Basic servlet structure

```
package testPackage; // Always use packages.
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletTemplate extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
PrintWriter out = response.getWriter();
}
}
```

Basic servlet structure

Generate plain text:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
PrintWriter out = response.getWriter();
out.println("Hello World");
}
}
```

Basic servlet structure

Generate plain HTML:

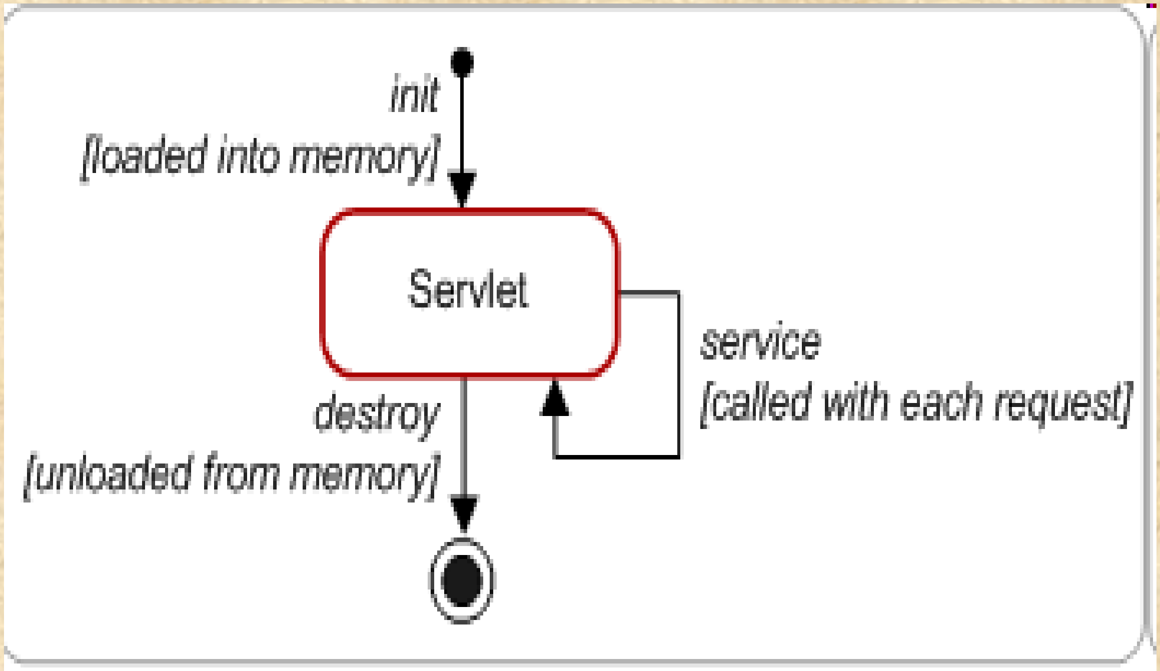
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
response.setContentType(“text/html”);
PrintWriter out = response.getWriter();
out.println(“<HTML>\n“ + “<h1>Hello</h1>”+”</html>”);
}
}
```

servlet Packaging

```
package coreservlets;

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

Servlet Life Cycle



Lifecycle of a Servlet

1) init() method
2) Service() method
3) Destroy() method

Servlet Life Cycle

init

– Executed once when the servlet is first loaded.

Not called for each request.

- **service**

– Called in a new thread by server for each request.

Dispatches to doGet, doPost, etc.

Do not override this method!

- **doGet, doPost, doDelete**

– Handles GET, POST, etc. requests.

– Override these to provide desired behavior.

- **destroy**

– Called when server deletes servlet instance.

Not called after each request.

Single Thread Model Interface

public abstract interface SingleThreadModel

Ensures that servlets handle only one request at a time. This interface has no methods.

If a servlet implements this interface, you are *guaranteed* that no two threads will execute concurrently in the servlet's service method. The servlet container can make this guarantee by synchronizing access to a single instance of the servlet, or by maintaining a pool of servlet instances and dispatching each new request to a free servlet.

This interface does not prevent synchronization problems that result from servlets accessing shared resources such as static class variables or classes outside the scope of the servlet.

Servlet Debugging

1. Use print statement
2. Use integrated debugger in your IDE
3. Use the log file
4. Use Apache Log4J
5. Write separate classes
6. Plan ahead for missing or malformed data
7. Look at the HTML source
8. Look at the request data separately
9. Look at the response data separately
10. Stop and restart the server