Dr. Ambedkar Institute of Technology, Bengaluru

Department of Computer Science & Engg.

1

UG Programme

Subject Name: Java Programming
Subject Code: 18CS52

PRESENTATION BY

DR. SMITHA SHEKAR B
ASSOCIATE PROFESSOR
DEPT. OF C.S.E
DR. AMBEDKAR INSTITUTE OF TECHNOLOGY
BANGALORE-560056

UNIT - 5

Servlet:
The Life Cycle of a Servlet;
Using Tomcat for Servlet Development;
A simple Servlet;
The Servlet API;
Packages;
Handling HTTP Requests and Responses;
Handling Cookies; Session Tracking.

Java Server page (JSP):
Overview of JSP;
JSP tags;
Invoking java code with Scripting Elements.

Dr. Smitha Shekar B

Agenda

- Servlet
- CGI
- HTTP
- Content Types
- Life Cycle of Servlet
- Servlet API

Dr. Smitha Shekar B

What is Servlets?

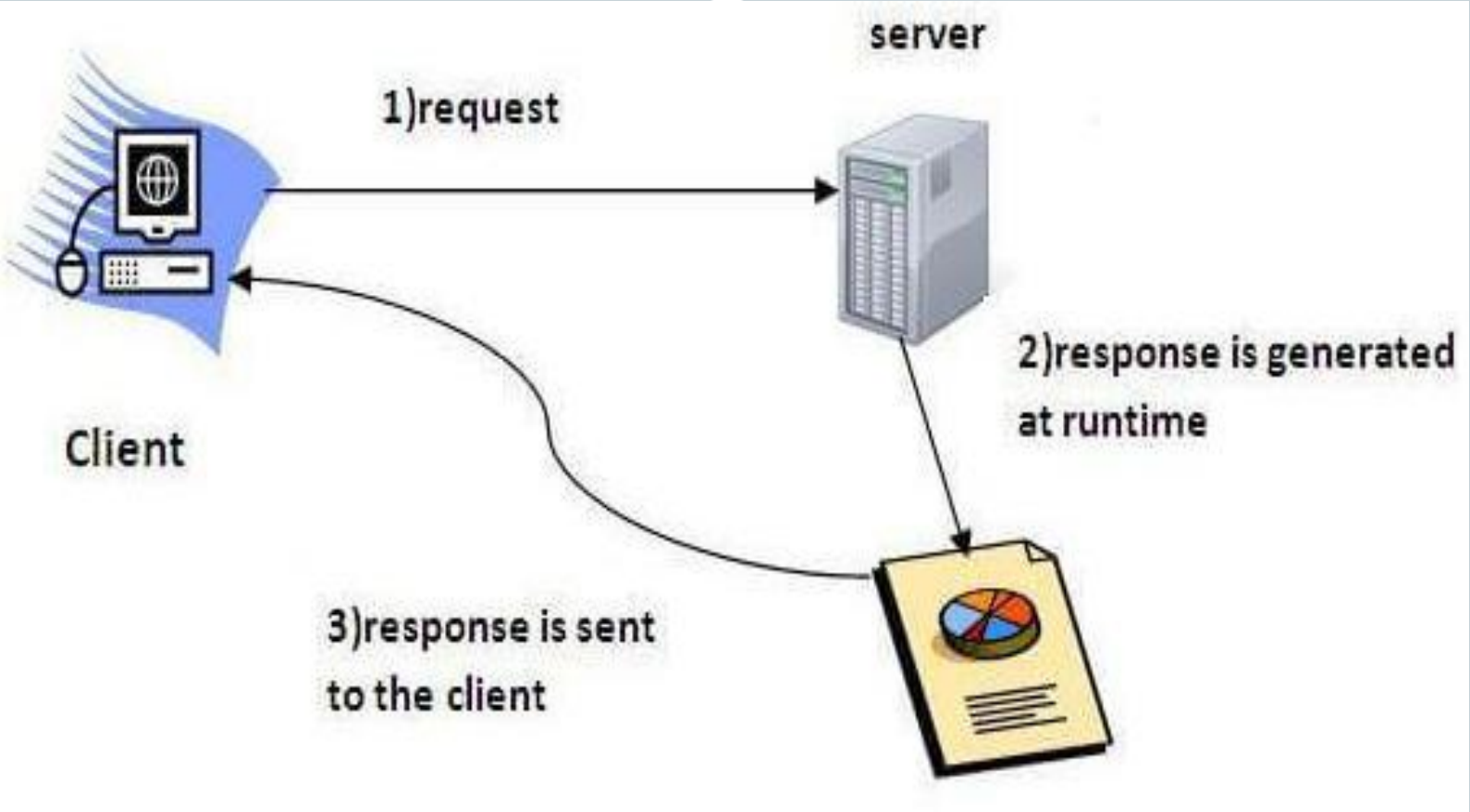
- **Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).
- **Servlet** technology is robust and scalable because of JAVA language.
  - Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology.
- There are many interfaces and classes in the Servlet API,
  - such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

Dr. Smitha Shekar B

What is a Servlet?

- **Servlet** can be described in many ways, depending on the context.
  - Servlet is a technology which is used to create a web application.
  - Servlet is an API that provides many interfaces and classes including documentation.
  - Servlet is an interface that must be implemented for creating any Servlet.
  - Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
  - Servlet is a web component that is deployed on the server to create a dynamic web page.

Dr. Smitha Shekar B



Dr. Smitha Shekar B

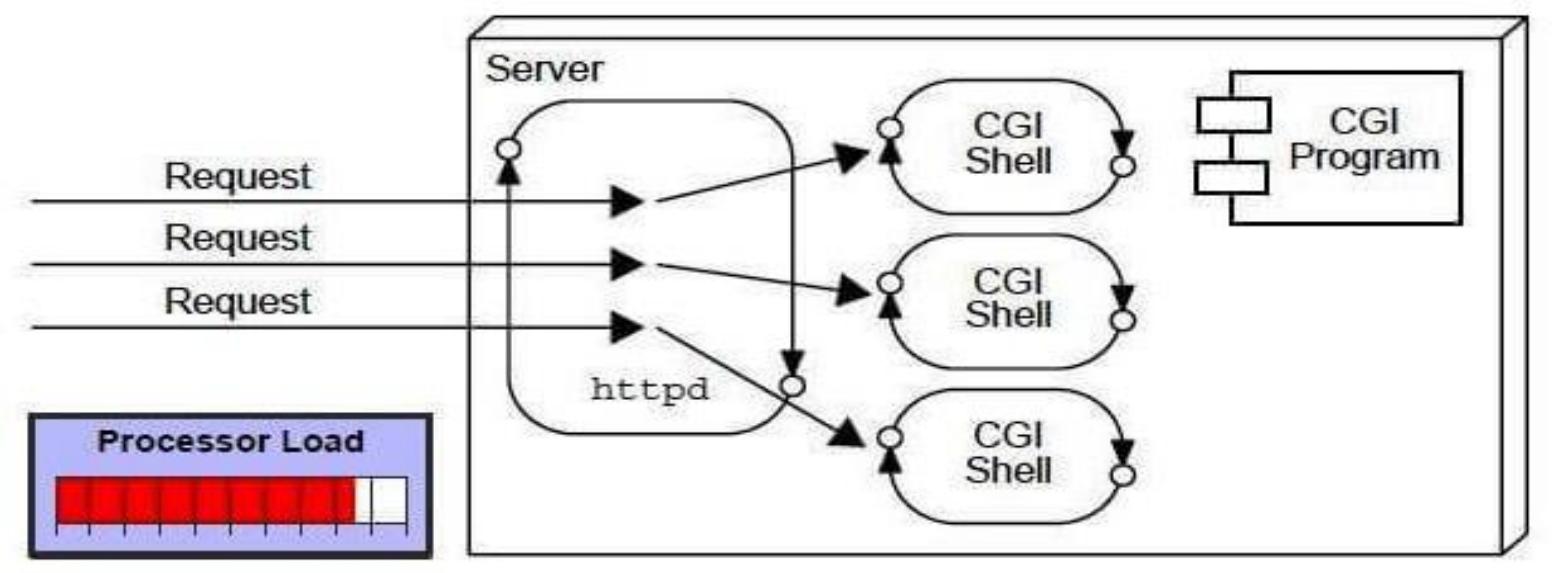
What is a web application?

- A web application is an application accessible from the web.
- A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript.
- The web components typically execute in Web Server and respond to the HTTP request.

Dr. Smitha Shekar B

CGI (Common Gateway Interface)

- CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request.
- For each request, it starts a new process.

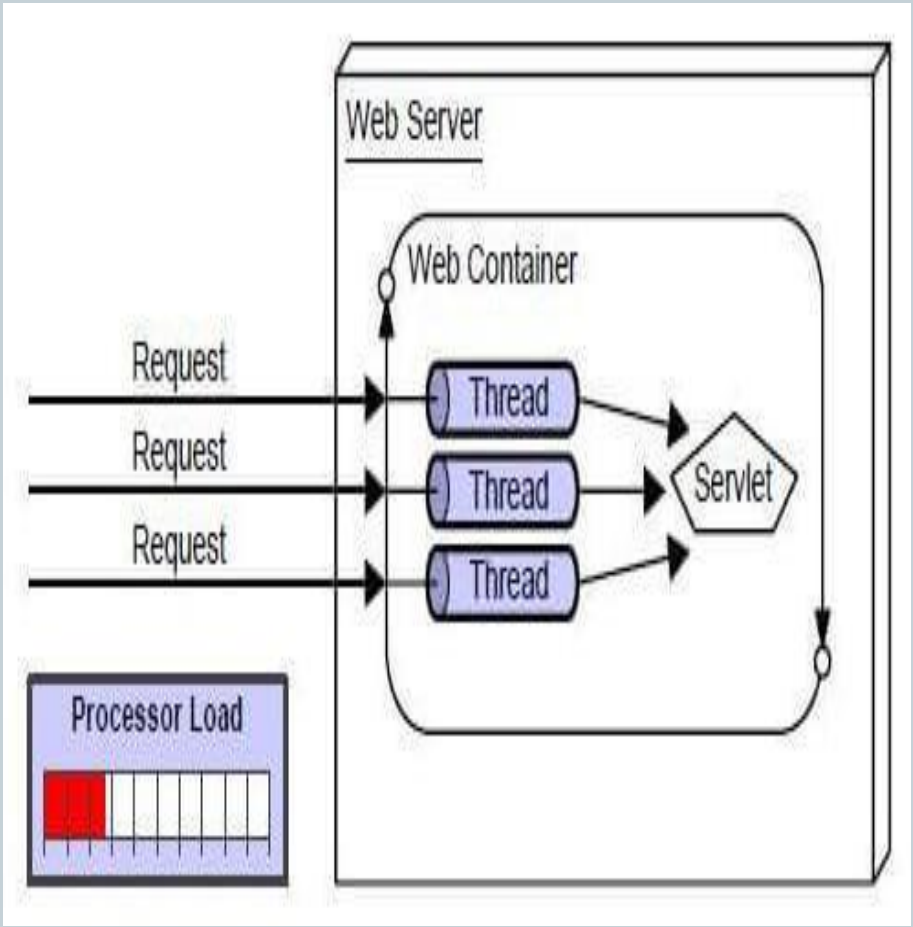


Disadvantages of CGI

- There are many problems in CGI technology:
  - If the number of clients increases, it takes more time for sending the response.
  - For each request, it starts a process, and the web server is limited to start processes.
  - It uses platform dependent language e.g. C, C++, perl.

Dr. Smitha Shekar B

Advantages of Servlet



- There are many advantages of Servlet over CGI.
- The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.
- The advantages of Servlet are as follows:
  - **Better performance:** because it creates a thread for each request, not process.
  - **Portability:** because it uses Java language.
  - **Robust:** JVM manages Servlets, so we don't need to worry about the

Web Terminology

Servlet Terminology	Description
Website: static vs dynamic	It is a collection of related web pages that may contain text, images, audio and video.
HTTP	It is the data communication protocol used to establish communication between client and server.
HTTP Requests	It is the request sent by the computer to a web server that contains all sorts of potentially interesting information.
Get vs Post	It gives the difference between GET and POST request.
Container	It is used in java for dynamically generating the web pages on the server side.
Server: Web vs Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

Website

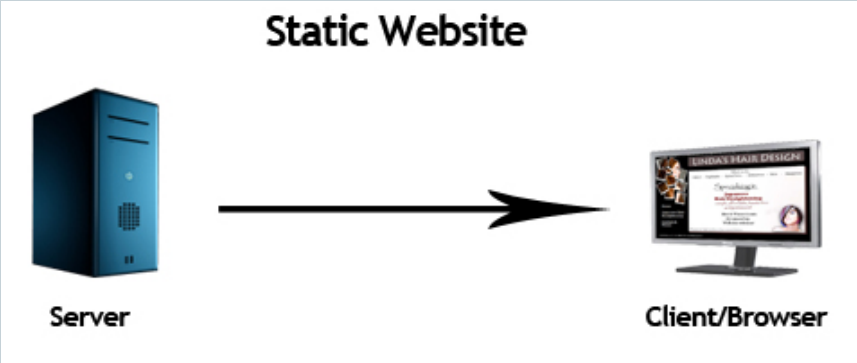
- Website is a collection of related web pages that may contain text, images, audio and video.
- The first page of a website is called home page.
- Each website has specific Internet address (URL) that you need to enter in your browser to access a website.
- Website is hosted on one or more servers and can be accessed by visiting its homepage using a computer network.
- A website is managed by its owner that can be an individual, company or an organization.
- A website can be of two types:
  - Static Website
  - Dynamic Website

Dr. Smitha Shekar B

Static website

- Static website is the basic type of website that is easy to create.
  - You don't need the knowledge of web programming and database design to create a static website. Its web pages are coded in HTML.

- The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.

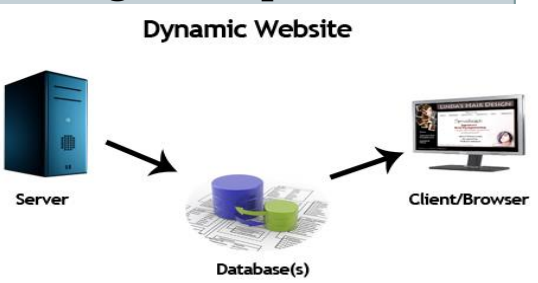


Dr. Smitha Shekar B

Dynamic website

- Dynamic website is a collection of dynamic web pages whose content changes dynamically.
  - It accesses content from a database or Content Management System (CMS). Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.
- Dynamic website uses client-side scripting or server-side scripting, or both to generate dynamic content.
- Client side scripting generates content at the client computer on the basis of user input. The web browser downloads the web page from the server and processes the code within the page to render information to the user.
- In server side scripting, the software runs on the server and processing is completed in the server then plain pages are sent to the user.

Dr. Smitha Shekar B



Static vs Dynamic website

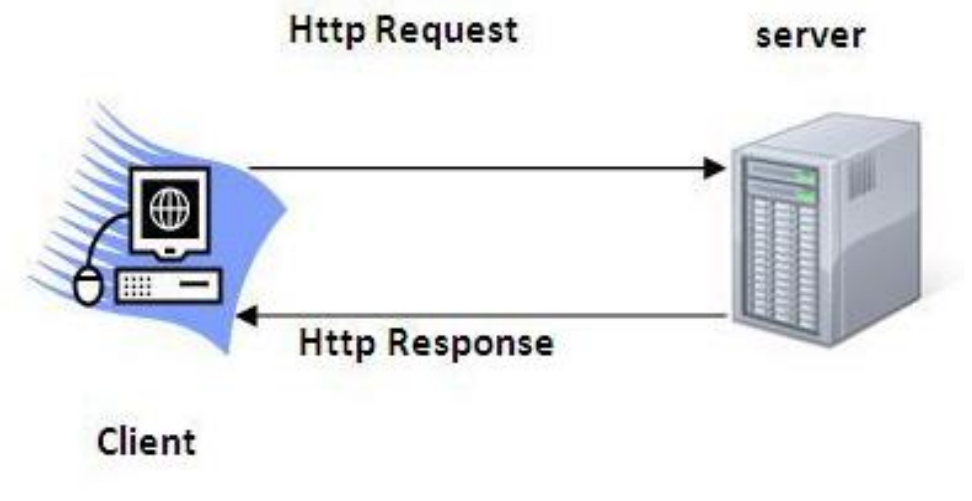
Static Website	Dynamic Website
Prebuilt content is same every time the page is loaded.	Content is generated quickly and changes regularly.
It uses the **HTML** code for developing a website.	It uses the server side languages such as **PHP(HyperText Preprocessor,SERVLET, JSP, and ASP.NET** etc. for developing a website.
It sends exactly the same response for every request.	It may generate different HTML for each of the request.
The content is only changed when someone publishes and updates the file (sends it to the web server).	The page contains "server-side" code which allows the server to generate the unique content when the page is loaded.
Flexibility is the main advantage of static website.	Content Management System (CMS) is the main advantage of dynamic website.

HTTP (Hyper Text Transfer Protocol)

- The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems.
- It is the data communication protocol used to establish communication between client and server.
- HTTP is TCP/IP based communication protocol
  - which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80. It provides the standardized way for computers to communicate with each other.

Dr. Smitha Shekar B

The Basic Characteristics of HTTP (Hyper Text Transfer Protocol)



- It is the protocol that allows web servers and browsers to exchange data over the web.
- It is a request response protocol.
- It uses the reliable TCP connections by default on TCP port 80.
- It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.

Dr. Smitha Shekar B

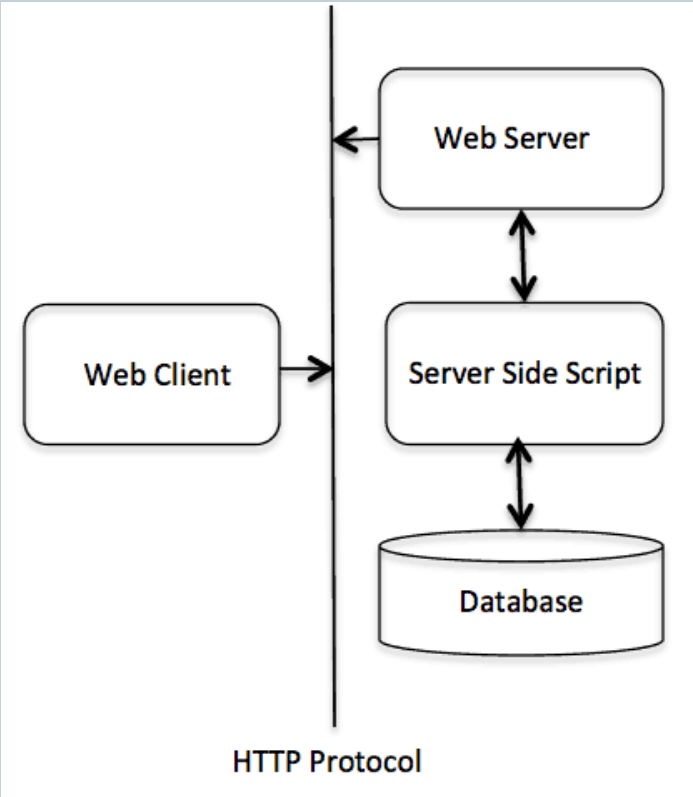
The Basic Features of HTTP (Hyper Text Transfer Protocol)

- There are three fundamental features that make the HTTP a simple and powerful protocol used for communication,
  - **HTTP is media independent:** It specifies that any type of media content can be sent by HTTP as long as both the server and the client can handle the data content.
  - **HTTP is connectionless:** It is a connectionless approach in which HTTP client i.e., a browser initiates the HTTP request and after the request is sent the client disconnects from server and waits for the response.
  - **HTTP is stateless:** The client and server are aware of each other during a current request only. Afterwards, both of them forget each other. Due to the stateless nature of protocol, neither the client nor the server can retain the information about different request across the web pages.

Dr. Smitha Shekar B

The Basic Architecture of HTTP (Hyper Text Transfer Protocol)

- The below diagram represents the basic architecture of web application and depicts where HTTP stands:



HTTP is request/response protocol which is based on client/server based architecture.

In this protocol, web browser, search engines, etc. behave as HTTP clients and the Web server like Servlet behaves as a server

...nitha Shekar B

HTTP Requests

- The request sent by the computer to a web server, contains all sorts of potentially interesting information; it is known as HTTP requests.

- The HTTP client sends the request to the server in the form of request message which includes following information,
  - The Request-line
  - The analysis of source IP address, proxy and port
  - The analysis of destination IP address, protocol, port and host
  - The Requested URI (Uniform Resource Identifier)
  - The Request method and Content
  - The User-Agent header
  - The Connection control header
  - The Cache control header

Dr. Smitha Shekar B


HTTP Requests
The Cache control header
The Request-line
The analysis of source IP address, proxy and port
The Connection control header
The User-Agent header
The analysis of destination IP address, protocol, port and host
The Request method and Content
The Requested URI (Uniform Resource Identifier)

- The HTTP request method indicates the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**.

- This method is case-sensitive and should be used in uppercase.

Dr. Smitha Shekar B

The HTTP request methods

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

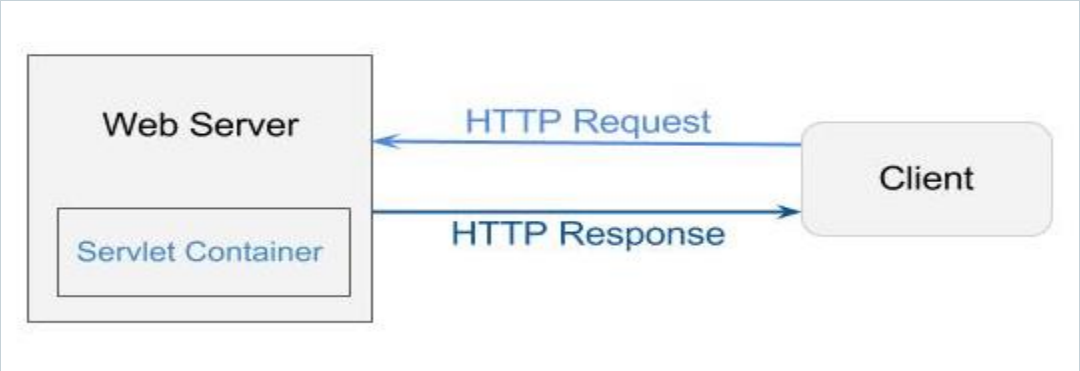
GET and POST

- Two common methods for the request-response between a server and client are,
  - **GET**- It requests the data from a specified resource
  - **POST**- It submits the processed data to a specified resource

Dr. Smitha Shekar B

Servlet Container

- It provides the runtime environment for JavaEE (J2EE) applications.
- The client/user can request only a static Web-Pages from the server. If the user wants to read the web pages as per input then the servlet container is used in java.
- The servlet container is the part of web server which can be run in a separate process.



Servlet container states in three types

- The servlet container is the part of web server which can be run in a separate process. We can classify the servlet container states in three types:
  - **Standalone:** It is typical Java-based servers in which the servlet container and the web servers are the integral part of a single program. For example:- Tomcat running by itself
  - **In-process:** It is separated from the web server, because a different program runs within the address space of the main server as a plug-in. For example:- Tomcat running inside the JBoss.
  - **Out-of-process:** The web server and servlet container are different programs which are run in a different process. For performing the communications between them, web server uses the plug-in provided by the servlet container.
- **The Servlet Container performs many operations that are given below:**
  - Life Cycle Management
  - Multithreaded support
  - Object Pooling
  - Security etc.

Dr. Smitha Shekar B

Server: Web vs. Application

- Server is a device or a computer program that accepts and responds to the request made by other program, known as client.
- It is used to manage the network resources and for running the program or software that provides services.
- There are two types of servers:
  - Web Server
  - Application Server

Dr. Smitha Shekar B

Web Server

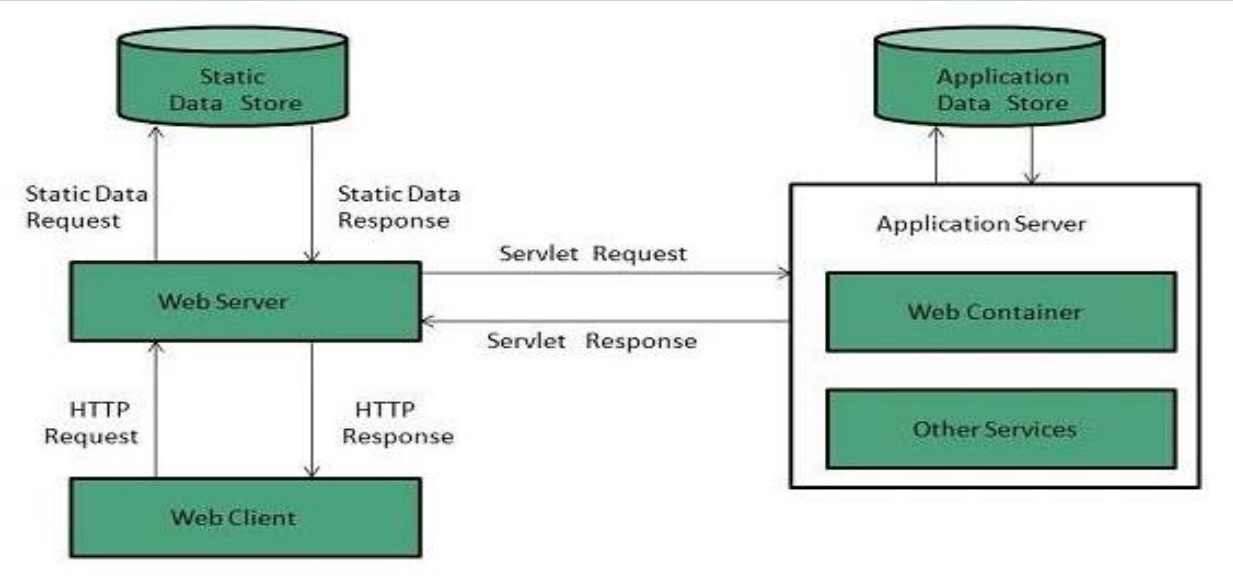
- Web server contains only web or servlet container. It can be used for servlet, jsp, struts. It can't be used for EJB.
- It is a computer where the web content can be stored. In general web server can be used to host the web sites but there also used some other web servers also such as FTP, email, storage, gaming etc.
- Examples of Web Servers are:

Apache Tomcat and **Resin**.

Dr. Smitha Shekar B

Web Server Working

- It can respond to the client request in either of the following two possible ways:
  - Generating response by using the script and communicating with database.
  - Sending file to the client associated with the requested URL.
- The block diagram representation of Web Server is shown below:



Important points

- If the requested web page at the client side is not found, then web server will sends the HTTP response: Error 404 Not found.
- When the web server searching the requested page if requested page is found then it will send to the client with an HTTP response.
- If the client requests some other resources then web server will contact to application server and data is store for constructing the HTTP response.

Dr. Smitha Shekar B

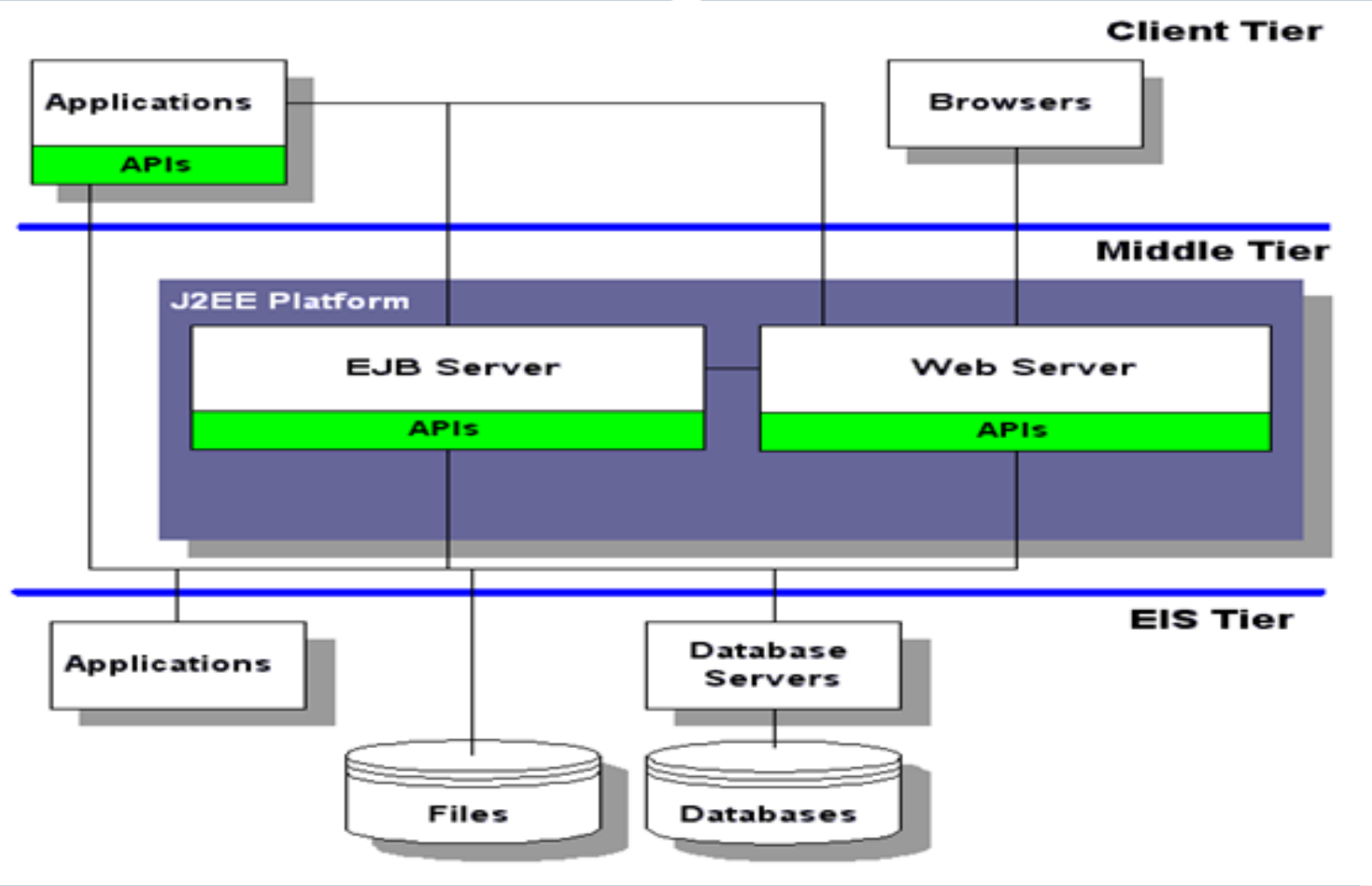
Application Server

- Application server contains Web and EJB containers.
- It can be used for servlet, jsp, struts, jsf, ejb etc.

- It is a component based product that lies in the middle-tier of a server centric architecture.
- It provides the middleware services for state maintenance and security, along with persistence and data access.
- It is a type of server designed to install, operate and host associated services and applications for the IT services, end users and organizations.

Dr. Smitha Shekar B

The block diagram representation of Application Server shown below:



Example of Application Servers

- **JBoss:** Open-source server from JBoss community.
- **Glassfish:** Provided by Sun Microsystem. Now acquired by Oracle.
- **Weblogic:** Provided by Oracle. It is more secured.
- **Websphere:** Provided by IBM.

Dr. Smitha Shekar B

Content Type

- Content Type is also known as **MIME (Multipurpose Internet Mail Extension)**Type.
- It is a **HTTP header** that provides the description about what are you sending to the browser.
- MIME is an internet standard that is used for extending the limited capabilities of email by allowing the insertion of sounds, images and text in a message.

Dr. Smitha Shekar B

The features provided by MIME to the email services are as given below:

List of Content Types

- **There are many content types**
  - The commonly used content types are given below:
  - text/html
  - text/plain
  - application/msword
  - application/vnd.ms-excel
  - application/jar
  - application/pdf
  - application/octet-stream
  - application/x-zip
  - images/jpeg
  - images/png
  - images/gif
  - audio/mp3
  - video/mp4
  - video/quicktime etc.

Dr. Smitha Shekar B

Life Cycle of a Servlet

(Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance

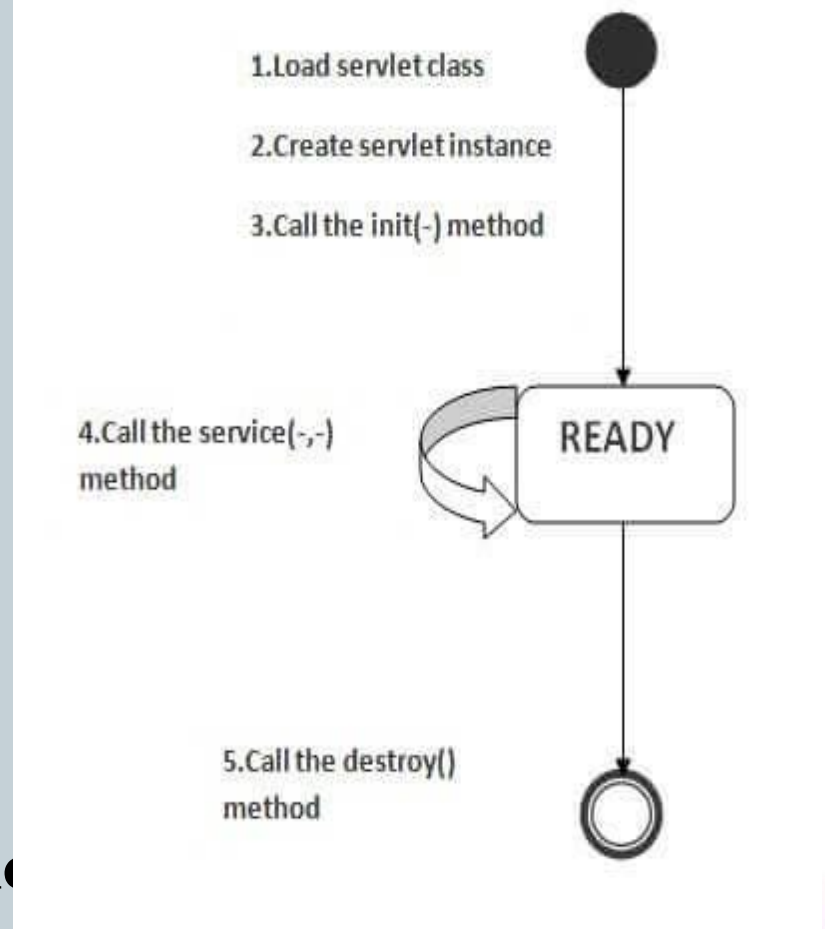
The life cycle of the servlet

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

As displayed in the above diagram, there are three states of a servlet:

new, ready and end.

- The servlet is in new state if servlet instance is created.
- After invoking the init() method, Servlet comes in the ready state.
- In the ready state, servlet performs all the tasks.
- When the web container invokes the destroy() method, it shifts to the end state.



Dr. Smitha Sh

1) **Servlet class is loaded**

- The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) **Servlet instance is created**

- The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

- The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

public void init(ServletConfig config) **throws** ServletException

Dr. Smitha Shekar B

4) service method is invoked

- The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

public void service(ServletRequest request, ServletResponse res ponse)

throws ServletException, IOException

5) destroy method is invoked

- The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

The Life Cycle of a Servlet

- Three methods are central to the life cycle of a servlet. These are **init( )**, **service( )**, and **destroy( )**. They are implemented by every servlet and are invoked at specific times by the server.
- Let us consider a typical user scenario to understand when these methods are called.
  - First, assume that a user enters a Uniform Resource Locator (URL) to a web browser. The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server.
  - Second, this HTTP request is received by the web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server.
  - Third, the server invokes the **init( )** method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may configure itself.

Dr. Smitha Shekar B

- Fourth, the server invokes the **service( )** method of the servlet. This method is called to process the HTTP request.

- You will see that it is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client.
- The servlet remains in the server's address space and is available to process any other HTTP requests received from clients.
- The **service( )** method is called for each HTTP request.
- Finally, the server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server.
- The server calls the **destroy( )** method to relinquish any resources such as file handles that are allocated for the servlet.
- Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

Dr. Smitha Shekar B

The Servlet API

Dr. Smitha Shekar B

Two packages contain the classes and interfaces that are required to build the servlets

- These are **javax.servlet** and **javax.servlet.http**.
- They constitute the core of the Servlet API.
- These packages are not part of the Java core packages. Therefore, they are not included with Java SE. Instead, they are provided by Tomcat. They are also provided by Java EE.
- The Servlet API has been in a process of ongoing development and enhancement.

Dr. Smitha Shekar B

The javax.servlet Package

- The **javax.servlet** package contains a number of interfaces and classes that establish the framework in which servlets operate.
- The most significant of these is **Servlet**. All servlets must implement this interface or extend a class that implements the interface.
- The **ServletRequest** and **ServletResponse** interfaces are also very important.

Dr. Smitha Shekar B

The following table summarizes several key interfaces that are provided in this package

Interface	Description
Servlet	Declares life cycle methods for a servlet.
ServletConfig	Allows servlets to get initialization parameters.
ServletContext	Enables servlets to log events and access information about their environment.
ServletRequest	Used to read data from a client request.
ServletResponse	Used to write data to a client response.

Dr. Smitha Shekar B

The following table summarizes the core classes that are provided in the **javax.servlet** package

Class	Description
GenericServlet	Implements the Servlet and ServletConfig interfaces.
ServletInputStream	Encapsulates an input stream for reading requests from a client.
ServletOutputStream	Encapsulates an output stream for writing responses to a client.
ServletException	Indicates a servlet error occurred.
UnavailableException	Indicates a servlet is unavailable.

Dr. Smitha Shekar B

The Servlet Interface

- All servlets must implement the **Servlet** interface. It declares the **init( )**, **service( )**, and **destroy( )** methods that are called by the server during the life cycle of a servlet.
- A method is also provided that allows a servlet to obtain any initialization parameters. The methods defined by **Servlet** are shown in Table 38-1.
- The **init( )**, **service( )**, and **destroy( )** methods are the life cycle methods of the servlet.
- These are invoked by the server. The **getServletConfig( )** method is called by the servlet to obtain initialization parameters.
- A servlet developer overrides the **getServletInfo( )** method to provide a string with useful information (for example, the version number). This method is also invoked by the server.

Dr. Smitha Shekar B

Method	Description
void destroy( )	Called when the servlet is unloaded.
ServletConfig getServletConfig( )	Returns a **ServletConfig** object that contains any initialization parameters.
String getServletInfo( )	Returns a string describing the servlet.
void init(ServletConfig *sc*) throws ServletException	Called when the servlet is initialized. Initialization parameters for the servlet can be obtained from *sc*. A **ServletException** should be thrown if the servlet cannot be initialized.
void service(ServletRequest *req*, ServletResponse *res*) throws ServletException, IOException	Called to process a request from a client. The request from the client can be read from *req*. The response to the client can be written to *res*. An exception is generated if a servlet or IO problem occurs.

Table 38-1 The Methods Defined by **Servlet**

Dr. Smitha Shekar B

The ServletConfig Interface

- The **ServletConfig** interface allows a servlet to obtain configuration data when it is loaded.
- The methods declared by this interface are summarized here:

Method	Description
ServletContext getServletContext( )	Returns the context for this servlet.
String getInitParameter(String *param*)	Returns the value of the initialization parameter named *param*.
Enumeration<String> getInitParameterNames( )	Returns an enumeration of all initialization parameter names.
String getServletName( )	Returns the name of the invoking servlet.

Dr. Smitha Shekar B

The ServletContext Interface

The **ServletContext** interface enables servlets to obtain information about their
environment. Several of its methods are summarized in Table 38-2.

Method	Description
Object getAttribute(String *attr*)	Returns the value of the server attribute named *attr*.
String getMimeType(String *file*)	Returns the MIME type of *file*.
String getRealPath(String *vpath*)	Returns the real (i.e., absolute) path that corresponds to the relative path *vpath*.
String getServerInfo( )	Returns information about the server.
void log(String *s*)	Writes *s* to the servlet log.
void log(String *s*, Throwable *e*)	Writes *s* and the stack trace for *e* to the servlet log.
void setAttribute(String *attr*, Object *val*)	Sets the attribute specified by *attr* to the value passed in *val*.

Table 38-2 Various Methods Defined by **ServletContext**

The ServletRequest Interface

The **ServletRequest** interface enables a servlet to obtain information about a client request. Several of its methods are summarized in Table 38-3.

Method	Description
Object getAttribute(String *attr*)	Returns the value of the attribute named *attr*.
String getCharacterEncoding( )	Returns the character encoding of the request.
int getContentLength( )	Returns the size of the request. The value –1 is returned if the size is unavailable.
String getContentType( )	Returns the type of the request. A **null** value is returned if the type cannot be determined.
ServletInputStream getInputStream( ) throws IOException	Returns a **ServletInputStream** that can be used to read binary data from the request. An **IllegalStateException** is thrown if **getReader( )** has been previously invoked on this object.
String getParameter(String *pname*)	Returns the value of the parameter named *pname*.
Enumeration<String> getParameterNames( )	Returns an enumeration of the parameter names for this request.
String[ ] getParameterValues(String *name*)	Returns an array containing values associated with the parameter specified by *name*.
String getProtocol( )	Returns a description of the protocol.
BufferedReader getReader( ) throws IOException	Returns a buffered reader that can be used to read text from the request. An **IllegalStateException** is thrown if **getInputStream( )** has been previously invoked on this object.
String getRemoteAddr( )	Returns the string equivalent of the client IP address.
String getRemoteHost( )	Returns the string equivalent of the client host name.
String getScheme( )	Returns the transmission scheme of the URL used for the request (for example, "http", "ftp").
String getServerName( )	Returns the name of the server.
int getServerPort( )	Returns the port number.

Table 38-3 Various Methods Defined by **ServletRequest**

The ServletResponse Interface

The **ServletResponse** interface enables a servlet to formulate a response for a client. Several of its methods are summarized in Table 38-4.

Method	Description
String getCharacterEncoding( )	Returns the character encoding for the response.
ServletOutputStream getOutputStream( ) throws IOException	Returns a **ServletOutputStream** that can be used to write binary data to the response. An **IllegalStateException** is thrown if **getWriter( )** has been previously invoked on this object.
PrintWriter getWriter( ) throws IOException	Returns a **PrintWriter** that can be used to write character data to the response. An **IllegalStateException** is thrown if **getOutputStream( )** has been previously invoked on this object.
void setContentLength(int *size*)	Sets the content length for the response to *size*.
void setContentType(String *type*)	Sets the content type for the response to *type*.

Table 38-4 Various Methods Defined by **ServletResponse**

The GenericServlet Class

- The **GenericServlet** class provides implementations of the basic life cycle methods for a servlet.

- **GenericServlet** implements the **Servlet** and **ServletConfig** interfaces.
- In addition, a method to append a string to the server log file is available. The signatures of this method are shown here:

  void log(String *s*)

  void log(String *s*, Throwable *e*)
- Here, *s* is the string to be appended to the log, and *e* is an exception that occurred.

Dr. Smitha Shekar B

The ServletInputStream Class

- The **ServletInputStream** class extends **InputStream**.
- It is implemented by the servlet container and provides an input stream that a servlet developer can use to read the data from a client request.
- In addition to the input methods inherited from **InputStream**, a method is provided to read bytes from the stream. It is shown here:
  - int readLine(byte[ ] *buffer*, int *offset*, int *size*) throws IOException
- Here, *buffer* is the array into which *size* bytes are placed starting at *offset*.
- The method returns the actual number of bytes read or –1 if an end-of-stream condition is encountered.

Dr. Smitha Shekar B

The ServletOutputStream Class

- The **ServletOutputStream** class extends **OutputStream**.
- It is implemented by the servlet container and provides an output stream that a servlet developer can use to write data to a client response.
- In addition to the output methods provided by **OutputStream**, it also defines the **print( )** and **println( )** methods, which output data to the stream.

Dr. Smitha Shekar B

The Servlet Exception Classes

- **javax.servlet** defines two exceptions.
- The first is **ServletException**, which indicates that a servlet problem has occurred.
- The second is **UnavailableException**, which extends **ServletException**.
- It indicates that a servlet is unavailable.

Dr. Smitha Shekar B

- The prorams have used the classes and interfaces defined in **javax.servlet**, such
  - as **ServletRequest**, **ServletResponse**, and **GenericServlet**,
  - to illustrate the basic functionality of servlets.

Dr. Smitha Shekar B

The javax.servlet.http Package

- However, when working with HTTP, you will normally use the interfaces and classes in **javax.servlet.http**.
- As you will see, its functionality makes it easy to build servlets that work with HTTP requests and responses.

Dr. Smitha Shekar B

The following table summarizes the interfaces

Interface	Description
HttpServletRequest	Enables servlets to read data from an HTTP request.
HttpServletResponse	Enables servlets to write data to an HTTP response.
HttpSession	Allows session data to be read and written.

The following table summarizes the classes

- The most important of these is **HttpServlet**.
- Servlet developers typically extend this class in order to process HTTP requests.

Class	Description
Cookie	Allows state information to be stored on a client machine.
HttpServlet	Provides methods to handle HTTP requests and responses.

Dr. Smitha Shekar B

The HttpServletRequest Interface

- The **HttpServletRequest** interface enables a servlet to obtain information about a client request.
- Several of its methods are shown in Table 38-5.

Dr. Smitha Shekar B

Method	Description
String getAuthType( )	Returns authentication scheme.
Cookie[ ] getCookies( )	Returns an array of the cookies in this request.
long getDateHeader(String *field*)	Returns the value of the date header field named *field*.
String getHeader(String *field*)	Returns the value of the header field named *field*.
Enumeration<String> getHeaderNames( )	Returns an enumeration of the header names.
int getIntHeader(String *field*)	Returns the **int** equivalent of the header field named *field*.
String getMethod( )	Returns the HTTP method for this request.
String getPathInfo( )	Returns any path information that is located after the servlet path and before a query string of the URL.
String getPathTranslated( )	Returns any path information that is located after the servlet path and before a query string of the URL after translating it to a real path.
String getQueryString( )	Returns any query string in the URL.
String getRemoteUser( )	Returns the name of the user who issued this request.
String getRequestedSessionId( )	Returns the ID of the session.
String getRequestURI( )	Returns the URI.
SringBuffer getRequestURL( )	Returns the URL.
String getServletPath( )	Returns that part of the URL that identifies the servlet.
HttpSession getSession( )	Returns the session for this request. If a session does not exist, one is created and then returned.
HttpSession getSession(boolean *new*)	If *new* is **true** and no session exists, creates and returns a session for this request. Otherwise, returns the existing session for this request.
boolean isRequestedSessionIdFromCookie( )	Returns **true** if a cookie contains the session ID. Otherwise, returns **false**.
boolean isRequestedSessionIdFromURL( )	Returns **true** if the URL contains the session ID. Otherwise, returns **false**.
boolean isRequestedSessionIdValid( )	Returns **true** if the requested session ID is valid in the current session context.

Table 38–5 Various Methods Defined by **HttpServletRequest**

The HttpServletResponse Interface

- The **HttpServletResponse** interface enables a servlet to formulate an HTTP response to a client.
- Several constants are defined. These correspond to the different status codes that can be assigned to an HTTP response.
- For example, **SC_OK** indicates that the HTTP request succeeded, and **SC_NOT_FOUND** indicates that the requested resource is not available.
- Several methods of this interface are summarized in Table 38-6.

Dr. Smitha Shekar B

Method	Description
void addCookie(Cookie *cookie*)	Adds *cookie* to the HTTP response.
boolean containsHeader(String *field*)	Returns **true** if the HTTP response header contains a field named *field*.
String encodeURL(String *url*)	Determines if the session ID must be encoded in the URL identified as *url*. If so, returns the modified version of *url*. Otherwise, returns *url*. All URLs generated by a servlet should be processed by this method.
String encodeRedirectURL(String *url*)	Determines if the session ID must be encoded in the URL identified as *url*. If so, returns the modified version of *url*. Otherwise, returns *url*. All URLs passed to **sendRedirect( )** should be processed by this method.
void sendError(int *c*) throws IOException	Sends the error code *c* to the client.
void sendError(int *c*, String *s*) throws IOException	Sends the error code *c* and message *s* to the client.
void sendRedirect(String *url*) throws IOException	Redirects the client to *url*.
void setDateHeader(String *field*, long *msec*)	Adds *field* to the header with date value equal to *msec* (milliseconds since midnight, January 1, 1970, GMT).
void setHeader(String *field*, String *value*)	Adds *field* to the header with value equal to *value*.
void setIntHeader(String *field*, int *value*)	Adds *field* to the header with value equal to *value*.
void setStatus(int *code*)	Sets the status code for this response to *code*.

Table 38–6 Various Methods Defined by **HttpServletResponse**

The HttpSession Interface

- The **HttpSession** interface enables a servlet to read and write the state information that is

  associated with an HTTP session.
- Several of its methods are summarized in Table 38-7.
- All of these methods throw an **IllegalStateException** if the session has already been invalidated.

Dr. Smitha Shekar B

Method	Description
Object getAttribute(String *attr*)	Returns the value associated with the name passed in *attr*. Returns **null** if *attr* is not found.
Enumeration<String> getAttributeNames( )	Returns an enumeration of the attribute names associated with the session.
long getCreationTime( )	Returns the creation time (in milliseconds since midnight, January 1, 1970, GMT) of the invoking session.
String getId( )	Returns the session ID.
long getLastAccessedTime( )	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the client last made a request on the invoking session.
void invalidate( )	Invalidates this session and removes it from the context.
boolean isNew( )	Returns **true** if the server created the session and it has not yet been accessed by the client.
void removeAttribute(String *attr*)	Removes the attribute specified by *attr* from the session.
void setAttribute(String *attr*, Object *val*)	Associates the value passed in *val* with the attribute name passed in *attr*.

Table 38-7 Various Methods Defined by **HttpSession**

The Cookie Class

- The **Cookie** class encapsulates a cookie. A *cookie* is stored on a client and contains state information. Cookies are valuable for tracking user activities.
- For example, assume that a user visits an online store. A cookie can save the user's name, address, and other information.
- The user does not need to enter this data each time he or she visits the store.
- A servlet can write a cookie to a user's machine via the **addCookie( )** method of the **HttpServletResponse** interface. The data for that cookie is then included in the header of the HTTP response that is sent to the browser.
- The names and values of cookies are stored on the user's machine. Some of the information that can be saved for each cookie includes the following:
    - The name of the cookie
    - The value of the cookie
    - The expiration date of the cookie
    - The domain and path of the cookie

Dr. Smitha Shekar B

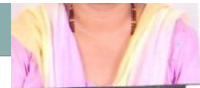
- The expiration date determines when this cookie is deleted from the user's machine.
- If an expiration date is not explicitly assigned to a cookie, it is deleted when the current browser session ends.
- The domain and path of the cookie determine when it is included in the header of an HTTP request.
- If the user enters a URL whose domain and path match these values, the cookie is then supplied to the web server. Otherwise, it is not.
- There is one constructor for **Cookie**. It has the signature shown here:

  Cookie(String *name*, String *value*)
- Here, the name and value of the cookie are supplied as arguments to the constructor.
- The methods of the **Cookie** class are summarized in Table 38-8.

Dr. Smitha Shekar B

Method	Description
Object clone( )	Returns a copy of this object.
String getComment( )	Returns the comment.
String getDomain( )	Returns the domain.
int getMaxAge( )	Returns the maximum age (in seconds).
String getName( )	Returns the name.
String getPath( )	Returns the path.
boolean getSecure( )	Returns **true** if the cookie is secure. Otherwise, returns **false**.
String getValue( )	Returns the value.
int getVersion( )	Returns the version.
boolean isHttpOnly( )	Returns **true** if the cookie has the **HttpOnly** attribute.
void setComment(String *c*)	Sets the comment to *c*.
void setDomain(String *d*)	Sets the domain to *d*.
void setHttpOnly(boolean *httpOnly*)	If *httpOnly* is **true**, then the **HttpOnly** attribute is added to the cookie. If *httpOnly* is **false**, the **HttpOnly** attribute is removed.
void setMaxAge(int *secs*)	Sets the maximum age of the cookie to *secs*. This is the number of seconds after which the cookie is deleted.
void setPath(String *p*)	Sets the path to *p*.
void setSecure(boolean *secure*)	Sets the security flag to *secure*.
void setValue(String *v*)	Sets the value to *v*.
void setVersion(int *v*)	Sets the version to *v*.

Table 38–8 The Methods Defined by **Cookie**



The HttpServlet Class

- The **HttpServlet** class extends **GenericServlet**.
- It is commonly used when developing servlets that receive and process HTTP requests.

- The methods defined by the **HttpServlet** class are summarized in Table 38-9.

Dr. Smitha Shekar B

Method	Description
void doDelete(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP DELETE request.
void doGet(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP GET request.
void doHead(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP HEAD request.
void doOptions(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP OPTIONS request.

Method	Description
void doPost(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP POST request.
void doPut(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP PUT request.
void doTrace(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Handles an HTTP TRACE request.
long getLastModified(HttpServletRequest *req*)	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified.
void service(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and response, respectively.

Table 38–9 The Methods Defined by **HttpServlet** *(continued)*

Handling HTTP Requests and Responses

- The **HttpServlet** class provides specialized methods that handle the various types of HTTP requests.
- A servlet developer typically overrides one of these methods. These methods are **doDelete( )**, **doGet( )**, **doHead( )**, **doOptions( )**, **doPost( )**, **doPut( )**, and **doTrace( )**.
- A complete description of the different types of HTTP requests is beyond the scope of this book.
- However, the GET and POST requests are commonly used when handling form input. Therefore, this section presents examples of these cases.

Dr. Smitha Shekar B

Using Cookies

- Now, let's develop a servlet that illustrates how to use cookies.
- The servlet is invoked when a form on a web page is submitted. The example contains three files as summarized here:

File	Description
AddCookie.html	Allows a user to specify a value for the cookie named **MyCookie**.
AddCookieServlet.java	Processes the submission of **AddCookie.html**.
GetCookiesServlet.java	Displays cookie values.

Dr. Smitha Shekar B

Session Tracking

- HTTP is a stateless protocol. Each request is independent of the previous one.
- However, in some applications, it is necessary to save state information so that information can be collected from several interactions between a browser and a server. Sessions provide such a mechanism.
- A session can be created via the **getSession( )** method of **HttpServletRequest**. An **HttpSession** object is returned.
- This object can store a set of bindings that associate names with objects. The **setAttribute( )**, **getAttribute( )**, **getAttributeNames( )**, and **removeAttribute( )** methods of **HttpSession** manage these bindings.
- Session state is shared by all servlets that are associated with a client.

Dr. Smitha Shekar B

Annotations vs. Deployment Descriptor

- we used deployment descriptor (web.xml file) to configure our servlets. Since Servlet 3.0 you can use the @WebServlet annotation instead. This annotation allows you to set several attributes to the servlet like name, URL and more.

- What's the difference?

  - Well, obviously the deployment descriptor is a separate file where you set configuration values in XML format, where the annotation is directly embedded in your source code. Use annotations if you prefer to have code and configuration at the same place for better readability.
  - Deployment descriptors are the exact opposite – you separate code and configuration. This way you do not need to recompile the entire project if you want to change a single configuration value.

- For many Java Enterprise components there are both versions available – annotation or descriptor. But others can be configured either only with annotations or via the deployment descriptor. In case of Servlets you can choose one or the other method.

Dr. Smitha Shekar B

- **The steps, you need to follow to create the servlet example.**

- Create a Dynamic web project
  - **click on File Menu -> New -> Project..-> Web -> dynamic web project -> write your project name e.g. first -> Finish**.
- create a servlet
  - For creating a servlet, **explore the project by clicking the + icon -> explore the Java Resources -> right click on src -> New -> servlet -> write your servlet name e.g. Hello -> uncheck all the checkboxes except doGet() -> next -> Finish**.
- add servlet-api.jar file
  - For adding a jar file, **right click on your project -> Build Path -> Configure Build Path -> click on Libraries tab in Java Build Path -> click on Add External JARs button -> select the servlet-api.jar file under tomcat/lib -> ok.**
- Run the servlet
  - For starting the server and deploying the project in one step, **Right click on your project -> Run As -> Run on Server -> choose tomcat server -> next -> addAll -> finish.**

Dr. Smitha Shekar B

Servlet API

- The **javax.servlet** and **javax.servlet.http** packages represent interfaces and classes for Servlet API.
- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

Dr. Smitha Shekar B

javax.servlet package

- The *javax.servlet package* contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment
- The *Servlet* interface is the central abstraction of the servlet API.
- All servlets implement this interface either directly, or more commonly, by extending a class that implements the interface.

Dr. Smitha Shekar B

- The two classes in the servlet API that implement the *Servlet* interface are *GenericServlet* and *HttpServlet* .
- For most purposes, developers will extend *HttpServlet* to implement their servlets while implementing web applications employing the HTTP protocol.
- The basic *Servlet* interface defines a *service* method for handling client requests.
  - This method is called for each request that the servlet container routes to an instance of a servlet.

Dr. Smitha Shekar B

THANK YOU