Интернет програмирање предавање 11

Проф. др Мирослав Лутовац

mlutovac@viser.edu.rs

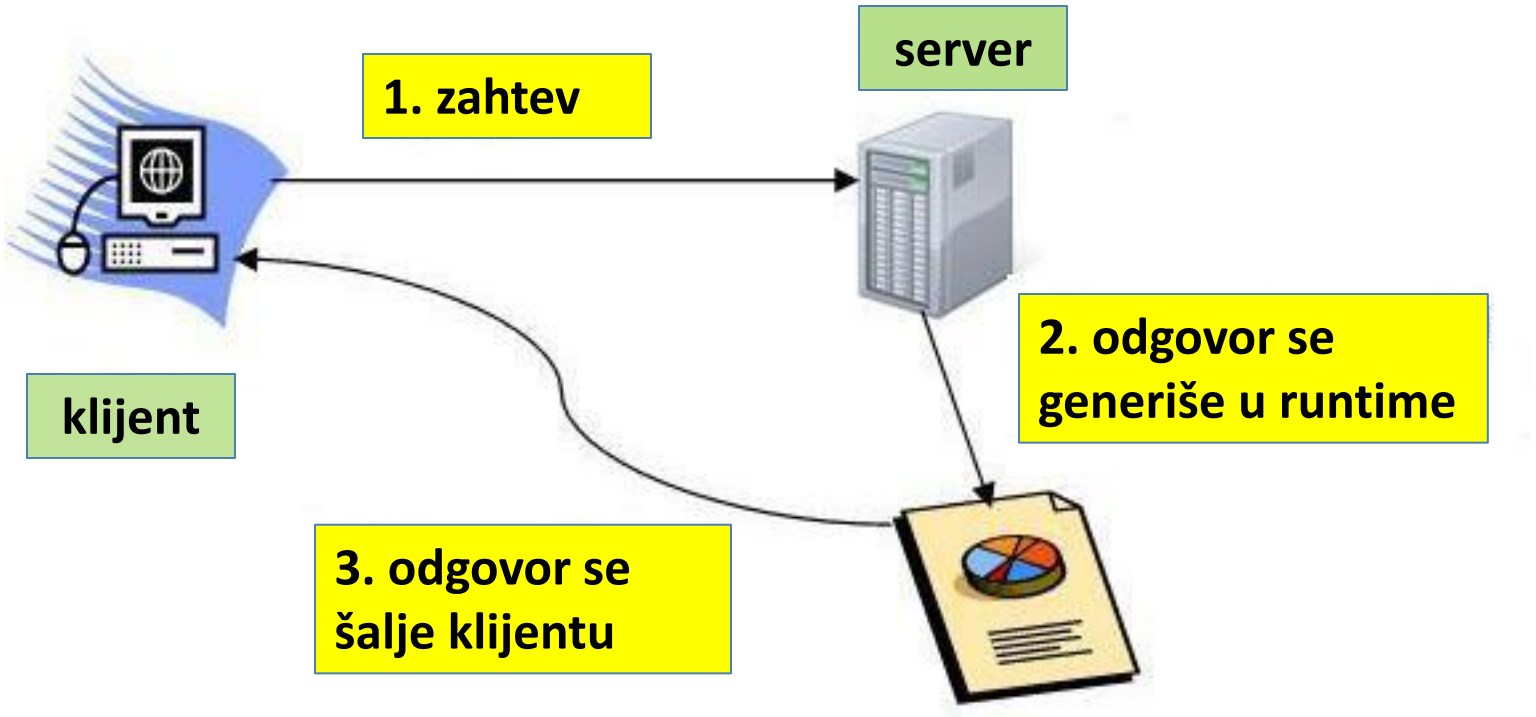
Servlet

- Servlet tehnologija se koristi **za kreiranje veb aplikacija**
- Postoji **na serveru** i generiše **dinamičnu veb** stranicu
- Servlet tehnologija je **robusna** i **skalabilna** (Java jezik)
- Postoji mnogo interfejsa i klasa u API servletima (Application Programming Interface) kao što su Servlet, GenericServlet, HttpServlet, ServletRekuest, ServletResponse

Servlet, u zavisnosti od konteksta

- Servlet je tehnologija za kreiranje veb aplikacija
- Servlet je API koji obezbeđuje brojne interfejse i klase uključujući dokumentaciju
- Servlet je interfejs koji mora da se implementira za pravljenje servleta
- Servlet je klasa koja proširuje mogućnosti servera i odgovara na dolazni zahtev;
  može da odgovori na bilo koji tip zahteva
- Servlet je veb komponenta koja je razvijena na serveru za kreiranje dinamičkih veb strana

Servlet, dinamičkih rad



Veb terminologija

Web Application (Webapp)

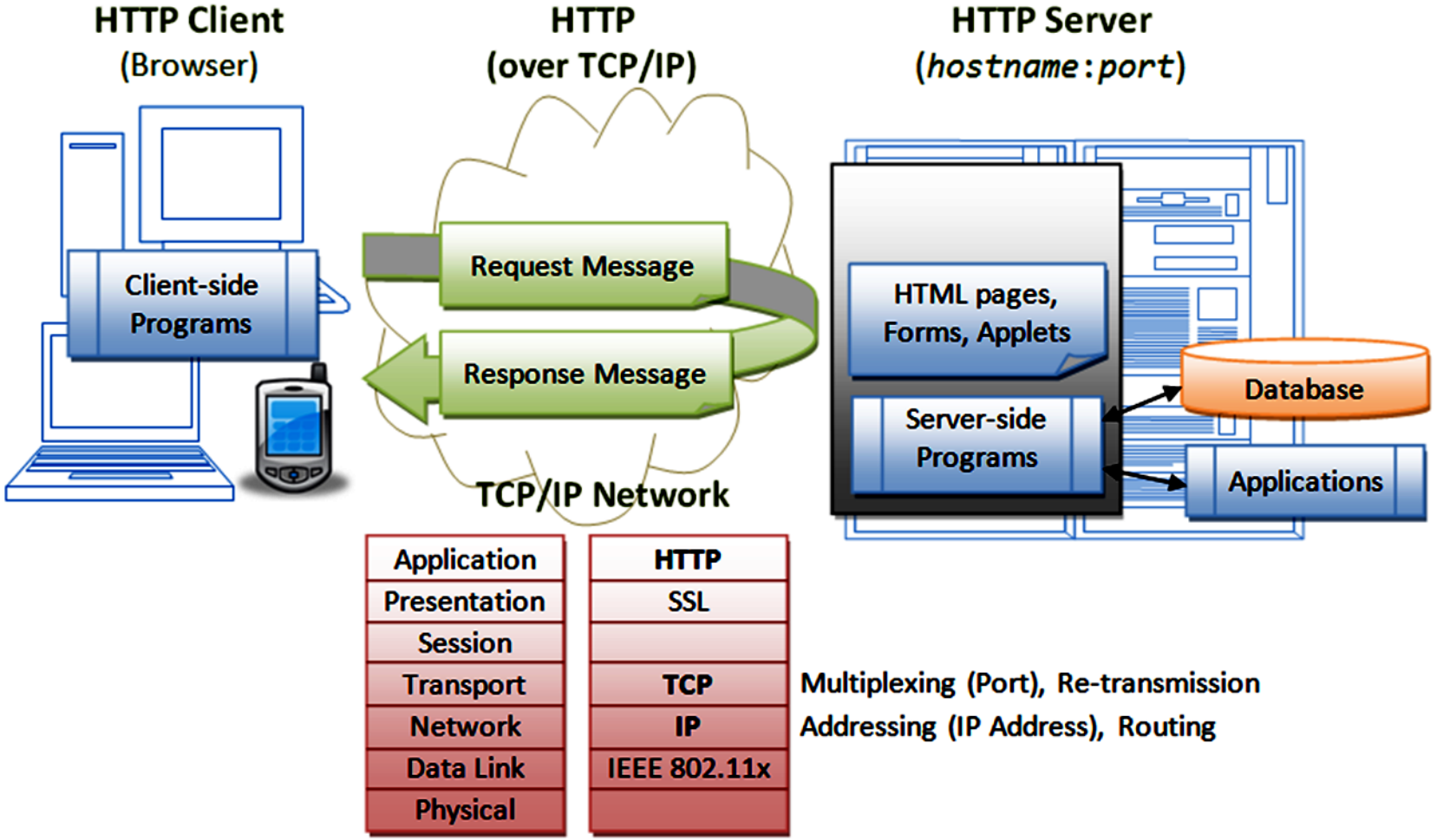
Za razliku od samostalne aplikacije, veb aplikacija se pokreće preko Interneta (google, amazon, ebay, facebook, twitter) .
Vebap je tipično višeslojna aplikacija baze podataka klijent-server koji se pokreće preko Interneta; sastoji se od 5 komponenti:

1. HTTP Server: Apache HTTP Server, Apache Tomcat Server, Microsoft Internet Information Server, Google Web Server…
2. HTTP Client (veb pregledač): Chrome, FireFox, Internet Explorer…
3. Database (baza podataka): MySQL, Apache Derby, mSQL, SQLite…
4. Client-Side programi: napisani u HTML, JavaScript, VBScript, Flash…
5. Server-Side programi: napisani u Java Servlet/JSP, ASP, PHP, Python…

Veb terminologija

Veb aplikacija (Webapp)

1. HTTP Server, 2. HTTP Client (veb pregledač), 3. Database (baza podataka), 4. Client-Side programi, 5, Server-Side programi

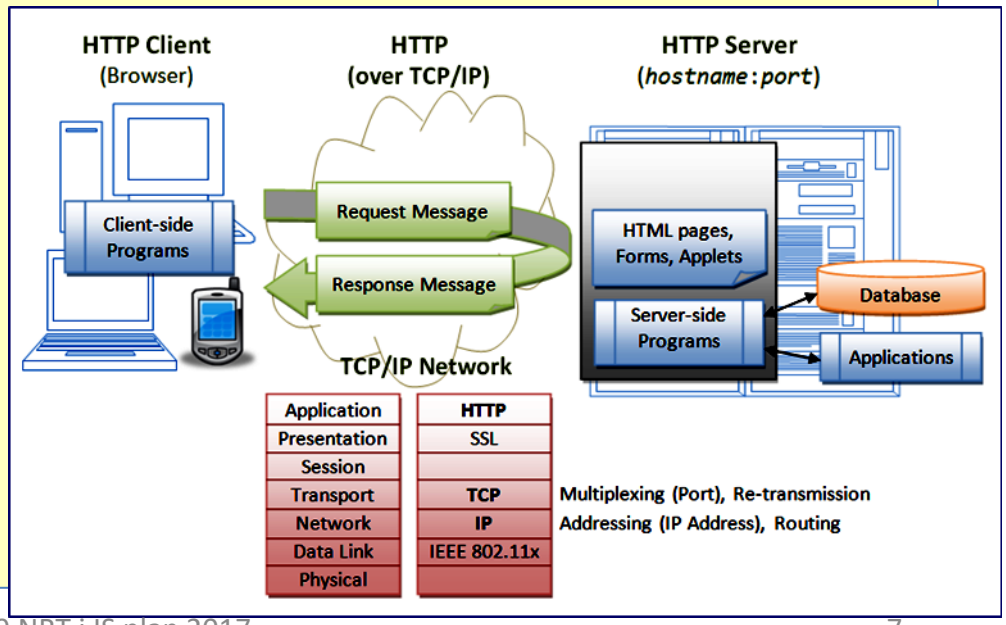


Veb aplikacija (Vebap)

Veb terminologija

Tipičan primer upotrebe:

1. Korisnik, preko veb pregledača (HTTP klijent), izdaje zahtev URL-a na HTTP server za pokretanje vebap
2. HTTP server vraća HTML formu (program na strani klijenta), koji se učitava u pregledaču klijenta
3. Korisnik popunjava upitnik u dobijenom obrascu i šalje ga
4. Klijentski program šalje parametre upita ka serverskom programu
5. Serverski program prima parametre upita, upit sa parametrima prosleđuje bazi podataka i vraća rezultat upita klijentskom programu
6. Klijentski program prikazuje rezultat upita na pregledaču
7. Proces se ponavlja za sledeći zahtev

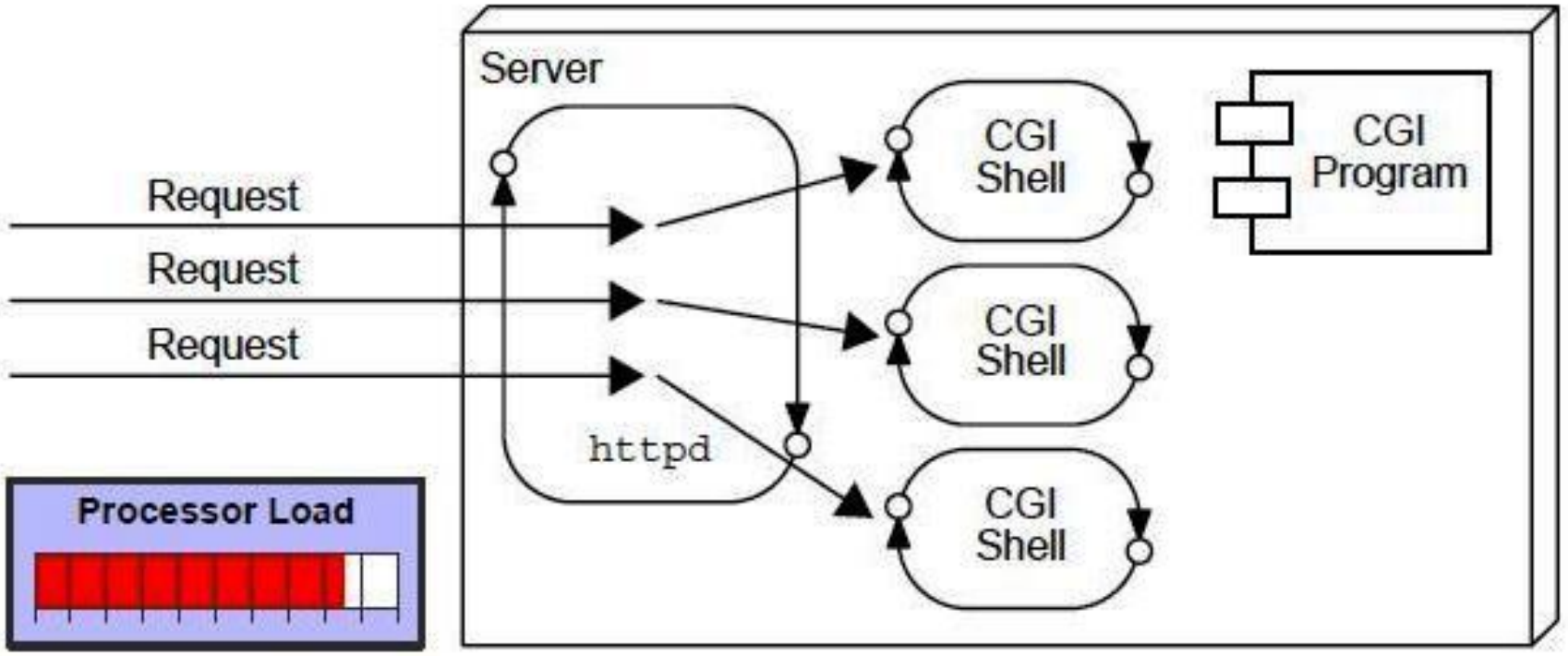


Veb aplikacija

- Veb aplikacija je aplikacija kojoj **može da se pristupi preko veba**
- Veb aplikacija se sastoji od **veb komponenti** kao što su Servlet, JSP, Filter, i HTML
- Veb komponente se tipično **izvršavaju na veb serveru** i odgovaraju na HTTP zahtev

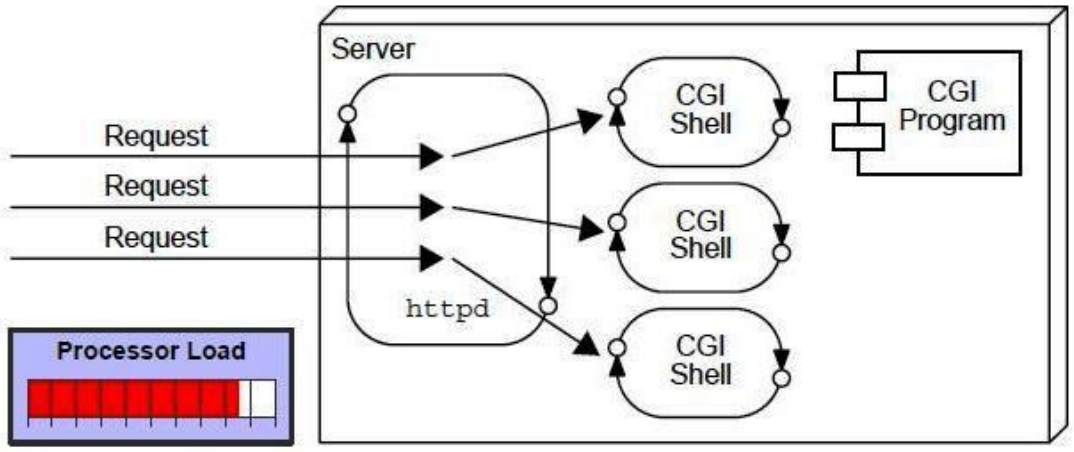
CGI (Common Gateway Interface)

- CGI tehnologija omogućava veb serveru da pozove eksterni program i prenese informacije HTTP-a na spoljni program za obradu zahteva; za svaki zahtev se započinje novi proces



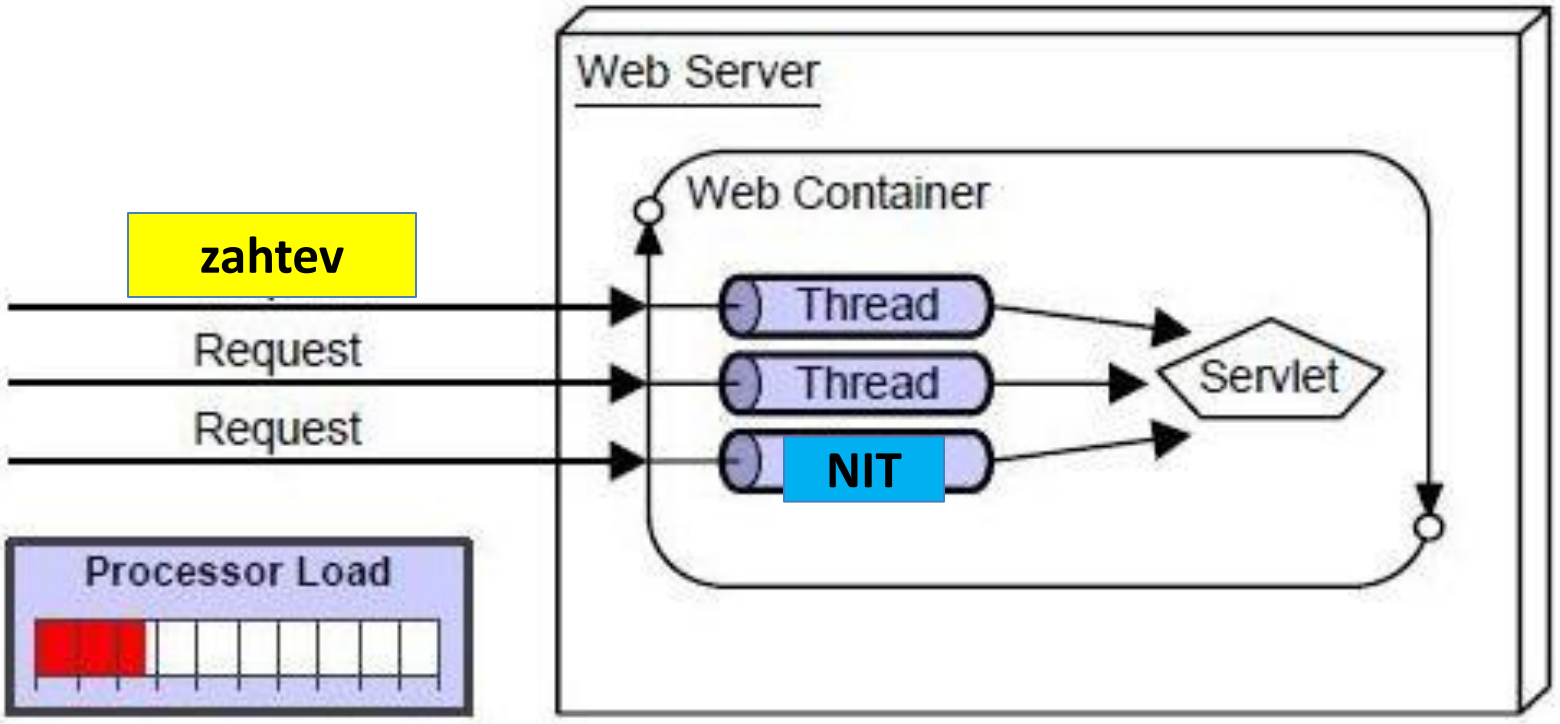
Nedostaci CGI

- Postoji mnogo problema u CGI tehnologiji:
  1. Ako se broj klijenata povećava,
     potrebno je više vremena za slanje odgovora
  2. Za svaki zahtev, započinje se proces i
     Veb server je ograničen na pokretanje procesa
  3. Koristi jezik koji zavisi od platforme



Servlet – CGI

- Servlet imaju brojne prednosti u odnosu na CGI



Servlet prednosti

- veb kontejner stvara niti za rukovanje višestrukim zahtevima servleta
- Niti imaju puno prednosti u procesima kao što su zajednička memorija,
  troškovi komunikacije između niti su mali

1. **Bolje performanse**:
   zato što stvara nit za svaki zahtev koji nije proces
2. **Prenosivost i sigurnost**: jer koristi Java jezik
3. **Robustnost**: servletima upravlja JVM,
   tako da se ne mora brinuti o memoriji i hardveru

Veb sajt i lokacija

- **Veb sajt** je zbirka povezanih veb stranica koje mogu sadržavati tekst, slike, audio i video;
  prva stranica veb stranice se zove home page
  - svaka veb lokacija ima određenu internet adresu (URL) koja se mora uneti u pregledač da bi se pristupilo veb lokaciji
  - sajt je hostovan na jednom ili više servera i može se pristupiti posećivanjem njegove početne stranice pomoću računarske mreže
  - sajtom upravlja vlasnik (pojedinac, kompanija, organizacija)
- **Statička veb lokacija** je osnovni tip veb stranice, jednostavna je za kreiranje; ne treba internet programiranje i dizajn baze podataka
  - Kod je fiksan za svaku stranicu;
    informacije na stranici se ne menjaju;
    izgleda kao štampana stranica

Dinamička veb lokacija

- Skup dinamičkih veb stranica čiji se sadržaj dinamički menja
  - Pristupa se sadržaju iz baze podataka ili Content Management Sistem-a (CMS)
  - Sadržaj veb stranice se menja kada se menja ili ažurira sadržaj baze podataka
  - Za generisanje dinamičkog sadržaja dinamička veb lokacija koristi skripte na klijent strani ili skripte na serveru
  - Skripte na klijent strani generišu sadržaj na klijentovom računaru, a na osnovu korisničkog unosa; veb pretraživač preuzima veb stranicu sa servera i obrađuje na korisničkoj stranici
  - Kod skripti na serveru, softver se pokreće na serveru i procesiranje se završava na serveru, zatim se korisnicima šalju jednostavne html stranice

GET i POST

- GET zahteva podatke od specificiranog resursa
  - Ograničena količina podataka se može poslati (u zaglavlju)
  - Zahtev nije sigurnosni zato što su podaci vidljivi u URL
  - Zahtev može biti bookmarkiran
  - Zahtev je idempotentan (operacija daje isti rezultat);
    2. zahtev se ignoriše dok se ne dostavi odgovor na 1. zahtev
  - Zahtev se efikasnije i češće koristi
- POST prosleđuje obrađene podatke određenom resursu
  - Velika količina podataka može se poslati (u telu)
  - Zahtevi su sigurnosni jer podaci nisu vidljivi u URL-u
  - Zahtev ne može biti bookmarkira
  - Zahtev nije idempotentan
  - Zahtev je manje efikasan i ređe se koristi

GET i POST

Veb terminologija

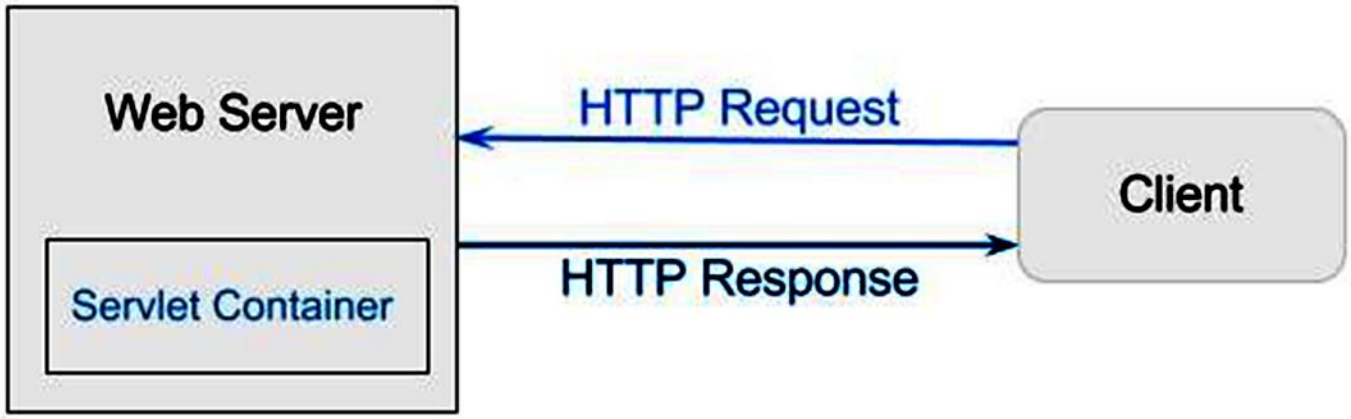
- GET
  - Ostaje u istoriji pregledača
  - Može da se bookmarkira
  - Može da bude keširan
  - Ima ograničenja dužine
  - Ne treba da se koristi za osetljive (poverljive) podatke
  - Treba da se koristiti samo za preuzimanje podataka
- POST
  - Zahtev ne može biti bookmarkiran
  - Zahtevi nemaju ograničenja u dužini podataka
  - Zahtevi se nikada ne keširaju
  - Zahtevi ne ostaju u istoriji pregledača

HTTP zahtev (request)

- zahtev koji računar šalje na veb server koji ima potencijal da pruži informaciju:
  - Linija zahteva
  - Analiza izvorne IP adrese, proksi, porta
  - Analiza odredišne IP adrese, protokola, porta, hosta
  - Zahtevani URI (Uniform Resource Identifier)
  - Zahtevana metoda **(GET, POST, HEAD, TRACE, PUT, DELETE, OPTIONS)** i sadržaj
  - Zaglavlje korisničkog agenta
  - Zaglavlje kontrole veze
  - Zaglavlje keš kontrole

Kontejner (Servlet Container)

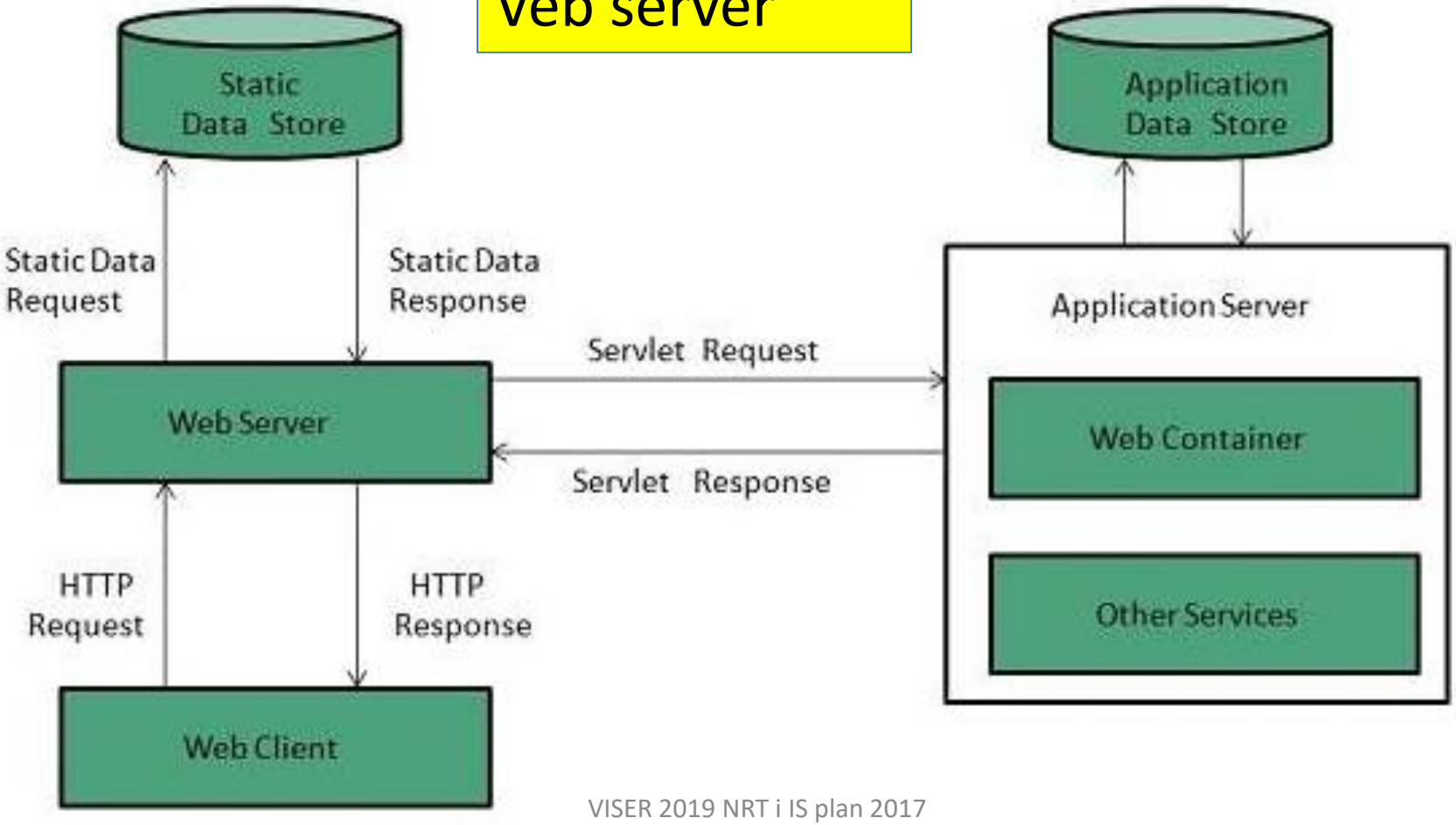
- koristi se u Javi za dinamičko generisanje veb stranica na serveru
  - obezbeđuje radno okruženje za Java aplikacije; klijent (korisnik) može da zahteva samo statičku veb stranu sa servera; kontejner servleta koristi u Javi za učitavanje podataka
  - koristi se u Javi za dinamičko generisanje veb stranica na strani servera; kontejner servleta je deo veb servera koji komunicira sa servletom za rukovanje dinamičnim veb stranama sa klijentom



Server: veb ←→ aplikacije

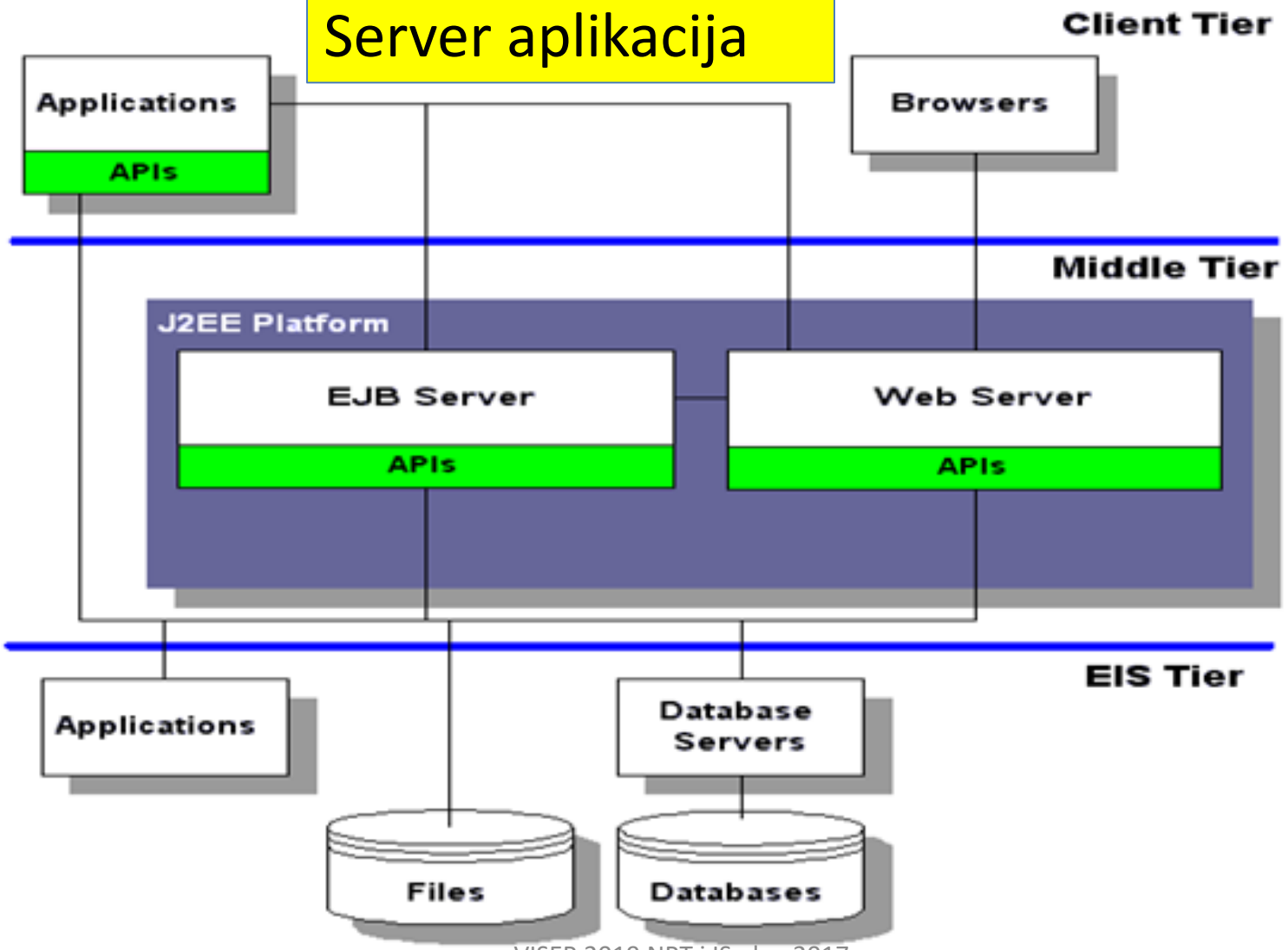
- Web Server: Apache Tomcat

Veb server



Server: veb ←→ aplikacije

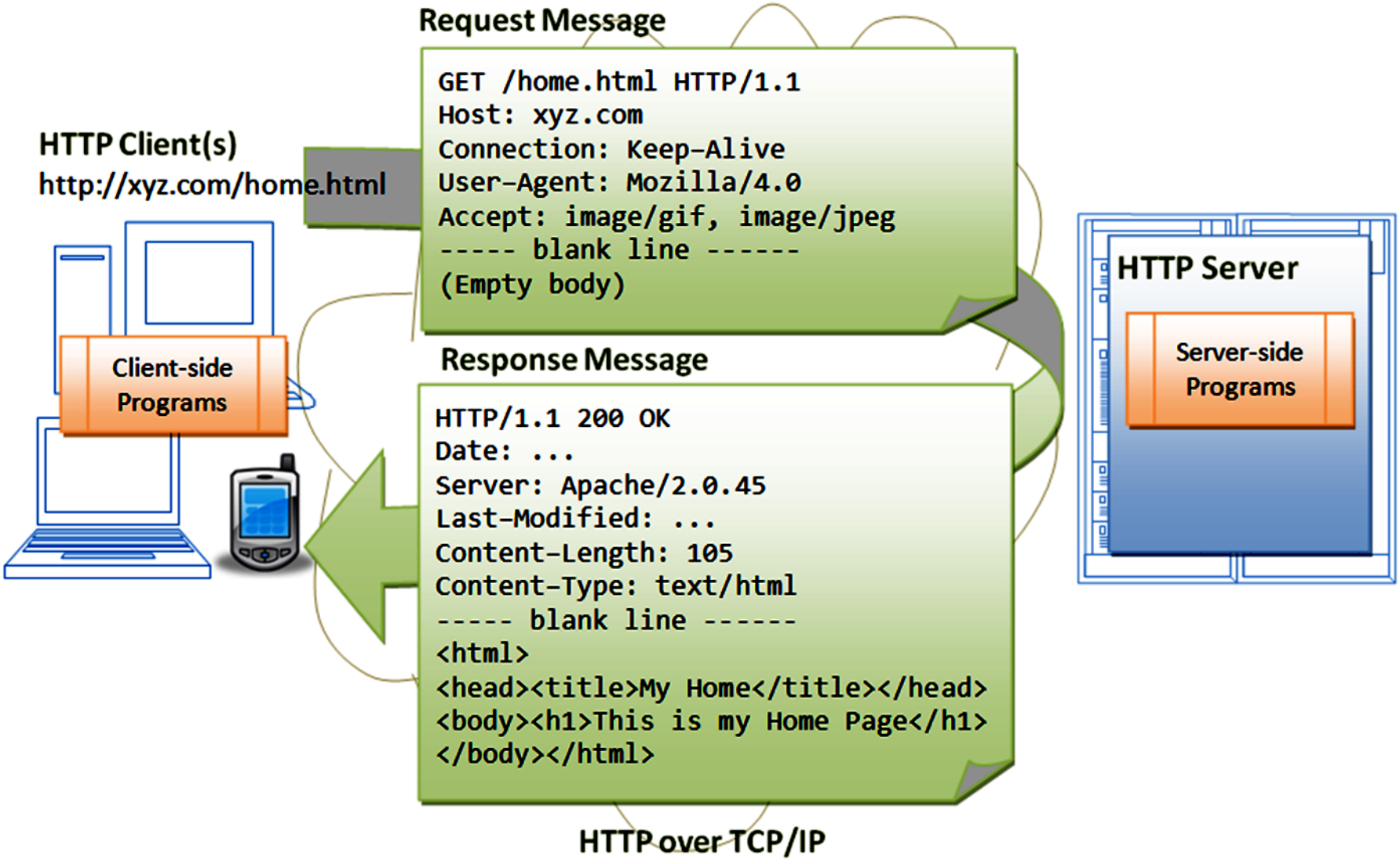
Server aplikacija



Content Type

- text/html
- text/plain
- application/msword
- application/vnd.ms-excel
- application/jar
- application/pdf
- application/octet-stream
- application/x-zip
- images/jpeg
- images/png
- images/gif
- audio/mp3
- video/mp4
- video/quicktime

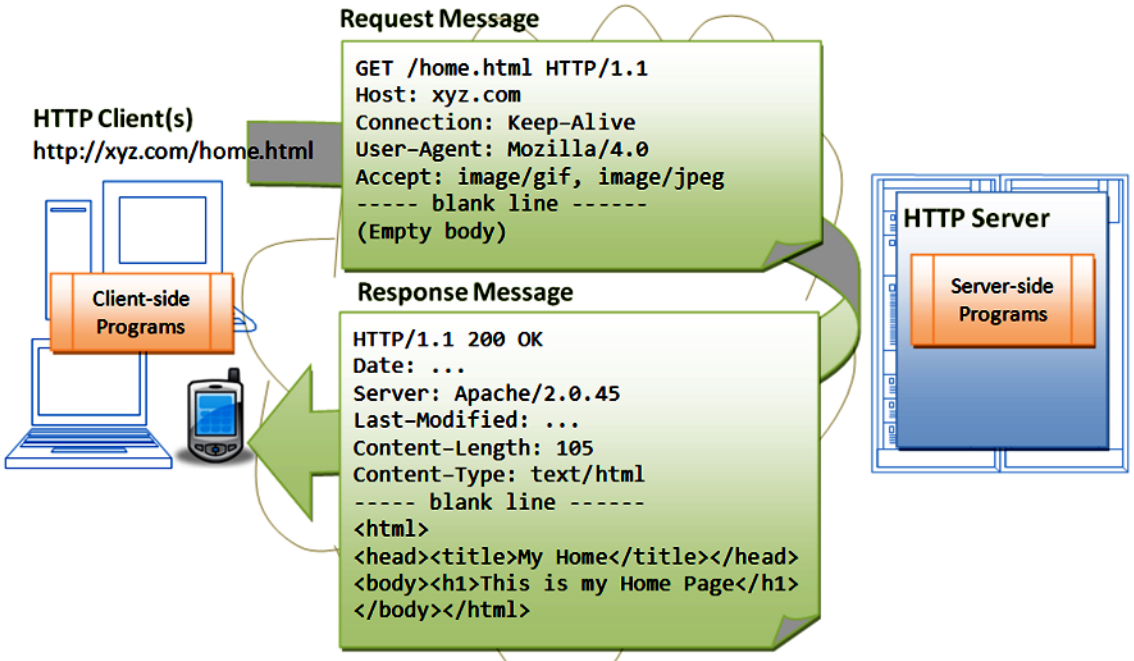
Hypertext Transfer Protocol (HTTP)



HTTP

Veb terminologija

- HTTP je protokol aplikacijskog sloja koji se pokreće preko TCP/IP; IP pruža podršku za usmeravanje i adresiranje (putem jedinstvene IP adrese za mašine na Internetu); TCP podržava multipleksiranje preko portova: podrazumevani broj porta dodeljen HTTP-u je TCP port 80.
- HTTP je asinhroni protokol aplikacijskog sloja zahteva i odgovora: klijent šalje zahtev serveru: server zatim vraća klijentu poruku sa odgovorom (sintaksa poruke je definirana u HTTP specifikaciji)

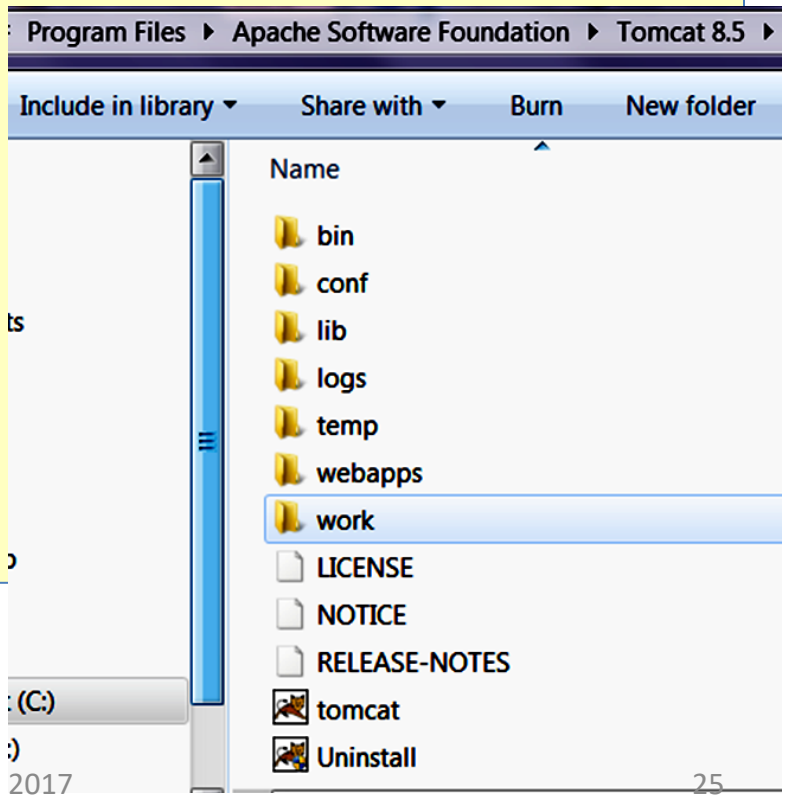


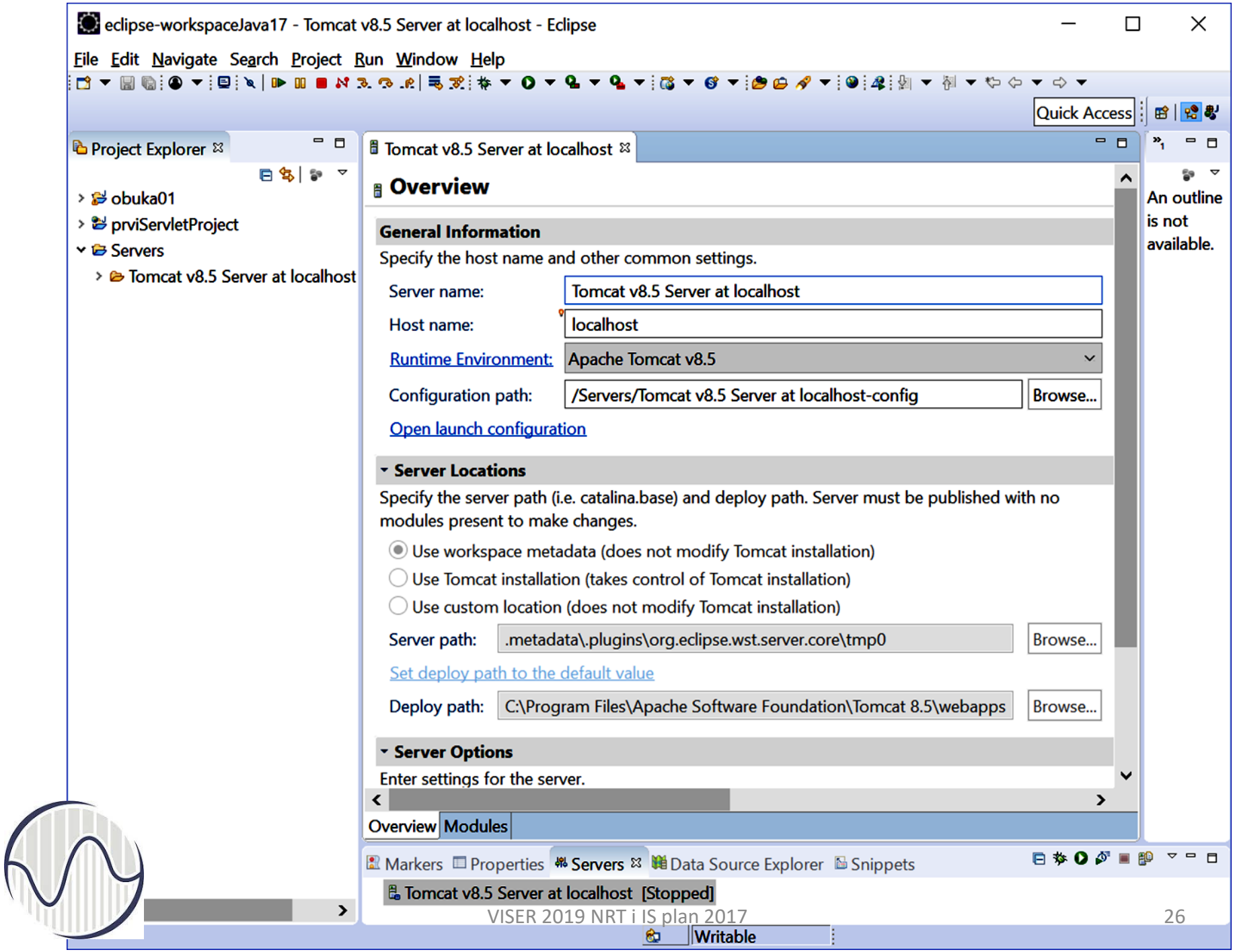
Apache Tomcat HTTP Server

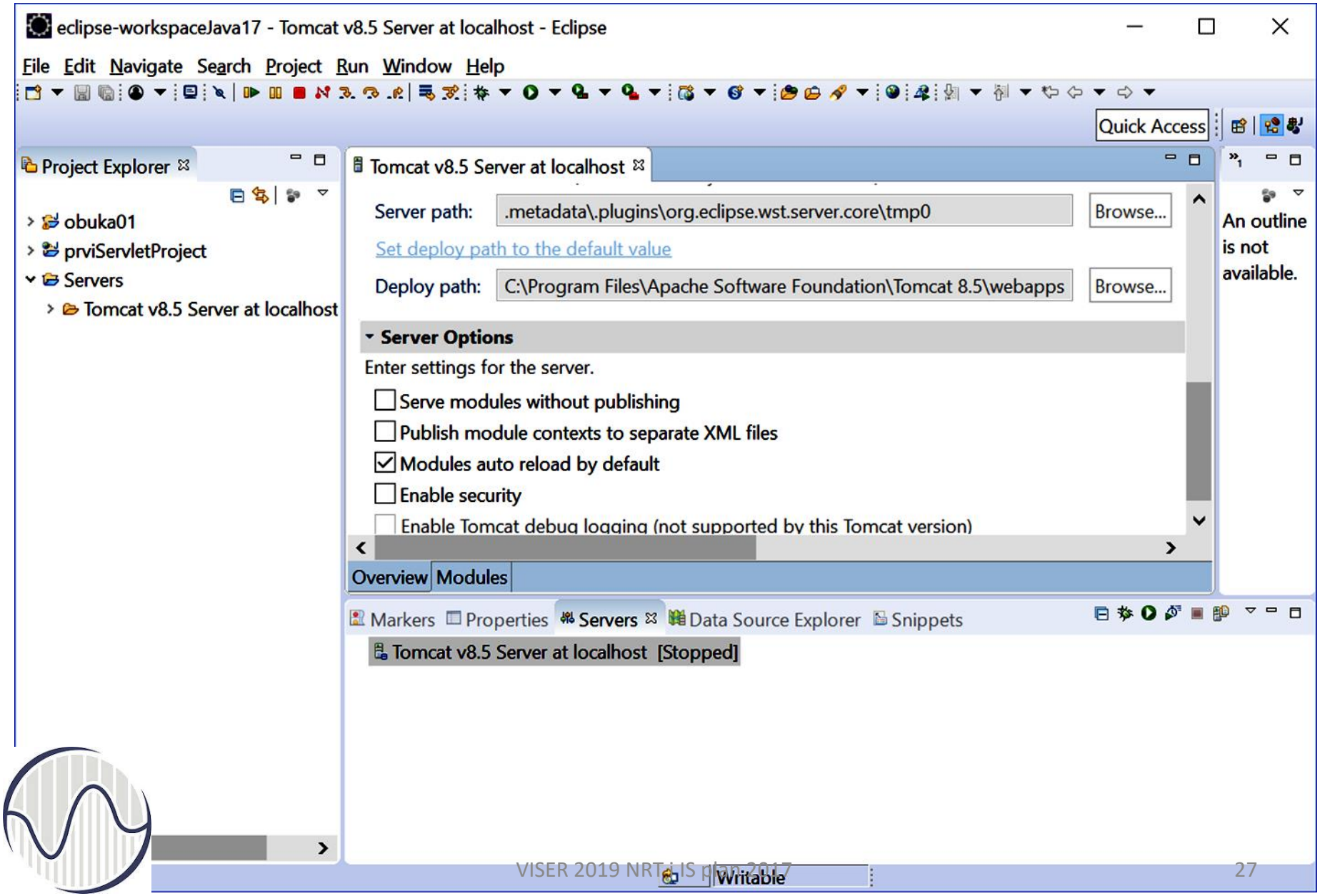
- Apache Tomcat je Java-orjentisan HTTP server,
  koji izvršava Java programe (Java Servlet, Java Server Pages):
  najkorišćeniji Apache HTTP server otvorenog koda
- Tomcat 8.x: Servlet 3.1, JSP 2.3, EL 3.0, Java WebSocket 1.0
- pokreće se preko TCP/IP
- radi na specifičnom TCP portu sa određene IP adrese
- podrazumevani broj TCP porta za HTTP protokol je 80
- za test HTTP server, možete izabrati bilo koji neiskorišćen port

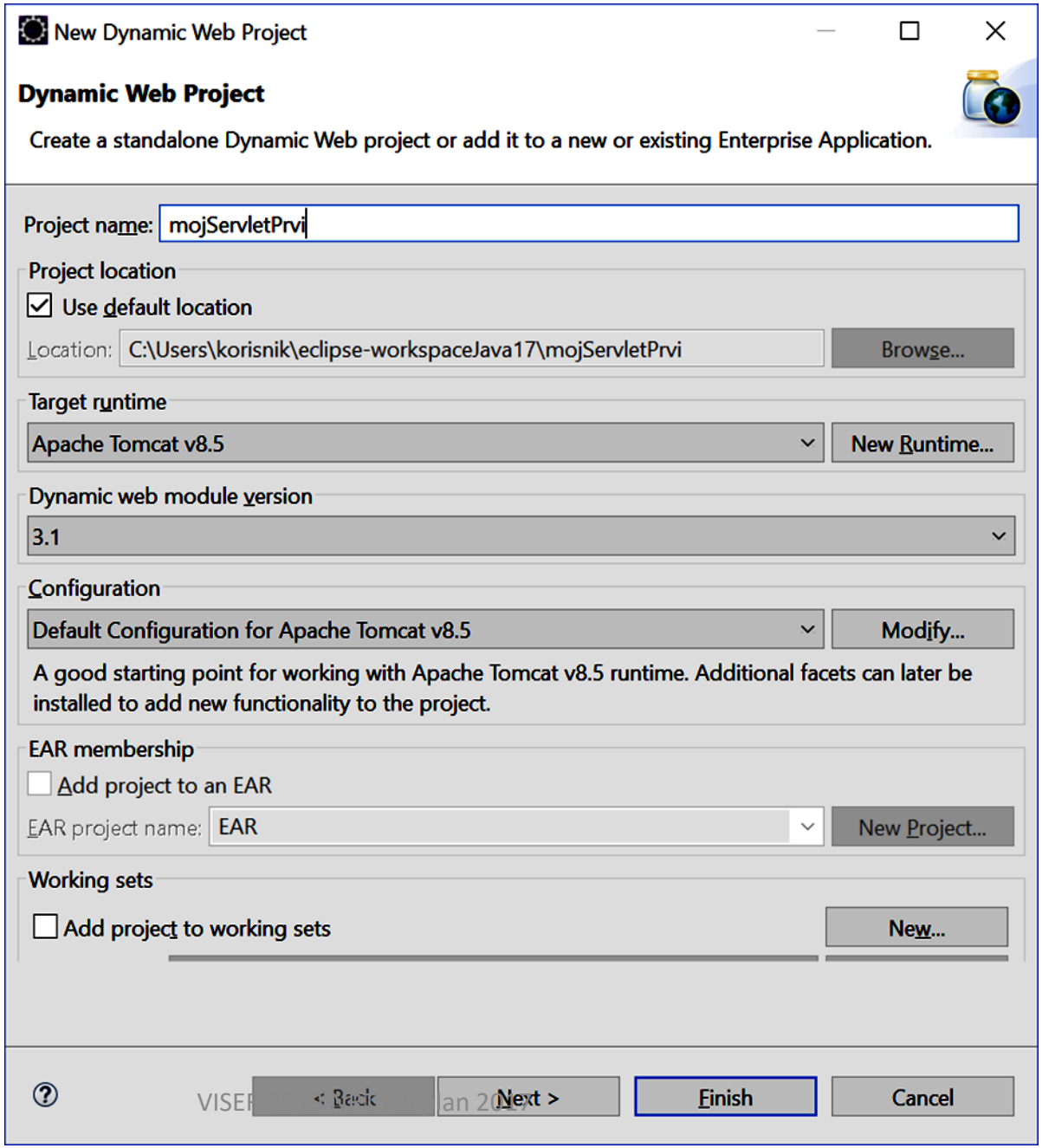
Instalacija Tomcat 8, Tomcat folderi:

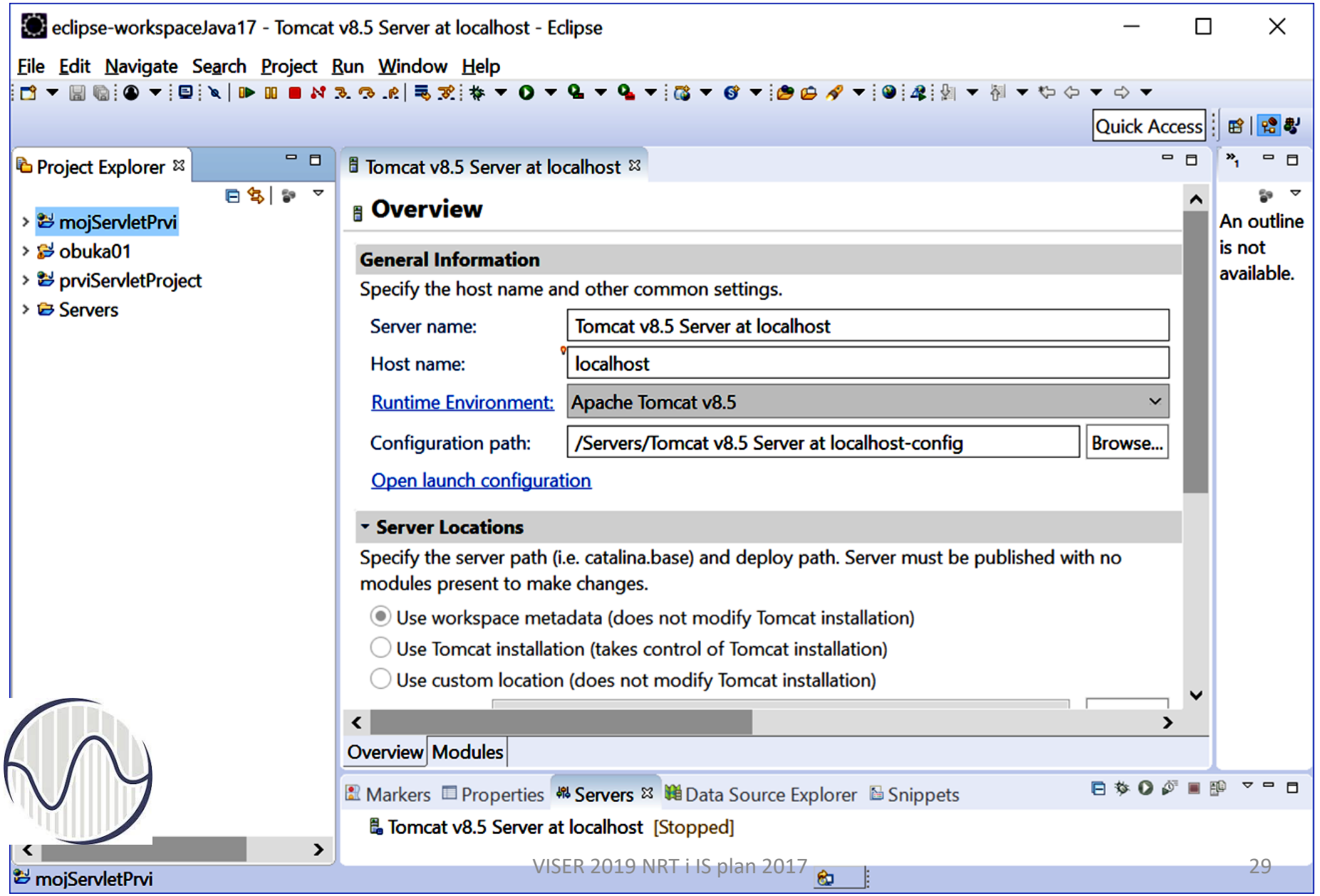
- Sa sajta tomcat.apache.org preuzme se instalacioni paket, startuje se ako je aplikacija i instalira kao i svaka druga aplikacija na računaru
- **bin**: sadrži binarne i skript fajlove, startup script (startup.bat for Windows), shutdown script (shutdown.bat for Windows)
- **conf**: sadrži konfiguracione fajlove; server.xml, web.xml, context.xml, tomcat-users.xml
- **lib**: sadrži Tomcat system JAR files, kojima pristupaju sve vebap; mogu da budu i eksterni JAR fajlovi (MySQL JDBC Driver)
- **logs**: Tomcat log fajlovi; error poruke
- **webapps**: sadrži vebap koje će se razvijati ; Webapp Archive
- **work**: Tomcat radni folder koji koriste JSP, JSP-to-Servlet konverzija
- **temp**: privremeni fajlovi




eclipse-workspaceJava17 - Tomcat v8.5 Server at localhost - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Project Explorer
obuka01
prviServletProject
Servers
Tomcat v8.5 Server at localhost
Tomcat v8.5 Server at localhost
Overview
General Information
Specify the host name and other common settings.
Server name:
Tomcat v8.5 Server at localhost
Host name:
localhost
Runtime Environment:
Apache Tomcat v8.5
Configuration path:
/Servers/Tomcat v8.5 Server at localhost-config
Browse...
Open launch configuration
Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.
Use workspace metadata (does not modify Tomcat installation)
Use Tomcat installation (takes control of Tomcat installation)
Use custom location (does not modify Tomcat installation)
Server path:
.metadata\.plugins\org.eclipse.wst.server.core\tmp0
Browse...
Set deploy path to the default value
Deploy path:
C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps
Browse...
Server Options
Enter settings for the server.
Overview
Modules
An outline is not available.
Markers
Properties
Servers
Data Source Explorer
Snippets
Tomcat v8.5 Server at localhost [Stopped]
Writable


eclipse-workspaceJava17 - Tomcat v8.5 Server at localhost - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Project Explorer
obuka01
prviServletProject
Servers
Tomcat v8.5 Server at localhost
Tomcat v8.5 Server at localhost
Server path:
.metadata\.plugins\org.eclipse.wst.server.core\tmp0
Browse...
Set deploy path to the default value
Deploy path:
C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps
Browse...
Server Options
Enter settings for the server.
Serve modules without publishing
Publish module contexts to separate XML files
Modules auto reload by default
Enable security
Enable Tomcat debug logging (not supported by this Tomcat version)
Overview
Modules
An outline is not available.
Markers
Properties
Servers
Data Source Explorer
Snippets
Tomcat v8.5 Server at localhost [Stopped]
Writable

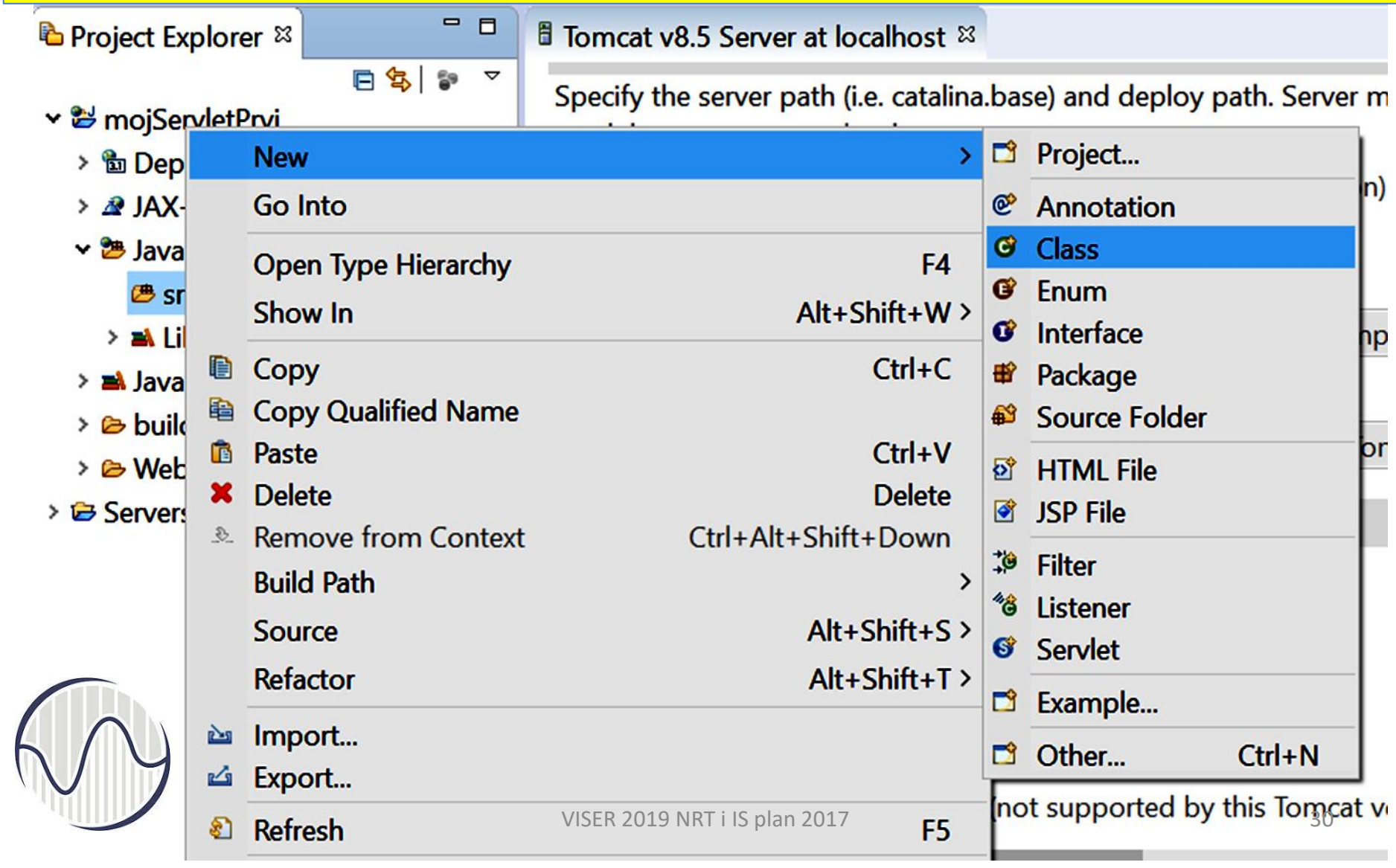

New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: mojServletPrvi
Project location
Use default location
Location: C:\Users\korisnik\eclipse-workspaceJava17\mojServletPrvi
Browse...
Target runtime
Apache Tomcat v8.5
New Runtime...
Dynamic web module version
3.1
Configuration
Default Configuration for Apache Tomcat v8.5
Modify...
A good starting point for working with Apache Tomcat v8.5 runtime. Additional facets can later be installed to add new functionality to the project.
EAR membership
Add project to an EAR
EAR project name: EAR
New Project...
Working sets
Add project to working sets
New...
< Back
Next >
Finish
Cancel


eclipse-workspaceJava17 - Tomcat v8.5 Server at localhost - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Project Explorer
mojServletPrvi
obuka01
prviServletProject
Servers
Tomcat v8.5 Server at localhost
Overview
General Information
Specify the host name and other common settings.
Server name:
Tomcat v8.5 Server at localhost
Host name:
localhost
Runtime Environment:
Apache Tomcat v8.5
Configuration path:
/Servers/Tomcat v8.5 Server at localhost-config
Browse...
Open launch configuration
Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.
Use workspace metadata (does not modify Tomcat installation)
Use Tomcat installation (takes control of Tomcat installation)
Use custom location (does not modify Tomcat installation)
Overview
Modules
An outline is not available.
Markers
Properties
Servers
Data Source Explorer
Snippets
Tomcat v8.5 Server at localhost [Stopped]
mojServletPrvi

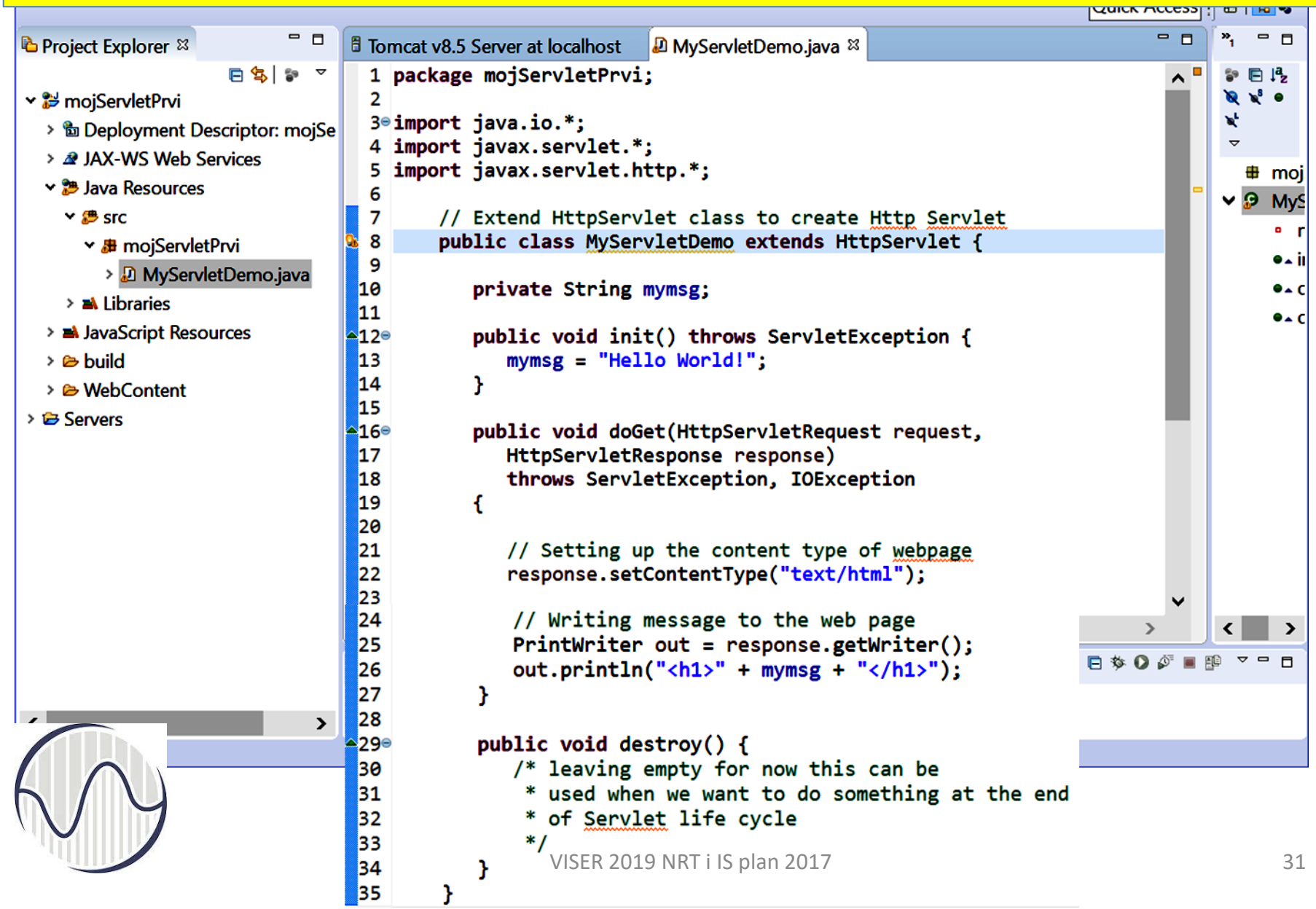
Kreiranje servlet klase kao ekstenzija HttpServlet klase

Desni klik na **src** folder i kreira se fajl – nova klasa **MyServletDemo**.

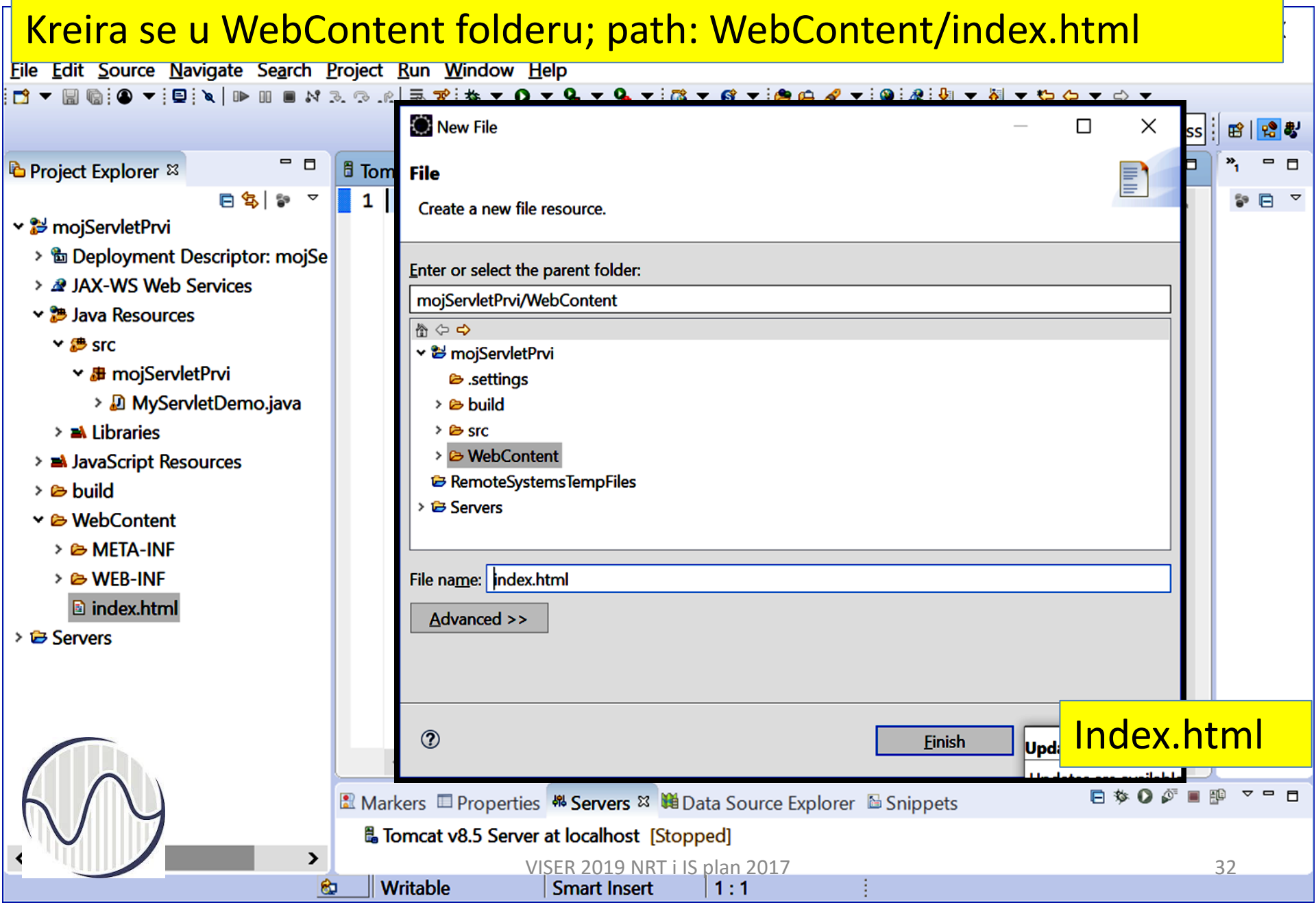
Putanja: Java Resources/src/default package/MyServletDemo.java



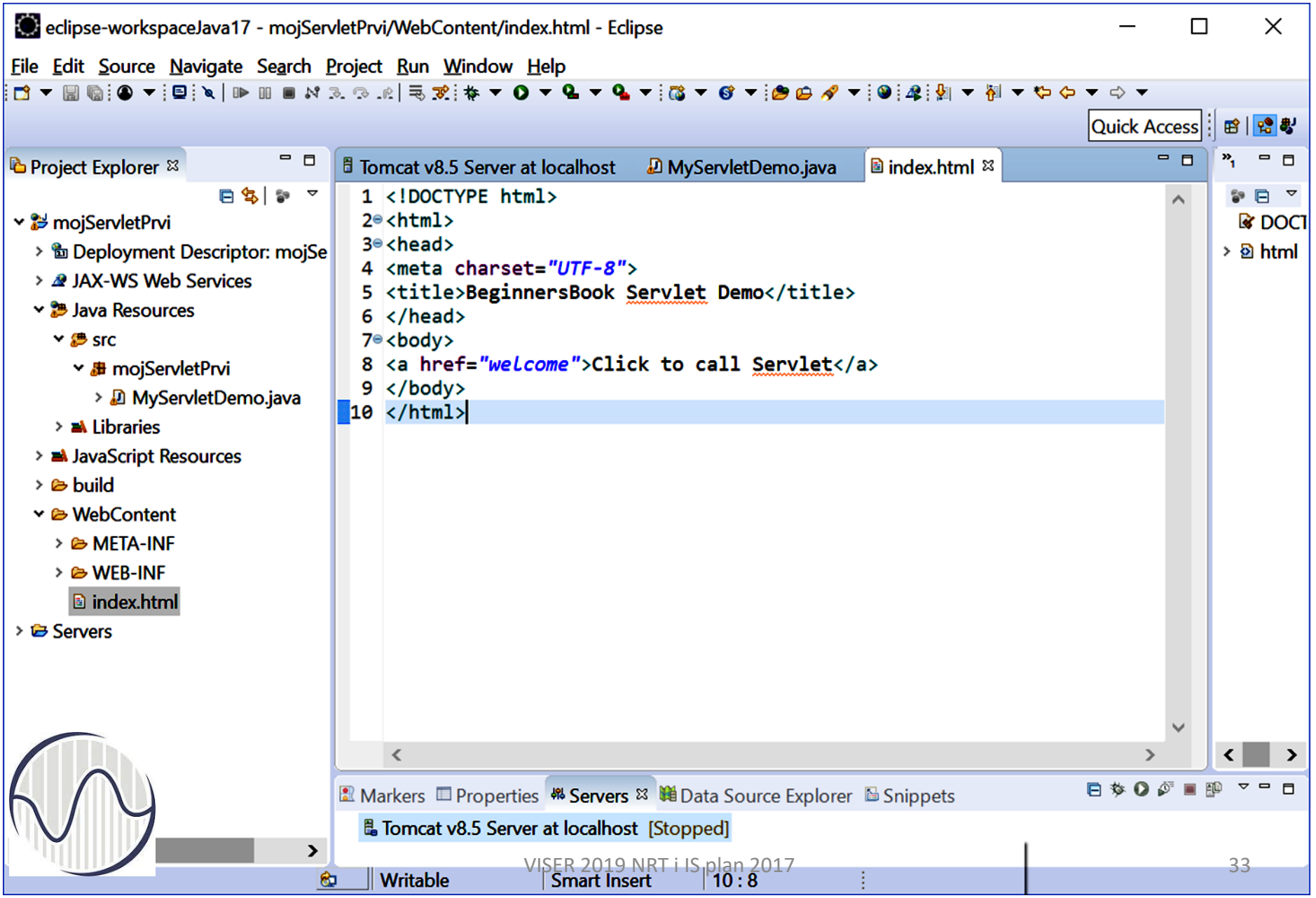
Napiše se u package/MyServletDemo.java kao u primeru putanja:moj Java Resources/src/default package/MyServletDemo.java



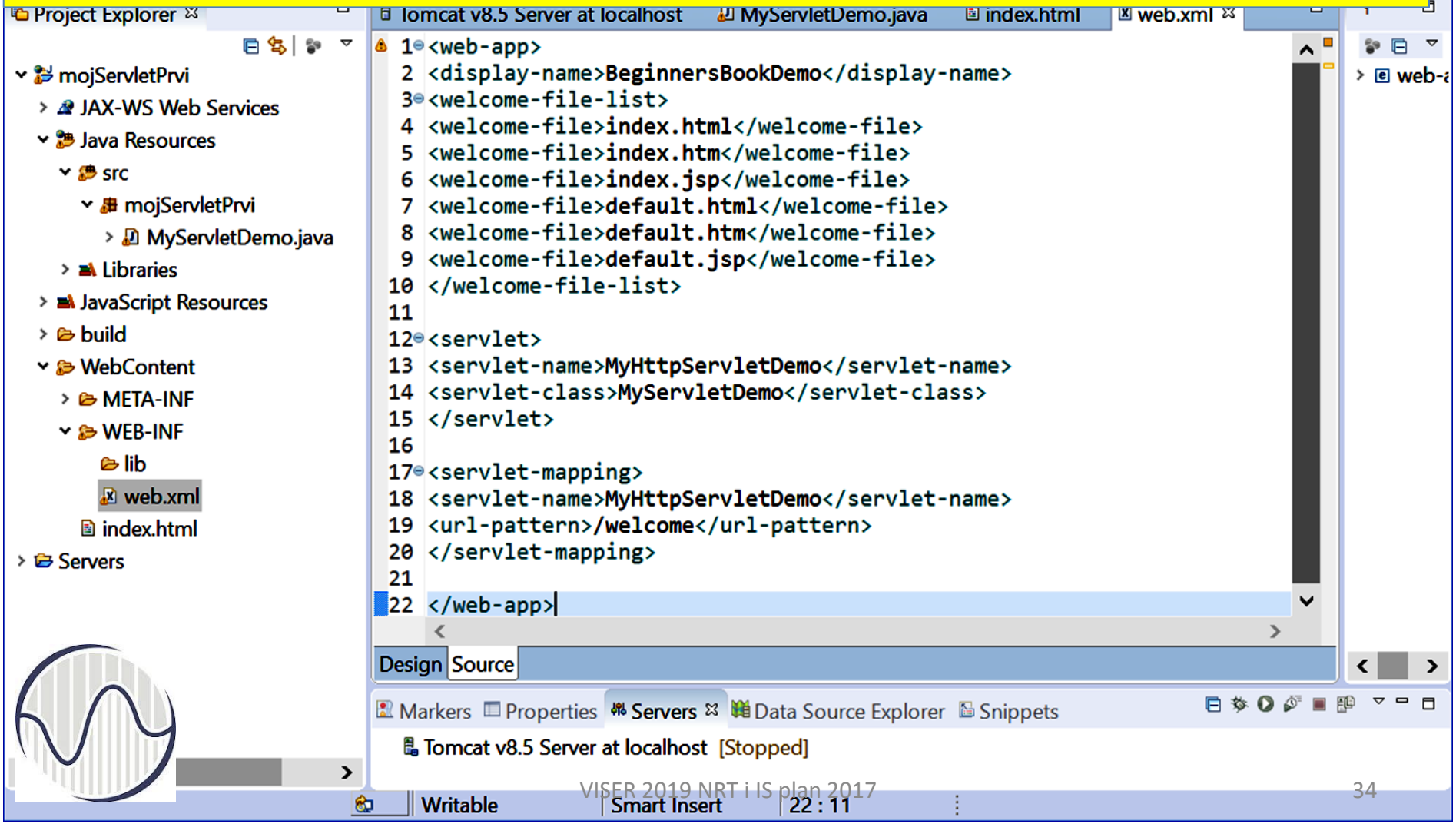
Kreira se html fajl koji će pozvati servlet kada se klikne na veb stranu.
Kreira se u WebContent folderu; path: WebContent/index.html



WebContent/index.html

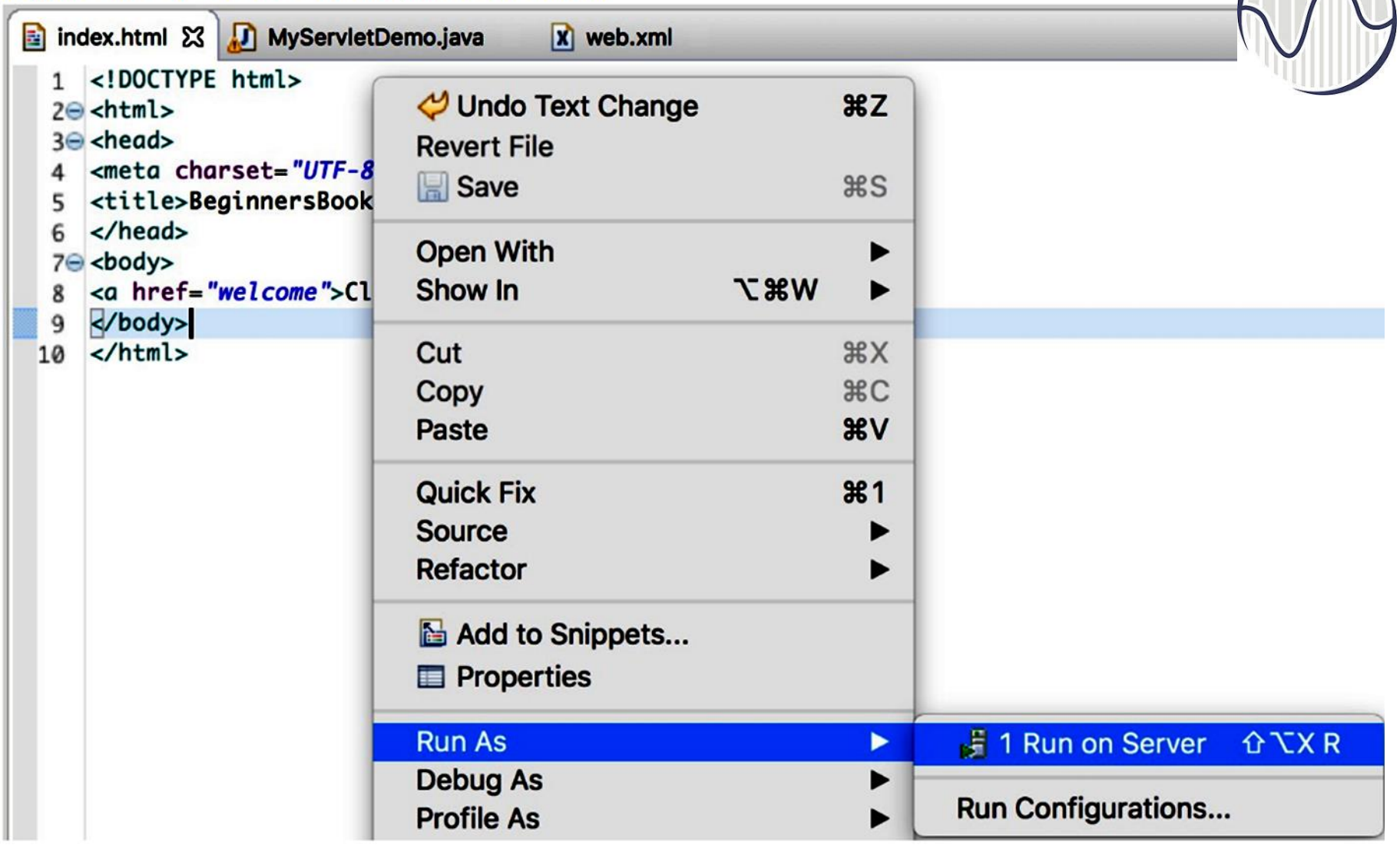


Kreirati web.xml fajl; WebContent/WEB-INF/web.xml
Mapirati Servlet sa specificiranim URL
Kako se poziva welcome page nakon klika, na index.html, mapiramo welcome page u servlet klasu koja je kreirana

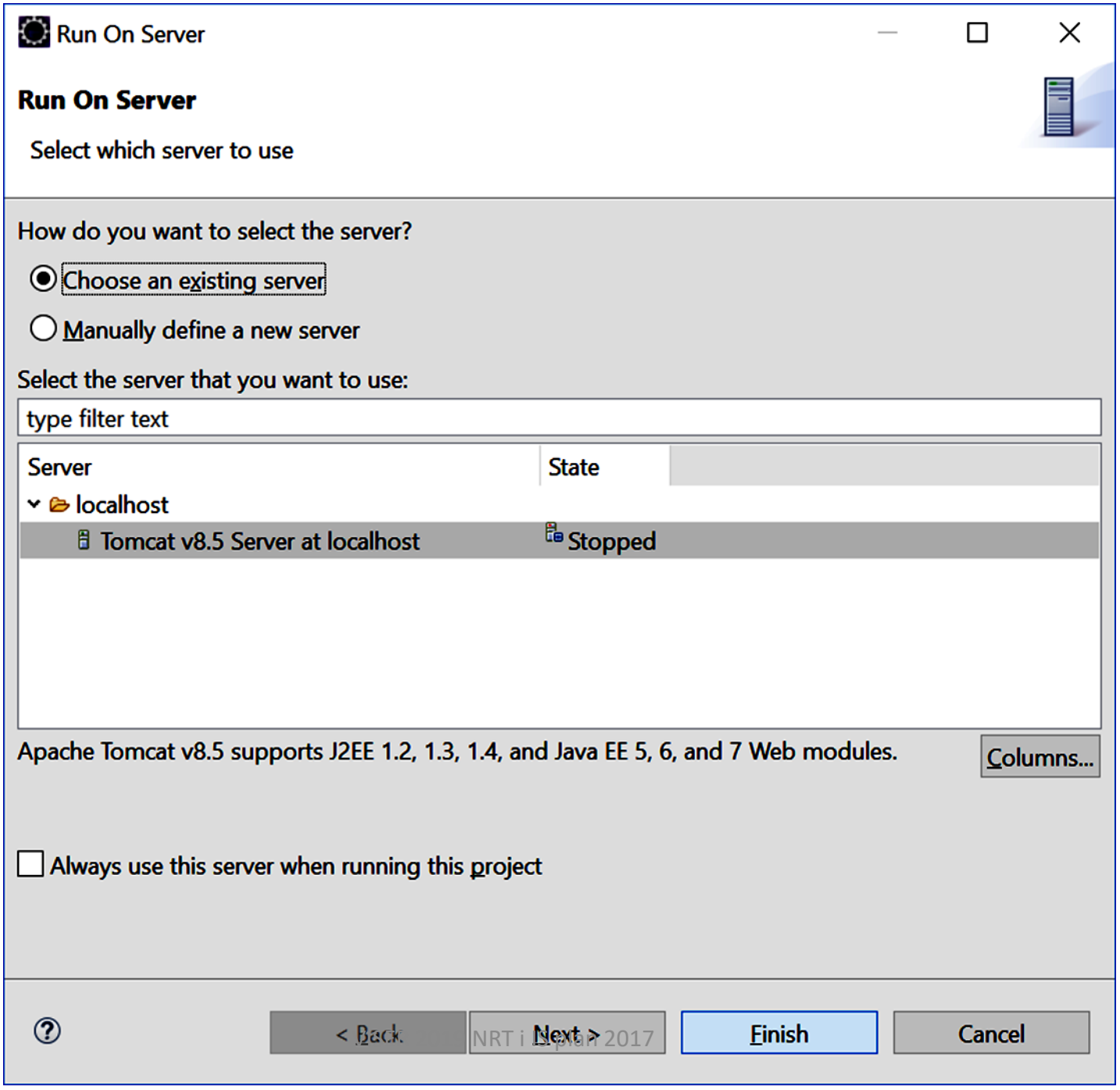


Run the project:

Right click on the index.html, run on server.



Click Add All to deploy the Project on Server. Click Finish


Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v8.5 Server at localhost
Stopped
Apache Tomcat v8.5 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel

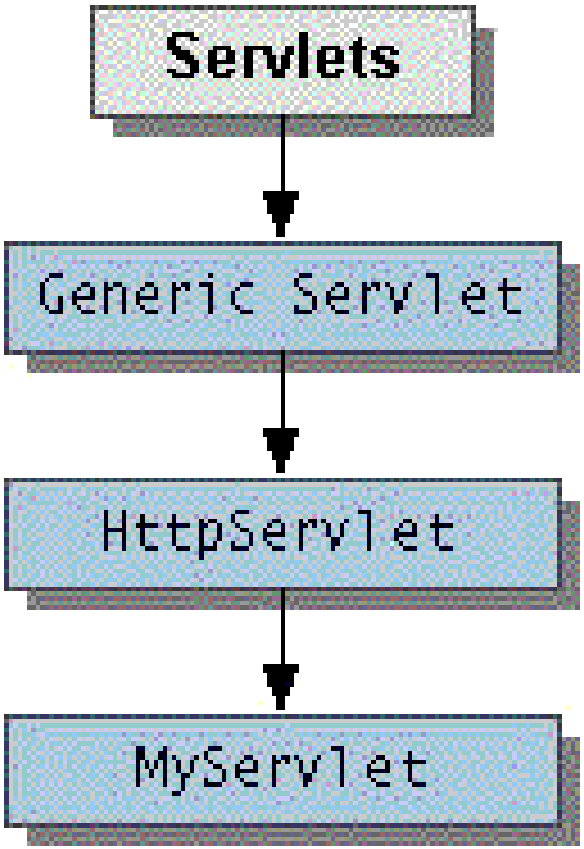
Primeri jednostavnog servleta

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```

Arhitektura Servlet Paketa

- Servleti koriste paket javax.servlet, koji sadrži interfejs i klase za njihovo pisanje
- Svi servleti implementuju interfejs Servlets, ili nasleđuju klasu koja je implementirala ovaj interfejs
- Najkorišćenija klasa ove vrste je klasa HttpServlet.



Interakcija sa klijentom

- Kada servlet dobije poziv od strane klijenta, on prihvata dva objekta
- **ServletRequest**, koji održava komunikaciju od klijenta do servera
- **ServletResponse**, koji održava komunikaciju od servera ponovo ka klijentu
- ServletRequest and ServletResponse su interfejsi definisani u javax.servlet paketu

ServletRequest

ServletRequest interfejs dozvoljava servletu da:

- Pristupi informacijama (imena parametara) koje šalje klijent, protokol koji koristi klijent, i ime udaljenog računara koji je poslao zahtev koji je server primio.
- ServletInputStream - servleti dobijaju podatke od klijenta koji su poslati preko određenog protokola.

```
<FORM name="MojaPostForma" action="ProbaServlet"
method=POST>
</FORM>
```

- ili

```
<FORM name="MojaGetForma"
action="ProbaServlet?par1=88" method=GET>
</FORM>
```

- HttpServletRequest interfejs sadrži metode koje pristupaju HTTP header informacijama

ServletResponse

- ServletResponse interfejs omogućava servletima metode za generisanje odgovora
- Dozvoljava servletu da definiše sadržaj odgovora i tip podataka (MIME)
- Obezbeđuje izlazni stream, ServletOutputStream i Writer preko kojih servlet šalje podatke

Dodatne mogućnosti HTTP Servleta

- HTTP servleti imaju dodatne objekte za rad sa sesijama
- Može da se sačuva stanje promenljivih između više poziva sa klijentske i serverske strane
- HTTP servleti imaju i objekte koji omogućavaju rad sa cookie

Generisanje HTML koda

- Potrebno je obavestiti pregledač da se šalju podaci koji su tipa HTML

```
response.setContentType("text/html");
```

- Menja se naredba println za korektnu veb stranicu
- Naredbe print koriste HTML tagove
- Provera dobijenog HTML koda sa formalnim sintaksnim validatorima

Servlet

```
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String docType =
"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 "+
"Transitional//EN\">\n";
out.println(docType +
"<HTML>\n" +
"<HEAD><TITLE>Hello</TITLE></HEAD>\n"+
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<H1>Hello</H1>\n" +
"</BODY></HTML>");
}
}
```

```
public class PrviServlet extends HttpServlet
{
public void doGet (HttpServletRequest zahtev,
HttpServletResponse odgovor)
throws ServletException, IOException
{
PrintWriter out;
String title = "Jednostavan odgovor Servera";
odgovor.setContentType("text/html");
out = odgovor.getWriter();
out.println("<HTML><HEAD><TITLE>");
out.println(title);
out.println("</TITLE></HEAD><BODY>");
out.println("<H1>" + title + "</H1>");
out.println("<P>Ova strana je generisana pomocu
Servleta.");
out.println("</BODY></HTML>");
out.close();
}
}
```

Generisanje HTML koda

- PrviServlet nasleđuje klasu HttpServlet, koja je implementirala Servlet interface.
- PrviServlet definiše svoj doGet metod. Ovaj metod se poziva kada klijent generiše GET zahtev i kao rezultat se jednostavna HTML stranica vraća klijentu. Zahtevi od strane klijenta su sadržani u HttpServletRequest objektu.
- Odgovor klijentu je u HttpServletResponse.
- Zato što se tekstualni podaci vraćaju klijentu, odgovor je poslat pomoću objekta Writer koji je sadržan u okviru HttpServletResponse objekta.

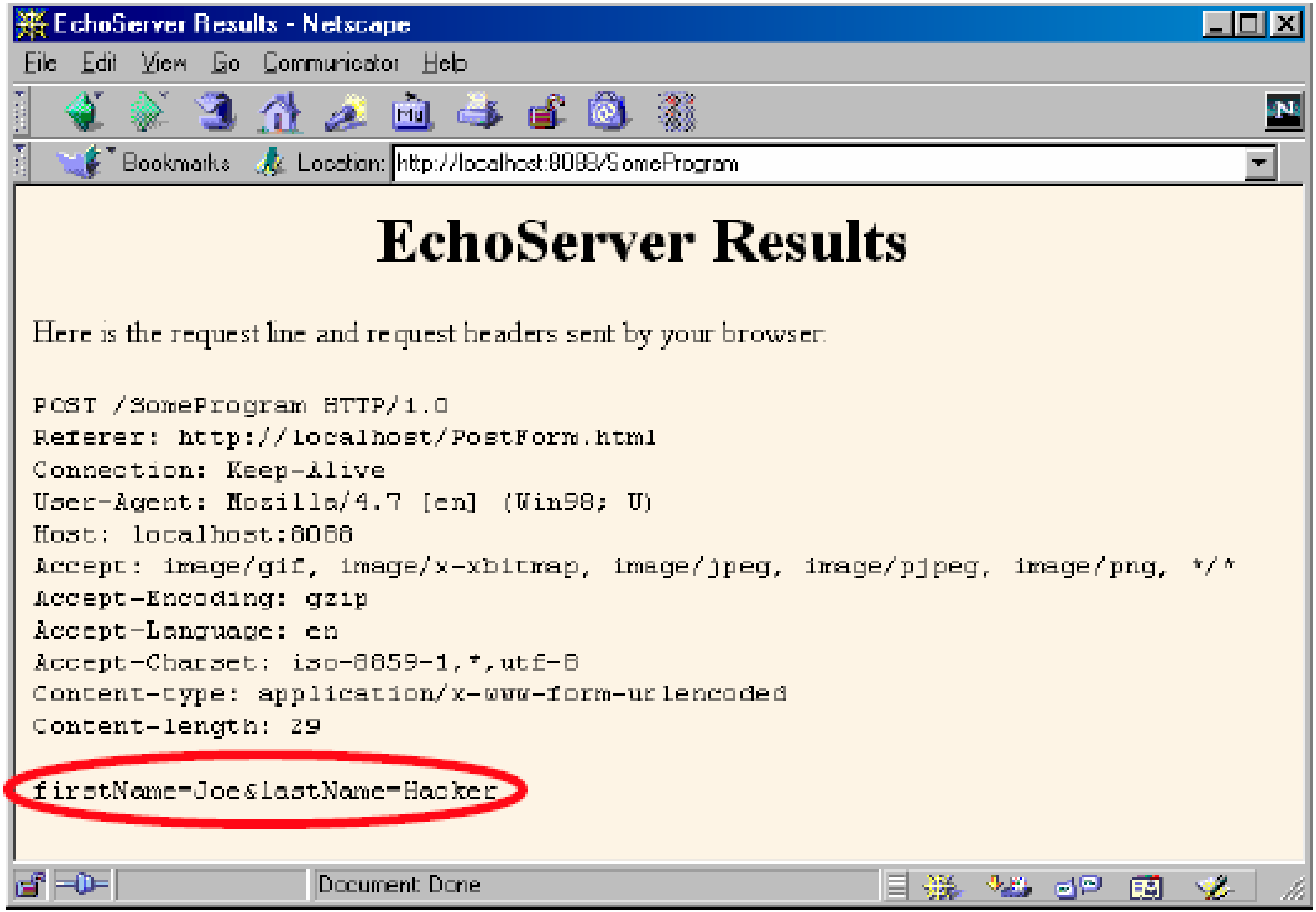
HttpServletRequest

- HttpServletRequest objekat omogućava pristup HTTP header podacima
- Omogućava prihvatanje argumenata koje klijent šalje kao deo svog zahteva
- Podacima koje šalje klijent može se pristupiti sa getParameter metodom, koji vraća vrednost imenovanog parametra
- Ako parametar ima više od jedne vrednosti, koristi se metod getParameterValues
- Metod getParameterValues vraća niz vrednosti za imenovani parametar
- Metod getParameterNames vraća imena svih parametera

HttpServletResponse

- Objekat HttpServletResponse daje
  dve mogućnosti za vraćanje podataka klijentu
- Metod getWriter vraća Writer
- Metod getOutputStream vraća ServletOutputStream
- Metoda getWriter se koristi
  kada su podaci koji se vraćaju klijentu tekstualnog tipa,
  metod getOutputStream za binarne podatke
- Pozivanje metoda close ovih objekata nakon slanja
  odgovora omogućava serveru da zna
  kada je odgovor komplentiran

GET i POST zahtev



GET zahtev

- **GET zahtev se obrađuje preklapanjem metoda doGet.**

```
public class ObradaGetMetodaServlet extends HttpServlet
{
public void doGet (HttpServletRequest zahtev,
HttpServletResponse odgovor)throws ServletException,
IOException
{ ...
odgovor.setContentType("text/html");
PrintWriter out = odgovor.getWriter();
out.println("<html>" +
"<head><title>Primer citanja vrednosti
parametra</title></head>" + ...);
String servletPar1 = zahtev.getParameter("par1");
if (servletPar1 != null) {
out.println("<body>" + servletPar1);
}
out.println("</body></html>");
out.close(); }
...}
```

POST zahtev

```
public class ObradaPostMetodaServlet extends
HttpServlet {
public void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
...
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>" + "<head><title>Primer
citanja vrednosti parametra</title></head>" +
...);
String servletPar1 = request.getParameter("par1");
if (servletPar1 != null) {
// prikazivanje procitane vrednosti
out.println("<body>" + servletPar1);
}
out.println("</body></html>");
out.close();}
...}
```

Poziv servleta

- **Servleti se mogu pozivati direktno upisom njihove URL putanje unutar browsera.**
- **Format URL putanje generalno zavisi od servera koji se koristi. Primer je za Jakarta Tomcat web server kod koga je je u direktorijumu TOMCAT_HOME/webapps instalirana aplikacija: (Tomcat) http://ime_mas:port/Dir_Aplik/Ime_Serv**
- **(Tomcat) http://localhost:8080/Proba/ObradaPostMetodaServlet**
- **Za Get metod poziva: (Tomcat) http://localhost:8080/Proba/ObradaGetMetodaServlet ?par1=88**

Poziv servleta sa HTML stranice

```
public class PozivHTMLServlet extends HttpServlet {
   public void doGet (HttpServletRequest request,
     HttpServletResponse response)
     throws ServletException, IOException {
  ...
  out.println(... +
     "<a href=\"" +
response.encodeURL("/Proba/ObradaPostMetodaServlet") +
   "\">Poziv servleta</a> " +
   ...);
 ...
 }
 ...
}
```

Poziv servleta sa HTML stranice

```
public class PozivHTMLServlet extends HttpServlet {
public void doGet (HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
...
out.println(... +
"<form action=\"" +
response.encodeURL("/Proba/ObradaPostMetodaServlet") +
"\" method=\"post\">" +
...
"<td><input type=\"text\" name=\"par1\"" +
"value=\"88\" size=\"19\"></td>" +
...
"<td><input type=\"submit\"" +
"value=\"Posalji informacije\"></td>" +
...
"</form>" +
...);
out.close();
}
...
}
```

Čitanje podataka sa forme

- request.getParameter(“ime")
  - Dobija se URL-dekodirana vrednost prvog elementa koji se zove ime
  - Ponaša se isto i za GET i za POST zahteve
  - Rezultat je null ako ne postoji takav parametar u elementima forme
- request.getParameterValues(“ime")
  - Dobija se niz URL-dekodiranih vrednosti za sve elemente koji se zovu ime
  - Dobija se niz sa jednim elementom, ako se ime pojavljuje samo jednom
  - Rezultat je null ako ne postoji takav parametar u elementima forme
- request.getParameterNames() ili request.getParameterMap()
  - Dobija se Enumeration ili Map objekti od poslatih elemenata
  - Uobičajeno je da je rezervisano za debagovanje

Životni ciklus servleta

- init
- Izvršava se jednom kada se servlet prvi put učitava
- Ne poziva se za svaki zahtev
- service
- Poziva se kod novog thread-a od strane servera za svaki zahtev
- Ne treba preklapati ovaj metod
- doGet, doPost, doXxx
- Obrađuju GET, POST, etc. zahteve
- Preklapanje ovih metoda omogućava željeno ponašanje servleta
- destroy
- Poziva se kada server briše instancu servleta
- Ne poziva se posle svakog zahteva

Servlet.service()

- **Svaki poziv servleta se svodi na poziv ove metode**
- **GET, HEAD, PUT, POST, DELETE, OPTIONS i TRACE**
- **Tipičan scenario poziva:**
  - postavi Content-type HTTP odgovora
  - uzimi PrintWriter ka klijentu
  - kroz PrintWriter šalji dinamički kreiran HTML

HttpServlet.init()

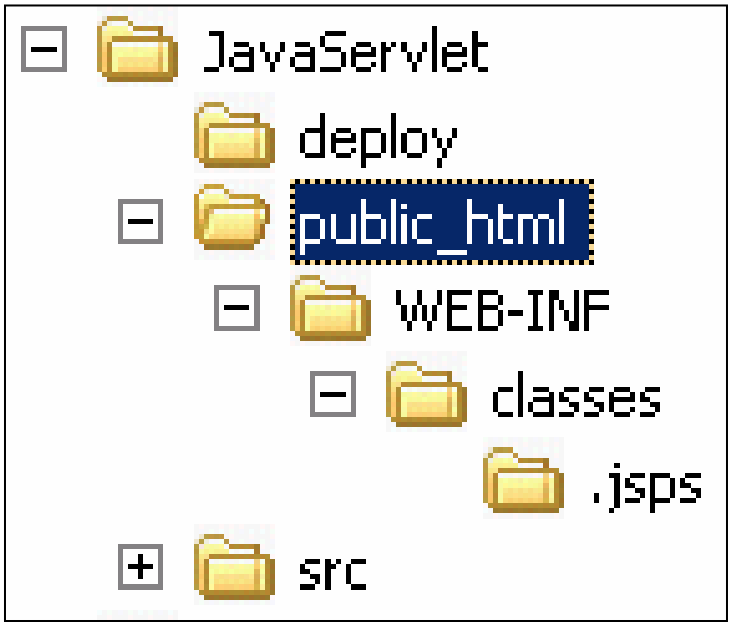
- namijenjena za inicijalizaciju servleta,
  pre njegove prve upotrebe
- poziva se samo jednom
- nema prepreke za postojanje konstruktora u servlet klasi u kome će se odvijati deo inicijalizacije, ali je na raspolaganju i init metoda

HttpServlet.destroy()

- Poziva se prilikom uništavanja servleta
- Namenjena za clean-up zadatke neposredno pre uništenja servleta – oslobađanje resursa koje je servlet zauzimao:
  - otvorene datoteke, konekcija sa bazom podataka
- obično prilikom zaustavljanja Web servera

Servlet Deployment

- **root folder (public_html)**
  - "Polazna tačka" Web aplikacije
  - Sve datoteke i poddirektorijumi su unutar ovog foldera: html, slike…
- **/public_html/WEB-INF/**
  - Sadrži konfiguracione datoteke i kompajlirane klase
  - Nije direktno dostupan putem Web-a
- **/public_html/WEB-INF/classes/**
  - Sve kompajlirane klase (servlet klase i druge klase) se nalaze



Mapiranje servleta

- Servlet klasa mora biti mapirana URI
- Pristup servletu - general pattern (invoker servlet)
  - http://[domain]/[context]/servlet/[ServletClassName]
  - http://localhost:8988/servletintro/servlet/SimpleServlet
- Mapiranje korištenjem konfiguracione datoteke web.xml
  - Servlet se mapira u URL koji je definisao administrator

web.xml

- web.xml se nalazi u “WEB-INF” folder
- Servlet klasa
  - HelloWorld.class
- Kontekst aplikacije:
  - http://localhost:8988/servletintro/
- Invoker class mapiranje
  - http://localhost:8988/servletintro/servlet/HelloWorld
- Mapiranje putem web.xml datoteke
  - http://localhost:8988/servletintro/hello

```
<servlet>
<servlet-name>HelloW</servlet-name>
<servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HelloW</servlet-name>
<url-pattern>hello</url-pattern>
</servlet-mapping>
```