

JAVA SERVLET PROGRAMMING

Learning Goals

- Understand Servlet
- Can use Servlet to develop web application

Structure

- Introduction
- Servlet Overview and Architecture
- Servlet Life Cycle
- Handling HTTP get Requests
- Handling HTTP post Requests
- Servlet Context



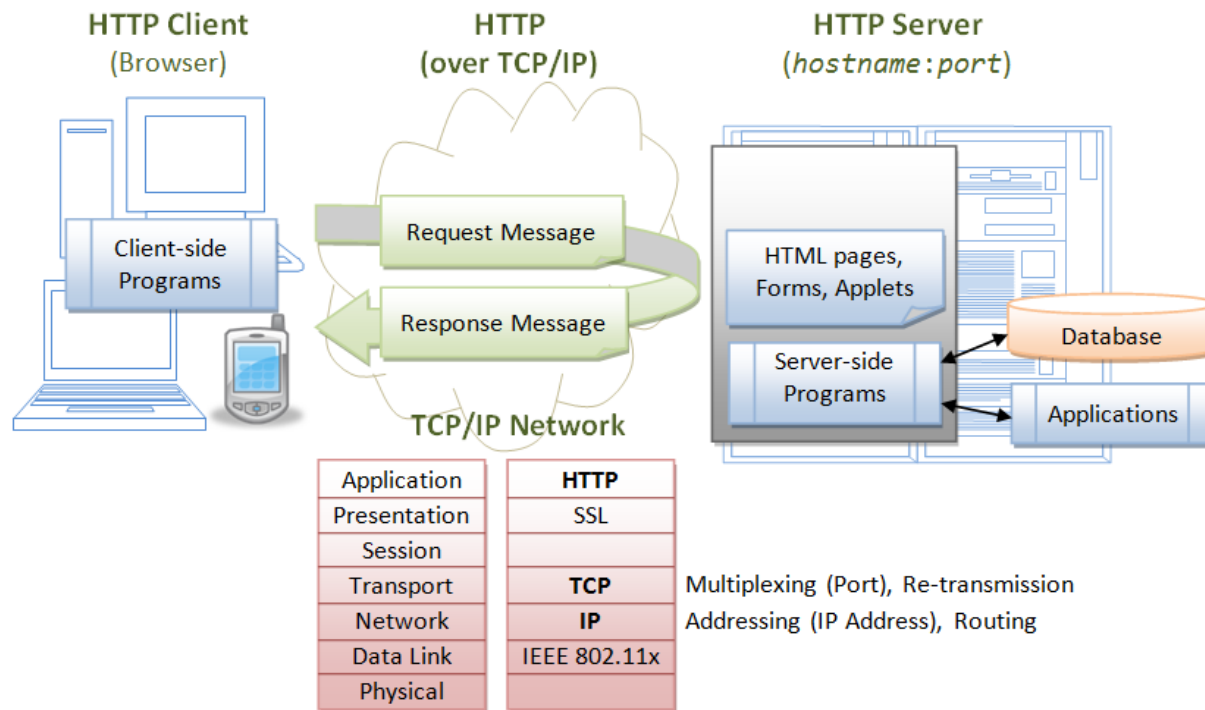
Trainee's missions

To complete this course and achieve goals, trainees must:

- ▶ Read Lecture, Reference
- ▶ Do Exercises
- ▶ Take quiz
- ▶ Complete Assignment

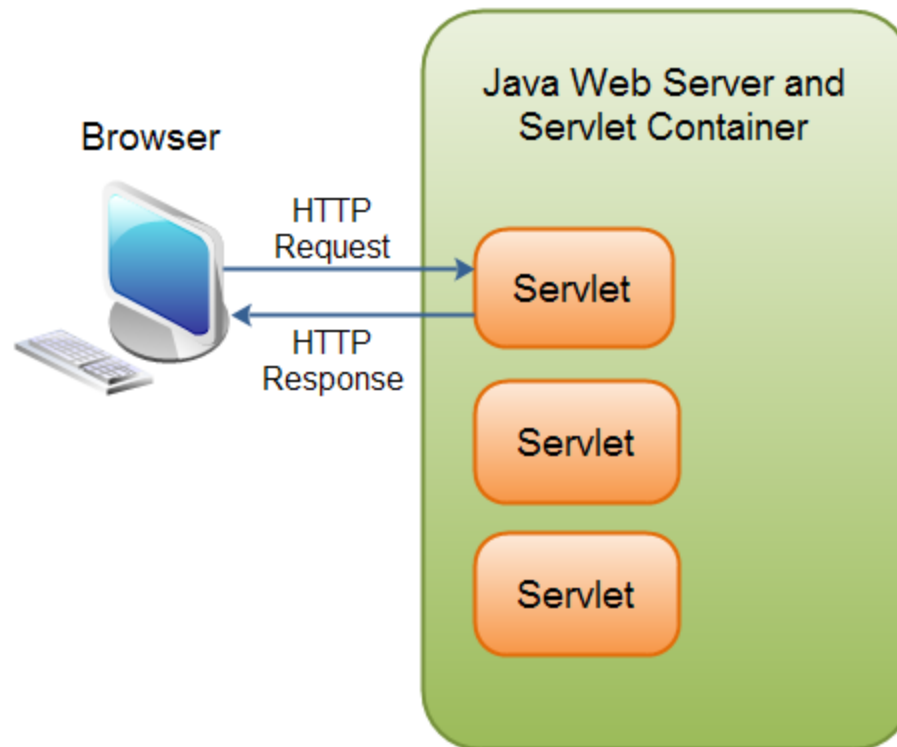
Introduction

- Servlets to build Web-based applications
- Without the performance limitations of CGI programs.
- Servlets have access to JDBC API to access enterprise databases.



Servlet Overview

- Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.
- Performance is significantly better.
- Servlets are platform-independent because they are written in Java.



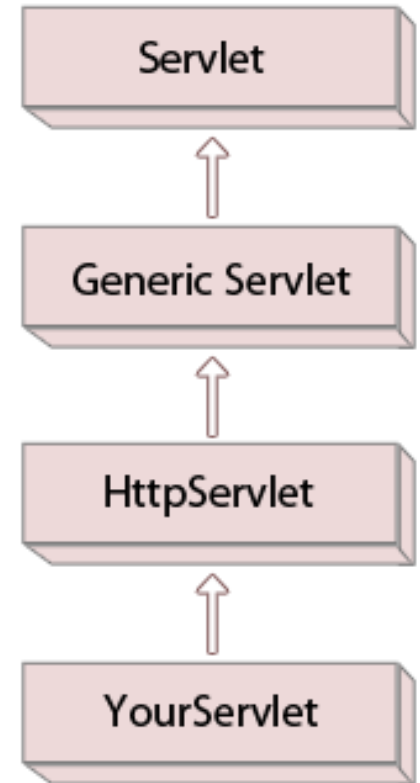
Servlet Architecture

There are two packages `javax.servlet` and `javax.servlet.http` that provides the interfaces and classes:

- **Servlet**: is a interface
- **Generic** implements **Servlet**
- **HttpServlet** extends **Generic**

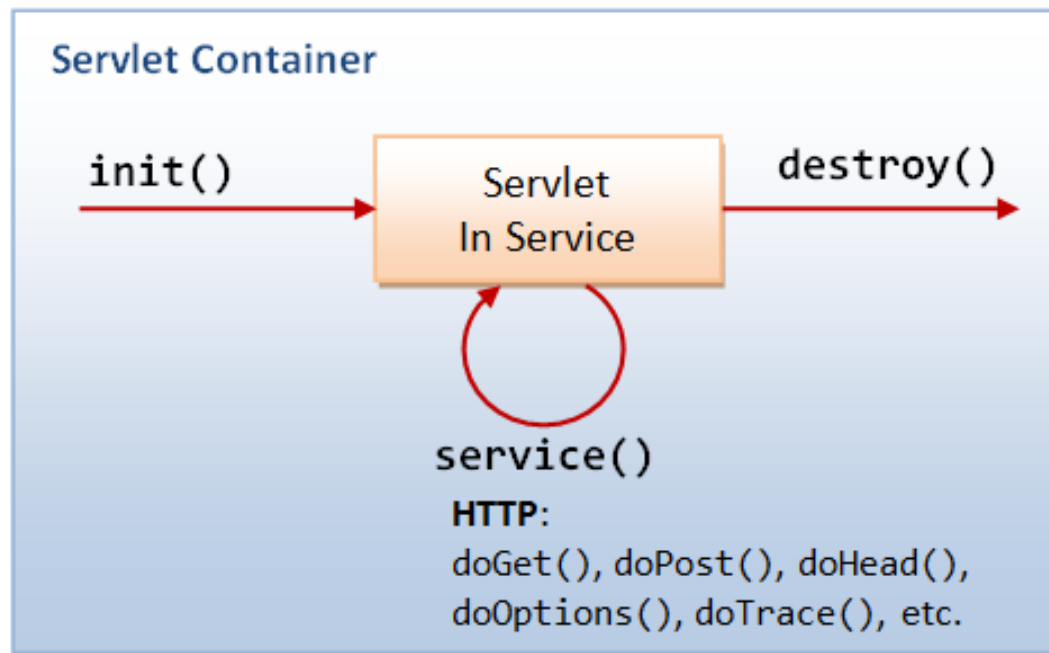
To write a servlet we need to implement Servlet interface. Servlet interface can be implemented directly or indirectly by extending **GenericServlet** or **HttpServlet** class.

Purpose of extending the `HttpServlet` class is to provide the HTTP specific services to your servlet.

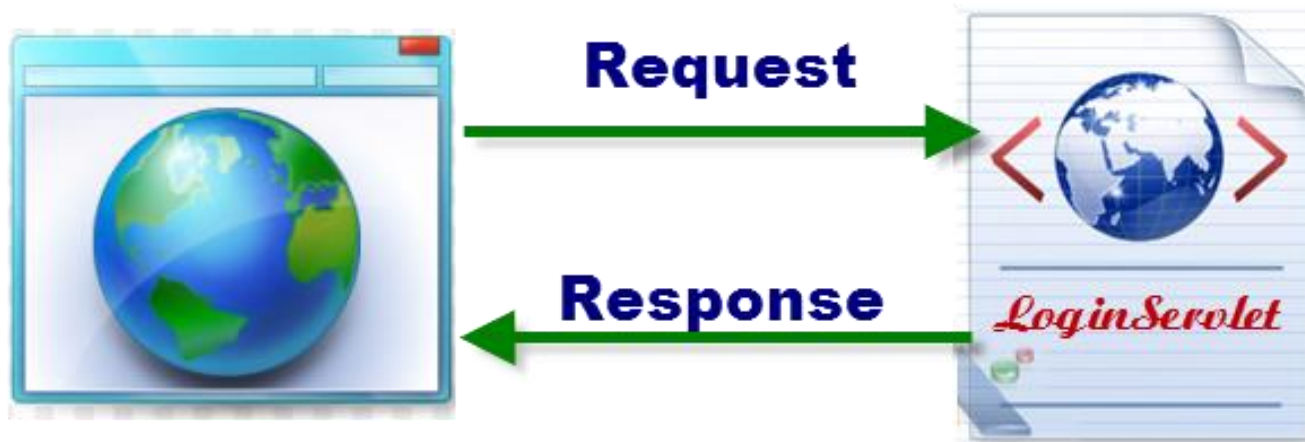


Servlet Life Cycle

- A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet:
- The servlet is initialized by calling the `init()` method.
- The servlet calls `service()` method to process a client's request.
- The servlet is terminated by calling the `destroy()` method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.



Servlet Demo workflow



Servlet Demo workflow

Create Dynamic Web Project

The screenshot displays an IDE interface with a Project Explorer on the left and a code editor on the right. The Project Explorer shows a project named 'JAVASERVLET' with a 'src' folder containing 'demo.servlet' and 'LoginServlet.java'. It also lists 'JRE System Library [J2SE-1.5]', 'Web App Libraries', 'servlet.jar - D:\PROJECT\JAVASERVLET\lib\servlet.jar', 'build', 'WebContent', 'META-INF', 'MANIFEST.MF', 'WEB-INF', 'lib', and 'web.xml'. The code editor shows the 'LoginServlet.java' file with the following code:

```
package demo.servlet;

import java.io.IOException;

/**
 * Servlet implementation class LoginServlet
 */
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static String USER_NAME;
    private static String PASSWORD;

    public void init(ServletConfig config) throws ServletException {

        System.out.println("LoginServlet::init::BEGIN");

        ServletContext context = config.getServletContext();
        USER_NAME = context.getInitParameter("USER_NAME");
        PASSWORD = context.getInitParameter("PASSWORD");

        System.out.println("LoginServlet::init::END");

    }
}
```

The bottom of the IDE shows a toolbar with icons for Problems, Javadoc, Declaration, Console, Servers, Search, History, Checkstyle violations, and djUnit.

Handling HTTP GET Requests

doGet() to process requests with method is "GET" from client

The screenshot displays the Eclipse IDE with the `LoginServlet.java` file open. The `doGet` method is highlighted with a red box and annotated with three numbered callouts:

- 1** points to the method signature: `protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {`
- 2** points to the content type and writer setup: `response.setContentType("text/html");
PrintWriter out = response.getWriter();`
- 3** points to the HTML output generation: `out.println("<html>");
out.println("<head>");
out.println("<title>Login - Servlet</title>");
out.println("</head>");

// body section of document
out.println("<body>");
out.println("<h1>Login Servlets!</h1>");
out.println("<form action='" + request.getContextPath() + "/Login' method='POST'>");
out.println("<table>");
out.println("<tr>");
out.println("<td>Username:</td>");
out.println("<td><input type='text' name='userName' /></td>");
out.println("</tr>");
out.println("<tr>");
out.println("<td>Password:</td>");
out.println("<td><input type='password' name='password' /></td>");`

The Project Explorer on the left shows the project structure for `SERVLET_DEMO`, including `src`, `lib`, and `web.xml`. The bottom status bar indicates the server is running: `<terminated> Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 6, 2014 11:08:38 AM)`.

Handling HTTP POST Requests

doPost() to process requests with method is "POST" from client

The screenshot displays the Eclipse IDE with the `LoginServlet.java` file open. The `doPost` method is highlighted with a red box. Three callouts are present:

- 1** Points to the method signature: `protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`.
- 2** Points to the parameter retrieval: `String userName = request.getParameter("userName");` and `String password = request.getParameter("password");`.
- 3** Points to the HTML response generation: `response.setContentType("text/html");` and the subsequent `out.println` statements.

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    String userName = request.getParameter("userName");
    String password = request.getParameter("password");
    PrintWriter out = response.getWriter();

    response.setContentType("text/html");
    out.println("<html >");
    out.println("<head>");
    out.println("<title>Welcome - Servlet</title>");
    out.println("</head>");
    // body section of document
    out.println("<body>");
    out.println("<h1>Login Successful!</h1>");
    out.println("<h1>Hello: " + userName + " with Password is "
        + password + "!</h1>");
    out.println("</body>");
    // end XHTML document
    out.println("</html>");
    out.close(); // close stream to complete the page
}
```

ServletContext

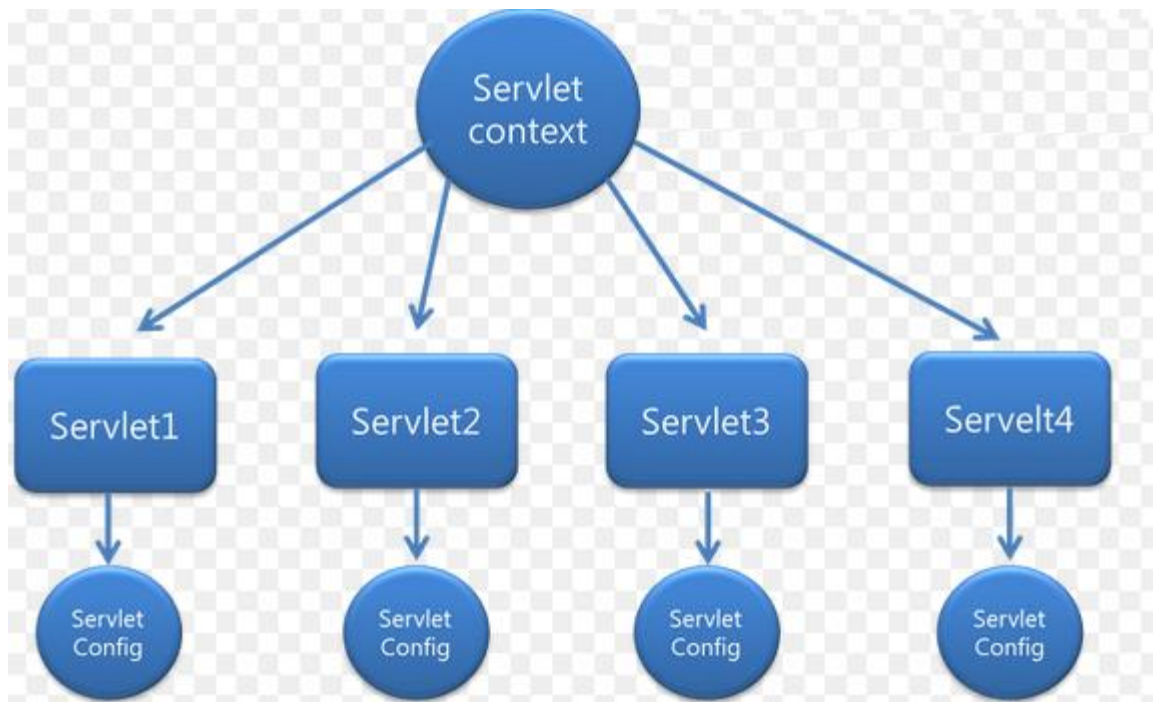
ServletContext is used to maintain state for web applications.

Servlet Context has 3 main methods:

- GetAttribute ()
- SetAttribute ()
- RemoveAttribute ()

Servlet Context help provides communication between the servlet

Servlet Context can also be used to obtain configuration information web.xml



ServletContext Example

web.xml X

```
<display-name>JAVASERVLET</display-name>

<welcome-file-list>
  <welcome-file>Login</welcome-file>
</welcome-file-list>

<context-param>
  <param-name>USER_NAME</param-name>
  <param-value>admin</param-value>
</context-param>

<context-param>
  <param-name>PASSWORD</param-name>
  <param-value>admin123</param-value>
</context-param>
```

1

web.xml X LoginServlet.java X

```
Servlet implementation class LoginServlet
*/
public class LoginServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  private static String USER_NAME;
  private static String PASSWORD;

  public void init(ServletConfig config) throws ServletException {

    System.out.println("LoginServlet::init::BEGIN");

    ServletContext context = config.getServletContext();
    USER_NAME = context.getInitParameter("USER_NAME");
    PASSWORD = context.getInitParameter("PASSWORD");

    System.out.println("LoginServlet::init::END");

  }
```

2

Summary

- Servlets to build Web-based applications
- **There are two packages `javax.servlet` and `javax.servlet.http` that provides the interfaces and classes:**
 - **Servlet**: is a interface
 - **Generic** implements **Servlet**
 - **HttpServlet** extends **Generic**
- Servlet life cycle
 - The servlet is initialized by calling the `init ()` method.
 - The servlet calls `service()` method to process a client's request.
 - The servlet is terminated by calling the `destroy()` method.
- `doGet()`: process request with “GET” method
- `doPost()`: process request with “POST” method
- `ServletContext`: is used to maintain state for web applications.