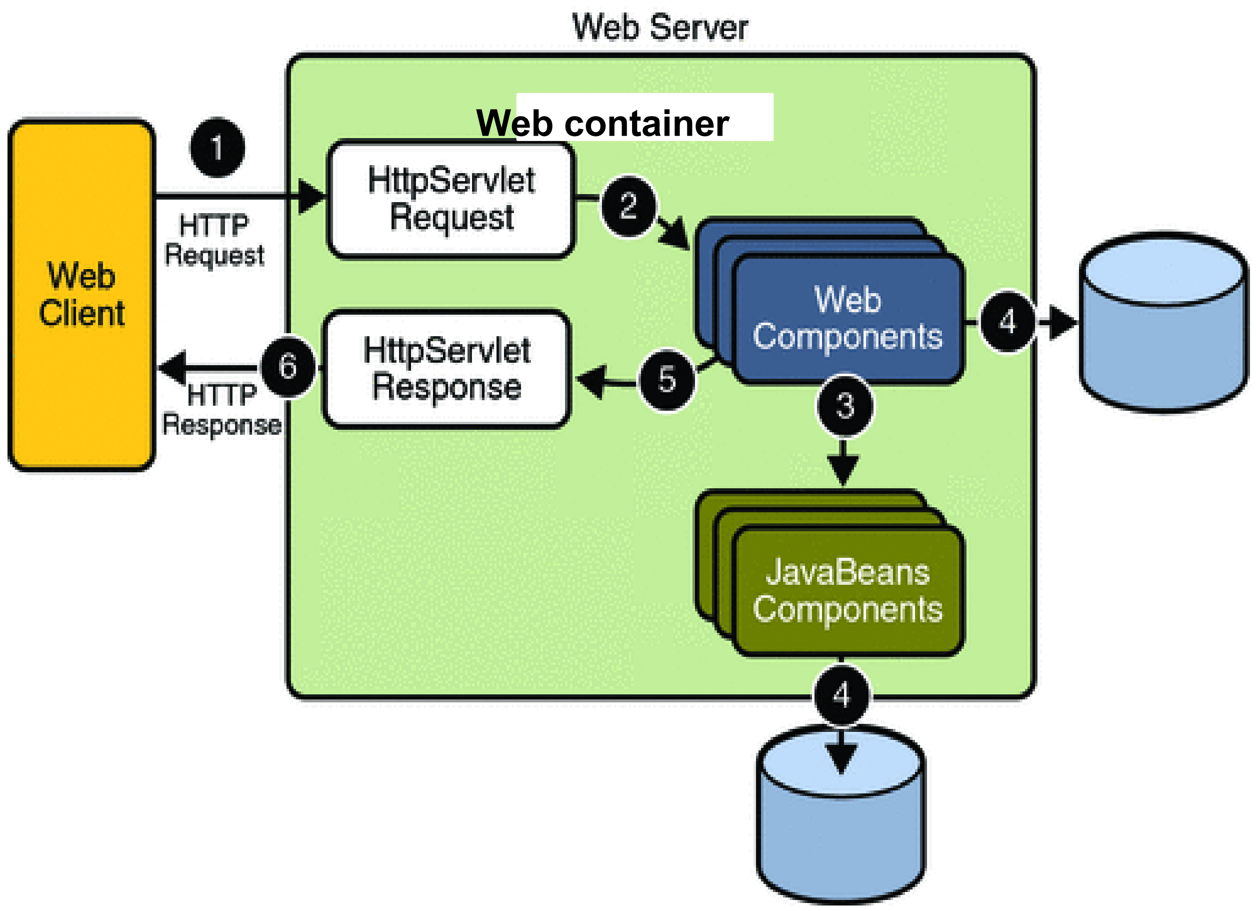
Мрежово програмиране

JAVA Web Components

доц. д-р Йордан Денев
denev@fmi.uni-sofia.bg


Web Server
Web container
Web Client
1
HTTP Request
HttpServlet Request
2
Web Components
4
3
JavaBeans Components
4
5
HttpServlet Response
6
HTTP Response

- **Servlets** are Java programming language classes that dynamically process requests and construct responses.
- **JSP pages** are text-based documents that execute as servlets but allow a more natural approach to creating static content.

- **Web container**– includes
  - a basic web server;
  - a request/response translator ;
  - a runtime environment for the web components;
  - supports specific objects and methods;

Servlets

❑The life cycle

1. If an instance of the servlet does not exist, the Web container
   - Loads the servlet class.
   - Creates an instance of the servlet class.
   - Initializes the servlet instance by calling the `init` method.
2. When the request is received it invokes the `service` method.
3. `service` calls do*Method* according the *Method* specified in the HTTP request and passes to it a request and response object.
4. When the servlet is removed or reloaded invokes `destroy` method.

❑ The Servlet Structure

The servlet is an Java class, which extends the base class `HttpServlet`.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class <ServletName> extends HttpServlet {
 //servlet methods
}
```

❑Initializing a Servlet

- After the Web container loads and instantiates the servlet class and before it delivers requests from clients, the Web container initializes the servlet.
- You can customize this process to allow the servlet to read persistent configuration data, initialize resources, and perform any other one-time activities by overriding the init method of the Servlet interface.
- A servlet that cannot complete its initialization process should throw UnavailableException.

- An example

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;
  public void init() throws ServletException {
   bookDB = OpenDB(“Book DB”);
      if (bookDB == null)
      throw new UnavailableException("Couldn't get
      database.");
  }
}
```

❑The `service` and do*Method* Methods

- The service provided by a servlet is implemented in the `service` method of a GenericServlet. It invokes the do*Method* methods (where *Method* can take the value Get, Delete, Options, Post, Put, Trace).
- An example:

```
public void doGet (HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {

            …………………………………………………………………… .

}
```

❑Getting Information from Requests

Parameters, which are typically used to conveyinformation between clients and servlets:

```
String bookId = request.getParameter("Add");
if (bookId != null) {
     ……………………………………………………………………………
}
```

❑Constructing Responses

- Retrieve an output stream to use to send data to the client. To send character data, use the `PrintWriter` object returned by the response's `getWriter` method.
- Indicate the content type (for example, text/html) being returned by the response with the `setContentType`(String) method.
- Indicate whether to buffer output with the `setBufferSize(int)` method.

- An example:

```
public class BookDetailsServlet extends HttpServlet {
public void doGet (HttpServletRequest request,
                 HttpServletResponse response)
throws ServletException, IOException  {
   // set headers before accessing the Writer
   response.setContentType("text/html");
   response.setBufferSize(8192);
   PrintWriter out = response.getWriter();
   // then write the response
   out.println("<html>" +"<head><title>+
           "TitleBookDescription"+</title></head>");
```

……

JSP

JSP page is a text document that contains two types of text: static data, which can be expressed in any text-based format (such as HTML, SVG, WML, and XML), and JSP elements, which construct dynamic content.

❑JSP elements

JSP Element	Syntax	Interpretation
JSP Expression	<%= expression %>	Expression is evaluated and placed in output.
JSP Scriptlet	<% code %>	Code is inserted in service method
SP Comment	<%-- comment --%>	Comment; ignored when JSP page is translated into servlet.
SP include Directive	<%@ include file="url" %>	A file on the local system to be included when the JSP page is translated into a servlet.

- Predefined objects
  - `request`, the HttpServletRequest;
  - `response`, the HttpServletResponse;
  - `session`, the HttpSession associated with the request (if any);
  - `out,` the PrintWriter (a buffered version of type JspWriter) used to send output to the client.
- An example:

```
Your hostname: <%= request.getRemoteHost() %>
Your name: <%= request.getParameter(“Name”) %>
```

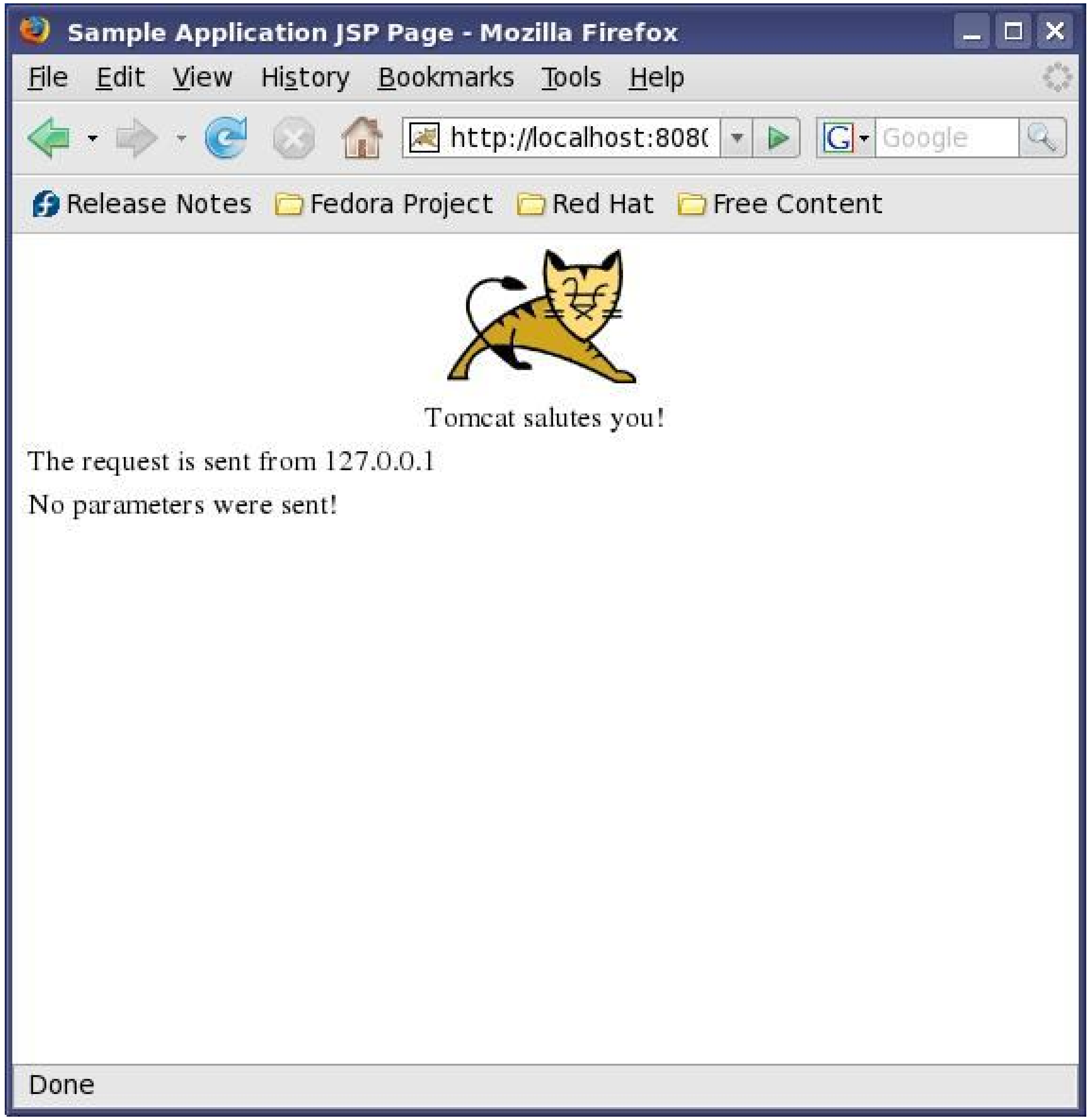
❑ Access to CGI variables

```
"AUTH_TYPE", request.getAuthType() ,
"CONTENT_LENGTH",
    String.valueOf(request.getContentLength())
   "CONTENT_TYPE", request.getContentType()
"DOCUMENT_ROOT", getServletContext().getRealPath("/")
   "PATH_INFO", request.getPathInfo()
"PATH_TRANSLATED", request.getPathTranslated()
   "QUERY_STRING", request.getQueryString()
"REMOTE_ADDR", request.getRemoteAddr()
"REMOTE_HOST", request.getRemoteHost()
"REMOTE_USER", request.getRemoteUser()
"REQUEST_METHOD", request.getMethod()
"SCRIPT_NAME", request.getServletPath()
"SERVER_NAME", request.getServerName()
"SERVER_PORT", String.valueOf(request.getServerPort())
   "SERVER_PROTOCOL", request.getProtocol()
   "SERVER_SOFTWARE", getServletContext().getServerInfo()
```

An example:

```
<html>
<head>
<title>Sample Application JSP Page</title>
</head>
<body bgcolor=white>

<CENTER>
<img src="images/tomcat.gif">
<%= new String("<BR>Tomcat salutes you!<BR>") %>
</CENTER>
<%= "The request is sent from " +request.getRemoteHost() %>
<%
    String queryData = request.getQueryString();
    if (queryData == null)
        out.println("<BR> No parameters were sent!");
    else
        out.println("<BR>Parameters are:" + queryData);
%>
</body>
</html>
```


Sample Application JSP Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:808(
Google
Release Notes
Fedora Project
Red Hat
Free Content
Done

Tomcat salutes you!

The request is sent from 127.0.0.1

No parameters were sent!