**Diffrences between ServletContext & ServletConfig:**

|  |  |
| --- | --- |
| **Servlet Context** | **Servlet Config** |
| SevletContext is an Interface and “an Object of ServletContext” is used by container to pass information to ALL the servlets which are part of an application | ServletConfig is an Interface and “an Object of ServletConfig” used by a container to pass information to a particular Servlet. |
| ServletContext object is created at the time of Server Startup & Garbage collected during the server shutdown | ServletConfig Object is created during the “initialization Phase” of Servlet Lifecycle & Garbage collected during “Desstruction Phase”. |
| So there will be “ONLY ONE Instance of ServletContext object exists per web application” | There will be “ONLY ONE Instance of ServletConfig object exists per Servlet” |
| Hence “Singleton” in Nature | “Non-Sighleton” in Nature |
| ServletContext object is obtained by calling “getServletContext()” method which we inherit from GenericServlet | ServletConfig object is obtained by calling “getServletConfig()” method which we inherit from “GenericServlet” |
| ServletContext context =  getServletContext(); | ServletConfig config = getServletConfig(); |
| In web.xml, context parameters are declared under <context-param> tag (one/more) | In web.xml, servlet config parameters are declared under <init-param> tag (one /more) which is a subtag of <servlet> tag. |
| ServletContext object “does not “ holds the object reference of ServletConfig  ServletContext context = config.getServletContext(); | ServletConfig object holds the object reference of ServletContext  ServletConfig config = getServletConfig(); |

**Note:**

1. Both ServletContext & ServletConfig objects has a method by name getInitParameter() which helps us to get parameter value information from web.xml

Syntax:

String getInitParameter(String paramName)

1. **We can ONLY get ServletContext & ServletConfig parameters at Runtime but “we can not set them”**

**SendRedirect():**

1. User makes the request to a servlet
2. Servlet redirects to another resource (Internal/External) . And Browser gets the URL as a response.
3. Borwser makes a New Request to this URL **When we redirect the request, it is always be a GET request.**
4. Browser displays whatever the response given by the new URL. Redirect happens at Browser side & In case of **Redirect URL in the browser changes.**
5. To redirect the request call sendRedirect() on the HttpServletResponse object.

**Syntax:**

void HttpServletResponse.sendRedirect(String url) throes IOException

Example:-

String url = null;

//Redirect – External URL (can be Static/ Dynamic)

url =” <http://localhost:8080/studentsApp/currentDate>;

url =”currentDateTimme”;

//Redirect- Interal URL –static Resource

url= <http://localhost:8080/studentsApp/index.html>;

url=”index.html”;

resp.sendRedirect(url);

**Create a HTML Form as shown below**

**GraingerSearch.html**

Create a Servlet which gets the request frm this form, takes the “keyword” & display the results for that keyword in [www.grainger.com](http://www.grainger.com) Website.

**FORWARD:**

1. User makes a request to a servlet
2. Servlet internally forwards that request to another servlet by passing Request & Response objects (i.e. forward happens at Server side), Another servlet handles that request & gives back the response.
3. Browser displays the response. In this case Browser will not have any clue on what went behind the scene. Also, URL in the browser doesn’t change.
4. To forward the request call forward() on the RequestDispatcher object.

void RequestDispatcher.forward(ServletRequest req, ServletResponse resp) thorws ServletException, IOException

* We can get the “RequestDispatcher” Object, by invoking “getRequestDispatcher()” Method on “Request” Object

Example:

RequestDispatcher

ServletRequest.getRequestDispatcher(String url)

RequestDispatcher dispatcher = null;

String url = null;

url = “currentDateTime”;//Internal Resource – Dynamic

url = “index.html” ; //Internal Resource – Static

dispatcher = req.getRequestDispatcher(url);

dispatcher.forward(req,resp);

Note:

* We cannot forward the request to External Resources For Example:

String url = “<http://www.google.com>”;

Dispatcher = req.getRequestDispatcher(url);

Dispatcher.forward(req,resp);

Diffrence between redirect and forward:-

|  |  |
| --- | --- |
| **Redirect** | **Forward** |
| 1. Redirect happens @ Browser side | 1. Forward happens @ “Server side” |
| 1. URL in the browser changes | 1. URL does not change |
| 1. We can Redirect the request to “both Internal & External Web Resources” | 1. We can forward the request “ONLY to Internal Web Resources” |
| 1. Redirect contains “More Than One” request & response cycle | 1. Forward contains “ONLY ONE “ request & response cycle |
| 1. In case of Redirect, “more than one set of Request & Response Objects get created | 1. In case of Forward, “ONLY ONE set of Request & Response Object” get Created |
| 1. Slower in operation | 1. Faster in operation |
| 1. Redirect makes request to contain HTTP GET method & hence it ALWAYS invokes doGet() method at destination | 1. When we forward the request, it will invoke the corresponding doXXX() method at destination, |
| 1. Redirect happens on “Response Object” | 1. Forward happens on “Request Object” |