Servlets

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

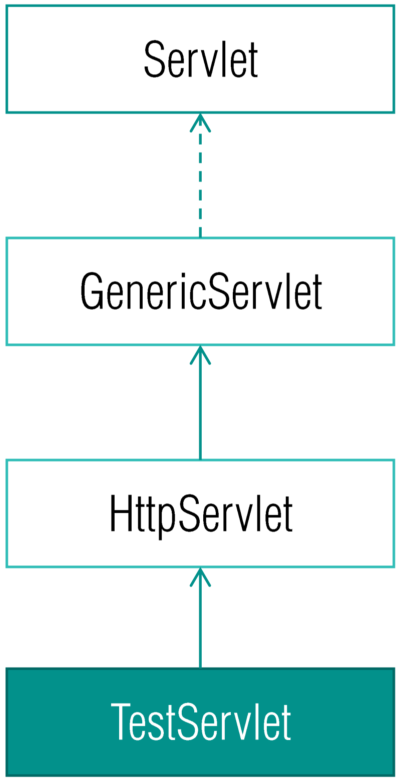
- Introduction to Servlets
- Building a Servlet
- Web App Deployment Descriptor(DD)
- Servlet API
- Servlet Lifecycle and methods
- Request-Response processing cycle
- HTTPServletRequest
- HTTPServletResponse
- ServletConfig
- ServletContext

Servlets

- Java technology to write powerful server side programs that extends the functionality of a Web Server
- Provides platform-independent methods for building interactive web applications
- Act as a middle layer between a request coming from a Web browser and databases or applications
- Advantages
  - Requests execute as separate threads and as threads are light weight the performance does not degrade
  - Object oriented and secure as it is purely written in java
  - Platform independent

Building a Servlet

- Packages
  - javax.servlet
  - javax.servlet.http
- Servlet: Interface
  - defines life-cycle methods
    - to initialize a servlet
    - to service requests
    - to remove a servlet from the server
- GenericServlet: Abstract Class
  - Defines a generic, protocol-independent servlet
  - Implements the Servlet and ServletConfig interfaces
- HttpServlet: Abstract Class
  - HTTP protocol specific



Servlet Template

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

public class TestServlet extends HttpServlet {

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

              // Use "request" to read incoming HTTP headers
              // and query data from HTML forms.

              // Use "response" to specify the HTTP response status
              // code and headers (e.g., the content type, cookies).

              PrintWriter out = response.getWriter();
              // Use "out" to send content to browser.
       }
}
```

Web App Deployment Descriptor(DD)

- Is a xml file named web.xml
- Root tag in web.xml is <web-app> …</web-app>
- Stored under the WEB-INF directory
- Provides configuration and deployment information for the Web components

- Used to define the following
  - Servlet Declaration, Servlet Mappings and Servlet initialization parameters
  - Context initialization parameters
  - Security configuration including login-config, security-constraint, security-role etc.
  - Welcome File list and Error Pages

Using DD to map URLs to servlets

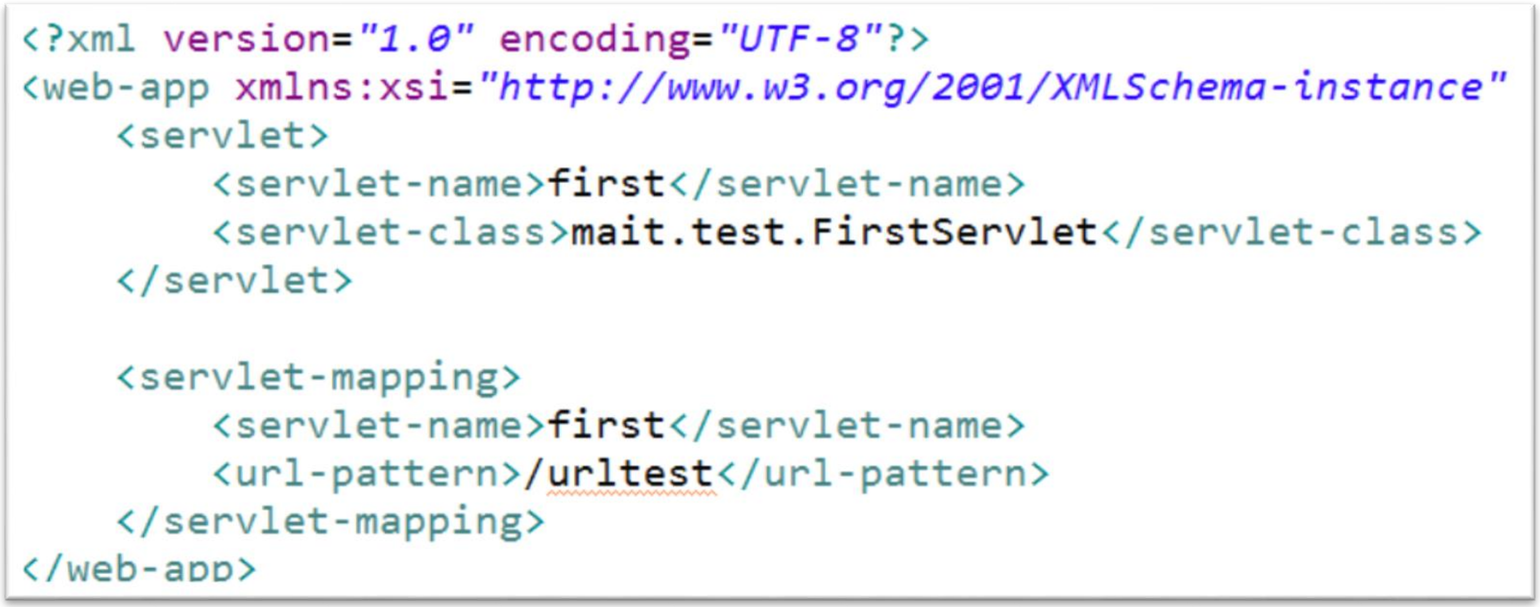
Tag used to define and map url to Servlets

- <servlet>

  maps internal name to fully-qualified class name

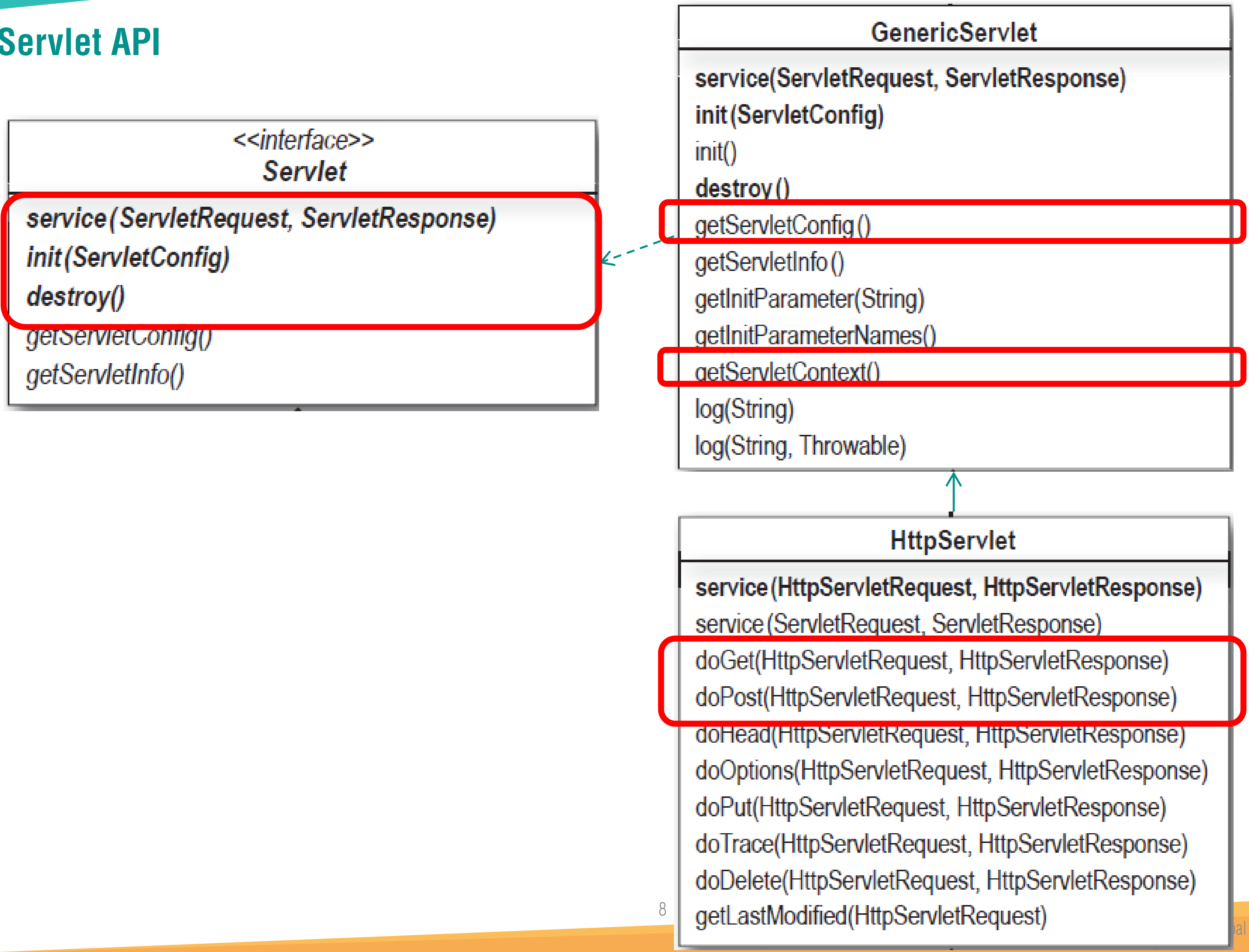
- <servlet-mapping>

  maps internal name to public URL name



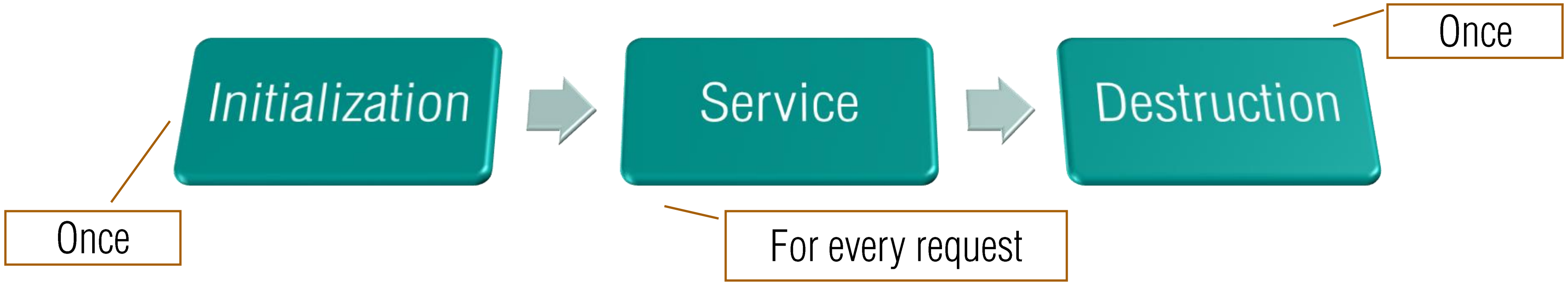
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <servlet>
        <servlet-name>first</servlet-name>
        <servlet-class>mait.test.FirstServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>first</servlet-name>
        <url-pattern>/urltest</url-pattern>
    </servlet-mapping>
</web-app>
```

Servlet API



<<interface>>
Servlet
service(ServletRequest, ServletResponse)
init(ServletConfig)
destroy()
getServletConfig()
getServletInfo()
GenericServlet
service(ServletRequest, ServletResponse)
init(ServletConfig)
init()
destroy()
getServletConfig()
getServletInfo()
getInitParameter(String)
getInitParameterNames()
getServletContext()
log(String)
log(String, Throwable)
HttpServlet
service(HttpServletRequest, HttpServletResponse)
service(ServletRequest, ServletResponse)
doGet(HttpServletRequest, HttpServletResponse)
doPost(HttpServletRequest, HttpServletResponse)
doHead(HttpServletRequest, HttpServletResponse)
doOptions(HttpServletRequest, HttpServletResponse)
doPut(HttpServletRequest, HttpServletResponse)
doTrace(HttpServletRequest, HttpServletResponse)
doDelete(HttpServletRequest, HttpServletResponse)
getLastModified(HttpServletRequest)

Servlet Lifecycle and methods



A Servlet goes through Initialization, Service and destruction phase in its lifecycle

- **Initialization**
  - Container initializes the servlet, when servlet receives the first client request or during web application startup
    - Loads the servlet class, creates and runs the constructor to create Servlet instance
    - Calls the init() method to initialize the Servlet
  - init method is called by container only once during the life of a servlet
  - Initialization gives the servlet access to the ServletConfig and ServletContext objects
  - Used for perform any setup processing

```
public void init(ServletConfig) throws ServletException
```

Lifecycle Methods

- Service
  - Most of a servlet's life is spent running a service() method for a client request.
  - For every client request, container finds a thread, which executes the servlet's service() method
  - Service() method checks the HTTP Request method(GET,POST,etc) and calls doGet, doPost ,etc based on the request method
  - Servlet should be initialized before it can service any client requests
  - Multiple threads may invoke the service() method of a given servlet at the same time

```
public void Service(ServletRequest,ServletResponse) throws ServletException, IOException
```

- Destruction
  - Destroys the servlet and release the resources
    - Container calls destoy() once during the life of a servlet

```
public void destroy()
```

doGet() and doPost()

- Programmer needs to override the doGet(), doPost, etc methods in the Servlet based on the type of request

- doGet() method is invoked by service() method when
  - User clicks on a link in a web page
  - User enters URL in browser address bar and presses enter
  - User submits a HTML form that has no method specified

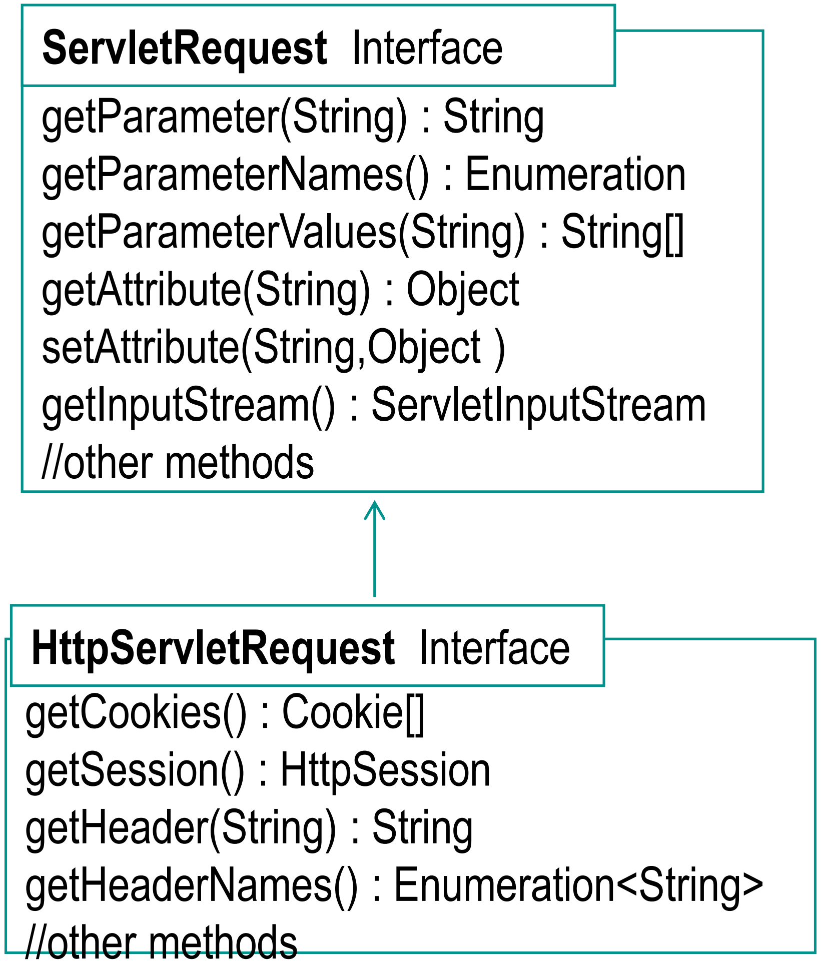
```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
 // Servlet code
}
```

- doPost() method is invoked by service() method when
  - User submits a HTML form with method specified as POST

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
 // Servlet code
}
```

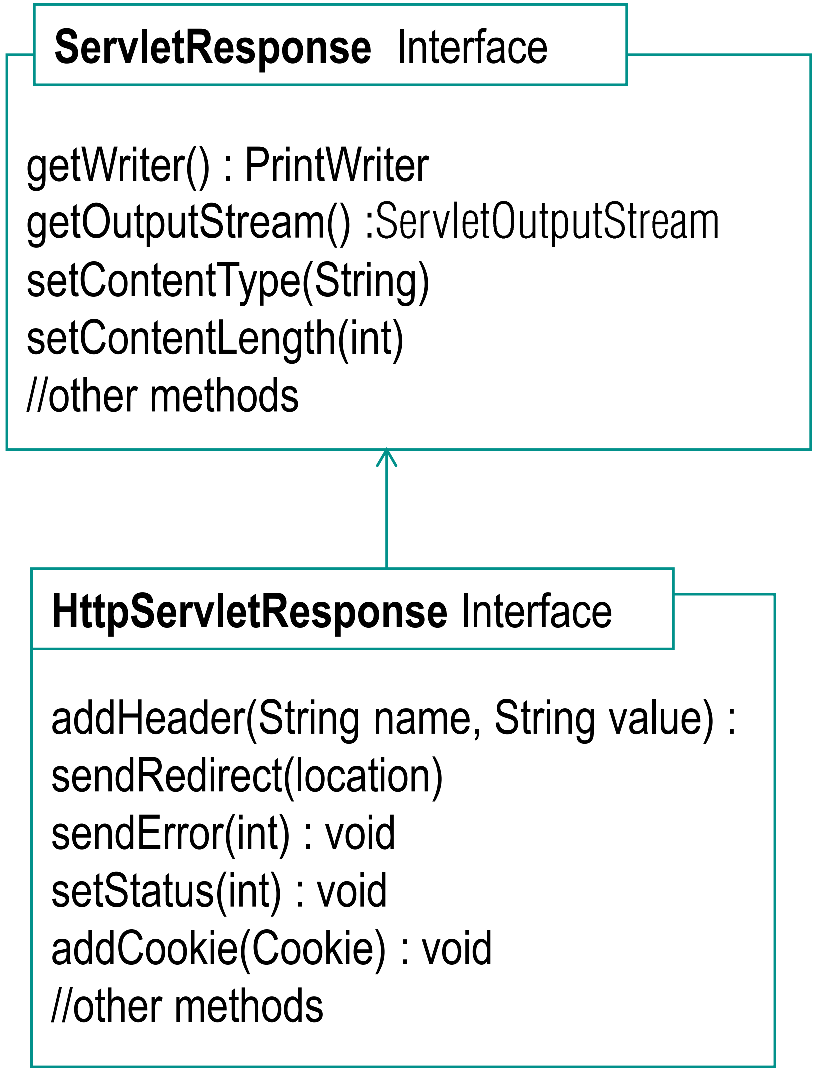
HTTPServletRequest

- Container creates a HttpServletRequest object and passes it as an argument to the servlet's service method
- Used to provide information about the client request to a servlet.
- Contains the parameters sent by the client
- Request object is valid only within the scope of a servlet's service method
- Methods
  - getParameter(String pname) method returns the value of a request parameter as a String
  - getInputStream() returns a ServletInput Stream, which can be used to read the binary data sent along with the request

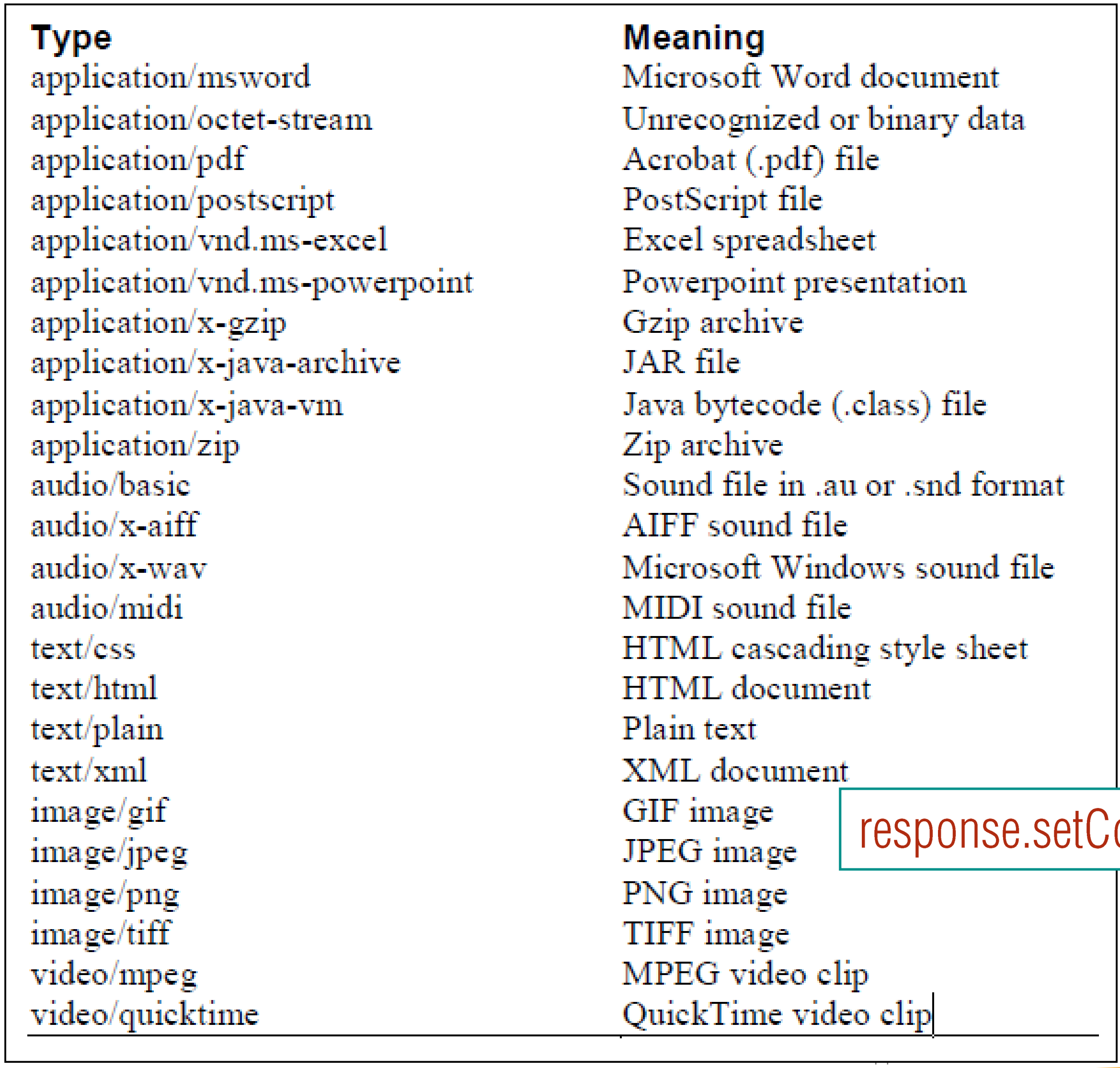

ServletRequest Interface
getParameter(String) : String
getParameterNames() : Enumeration
getParameterValues(String) : String[]
getAttribute(String) : Object
setAttribute(String,Object)
getInputStream() : ServletInputStream
//other methods
HttpServletRequest Interface
getCookies() : Cookie[]
getSession() : HttpSession
getHeader(String) : String
getHeaderNames() : Enumeration<String>
//other methods

HTTPServletResponse

- Container creates a HttpServletResponse object and passes it as an argument to the servlet's service method
- Used by the Servlet to send response to the client. Servlet writes to the response object
- Response object is valid only within the scope of a servlet’s service method
- Methods
  - setContentType(String) : Used to set the contentType header of the response
  - getWriter() method returns a PrintWriter object that can send character text to the client
  - getOutputStream() method returns a ServletOutputStream suitable for writing binary data in the response

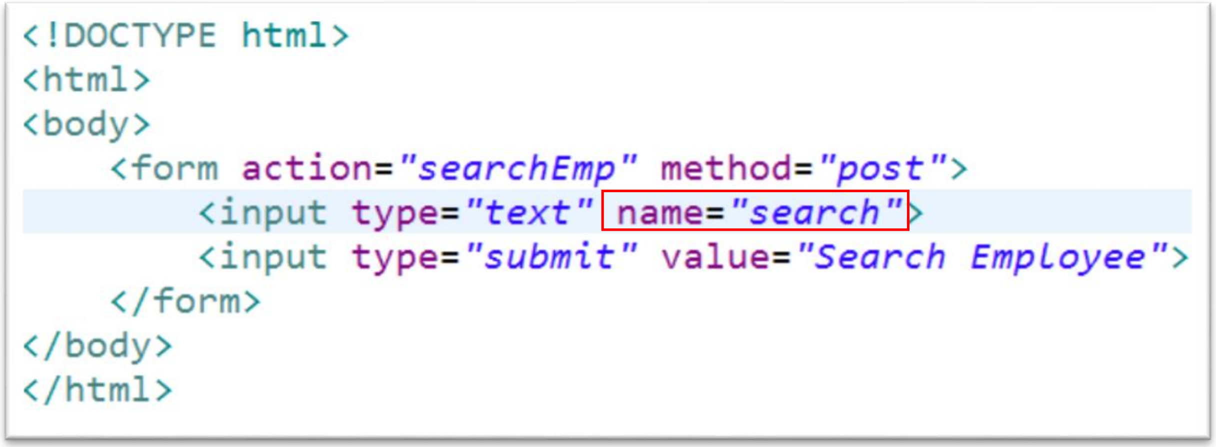

ServletResponse Interface
getWriter() : PrintWriter
getOutputStream() :ServletOutputStream
setContentType(String)
setContentLength(int)
//other methods
HttpServletResponse Interface
addHeader(String name, String value) :
sendRedirect(location)
sendError(int) : void
setStatus(int) : void
addCookie(Cookie) : void
//other methods

Common mime types

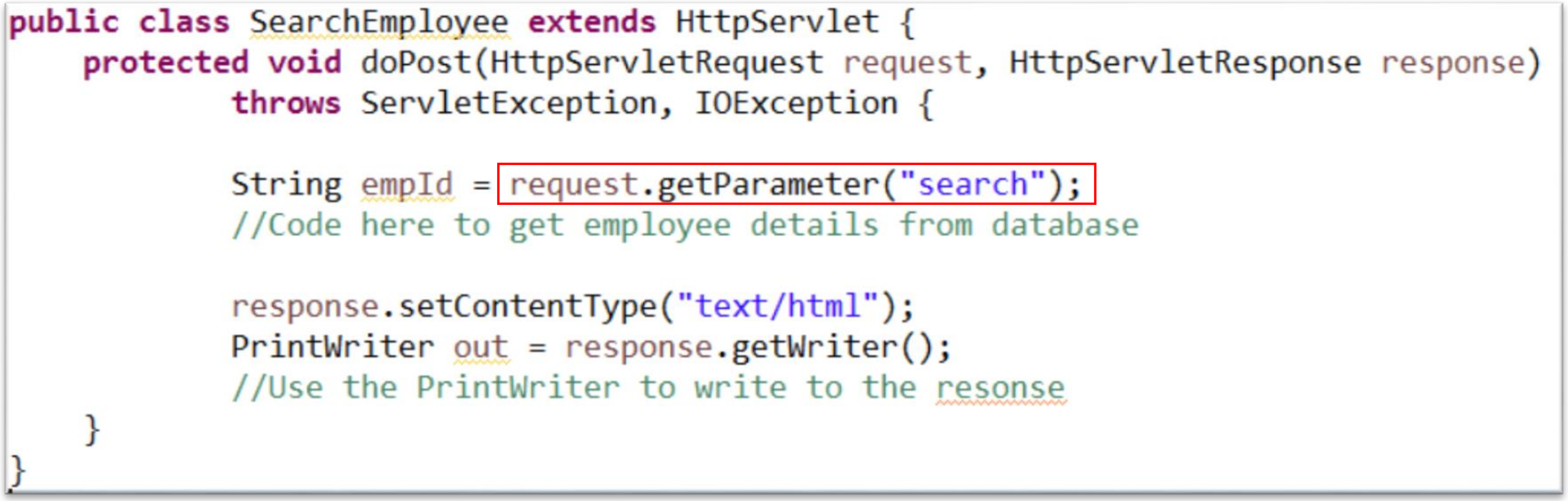


Type	Meaning
application/msword	Microsoft Word document
application/octet-stream	Unrecognized or binary data
application/pdf	Acrobat (.pdf) file
application/postscript	PostScript file
application/vnd.ms-excel	Excel spreadsheet
application/vnd.ms-powerpoint	Powerpoint presentation
application/x-gzip	Gzip archive
application/x-java-archive	JAR file
application/x-java-vm	Java bytecode (.class) file
application/zip	Zip archive
audio/basic	Sound file in .au or .snd format
audio/x-aiff	AIFF sound file
audio/x-wav	Microsoft Windows sound file
audio/midi	MIDI sound file
text/css	HTML cascading style sheet
text/html	HTML document
text/plain	Plain text
text/xml	XML document
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
image/tiff	TIFF image
video/mpeg	MPEG video clip
video/quicktime	QuickTime video clip

```
response.setContentType("text/html“}
```



```
<!DOCTYPE html>
<html>
<body>
    <form action="searchEmp" method="post">
        <input type="text" name="search">
        <input type="submit" value="Search Employee">
    </form>
</body>
</html>
```



```
public class SearchEmployee extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

            String empId = request.getParameter("search");
            //Code here to get employee details from database

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            //Use the PrintWriter to write to the resonse
    }
}
```

Request-Response processing cycle

- Servlets are managed by the Web Container
- User clicks a link that refers to the URL of a servlet
- Web Server receives the request and passes it to the container
- Container creates two objects, HttpServletRequest and HttpServletResponse
- Container locates the servlet from web.xml, loads and instantiates it, and runs its **init()** method
- Container creates/allocates a thread for that request. The Thread executes servlet's **service()** method. Request and response object are passed to the service method
- service() method calls the **doGet()** or **doPost()** method based on the request method
- **doGet()** or **doPost()** method is responsible for generating the response
- The output from this method is stuffed into the response object
- Container converts the response object to an HTTP response and sends it to browser

Http Status codes

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error

200 (OK)
Default for servlets

301 (Moved Permanently)
Requested document permanently moved elsewhere
Browsers go to new location automatically indicated in Location header

302 (Moved Temporarily`)
Requested document temporarily moved elsewhere
Browsers go to new location automatically indicated in location header
Servlets should use sendRedirect, not setStatus, when setting this header.

401 (Unauthorized)
Browser tried to access password-protected page without proper Authorization

404 (Not Found)
No such page. Servlets should use sendError to set this

Status Code and Errors

This method is used to set the return status code when there is no error

<u>setStatus(int code);</u>

resp.setStatus(HttpServletResponse.SC_FOUND);

SC_OK, SC_NOT_FOUND, etc . are constants in HttpServletResponse

<u>sendError(int code, String message)</u>

resp.sendError(HttpServletResponse.*SC_UNAUTHORIZED, "User not Authorized");*

- Sends an error response to the client using the specified status code
- If error page is defined for the error code in web.xml, Container displays the page
- If error page is not defined for the error code, Container generates a error page

Setting Response Headers

- setHeader(String headerName, String headerValue)

  resp.setHeader("Cache-Control", "no-cache")

- addHeader(String headerName, String headerValue)
  - Adds new occurrence of header instead of replacing
- setDateHeader(String name, long millisecs)

  setDateHeader("myDate",new Date());

- Some Important Headers
  - Content-type
  - Cache-Control
    - 'No-cache' value prevents browsers from caching page
  - Refresh
    - The number of seconds after which browser should reload page.
  - Expires
    - The time at which document should be considered out-of date and should no longer be cached

ServletConfig

ServletConfig Interface

getServletContext
getInitParameter
getInitParameterNames
getServletName
//other methods

- A servlet configuration object used by a Container to pass information to a servlet during initialization
- One ServletConfig object per servlet
- Container passes the ServletConfig object as an argument when calling servlet's init() method
- ServletConfig stores the initialization parameters of the servlet, which are defined in web.xml
- Used to access the ServletContext
- ServletConfig interface is implemented by Servlet container provider

```
String name = getServletConfig().getServletName();
```

Servlet Init Parameters

In the web.xml file:

```
<servlet>
    <servlet-name>ParamTest</servlet-name>
    <servlet-class>com.mycompany.ParamServlet</servlet-class>
    <init-param>
        <param-name>JDBC_URL</param-name>
        <param-value>jdbc:oracle:thin@localhost:1521/XE</param-value>
    </init-param>
</servlet>
```

In the servlet code:

```
String url = getServletConfig().getInitParameter(“JDBC_URL”);
```

ServletContext

ServletContext Interface

getInitParameter()
getContextPath()
getServerInfo()
getAttribute(String name)
setAttribute(String name, Object obj)
//other methods

- One ServletContext object per web app
- Created/destroyed when the web application is started/shutdown
- Most commonly used to share the data across application, obtain URL references to resources, dispatching a request etc..
- Gives servlets, access to information about their Environment like the name and version of the Container
- Used as a kind of application bulletin-board, where messages (called attributes) can be put for providing access to other parts of application
- Available to all Servlet's and JSP's in the web app

```
String server = getServletContext().getServerInfo();
```

Context Init Parameters

In the DD (web.xml) file:

```
<context-param>
    <param-name>adminEmail</param-name>
    <param-value>support@mycompany.com</param-value>
</context-param>
```

In the servlet code:

```
String mail = getServletContext().getInitParameter(“adminEmail”);

String mail = getServletConfig().getServletContext().getInitParameter(“adminEmail”)
```

More on Deployment Descriptor

```
 <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>

<servlet>
    <servlet-name>ServletOne</servlet-name>
    <servlet-class>com.mycompany.TestServlet</servlet-class>
     <load-on-startup>1</load-on-startup>
</servlet>
```

Directory Match

```
<url-pattern>/Test/*</url-pattern>
```

Extension match

```
 <url-pattern>*.test</url-pattern>
```

```
<error-page>
    <error-code>404</error-code>
    <location>/CustomerErrorPage.html</location>
 </error-page>

<error-page>
    <exception-type>java.io.IOException</exception-type>
    <location>/html/IOError.html</location>x
 </error-page>
```

SUMMARY

Servlets

SUMMARY

In this lesson, you've learned to:

- Introduction to Servlets
- Building a Servlet
- Web App Deployment Descriptor(DD)
- Servlet API
- Servlet Lifecycle and methods
- Request-Response processing cycle
- HTTPServletRequest
- HTTPServletResponse
- ServletConfig
- ServletContext