

ՅՈՒՐԱԿԱՆ ԵՐԱՆՈՒՄ ԵՎ ԵՐԱՆՈՒՄ

ԳՅՈՒՄԱՆԱՅԻՆ

ՈՒՆԱԾԱՆ ԵՎ ԵՐԱՆՈՒՄ

Otvoreno računarstvo

Web tehnologije i aplikacijski poslužitelji

- Skalabilni svijet Jave i aplikacijski poslužitelji
- Tehnologije za izradu aplikacija Weba
 - Servleti
 - MVC, JavaServer Pages, JavaBeans

Mario Žagar





Skalabilni svijet Jave i aplikacijski poslužitelji

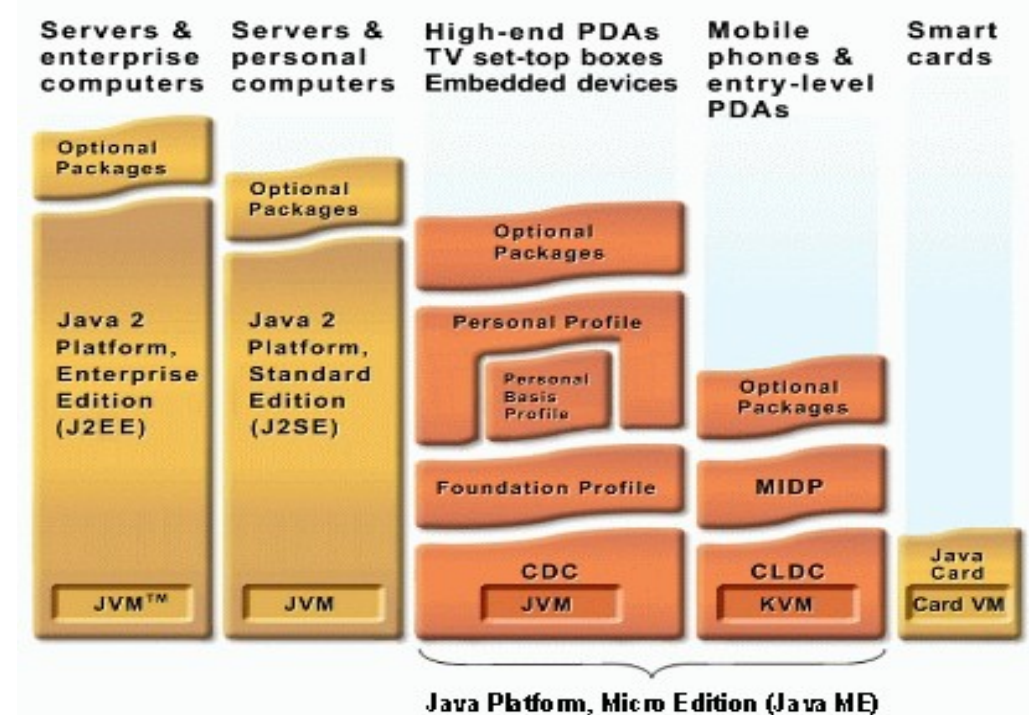
Skalabilnost

- Da li se **ista** programska platforma može primijeniti na razne **uređaje** i razne **razine kompleksnosti**?
 - **Mikro aplikacije** - na pametnim karticama (*smart card*)
 - Npr. aplikacija za digitalno potpisivanje podataka
 - **Male aplikacije** - na pametnim telefonima (*smartphone*)
 - Npr. aplikacija za pohranu lozinki ili igrice
 - **Aplikacije srednje kompleksnosti** na ugrađenim računalima ili specijaliziranim uređajima
 - Npr. aplikacija na igraćoj konzoli ili dlanovniku
 - **Samostalne klijentske aplikacije** (ili klijent-poslužitelj)
 - Npr. GUI aplikacija s bazom podataka
 - **Aplikacije na poslužiteljima** (višestrukim)
 - Npr. Web trgovina ili Internet bankarstvo



Skalabilnost #2

- Naravno da se **ista** programska platforma **može** primijeniti na razne **uređaje** i razne **razine kompleksnosti** 😊
 - Načelno razlika je samo u količini funkcionalnosti
- Primjer – “svijet Java”
- Postoje “inačice” JVM:
 - Java ME
 - Java SE
 - Java EE



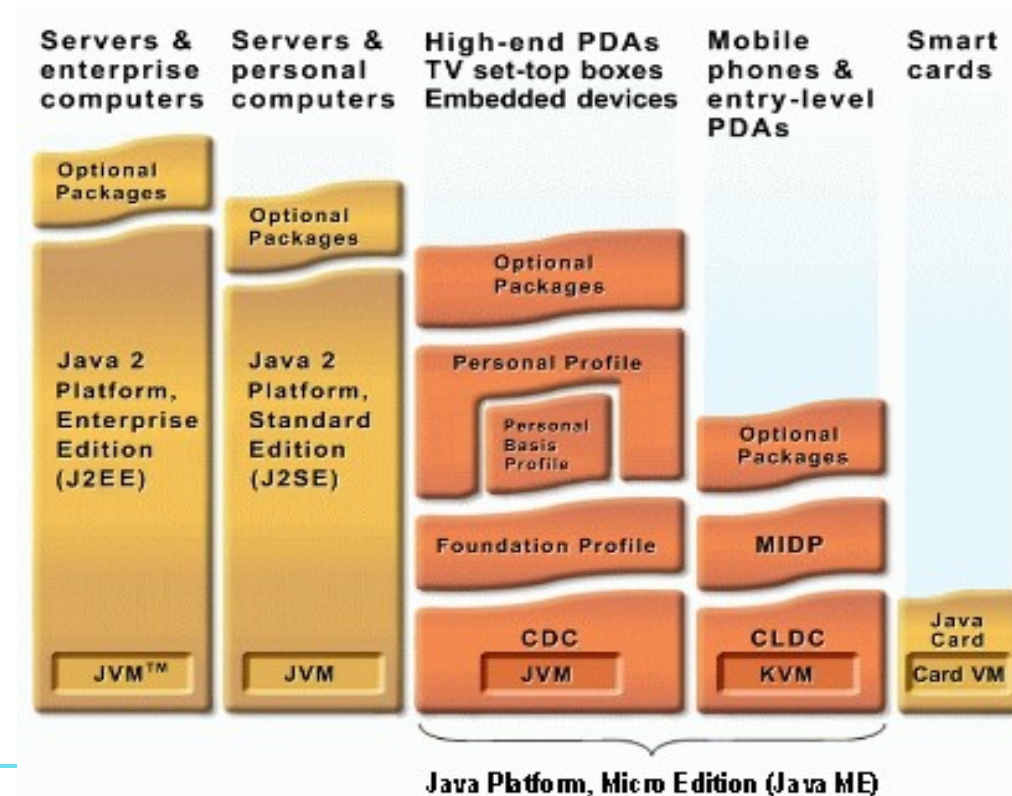
Java ME – Java SE

- Java ME (Micro Edition)
 - **Robusna, mala i prilagodljiva** platforma za aplikacije na mobilnim i ugrađenim uređajima
 - Npr. mobilni telefon, PDA, TV set-top box, ...
 - Naglasak na **prenosivost, mrežnu povezivost i prikaz na raznorodnim uređajima** (zaslonima)
- Java SE (Standard Edition)
 - Izvršna platforma za razvoj Java aplikacija
 - Namjena - izrada samostalnih (*standalone*) aplikacija
 - Sadrži sučelje za pristup **bazama** podataka, **sigurnosne i mrežne** tehnologije, te tehnologiju za izgradnju **korisničkih sučelja**
 - **Osnova** Java platforme

Java SE - Java EE



- Java EE (Enterprise Edition)
 - “Nadogradnja” Java SE
 - Izvršna platforma za razvoj, *deployment* i upravljanje višeslojnim poslužiteljskim aplikacijama na *enterprise* razini
 - Sagrađena na osnovama Java SE
 - Sadrži:
 - Distribuiranu komunikaciju
 - Upravljanje procesima
 - Skalabilnu arhitekturu
 - Sigurnosne mehanizme
 - Upravljanje transakcijama



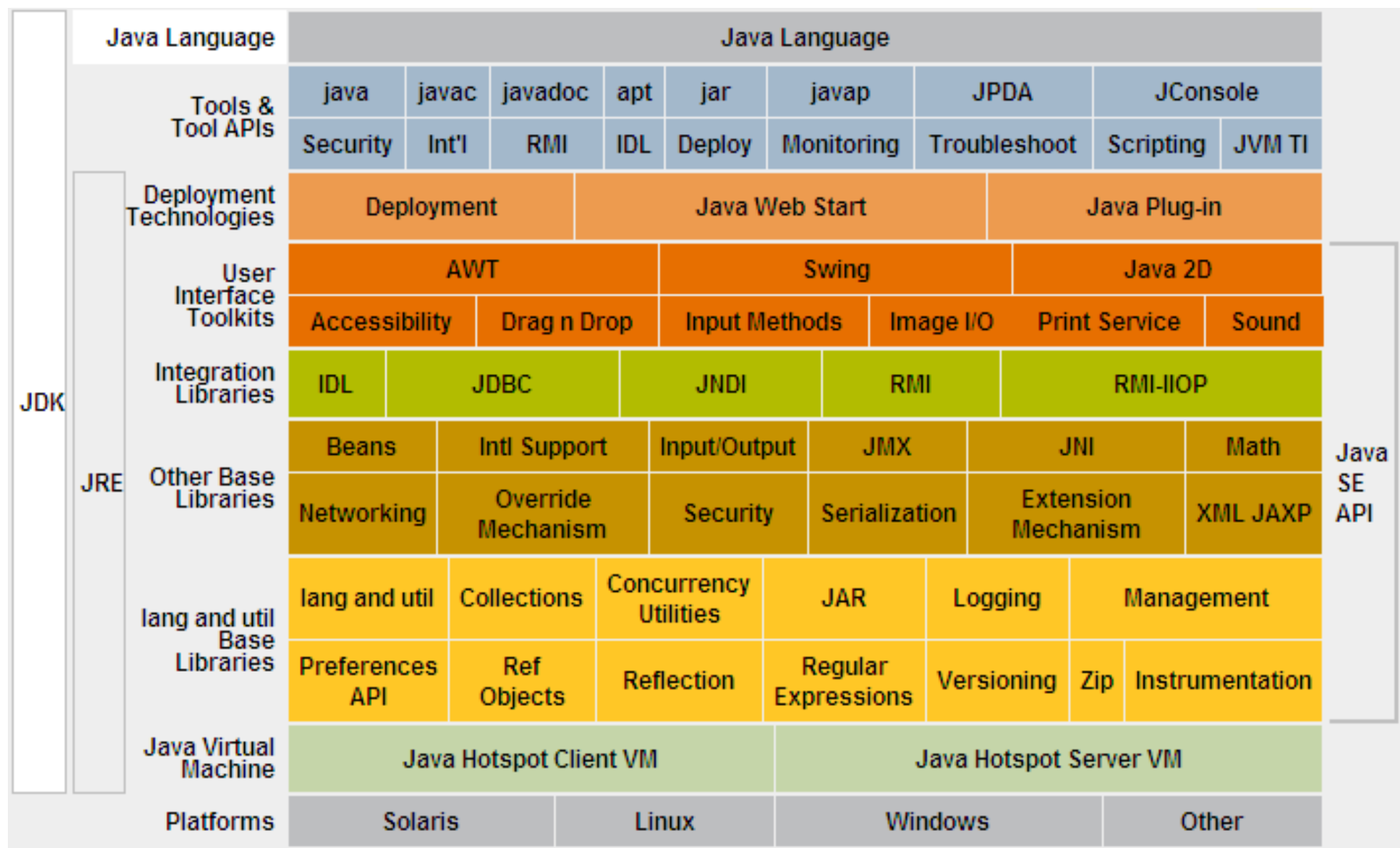
Mala promjena naziva ☺

- Nekad se zvalo **J2SE, J2EE i J2ME**
- **Java Enterprise Edition**
 - Prošlost: **Java 2 Platform Enterprise Edition** ili **J2EE** - zapravo “**Java EE v1.2**”
 - U međuvremenu izašle nove verzije **J2EE 1.3** i **J2EE 1.4**
 - Danas: **Java EE 5** ili **Java Platform, Enterprise Edition 5**
- **Java Standard Edition**
 - Prošlost : prvo **J2SE 1.2**, pa **J2SE 1.3.1** i **J2SE 1.4.2**
 - Onda došao **J2SE 5.0** (to je i zapravo “**J2SE 1.5**”)
 - Danas: **Java SE 6** ili **Java Platform, Standard Edition 6**
- Danas se sve zove: **Java SE, Java EE i Java ME**

Java SE – dijelovi standardnog API-ja

- Spoj na **baze podataka** i upravljanje pohranom podataka
 - JDBC i ekstenzije za upravljanje podacima (javax.sql)
- **Imenici podataka** – imenovanje resursa (za dohvat)
 - Java Naming and Directory Interface (JNDI)
- Usluge za **manipulaciju XML-om** (DOM, SAX)
 - Java API for XML Parsing (JAXP)
- Usluge za **sigurni pristup** podacima i obradi
 - Java Authentication and Authorization Service (JAAS)
- **Udaljeni pristup** komponentama
 - Remote Method Invocation Internet Inter-ORB Protocol (RMI-IIOP)
- Jezik za **opis sučelja**
 - Java Interface Definition Language (Java IDL)

Pregled Java SE platforme



Specifikacija Java Enterprise Edition



- Dijelovi specifikacije:
 - **Java EE Platform**
 - Platforma koja definira komponente, usluge i komunikaciju
 - **Java EE Compatibility Test Suite (CTS)**
 - Skup testova (~ 5000) koji osiguravaju kompatibilnost aplikacije sa Java EE standardom (po specifikaciji)
 - **Java EE Reference Implementation**
 - Izrada prototipova aplikacija i definicija operacija platforme
 - **Java EE Blueprints**
 - Najbolja praksa izrade aplikacija
- Specifikaciju osmislili i podržavaju “najveći igrači”
 - Sun, IBM, Oracle, BEA, ...

Java Community Process



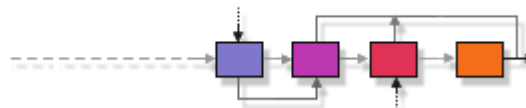
- Java Community Process (JCP) - jcp.org



- Osnovan 1998
- **Otvoreni** proces raznih sudionika za razvoj i reviziju specifikacija Java tehnologija, referentnih implementacija i skupina testova
- Promiče evoluciju Java platforme u suradnji s internacionalnom Java razvojnom zajednicom

- Članstvo:

- Više od 1200 članova
- Potpis Java Specification Agreement-a (JSPA) između Sun-a i pojedinca, tvrtke ili organizacije
- Godišnje \$5000 za tvrtke, \$2000 za edukacijske, državne i neprofitne institucije i **\$0** za pojedince



Community Development of
Java Technology Specifications

JCP – sudjelovanje i JSR



- Sudjelovanje u JCP uključuje:
 - Predstavljanje JSR i izražavanje mišljenja o JSR
 - Sudjelovanje u raznim razvojnim i nadglednim grupacijama
- **JSR (Java Specification Request)** je zahtjev za specifikacijom temeljenom na Java tehnologijama
 - Dokument koji se predstavlja PMO-u (Program Management Office kod Sun-a) kao prijedlog razvoja nove specifikacije (ili revizije postojeće)
 - Trenutno postoji više od 90 Java tehnoloških specifikacija, koje uključuju nove verzije Java ME, Java SE i Java EE
 - Popis svih JSR na <http://jcp.org/en/jsr/all>

Java EE kao norma



- Java EE 5 je definirana specifikacijom JSR 244
 - Smatra se neformalnom, *de facto* normom
 - Ne postoji kao ISO ili ECMA norma
 - Korisnik Java EE licence (*licensee*) potpisuje komercijalnu distribucijsku licencu
 - Posjeduje testove kompatibilnosti i obvezao se na kompatibilnost, ali njegovi produkti (još) ne moraju biti kompatibilni
 - Java **Compatible**, Enterprise Edition oznaka
 - **Proizvodi** koji su prošli skupinu testova (CTS - Compatibility Test Suite) smiju iskazati oznaku
 - Aplikacije kompatibilne s Java EE normom se **moraju moći izvoditi na bilo kojem** Java EE kompatibilnom poslužitelju



http://java.sun.com/javaee/overview/faq/javaee_faq.jsp

Koristi od Java EE

- **Standardiziranost**
 - Mnogo standardnih komponenti, usluga i alata
 - Prenosivost aplikacija na druge Java EE aplikacijske poslužitelje
- **Skalabilnost**
 - Učinkovitost i dostupnost – skalabilnost
- **Standardne usluge**
 - Već ugrađene (podržane) kod Java EE aplikacijskih poslužitelja
 - Transakcije, sigurnost, postojanost
- **Komponentni model**
 - Razdvajanje razvojnih odgovornosti
 - Ponovno korištenje programskog koda
 - Dijeljenje koda između aplikacija
- **Sposobnost zajedničkog rada**
 - Standardni komunikacijski protokoli

Aplikacijski poslužitelj

- **Programska platforma za izvođenje poslužiteljskih aplikacija**
 - Analogija: kao operacijski sustav na osobnom računalu
- **Svrha:**
 - **Programska osnova za kompleksne aplikacije ili cijele sustave**
 - **Isporuka aplikacija ili dijelova aplikacija klijentima**
 - **Sadrži (većinu) poslovne logike i upravlja pristupom podacima (bazi podataka) – centralizacija**
- **Najčešće koriste HTTP protokol i Internet**
 - **Ne miješati s Web poslužiteljima – oni “samo” isporučuju statički sadržaj**

Aplikacijski poslužitelj #2

- Web poslužitelj obično postoji uz aplikacijski poslužitelj i služi za:
 - **Isporuku statičnih sadržaja** prema klijentu (npr. dizajn stranica, slike, ...)
 - **Prosljeđivanje** dinamičkih sadržaja primljenih od aplikacijskog poslužitelja prema klijentu
- Neke najčešće uporabe aplikacijskih poslužitelja:
 - **Web aplikacije**
 - **Upravljanje sadržajem**
 - **Elektroničko poslovanje** (*e-business*)
 - **Integracijska infrastruktura i povezivanja** (*interfacing*) s drugim aplikacijama/sustavima
 - **Posredničke tehnologije i međuprogrami** (*middleware*)

Poznatiji aplikacijski poslužitelji



- **Komercijalni Java EE** aplikacijski poslužitelji:
 - **WebSphere** Application Server (IBM)
 - **Oracle** Application Server (Oracle)
 - **WebLogic** Server (BEA, kupio Oracle)
 - **SAP** Web Application Server (SAP)
 - **JRun** (Adobe)
- **Open-source Java EE** aplikacijski poslužitelji:
 - **JBoss** (Red Hat)
 - **WebSphere** Application Server **Community Edition** (IBM)
 - **Glassfish** Application Server (Sun Microsystems)
 - Apache **Geronimo** (Apache Software Foundation)
- **Ostali** aplikacijski poslužitelji (koji nisu temeljeni na Javi):
 - **.NET framework** (Microsoft)
 - ...

Aplikacijski poslužitelj - prednosti



- Prednosti:
 - Korištenje niza **postojećih** funkcionalnosti koje su već ugrađene na samom aplikacijskom poslužitelju
 - Aplikacije se **grade od** postojećih (ponovno korištenje) i novih funkcionalnih (programskih) **blokova**
 - **Integritet** podataka i programskog kôda - jedno mjesto
 - **Centralizacija** poslovne logike na poslužitelju
 - Praktički **nepotrebna instalacija** na klijentu
 - Praktički **nepotrebna administracija** klijenta
 - **Centralizirana konfiguracija**
 - **Centralizirane promjene**
 - **Sigurnost** pristupa i promjena podataka
 - Povećana **učinkovitost** (*cluster, high-availability*)

Java EE – aktualna specifikacija



- Java Platform, Enterprise Edition 5 Specification
 - Java EE 5 Specification – JSR 244
 - Dovršena 11.05.2006.
 - Ekspertni tim:
 - Bill Shannon - vođa (Sun Microsystems) + niz pojedinaca
 - Niz tvrtki: BEA, Borland, E.piphany, Hewlett-Packard, IBM, Inria, Ironflare, Novell, Oracle, Pramati, Red Hat, SAP, SeeBeyond, Sun Microsystems, Sybase, Tmax Soft, Trifork
 - Prethodne specifikacije:
 - J2EE 1.2 (prosinac 1999.) - razvijena od Sun-a
 - J2EE 1.3 (rujan 2001.) i J2EE 1.4 (studen 2003.) - razvijene pod JCP
- Dio navedenih aplikacijskih poslužitelja je Java EE 5 kompatibilan
 - Posjeduje Java **Compatible**, Enterprise Edition oznaku
 - Prošli su skupinu testova (CTS)



Java EE aplikacijski poslužitelji



- Trenutno **kompatibilne** implementacije Java EE 5 poslužitelja:
 - **Apache** Geronimo-2.1
 - **BEA** WebLogic Server v10.0
 - **IBM** WebSphere Application Server Community Edition (WASCE) 2.0
 - **Kingdee** Apusic Application Server v5.0
 - **Oracle** Application Server 11
 - **SAP** NetWeaver 7.1
 - **Sun** Java System Application Server Platform 9
 - **TmaxSoft** JEUS 6
 - **GlassFish** Application Server



<http://java.sun.com/javaee/overview/compatibility.jsp>



Java EE – komponente



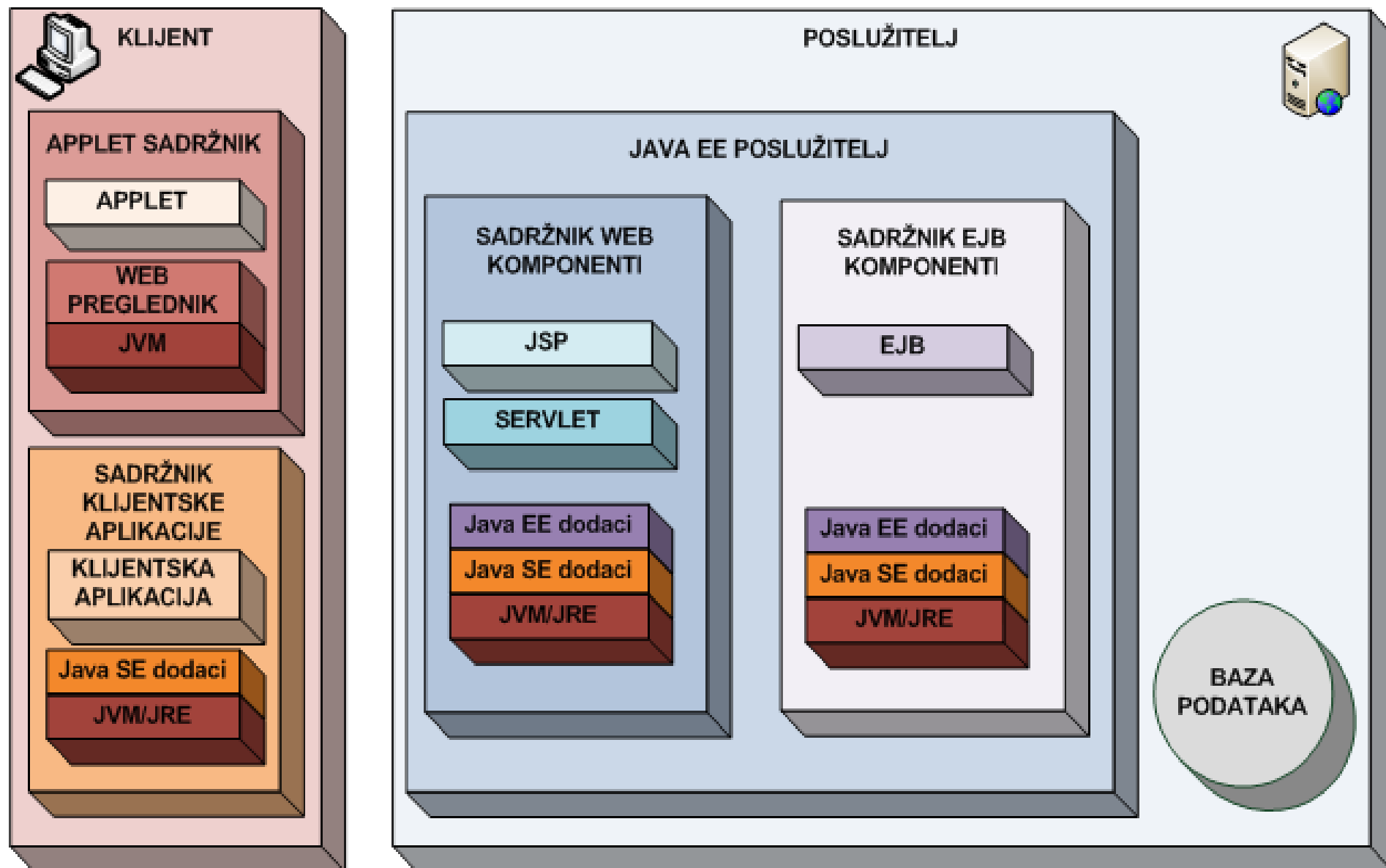
- **Appleti**
 - Komponente s grafičkim korisničkim sučeljem najčešće izvođene unutar Web preglednika
 - Izvršavaju se u JVM plug-inu Web preglednika
- **Klijentske aplikacije**
 - Aplikacije koje se izvode na klijentima i mogu pristupati komponentama na poslužiteljima
 - Izvršavaju se u JVM na klijentu – JRE
- **Web komponente**
 - Servleti i JavaServer Pages
 - Izvršavaju se u JVM na Java EE aplikacijskom poslužitelju
- **Enterprise JavaBeans**
 - Raspodijeljene transakcijske komponente poslovne logike i pristupa podacima
 - Izvršavaju se u JVM na Java EE aplikacijskom poslužitelju

Java EE – komponente i sadržnici



- Svaka komponenta se nalazi i izvršava u nekom sadržniku (*container*)
- **Applet sadržnik** – izvršava Applete u JVM plug-inu Web preglednika
- **Sadržnik klijentske aplikacije** – izvršava klijentske aplikacije (GUI klijent) u JVM (JRE) na klijentu
- **Sadržnik Web komponenti** – izvršava Servlete i JSP-ove u JVM na Java EE aplikacijskom poslužitelju
- **Sadržnik Enterprise JavaBean-ova** – izvršava EJB komponente u JVM na Java EE aplikacijskom poslužitelju

Java EE – komponente i sadržnici #2



Java EE dijelovi (paketi)

- **Java EE platforma**
- **Tehnologije Web aplikacija**
 - **Java Servlet**
 - **JavaServer Pages (JSP)** i JSP Standard Tag Library (JSTL)
 - JavaServer Faces (JSF)
- **Tehnologije upravljanja i sigurnosti**
 - Java EE Application Deployment
 - Java EE Management
- **Tehnologije Web usluga (servisa)**
 - Enterprise Web Services
 - Java API for XML-Based Web Services (JAX-WS)
 - Java API for XML-Based RPC (JAX-RPC)
 - SOAP with Attachments API for Java (SAAJ)
 - JAXB, StAX, ...

Java EE dijelovi (paketi) #2

- **Tehnologije za *enterprise* aplikacije**
 - Enterprise JavaBeans (EJB)
 - Java Message Service (JMS)
 - Java Transaction API (JTA)
 - Common Annotations
 - J2EE Connector Architecture (JCA)
 - JavaBeans Activation Framework (JAF)
 - JavaMail
 - Java Persistence API i Java Data Objects (JDO)
- **Tehnologije za upravljanje i sigurnost**
 - J2EE Application Deployment
 - J2EE Management
 - Java Authorization Contract for Containers

Otkuda krenuti u sve ovo???

- Mi ćemo krenuti od - tehnologija za izradu Web aplikacije 😊
- Tehnologije za izradu Web aplikacija mogu služiti za izradu **cijelih** aplikacija
 - Npr. u prezentacijskom sloju se može pristupiti bazi podataka
 - **Ali nisu tako zamišljene!**
- Tehnologije za izradu Web aplikacija služe:
 - Za izradu **prezentacijskog sloja i sloja interakcije s korisnikom** putem korisničkog sučelja
- Dodatni razlog – treba nam za 5. lab. vježbu 😊

Tehnologije za izradu Web aplikacija

Servleti

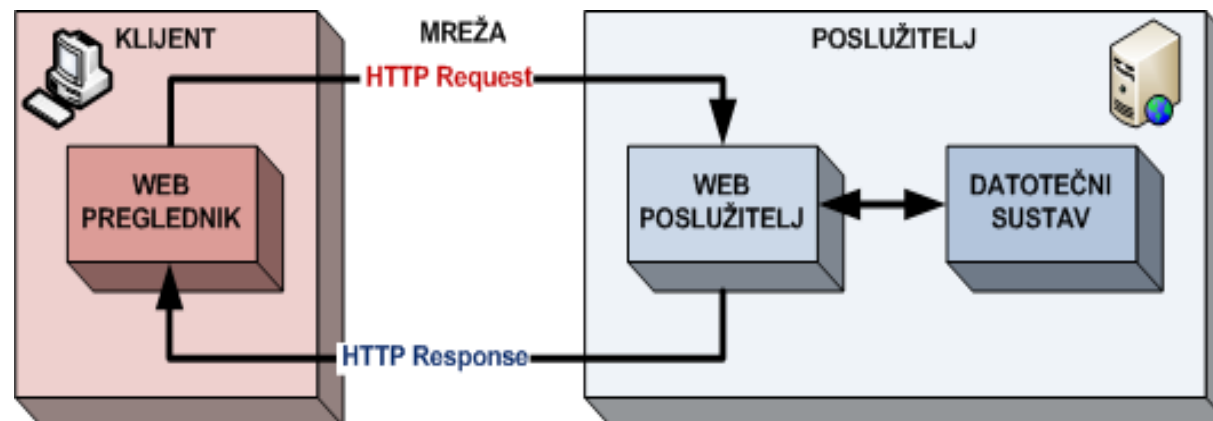
Malo ponavljanja

- Dosad znamo što je/su:
 - Web stranice i Web sjedišta
 - URL, resurs, HTTP protokol
 - Web preglednik
 - Web poslužitelj
 - Statičke Web stranice
 - Dinamičke Web stranice (npr. PHP, CGI)
- A sada ćemo pogledati i kako to izgleda kod kompleksnih višeslojnih aplikacija

Malo ponavljanja – statičke stranice



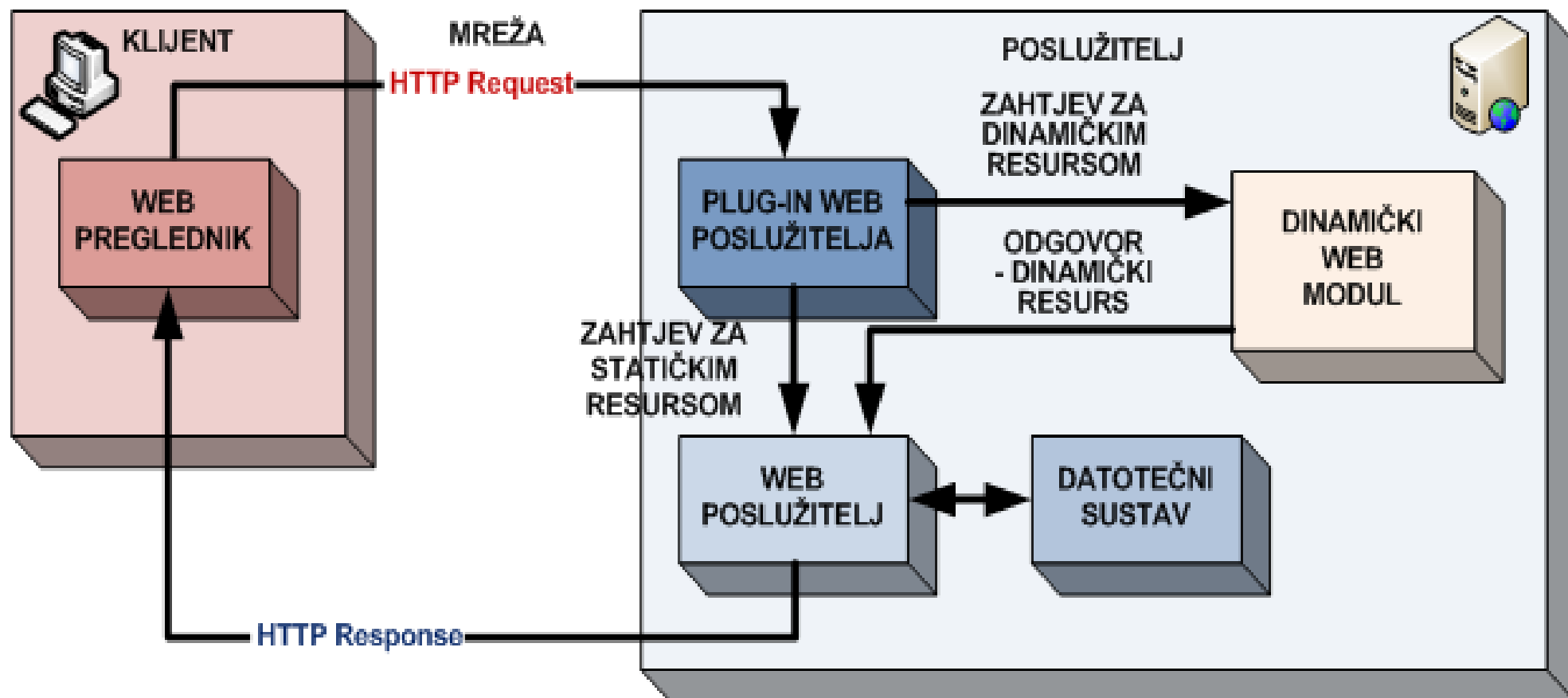
- Web preglednik (klijent) zatraži određenu Web stranicu
- Protokol HTTP, vrata (port) 80
- Pitanje – odgovor: HTTP Request – HTTP Response
- Web poslužitelj za traženi URL učitava s datotečnog sustava (diska) stranicu
- Web poslužitelj pošalje stranicu Web pregledniku



Dinamičke stranice

- Početak komunikacije od Web preglednika – **isti**
- Na poslužitelju:
 - Dodatni **dinamički Web modul**
 - Npr. Servlet engine ili Servlet container ili Web container
 - Odluka: da li je zatražena dinamička ili statička stranica?
 - Ako statička – **isto** kao i prije
 - Ako dinamička – **poziv dinamičkom Web modulu**
- Odluke su **konfigurabilne** – konfiguracijska datoteka
- “Posao” dinamičkog Web modula
 - **Generiranje** dinamičke stranice
 - **Prosljeđivanje** Web poslužitelju
- Web poslužitelj vraća generiranu dinamičku stranicu Web pregledniku – **isto** kao i prije

Dinamičke stranice #2



Što su Servleti?





- Java razredi, odnosno objekti koji se nalaze na poslužitelju
- Odgovaraju na zahtjev (Request) odgovorom (Response) korištenjem protokola HTTP
- Alternativa “zastarjelim” CGI programima
- Ovisno o ulaznim parametrima koje je poslao Web preglednik može se generirati različita stranica
- Povratna informacija je dinamički generirana HTML stranica
 - Stranica se generira unutar programskog koda
- Java Servlet API – sučelje za rad sa Servletima

Servlet – definicija

- Standardne poslužiteljske Java EE komponente
 - *“A servlet is a **Web component**, managed by a **container**, that generates dynamic content. Servlets are **small, platform-independent Java classes** compiled to an architecture-neutral bytecode that can be loaded dynamically into and run by a Web server”*
- Sadrže poslovnu logiku koja odgovara na HTTP Request – odgovaraju na GET i POST metode
- Pokreće i održava ih poslužitelj – zapravo Web sadržnik (*container*) poslužitelja

Napomena: **Servlet sadržnik (*container*)** i **Web sadržnik (*container*)** – oboje predstavlja **dinamički Web modul** “spojen” na Web poslužitelj, odnosno **dio aplikacijskog poslužitelja** koji služi za upravljanje dinamičkim sadržajima (Servleti i JSP-ovi)

Servlet – primjer 1 – pretraga sadržaja

- Web pretraživanje s obrascem
 - Dohvat statičke stranice s obrascem pretrage
 - Request: GET search.html - Response: HTML dokument “search.html” s podatkovnog sustava poslužitelja (diska)
- 
- Upis uvjeta pretrage u stranicu i potvrda na gumb
 - Request: POST /SearchServlet- Response: generirani HTML dokument u ovisnosti o ulaznim parametrima (uvjet pretrage)


Servlet – primjer 1 – HTML obrazac



- HTML kod za poziv servleta:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>Pretraga</title>
  </head>
  <body>
    <form action="/SearchServlet" method="POST">
      <h1>Pretraga</h1>
      <strong>Upišite uvjet pretrage:</strong>
      <input type="text" size="30" name="uvjet"><br>
      <input type="submit" value="Traži"><br>
    </form>
  </body>
</html>
```

Servlet – primjer 1 – komunikacija



- Komunikacija:

preglednik – poslužitelj

- Request: POST metoda i URL servleta
- Zaglavlje
- Prazna linija
- Predani podaci (argumenti, parametri)

- Sadržaj poziva:

POST /SearchServlet HTTP/1.0

Referer: http://www.fer.hr/search.html

Connection: Keep-Alive

User-Agent: Mozilla/4.72 [en] (WinNT 5.0; U)

Host: localhost:8080

Cookie: USERID=spot

Accept: image/gif, image/x-xbitmap, image/jpeg, */*

Accept-Language: hr

Accept-Charset: iso-8859-2,*,utf-8

Content-type: application/x-www-form-urlencoded

Content-length: 50

uvjet=Java

Servlet – primjer 1 – komunikacija



- Komunikacija:
poslužitelj – preglednik

- Response: statusna informacija
- Zaglavlje
- Prazna linija
- HTML dokument

- Sadržaj:

HTTP/1.1 200 ok

Content-Type: text/html

Set-Cookie:

sessionid=5H2HXGYAAAAAEWQAAAA
ZJCl;Path=/

Cache-Control: no-cache="set-cookie,set-
cookie2"

Expires: Thu, 01 Dec 1994 16:00:00 GMT

Set-Cookie: USERID=spot; Expires=Fri,
08-Jun-2001 21:54:37 GMT

Content-Language: hr

<HTML><BODY>

<H1>Najjednostavniji HTML
dokument</H1>

</BODY></HTML>

Servlet – koraci komunikacije

- Preglednik poziva Servlet na poslužitelju – URL
 - Ime servleta je dio URL-a
- Web poslužitelj prima zahtjev i zaključuje da se radi o Servletu
- Web poslužitelj prosljeđuje poziv dinamičkom Web modulu (Web sadržniku)
- Pronalazi se postojeća ili se kreira nova instanca (objekt) Servlet razreda istog imena
- Poziva se metoda Servlet objekta
- Servlet generira HTML kôd i vraća ga Web poslužitelju
- Web poslužitelj prosljeđuje HTML kod pregledniku

Servlet - izgradnja

- Izgradnja razreda koji nasljeđuje **javax.servlet.http.HttpServlet**
- Izgradnja (nadjačanih) **doGet()** i/ili **doPost()** metoda
 - Procesuiranje ulaznih parametara iz dobivenog **HttpServletRequest** objekta
 - Izgradnja odgovarajućeg poslovnog procesa
 - Izvršavanje metoda poslovne logike
 - Izgradnja HTML kôda
 - Postavljanje odgovarajućih vrijednosti na **HttpServletResponse** objekt
 - Prosljeđivanje (generiranog) HTML kôda **PrintWriter** objektu
 - Hvatanje iznimaka (IOException, ServletException)

Servlet – primjer 2

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;
```

```
public class SimpleServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String browser = request.getHeader("User-Agent");
        response.setStatus(HttpServletResponse.SC_OK);
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("<HTML><BODY>");
        out.println("Koristite preglednik: " + browser);
        out.println("</BODY></HTML>");

    }
}
```

Servlet – općenito

- Pokazano je da svaki Servlet nasljeđuje HttpServlet razred – ali to se odnosi samo na Servlete koji komuniciraju HTTP protokolom s okolinom
- Općenito Servlet mora samo implementirati **javax.servlet.Servlet** sučelje
 - Definira 5 metoda:
 - void **destroy()** - zove je sadržnik kada uništava Servlet
 - ServletConfig **getServletConfig()** - vraća ServletConfig object, koji sadrži određene parametre potrebne za rad Servleta
 - java.lang.String **getServletInfo()** - vraća informaciju o Servletu
 - void **init**(ServletConfig config) - zove je sadržnik prilikom učitavanja Servleta
 - void **service**(ServletRequest req, ServletResponse res) - zove je sadržnik kako bi omogućio Servletu da odgovori na zahtjev

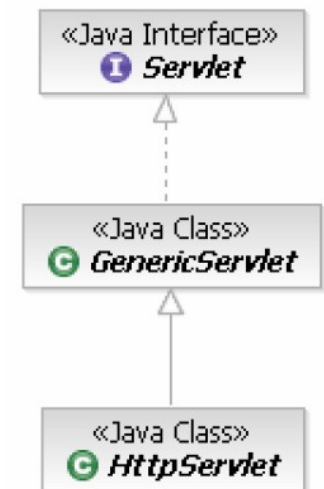
Servlet API

- Skup Java razreda i (sučelja) koji definiraju standardno sučelje između poziva Web preglednika i odgovora Web poslužitelja
- Paketi:
 - **javax.servlet** – generična podrška za Servlete
 - **javax.servlet.http** – podrška za Servlete koji komuniciraju protokolom HTTP
- ServletConfig referenca
 - Prosljeđuje Servletu podatke o okruženju u kojem radi
 - Predaje se prilikom inicijalizacije - u metodi init()
 - Sadrži podatke u obliku: naziv – sadržaj
 - Sadrži ServletContext referencu

Servlet API #2

- Ostale koristi od postojećih Servlet razreda:
 - Dekodiranje ulaznih podataka
 - Npr. ime=predavanje+o+Javi&trajanje=3+sata se dekodira u parove naziv - sadržaj:
 - ime – predavanje o Javi
 - trajanje – 3 sata
 - Rješava duplikate vrijednosti – višestruke vrijednosti

- Nasljeđivanje:
 - Sučelje javax.servlet.**Servlet**
 - Apstraktni razred javax.servlet.**GenericServlet**
 - Apstraktni razred javax.servlet.http.**HttpServlet**



HttpServlet

- Razred HttpServlet je **apstraktni** razred koji se mora naslijediti kako bi se stvorio HTTP servlet
- Podklasa razreda HttpServlet mora nadjačati (*override*) barem jednu od sljedećih metoda:
 - **doGet()** - ako Servlet treba podržati GET metodu
 - **doPost()** - ako Servlet treba podržati POST metodu
 - **doPut()** - ako Servlet treba podržati PUT metodu
 - **doDelete()** - ako Servlet treba podržati DELETE metodu
 - **init()** i **destroy()** - ako Servlet upravlja resursima prije/nakon njegovo životnog vijeka
 - **getServletInfo()** - ako Servlet mora isporučiti informaciju o sebi

Servlet – neka objašnjenja

- Zašto se ne nadjačava metoda **service()**?
 - Nema potrebe za nadjačavanjem `service()` metode, jer ona prosljeđuje HTTP zahtjeve odgovarajućim `doXXX()` metodama ovisno o HTTP metodi u zahtjevu (npr. `doPost()` za POST metodu)
- Zašto postoji metoda **init()**, a ne konstruktor?
 - Povijesno, konstruktori za dinamički stvorene objekte (kakvi su Servleti) ne mogu primiti argumente, a Servletu se prosljeđuje objekt koji implementira `ServletConfig` sučelje i sadrži informacije o okruženju
 - `javax.servlet.Servlet` je sučelje, pa ne može deklarirati konstruktor sa `ServletConfig` argumentom, već deklarira metodu `init()`
- Zašto postoji metoda **destroy()**?
 - Kako bi Servlet objekt mogao osloboditi sve resurse koji se ne mogu/znaju automatski očistiti (*garbage collection*), a mogu se i zapisati sve informacije koje je potrebno sačuvati.

Životni ciklus Servleta

- Servlet se po potrebi učitava, instancira, ali i uništava
 - Pri pokretanju poslužitelja ili
 - Kad poslužitelj zaključi da je to potrebno
- Servleti se izvršavaju u JVM Web sadržnika (*container*)
 - Web sadržnik (a ne programer) **brine** kad će **stvoriti** novi Servlet objekt ili **uništiti** postojeći
- Zašto uopće imamo više instanci Servleta?
 - Zato jer istovremeno može **više korisnika** pristupiti aplikaciji i svaki želi da se njegov **zahtjev obradi** (trenutno, odnosno čim prije može)

Servlet – višekorisnički rad

- Smisao svake Web aplikacije je da joj može istovremeno pristupiti veći broj korisnika
 - Aplikacije i sustavi se “grade” kako bi zadovoljile određeni (predviđeni) broj konkurentnih korisnika
 - Svaki korisnik pristupa svom skupu resursa koji rješavaju određenu funkcionalnost
- Ako više korisnika pristupi istom Servletu, na poslužitelju, odnosno Web sadržniku je da svakom korisniku pruži jednu instancu Servleta te može:
 - Iskoristiti postojeću, ako je objekt stvoren i nitko ga ne koristi
 - Stvoriti novi objekt, ako nema dovoljno već stvorenih
 - Princip višenitnog poslužitelja i bazena (*pool*) resursa

Servlet – konfiguracija

- Aplikacijski poslužitelj se konfigurira kako bi znao da Servlet **postoji** i kako bi ga mogao **pokrenuti**
 - Konfiguracija zapisana kao XML datoteka **web.xml** (tzv. Web Application Deployment Descriptor)
 - Bit će detaljno kasnije objašnjena ... zasad samo ukratko

<web-app>

- korijenski element

...

<servlet>

- deklaracija Servleta (za svaki Servlet)

...

</servlet>

...

<servlet-mapping>

- definicija mapiranja Servleta na URL

...

</servlet-mapping>

...

</web-app>

Servlet – konfiguracija #2

- Deklaracija Servleta sadrži:
 - **Naziv** samog Servleta
 - **Naziv razreda** koji implementira Servlet
 - Uputa za poredak učitavanja pri pokretanju (opcionalno)
 - Naziv Servleta za prikaz (opcionalno)
 - Niz drugih stvari (npr. inicijalizacijski parametri) (opcionalno)

`<servlet>`

`<servlet-name>MySearchServlet</servlet-name>`

`<servlet-class>hr.fer.or.SearchServlet</servlet-class>`

`<load-on-startup>1</load-on-startup>`

`<display-name>Moj Servlet</display-name>`

`</servlet>`

- Napomena: ovo postoji za svaki Servlet koji se nalazi u Web sadržniku

Servlet – konfiguracija #3

- Definicija mapiranja Servleta sadrži:
 - **Naziv** samog Servleta
 - Mora biti isti naziv kao u deklaraciji!!!
 - **URL** koji odgovara pozivom Servleta
 - Može biti proizvoljan
 - Kada dođe zahtjev za resursom koji je označen ovim URL-om Web sadržnik prosljeđuje zahtjev tom Servletu
 - ...

</servlet-mapping>

<servlet-name>MySearchServlet</servlet-name>

<url-pattern>/servlets/SearchServlet</url-pattern>

</servlet-mapping>

- Napomena: ovo postoji za svaki Servlet koji se nalazi u Web sadržniku

Servlet – konfiguracija – ponavljanje

`<web-app>`

- korijenski element

...

`<servlet>`

- deklaracija Servleta

`<servlet-name>MySearchServlet</servlet-name>`

`<servlet-class>hr.fer.or.SearchServlet</servlet-class>`

`<load-on-startup>1</load-on-startup>`

`<display-name>Moj Servlet</display-name>`

`</servlet>`

...

`</servlet-mapping>`

- definicija mapiranja Servleta na URL

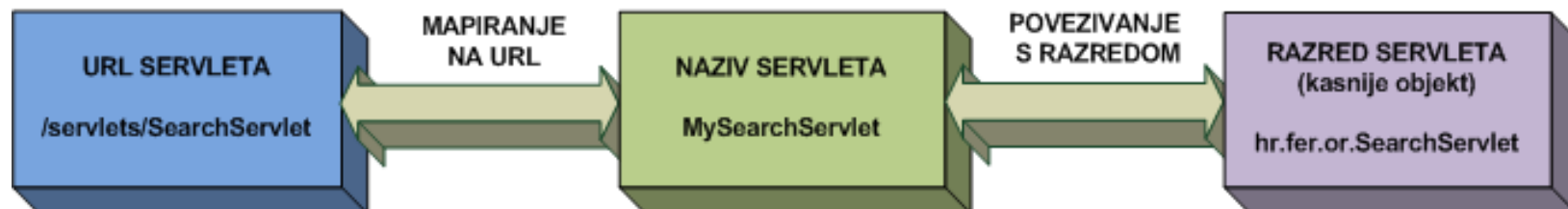
`<servlet-name>MySearchServlet</servlet-name>`

`<url-pattern>/servlets/SearchServlet</url-pattern>`

`</servlet-mapping>`

...

`</web-app>`



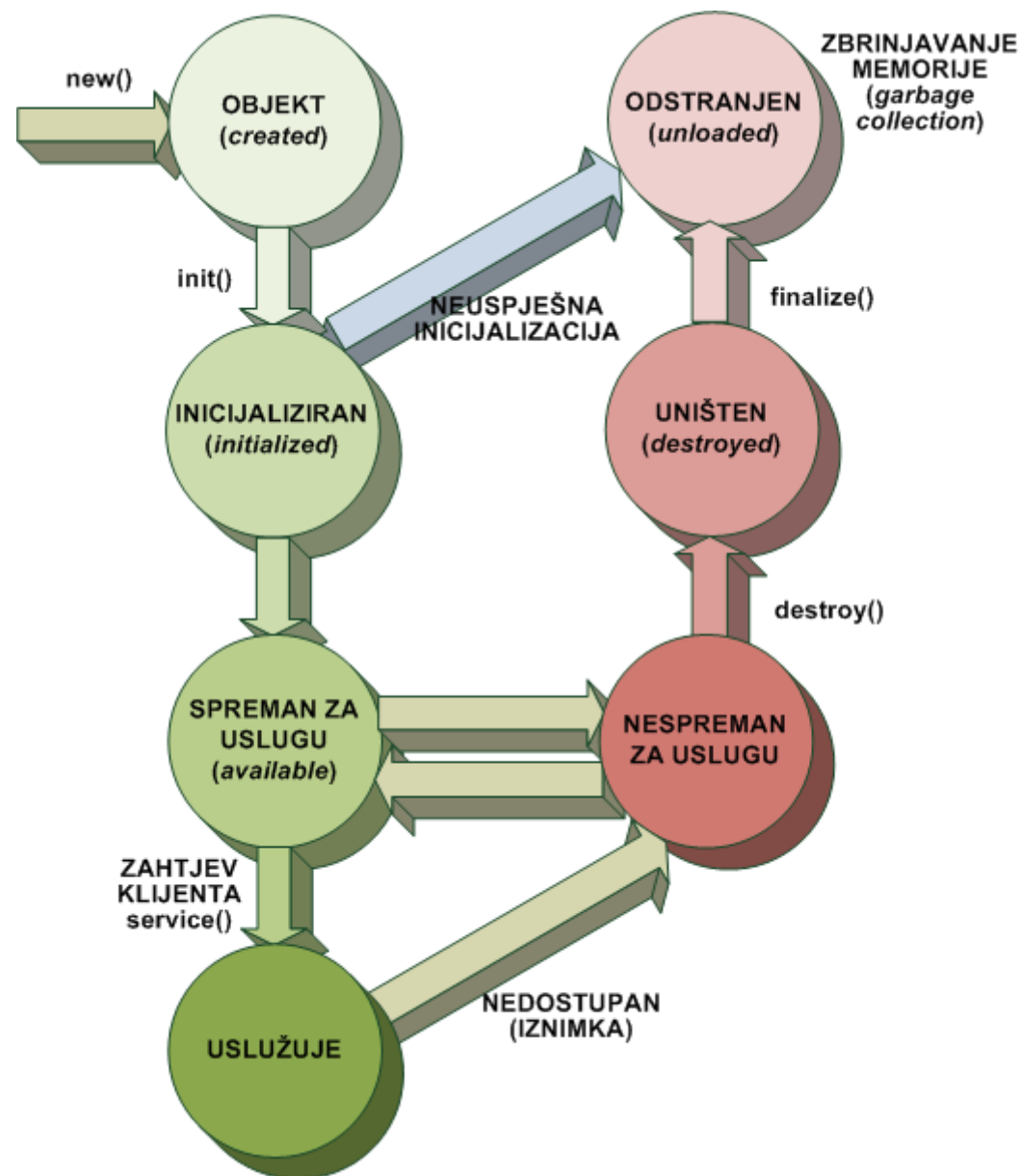
Životni ciklus Servleta – kreiranje



- Kada će Web sadržnik kreirati instancu Servleta?
 - Kada mu se eksplicitno naredi da unaprijed učitava Servlet
 - Konfiguracija u web.xml
 - U oznaci `<load-on-startup>` upisati vrijednost veću od 0
 - Servlet je učitani i spreman za uporabu kada dođe zahtjev
 - Ako ne postoji učitani Servlet, onda će se kreirati instanca kada dođe novi zahtjev za Servletom
- Prilikom učitavanja Servleta u memoriju, Web sadržnik pokreće inicijalizaciju

Životni ciklus Servleta – inicijalizacija

- Inicijalizacija – **init()** metoda
 - Poziva se samo jednom
 - Služi za:
 - Učitavanje parametara
 - Npr. opće postavke
 - Inicijalno konfiguriranje
 - Podešavanje kako će Servlet raditi
 - Npr. kodna stranica i sl.
 - Veze prema resursima
- Postoje 2 init() metode:
 - Bez parametara
 - S ulaznim parametrom ServletConfig
 - Dostup do okruženja u kojem se Servlet izvršava



Životni ciklus Servleta – zahtjev



- Ako je inicijalizacija uspješna Servlet je **spreman za uslugu** (*available for service*)
 - Ako nije Servlet se **odstranjuje iz memorije** (*unload*)
- **Upravljanje zahtjevom**
 - Kada Web sadržnik dobije zahtjev za Servletom, **prosljeđuje** ga **service()** metodi Servleta
 - Kod HTTP zahtjeva nadogradnja **service()** metode s metodama **doXXX()** ovisno o HTTP metodi zahtjeva:
 - **doGet()** - odgovara na zahtjev poslan metodom **GET**
 - **doPost()** - odgovara na zahtjev poslan metodom **POST**
 - **doOptions()** - odgovara na zahtjev poslan metodom **OPTIONS**
 - **doPut()** - odgovara na zahtjev poslan metodom **PUT**
 - **doDelete()** - odgovara na zahtjev poslan metodom **DELETE**

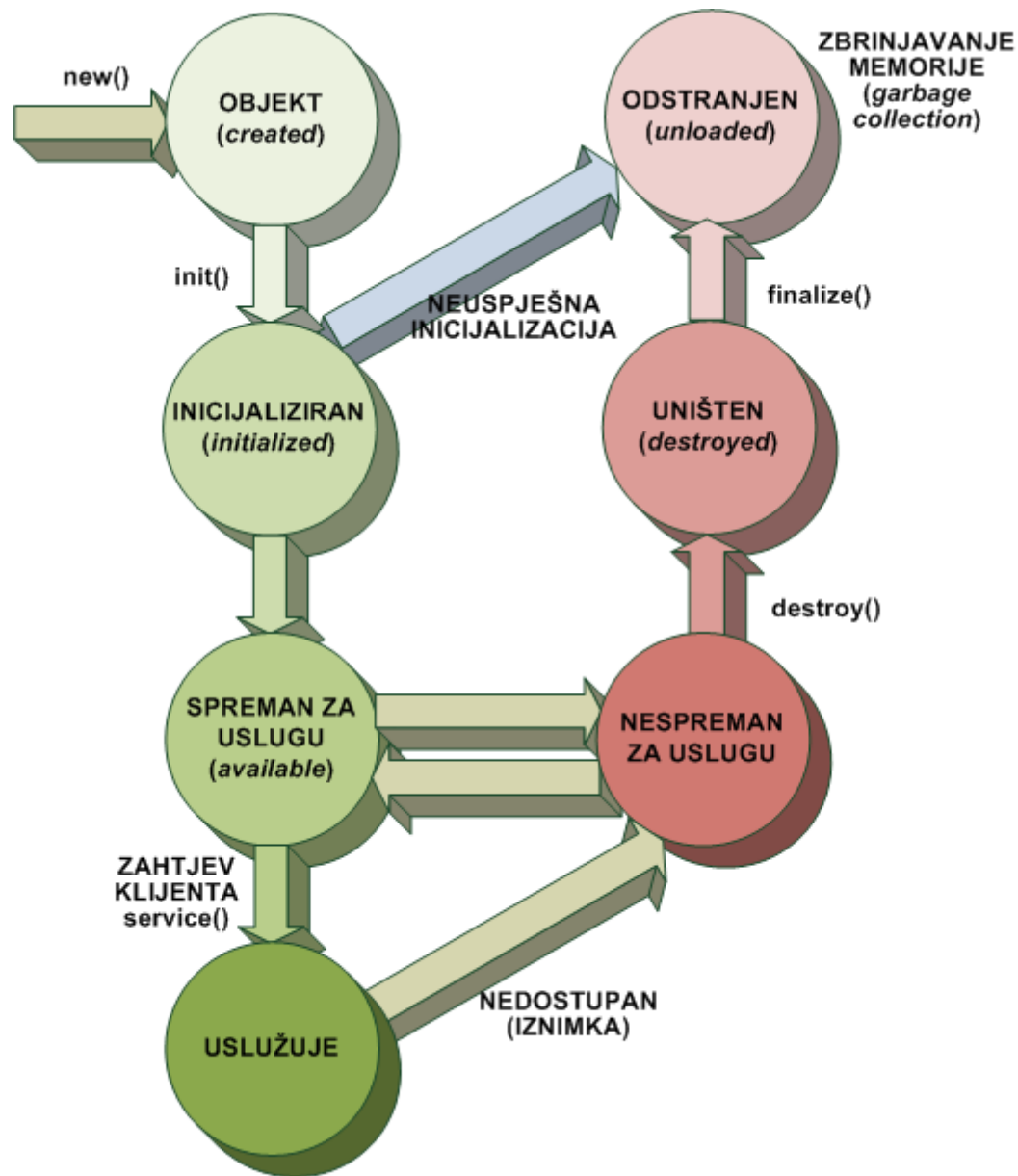
Servlet – obrada zahtjeva

- Servlet prihvata zahtjev od klijenta u metodi **doXXX()** (npr. doPost())
- **Svaki zahtjev – nova nit**
- Metoda doXXX() obrađuje zahtjev i vraća odgovor
 - Preuzima ulazne parametre
 - Odrađuje “posao”
 - Vraća rezultat u HTML obliku

Životni ciklus Servleta – dostupnost



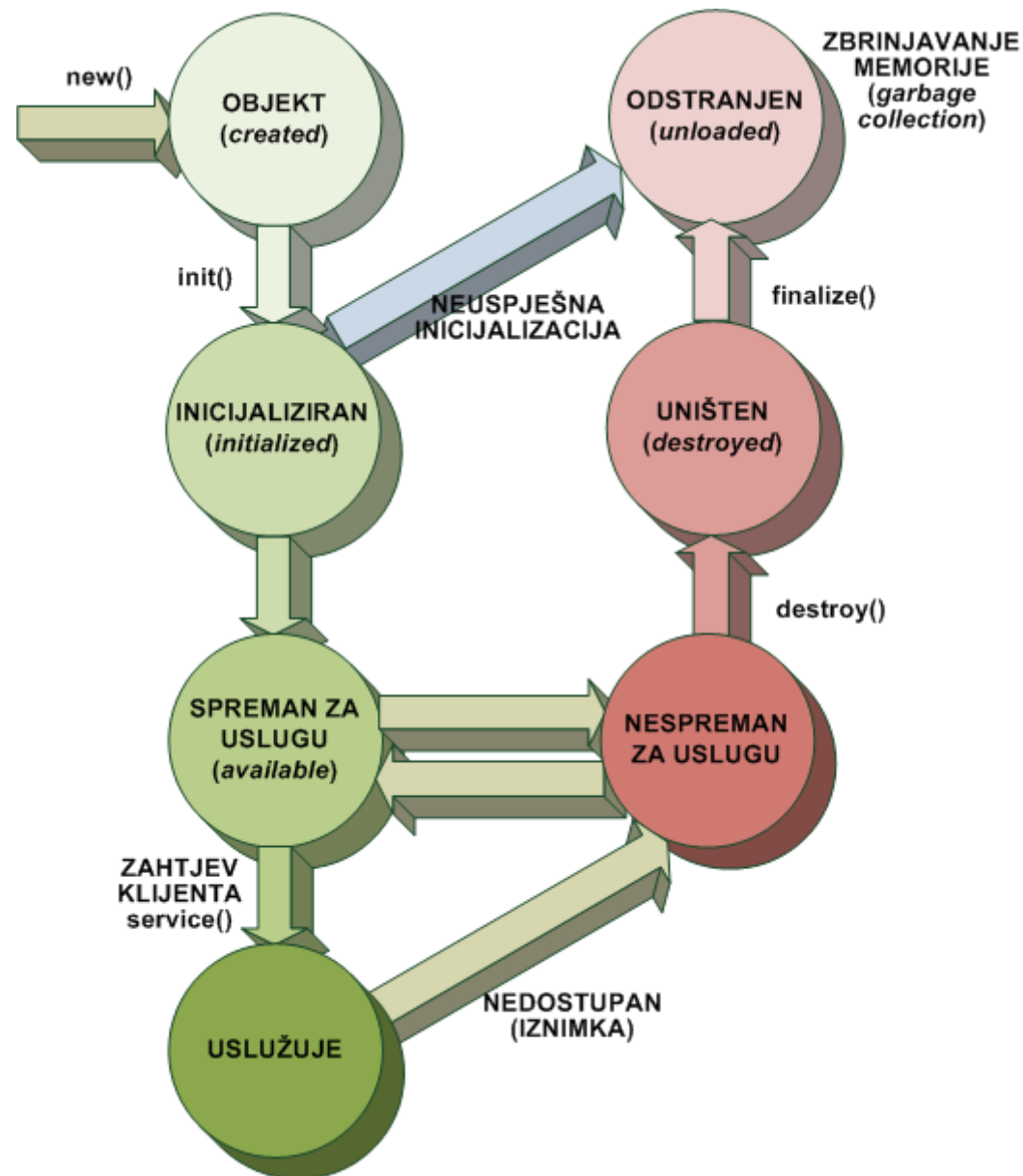
- Servlet može biti nedostupan
 - Kada uslužuje zahtjev
 - Kada je nespreman za uslugu iz nekog drugog razloga
 - Npr. greška
- Ako je Servlet privremeno nedostupan, on može postati ponovno dostupan kada:
 - Usluži zahtjev
 - Razriješi problem (grešku)



Životni ciklus Servleta – završetak



- Servlet se uništava kod:
 - Čuvanja memorijskih resursa
 - Gašenja poslužitelja
 - Greške koja onemogućava njegovu dostupnost
- Poziv metode **destroy()**
 - Provjera da li su sve niti (procesi) završili “posao”
 - Poništavanje svih promjena koje se nisu automatski riješile
- Uništavanje Servleta (*unload*)
 - Zbrinjavanje memorije (*garbage collection*)




Ograničenja Servleta

- **Problem** - Servlet služi za neki “pametni posao”, ali i vraća generirani HTML sadržaj
 - U Servletu se **miješa programski kôd** (poslovna logika) i **HTML kôd** same rezultatne Web stranice (prezentacije)
 - **Programer mora biti i Web dizajner** (i obratno)
 - Realno su to dvije **odvojene** uloge
 - **Miješanje toka programske logike s prezentacijom**
 - **Otežano:**
 - Ponovno korištenje koda,
 - Održavanje
 - Paralelni razvoj (u timu)
 - HTML kôd se **uređuje unutar** programskog (Java) kôda
 - **Nemogućnost korištenja** specijaliziranih **uređivača** HTML kôda
 - Dodatni problem s CSS, JavaScript kôdom i sl.
- **Zaključak: HTML kôd ne bi trebao biti dio programskog kôda**

Rješenje Servlet problema

- **Odvajanje sloja poslovne logike od sloja prezentacije**
 - Korištenje nekih drugih tehnologija osim Servleta:
 - **JavaServer Pages (JSP) tehnologija**
 - **JavaBeans tehnologija**
- **Kako?**
 - Dolazni zahtjev se “čita” i ovisno o tipu zahtjeva se prosljeđuje odgovarajućoj poslovnoj logici
 - Poslovna logika **odrađuje “pametni dio posla”** i stvara rezultate (**podatke**) koji će se vratiti korisniku
 - **Odlučuje** se na koji **način** će se podaci **prikazati** korisniku i **odabire se predložak prikaza** za prezentaciju korisniku
 - **Predložak se “puni” s podacima** i šalje natrag klijentu



Tehnologije za izradu aplikacija Weba

MVC, JSP, JavaBeans

(u nastavku - 2. dio)



Pitanja ?