Java Servlets

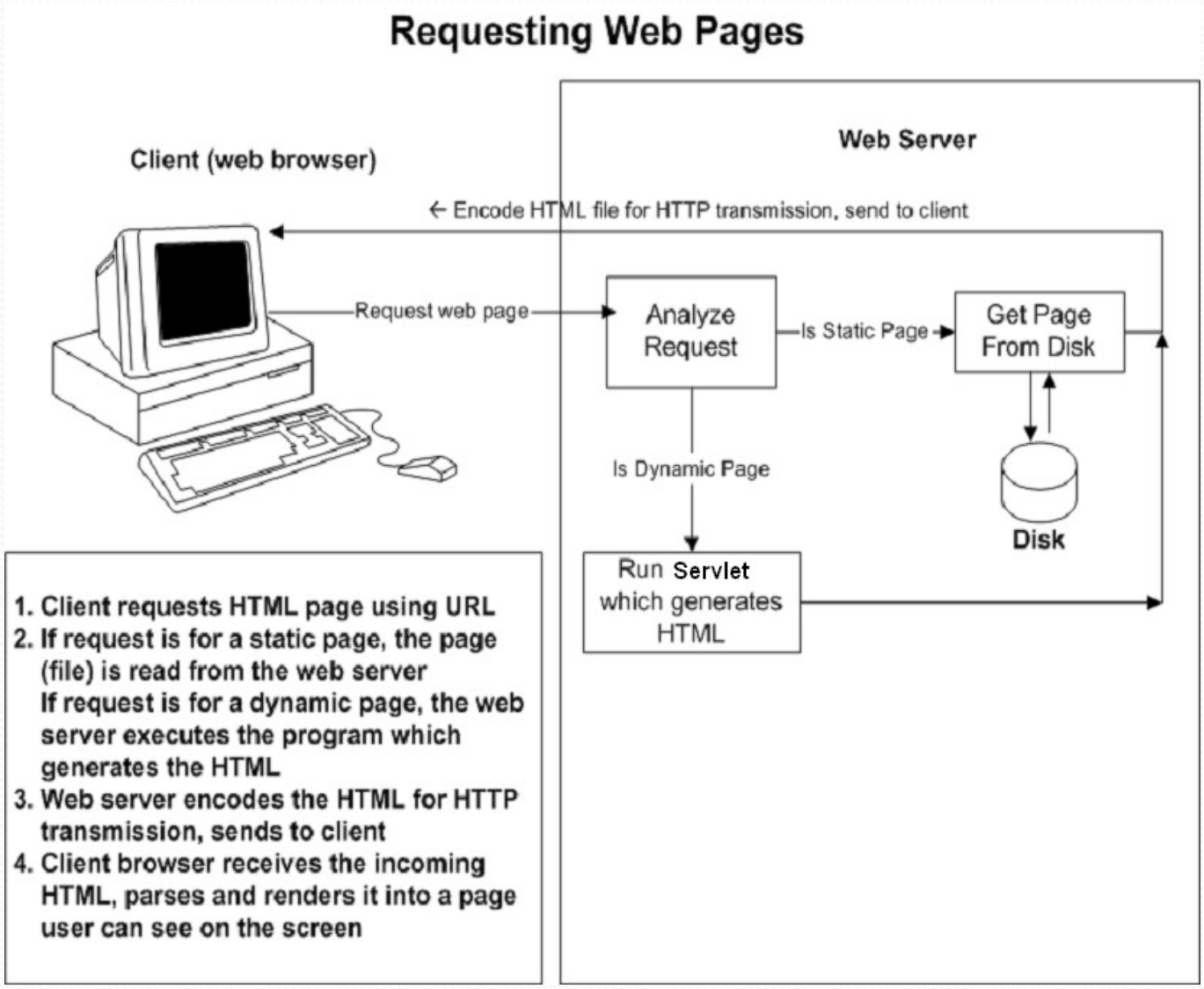
What is a Servlet?

- Servlets are server side Java classes that add functionality to a web server.
- Servlets support the standard request/response protocol supported by web servers.
- In this request/response protocol, a client sends a request message to a server and the server responds by sending back a reply message
  - Reply is usually as HTML
  - Could be image, audio stream, video stream, etc.

What is a Servlet? -2

- Servlets generally implement either the Servlet interface or extend the HttpServlet abstract class
- Most programmers extend HttpServlet class
- Servlets run in a servlet container (Tomcat is a popular one)
- Servlets are platform independent
  - Your servlets will work on an Apache web server running on Linux or a IIS web server running on Windows OS
- Servlets are generally invoked from a hyperlink or from the action attribute of an HTML form

Web/Servlet Architecture



Container

- It provides runtime environment for JavaEE (j2ee) applications.
- It performs many operations that are given below:
  - Life Cycle Management
  - Multithreaded support
  - Object Pooling
  - Security etc.

Server

- It is a running program or software that provides services.
- There are two types of servers:
  - Web Server
  - Application Server

Servers

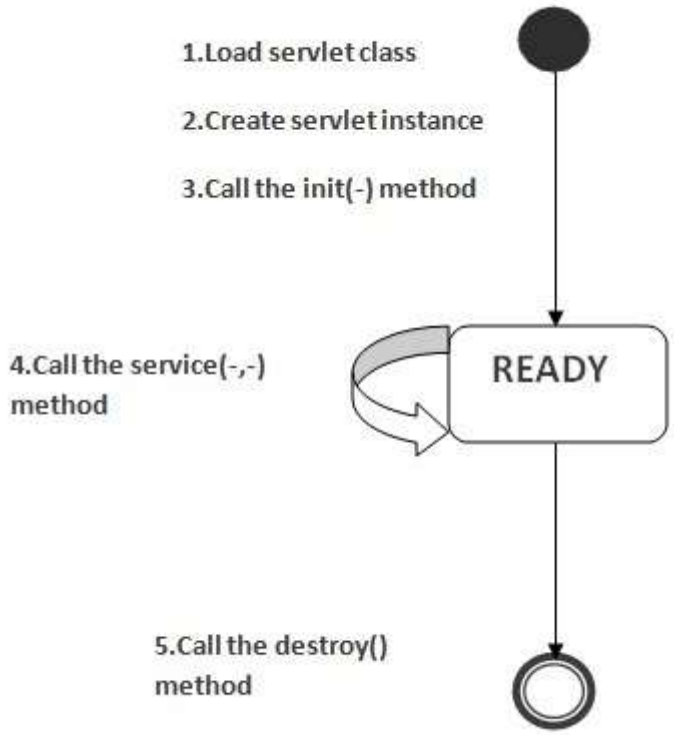
- Web Server
  - Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
    - Example of Web Servers are: Apache Tomcat and Resin.
- Application Server
  - Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc.
    - Example of Application Servers are:
      - JBoss Open-source server from JBoss community.
      - Glassfish provided by Sun Microsystem. Now acquired by Oracle.
      - Weblogic provided by Oracle. It more secured.
      - Websphere provided by IBM.

Servlets

- The web server communicates with servlets via a simple interface: javax.servlet.Servlet. This interface defines three methods.
  - init() -- invoked on servlet startup, put init code here
  - service() -- each client request invokes this method
  - destroy() -- put cleanup code here
- The servlet container controls the lifecycle of the servlet objects
  - initializes, invokes, and destroys each servlet instance.
- You can write a servlet by extending the Servlet class and implement these three methods

Servlet Life Cycle

- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.



Servlet Example

```
import java.io.*;
import javax.servlet.*;
public class MyServlet implements Servlet
{
  private ServletConfig config;

  public void init (ServletConfig config)
    throws ServletException
  {
    this.config = config;
  }
  public void destroy() {} // do nothing
```

Servlet Example…

```
  public ServletConfig getServletConfig() //optional
  {                                       //method
    return(config);
  }
  public void service(ServletRequest req,
                      ServletResponse res)
                  throws ServletException, IOException
  {
    res.setContentType( "text/html" );
    PrintWriter out = res.getWriter();
    out.println( "<html>My Servlet!!!</html>" );
    out.close();
  }
}
```

Other Servlet Methods

- getRealPath() -- method translates a relative or virtual path to a new path relative to the server's HTML documentation root location.
- getServlet() -- Returns a Servlet object of a given name.
- log() -- Writes information to a servlet log file. The log file name and format are server specific.

Note: When extending the HttpServlet class, you access these methods via the ServletContext. For example

realPath = getServletContext().getRealPath(virtualPath);

HTTP GET vs. POST

- HTTP GET sends the contents of the form fields as extra data appended to the URL specified in the form's ACTION attribute.
- For example, http://www.noaa.gov/MyServlet?search=football
- Note that you could bookmark this URL to quickly reload it in a browser.
- HTML forms that use do HTTP GETs should not contain too much data (1000 chars?) or they might overflow the web server

HTTP GET vs. POST

- HTTP POST sends the contents of the form fields as a separate data stream to the web server
- HTTP POSTs cannot be bookmarked by the browser
- HTML forms that use do HTTP POSTs can send large amounts of data to the web server without any problems
- The data that is sent is not visible in the URL that appears in the browser
  - For example, if you have a password field, you would want to do a HTTP POST so the password doesn't appear in the URL

HttpServlet

- Instead of implementing the Servlet interface, most servlets extend the HttpServlet class because it has http support built in, making it easier to access the request and response objects.
- When you extend HttpServlet, you generally write code for one or both of the following methods:
  - doGet() -- corresponds to HTTP GET
  - doPost() -- corresponds to HTTP POST

HttpServlet

- The doGet() and doPost() methods take two parameters:
  - HttpServletRequest -- contains all the information about the request made by a web page, including any form data
  - HttpServletResponse -- object you can use to send data (usually html) back to the calling web page

HttpServlet

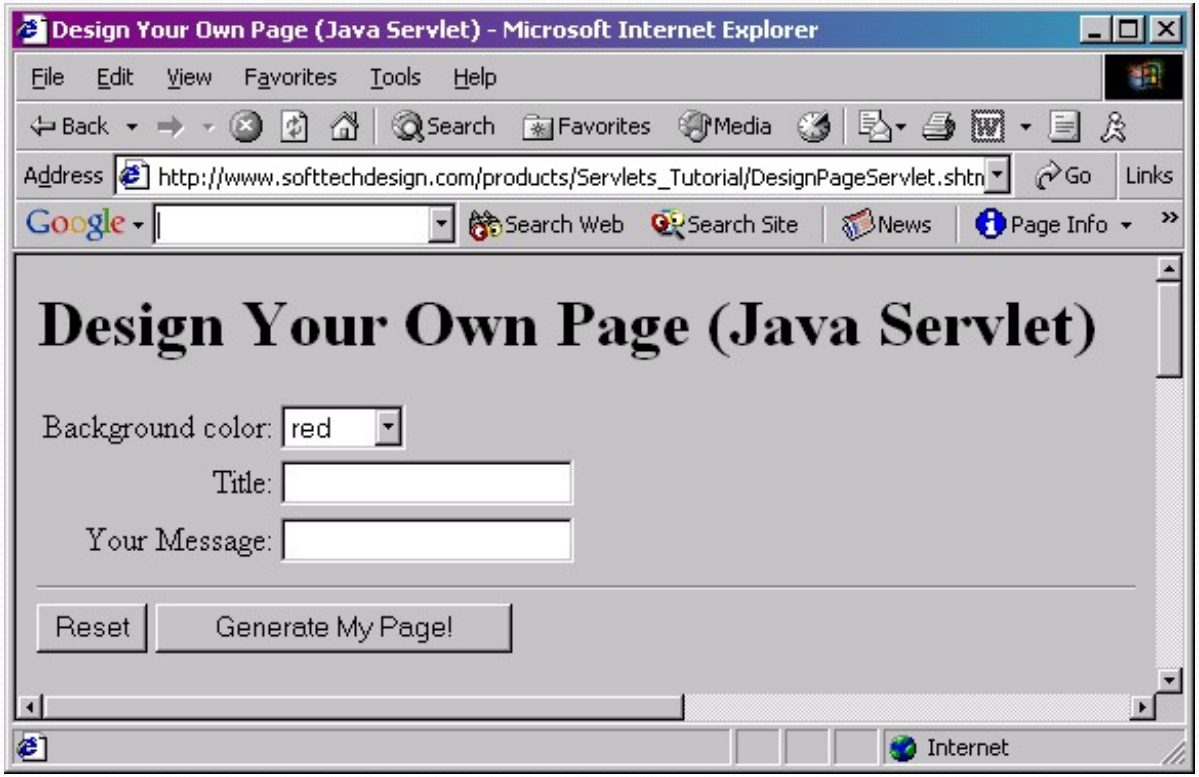
- doGet() and doPost() methods generally do the following:
  - Read the request data (html form data)
  - Process the request
  - Set response object headers
  - Write the response data

HttpServlet Style Suggestions

- Keep your Servlet classes as lean as possible
- Put your "business" or "application" logic in a separate class and have your servlet's doPost() or doGet() methods invoke that code
- Your servlet can directly return HTML, but in most real world (and complex) web apps, your servlets would not return HTML via out.println() statements.
  - Instead, your servlets would return JSP pages with all the HTML formatting already in it.

HttpServlet Example

- This web page prompts the user for the background color, title, and message that will be passed to a servlet. The servlet will read this data and create a new web page using the supplied information.



HttpServlet Example

- In this example, the servlet will read the information typed into the web form and generate a new html document based on the user supplied information
- To keep things simple, it will build the html string in the Java code and return it.
- The form action for this web page might look like:

```
<form action="http://www.noaa.gov/DesignPageServlet"
  method="post">
```

HttpServlet Example

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

public class DesignPageServlet extends HttpServlet
{
  //Initialization
  public void init(ServletConfig config)
                throws ServletException
  {
    super.init(config);
  }
```

HttpServlet Example

```
public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
      throws ServletException, IOException
  {
    String sBKCOLOR = request.getParameter("BKCOLOR");
    String sTITLE = request.getParameter("TITLE");
    String sMESSAGE = request.getParameter("MESSAGE");
    response.setContentType("text/html");
    PrintWriter out = new PrintWriter
                      (response.getOutputStream());
    out.println("<html><head><title>");
    out.println(sTITLE + "</title></head>");
    out.println("<body bgColor=" + sBKCOLOR + ">");
    out.println("<h1>" + sTITLE + "</h1>");
    out.println("<br><br>" + sMESSAGE);
    out.println("</body></html>");
    out.close();
  }
}
```

Servlet Development Summary

- Step 1 – Create your html web form
- Step 2 – Create your Java Servlet
- Step 3 – Create app directory under [tomcat]/webapps
  - Tip--copy webdav directory and give it new name
- Step 4 – Modify web.xml file with your Servlet mappings
- Step 5 – Start or re-start tomcat
- Step 6 – Test servlet from your web form, remembering to restart Tomcat anytime you change your Servlet or your web.xml file
  - Alternatively, you may be able to set reloadable=true