Java Enterprise Edition

Java Beans

POJO class :

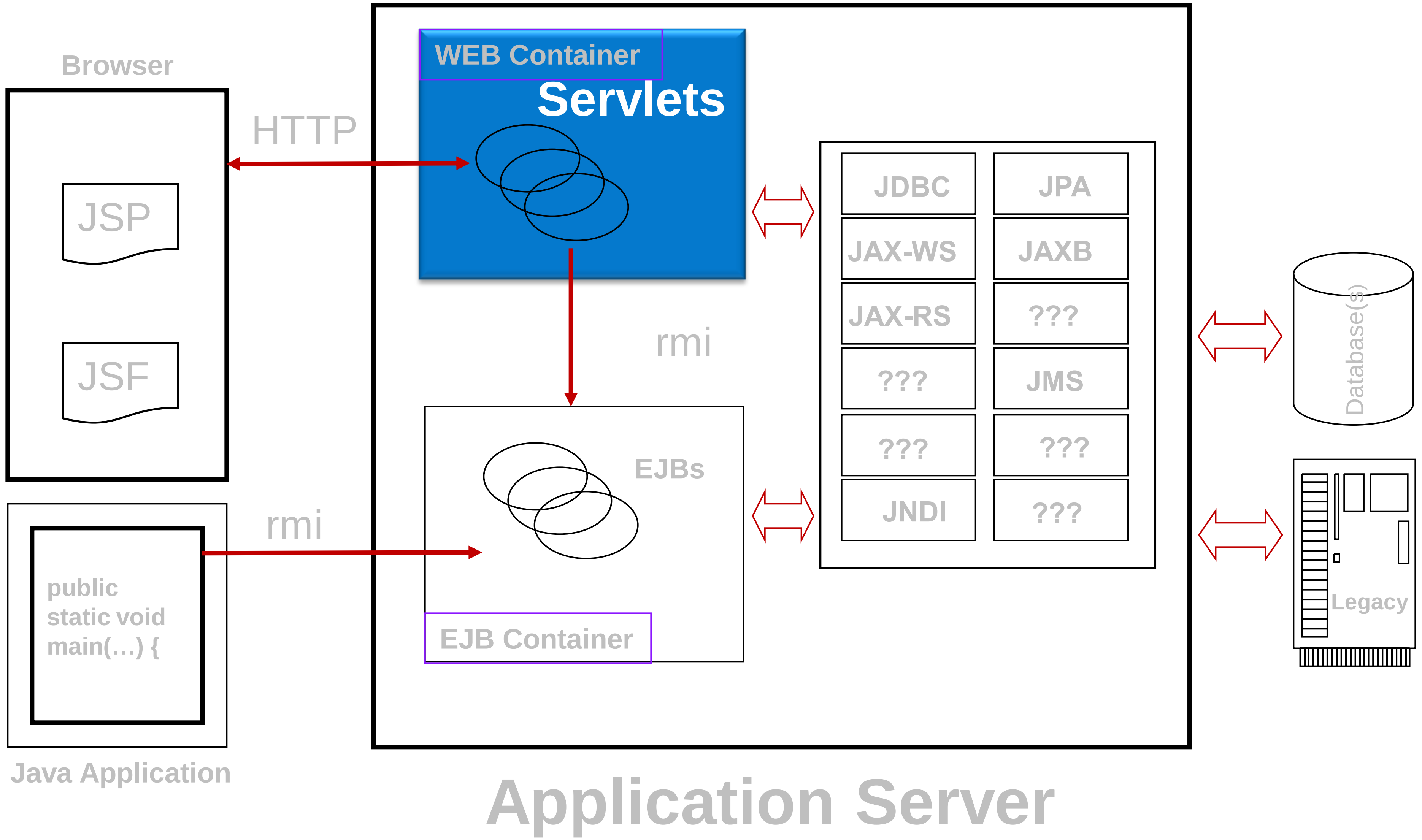
- *private* Attributes
- *public* getters and setters
- Default constructor

Java Bean : example

```
public class User {
    private String login;
    private String pass;

    public String getLogin() {
        return login;    }

    public void setLogin(String login) {
        this.login = login;    }

    public String getPass() {
        return pass;    }

    public void setPass(String pass) {
        this.pass = pass;    }
  }
```

Java EE APIs – The big picture

Focus on Servlets

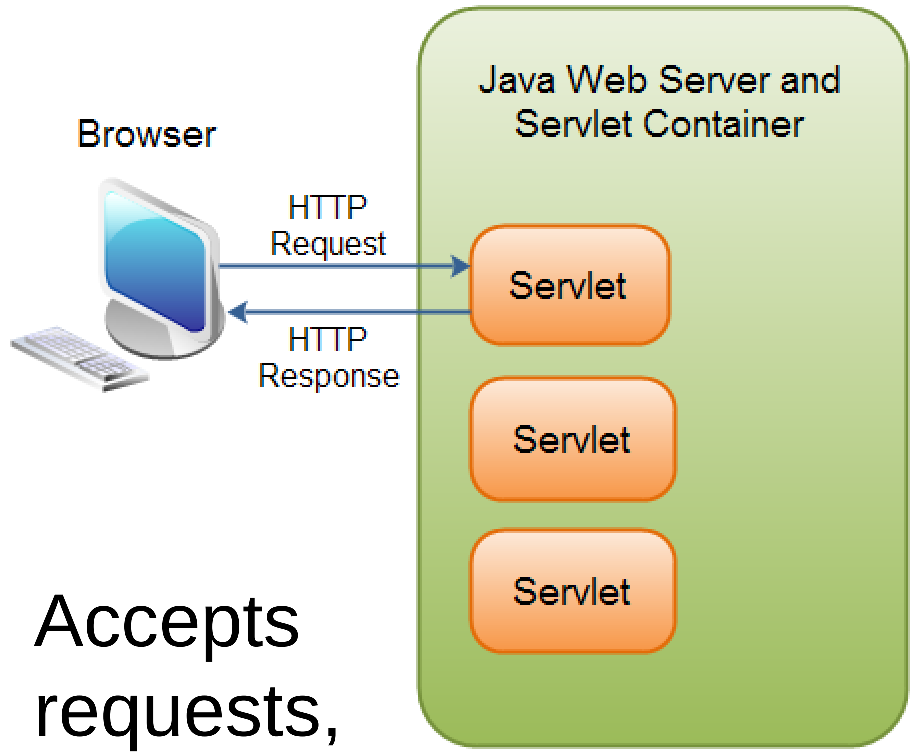


Web Application

- **Application software, that relies on web browser to render it**

- Building blocks in Java EE:
  - Web Container
  - Servlet
  - JSP or JSF

Web Container

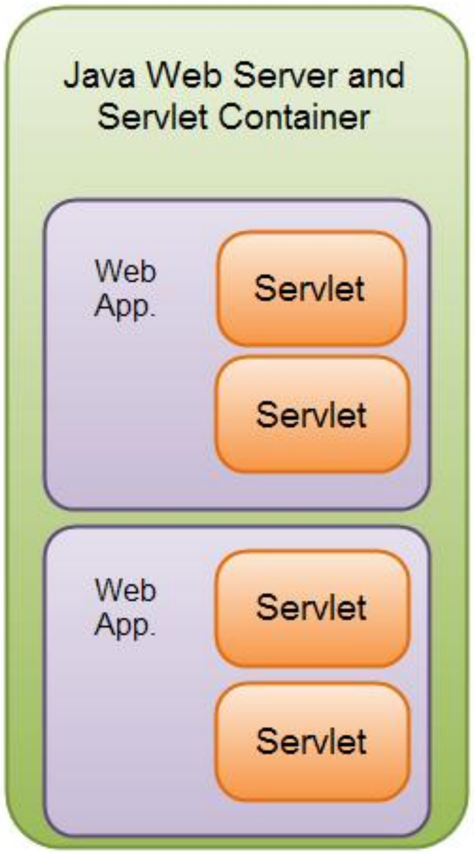


Manages component life cycles

Routes requests to applications

Accepts requests, sends responses

Web Container



Multiple applications inside one container

Deployment descriptor : WEB.XML (1)

- Instructs the container how handle this application

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi=
   <servlet>
       <servlet-name>HelloWorld</servlet-name>
       <servlet-class>exemple.HelloWorld</servlet-class>
   </servlet>
   <servlet-mapping>
       <servlet-name>HelloWorld</servlet-name>
       <url-pattern>/hello</url-pattern>
   </servlet-mapping>
   <session-config>
       <session-timeout>
           30
       </session-timeout>
   </session-config>
   <welcome-file-list>
       <welcome-file>index.jsp</welcome-file>
       </welcome-file-list>
   </web-app>
```

WEB.XML (2)

- In Servlet API version 3.0 most components of web.xml are replaced by annotations that go directly to Java source code.

- **Before Servlet 3.0 web.xml**

```
<servlet>
   <servlet-name>hello</servlet-name>
   <servlet-class>example.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

- **In Servlet 3.0 via annotations**

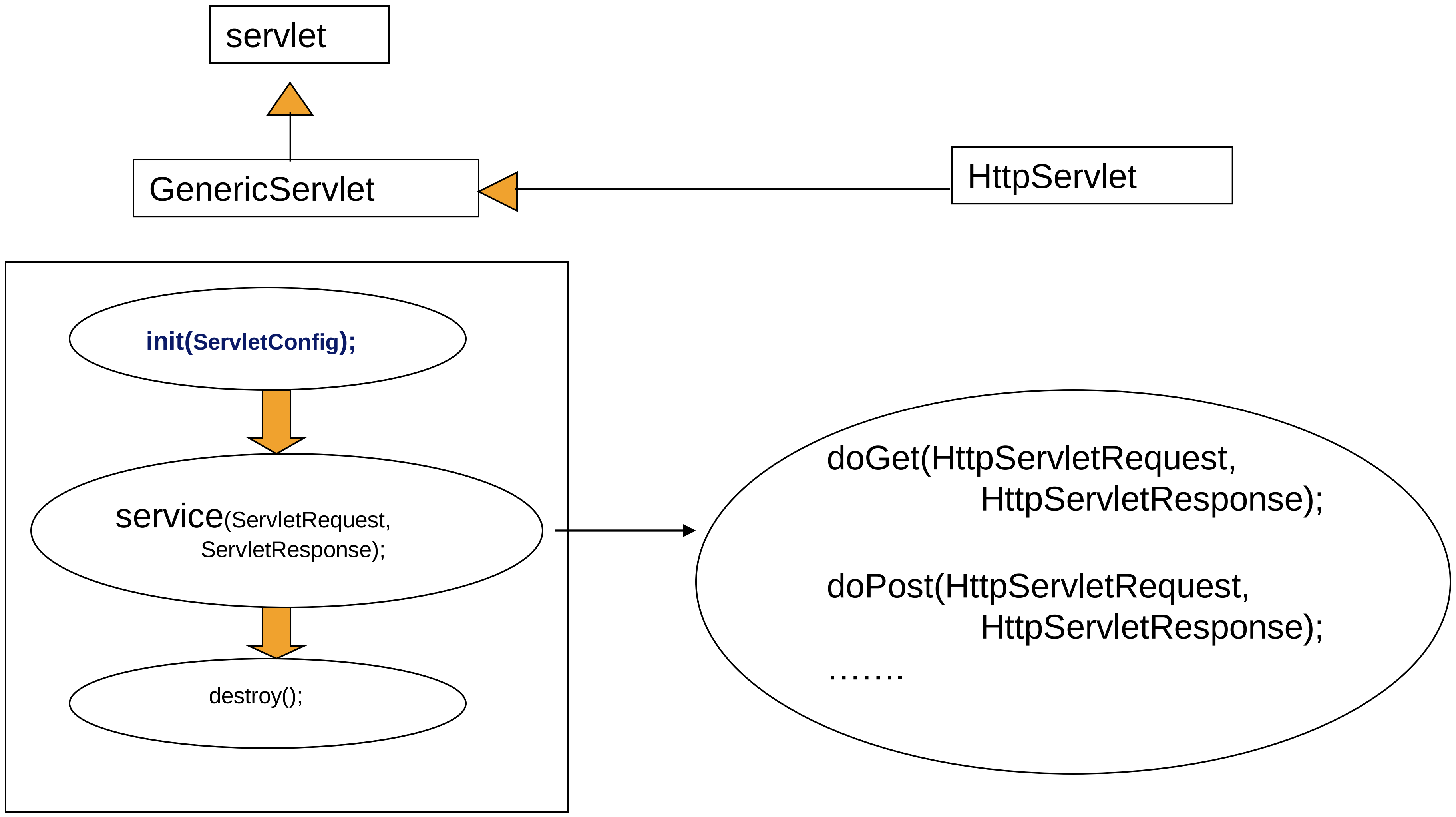
```
@WebServlet("/hello")
public class HelloServlet
   extends HttpServlet {
...
```

Servlets

Methods

- `ServletConfig getServletConfig()`
  - Returns reference to object, gives access to config info
- `void service ( ServletRequest request, ServletResponse response )`
  - Key method in all servlets
  - Provide access to input and output streams
    - *Read from and send to client*
- `void destroy()`
  - Cleanup method, called when servlet exiting

Life Cycle of Servlet



`HttpServlet` Class

- **`HttpServlet`**
  - Base class for web-based servlets
  - Overrides method **`service`**
    - Request methods:
      - ***`GET`*** *- retrieve HTML documents or image*
      - ***`POST`*** *- send server data from HTML form*
  - Methods **`doGet`** and **`doPost`** respond to **`GET`** and **`POST`**
    - Called by **`service`**
    - Receive **`HttpServletRequest`** and **`HttpServletResponse`** (return **`void`**) objects

`HttpServletRequest` Interface

- **`HttpServletRequest` interface**
  - Object passed to **`doGet`** and **`doPost`**
  - Extends **`ServletRequest`**
- **Methods**
  - **`String getParameter( String name )`**
    - Returns value of parameter **`name`** (part of **`GET`** or **`POST`**)
  - **`Enumeration getParameterNames()`**
    - Returns names of parameters (**`POST`**)
  - **`String[] getParameterValues( String name )`**
    - Returns array of strings containing values of a parameter
  - **`Cookie[] getCookies()`**
    - Returns array of **`Cookie`** objects, can be used to identify client

`HttpServletResponse` Interface

- **`HttpServletResponse`**
  - Object passed to **`doGet`** and **`doPost`**
  - Extends **`ServletResponse`**
- **Methods**
  - **`void addCookie( Cookie cookie )`**
    - Add **`Cookie`** to header of response to client
  - **`ServletOutputStream getOutputStream()`**
    - Gets byte-based output stream, send binary data to client
  - **`PrintWriter getWriter()`**
    - Gets character-based output stream, send text to client
  - **`void setContentType( String type )`**
    - Specify MIME type of the response (Multipurpose Internet Mail Extensions)
    - MIME type “text/html” indicates that response is HTML document.
    - Helps display data

Handling HTTP GET Requests

- **HTTP `GET` requests**
  - Usually gets content of specified URL
    - Usually HTML document (web page)
- **Example servlet**
  - Handles HTTP **`GET`** requests
  - User clicks **Get Page** button in HTML document
    - **`GET`** request sent to servlet **`HTTPGetServlet`**
  - Servlet dynamically creates HTML document displaying "Welcome to Servlets!"

Handling HTTP GET Requests

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

- Use data types from **`javax.servlet`** and **`javax.servlet.http`**

```java
public class HTTPGetServlet extends HttpServlet {
```

- **`HttpServlet`** has useful methods, inherit from it

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

- Method **`doGet`**
  - Responds to **`GET`** requests
  - Default action: **`BAD_REQUEST`** error (file not found)
  - Override for custom **`GET`** processing
  - Arguments represent client request and server response

Handling HTTP GET Requests

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

- **`setContentType`**
  - Specify content
  - **`text/html`** for HTML documents

```
        PrintWriter out;
        out = response.getWriter();                // get writer
```

- **`getWriter`**
  - Returns **`PrintWriter`** object, can send text to client
  - **`getOutputStream`** to send binary data (returns **`ServletOutputStream`** object)

Handling HTTP GET Requests

```
19        out.println( "<HTML><HEAD><TITLE>\n" );
20        out.println( "A Simple Servlet Example\n" );
21        out.println( "</TITLE></HEAD><BODY>\n" );
22        out.println( "<H1>Welcome to Servlets!</H1>\n" );
23        out.println( "</BODY></HTML>" );
```

- Lines 19-23 create HTML document
  - **`println`** sends response to client

Handling HTTP GET Requests

- **Running servlets**
  - Must be running on a server
    - Either a full application server (Glassfish)
    - Or 'just' a Web container (Tomcat)

Handling HTTP GET Requests

- **Port number**
  - Where server waits for client (handshake point)
  - Client must specify proper port number
    - Integers 1 - 65535, 1024 and below usually reserved
  - Well-known port numbers
    - Web servers - port 80 default
    - JSDK/Apache Tomcat 4.0 Webserver- port 8080
      - *Change in* ***`default.cfg`*** *(****`server.port=8080`****)*

Handling HTTP GET Requests

▶ HTML documents

```
<!-- HTTPGetServlet.html -->
<HTML>
   <HEAD>
      <TITLE>
         Servlet HTTP GET Example
      </TITLE>
   </HEAD>
```

- Comments: **`<!-- text -->`**
- Tags: **`<TAG> ... </TAG>`**
  - **`<HTML> ... <HTML>`** tags enclose document
  - **`<HEAD> ... </HEAD>`** - enclose header
    - *Includes **`<TITLE> Title </TITLE>`** tags*
    - *Sets title of document*

Handling HTTP GET Requests

```
      <FORM
         ACTION="http://localhost:8080/BasicReqHandlingServlet"
         METHOD="GET">
         <P>Click the button to have the servlet send
            an HTML document</P>
         <INPUT TYPE="submit" VALUE="Get HTML Document">
      </FORM>
   </BODY>
```

- Document body (**`<BODY>`** tags)
  - Has literal text and tags for formatting
- Form (**`<FORM>`** tags)
  - **`ACTION`** - server-side form handler
  - **`METHOD`** - request type

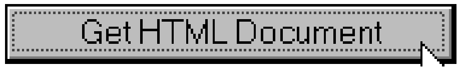
Handling HTTP GET Requests

```
ACTION="http://localhost:8080/BasicReqHandlingServlet"
```

- **ACTION**
  - **`localhost`** - your computer
  - **`:8080`** - port
  - **`/servlet`** - servlet name

```
<INPUT TYPE="submit" VALUE="Get HTML Document">
```

- GUI component
  - **`INPUT`** element
  - **`TYPE`** - **`"submit"`** (button)
  - **`VALUE`** - label
  - When pressed, performs **`ACTION`**
  - If parameters passed, separated by `?` in URL



```java

// Creating and sending a page to the client
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HTTPGetServlet extends HttpServlet {
   public void doGet( HttpServletRequest request,
                      HttpServletResponse response )
      throws ServletException, IOException
   {
      PrintWriter out;

      response.setContentType( "text/html" );  // content type
      out = response.getWriter();             // get writer

      // create and send HTML page to client

      out.println( "<HTML><HEAD><TITLE>\n" );
      out.println( "A Simple Servlet Example\n" );
      out.println( "</TITLE></HEAD><BODY>\n" );
      out.println( "<H1>Welcome to Servlets!</H1>\n" );
      out.println( "</BODY></HTML>" );
   }
}
```

Import necessary classes and inherit methods from **`HttpServlet`**.

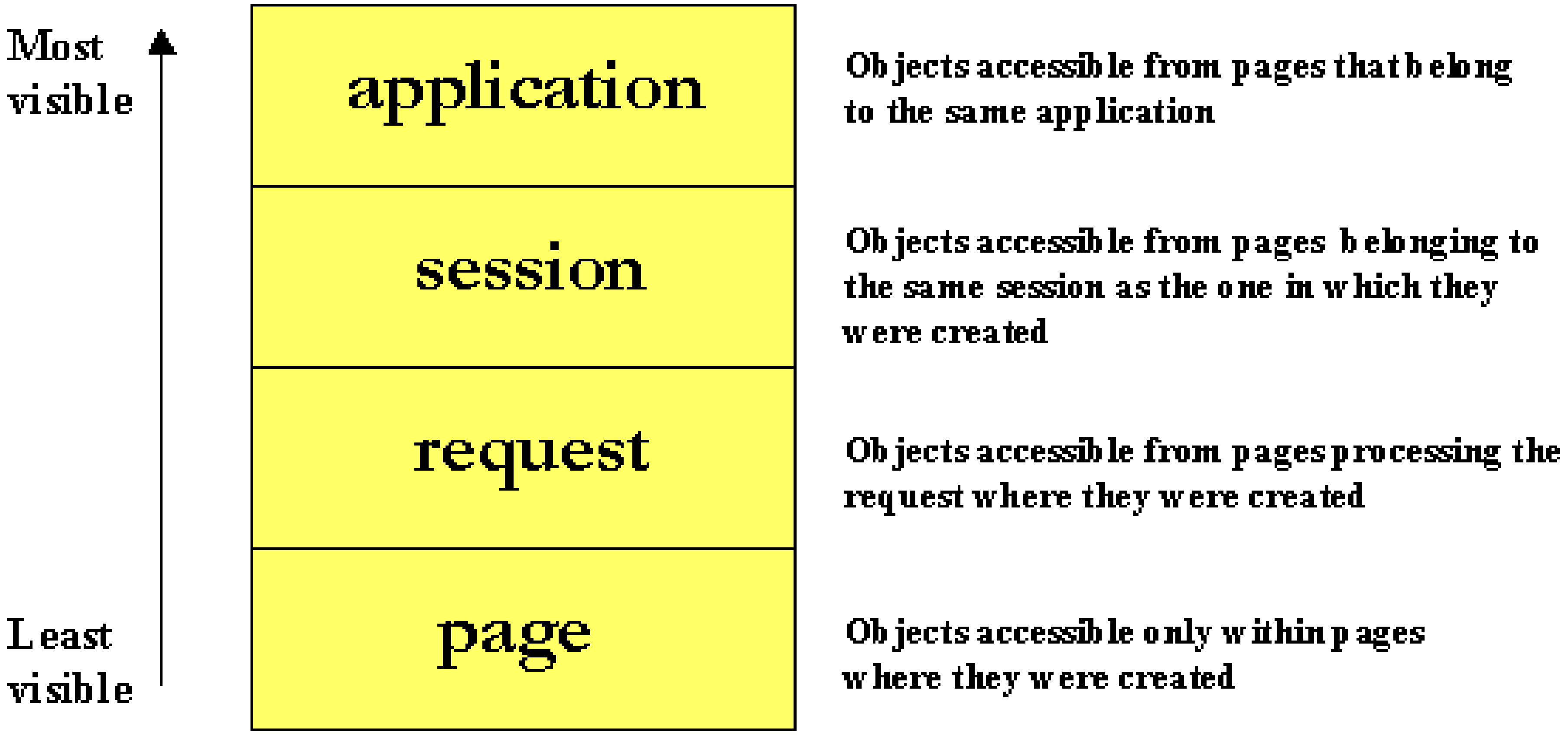
Create PrintWriter object. Create HTML file and send to client.

```

<HTML>
   <HEAD>
      <TITLE>
         Servlet HTTP GET Example
      </TITLE>
   </HEAD>
   <BODY>
      <FORM
      ACTION="http://localhost:8080/BasicReqHandlingServlet"
         METHOD="GET">
         <P>Click the button to have the servlet send
            an HTML document</P>
         <INPUT TYPE="submit" VALUE="Get HTML Document">
      </FORM>
   </BODY>
</HTML>
```

ACTION specifies form handler, **METHOD** specifies request type.

Creates submit button, performs **ACTION** when clicked.

Scopes (First look)



Objects accessible from pages that belong to the same application

Objects accessible from pages belonging to the same session as the one in which they were created

Objects accessible from pages processing the request where they were created

Objects accessible only within pages where they were created