UNIT 2 BASICS OF SERVLET

Structure

2.0 INTRODUCTION

In this new era of technology, the Internet is just like a bottomless ocean where you will find anything from a website to a web application. As you know very well, a website is a collection of related web pages that contains static (HTML pages, images, graphics) as well as dynamic files, whereas a Web application is a piece of software with dynamic functionality on the server. Google, Facebook, Twitter are some popular examples of web applications.

Today, you are aware of the need of creating dynamic web pages. For this, you can use server-side programming languages such as Java Servlets, Java Server Pages, ASP.Net and PHP etc. This Unit and the next unit will provide you with more details on Java Servlets. Next th Java Server Pages will be discussed in unit 4 of this block.

You are also aware of core Java concepts and compiling and running of java classes. In this unit, you will learn the basics of Servlet, Servlet-API and life-cycle of the servlet. Servlets are java classes that run on the Java-enabled web server and are widely used for web processing as well as are capable of handling complex requests obtained from the web server. Servlets need some protocol and methods for communicating with the server. This unit will also cover HTTP protocol and methods and guides you on how to write, run and deploy a servlet.

2.1 OBJECTIVES

After completion of this unit, you will be able to:

- Describe basics of Servlet,
- Difference between HTTP request and HTTP response,
- Use of HTTP methods,

- Use different classes and interface included in Servlet API and some of its methods,
- Describe how servlet container maintains the Servlet Life Cycle, and
- Write simple servlet for the web applications.

2.2 INTRODUCTION OF SERVLETS

You all are aware of java classes. A servlet is also a java class that is descended from the class javax.servlet.http.HttpServlet. So, servlet is a java class, but not all java classes are servlets. In this section, you will learn more about Java Servlet.

Java Servlets are small, platform-independent java programs that run on the Java-enabled web server. It extends from a Java class or rather interface and requires you to implement certain methods so that web container or servlet container (Tomcat, etc.) is able to send execution to your servlet. Basically it creates a class that extends either GenericServlet or HttpServlet, overriding the appropriate methods, so it handles requests.

Web containers job is to handle the request of the web server; process these requests, produce the response and send it back to the web server. Servlets are executed within the address space of a Web Server.

As Servlet technology is based on java language, it is robust and scalable . It is used to create a program for developing web applications. These programs reside on server side. Java Servlet is the foundation technology for Java server-side programming. JSP (Java Server Pages), JSF (Java Server Faces), Struts, Spring, Hibernate and others are extensions of the servlet technology.

Servlet extends the capabilities of web servers that host applications accessed by means of a request-response programming model. For developing web applications, Java Servlet technology defines HTTP-specific servlet classes. A HTTP Servlet runs under the HTTP protocol. This protocol is an asymmetrical request-response protocol where the client sends a request message to the server, and the server returns a response for requested data. You will get more details on HTTP protocol and methods in the next section of this unit.

There are many interfaces and classes in the Servlet API for creating servlets, such as GenericServlet, HttpServlet, ServletRequest, ServletResponse, and Servlet, which will be discussed in section 2.4 of this unit.

A java servlet program has a life cycle that defines:

i. how the servlet is loaded and initiated,

ii. how a Servlet receives and responds to requests, and

iii. how it is taken out of service.

The Servlet life cycle will be discussed in detail in section 2.5 of this unit.

2.3 HTTP PROTOCOL AND HTTP METHODS

In the previous section, you have been explained about the servlets which are a server-side scripting language. It follows client-server architecture in which client (browser) sends data to server and it requires some protocol (set of rules) for communication between the two. In this section, you will learn some of the basic protocol and methods that make this communication possible.

Hypertext Transfer Protocol (HTTP) is specially meant to communicate between Client and Server using Web (or Internet). The HTTP is a stateless protocol, it supports only one request per connection. It can be further simplified as; HTTP client connects to the server to send one request only and then it it is disconnected. This mechanism facilitates connecting more users to a given server over a period of time.

To run web applications, web browsers communicate with the web servers using the HTTP. When you type any URL in the browser, the browser sends an HTTP Request to the server. The server receives the request and sends back the requested data to the browser. In the following sections, you will learn more about the HTTP Protocol and HTTP methods.

2.3.1 HTTP Protocol Overview

HTTP Protocol is a network protocol used for transferring files on the Internet . HTTP is the primary protocol used for most web applications on the Internet. Whether the application are written in Java, ASP.Net or PHP; all web applications use HTTP. This protocol is the foundation of any data exchange on the Web. HTTP works based on request-response model between a client and the server. An HTTP client sends a request message to a HTTP server. The server returns a response message to the client. As HTTP is a stateless protocol, the current request does not know what has been done in the previous requests. For communication between the different web pages, you need to establish sessions or create cookies. These will be discussed in Unit 3 of this block.

HTTP is also called connectionless protocol, which means that the client establishes a connection with the server before sending a request, and the server sends back response to the client over this connection only. When a response is delivered, the connection between the client and the server is destroyed. If the same client wants to connect to same server again, the client should establish a new connection. So, HTTP is designed as connectionless for the above said reason; the server should share resources equally to the clients who exist all over the world. If one client is connected all the time with the server then the server cannot allocate the time to other clients.

HTTP was initially developed by Tim Berners-Lee and his team in early 1990. HTTP/1.0 defines the basic protocol with some methods such as GET, POST and HEAD, and this version did not support informational status codes. The HTTP/1.1 defines seven HTTP methods GET, POST, HEAD, OPTIONS, TRACE, PUT, DELETE, and also add many headers and facilities to the original HTTP/1.0.
Subsequently, in HTTP/2.0 there has been a major revision of the HTTP network protocol. HTTP/2.0 is the first new version of HTTP since HTTP 1.1. HTTP/3 is the third major version of the HTTP used to exchange information on the World Wide Web.When a web page link is clicked on a web page to look for a search or submit a form from your browser, the browser translates your requested URL into a **request message** according to the specified protocol and sends it to the HTTP server. The

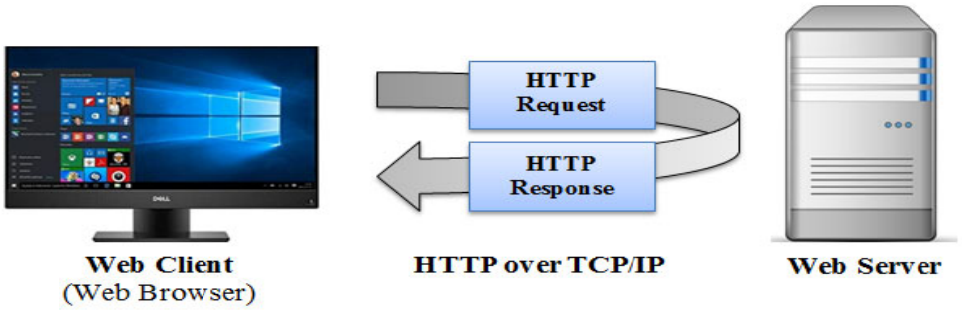


Figure 1: Process of communication between Web Client and Web Server

HTTP Server (Web Server) interprets the request message received, maps the request into a *program* kept in the server, executes the program and returns you an appropriate **response message.** in the form of the resource has been requested or as an error message. In figure 1 this request and response process is displayed.

HTTP messages manage the information exchange between a server and a client. There are two types of HTTP messages: **HTTP requests** sent by the client to the server, and **HTTP responses**, the answer from the server. Both the messages are described below:

2.3.2 HTTP Requests

Whenever a client sends a message to a server it is an HTTP request. As you have learned until now, when the client sends a request to the server, the server returns a response to the client. Web Client sends a request specifying one of the seven HTTP request methods (you can read more about these methods in the next section), the location of the resource to be invoked, protocol version, a set of optional headers and an optional message body.

In a simple terminology, as a client, when you type a URL of the website in the browser, for example, you want to download your Hall-ticket from the IGNOU website and open the website by typing the URL as http://www.ignou.ac.in/studentzone/hallticket.jsp; a request is initiated from your computer to the web server. This process sets up a connection from your computer to the host computer where the IGNOU web server is hosted. The domain name from your URL is converted into a machine-readable state known as an IP address by the Domain Name System (DNS) server during the connection setup. The rest part of the URL are path i.e. /studentzone/ and resource file (hallticket.jsp). The client computer, after knowing the IP address of web server, path, resource file, your request goes to the web server using HTTP methods such as GET. Some more technical operations are needed to establish a connection between client and server. A detailed discussion on this is beyond the scope of this unit.

For the requested IGNOU page, your request may look like the following line: method name, path, resource name, and protocol version.

```
GET /studentzone/hallticket.jsp  HTTP/1.1
```

You can also find these details using the following example. The 'RequestServlet' example illustrates some of the resources available in the request object. The detailed running procedure is given in section 2.7 of this unit.

Source Code 'RequestServlet' example

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class RequestServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,   HttpServletResponse response)
  throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    java.util.Date date = new java.util.Date();
    out.println("Current Date & Time: " +date.toString());
    out.println("<h3>Request Information Example</h3>");
    out.println("Method: " + request.getMethod()+"<br>");
```

```
    out.println("Request URI: " + request.getRequestURI()+"<br>");
    out.println("Protocol: " + request.getProtocol()+"<br>");
    out.println("PathInfo: " + request.getPathInfo()+"<br>");
    out.println("Remote Address: " + request.getRemoteAddr());
  }
  //doGet
}
```

The key methods included in the above program are described in the table below:

Table 1: Some HttpServletRequest Methods

Method	Description
getMethod()	Returns the HTTP request method
getRequestURI()	Returns the complete requested URI
getProtocol()	Returns the protocol used by the browser
getPathInfo()	only returns the path passed to the servlet
getRemoteAddr()	Returns the IP address of the client

The output of this Servlet should look like this (figure-2):

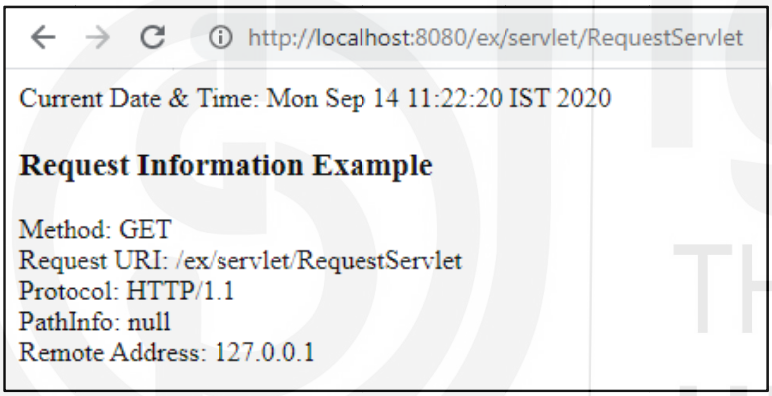


Figure 2: Output Screen for 'RequestServlet' example

As you know that the request message shows the method, Request URI, Protocol, PathInfo and the remote address of client machine. Here, the method is GET, Request URI is the path of your 'webapps' and the Protocol version is HTTP 1.1 (it may be different on your server). This servlet is running on the local machine. So, the address is 127.0.0.1. These details depend on your web server.

You can also check your logs under the local web server. For example, the above program shows the following line in the logs folder under the Tomcat Server:

127.0.0.1 - - [14/Sep/2020:11:22:20 +0530] "GET /ex/servlet/RequestServlet HTTP/1.1"

2.3.3 HTTP Response

Once a server receives a request, it interprets the request message and responds with an HTTP response message. The server sends back a response to the client containing the version of HTTP we are using, a response or status code, a description of the response code, a set of optional headers, and an optional message body.

HTTP response status codes is used to indicate whether a specific HTTP request has been successfully completed or not. Responses are grouped in five classes:

Table 1: HTTP Response Classes

S.No.	Status Code	Description
1	Informational responses (100–199)	Request has been received and the process is continuing.
2	Successful responses (200–299)	Action was successfully received, understood, and accepted.
3	Redirects (300–399)	Action must be taken to complete the request.
4	Client errors (400–499)	Request contains incorrect syntax or cannot be fulfilled - It means the server failed to fulfil an apparently valid request.
5	Server errors (500–599)	Server failed to fulfil a valid request.

It is not necessary for HTTP applications to understand the meaning of all registered status codes. More details of the status code can be explored at: https://www.w3.org

For example, if the response line of HTTP applications is:

HTTP/1.1 200 222

Then it can be interpreted as: HTTP/1.1 is the HTTP version; The HTTP Status 200 indicates the successful processing of the request on the server.

2.3.4 HTTP Method

In the previous section, you read the term method in the context of the request message. In this section, you will learn some frequently used methods for getting response from the server. You are already aware of the HTTP request which is a message that a client sends to a server. To send these requests, clients can use various HTTP methods. These request methods are case-sensitive and should always be noted in upper case. There are various HTTP request methods such as GET, POST, PUT, HEAD, DELETE, OPTIONS and PATCH and each one is assigned a specific purpose.

GET Method

GET method is used to retrieve requested data from a specified resource in the server. GET is one of the most popular and commonly used HTTP request methods. GET request is only used to request data (not modify). It means that the GET method is used to retrieve information and does not make any change in the server. These types of methods are called ‘safe’ methods.

The following simple HTML programme will demonstrate you how the ‘GET’ method will work:

```
<html><body>
<form name ="InfoForm"  action="Info.jsp" method="GET">
  <div>  <label >Name</label>  <input name="name1"  value=""> </div>
  <div>  <label >Course</label>   <input name="course" value="">   </div>
  <div>  <input type="Submit">   </div>
</form>
</body>
<html>
```

The above programme consists of two input boxes i.e. name and course. When you run the above program by using the procedure defined in the section 2.7, the result screen will display as shown in the figure 3.

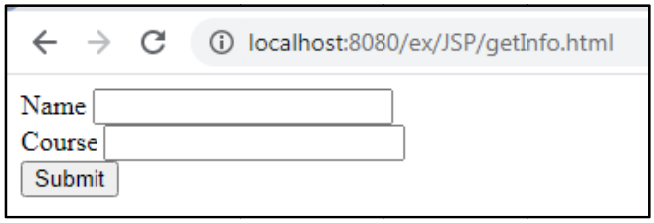
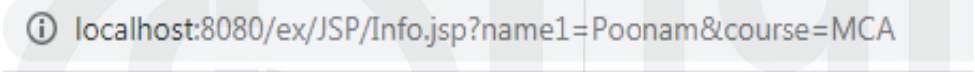


Figure 3: Simple Form for 'GET' method Example

When you put some values in the input boxes i.e. name & course, and press the 'Submit' button as displayed in Figure 3, then action control will transfer on the following HTML code line in the form:

```
<form name ="InfoForm" action="Info.jsp" method="GET">
```

So, action will be taken by the resource file:Info.jsp and the method will be GET. Here, GET method is used to retrieve values from the server. Note that the query string (name/value pairs) is sent in the URL of a GET request. After submitting the form, the name/value is visible as like the following:



Here, 'localhost:8080' indicates that you are referring to the local web server on this machine at a 8080 port number. The web page uses the path /ex/JSP of the web server and accesses the resource file 'Info.jsp' and after the resource file name, there is a text ?name1=Poonam&course=MCA which indicates passing additional information to the server in the form of the parameter. In this case, name1 is defined as the name of 'Name' input box (<input name="name1" >) and value of this box is 'Poonam'. In the similar manner, course (<input name="course" >) is the name of 'Course' input box and value of this box is MCA. This example shows how the GET method works. You can use this type of query string to find the data from the database or simply get the parameter value to display on the web page. In the above example, server may use the operation which is defined in the 'Info.jsp' file to display data.

HEAD Method

The HEAD method is similar to GET method but doesn't have a message-body in the response. In a web application, the server replies with a response line and headers section only.

POST Method

Another popular HTTP request method is POST. In web communication, this method is used to send data to a server to create or update a resource. The information submitted using POST request method to the server is archived in the request body of the HTTP request. The information /data sent through the POST method will not be visible in the URL as parameters are not sent along with the URL.

For example, you can use the same form defined in GET method, only change the name of method as POST. So, form contains the following HTML code line for <form> element:

```
<form name ="InfoForm" action="Info.jsp" method="POST">
```

When you submit the form by pressing 'Submit' button, information must be sent to the server. In this example, parameters like name and course will be put inside the request body. Like the GET method, parameter and their values are not visible in the URL.

Both the Get and Post method of HTTP protocol is frequently used methods in web programming. Using a GET request, information is sent to the server via URL parameters. On the other hand with a POST method , some additional data is also supplied from the client to the server in the message body of the HTTP request. An advantage of the POST over the GET request is that it is more secure - it cannot be bookmarked, and it is difficult to hack. Also it is not stored in the browser history. This method is, therefore, more commonly used when sensitive information is involved.

There are several other request methods. You may refer to more of these methods from the 'Further Readings' section at the end of this unit.

HTTP and Java

Servlets can be seen as java components that can respond to an HTTP request. There are methods in Java Servlets corresponding to each of the HTTP methods. The seven HTTP methods map on to seven servlet methods of the same name with a "do" in front of the method. For example, GET maps on to doGet(). Mostly you can use either GET or POST method while developing a web page.

☞ Check Your Progress 1

1. What is a Servlet? Define the workflow of a servlet,

   ---
2. What are the basic Features of HTTP?

   ---
3. What is HTTP Message? What are request and response in the context of HTTP?

   ---
4. How is GET method different to POST method?

2.4 SERVLET ARCHITECTURE

Servlets are the Java program that runs on the Java-enabled web server. The client or web browser sends the HTTP request to the web server. The web server receives the request and passes the request to the servlet container and subsequently the servlet container forwards this request to the corresponding servlet. The servlet processes the request and generates the response in the form of output. This process may require communication to a database and may also invoke a web service, or computing the response directly. After processing, the servlet builds the response object and sends it back to the Web Container.

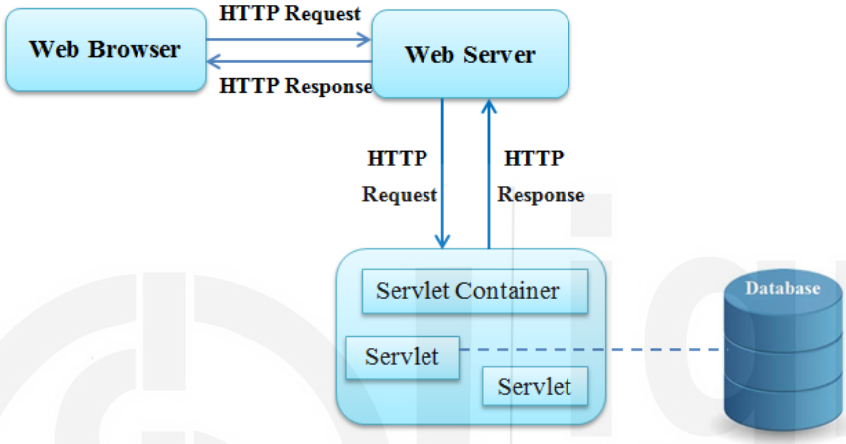


Figure 4: Servlet Architecture

The servlet container sends the response back to the web server; the web server sends the response back to the browser and the browser displays it on the screen. The diagram shows the execution of servlet in Figure 4.

You need to use Servlet API to create servlets. These APIs allow developer to build programs that can run with a web server. The Servlet APIs (Application Programming Interface) contains two packages: javax.servlet and javax.servlet.http. These two packages make up the servlet architecture. The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. The javax.servlet package contains generic interfaces and classes that are implemented and extended by all the servlet. The javax.servlet.http package contains a number of classes and interfaces that are used to create HTTP protocol-specific Servlets.

GenericServlet Class

The javax.servlet package provides an abstract class called GenericServlet that implements **Servlet**, **ServletConfig** and **java.io.Serializable** interfaces. A servlet can directly extend it. The subclass of GenericServlet is HttpServlet. It can handle any type of request. You may create a generic servlet by inheriting the GenericServlet class and implementing the service( ) method. The prototype of service() method is defined as follows:

```
public abstract void service(ServletRequest req, ServletResponse res)
throws ServletException, IOException;
```

The two objects of service() method are ServletRequest and ServletResponse. The ServletRequest object holds the information that is being sent to the servlet and the ServletResponse object is to assist a servlet in sending a response back to the client.

The javax.servlet.ServletException is a general exception that a Servlet can throw. IOException is a checked exception and is thrown when there is any input/output file operation issues while application is performing certain tasks accessing the files.

Servlet Interface

All servlets must implement the Servlet interface either directly or indirectly. Java servlets class does not have a main() method, so all servlets must implement the javax.servlet.Servlet interface. It defines five methods, including three life-cycle methods. You may read the life-cycle methods such as init(), service() and destroy() in the subsequent section 2.5 and other two methods are getServletConfig() and getServletInfo().

ServletConfig Interface

An object of ServletConfig is created by the servlet container for each servlet. This object is used to pass configuration related information to a servlet during start up. The getServletConfig() method is used to return the object of ServletConfig. It contains name/value pairs of initialization parameters for the servlet. It defines four methods such as getServletContext(), getServletName(), getInitParameter() and getInitParameterNames() for accessing this information.

Serializable interface

The Java Serializable interface (java.io.Serializable) is a marker interface. It means that it contains no methods. Therefore, a class implementing Serializable does not have to implement any specific methods.

HttpServlet Class

The javax.servlet.http package contains a number of classes and interfaces that are used to create HTTP protocol-specific Servlets. The abstract class HttpServlet is a base class for user defined HTTP Servlets which provide methods. The basic methods such as doGet and doPost are for handling HTTP-specific services. When you are developing your own servlets then you can use HttpServlet class which is extended from GenericServlet.

```
public abstract class HttpServlet
```

Unlike with GenericServlet, when you extend HttpServlet, you can skip implement ing the service() method. The HttpServlet class has already implemented the service() method. The following are the prototype of the service() method:

```
protected void service(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
```

When the HttpServlet.service() method is invoked, it first reads the method type stored in the request, and on that basis it determines the method to be invoked. For example if the method type is GET, it will call doGet() method, and if the method type is POST, it will call doPost() method. These methods have the same parameter as the service() method.

There are many interfaces in javax.servlet.http package but the three important interfaces HttpServletRequest, HttpServletResponse and HttpSession are described below:

Interface HttpServletRequest

The HttpServletRequest interface captures the functionality for a request object that is passed to a HTTP servlet. It provides access to an input stream and allows the servlet

to read data from the client. The HttpServletRequest interface extends the ServletRequest interface.

```
public interface HttpServletRequest extends ServletRequest
```

The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost etc). The HttpServletRequest interface has many methods. Some of the commonly used methods of HttpServletRequest are:

1. **getParameter()**

   It returns the value associated with a parameter sent to the servlet as a part of a GET or POST request. The name argument represents the parameter name.

   ```
   public String getParameter(String name)
   ```

2. **getQueryString()**

   It returns the query string that is contained in the request URL if any. This method returns null if the URL does not have a query string. A query string is defined as any information following a ? character in the URL.

   ```
   public String  getQueryString()
   ```

3. **getCookies()**

   The getCookies() method returns an array of Cookie objects found in the client request. This method does not take any parameters and throws no exceptions. Cookies are used to uniquely identify clients to servlet. In case of no cookies in the request, an empty array is returned.

   ```
   public Cookie[]  getCookies()
   ```

4. **getHeader()**

   It returns the value of the specified request header as a String. In case if the request did not include a header of the specified name, this method will return null. If there are multiple headers with the same name, this method returns the first header in the request. The header name is case insensitive.

   ```
   public String getHeader(String name)
   ```

5. **getMethod**

   It returns the name of the HTTP method used to make the request. Typical return values are ‘GET’, ‘POST’, or ‘PUT’

   ```
   public String getMethod()
   ```

6. **getSession()**

   It returns the current session associated with the request or if the request does not have a session, creates one.

public HttpSession getSession(boolean create)

public HttpSession getSession()

Interface HttpServletResponse

The HttpServletResponse extends the ServletResponse interface to provide HTTP specific functionality while sending a response to the client. It provides access to an output stream and allows the servlet to send data to the client.

public interface HttpServletResponse extends ServletResponse

This interface has many methods; some are described in the following sections.

1. **getWriter() Method**

   It obtains a character-based output stream that enables text data to be sent to the client.

   public PrintWriter getWriter()

2. **addCookie() Method**

   This method is used to add a Cookie to the HttpServletResponse object. It returns no value and throws no exception. This method can be called many times to set more than one cookie.

   public void addCookie(Cookie cookie)

3. **addHeader() Method**

   This method is used to add a response header with the given name and value.

   public void addHeader(String name, String value)

4. **getOutputStream() Method**

   It obtains a byte-based output stream that enables binary data to be sent to the client.

   public ServletOutputStream getOutputStream()

5. **sendError() Method**

   This method sends an error to the client in the response object. The error consists of only the 'int' status code and returns no value.

   public void sendError(int statusCode) throws IOException

   The following sendError() method sends an error to the client in the response object. The error consists of an 'int' status code and a String message. It returns no value.

   public void sendError(int statusCode, String message) throws IOException

6. **sendRedirect() Method**

   This method redirects the client to the passed-in URL, which must be an absolute URL. It returns no value.

public void sendRedirect(String uri) throws IOException

Note- You can reference the Servlet-API from your Tomcat installation at https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html

Interface HttpSession

This is an important interface used to identify a user across more than one page request or visits to a website and stores information about that user. Servlet Container uses this interface for the creation of session between HTTP client and HTTP server. Using HttpSession, you can maintain state (data) between transactions. The HttpServletRequest interface provides two methods such as getSession() and getSession(boolean create) to get the object of HttpSession. Both the methods getSession() and getSession(boolean create) are explained in the "HttpServletRequest interface' section of this Unit. The signature of this interface is as under:

public interface HttpSession

The commonly used methods of HttpSession interface are defined below. You can find an example of these methods in the next unit of this course.

1. **getId() Method**

   The getId() method returns a string containing a unique identifier assigned to the current HttpSession.

   public String getId()

2. **getCreationTime() Method**

   The getCreationTime() method returns the time in which the session was created.

   public long getCreationTime()

3. **getLastAccessedTime() Method**

   This method returns the last time the client sent a request associated with this session object.

   public long getLastAccessedTime()

4. **getAttribute() Method**

   This method is used to return the value of given parameter from the session.

   public Object getAttribute(String name)

5. **setAttribute() Method**

   This method is used to set the attribute in session.

   public void setAttribute(String name, String value)

6. **invalidate() Method**

   This method is used to destroy the session.

   public void invalidate()

   For example, session.invalidate();

Note: You can find more methods of HttpSession interface in the following link:
https://docs.oracle.com/javaee/5/api/javax/servlet/http/HttpSession.html

☞ Check Your Progress 2

1. What is Servlet interface and what is the use of it?

   ---
2. Write the difference between GenericServlet and HTTPServlet?

   ---
3. What is ServletConfig?

   ---
4. What is ServletContext?

2.5 SERVLET LIFE CYCLE

A Java Servlet has a life cycle that defines the steps in the whole processing of a servlet. This includes:

1. The Servlet loading and initialization,

2.How servlet receives and responds to requests, and

3.How a servlet is taken out of service.

The servlet life cycle is very simple object oriented design. A servlet life cycle includes the entire process from its creation to the destruction. A servlet is first initialized and then it serves to zero or more service requests until the service ends and shuts down. Servlet is loaded only once, and it stays resident in the memory till it is servicing a request. **Servlet container manages the** entire life cycle of a Servlet, using the **javax.servlet.Servlet** interface. Every servlet you write must implement the **javax.servlet.Servlet** interface either directly or indirectly. The Servlet interface defines all the methods of servlet life cycle such as init( ), service( ) and destroy( ).

Now let us discuss the life cycle methods in detail.

2.5.1 The init() Method

The init() method is where the servlet's life cycle begins. The servlet container calls this method after the servlet class has been instantiated. This method is called exactly once by the servlet container to indicate to the servlet that the servlet is being placed into service. In this method, servlet creates and initializes the resources, including the data members that it will be using while handling requests.

The signature of this method is defined as follows:

public void init (ServletConfig config) throws ServletException

This method takes a ServletConfig object as a parameter. The ServletConfig object contains the servlet's configuration and initialization parameters. This init() method can also throw a ServletException. The ServletException is the most important exception in servlet programming. If the servlet cannot initialize the resource to handle the request, then the method will throw an execution.

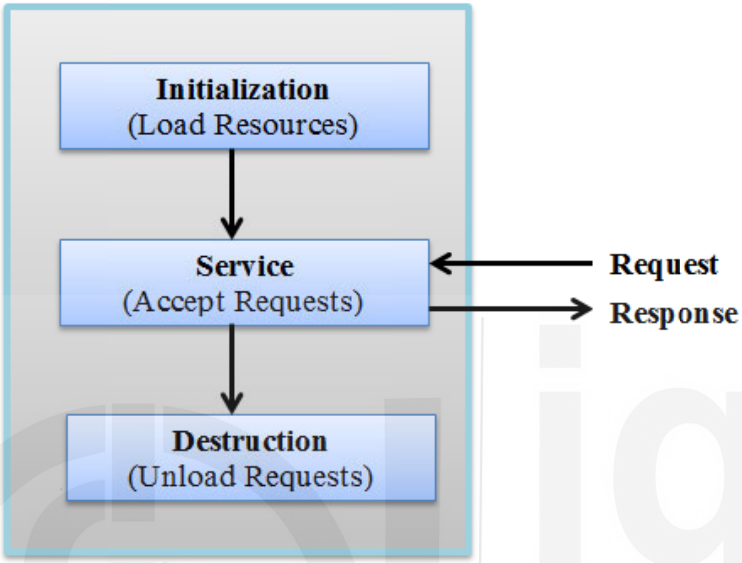


Figure 5: Servlet life cycle

2.5.2 The service() Method

The service() method is used to handle incoming requests and generate a response. This method is called by the servlet container and cannot start servicing the request until the servlet's init() method has been executed. The most common use of this method is in the HttpServlet class. The HttpServlet class provides http specific methods such as doGet, doPost, doHead, doTrace etc. In order to generate a response you should override the doGet or doPost methods as per your requirement.

This method has the following signature:

public void service(ServletRequest req, ServletResponse res) throws ServletException, java.io.IOException

This method implements a request and response paradigm. The servlet container passes two objects, ServletRequest object and ServletResponse object. The ServletRequest object contains the client's request and ServletResponse object contains the servlet's response. Both objects are important because they enable you to write code that determines how the servlet fulfils the client request. The service() method may throw the ServletException and IOException while processing the request. This method throws a ServletException if any exception occurs that interferes with the servlet's normal operation and also throw a java.io.IOException if an input or output exception occurs during the execution of service() method.

2.5.3 The destroy() Method

The destroy() method is called only once when all the processing of the servlet is over and it is at the end of its life cycle. When your application is stopped or Servlet Container shuts down, this method will be called to close down those shared resources and other clean up required before the servlet is taken out of service. This is the place where any resources that were created in init() method will be cleared up.

The signature of this method is defined as follows:

```
public void destroy( )
```

2.6 CREATING A SERVLET

In this section, we will try to create a basic servlet. This will help you in becoming familiar with the basic parts of the servlets. Here is a simple servlet program code for printing a text message such as 'Welcome to the IGNOU Family!'.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class WelcomeServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,   HttpServletResponse response)
  throws ServletException, IOException
  {
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>Servlet Testing</TITLE>");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println("Welcome to the IGNOU Family!!");
    out.println("</BODY></HTML>");
  }
}
```

Write the above code in a notepad and save it as WelcomeServlet.java on your PC. You will find running procedure for the servlet in the next section.

2.7 RUNNING AND DEPLOYMENT OF SERVLET

In this section, you will find the installation process of Apache's Tomcat. You can use it for running and deployment of your Servlets as well as JSP programs. You will study JSP in Unit 4 of this course. It is the best practice to use IDE for web development. Here, you can go through the **simple steps for running your servlet and JSP program using Apache Tomcat** without any IDE. Apache Tomcat is an open source web server for testing servlets and JSP programs. You can use Notepad to write your program. Also, there are some open source Integrated Development Environment (IDE) for developing J2EE applications. NetBeans IDE supports the development of all Java application types, including Java SE (including JavaFX), and J2EE. More about the IDEs you will learn in MCSL-222 course.

The following steps are based on Windows Operating System(Windows XP/8/10):

Step 1 - Download and install Java Development Kit.

Step 2 - Download the latest version of **Tomcat Server** and install it on your machine.

Step 3 - After successful installation of Java and Tomcat, you can set the environment variables by using the Environment Variables option. For this, do right click on System icon → properties →Advanced System Setting → environment variables. Now, click on 'New' button and enter the variable name as JAVA_HOME and in variable value write the path of Java installation directory and click on the 'ok' button. The following screen comes after the selection of the step3.

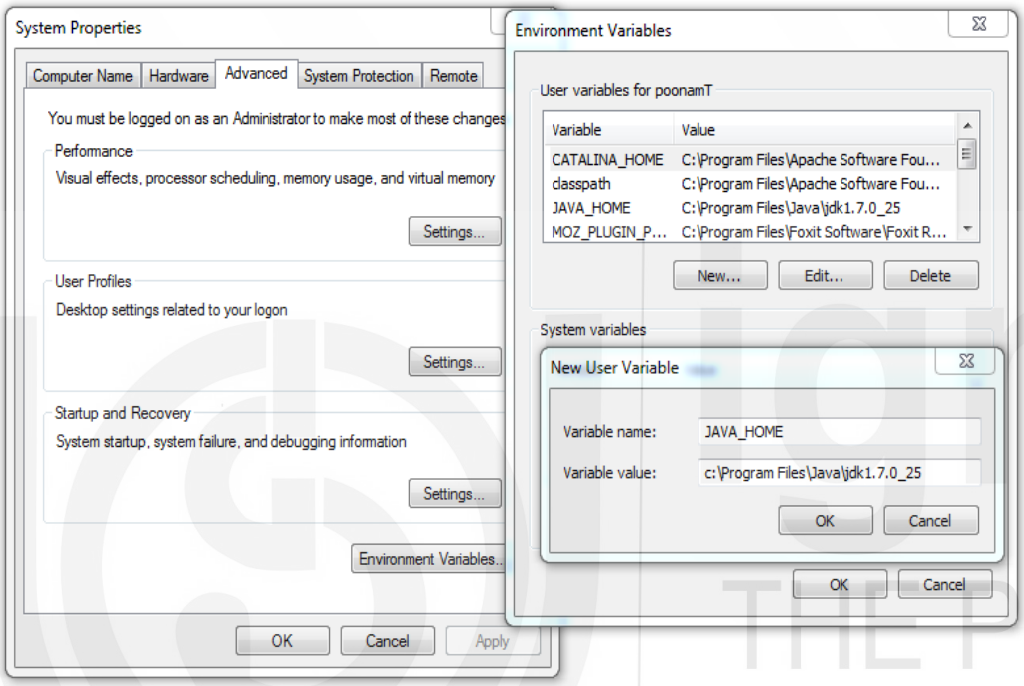


Figure 6: Screen for setting the environment variable.

In a similar manner, you can set the variable name and variable value for the following environment variable by using step 3:

variable name	variable value:
classpath	C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\lib\servlet-api.jar;C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\lib\jsp-api.jar;C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\lib\msbase.jar;C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\lib\msutil.jar;C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\lib\mssqlserver.jar;
Path	C:\ProgramFiles\Java\jdk1.7.0_25\bin;C:\ProgramFiles\Java\jdk1.7.0_25\lib;
CATALINA_HOME	C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37
JRE_HOME	C:\Program Files\Java\jdk1.7.0_25\jre

tep 4 - After installation, you will find the tomcat folder, which contains the following folders:

- **bin** (binary files for Tomcat and friends),
- **conf** (configuration files),
- **lib** (library JAR files),
- **logs** (server log files),
- **temp** (temporary files),
- **webapps** (area Tomcat looks for web application, it contains JSP files, Servlets and other content), and
- **work** (working space for holding translated JSP files).

You can see these folders under your web application like the following figure 7.

Step 5 - Create directory "ex" under the 'webapps' folder as per the following figure 7:

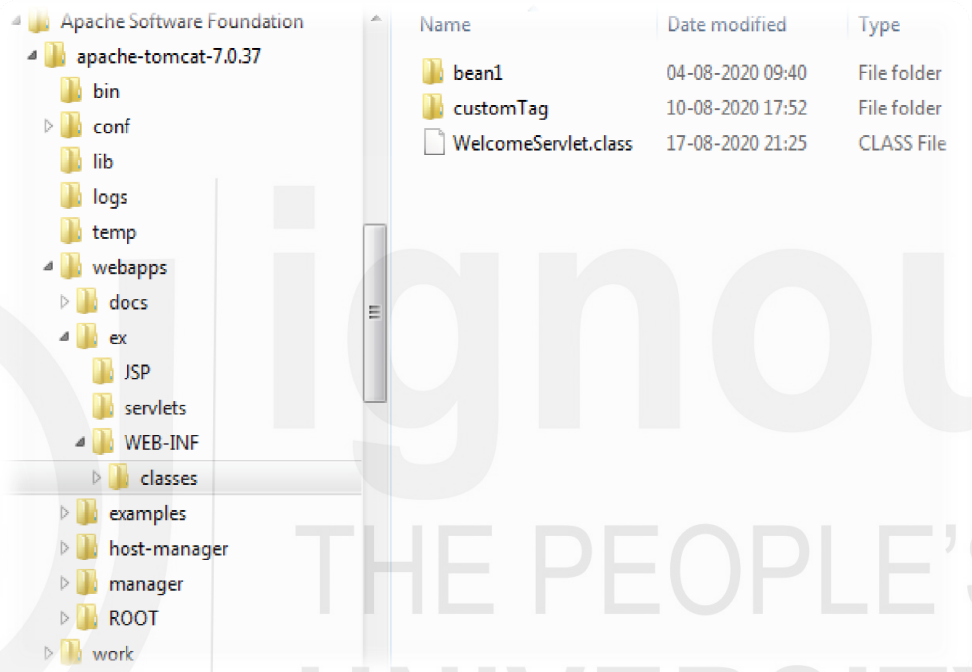


Figure 7: Directory Structure for Tomcat

Step 6 – Create one 'servlets' folder for source file and one "JSP" folder (for JSP program) under 'ex' folder. Also create a directory 'WEB-INF' under the 'ex' folder. Under WEB-INF folder create a folder named 'classes'. All your java classes used in your web application should be placed in 'classes' folder.

Step 7 - Copy web.xml file from ROOT directory and paste into "WEB-INF" folder under the "ex" folder.

Now, the Java and Tomcat installation process is completed.

Compiling and Running the Servlet

Step 8 – Under the 'servlets' folder, place your servlet source file. Now, you will compile this servlet by using the following command at the Command Prompt:

```
C:\Program Files\Apache Software Foundation\apache-tomcat-
7.0.37\webapps\ex\servlets>javac WelcomeServlet.java
```

Figure 8: Command Prompt for Step 8

Step 9- After successful compilation, 'WelcomeServlet.class' file will be created. Place this file in the location: C: / Program Files/…/…/webapps/ex/Web-INF/classes folder.

Step 10 – In the deployment descriptor (web.xml) file under the WEB-INF folder, write the following code as the sub-element of <webapp> element.

```
<servlet>
    <servlet-name>WelcomeServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>

<servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-name>
    <url-pattern>/servlet/WelcomeServlet/*</url-pattern>
</servlet-mapping>
```

Step 11- Start Tomcat Server. In any case, if Tomcat Server is not running from your program menu or shortcut, run the 'startup' command from Command Prompt like the following way:

C:\Program Files\Apache Software Foundation\apache-tomcat-7.0.37\bin>startup

Step 12- Open new tab in the browser or open new window and type the following URL to execute your servlet.

http://localhost:8080/ex/servlet/WelcomeServlet

Congratulations!! Your first servlet is successfully created and run at Tomcat Server. Now, you are able to create a servlet. This is a simple servlet example. In the next Unit of this course, you will learn some other complex servlets with database connectivity.

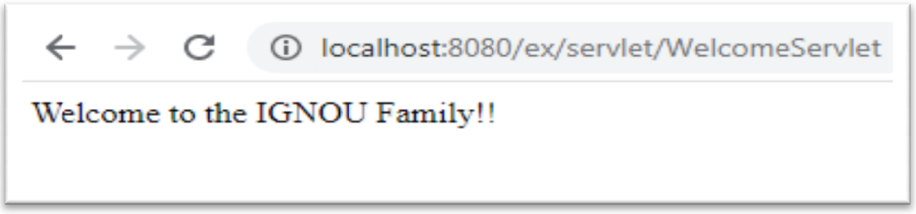


Figure 9: Output Screen for 'WelcomeServlet' program

Steps to Create Servlet Application in Netbeans IDE

In the above section, you have created your first Servlet Application but without using any Integrated Development Environment (IDE). Using IDE, you can easily create Servlet Applications. Here are steps to create Servlet Application in Netbeans IDE. Installation process are given in the lab manual.

Now, start Netbeans and performs the following steps:

Select File -> New Project (CTRL+SHIFT+N) from Menu. In this window you can select Java Web category for your new Servlet/JSP project and select the Next button.

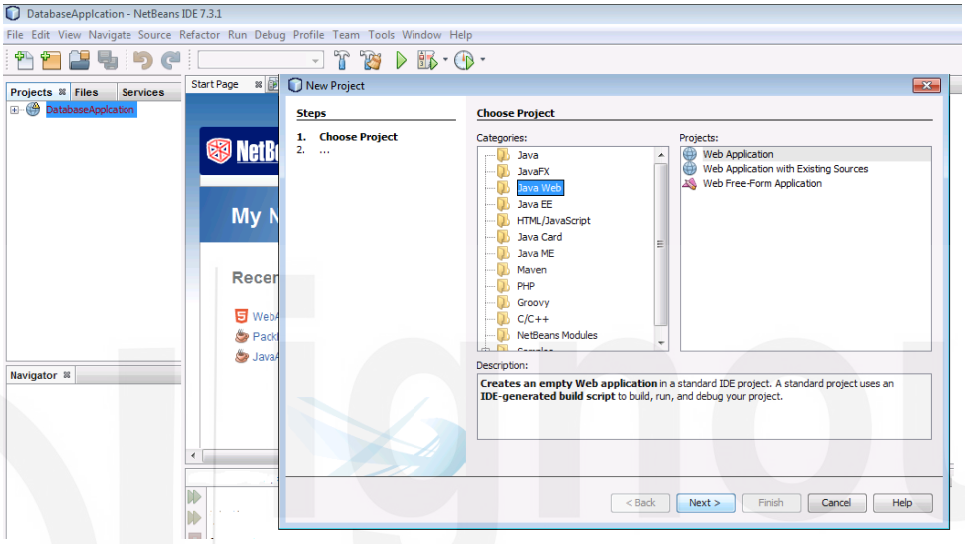


Figure 10: New Project Window

In the next window you can enter project name and location of your choice. For example, you have given the project name as MyProject and selected a Folder C:\Projects for location as shown in the figure-11 and select the Next button.

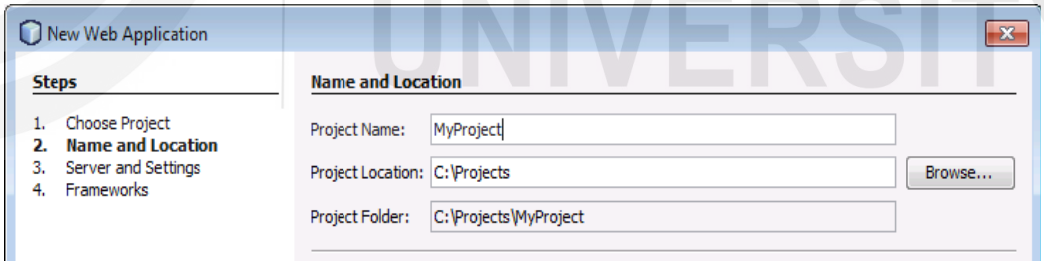
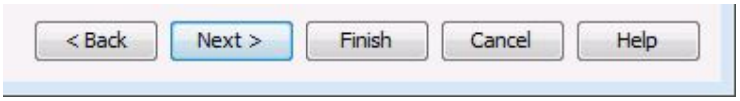


Figure 11: Selection of Name and Location of Project

Now a window appears with Server settings, you can select Finish button from the button Panel.



The complete directory structure required for the Servlet Application will be created automatically by the IDE.

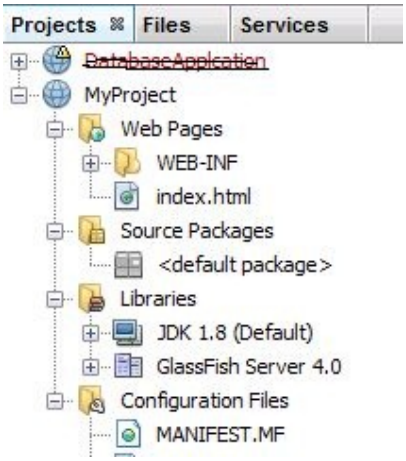


Figure 12: Directory Strucure

Now you are ready to create your web application. To create a Servlet, open Source Package, right click on default packages -> New -> Servlet. Give a Name to your Servlet class file and click on the Next button.

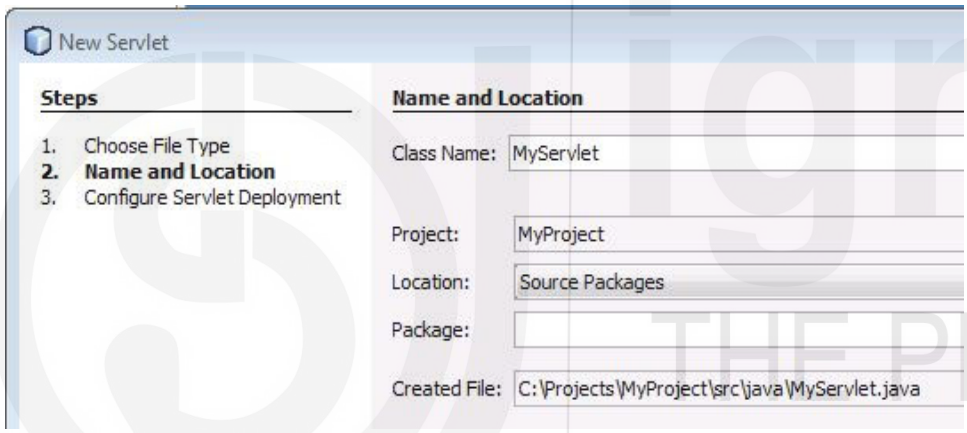


Figure 13: Name and Location window for new Servlet

In the next window you can change servlet name and patterns and finaly click on finish button.

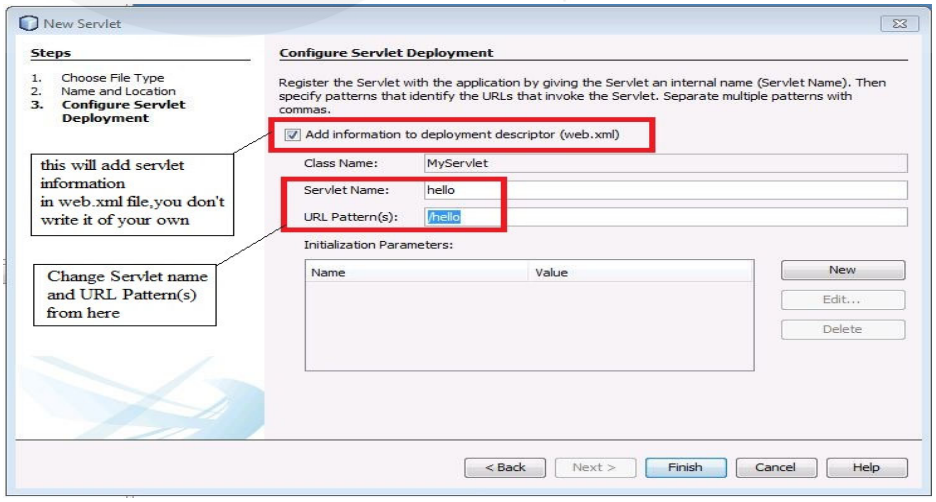


Figure 14: Configure Servlet Deployment Window

Now that Servlet class is ready, you can change the code as per your reqiurement . Once this is completed, edit the index.html file and add the following code in the index.html. This file is read as the first page of the web application.

<h1>Click here to go MyServlet Program</h1>

Edit web.xml file. You can see here the specified url-pattern and servlet-name, this means when hello url is accessed your Servlet file will be executed. If the index.html is not listed as welcome file in web.xml and then you can add this file.

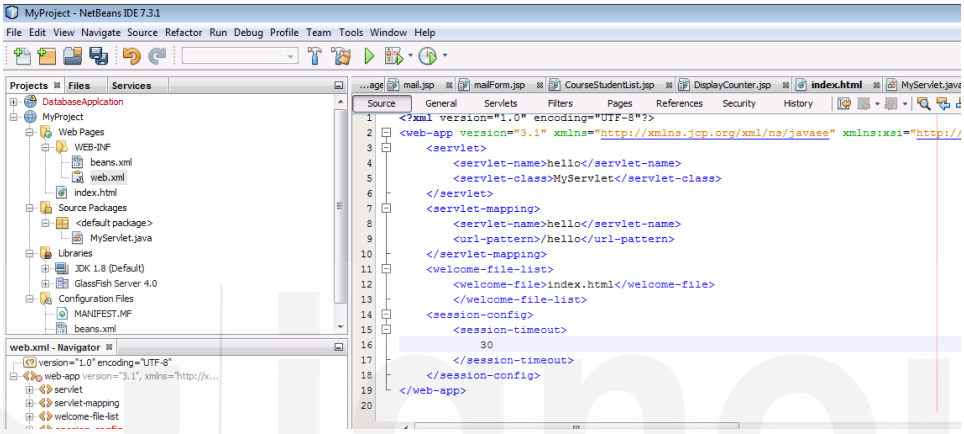


Figure 15: Web.xml file

Now Run your application, right click on your Project name and select **Run**. Click on the link created, to open your Servlet and your first servlet in NetBeans will run. The output of this servlet is shown in Figure-16.

Figure 16:Output of first Servlet program in NetBeans

☞ Check Your Progress 3

1. Which interface contains Servlet life cycle methods?

   ---
2. Explain servlet life cycle methods.

   ---
3. What is the difference between init() and destroy() methods in servlet?

---

4. Write a servlet program to display "We are student of SOCIS, IGNOU!!".

2.8 SUMMARY

This unit introduced you to the basics of Servlet. Servlet is a Java class that creates dynamic content. It takes user request which was sent from the web browser, for processing. Java Servlet is an established technology used for creating dynamic web applications.

After defining the basics of servlet, the unit described the HTTP protocol and HTTP methods. HTTP refers to Hyper Text Transfer Protocol. It is a protocol which governs any data exchange on the Web. HTTP is a client-server protocol i.e. each request is sent by the client to a server that handles it and provides an answer, called the response. While communicating with the Server, clients can use various requests methods. Two of them i.e. GET, and POST are widely used methods that take part in requesting the data in applications.

This Unit also introduced you two packages i.e. javax.servlet and javax.servlet.http that make up the servlet architecture. The two main classes GenericServlet and HttpServlet are defined in Servlet-API. The GenericServlet class provides implementations for all methods; most of them are blank. You can extend GenericServlet and override only methods that you need to use. If you are going to create a web application, then you can extend the HttpServlet class. In the life cycle of a servlet, you have learned three methods such as init(), service(), and destroy() methods. At the end of the unit, you are able to write a simple servlet.

2.9 SOLUTIONS/ANSWERS

☞ Check Your Progress 1

1. A servlet is a small Java program that runs on a web server. Servlets are often run when the user clicks a web link, submits a form or performs some other type of action on a website. Servlets receive and respond to requests of the clients/users of web applications. Using servlets, dynamic web pages are created. It can be seen as a mediator between the incoming HTTP request from the browser and the database. Servlet is a robust server-side programming language.

   When a web client sends an HTTP request to web server, the webserver receives the request and passes the request to the servlet container. The servlet container is responsible for instantiating the servlet or creating a new thread to handle the request. The servlet container forwards this request to the corresponding servlet. The servlet processes the request and generates the response in the form of output. During processing, servlet follows all the phases of life cycle. When servlet container shuts down, it unloads all the servlets and calls destroy() method for each initialized servlets.

2. The Hypertext Transfer Protocol (HTTP) is an application-level protocol for developing distributed applications. This protocol is the foundation for data communication for the WWW. The basic features of HTTP are as follows:
   - HTTP is a request and response protocol.
   - HTTP is a media independent protocol.
   - HTTP is a stateless protocol.
3. HTTP messages consist of textual information encoded in ASCII format. It is used to show how data is exchanged between client and server. There are two types of messages: requests sent by the client for action to the server and responses or the answer from the server. It is based on client-server architecture. An HTTP client is a program that establishes a connection to a server to send one or more HTTP request messages. An HTTP server is a program that accepts connections to serve HTTP requests send by the clients and send HTTP response messages to the client.
4. Both GET, and POST method are used for transfer of data from client to server in HTTP protocol. The main difference between POST and GET method is that GET carries request parameter appended in URL string while POST carries request parameter in message body which makes it a more secure way of transferring data from client to server in HTTP protocol. GET method can send a limited amount of data to the server, while POST method can send data in bulk to the server.

☞ Check Your Progress 2

1. Servlet interface is an API for servlets. Every Servlet should either implement the servlet interface or extends the class which already implements the interface. javax.servlet.GenericServlet and javax.servlet.http.HttpServlet are the Servlet classes that implement the Servlet interface; hence, every servlet should either implement the Servlet interface directly or by extending any of these classes.
2. GenericServlet is an abstract class that implements Servlet interface while HTTPServlet abstract class extends the GenericServlet class. GenericServlet class is a base class for HTTPServlet. GenericServlet does not support any protocol while HTTPServlet support HTTP and HTTPS protocol. HTTPServlet can handle cookies and session but GenericServlet cannot handle them.
3. ServletConfig interface belongs to the package javax.servlet.ServletConfig. It is used for passing the configuration parameters to the servlet. Each Servlet has a separate ServletConfig object. Parameters of ServletConfig are defined under <init-param> tags in web.xml file.
4. Each web application has a common ServletContext. All the servlets in the web application can access the ServletContext. It has the web-application information and resources, which are common and accessible to all the servlets present in the web application. ServletContext is common for all the servlets in the web application.

☞ Check Your Progress 3

1. Servlet interface contains the common methods for all servlets i.e. provides the common behaviour for all servlets. All the three lifecycle methods of a Servlet are defined in Servlet interface, javax.servlet.Servlet. The Servlet Container calls the init() after instantiating the servlet and indicates that the servlet is ready for service. The service() is the method that accepts the

request and response only after init() has been completed. The servlet is terminated by calling destroy() method.

2. A servlet life cycle includes the entire process from its creation till the destruction. The servlet is initialized by calling the init() method. The servlet calls service() method to process a client's request. Destroy() method is used to destroy the servlet. It is called only once at the end of the life cycle of a servlet.
3. The init() method is used to create or load objects used by the servlet to handle its requests while the server calls a servlet's destroy() method when the servlet is about to be unloaded. In the destroy() method, a servlet should free any resources allocated during initialization and shutdown gracefully. Hence this acts as an excellent place to deallocate resources such as an open file or open database connection.
4.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class studentServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,   HttpServletResponse response)
  throws ServletException, IOException
  {
      PrintWriter out = response.getWriter();
      out.println("<HTML>");
      out.println("<HEAD>");
      out.println("<TITLE>Servlet program</TITLE>");
      out.println("</HEAD>");
      out.println("<BODY>");
      out.println("We are student of SOCIS, IGNOU!!");
      out.println("</BODY></HTML>");
  }
}
```

2.10 FURTHER READINGS

- Jason Hunter, and William Crawford “Java Servlet Programming”, O'Reilly Media, Inc.,2001.
- Kathy Sierra, Bryan Basham, anddd Bert Bates ,“Head First Servlets and JSP”, O'Reilly Media, Inc., 2008.
- https://www.w3.org/
- https://www.liquidweb.com/kb/installing-tomcat-9-on-windows/
- https://docs.oracle.com/cd/E19857-01/820-0261/abxbh/index.html
- https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html
- https://www.w3adda.com/servlet-tutorial
- https://www.w3schools.com/tags/ref_httpmethods.asp
- https://docs.oracle.com/javaee/5/api/javax/servlet/http/HttpSession.html