor druge strane
  - Primjer – **telefonski razgovor**

- Asinkrona komunikacija
  - Manja osjetljivost na trenutnu nedostupnost komunikacijskog kanala ili nespremnost druge strane
  - Poruke se isporučuju kad je to moguće
  - Mehanizmi koji osiguravaju sigurnu isporuku poruke
  - Potreban posrednik
  - Primjer – **SMS**

Komunikacija porukama

- Raspodijeljene aplikacije se sastoje iz niza raznolikih komponenti koje međusobno komuniciraju
- Heterogene komponente otežano komuniciraju ako ne koriste poznate i široko korištene protokole
- Prijedlog rješenja:
  - **Komunikacijska infrastruktura temeljena na porukama**
    - Omogućuje relativno jednostavan način povezivanja različitih komponenata i/ili sustava putem dijeljenog skupa sučelja
    - Mogućnosti komunikacije:
      - Sinkrona komunikacija
      - Asinkrona komunikacija

Komunikacija porukama #2

- Oblik labave (*loosely coupled*) raspodijeljene komunikacije
  - Pod komunikacijom se podrazumijeva razmjena poruka između programskih komponenata

- Porukama usmjerene (*message-oriented*) tehnologije:
  - "opuštaju" čvrsto povezane (*tightly coupled*) oblike komunikacije
    - npr. priključnice, CORBA, RMI
    - često imaju samo sinkronu komunikaciju specifičnim komunikacijskim protokolom
  - uvode posredničku komponentu između strana
    - komponente mogu indirektno komunicirati preko posrednika
  - pošiljatelj **ne treba** precizno **poznavati primatelja**, već samo posrednika

Middleware

- Middleware se prevodi na razne načine:
  - međuprogramska podrška
  - posrednička programska podrška
  - programski posrednički (među)sloj
- Definicije su razne (www.middleware.org/whatis.html):
  - "*softversko ljepilo*"
  - najkraća – to je "**/**" u klijent/poslužitelj
  - programski sloj između operacijskih sustava i aplikacija na obje strane raspodijeljenog sustava
  - programska podrška koja omogućuje heterogenim klijentima spoj na zajedničke poslužitelje
  - programska podrška koja posreduje između aplikacije i mreže

Tipovi *middlewarea*

- Podatkovna međuprogramska podrška (*database middleware*)
- Objektno-orijentirana međuprogramska podrška
  - Object Request Broker – ORB
- Međuprogramska podrška pozivima udaljenih procedura (*Remote Procedure Call – RPC*)
- Transakcijska međuprogramska podrška
- Porukama usmjerena međuprogramska podrška (*Message-Oriented Middleware – MOM*)
- Portalski poslužitelji
- ...

Objektno-orijentirana međuprogramska podrška

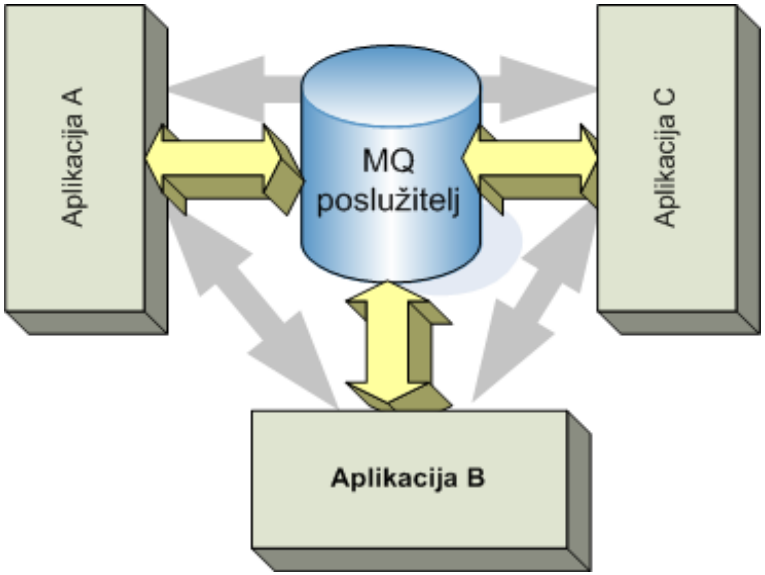
- Object Oriented Middleware
- Temelji se na udaljenom pozivu procedura (*Remote Procedure Call* – RPC)
- To su zapravo većinom dijelovi objektno-orijentiranih komponentnih modela
- Primjeri RPC orijentiranih protokola:
  - **CORBA** (*Common Object Request Broker Architecture*)
  - **RMI** (*Remote Method Invocation* – Java)
  - **DCOM** (*Distributed Component Object Model* – Microsoft)
    - To smo već obradili ☺

Porukama usmjerena međuprogramska podrška

- Message Oriented Middleware – MOM
  - Tip međuprogramske podrške koji omogućuje razmjenu poruka opće namjene u okruženju raspodijeljenih aplikacija
- Podaci se razmjenjuju prosljeđivanjem poruka ili stavljanjem u redove poruka (*message queue*)
- Koristi mehanizme sigurne isporuke poruka
- Podržava:
  - Sinkronu komunikaciju porukama
  - Asinkronu komunikaciju porukama
- Najčešće se ipak identificira s asinkronom komunikacijom pomoću redova poruka

MOM

- Načelno podržava razne protokole komunikacije
- Najčešće implementiran s mogućnostima **pohrane i prosljeđivanja** (*store-and-forward*) u **redovima poruka** (*message queue*)
  - Zato kod MOM najčešće i govorimo o **asinkronoj komunikaciji redovima poruka**
- Međuprogramska podrška zasnovana na redovima poruka (Message Queuing Middleware – MQM)
  - Najčešći tip implementacije MOM



Redovi poruka (*message queues*)

- Redovi poruka su međuspremnici između sudionika komunikacije
  - Pošiljatelji (proizvođači) **spremaju poruke u red**
  - Primatelji (potrošači) **dohvaćaju poruke iz reda**
  - **Nema izravnog komunikacijskog kanala** između aplikacija pošiljatelja i primatelja
  - Komunikacija je **asinkrona**
  - Potrebno je samo implementirati komunikacijski protokol

Redovi poruka – način rada

- Poruke se nalaze u redu dok ih netko ne preuzme
- Poslužitelj ne mora biti dostupan u trenutku slanja
- Poslužitelj može dohvatiti poruku kad može/stigne
- Poruke se mogu preuzimati iz reda u poretku neovisnom o poretku stavljanja
- Posrednik može povećati prioritet određenim porukama ili upravljati opterećenjem
- Posrednik je otporan na grješke (gubitak poruke, neisporuku) korištenje postojanih redova
  - Sustav za pohranu poruka (baza, datotečni sustav)

MOM – prednosti i nedostaci

- Prednosti:
  - Mogućnost pohrane, usmjeravanja i pretvorbe poruke prije isporuke
  - Jednostavniji model razvoja i komunikacijskog sustava
  - Manja osjetljivost na nedostupnost mreže
  - Mehanizmi sigurne isporuke poruka
  - Jednostavna integracija heterogenih sustava/aplikacija
  - Mogućnost upravljanja prioritetom i brzinom isporuke
  - Manje kontrolne komunikacije
- Nedostaci
  - Pomanjkanje norma
  - Teško stvaranje netekstualnih poruka
  - Skupoća postojeće programske podrške
  - Potreba za specifičnim klijentima i poslužiteljima

Odluke za korištenje redova poruka

- Kada nije bitna trenutna isporuka poruke
- Kada je potrebna sigurnost da će poruka biti isporučena (npr. kod pada jednog dijela mreže)
- Kada komuniciraju heterogeni sustavi ili aplikacije

Produkti temeljeni na redovima poruka

- Komercijalni:
  - **IBM** WebSphere MQ
  - **Microsoft** Message Queue Server – MSMQ
  - **Oracle** Streams
  - **BEA** MessageQ
  - **TIBCO** Enterprise Message Service
- Open source:
  - **JBoss** Messaging
  - **ObjectWeb** JORAM
  - Open Source Message Queue – OSMQ
  - **Apache** Qpid
  - **Rabbit** RabbitMQ
  - **Red Hat** Enterprise MRG

Advanced Message Queuing Protocol

- Advanced Message Queuing Protocol (AMQP)
  - Otvorena norma protokola aplikacijskog sloja za MOM (Message Oriented Middleware)
  - Nalazi se na razini iznad API
    - Za razliku od Java Message Service
  - Definira mrežni protokol i posredničke usluge:
  - Definirani skup mogućnosti komunikacije porukama nazvanih **Advanced Message Queuing Protocol Model**
    - Skup komponenata koje usmjeruju i pohranjuju poruke
    - Koristi se posrednik
    - Skup pravila za spajanje tih komponenata
  - Omogućava klijentima komunikaciju s posrednikom

Java Message Service (JMS)

- Pripada u skupinu **međuprogramske podrške usmjerene prema porukama** (Message Oriented Middleware - MOM) u Javi
  - Slanje poruka između dva ili više klijenat
  - Dio Java EE platforme
  - Definiran kao JSR 914 pod JCP
- Podržava 2 modela:
  - **Point-to-point** ili queuing model
    - Slanje poruke (jednom) **poznatom** primatelju
  - **Publish and subscribe** model
    - Slanje poruke s određenom **temom** nepoznatim primateljima

JMS - elementi

- Elementi:
  - **JMS provider** – implementacija MOM JMS sučelja kao Java JMS ili kao ne-Java adapteri
  - **JMS client** – aplikacija ili proces koji proizvodi i prima poruke
  - **JMS producer** – JMS klijent koji proizvodi i šalje poruke
  - **JMS consumer** – JMS klijent koji prima poruke
  - **JMS message** – sama poruka koju razmjenjuju JMS klijenti
  - **JMS queue** – red koji sadrži poslane nepročitane poruke
  - **JMS topic** – tema poruke

JMS kompatibilni produkti

- Komercijalni:
  - IBM WebSphere MQ (ex MQSeries)
  - Sun Java System Message Queue
  - Oracle AQ
  - SAP NetWeaver WebAS Java JMS
  - BEA Weblogic
  - Progress SonicMQ
  - TIBCO Enterprise Message Service
  - webMethods Broker Server
- Open Source:
  - Apache ActiveMQ
  - FUSE Message Broker
  - OpenJMS
  - JBoss Messaging
  - Objectweb JORAM

Publish-and-Subscribe

- **Publish-and-Subscribe** ili **Publish/Subscribe** ili **Pub/Sub**
  - Komunikacijski predlošci **objave i pretplate**
- **Način komunikacije:**
  - **Izdavač, objavitelj** (*publisher*) ili pošiljatelj (*sender*) **objavljuje** poruke (*publish*)
  - **Pretplatnik** (*subscriber*) ili primatelj (*receiver*) se **pretplaćuje** (*subscribe*) na određene poruke
  - Pošiljatelji nisu programirani za slanje poruka određenim specifičnim primateljima (pretplatnicima)
  - Poruke se svrstavaju u klase
  - Pri slanju se ne zna da li postoje pretplatnici
  - Pretplatnici izražavaju interes za određenim razredima poruka

Publish-and-Subscribe – značajke

- Pub/Sub je dio MOM (Message Oriented Middleware)
  - Srodnik MQM (Message Queuing Middleware)
- **Filtriranje** – proces odabira poruka za prihvat:
  - Po **temi** (*topic*) – poruke se svrstavaju u određene teme
    - Pošiljetalj postavlja temu poruke
  - Po **sadržaju** (*content*) – poruke se pretražuju za određeni sadržaj
- Neki nedostaci kod konkretnih implementacija:
  - Često ne uključuju sigurnu dostavu poruke
  - Često nema mehanizma indikacije na strani pošiljatelja da postoje problemi s primanjem poruka / primateljem

Pitanja ?