|  |
| --- |
| **Servlet Basics** |

**Servlet Life Cycle Methods and Phases**

For a java program, there is one method which serves as the entry point to the application which is the main() method. For a servlet, we have three methods listed below.

**Life Cycle methods**

init()

service()

destroy()

These methods are called by the Web Container such as Apache Tomcat, WebLogic.

Init() method is called once. So, this is a good place to write code to establish connections to the database server.

The Service method is where all our business logic is kept. The service method is called n-times.

The destroy method should have code which does the opposite of what the init() method does. The destroy method is also called once.

Putting code in these method is our responsibility but calling the methods is the container’s responsibility.

There are 4 life cycle phases associated with the above three methods, which are listed below.

**Life Cycle Phases**

Instantiation – This is when the classes are loaded by the container.

Initialization – This is when the init method is used to open resources.

Servicing – this is where the service method is executed.

Destruction – this is when the destroy method is called.

**Web Application Folder Structure**

com/samsonmarikwa/OrderServlet.class

login.jsp

Mysql.jar spring.jar hibernate.jar

lib

classes

web.xml

home.html

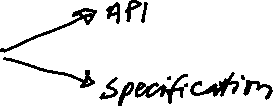
WEB-INF

WEBAPP

**Servlets**

A technology in the JEE that allows us to build dynamic web applications using java. The Servlet standard has an API and a Specification just like other standards from Oracle.

The API is for developers to develop dynamic web applications. There is a set of interfaces and classes. The Specification itself is a set of rules written in plain English which is for Application Server or Web Container developers like Apache Tomcat, WebSphere, WebLogic. The developers follow the rules described in the spec.



**Servlet**

is a program which runs on a web container. It receives input from a browser, processes it and depending on the request, can make calls to a database server. It can do a lot of things such as writing to files and reading from files. It receives responses from the database server and processes it before sending the response back to the client (browser).

OrderServlet.java

Servlet

Web Browser

DB

Web Container

**Servlet Annotations 3.0**

Servlet 3.0 version introduces annotations that can be used to configure various servlet components instead of using the web.xml file.

Examples include

@WebServlet

@WebInitParam

@WebFilter

@WebListener

*web.xml configurations override the annotations defined for the same component.*

**Registering a servlet**

Within the web.xml, create the following entries

|  |
| --- |
| <servlet>  <servlet-name>HelloServlet</servlet-name>  <servlet-class>com.samsonmarikwa.training.servlets.HelloWorldServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>HelloServlet</servlet-name>  <url-pattern>/hello</url-pattern>  </servlet-mapping> |

**Application Flow**

When a request comes from the browser to <http://localhost:8080/ServletBasics/hello> Apache Tomcat will handle the request as it is running on localhost port 8080. Tomcat determines the project name from the url that should handle the request. It then looks at the web.xml file for the /hello path configured in the servlet-mapping url-pattern to see which servlet should handle the request. It then instantiates the servlet HelloWorldServlet to handle the request. It will then execute the service method that is in the logic in our servlet.

**When using html**

To send input to the servlet via html form

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"ISO-8859-1"*>  <title>Addition</title>  </head>  <body>  <form action=*"additionServlet"*>  <h2>Enter the Numbers</h2><br />  Number1: <input type=*"text"* name=*"number1"* /><br/>  Number2: <input type=*"text"* name=*"number2"* /><br/>  <input type=*"submit"* name=*"submit"* />  </form>  </body>  </html> |

Add a servlet to retrieve the input, add and send the response

|  |
| --- |
| **public** **class** AdditionServlet **extends** GenericServlet {  @Override  **public** **void** service(ServletRequest request, ServletResponse response)  **throws** ServletException, IOException {  **if** (request.getParameter("number1") != **null** && request.getParameter("number2") != **null**) {  **int** num1 = Integer.*parseInt*(request.getParameter("number1"));  **int** num2 = Integer.*parseInt*(request.getParameter("number2"));  PrintWriter out = response.getWriter();  out.println("The result is " + (num1 + num2));  }  }  } |

Add a servlet to retrieve the input, add and send the response

|  |
| --- |
| <servlet>  <servlet-name>AdditionServlet</servlet-name>  <servlet-class>com.samsonmarikwa.training.servlets.AdditionServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>AdditionServlet</servlet-name>  <url-pattern>/additionServlet</url-pattern>  </servlet-mapping> |
| **JDBC** |

**JDBC Architecture**

|  |  |
| --- | --- |
| JDBC Client | The client is the code that we write. Connects to the DB. Perform CRUD operations. Processes the response. Close the connection. |
| JDBC API | Part of java.sql.\* package  We as developers, we learn and use the JDBC API  Comprises Connection, Statement, ResultSet and many more |
| JDBC Driver | Exists between the client software and the DB. Implements the JDBC API under the JDBC Specification. |
| JDBC Manager | Driver Manager establishes connection between the Client and Driver. After that the communication takes place between the client and the driver. |

Only Once to get a connection

Communicates ‘n’ times after a connection is acquired via the Driver Manager

DatabaseServer

JDBC API

Driver Manager

JDBC Driver

JDBC Client

**Steps to perform CRUD**

1. Establish the connection
2. Create the statement object
3. Submit the SQL query to DBMS
4. Close the statement
5. Close the connection