**HISTORY OF JAVA WEB SERVLET and JavaServer Pages (JSP)**

* Java is built naturally with internet functionality. In **June 1997**, Sun Microsystems announced the Java Servlet.
* Java Servlet is one of the fundamental building block in developing different components of a Java Web specifically mainstream server-side Java which runs in a single process by using grain threads.
* While in **1998,** Sun released the JavaServer Pages **(JSP).**
* JSP is used to easily code a dynamic content of the web’s HTML pages.
* Both Servlets and JSPs lets the programmers build or develop a portable, easy to maintain, modular and scalable Web applications.

**THE JAVA SERVLET**

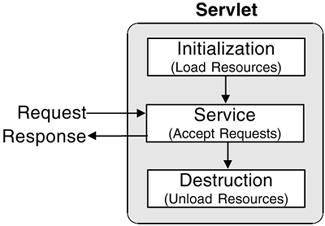
* In web applications, servlets were the first one which got the full access and power of Java. Wherein just like applets, it is completely and fully written in Java programming language.
* It is used for extending the server’s capability to host different applications that are accessed through a programming model, the request-response model. Which means it takes the request of the client and generates the response to the request.
* Java Servlet’s version 2.4 (current) which is included in the Java 2 Enterprise Edition (J2EE). Through the Tomcat project, it is available for free and also an open source.

**Features of Servlet 2.4:**

* Web Applications: In this section, servlet is always part of it (the web application), wherein it provides all the resources of a website.

**Servlet Container** – it manages all the Servlets in the basis of Web application.

*The web container facilitates the conversion to and from the HTTP request/response message to HTTP Servlet Request/HTTP Servlet Response*.

* Servlets and HTTP Servlets: With the support of HTTP, Servlet is used to provide a dynamic web pages and also plays an essential mechanism in Client/Server model.
* Filters: Used in authentication, logging and compression and it is an abstracted method that manipulates the request and response of the client before the request ends.
* Security: Servlets used the security that has been provided by JVM but also provides a way to control access of resources in the web application.
* Internalization: One of the best feature of a Servlet isthat it can be develop content using variety of different languages by means of Servlet API.
* Servlet Life Cycle

1. *Instantiation:*

* The web container created the instance only once in the cycle.

1. *Initialization:* **init() method is invoked**

* This phase represents the creation of different resources to service requests. The init() method is invoked only once and before servicing of request takes place, the servlet invokes it first.

Syntax:

public void init(ServletConfig config) throws ServletException

1. *Request Handling:* **service() method is invoked**

* This phase represents all requests interactions invoked by each client.

Two parameters of service() method: Represents client’s request and Servlets’s client’s response.

1. javax.servlet.ServletRequest
2. javax.servlet.ServletResponse

Syntax:

public void service(ServletRequest request, ServletResponse response)

  throws ServletException, IOException

1. *Destruction:* **destroy() method is invoked**

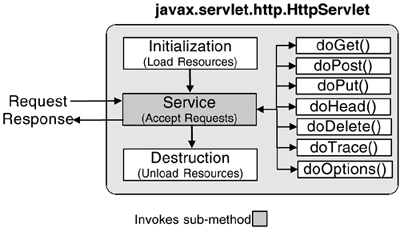
* Represents when the Servlet is removed in a container, simply, the destruction of a life cycle phase. The container calls the method destry() and terminate the resources that have been created.

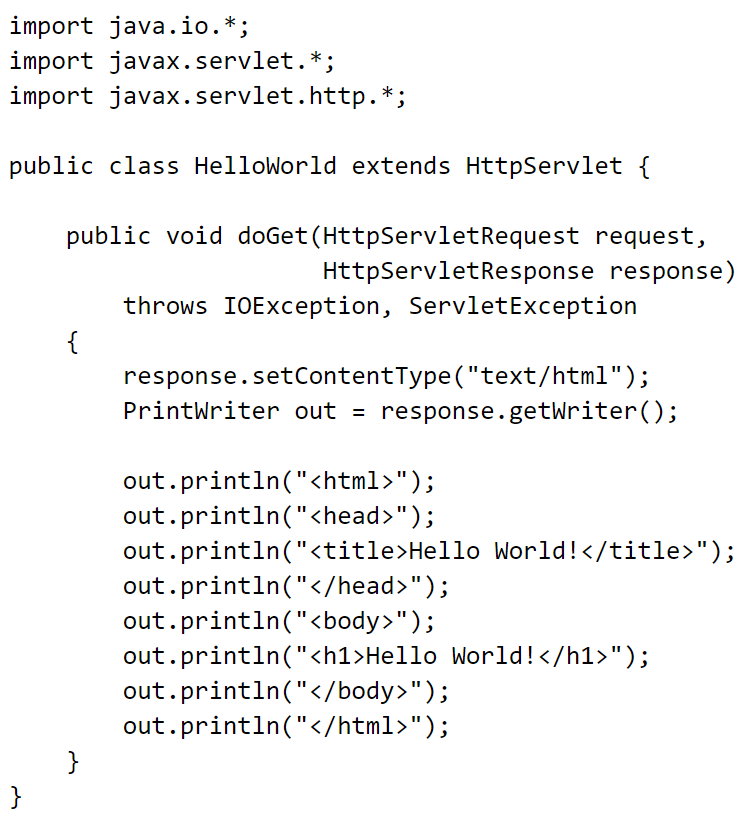
Syntax:

public void destroy()

* The HttpServlet object also uses the Servlet Life Cycle yet there are some modifications when it comes to HTTP protocol.
* During the phase of the service request, one of the seven helper methods is called which is appropriate to the type of HTTP request, named as follows:

1. doGet()
2. doPost()
3. doPut()
4. doHead()
5. doOptions()
6. doDelete()
7. doTrace()

The HttpServlet Life Cycle

* The figure below is a sample code for HttpServlet which will generate a simple HTML page.
* When deploying a servlet, it is not the case that it is written fully in java. For the client to access the Servlet, a unique URL is declared within the Web Application Deployment Descriptor which is the web.xml.
* The web.xml relies on the following two new elements:

1. servlet element – loaded by the web application to define the a Servlet.
2. Sevlet-mapping element – map the Servlet in a URL with the following servlet specification:
3. **An exact pattern to match.**

To use a one-to-one mapping, a / is used to mapped a request within a specific server.

1. **An extension match, \*.extension.**

All URL’s with an extension are being send into a specific Servlet which is used by Servlet frameworks.

1. **A path mapping.**

Begins with a / then ends with /\* and anything can be written between them. It forwards all the request that lands in a specific directory with its specific Servlet respectively.

1. **Default Servlet, /.**

When there is no path provided, requests forwarded is defined by the default servlet mapping.

## **Servlet Configuration**

* Consist of an array of string values which allows a servlet to have an initial parameters outside the code already compiled without the need to recompilation.
* ServletConfig - parameters are being initialize by the two following methods.

1. getInitParameter(String name) - If the parameter didn’t exist, a String object together with the value of the initialized parameter is returned.
2. getInitParameterNames() - If there is no parameters being initialized by the servlet, it returna an empty Enumeration.

**HttpServletRequest and HttpServletResponse**

* **HttpServletResponse**
* HttpServletResponse

To be able to consistently write a content within the output stream, the methods **getWriter()** or **getOutputStream()** can be used to send back the customize content to the client such as binary or text. If either one of the two methods is not called, an exception is thrown.

**Table 1-1.HTML Markup from HelloWorld Servlet**

| **Generated Markup** | **HelloWorld.java** |
| --- | --- |
| <html> | out.println("<html>"); |
| <head> | out.println("<head>"); |
| <title>Hello World!</title> | out.println("<title>Hello World!</title>"); |
| </head> | out.println("</head>"); |
| <body> | out.println("</head>"); |
| <h1>Hello World!</h1> | out.println("<h1>Hello World!</h1>"); |
| </body> | out.println("</body>"); |
| </html> | out.println("</html>"); |

#### HttpServletRequest

* Is used by retrieving request files, headers, and parameters from the clients. One of the helpful features of this object is the session management.

**ServletContext**

#### Distributed Environments

#### ServletContext serves as a reference oject of a Web Application.

#### By the use of J2EE server, lets different server handles the deficiency of a large-scale Web applications.

## **Servlet Event Listeners**

* In cases where there is an important event a container can be used to notify an Application of the Website. Seeing a Web Application being initialized by a container and when it is removed from use. A listener which waits for a clients’ new session can be used to make completion easy.
* Below are the interfaces for event listeners which corresponds to request life cycle events, request attribute binding, Web Application life cycle events, Web Application attribute binding, session[36](javascript:popUp('/content/images/chap2_0321136497/elementLinks/ch02fn36.html')) life cycle events, session attribute binding, and session serialization:
* javax.servlet.ServletRequestListener
* javax.servlet.ServletRequestAttributeListener
* javax.servlet.ServletContextListener
* javax.servlet.ServletContextAttributeListener
* javax.servlet.http.HttpSessionListener
* javax.servlet.http.HttpSessionAttributeListener
* javax.servlet.http.HttpSessionAttributeListener

#### SUMMARY:

#### Server-side java has a fundamental building block called Servlets.

#### Initialization, service, and destruction is the three-phase life cycle of Servlet which outperformed traditional CGI.

#### On the World Wide Web(WWW), the HTTPServlets is being referred by the word “Servlets”.

#### There is such a class which is made specifically for users and for the server-side it simplifies the Java support for HTTP, this class is the HttpServlet.

#### The Servlet API consists of the HttpServletRequest, which is the client’s request; the HttpServletResponse, which is the server’s response; the HttpSession, which is a session for connecting separate requests; and the ServletContext, which the entire Web Application.

**QUIZ!!!**

**Identification/Fill in the blanks.**

Servlets are the fundamental building block of server-side\_\_\_(a)\_\_\_. A \_\_\_(b)\_\_\_ is highly scalable and easily outperforms traditional CGI by means of a simple four-phase life cycle: \_\_\_(d)\_\_\_, \_\_\_(e)\_\_\_, \_\_\_(f)\_\_\_, and \_\_\_(g)\_\_\_. Commonly, the term Servlets actually refers to \_\_\_(h)\_\_\_ used on the World Wide Web. The \_\_\_(i)\_\_\_ class is designed especially for this user and greatly simplifies server-side Java support for HTTP.

The basics of the Servlet API consist of objects that represent a client's request, \_\_\_(j)\_\_\_, the server's response, \_\_\_(k)\_\_\_, a session for connecting separate requests, \_\_\_(l)\_\_\_, and an entire Web Application, \_\_\_(m)\_\_\_. Each of these objects provides a complete set of methods for accessing and manipulating related information. These objects also introduce two new scopes for Servlet developers to use: \_\_\_(n)\_\_\_and \_\_\_(o)\_\_\_.Binding an object to these various scopes allows a Servlet developer to share an object between multiple Servlets and requests for extended periods of time.