Web aplikacije (1).

Tehnologije Servlet i JSP.

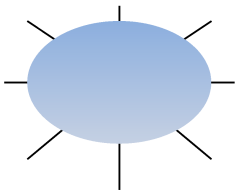
Marko Čupić – AG 2013/2014 – Java tečaj

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
  - Java Appleti
  - 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)

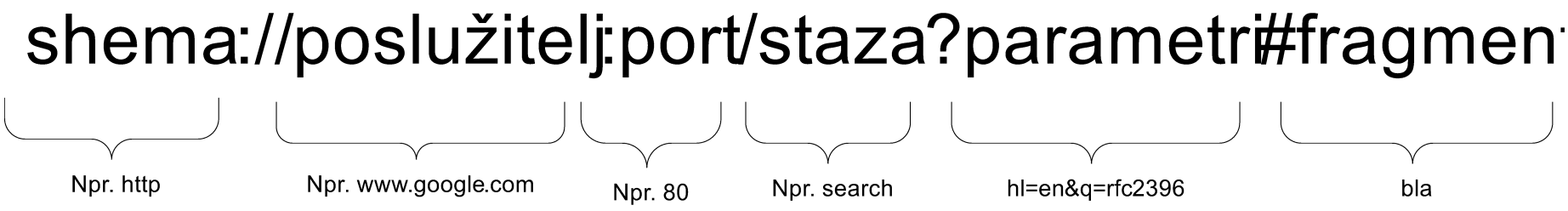


Web aplikacija

- 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



- 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 prosljeđuje 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

```
<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>
```

PHP

```
<?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>';
 }
?>
```

HTML

```
</table>
</body>
```

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~~

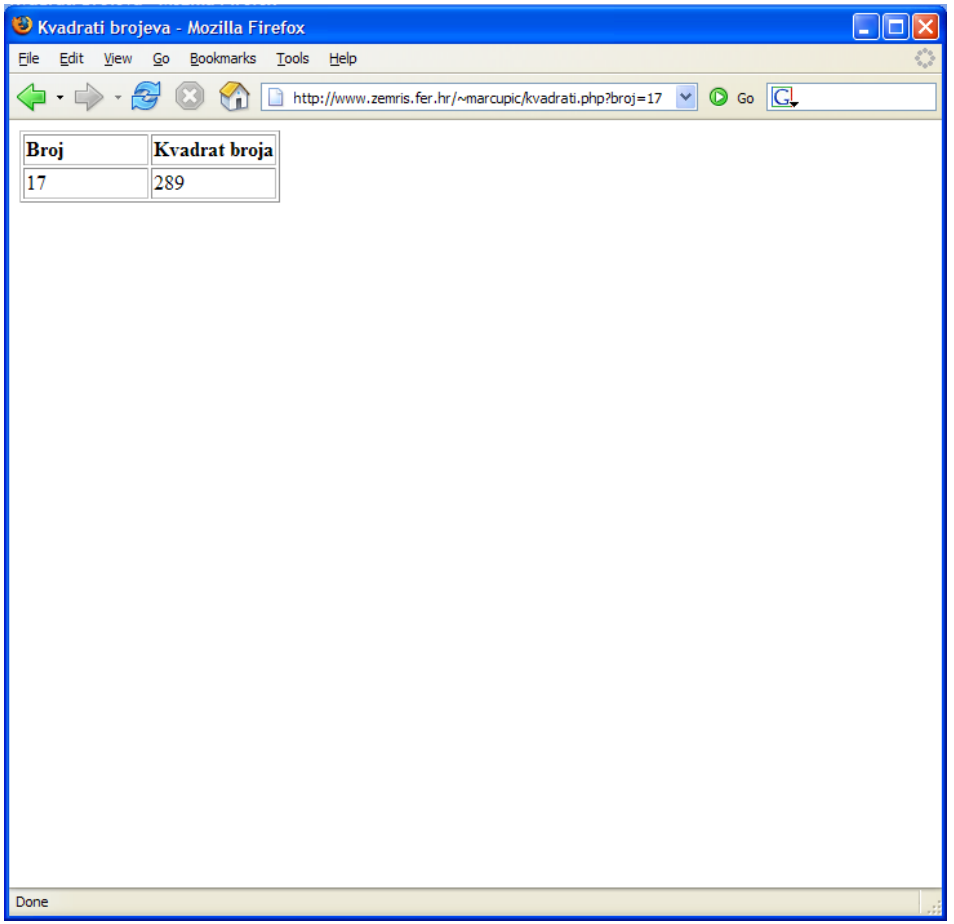
HTML

```
<tr><td>17</td><td>289</td></tr>
```

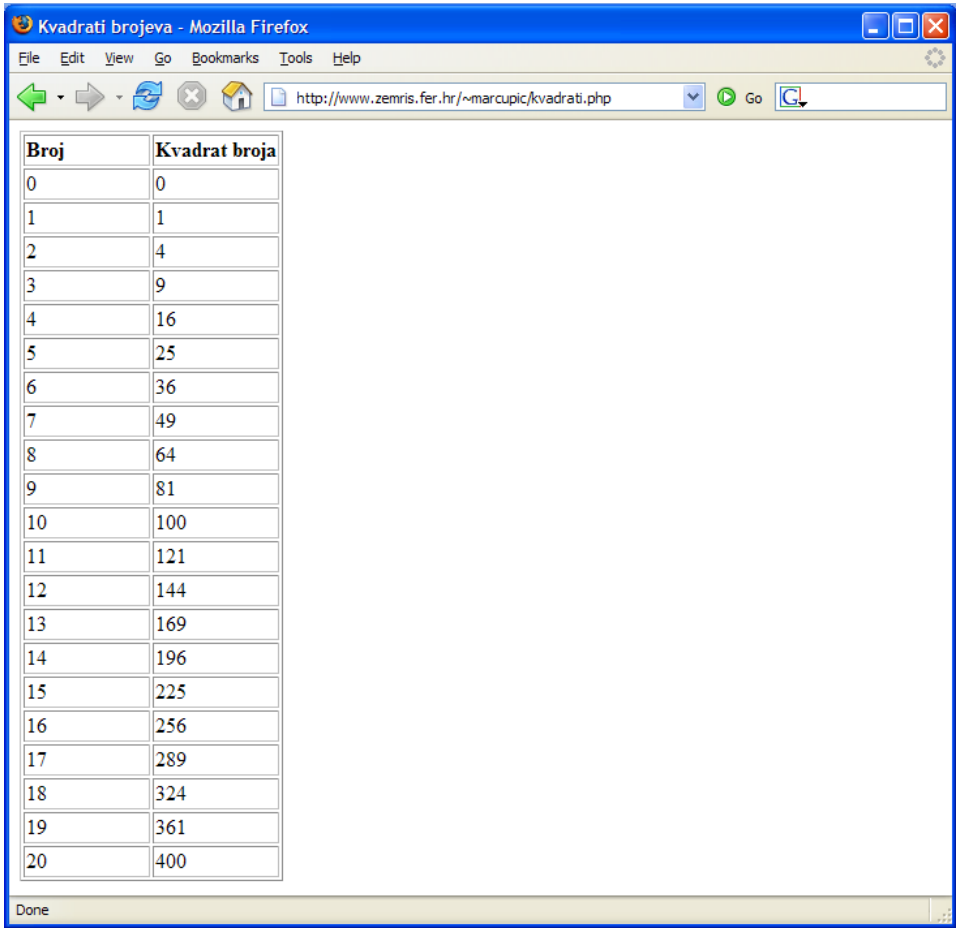
```
</table>
</body>
```

HTML

Web aplikacija: skriptni jezici: PHP



Broj	Kvadrat broja
17	289



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

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 odrađivali 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();

}
```

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 {…}
    …
}
```

HttpServletRequest

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

HttpServletResponse

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

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 eksplicitno definiranih objekata koje može koristiti:
  - request: javax.servlet.ServletRequest ili podtip (npr. javax.servlet.HttpServletRequest)
  - response: javax.servlet.ServletResponse 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

HTML

```
<%@ 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>
```

JSP

```
<%
 String sBroj = (String)request.getParameter("broj");
 Integer br = null;
 if(sBroj!=null) {
   try {
     br = Integer.valueOf(sBroj);
   } catch(Exception ex) {}
 }
 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>");
 }
%>
```

HTML

```
</table>
</body>
```

Čest obrazac obrade

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

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
- Korisničko sučelje izvodi se kroz JSP
  - Dohvaća preko mape atributa rezultat obrade i prikazuje ga

Spremnici podataka

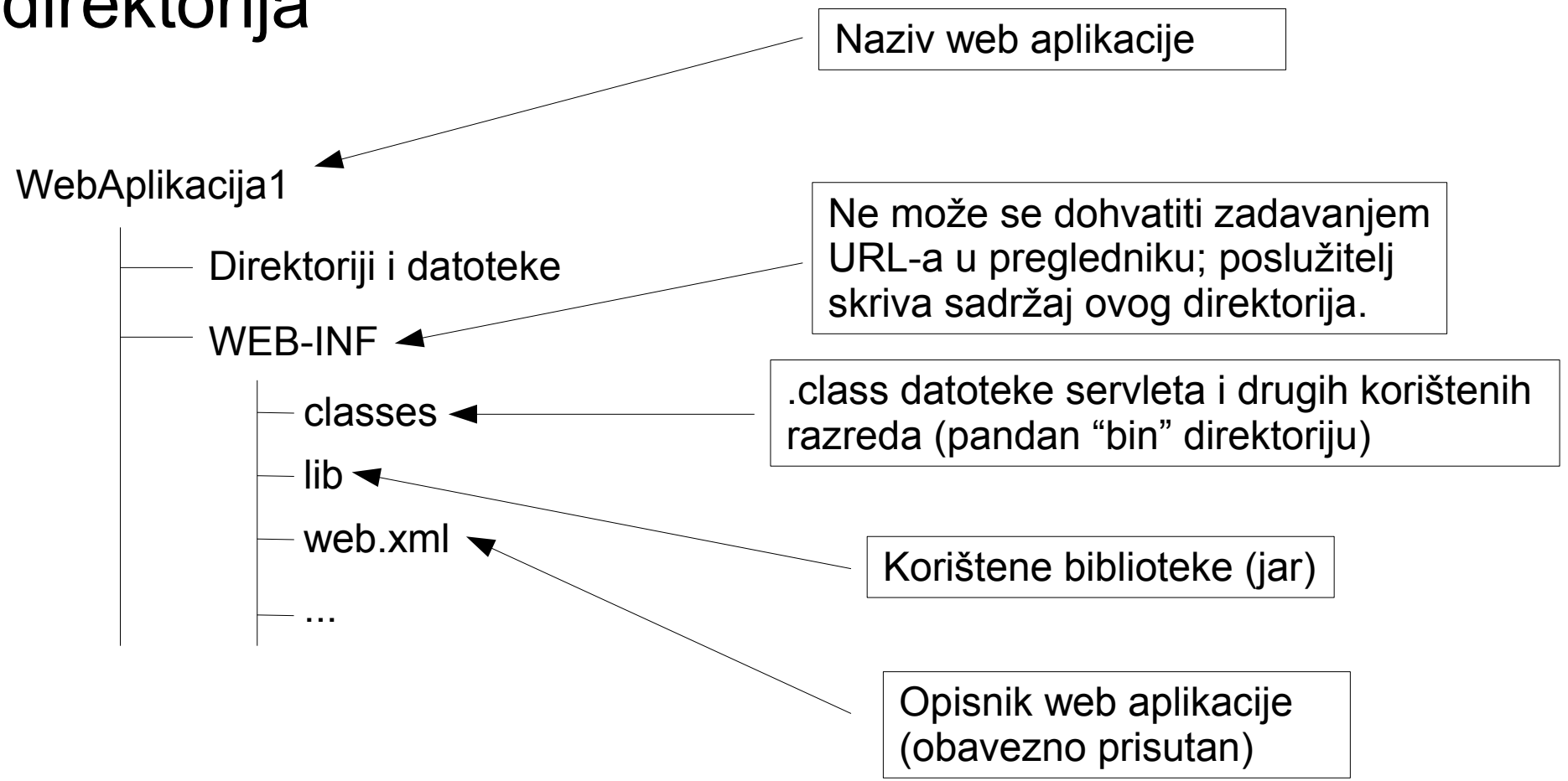
- Specifikacija Servleta razlikuje 4 mjesta za dohvat / pohranu podataka
  - Globalni parametri: **`req.getServletContext().getAttribute / req.getServletContext().setAttribute`**
  - Sjednički parametri (primjerak po korisniku): **`req.getSession().getAttribute / req.getSession().setAttribute`**
  - 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



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

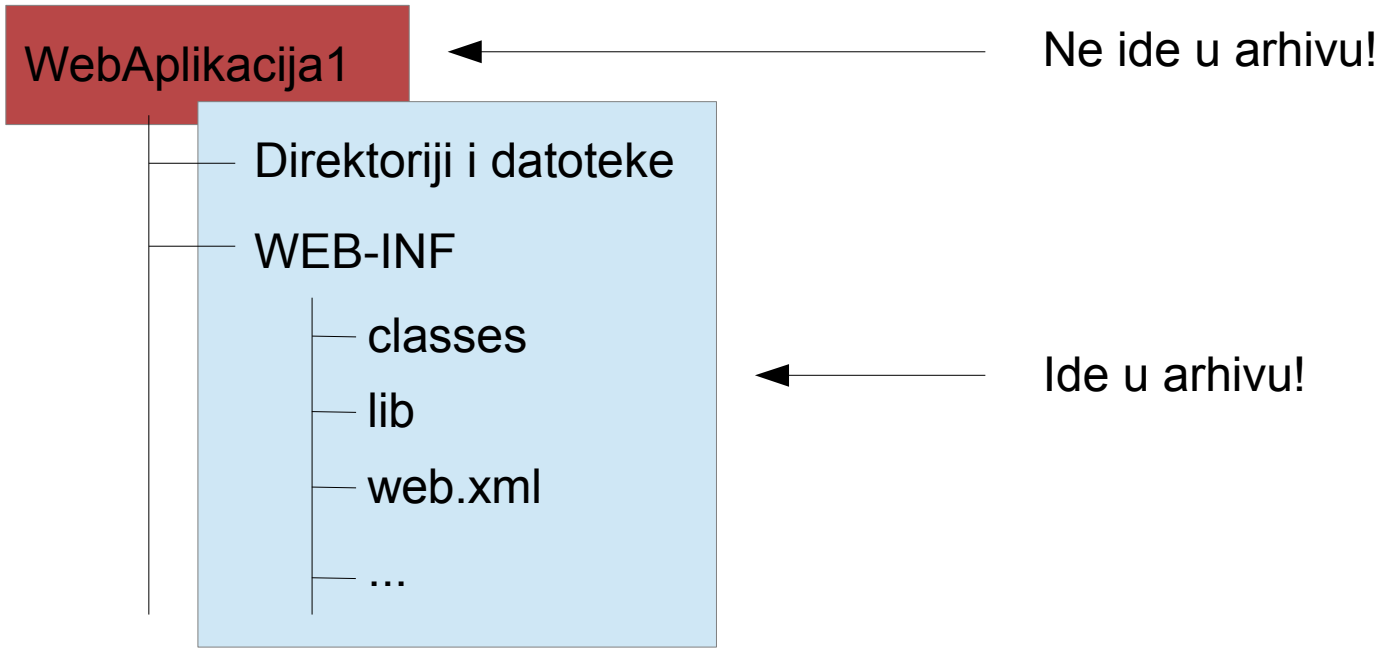
Upogonjavanje web aplikacije

- 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 dohvatljiva 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 (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")`**