Servlet Context / Servlet Config

Init parameters

- Init parameters refers to the initialization parameters of a servlet or filter. <init-param> attribute is used to define a init parameter. <init-param> attribute has two main sub attributes <param-name> and <param-value>. The <param-name> contains the name of the parameter and <param-value> contains the value of the parameter.

Servlet Context

- Servlet Context is used to communicate with the servlet container to get the details of web application.
- It is created at the time of deploying the project.
- There is only one Servlet Context for entire web application. Servlet Context parameter uses <context-param> tag in **web.xml** file.

Methods	Description
String getInitParameter(String)	It returns the Servlet initialization parameter of the given name and if the requested parameter is not available then it returns null.
Enumeration getInitParameterNames()	It returns names of all Servlet initialization parameters defined in web.xml file.
void setAttribute(String, Object)	Sets an attribute with the current request.
Object getAttribute(string)	Returns the value of an attribute located with the given name or returns null value if an attribute with the given name is not found.
Enumeration getAttributeNames()	Returns all the attribute names in the current request.
void removeAttribute(String)	Removes an attribute identified by the given name from the request.
getRequestDispatcher( )	This method takes a string argument describing the path to located the resource to which the request is to be dispatched..

Use Servlet Context

```
ServletConfig conf = getServletConfig();
ServletContext context = conf.getServletContext();
```

- Here **getInitParameter()** is used to initialize the parameter from the *web.xml* file.

Web.xml

```
<web-app>
    <servlet>
            <servlet-name>ServletExample</servlet-name>
            <servlet-class>com.ServletExample</servlet-class>
  </servlet>
<context-param>
            <param-name>Welcome</param-name>
            <param-value>Welcome to javabeat!!!!!!!!!</param-value>
 </context-param>
<servlet-mapping>
            <servlet-name>ServletExample</servlet-name>
            <url-pattern>/ServletExample</url-pattern>
</servlet-mapping></ web-app>
```

- Here <context-param > tag contains two tags namely <param-name> and <param-value> which is used to initialize the attributes of the servlet.

ServletConfig

- **ServletConfig** Interface and how it works. The config object is created by the web container based on the initialization parameters specified in the deployment descriptor. It is used to send information to a servlet during initialization.
- Ways
  - ServletConfig conf = getServletConfig();
  - public void init(ServletConfig config){}

Methods	Description
String getInitparameter(String Name)	It returns the Servlet initialization parameter of the given name and if the requested parameter is not available then it returns null.
Enumeration getInitparameter Names( )	It returns names of all Servlet initialization parameters defined in web.xml file.
String getServletName()	Returns the servlet name as defined in the deployment descriptor .
ServletContext getServletContext()	Returns the ServletContext object reference.

Example

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
ServletConfig config = getServletConfig();
String S1 = config.getInitParameter("websitename");
out.print("I'm : " + S1 + "website!!!!");
out.close();
```

Web.xml

```
</web-app>
    <servlet>
        <servlet-name>ServletconfigExample</servlet-name>
        <servlet-class>com.ServletconfigExample</servlet-class>
    </servlet>
    <init-param>
        <param-name>websitename</param-name>
        <param-value>Sample</param-value>
    </init-param>
    <servlet-mapping>
        <servlet-name>ServletconfigExample</servlet-name>
        <url-pattern>/ServletconfigExample</url-pattern>
    </servlet-mapping>
</web-app>
```

Here we have defined the <init-param> which defines a param-name as websitename and param-value as Sample.

ServletContext/ServletConfig

- ServletConfig object is one per servlet class
- Object of ServletConfig will be created
  during initialization process of the servlet
- This Config object is public to a particular servlet only
- *Scope*: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.
- We should give request explicitly, in order to create ServletConfig object for the first time

- ServletContext object is global to entire web application
- Object of ServletContext will be created at the time of web application deployment
- *Scope*: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.
- ServletContext object will be available even before giving the first request

ServletInputStream & ServletOutputStream

ServletInputStream

- **javax.servlet.ServletInputStream** and
- **javax.servlet.ServletOutputStream** are abstract classes. These classes define several key methods that the other stream classes implement. Two of the most important methods are **read( )** and **write( )**, which are used for reading the client request and writing servlet response to client respectively.

ServletInputStream

- The *ServletInputStream* class extends *java.io.InputStream*. It is implemented by the servlet container and provides an input stream, that a developer can use to read the data from a client request.
- **Methods of ServletInputStream**
  - **readline( )** method is used to read the input values.

print(boolean value)	Prints boolean value(true/false).
print(string)	Prints string values.
print(char)	Prints a single character .
print(float)	Prints float value(uses 32 bits).
print(double)	Prints double value(uses 64 bits).
print(int)	This prints an integer type value.
print(long)	It prints long value i.e. large value(64 bit).
println(boolean value)	Prints boolean value(true/false).
println(string)	Prints string values.
println(char)	Prints a single character.
println(float)	Prints float value(uses 32 bits).
println(double)	Prints double value(uses 64 bits) .
println(int)	This prints an integer type value.
println(long)	It prints long value i.e. large value(64 bit).

Example for ServletIO

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ServletIOExample extends HttpServlet
{
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws IOException {
                response.setContentType("image/jpeg");
                ServletOutputStream out;
                out = response.getOutputStream();
                FileInputStream img = new FileInputStream("/home/rags/Pictures/Img1.JPG");
                BufferedInputStream bin = new BufferedInputStream(img);
                BufferedOutputStream bout = new BufferedOutputStream(out);
                int ch = 0;
                while ((ch = bin.read()) != -1) {
                        bout.write(ch);
                }
                bin.close();
                img.close();
                bout.close();
                out.close();
        }
```