LINKgroup

Distance Learning System

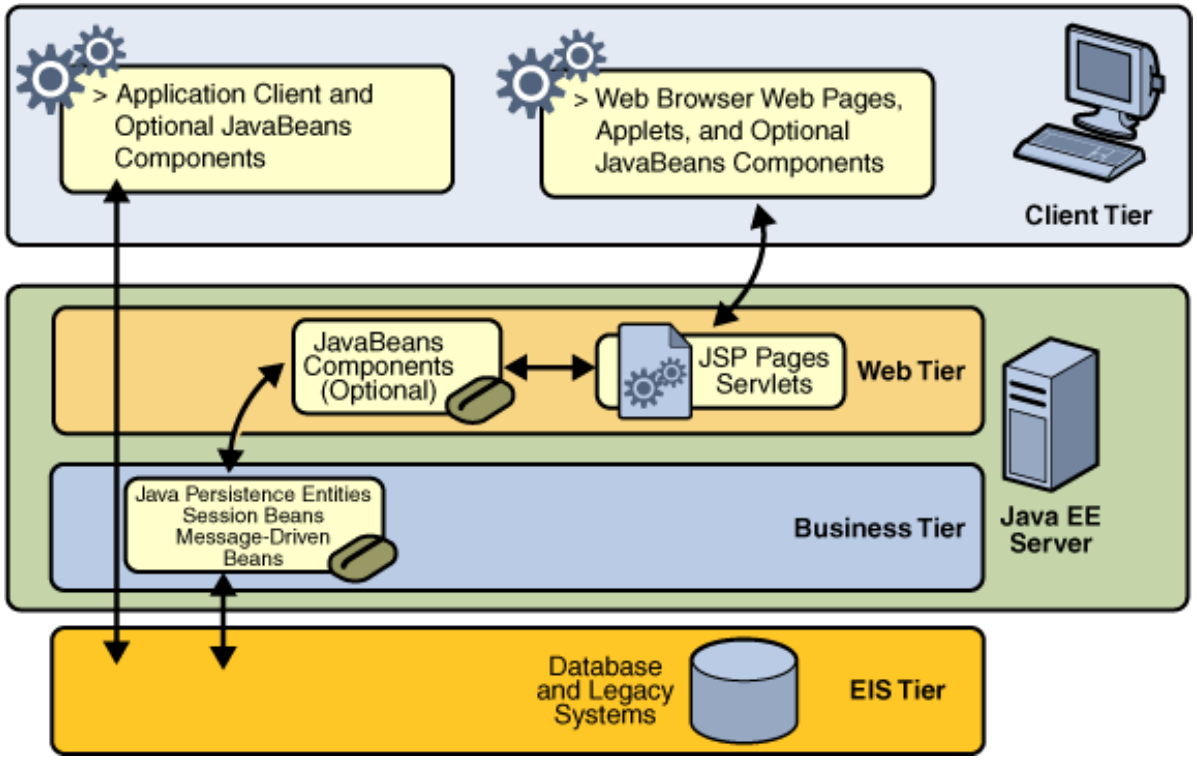
Java Web development

Uvod u Java Web

Šta je Java EE

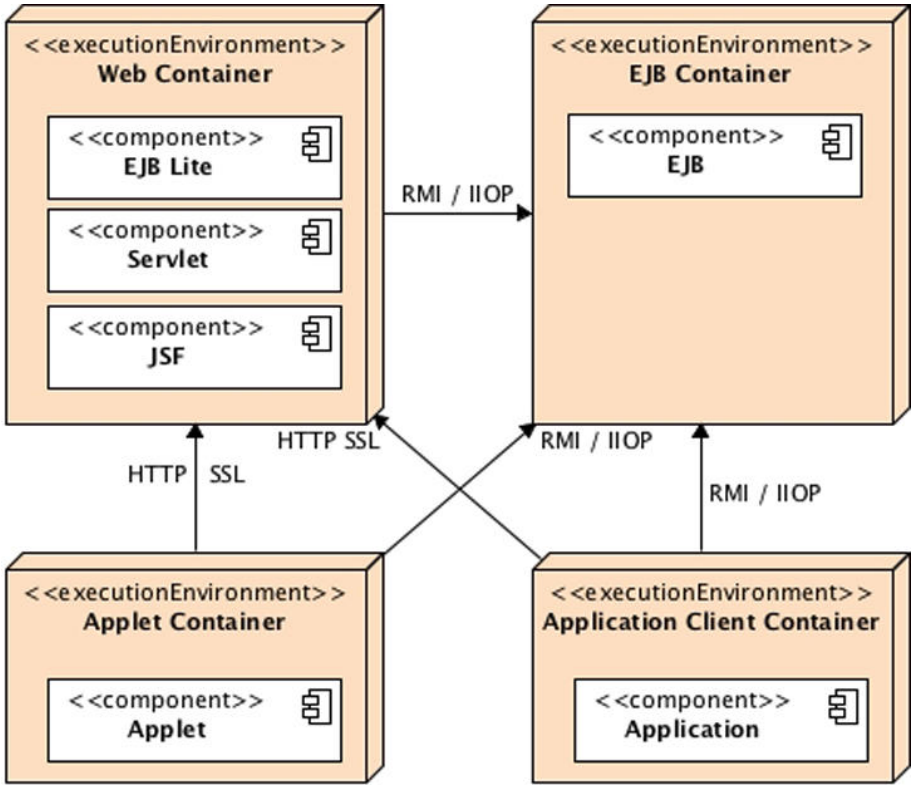
- Java EE (Enterprise Edition) je set biblioteka kojima se proširuje standardna Java funkcionalnost
- Java EE dodaje javi mogućnost korišćenja distribuiranih i nedistribuiranih transakcija (**JTA**), perzistentnih objekata (**JPA**), sistema poruka (**JMS**), web servisa (JAX-WS)
- Glavni fokus Java EE tehnologije su web i enterprise aplikacije

Java EE arhitektura



Java EE arhitektura

- Ultimativni proizvod u Java EE je enterprise aplikacija
- Enterprise aplikacija se sastoji od jednog ili više kontejnera
  - Kontejneri su logičke celine koje sadrže skup komponenti koji obezbeđuju funkcionalnost jednog aplikativnog sloja
  - Kontejneri se mogu smatrati nezavisnim aplikacijama koje uzajamno mogu a ne moraju interagovati

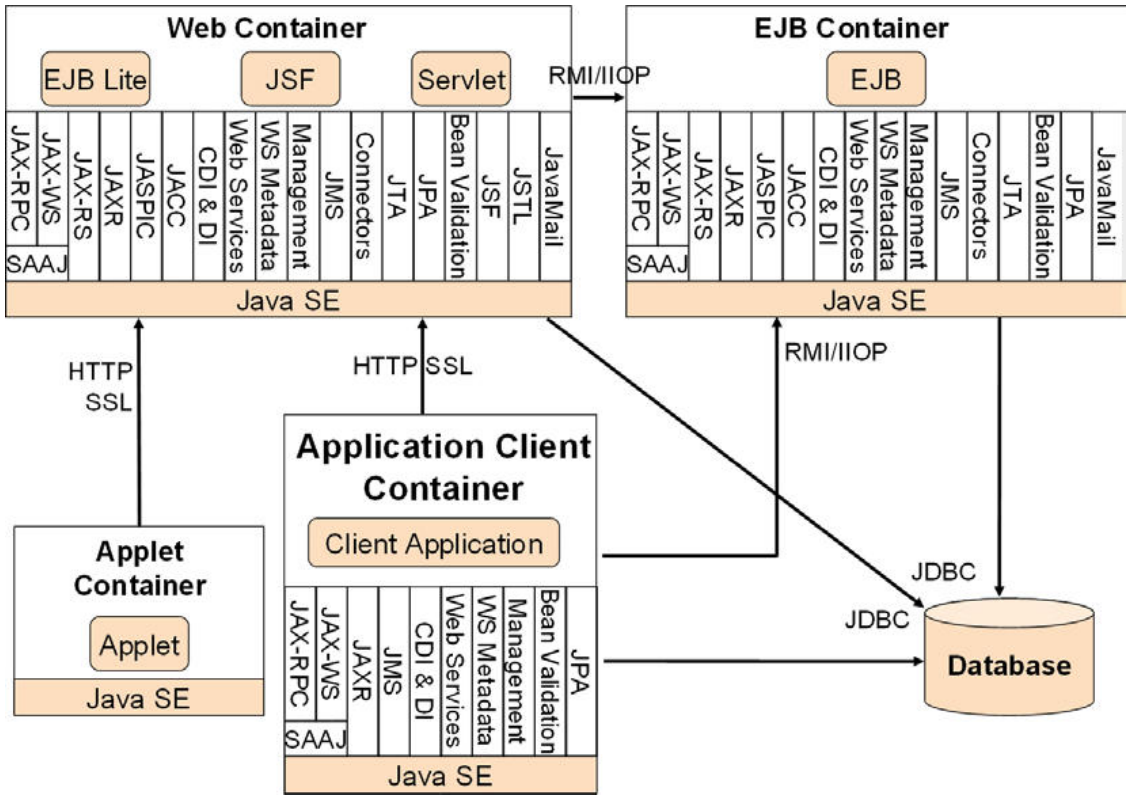


Java EE kontejneri

- Java EE razlikuje četiri vrste kontejnera
  - Applet kontejner
    - Klijentski deo aplikacije (RIA)
  - Application client container (ACC)
    - Stand alone klijentske aplikacije koje imaju jaku vezu sa serverom
  - Web kontejner (standardne web aplikacije)
  - EJB kontejner
    - EJB aplikacije

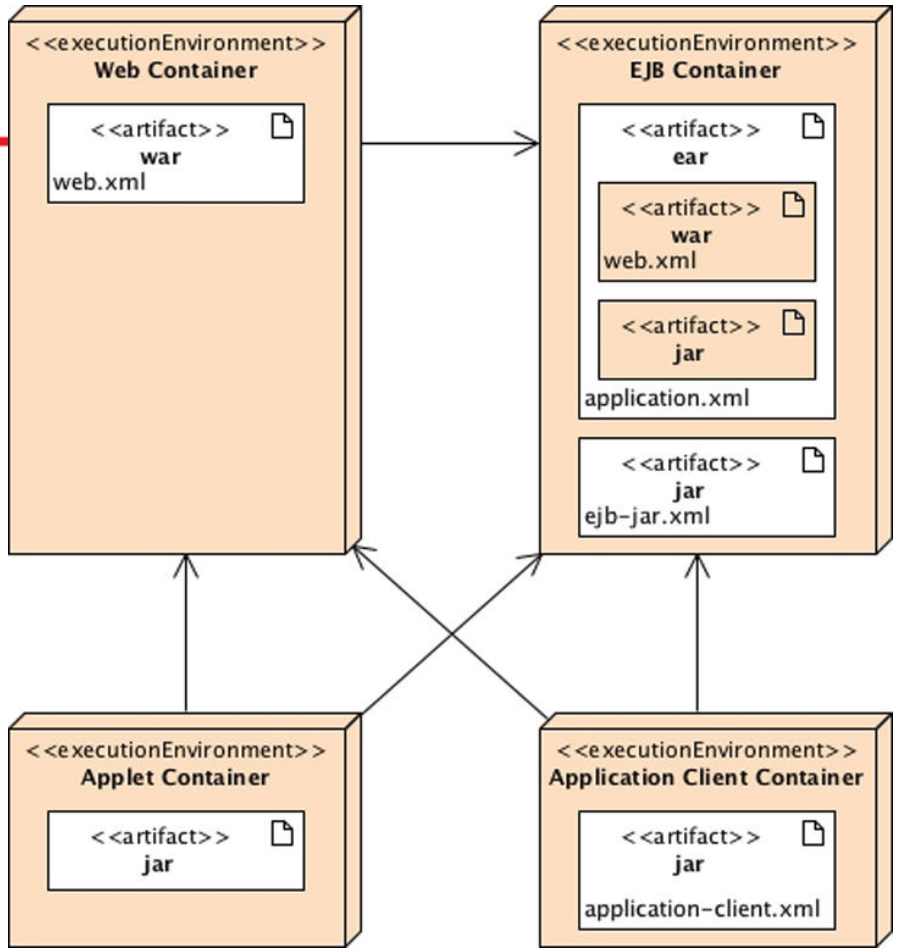
Servisi

- U zavisnosti od mogućnosti kontejnera, na raspolaganju su neki od odgovarajućih servisa



Java EE deploy

- Java EE aplikacije se pakuju na različite načine u zavisnosti od kontejnera na koji se vrši deploy
- Web aplikacije se pakuju u war pakete
  - EJB aplikacije u ear
  - AAC i Applet aplikacije se pakuju u jar
  - Svaka vrsta pakovanja podrazumeva odgovarajući **deployment descriptor**



Deployment descriptori

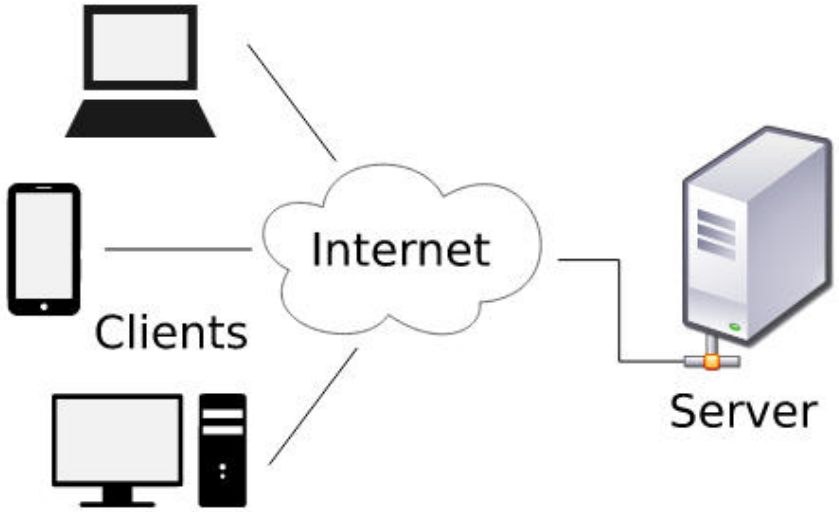
- Deployment descriptori su fajlovi koji sadrže konfiguracione direktive projekta
- Ovi descriptori mogu biti minimalni, ali takođe mogu sadržati kompletan deklarativni deo aplikacije

Java EE deployment descriptor-i

File	Specification	Paths
application.xml	Java EE	META-INF
application-client.xml	Java	EE META-INF
beans.xml	CDI	META-INF or WEB-INF
ra.xml	JCA	META-INF
ejb-jar.xml	EJB	META-INF or WEB-INF
faces-config.xml	JSF	WEB -INF
persistence.xml	JPA	META-INF
validation.xml	Bean Validation	META-INF or WEB-INF
web.xml	Servlet	WEB-INF
web-fragment.xml	Servlet	WEB-INF
webservices.xml	SOAP Web Services	META-INF or WEB-INF

Uvod u (Java) web programiranje

- Web aplikacije na Java platformi u mnogome funkcionišu kao i web aplikacije na ostalim programskim platformama, ali se njihova funkcionalnost, sa druge strane, drastično razlikuje u odnosu na java desktop aplikacije. Zbog toga je, pre kreiranja java web aplikacija, potrebno dobro se upoznati sa konceptom web programiranja uopšte, kao i implementacijom web programiranja na java platformi. Za razliku od desktop aplikacija, web aplikacije razlikuju dve celine u toku svog izvršavanja. Jednu celinu čini serverska logika, a drugu klijentska. Zato ćemo u opisu web aplikacije, često čuti pojmove **server** i **klijent**.



Web server

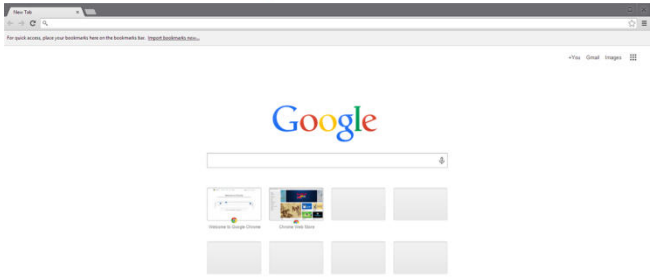
- Server je program koji preuzima i obrađuje klijentske zahteve, a zatim, na osnovu parametara iz tih zahteva, kreira odgovarajuće odgovore. Svaka Internet aplikacija, tačnije, svaki sajt koji otvorimo, funkcioniše po ovom principu, ali se tehnologije, koje izvršavaju ove procese, (preuzimanje zahteva i slanje odgovora) razlikuju.
- Ova razlika nije konceptualna, jer svaka tehnologija poštuje isti princip preuzimanja i vraćanja odgovora, ali, obzirom da se same pozadinske tehnologije razlikuju, neophodno je i poznavanje svake od njih da bi se njome moglo adekvatno rukovati.
- Postoji nekoliko dominantnih web server aplikacija.
  - **Microsoft Internet Information Services** (IIS).
  - **Apache** web server
  - **Tomcat**, **JBoss**, **Geronimo, GlassFish**...

Izvršavanje web aplikacija

- Često se web serveri nazivaju i application serveri, zato što su u stanju da startuju i određene aplikacije prilikom obrade klijentskog zahteva.
- Onog trenutka kada određena web prezentacija (sajt) u svom postojanju upotrebi neku od aplikacija (funkcionalnosti) na serveru, ona, zapravo, postaje web aplikacija. Programski kod (program) koji se izvršava na serveru prilikom aktivacije klijentskog zahteva, naziva se **serverski kod**.
- Razlikuje se više načina izvršavanja serverskog koda, odnosno, više mogućnosti njegove implementacie. Kada web server primi klijentski zahtev, čita njegove karakteristike (sadržane u zaglavlju zahteva) i na osnovu njih izvršava određenu akciju. Ukoliko zahtev ne sadrži zahtev za aktivaciju web aplikacije, već samo potražuje neki dokument sa servera, server će pronaći dokument na fajl sistemu, a zatim ga proslediti klijentu kroz odgovarajući odgovor. Ovakav scenario podrazumeva preuzimanje statičkih sadržaja sa web servera (html dokumenata, slika i sl.).
- Ukoliko zahtev zahteva angažovanje serverskog koda, procedura je drugačija. Server, takođe, pronalazi dokument, ali ga prosleđuje **java virtualnoj mašini**, koja ga izvršava i prosleđuje klijentu. Svi elementi web aplikacije, koji se u javi izvršavaju na ovaj način, posredstvom serverskog koda, nazivaju se **java web components**.

Web klijent

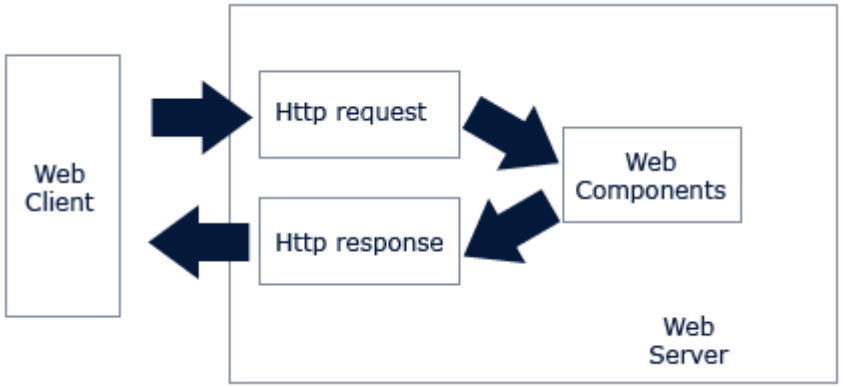
- Svaka web aplikacija ima dve strane u toku svog izvršavanja. Jedna strana je serverska, a druga klijentska. Klijentska strana je krajnja tačka izvršavanja web aplikacije i ona, obično, takođe, sadrži logiku koja je u stanju da pošalje zahtev serveru, kao i da preuzme i adekvatno pročita odgovor. Web pretraživač (mozilla, internet explorer, opera, safari...) je najčešći oblik klijentske web aplikacije. Ali konzument web aplikacije ne mora biti obavezno web pretraživač, već može biti bilo koja aplikacija, sve dok u svom izvršavanju podrazumeva i komunikaciju sa web serverom.



Problemi u web programiranju

- Koncept klijent/server je izuzetno problematičan sa stanovišta
  - Bezbednosti
  - Brzine transporta informacija

- Postoje razne tehnike kojima se ovi problemi zaobilaze, ali je i dalje, baš zbog njih, kreiranje web aplikacija jedan od najvećih programerskih izazova danas

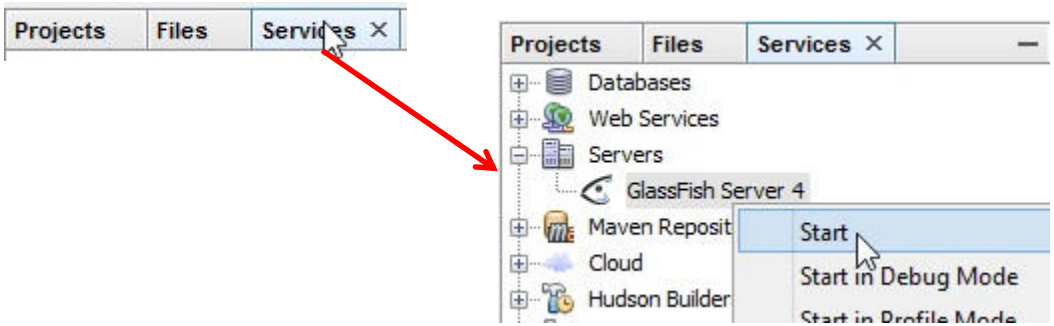


Web programiranje u Javi

- Tehnologije za kreiranje web aplikacija u Javi nazivaju web components
- Web components su podrazumevani element Java EE edicije i podrazumevaju dve ključne stavke **Java Server Pages** (JSP) i **Java Servlets**.
- Java Server Pages je tehnologija koja podrazumeva implementaciju java koda u markup kod (html, xml i slično), omogućavajući da se inače nefunkcionalan HTML kod obogati nekom funkcionalnom logikom (kroz Javu).
- Servleti funkcionišu kao Java komponente iz kojih je neophodno eksplicitno emitovati HTML (ili neki drugi) kod, zbog čega se najčešće koriste kao pozadinska logika unutar web aplikacije, kao i sistemi za dobavljanje podataka u procesu zahteva i odgovora.Web komponenta egzistira u okruženju koje se naziva **web kontejner**
- Da bi bila izvršena u web kontejneru (na app serveru) Web komponenta (java web aplikacija) mora biti instalirana/pripremljena za server. Odnosno, mora biti izvršen njen **deploy**. Prilikom procesa pripreme (deploy) aplikacije, poziva se alat deploytool, koji gradi aplikaciju i generiše **war** (web application archive) fajl, odnosno, arhivu u kojoj se nalazi web aplikacija.
- War u sebi sadrži strukturu web aplikacije, uključujući i sve njene resurse (slike, html dokumente i slično).

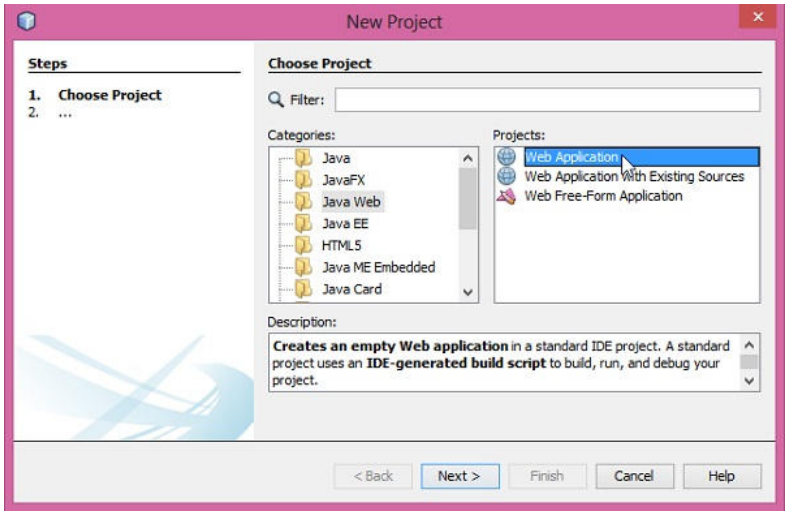
Kreiranje prve web aplikacije

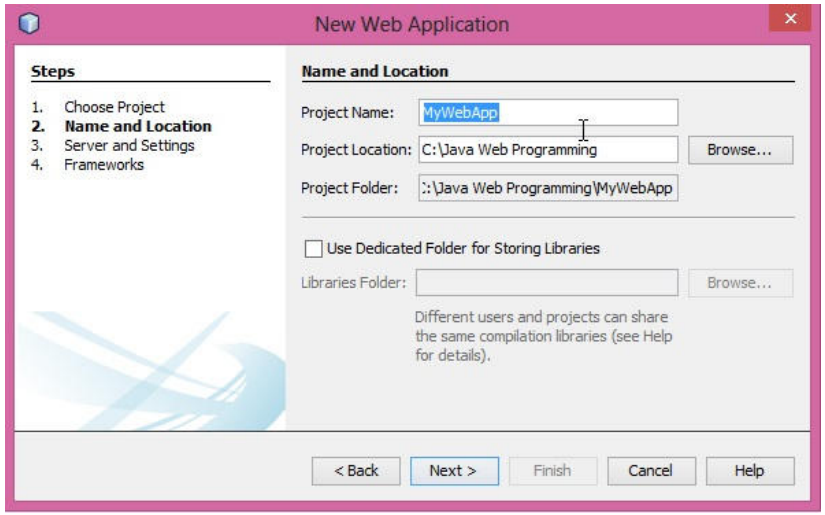
- GlassFish aplikativni server je sastavni deo Java EE developers kit-a
- Ukoliko je NetBeans instaliran sa podrškom za Java web, zajedno sa njim je automatski instaliran i glassfish server
- Glassfish server se može pokrenuti putem NetBeans okruženja



Kreiranje web projekta

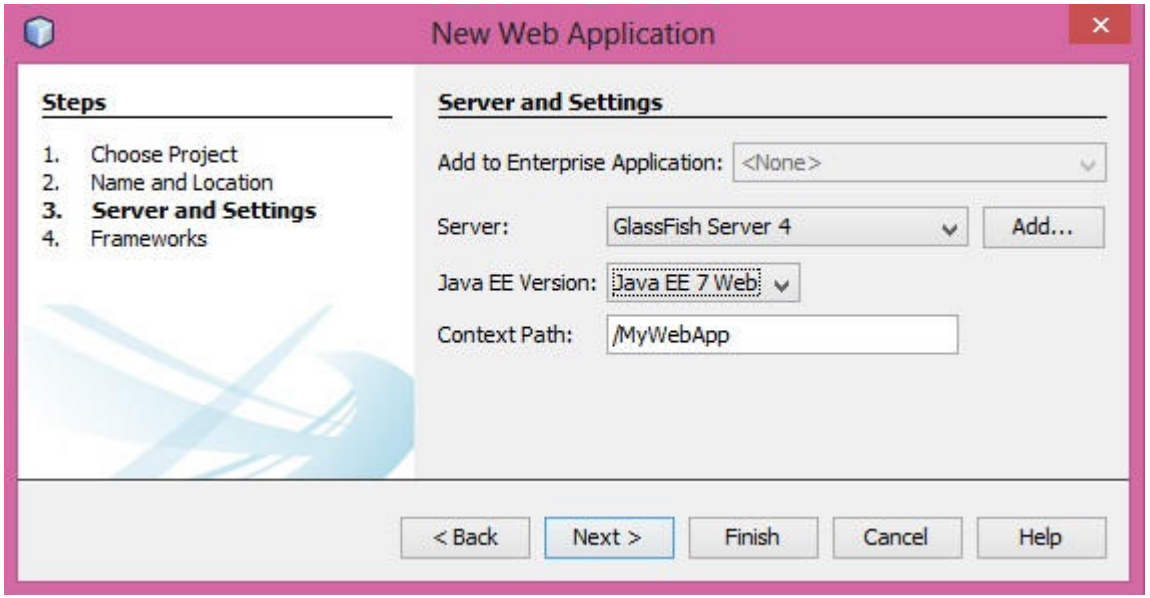
- Da bi se kreirala jedna Java veb-aplikacija, potrebno je uraditi sledeće:
- **1.** *File –> New Project...*
  2. za kategoriju odabrati Java Web, a za projekat Web Application





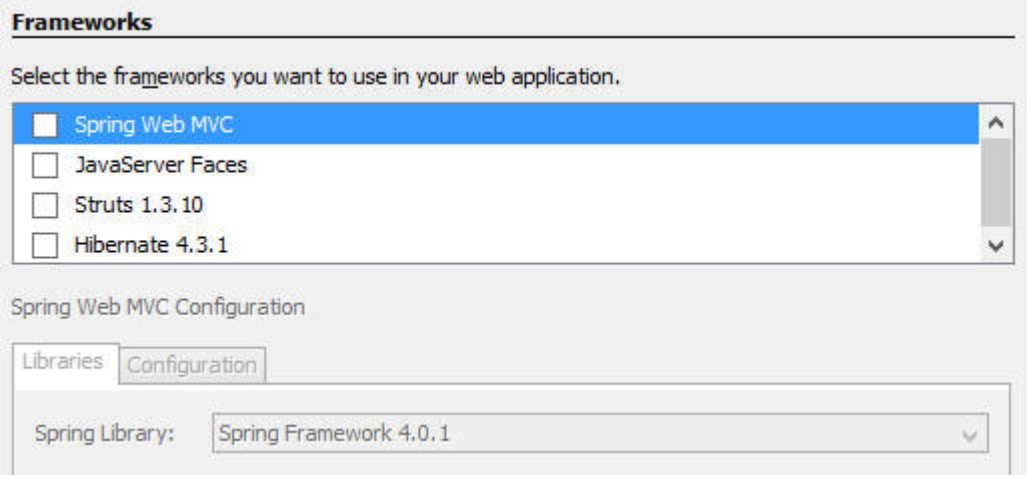
Kreiranje web projekta

- U dijalogu Server and Settings moguće je postaviti podešavanja koja se tiču servera; da server nismo pokrenuli pre kreiranja projekta, ovde bismo imali opciju da to uradimo: Context Path, ukazuje na deo putanje pod kojom će naša aplikacija biti dostupna, na primer iz nekog browsera;

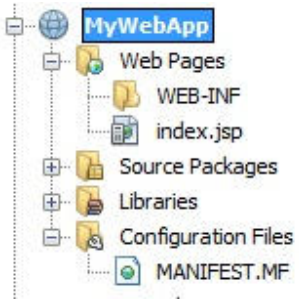


Kreiranje web projekta

- U dijalogu Frameworks moguće je odabrati neki od frameworka, koji se mogu koristiti u projektu



Pritiskom na taster FINISH, projekat biva uspostavljen.



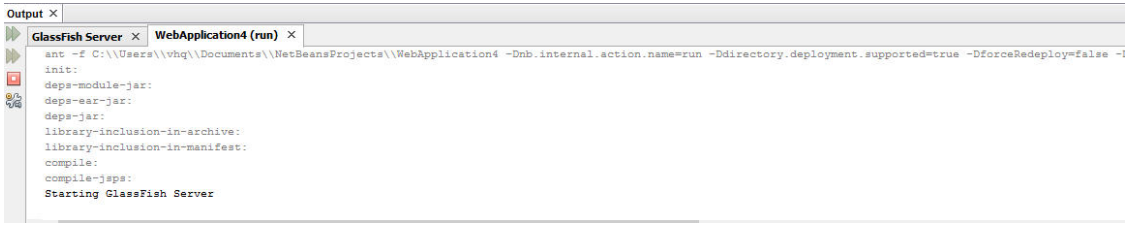
Web (pages) direktorijum

- Ovaj direktorijum sadrži javni sadržaj web aplikacije
- Sve ono što se nalazi u ovom direktorijumu biće izloženo, osim direktorijuma WEB-INF. U ovom direktorijumu se najčešće nalaze fajlovi koji ne treba da budu dostupni javnosti
- Prvi stanovnik ovog direktorijuma često je fajl **web.xml** – takozvani **web deployment descriptor**
- U ovom fajlu se nalaze informacije koje objašnjavaju serveru na koji način će tretirati određene delove aplikacije

Startovanje aplikacije

- Startovanje web aplikacije iz NetBeans-a se vrši kao i startovanje bilo koje druge aplikacije – aktivacijom tastera F6 ili zelene strelice

- Ipak, koraci koje u trenutku startovanja NetBeans (ant) mora da izvrši kako bi startovanje bilo uspešno, razlikuju se od startovanja obične java aplikacije, jer se pre samog startovanja mora aktivirati web / aplikativni server



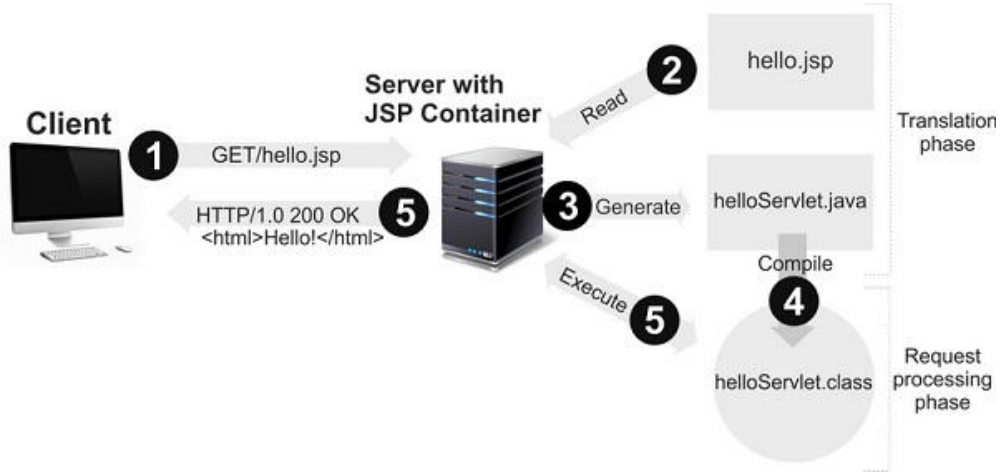
Ubacivanje korisnički definisanog sadržaja na stranu

- Osnova svrha korišćenja jave za veb-programiranje jeste mogućnost izvršenja različite logike na serveru.
- Ovo se postiže ili ugradnjom Java koda u html ili kreiranjem zasebnih fajlova sa isključivo Java kodom. U primeru nemamo nikakav Java kod integrisan u html. Radi boljeg razumevanja, kao demonstraciju ćemo promeniti liniju sa h1 tagom. Umesto postojeće linije postavite sledeću:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1><% out.println("Hey, I am created dynamically!"); %></h1>
    </body>
</html>
```

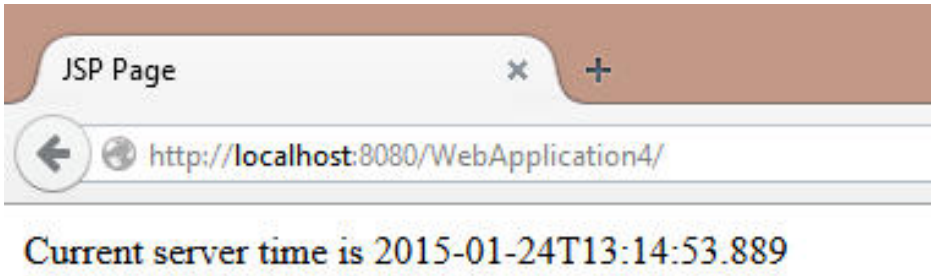
Ubacivanje korisnički definisanog sadržaja na stranu

- Kada zatražimo neku JSP stranicu putem pretraživača, aplikacioni server proveri da li ta strana postoji u prevedenom obliku. Ukoliko ne postoji, izvršava se prevođenje strane i emitovanje korisniku; u suprotnom, odmah se preuzima prevedena verzija.
- U trenutku prevođenja JSP strane, prevodilac kompletan HTML, XML ili neki drugi ne-Java kod prosleđuje u njegovom izvornom obliku, da bi, u trenutku kada naiđe na tag sa oznakom %, izvršio prevođenje sadržaja taga, a zatim nastavio sa prosleđivanjem ne-Java koda. Nakon prevođenja, JSP strana dobija formu servleta i više se ne prevodi sve do promene sadržaja dokumenta



Vežba (jwex012014 ShowTime)

- Potrebno je napraviti jednostavnu veb-aplikaciju koja prikazuje trenutno vreme na stranici.

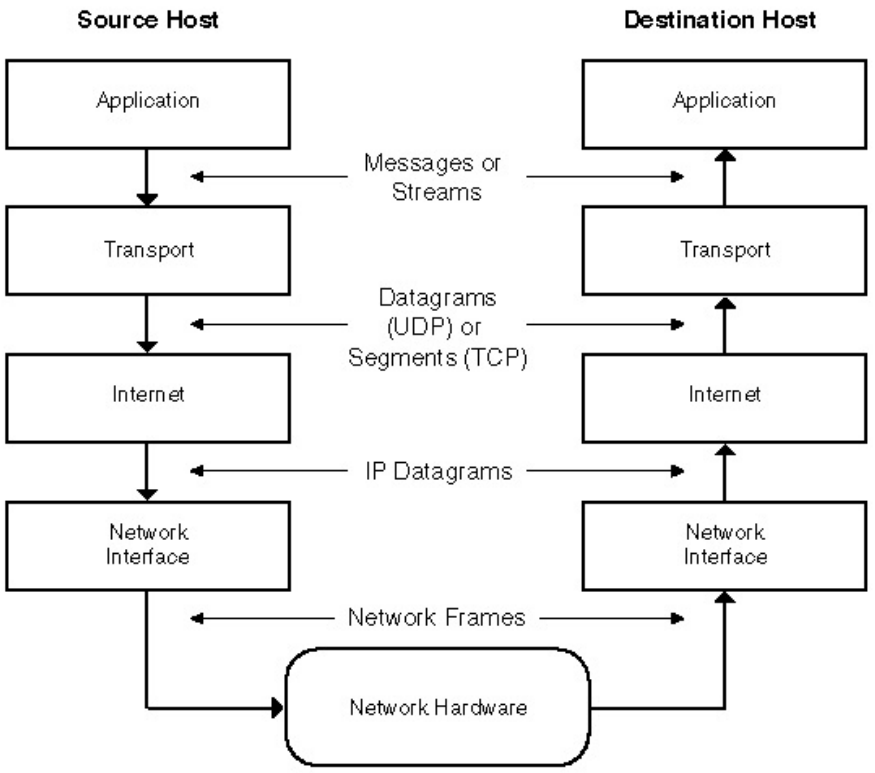


Uvod u web programiranje

- Osnovni internet protokoli predstavljaju temelj njegovog funkcionisanja. Zbog toga, poznavanje ovih protokola predstavlja osnovu za razumevanje funkcionisanja veb-aplikacija i za njihovu uspešnu izgradnju.
- Sve mašine na internetu mogu biti klasifikovane na dva tipa: servere i klijente. Klijenti su mašine koje zahtevaju određene informacije, a server je taj koji te informacije stavlja na raspolaganje klijentima. Informacije putuju od provajdera informacija (servera) do primaoca informacija (klijenta) i pritom je ta komunikacija uslovljena određenim skupom pravila. Ova pravila se nazivaju protokol.
- Veb-pretraživač (browser), veb-server i veb-aplikacija za međusobnu komunikaciju koriste Hypertext Transfer Protocol (HTTP). Klijenti šalju HTTP **zahtev** upućen veb-serveru, a server uzvraća podatke u formi HTTP **odgovora**

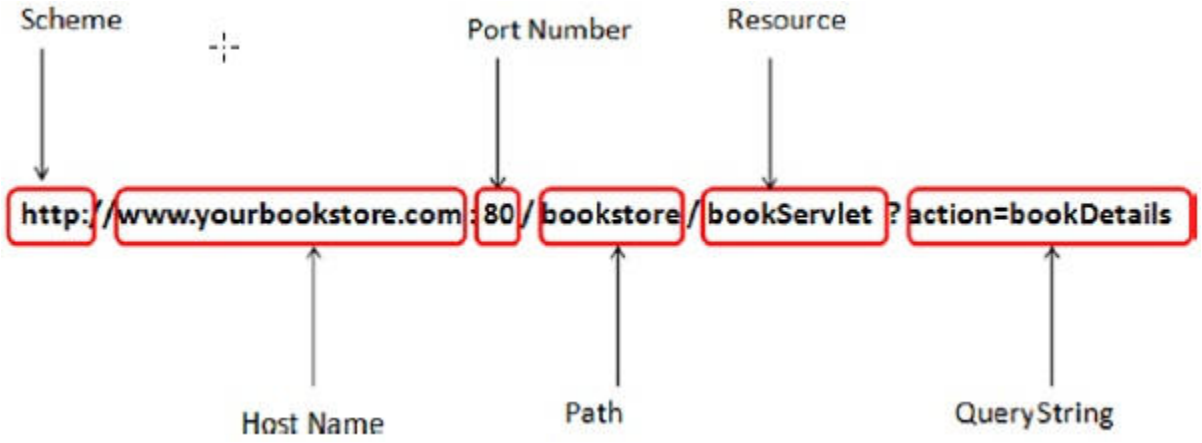
TCP/IP

- Da bi jedna veb-strana bila transportovana od tačke A (server) do tačke B (klijent), mnoštvo uslova mora biti ispunjeno i mnoštvo aktera učestvuje u tom procesu. Sam proces podrazumeva četiri različita nivoa
- Prvi, najniži nivo je **Network Access Layer**. Na ovom nivou obezbeđuje se validnost komunikacije između uređaja, ispravnost dostave sirovih podataka i slično. Ovo je sloj koji najčešće nije dostupan (niti potreban) programeru veb-aplikacije.
- **Internet Layer** je sloj na kome se razmenjuju paketi između računara. Ovaj sloj je velika saobraćajnica IP (Internet Protocol) paketa, na kojoj se nalazi mnoštvo „stanica" identifikovanih IP adresama. Svaki IP paket u sebi nosi podatak i IP adresu, na osnovu koje sistem zna gde taj podatak treba dostaviti.
- Kroz **Transport Layer** se razmenjuju podaci na osnovu jednog od dva sistema (protokola)
  - **TCP (Transport Control Protocol)**
  - **UDP (User Datagram Protocol)**
- **Application Layer** je sloj kroz koji putuju podaci u raznim preformatiranim oblicima. „Formati" podataka koji se koriste prilikom ovih putovanja zapravo su protokoli aplikacionog sloja (FTP, HTTP, SMTP...). Ti protokoli su veoma bitni za veb-programiranje, a svakako najbitniji među njima je **HTTP (Hyper Text Transport Protocol)**



Osnove HTTP-a

- HTTP je request-response protokol, što znači da njegov način funkcionisanja počiva na međusobnom smenjivanju zahteva i odgovora između klijenta i servera. Kada klijent zatraži neki resurs, taj zahtev obično sadrži identifikaciju resursa koji je zahtevan, i to u formi Uniform Resource Locator-a (URL). URL je hijerarhijska sekvenca komponenti, struktuirana kao na slici



HTTP zahtevi i odgovori (zahtev)

- HTTP komunikacija između klijenta i servera započinje HTTP zahtevom klijenta
- Zahtev sadrži u sebi i HTTP metod, odnosno obrazac koji veb-server treba da ispoštuje prilikom obrade zahteva i rukovanja resursima. Metod bi najbolje mogao da se okarakteriše kao najobičnija naredba. Ako bismo npr. otkucali u pretraživaču www.mysite.com/index.html, mogli bismo biti sigurni da će negde u našem zahtevu postojati linija:

  GET /index.html HTTP/1.1

  Kao i linija:

  host: www.mysite.com
- gde je:

  GET naziv metoda,
  /index.html adresa koja se zahteva,
  HTTP/1.1 verzija protokola i
  host: www.mysite.com naziv top domena za taj sajt

HTTP zahtevi i odgovori (odgovor)

- Na osnovu ovih informacija web server formira odgovor i prosleđuje ga klijentu.
- Na zahtev formatiran na pomenuti način, server formira odgovor i prosleđuje ga nazad pošiljaocu (a server u svakom trenutku zna IP adresu i port, gde se od njega očekuje da prosledi odgovor).
- Odgovor servera sadrži status – **200** ako je sve u redu i mnoge druge brojeve grešaka, ukoliko nešto nije kako treba. Brojevi sa kojima se najčešće susreće su **404** (tražena strana ne postoji) i **500** (greška u veb-aplikaciji). Zapravo, statusi se po brojevima dele na određene grupacije: **1xx za informacije, 2xx sve vrste uspešnih apstrakcija, 3xx redirekcije, 4xx klijentske greške, 5xx serverske greške**.

HTTP Request / response

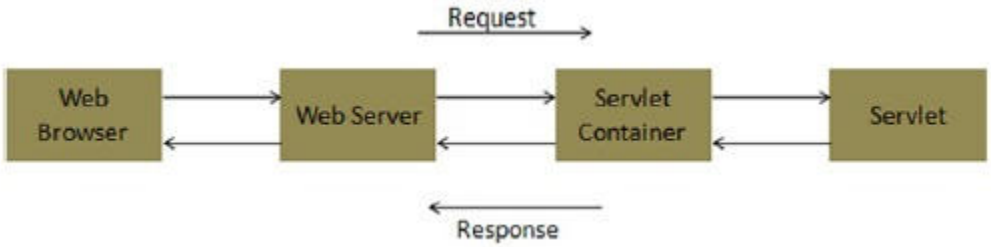
- Korisnik unosi adresu:
  - http://www.mysite.com/index.html
- User Agent otvara Socket preko porta 80 i šalje zahtev preko njega:
  - GET /path/file.html HTTP/1.0
    host: www.mojsajt.com
- Server, nakon što prihvati zahtev, na isti port, preko istog Socketa, šalje sledeći odgovor:

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 500

<html>
        <body>
                some server response...
        </body>
</html>
```

Java web kontejner

- Servlet je klasa čijem objektu se daje na raspolaganje svaki dobijeni zahtev



Java veb-aplikacija je sastavljena od više komponenti koje se izvršavaju unutar **veb-kontejnera**, koji se drugačije naziva **servlet container**. Kada veb-server dobije zahtev od strane klijenta za određenim resursom, on ovaj zahtev prosleđuje servlet kontejneru u kojem se tražena komponenta nalazi. U kontejneru zahtev dobijaju **servlet** ili **jsp** strana.

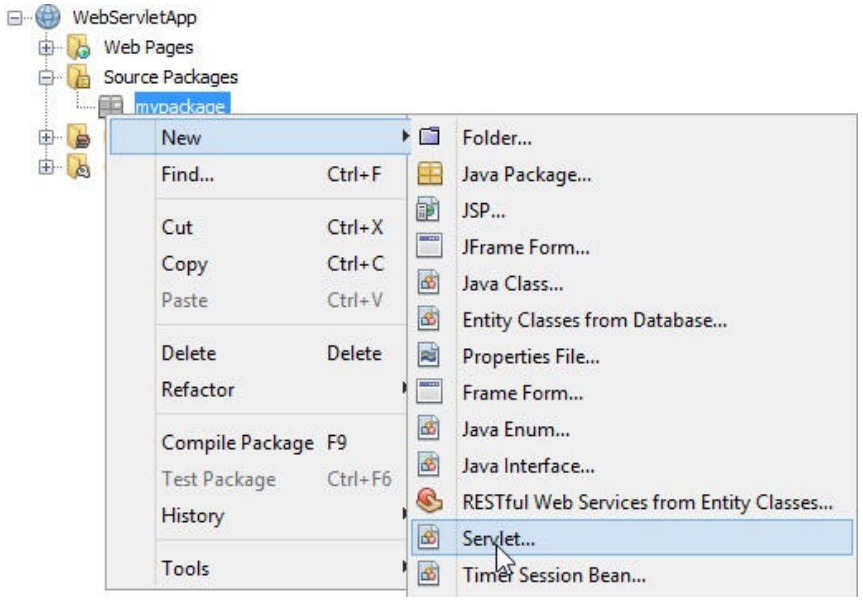
Servlet

- Servleti predstavljaju centralnu jedinicu za procesiranje u Java veb-aplikacijama, i odgovorni su za izvršenje većine logike koju sama aplikacija zahteva.
- Servlet je, zapravo, obična java klasa koja implementira interfejs **javax.servlet.Servlet**.
- Ovaj interfejs definiše metode koje svi servleti moraju da implementiraju. Ovo su metode koje definišu osnovne, odnosno ključne tačke životnog ciklusa jednoj servleta. To su metode za inicijalizovanje servleta, servisiranje zahteva i brisanje servleta sa servera

povratna vrednost i tip	**metod**
void	init(ServletConfig config)
void	service(ServletRequest req, ServletResponse res)
void	destroy()
ServletConfig	getServletConfig()
String	getServletInfo()

Kreiranje servleta u NetBeans-u

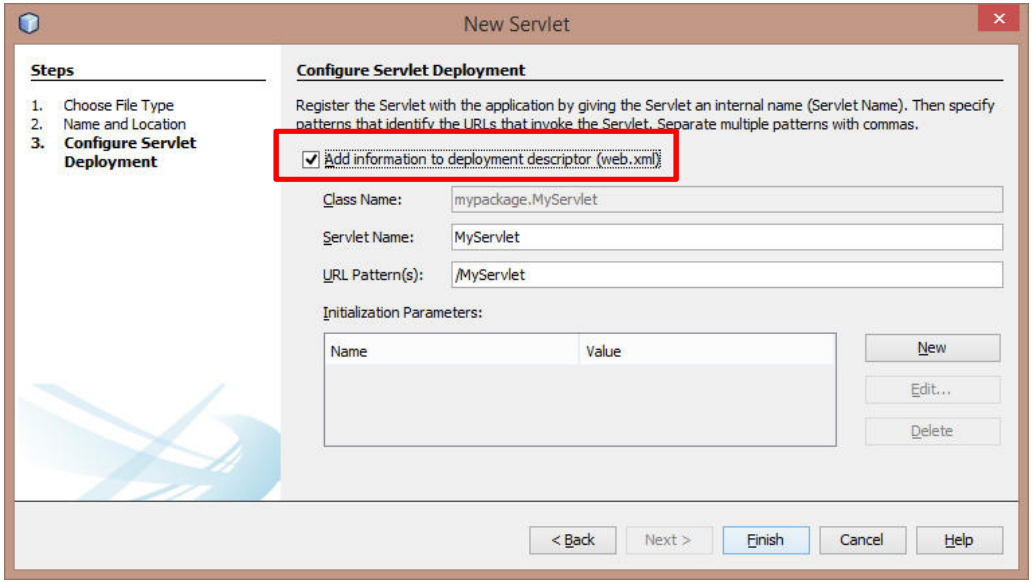
- Nakon kreiranja web projekta u NetBeans-u, i paketa unutar njega, servlet se dodaje kao i bilo koja druga klasa:



Osnovno konfigurisanje servleta

- Da bi servlet bio dostupan, treba da pored klase, sadrži i putanju koja će ga inicirati (url pattern)
- Ova informacija se smešta u deployment descriptor

  - Prilikom kreiranja servleta, netbeans omogućava automatsku registraciju servleta u descriptoru



Struktura servlet klase

- U kreiranom servletu mogu se identifikovati četiri generisana metoda:
- **processReques**
  - **doGet**
  - **doPost**
- **getServletInfo**

```
public class MyServlet extends HttpServlet {

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...15 lines }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
de.">
    /** Handles the HTTP <code>GET</code> method ...8 lines */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...3 lines }

    /** Handles the HTTP <code>POST</code> method ...8 lines */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...3 lines }

    /** Returns a short description of the servlet ...5 lines */
    @Override
    public String getServletInfo() {...3 lines }// </editor-fold>

}
```

Pokretanje servleta

- Kreirani servlet će funkcionisati čim startujemo aplikaciju (na standardan način)
- Ako hoćemo da servlet bude automatski otvoren prilikom startovanja aplikacije (indeks strana), tada moramo da ga stavimo u welcome file listu u web.xml fajlu:

```
s1:schemaLocation="http://xmlns.jcp.org/xml/
 <welcome-file-list>
     <welcome-file>MyServlet</welcome-file>
 </welcome-file-list>
```

Tok zahteva i proces pronalaska servleta

- Prilikom navođenja gore navedene adrese na kojoj se nalazi naš servlet, browser formira HTTP zahtev, koji izgleda ovako:

GET / HTTP/1.1
Host: localhost:8080
...

- Nakon dobijanja zahteva, web server utvrđuje njegov sadržaj i pokušava da formira odgovor. Ukoliko se u zahtevu traži statički dokument, odmah ga šalje nazad. U suprotnom, angažuje program za obradu zahteva (servlet ili jsp stranu)

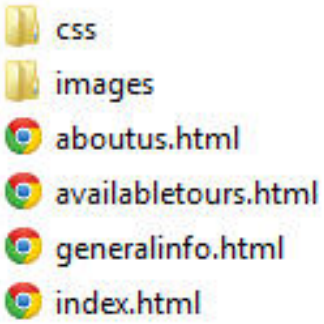
Podešavanje servleta korišćenjem anotacija

- Od verzije Servlet 3.0 moguća je njihova konfiguracija korišćenjem anotacija i korišćenjem web.xml fajla (ili oba). U narednoj tabeli prikazane su anotacije podržane od strane Servlet 3.0 kompatibilnog veb-kontejnera.

Anotacija	**Opis**
@WebFilter	definiše filter u veb-aplikaciji
@WebInitParam	definiše inicijalne parametre koji se prosleđuju servlet
@WebListener	prijem događaja
@WebServlet	definiše komponentu veb-aplikacije
@MultipartConfig	ukazuje da je zahtev tipa mime/multipart

Vežba
(jwex012014 BelgradeServlet)

- Date su četiri html strane (potreban direktorijum belgrade iz repozitorijuma)
- Potrebno je ove četiri html strane preformatirati tako da se realizuju dinamički, pomoću servleta



Parametrizacija servleta

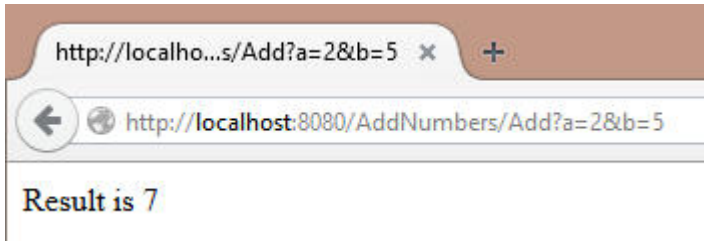
- Kod web aplikacija, parametrizacija ima veoma važnu ulogu jer se njome utiče na tok programa koji procesira zahtev
- U http-u postoje dva načina za transport parametara. I oni se koriste u zavisnosti od metode koja se koristi za prosleđivanje zahteva.
- Ukoliko se koristi GET metod, za definisanje parametara može se koristiti query string. Tada se parametar ili parametri pridružuju samom URL-u.
- Ukoliko se koristi metod POST, parametri se nalaze u telu zahteva

Preuzimanje parametara servleta

- Ako se u zahtevu koristi get metod, tada se parametri mogu preuzeti po nazivu, pomoću metode **getParameter**, objekta **request**
- Na primer, za zahtev: **/MyServlet?a=Hello**
  Vrednost promenljive poslate na server će biti dostupna unutar sevleta kroz sledeći kod:
  request.getParameter("a")
- Kada se koristi post metod za slanje zahteva, tada se parametri prosleđuju kroz telo zahteva, ali je procedura za njihovo preuzimanje ista kao i kod get metode

Vežba
(jwex012014 AddNumbers)

- Potrebno je kreirati servlet za sabiranje brojeva
- Servlet treba da prihvata dva parametra sa nazivima a i b, koji će predstavljati sabirke
- Servlet na osnovu ulaznih parametara treba da izvrši sabiranje i da prikaže rezultat na izlazu



Java Server Pages

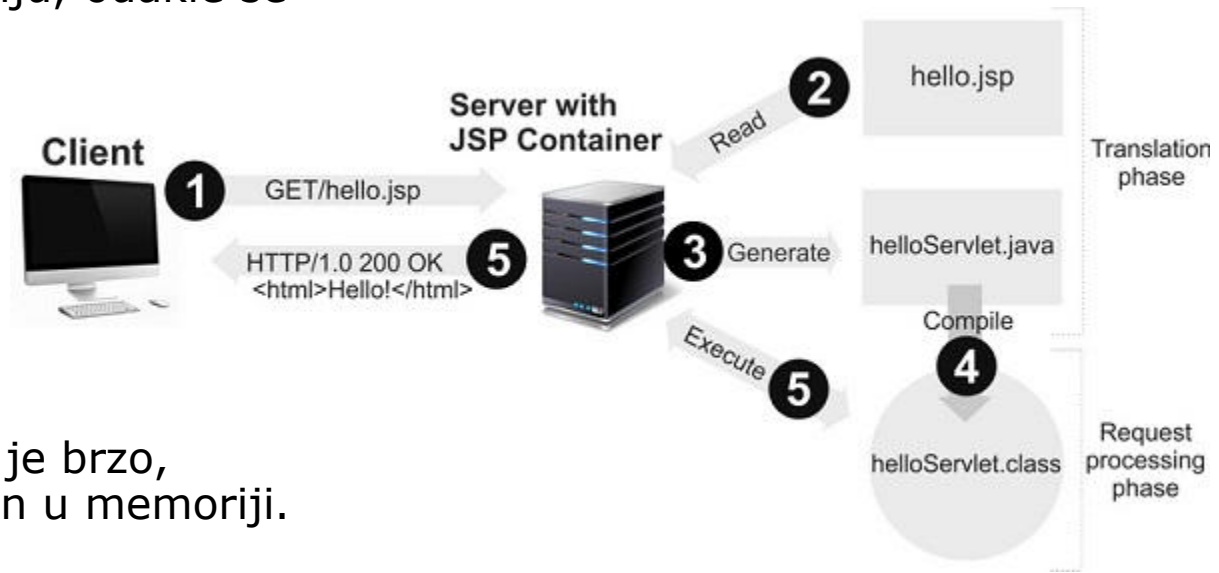
- U radu sa Java servletima, vrši se integraciju HTML koda u Java kod. Ovo se može smatrati nedostatkom servleta, s obzirom da dolazi do mešanja funkcionalnosti više slojeva aplikacije.

- Da bi se prevazišli ovi nedostaci, kreirana je tehnologija Java Server Pages (JSP). JSP koristi kombinaciju statičkog HTML-a i dinamičkog sadržaja i na taj način omogućava uspešno fizičko razdvajanje slojeva aplikacije, i kreiranje dinamičkih aplikacija, kod kojih se funkcionalnosti servleta pozivaju iz HTML-a

- Slične implementacije postoje i u drugim programskim jezicima, na drugim veb-serverima. Najpoznatije su ASP (Active Server Pages) skripte, koje funkcionišu na Microsoft veb-serveru (IIS) i u pozadini rukuju sa različitim programskim jezicima (Visual Basic, C# ...), i PHP (PHP Hypertext Processor) skripte, koje funkcionišu izvorno na Linux platformi, ali su portabilne, pa se mogu sresti i na drugim platformama (Windows).

Životni ciklus JSP dokumenta

- JSP dokumenti su dokumenti na veb-serveru, koji imaju ekstenziju .jsp. Kada veb-server prihvati zahtev za ovakvom stranom, on je prosleđuje direktno Java prevodiocu. Java prevodilac, zatim, pretvara dokument u Java klasu, a zatim se ta klasa pretvara u Java Servlet. Servlet se, zatim, pretvara u Java Bytecode i smešta u memoriju, odakle se poziva prilikom svakog sledećeg zahteva. Ovakav koncept podrazumeva, nešto sporije, prvo otvaranje .jsp dokumenta, jer se strana prevodi i prolazi kroz ceo opisani proces. Sa druge strane, svako sledeće otvaranje
- ovakvog dokumenta veoma je brzo, jer se on već nalazi preveden u memoriji.



Dispatcher servlet

- Servlet može inicirati otvaranje jsp strane
- Ovakav servlet naziva se dispatcher servlet

```
RequestDispatcher rdd = request.getRequestDispatcher("index.jsp");
rdd.forward(request, response);
```

Java Server Pages

- U radu sa Java servletima, vrši se integraciju HTML koda u Java kod. Ovo se može smatrati nedostatkom servleta, s obzirom da dolazi do mešanja funkcionalnosti više slojeva aplikacije.
- Da bi se prevazišli ovi nedostaci, kreirana je tehnologija Java Server Pages (JSP). JSP koristi kombinaciju statičkog HTML-a i dinamičkog sadržaja i na taj način omogućava uspešno fizičko razdvajanje slojeva aplikacije, i kreiranje dinamičkih aplikacija, kod kojih se funkcionalnosti servleta pozivaju iz HTML-a.

JSP Elementi (Direktive)

- Direktive su u JSP-u posebni delovi dokumenta koji obično ne produciraju nikakav izlaz, ali daju određene informacije prevodiocu o strani. Na ovaj način moguće je definisati ponašanje same strane.
- Postoje tri vrste direktiva, a to su **Page**, **Include** i **Taglib** direktive.
- Sledeća direktiva vrši import java.util paketa u JSP stranu:

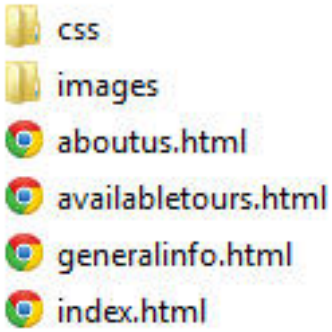
```
<%@page import="java.util.*"%>
```

- Sledeća direktiva implementira kompletan sadržaj fajla myFile.jsp u aktuelnu stranu:

```
<%@include file="myFile.jsp" %>
```

Vežba
(jwex012014 IncludePages)

- Date su html strane (belgrade_includepages iz repozitorijuma)
- Potrebno je modifikovati ove strane tako da budu jsp strane, i optimizovati jsp strane tako da se ne ponavlja sadržaj



Expressions tagovi

- Expressions tagovi su najjednostavniji način za implementiranje neke Java promenljive u kod, tj. za kreiranje nekog dinamičkog zapisa. Oni ne mogu sadržati blokove koda, već samo jednu liniju koja je zadužena za prikaz neke vrednosti. Za kreiranje Expressionsa koristi se sledeća sintaksa:

<%= expression %>

Skriplet tagovi

- Scriptlet tagovi su tagovi koji „presecaju” HTML kod u .jsp dokumentu i u njega implementiraju Java kod. Upotreba ovih tagova je najjednostavniji način za implementaciju Java koda u HTML dokument, jer je, naizgled, pregledan (jednostavno se smenjuju Java i HTML kodovi).
- Ovakav pristup nije dobar za kompleksnije .jsp dokumente jer podrazumeva česte izmene između HTML i JSP kodova, što dokument (kod) može veoma brzo učiniti nepreglednim, a samim tim i nepraktičnim za rad. Ovakav način pisanja koda često se, baš zbog pomenute osobine, naziva **špageti kod**.
- Scriptlet tag započinje oznakom <%, a završava oznakom %> i sve unutar ovih tagova biće tretirano kao Java kod
- Ukoliko hoćemo da deklarišemo neki pojam unutar scriptlet taga, njegov početak mora sadržati i znak uzvika <%!, ovde se može izvršiti deklarisanje %>

Scriptlet tag - upotreba

- Sledeći primer predstavlja upotrebu scriptlet tagova

```
<%!
    public String hello() {
        String msg = "Hello World";
    return msg;
}
%>
Message from <b>Scriptlet</b>: <% out.println(hello()); %><br/> Message
from <b>Expression</b>: <%=hello() %>
```

Komentari

- Komentarisanje koda može se u JSP-u izvršiti na dva načina.
- Jedan način je jednostavnom upotrebom Slash oznaka, kao i u bilo kom drugom Java kodu:

```
<% //this is my comment %>
```

- Drugi način je upotrebom JSP komentar tagova. Ovi tagovi izgledaju slično Scriptlet tagovima, ali imaju i dodatnu oznaku -- na početku i kraju:

```
<%-- This is my comment --%>
```

Integracija HTML-a u Scriptlet

- Do sada smo vršili integraciju scriptleta u HTML kod. Veoma često, zapravo gotovo uvek, implementiraćemo i HTML kod u Scriptlet, što je moguće uraditi na više načina: jedan je da emitujemo HTML kod iz Jave, a drugi, da presečemo Scriptlet, ubacimo željeni HTML kod i zatim nastavimo Scriptlet.
- Zamislimo da hoćemo da emitujemo HTML kod iz Jave. Za primer, recimo da postoje dve promenljive u Javi: jedna predstavlja naziv neke strane, a druga njenu adresu.
- Ako bismo želeli da ova dva podatka smestimo u HTML hiperlink, mogli bismo napisati sledeći Scriptlet:

```
<%
    String title = "Link e-learning";
    String address = "http://www.link-elearning.com";
    out.print("<a href='"+address+"'>"+title+"</a>");
%>
```

Presecanje Scriptleta

- Prekid scriptlet taga na strani ne znači i gubitak njegovog konteksta
- Sve što se u scriptletu definiše na bilo kom delu strane, važi i unutar drugih scriptlet-a na istoj strani ili strana koje ona includ-uje

```
<%
    String x = "I am still x";
%>
Some HTML is happening here!!!<br />
<%
    out.println(x);
%>
```

Vežba
(jwex012014 MultiplicationTable)

- Potrebno je kreirati stranu "tablica množenja" koja će imati sledeći izgled:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100