

Web aplikacije (1).

Tehnologije Servlet i JSP.

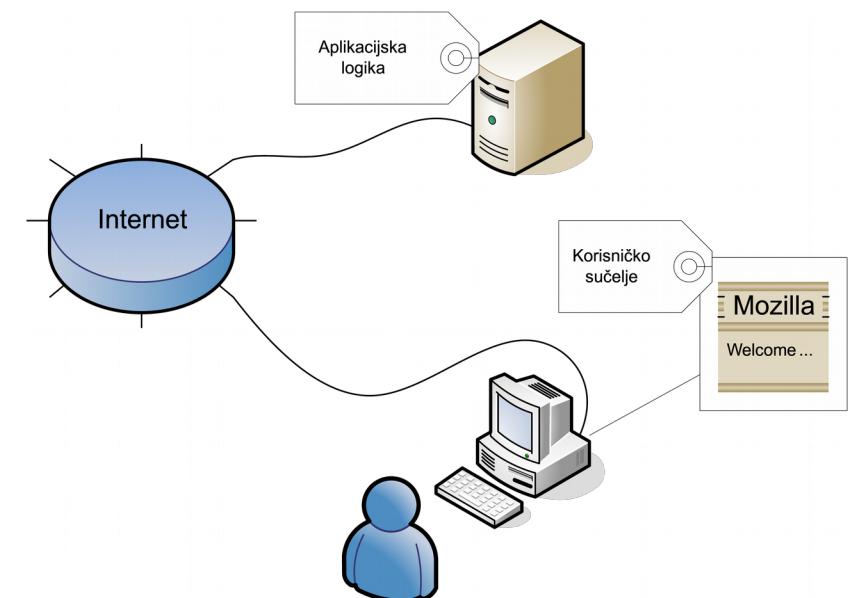
Marko Čupić – AG 2018/2019 – Java tečaj

Web aplikacija

- Prema Servlet/JSP specifikaciji, Web aplikacija je kolekcija resursa dostupnih na određenoj adresi (URL); sastoji se od komponenti:
 - Servleti
 - JSP-ovi
 - Filtri
 - Listeneri
 - Statičke stranice (html i drugo)
 - Resursi
 - JavaBeans objekti
 - Poslužitelj (Servlet container)
 - Java okruženje

Web aplikacija

- Inherentno raspodijeljeni (distribuirani) program:
 - Aplikacijska logika se nalazi na poslužitelju
 - Korisničko sučelje se nalazi na klijentovom računalu (prikazuje ga web-preglednik)



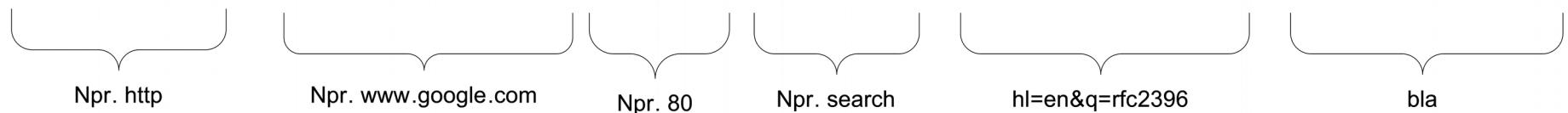
Web aplikacija

- Interakcija: slijed stranica
- Sa stajališta aplikacije:
paradigma *zahtjev – odgovor*
- Komunikacija: TCP/IP,
protokoli HTTP ili HTTPS
- Tipična interakcija:
 - Klijent uspostavlja spoj i šalje zahtjev
 - Poslužitelj obavlja zahtjev, isporučuje odgovor te raskida spoj

Web aplikacija

- URL: Uniform Resource Locator

shema://poslužitelj:port/staza?parametri#fragment



- Korijen se mapira u \$document_root
- Čemu služi fragment?
- Što se radi s “nedozvoljenim” simbolima u parametrima?
- Port 80; heuristički odabir dokumenta za /

Web aplikacija

- Tipično su mješavina statičkih i dinamički generiranih sadržaja
- Začetak razvoja: CGI (Common Gateway Interface) skripte
- Poslužiteljski skriptni jezici
 - Perl, ASP, PHP, JSP, Python, ...
- Servleti

Web aplikacija: skriptni jezici

- Služe za dinamičko nadopunjavanje dijelova HTML stranica (stranice su *predlošci*)
- Kada se zahtjeva takav dokument:
 - Poslužitelj ga ne vraća direktno
 - Konceptualno: pokreće odgovarajući interpreter
 - Interpreter čisti HTML direktno proslijeđuje pregledniku
 - Posebne dijelove interpretira i pregledniku šalje podatke koji nastanu izvođenjem tih dijelova – programski kôd skripte se ne šalje klijentu

Web aplikacija: skriptni jezici: PHP

- Aplikacija za izračun kvadrata broja
- Očekuje parametar “broj”
- Ako ga ne dobije, ispisuje tablicu kvadrata brojeva od 0 do 20

Web aplikacija: skriptni jezici: PHP

```
<html>
<head><title>Kvadrati brojeva</title></head>
<body>
<table cols="2" border="1">
<tr><td><b>Broj</b></td><td><b>Kvadrat broja</b></td></tr>
```

HTML

```
<?php
$br = $_GET['broj'];
$tablica = 1;
if(!isset($br) || $br=="") { $br = 20; } else { $br = (int)$br; $tablica = 0; }
if($tablica==1) {
    for( $n = 0; $n <= $br; $n++ ) {
        echo '<tr><td>' . $n . '</td><td>' . ($n * $n) . '</td></tr>';
    }
} else {
    echo '<tr><td>' . $br . '</td><td>' . ($br * $br) . '</td></tr>';
}
?>
</table>
</body>
```

PHP

HTML

Web aplikacija: skriptni jezici: PHP

```
<html>
<head><title>Kvadrati brojeva</title></head>
<body>
<table cols="2" border="1">
<tr><td><b>Broj</b></td><td><b>Kvadrat broja</b></td></tr>
<tr><td>17</td><td>289</td></tr>
</table>
</body>
```

HTML

~~PHP~~
HTML

HTML

Web aplikacija: skriptni jezici: PHP

Kvadrati brojeva - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.zemris.fer.hr/~marcupic/kvadrati.php?broj=17

Broj	Kvadrat broja
17	289

Done

Kvadrati brojeva - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.zemris.fer.hr/~marcupic/kvadrati.php

Broj	Kvadrat broja
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144
13	169
14	196
15	225
16	256
17	289
18	324
19	361
20	400

Done

Servlet

- Predstavlja enkapsulaciju programskog koda koji je zadužen za obradu određenog zahtjeva
- *Servlet container* omogućava definiranje mapiranja: koja staza odabire koji servlet
- U domaćoj zadaći konceptualno istu stvar odradivali su objekti tipa **IWebWorker**
 - koji worker obrađuje koju stazu definirali smo u **workers.properties**.

Servlet

```
package javax.servlet;
import java.io.IOException;

public interface Servlet {

    public void init(ServletConfig config)
            throws ServletException;
    public ServletConfig getServletConfig();
    public void service(ServletRequest req, ServletResponse
res)
            throws ServletException, IOException;
    public String getServletInfo();
    public void destroy();

}
```

Servlet

- Zahtjev se prema sučelju **Servlet** odraduje u metodi **service** koja prima reference na objekte tipa **ServletRequest** i **ServletResponse**, što smo u zadaći imali objedinjeno u razredu **RequestContext**
- Umjesto da izravne uporabe ovog sučelja, koristit ćemo razred **HttpServlet**, koji ima metodu **service** implementiranu tako da izvođenje delegira dalje metodi koju odabire prema traženoj metodi u zaglavlju zahtjeva

HttpServlet

```
package javax.servlet.http;
public abstract class HttpServlet extends GenericServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected long getLastModified(HttpServletRequest req) {...}
    protected void doHead(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void doPut(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void doOptions(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void doTrace(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    ...
}
```

Servlet

- U razredu **HttpServlet**, metode **doGet**, **doPost**, **doPut** i **doDelete** su implementirane tako da automatski generiraju odgovor sa zaglavljem `405 METHOD_NOT_ALLOWED` (za zahtjeve `HTTP/1.1`) odnosno `400 BAD_REQUEST` (za zahtjeve koji nisu `HTTP/1.1`)
- Stoga pri izradi naših servleta trebamo nadjačati metodu koja nas interesira (i nikako ne pozvati **super.doXXX(...)**).

Komunikacija s klijentom

- U domaćoj zadaći, gledano iz **IWebWorkera** te skripti, vezu prema klijentu modelirao je **RequestContext**
 - Podešavanje mime-tipa odgovora
 - Mogućnost pisanja okteta
 - Mogućnost prosljedivanja obrade nekom drugom entitetu (kroz **IDispatcher**)
- Kod Servlet-specifikacije ovo je razdvojeno u dva objekta: **HttpServletRequest** i **HttpServletResponse**

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Pristup parametrima iz URL-a te kolačićima
 - Pristup sjedničkim podatcima
 - Pristup privremenim atributima
 - Pristup podatcima o korištenoj stazi (URL)
 - Prosljeđivanje na daljnju obradu
 - ...

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Pristup parametrima iz URL-a
`http://...?a=pero&b=jasna`

```
String a = req.getParameter("a");  
String b = req.getParameter("b");
```

```
Map<String, String[]> svi =  
    req.getParameterMap();
```

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Pristup sjedničkim podatcima

```
HttpSession session = req.getSession();
```

```
Object val = session.getAttribute("xxx");  
session.setAttribute("xxx", "yyy");  
session.removeAttribute("zzz");  
session.invalidate();
```

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Pristup privremenim atributima

```
Object val = req.getAttribute("xxx");  
req.setAttribute("xxx", "yyy");  
req.removeAttribute("zzz");
```

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Pristup podatcima o korištenoj stazi (URL)
`http://host/app1/p/t?a=b&c=d`

```
req.getScheme()          ==> "http"  
req.getServerName()     ==> "host"  
req.getServerPort()      ==> 80  
req.getContextPath()    ==> "/app1"
```

HttpServletRequest

- `http://host/app1/p/t?a=b&c=d`

Servlet mapping	<code>/p/t</code>	<code>/p/*</code>
<code>req.getServletPath()</code>	<code>/p/t</code>	<code>/p</code>
<code>req.getPathInfo()</code>	<code>null</code>	<code>/t</code>
<code>req.getQueryString()</code>	<code>a=b&c=d</code>	<code>a=b&c=d</code>

HttpServletRequest

- Razred **HttpServletRequest** enkapsulira zahtjev klijenta
- Nudi:
 - Prosljeđivanje unutar poslužitelja na daljnju obradu (klijent ovoga nije svjestan; MVC)

```
req.getRequestDispatcher(  
    "/WEB-INF/pages/profil.jsp"  
).forward(req, resp);
```

HttpServletResponse

- Razred **HttpServletResponse** enkapsulira odgovor
- Nudi:
 - Pristup **outputStream**-u i **writer**-u
 - Podešavanje kodne stranice i mime-tipa odgovora
 - Slanje redirekcijskog statusa klijentu
 - Pristup zaglavljima odgovora te dodavanje kolačića
 - Parametri odgovora smiju se mijenjati sve dok se ne dohvati izlazni tok – tada se stvara zaglavljje (Content-Type, ...)
 - ...

HttpServletResponse

- Razred **HttpServletResponse** enkapsulira odgovor

```
resp.setStatus (HttpServletResponse.SC_OK) ;  
resp.setContentType ("text/html; charset=UTF-8") ;  
resp.setCharacterEncoding ("UTF-8") ;  
resp.setContentLength (555) ;  
resp.getOutputStream () .write (...) ;  
resp.getWriter () .write ("Štefanija") ;  
-----  
resp.sendRedirect (resp.encodeRedirectURL (  
    req.getContextPath () + "/prikazi"  
)) ;
```

Tehnologija JSP

- Omogućava umetanje dijelova Java koda direktno u HTML dokument
- Kao **smscr skripte** u domaćoj zadaći:
 - Napisani tekst (HTML) se direktno propušta
 - Izvršivi dio (*scriptlet*) se izvodi i dalje se propušta generirani sadržaj
 - Formalno, postoje deklaracijski blokovi, scriptleti, izrazi i direktive

Tehnologija JSP

- *Deklaracijski blokovi* omeđuju se s `<%!` i `%>` i omogućavaju definiranje pomoćnih funkcija koje će se koristiti u stranici
- *Scriptleti* koji se direktno izvode se omeđuju s `<%` i `%>`
- *Izrazi* se omeđuju s `<%=` i `%>`
 - Izraz `<%= expr %>` ekvivalentan je skriptletu:
`<% out.print(expr); %>`
- *Direktive* se omeđuju s `<%@` i `%>`

Tehnologija JSP

- Svaka JSP stranica ima nekoliko eksplisitno definiranih objekata koje može koristiti:
 - **request**: javax.servlet.ServletRequest ili podtip (npr. javax.servlet.HttpServletRequest)
 - **response**: javax.servlet.HttpServletResponse ili podtip (npr. javax.servlet.HttpServletResponse)
 - **pageContext**: javax.servlet.jsp.PageContext
 - **session**: javax.servlet.http.HttpSession
 - **application**: javax.servlet.ServletContext (getServletConfig().getContext())
 - **out**: javax.servlet.jsp.JspWriter, omata response.getWriter()
 - **config**: javax.servlet.ServletConfig
 - **page**: java.lang.Object (za JSP skriptu koja koristi jezik Javu je "page" sinonim za "this")

Web aplikacija: skriptni jezici: JSP

```
<%@ page language="java" session="false" contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Kvadrati brojeva</title></head>
<body>
<table cols="2" border="1">
<tr><td><b>Broj</b></td><td><b>Kvadrat broja</b></td></tr>
<%
String sBroj = (String) request.getParameter("broj");
Integer br = null;
if(sBroj!=null) {
    try {
        br = Integer.valueOf(sBroj);
    } catch(Exception e) {}
}
boolean tablica= true;
if(br==null) { br = Integer.valueOf(20); } else { tablica = false; }
if(tablica) {
    for( int n = 0; n <= br.intValue(); n++ ) {
        out.print("<tr><td>" + n + "</td><td>" + (n*n) + "</td></tr>");
    }
} else {
    out.print("<tr><td>" + br + "</td><td>" + (br.intValue() * br.intValue()) + "</td></tr>");
}
%>
</table>
</body>
```

HTML

JSP

HTML

Čest obrazac obrade

- Korisnikov zahtjev pokrene neki servlet
 - Servlet izračuna rezultat, pohrani ga u privremene atributе i proslijedi obradu JSP-u
 - JSP dohvati pripremljene rezultate i izgenerira HTML dokument koji ih prikazuje
- demonstrirali smo ga u domaćoj zadaći

Čest obrazac obrade

- To je MVC obrazac primijenjen na web
- Aplikacijska logika se izvodi kroz servlete:
 - Jednostavni izračuni
 - Raspodijeljeno izračunavanje
 - Pristup bazi podataka
 - Rezultat obrade se pohranjuje u `HttpServletRequest` mapu atributa (priv. parametri)
- Korisničko sučelje izvodi se kroz JSP
 - Dohvaća preko mape atributa rezultat obrade i prikazuje ga

Spremnici podataka

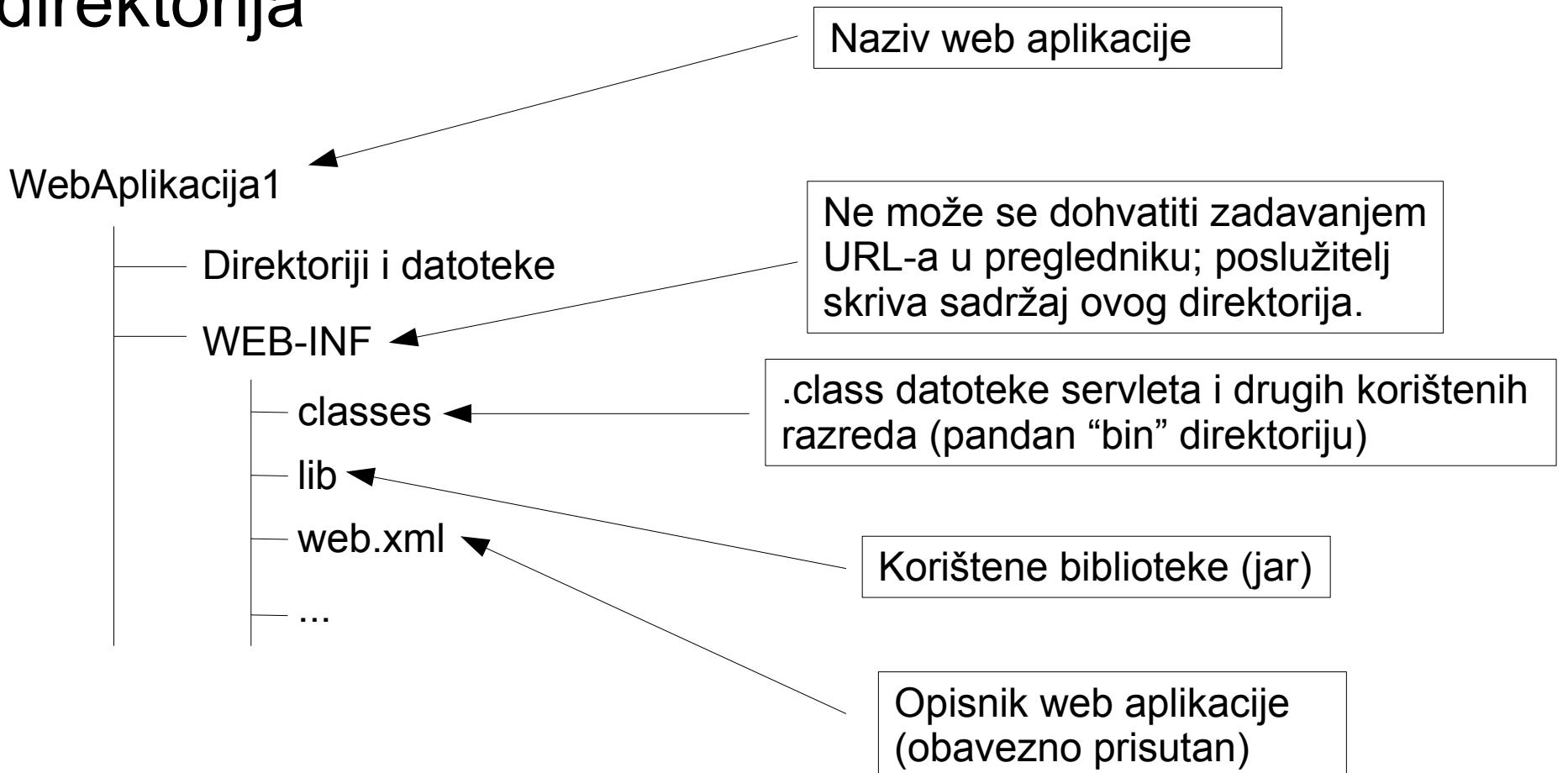
- Specifikacija Servleta razlikuje 4 mesta za dohvat / pohranu podataka
 - Globalni parametri:
`req.getServletContext().getAttribute` /
`req.getServletContext().setAttribute`
 - Sjednički parametri (primjerak po korisniku):
`req.getSession().getAttribute` /
`req.getSession().setAttribute`
 - Privremeni parametri (primjerak po zahtjevu):
`req.getAttribute` / `req.setAttribute`
 - Parametri zahtjeva: `req.getParameter`

Veza Servlet - JSP

- JSP stranice poslužitelj transparentno prevodi u Servlete i prevodi u .class datoteke
- JSP-ovi se izvršavaju jednakom brzinom kao i servleti
 - “skriptni” jezik ali bez velike cijene po performanse!

Web aplikacija

- Web aplikacija ima propisanu strukturu direktorija



Podrška za web-aplikacije

- Trebamo implementaciju poslužitelja koji razumije ovaku strukturu web-aplikacije
 - Automatski radi parsiranje parametara
 - Obavlja praćenje sjednica (šalje Cookieje, čuva sjedničke mape)
 - Omogućava definiranje “koja staza poziva koji objekt za obradu”
 - Zna kako izvoditi “skripte”

Podrška za web-aplikacije

- Takvi poslužitelji nazivaju se *servlet containers*
- Omogućavaju izvođenje više aplikacija istovremeno, pri čemu se tražena aplikacija prepoznaće po prvoj “razini direktorija” u URL-u; npr.:

`http://localhost:8080/app1/xxx`

`http://localhost:8080/app2/xxx`

Podrška za web-aplikacije

- Pojedina web-aplikacija u tom se smislu naziva *servlet context*
- *Servlet containeri* koji nude podršku za izvođenje još složenijih aplikacija (Java EE) nazivaju se *Application Serveri*.
- Mi ćemo koristiti **Apache Tomcat** koji je *Servlet Container* (alternativa: Eclipse Jetty, ...)

Upogonjavanje web aplikacije

- Upogonjavanje (engl. Deployment)
- Moguće na više načina
 - Način 1.

Raspakirana web aplikacija je negdje na disku i napravljen je deployment descriptor koji pokazuje na nju → taj se opisnik može zadati tomcatu preko aplikacije /manager

Upogonjavanje web aplikacije

- Upogonjavanje (engl. Deployment)
- Moguće na više načina
 - Način 2.

Napravi se arhiva web aplikacije (WAR)
→ ta se arhiva može uploadati na tomcat preko aplikacije /manager

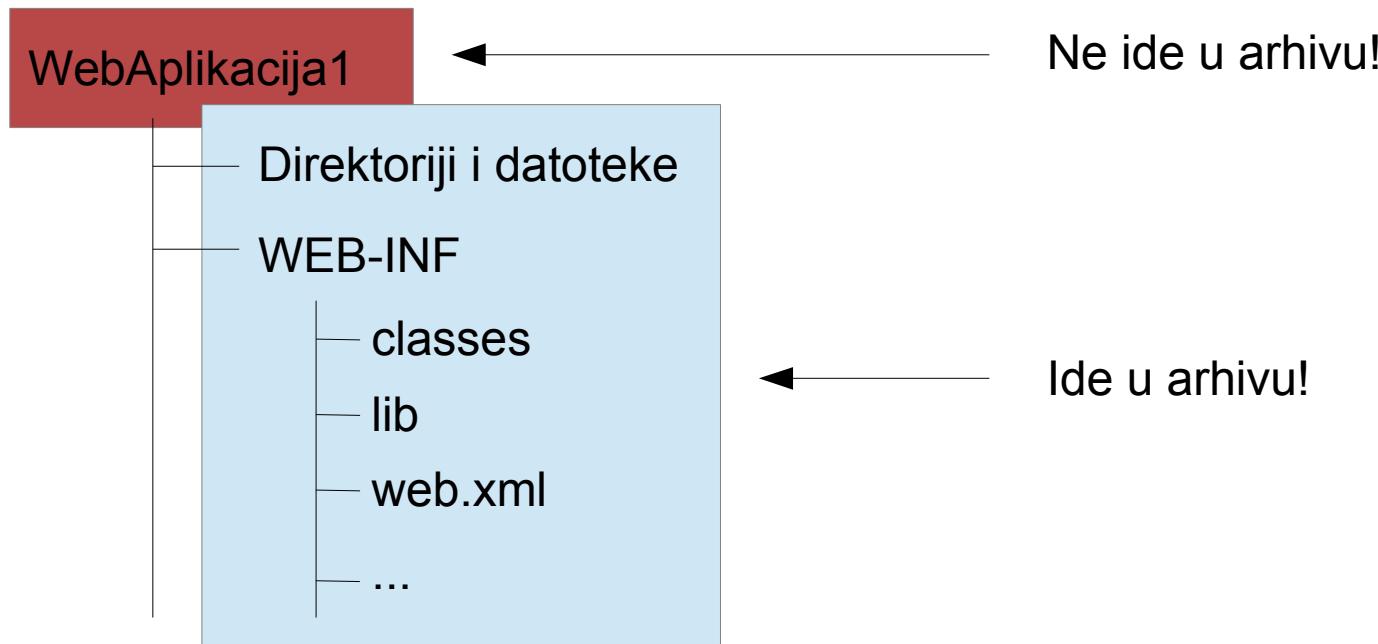
Upogonjavanje web aplikacije

- Upogonjavanje (engl. Deployment)
- Moguće na više načina
 - Način 3.

Napravi se arhiva web aplikacije (WAR)
→ ta se arhiva može fizički iskopirati u tomcatov webapps direktorij; ako se arhiva zove abc.war, tomcat će u webapps napraviti poddirektorij abc i unutra raspakirati arhivu; /abc **postaje naziv konteksta** preko kojeg je aplikacija dohvataljiva preko URL-a

Izrada arhive WAR

- WAR je najobičnija ZIP arhiva i može se izraditi bilo čime što stvara ZIP arhive
- U arhivu se ne pakira vršni direktorij!



Filter

- Zadaća filtera je omogućiti izvođenje koda prije i/ili nakon što servlet/JSP obave svoj posao
- Omogućavaju transparentne manipulacije nad zahtjevom / odgovorom
 - Postavljanje kodne stranice
 - Komprimiranje odgovora
 - ...
- Objekt je potrebno označiti
(u web.xml ili anotacija)

Filter

```
public interface Filter {  
  
    public void init(FilterConfig filterConfig)  
        throws ServletException;  
  
    public void doFilter(  
        ServletRequest request, ServletResponse  
response,  
        FilterChain chain)  
        throws IOException, ServletException;  
  
    public void destroy();  
}
```

Listeneri

- Poslužitelj omogućava dojavu različitih vrsta informacija posebnim objektima web aplikacije
- Podržani su:
 - javax.servlet.ServletContextListener
 - javax.servlet.ServletContextAttributeListener
 - javax.servlet.ServletRequestListener
 - javax.servlet.ServletRequestAttributeListener
- Objekt je potrebno označiti
(`web.xml` ili anotacija)

Anotacije

- Od specifikacije Servlet 3.0 kao nadopuna datoteke `web.xml` moguće je koristiti anotacije direktno u izvornom kodu
 - `@WebServlet(name="id", urlPatterns={"/a", "/b"})`
 - `@WebFilter(filterName="id", urlPatterns={"/a"})`
 - `@WebListener`
- ili kraće:
- `@WebServlet("/a")`
 - `@WebFilter("/a")`