SERVLETS PART-7

Initialization Parameters:

Initialization parameters are used to initialize servlet.

Initialization parameters are specific to servlet.

To configure initialization parameters we use <init-param><param-name> & <param-value> tags in web.xml.

To retrieve initialization parameters we use getInitParameter() method of javax.servlet.ServletConfig interface in a Servlet.

ServletConfig Example:

web.xml

```
<web-app>

<servlet>

<servlet-name>login</servlet-name>

<servlet-class>login.LoginServlet</servlet-class>

<init-param>

<param-name>driver</param-name>

<param-value>oracle.jdbc.driver.OracleDriver</param-value>

</init-param>

<init-param>

<param-name>url</param-name>

<param-value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
```

```
</init-param>
<init-param>
<param-name>username</param-name>
<param-value>system</param-value>
</init-param>
<init-param>
<param-name>password</param-name>
<param-value>manager</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>login</servlet-name>
<url-pattern>/login</url-pattern>
</servlet-mapping>
</web-app>
```

LoginServlet.java:

```
package loginapp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public LoginServlet() {
      super();
      // TODO Auto-generated constructor stub
   }
   Connection con;
   public void init(ServletConfig config)
   {
	try {
		String s1=config.getInitParameter(“driver”);
		String s2=config.getInitParameter(“url”);
		String s3=config.getInitParameter(“username”);
		String s4=config.getInitParameter(“password”);
		Class.forName(s1);
		con=DriverManager.getConnection(s2,s3,s4);
		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
  }
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		try {
			String s1=request.getParameter("uname");
			String s2=request.getParameter("pword");
			PreparedStatement
pstmt=con.prepareStatement("select * from uinfo where uname=? and
pword=?");
			pstmt.setString(1, s1);
			pstmt.setString(2, s2);
			ResultSet rs=pstmt.executeQuery();
			PrintWriter pw=response.getWriter();
			pw.println("<html><body bgcolor=red
text=yellow><h1>");
			if(rs.next())
			{
				pw.println("Welcome "+s1);
			}
			else
			{
				pw.println("Invalid Username/Password");
			}
			pw.println("</h1></body></html>");
```

```
                    } catch (SQLException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                    } catch (IOException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                    }
          }
}
```

Context Parameters:

Context parameters are also used to initialize servlet.

Context parameters are common to all servlets in a WAR file.

To configure context parameters we use <context-param>, <param-name> & <param-value> tags in web.xml.

To retrieve context parameters we use getInitParameter() method of javax.servlet.ServletContext interface in a Servlet

ServletContext Example:

web.xml

```
<web-app>

<context-param>

<param-name>driver</param-name>

<param-value>oracle.jdbc.driver.OracleDriver</param-value>

</context-param>

<context-param>
```

```
<param-name>url</param-name>
<param-value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
</context-param>
<context-param>
<param-name>username</param-name>
<param-value>system</param-value>
</context-param>
<context-param>
<param-name>password</param-name>
<param-value>manager</param-value>
</context-param>
<servlet>
<servlet-name>login</servlet-name>
<servlet-class>login.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>login</servlet-name>
<url-pattern>/login</url-pattern>
</servlet-mapping>
</web-app>
```

LoginServlet.java:

```
package loginapp;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
    Connection con;
    public void init(ServletConfig config)
```

```
  {
        try {
                ServletContext sc=config.getServletContext();
                String s1=sc.getInitParameter(“driver”);
                String s2=sc.getInitParameter(“url”);
                String s3=sc.getInitParameter(“username”);
                String s4=sc.getInitParameter(“password”);
                Class.forName(s1);
                con=DriverManager.getConnection(s2,s3,s4);
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
  }
        /**
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                // TODO Auto-generated method stub
                try {
                        String s1=request.getParameter("uname");
                        String s2=request.getParameter("pword");
                        PreparedStatement
pstmt=con.prepareStatement("select * from uinfo where uname=? and
pword=?");
                        pstmt.setString(1, s1);
                        pstmt.setString(2, s2);
                        ResultSet rs=pstmt.executeQuery();
```

```
				PrintWriter pw=response.getWriter();
				pw.println("<html><body bgcolor=red 
text=yellow><h1>");
				if(rs.next())
				{
					pw.println("Welcome "+s1);
				}
				else
				{
					pw.println("Invalid Username/Password");
				}
				pw.println("</h1></body></html>");
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
}
```

Initialization Parameters Vs. Context Parameters:

Initialization parameters are specific to servlet where as context parameters are common to all servlets.

To configure initialization parameters, we use <init-param>, <param-name> & <param-value> tags in web.xml where as to configure context parameters, we use <context-param>, <param-name> & <param-value> tags in web.xml.

To retrieve initialization parameters from web.xml, we use getInitParameter() method of ServletConfig interface in a servlet where as to retrieve context parameters from web.xml, we use getInitParameter() method of ServletContext interface in a servlet.

ServletConfig Vs. ServletContext:

ServletConfig is created by web container whenever init() method is called whereas ServletContext is created by web container whenever web application is depolyed on server.

ServletConfig is created by web container one per Servlet whereas ServletContext is created by web container one per WAR file.

ServletConfig is used to retrieve initialization parameters whereas ServletContext is used to retrieve context parameters.

By

Mr. Venkatesh Mansani

Naresh i Technologies