Basics of Servlet (Part-1)

Servlets are server side components that provide a powerful mechanism for developing server side programs. Servlets provide component-based, platform-independent methods for building Web-based applications, without the performance limitations of CGI programs.

Using servlets web developers can create fast and efficient server side application which can run on any servlet enabled web server. Servlets run entirely inside the Java Virtual Machine. Since the Servlet runs at server side so it does not checks the browser for compatibility. Servlets can access the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets can also access a library of HTTP-specific calls, receive all the benefits of the mature java language including portability, performance, reusability, and crash protection.

Servlets are not designed for a specific protocols. It is different thing that they are most commonly used with the HTTP protocols Servlets uses the classes in the java packages "javax.servlet" and "javax.servlet.http".

HOW TO RUN A SERVLET

1. Install Tomcat server in your computer and start the service.
2. create a servlet java file. e.g:-HelloServlet.java

Code:

```
import java.io.*;
import javax.servlet.*;
public class HelloServlet extends GenericServlet
      {
      public  void  service(ServletRequest  request,ServletResponse  response)
throws   ServletException, IOException
          {
           response.setContentType("text/html");
          PrintWriter pw = response.getWriter();
          pw.println("<B>Hello!");
          pw.close();
          }
       }
```

3. Compile the servlet file.
   note:- if jcreator is not able to recognize the servlet-api.jar then set the path in the jcreator i.e configure->options->jdk profiles.
   There u edit the classpath with
   C:\Program Files\Apache Software Foundation\Tomcat 6.0\lib\servlet-api
   Now the file will compile properly.
4. Now you can create a new folder in the "webapps" folder in the tomcat installation folder.
   e.g:-the folder name is “techgeek”

5. Inside that folder create another folder of name “WEB-INF” inside which create another folder of name “classes”.
6. Place the generated .class file in the “classes” folder.e.g:the class file generated is “HelloServlet.class”
7. Now edit an XML file which will help in the translation of the url to the servlet class file.
   The XML file is always created in the “WEB-INF” folder with the name “web.xml”

   The content of the XML file is as follows:-

   Code:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   version="2.5">



<servlet>
      <servlet-name>HelloServlet</servlet-name>
      <servlet-class>HelloServlet</servlet-class>
    </servlet>

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

8. Keep the entry for every .class file.
9. Now open any bowser and write

   http://localhost/8080/techgeek/HelloServlet

   Output:-

   Hello!
   Explanation:-Let's look closely at this program. First, note that it imports the javax.servlet package.This package contains the classes and interfaces required to build servlets.

   Next, the program defines HelloServlet as a subclass of GenericServlet. The GenericServlet class provides functionality that makes it easy to handle requests and responses.Inside HelloServet, the service( ) method (which is inherited from GenericServlet)is overridden. This method handles requests from a client. Notice that the first argument is a ServletRequest object. This enables the servlet to read data that is provided via the client request. The second argument is a ServletResponse object. This enables the servlet to formulate a response for the client.The call to setContentType( ) establishes the MIME type of the HTTP response. In this program, the MIME type is text/html. This indicates that the browser should interpret the content as HTML source code. Next, the getWriter( ) method obtains a PrintWriter. Anything written to this stream is sent to the client as part of the HTTP response. Then println( ) is used to write some simple HTML source code as the HTTP response. Compile this source code and place the HelloServlet.class file in the Tomcat class files directory as described in the previous section.

   A Generic servlet contains the following five methods:

   init()

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

   The init() method is called only once by the servlet container throughout the life of a servlet. By this init() method the servlet get to know that it has been placed into service.

   The servlet cannot be put into the service if the init() method does not return within a fix time set by the web server. It throws a ServletException

   Parameters - The init() method takes a ServletConfig object that contains the initialization parameters and servlet's configuration and throws a ServletException if an exception has occurred.

   service()

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

   Once the servlet starts getting the requests, the service() method is called by the servlet container to respond. The servlet services the client's request with the help of two objects. These two objects javax.servlet.ServletRequest and javax.servlet.ServletResponse are passed by the servlet container.

   The status code of the response always should be set for a servlet that throws or sends an error.

   Parameters - The service() method takes the ServletRequest object that contains the client's request and the object ServletResponse contains the servlet's response. The service() method throws ServletException and IOExceptions exception.

getServletConfig()

```
public ServletConfig getServletConfig()
```

This method contains parameters for initialization and startup of the servlet and returns a ServletConfig object. This object is then passed to the init method. When this interface is implemented then it stores the ServletConfig object in order to return it. It is done by the generic class which implements this inetrface.

Returns - the ServletConfig object

getServletInfo()

```
public String getServletInfo()
```

The information about the servlet is returned by this method like version, author etc. This method returns a string which should be in the form of plain text and not any kind of markup.

Returns - a string that contains the information about the servlet

destroy()

```
public void destroy()
```

This method is called when we need to close the servlet. That is before removing a servlet instance from service, the servlet container calls the destroy() method. Once the servlet container calls the destroy() method, no service methods will be then called . That is after the exit of all the threads running in the servlet, the destroy() method is called. Hence, the servlet gets a chance to clean up all the resources like memory, threads etc which are being held.

LIFE CYCLE OF A SERVLET

The life cycle of a servlet can be categorized into four parts:

Loading and Inatantiation: The servlet container loads the servlet during startup or when the first request is made. The loading of the servlet depends on the attribute <load-on-startup> of web.xml file. If the attribute <load-on-startup> has a positive value then the servlet is load with loading of the container otherwise it load when the first request comes for service. After loading of the servlet, the container creates the instances of the servlet.

Initialization: After creating the instances, the servlet container calls the init() method and passes the servlet initialization parameters to the init() method. The init() must be called by the servlet container before the servlet can service any request. The initialization parameters persist untill the servlet is destroyed. The init() method is called only once throughout the life cycle of the servlet.

The servlet will be available for service if it is loaded successfully otherwise the servlet container unloads the servlet.

Servicing the Request: After successfully completing the initialization process, the servlet will be available for service. Servlet creates seperate threads for each request. The sevlet container calls the service() method for servicing any request. The service() method determines the kind of request and calls

the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.

Destroying the Servlet: If the servlet is no longer needed for servicing any request, the servlet container calls the destroy() method . Like the init() method this method is also called only once throughout the life cycle of the servlet. Calling the destroy() method indicates to the servlet container not to sent the any request for service and the servlet releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.

The HttpServlet Class

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

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( ). Out of all the functions doGet() and doPost() method is most important.

(note:- Before knowing about the above two functions you need to know about the the http get and post request)

Handling http get request

Here we will develop a servlet that handles an HTTP GET request. Notice that the action parameter of the form tag specifies a URL. The URL identifies a servlet to process the HTTP GET request.

Code:

```
<html>
<body>
<center>
        <form name="Form1"
        action="http://localhost:8080/examples/servlet/ColorGetServlet">
                <B>Color:</B>
                <select name="color" size="1">
                        <option value="Red">Red</option>
                        <option value="Green">Green</option>
                        <option value="Blue">Blue</option>
                </select>
        <br /><br />
        <input type=submit value="Submit">
        </form>
</center>
</body>
</html>
```

The source code for ColorGetServlet.java is shown in the following listing. The doGet( ) method is overridden to process any HTTP GET requests that are sent to this servlet. It uses the getParameter( ) method of HttpServletRequest to obtain the selection that was made by the user. A response is then formulated.

Code:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorGetServlet extends HttpServlet
     {
       public   void   doGet(HttpServletRequest   request,HttpServletResponse
response throws ServletException, IOException
          {
         String color = request.getParameter("color");
         response.setContentType("text/html");
         PrintWriter pw = response.getWriter();
         pw.println("<B>The selected color is: ");
         pw.println(color);
         pw.close();
          }
    }
```

Compile the servlet and perform these steps to test this example:

1. Start Tomcat, if it is not already running.
2. Display the Web page in a browser.
3. Select a color.
4. Submit the Web page.

After completing these steps, the browser will display the response that is dynamically generated by the servlet.

One other point: Parameters for an HTTP GET request are included as part of the URL that is sent to the Web server. Assume that the user selects the red option and submits the form. The URL sent from the browser to the server is http://localhost:8080/examples/servl...vlet?color=Red The characters to the right of the question mark are known as the query string.

Handling of the http post is same. Just we need to give the method=”post” in the html file and in the servlet file we need to define doPost method instead of doGet method.

These are all about the basics of Servlet.

Source: http://www.go4expert.com/articles/basics-servlet-part-1-t21602/