ServletConfig & ServletContext

Agenda

1 **ServletConfig and ServletContext**

Objectives

At the end of this module, you will be able to:

- Use ServletConfig and ServletContext object in web applications
- Create web applications that implement Servlet Chaining

ServletConfig and ServletContext

ServletConfig interface

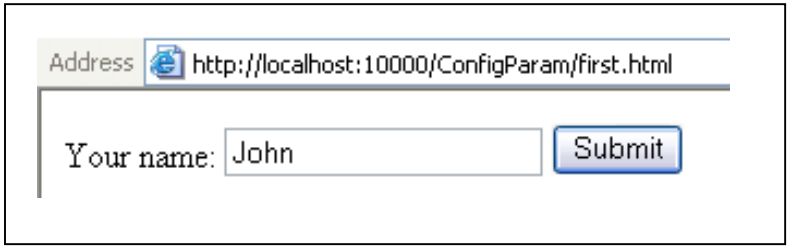
- Provided to a servlet upon initialization by the web container

- Simple read only interface to configuration details
  - String getInitParameter(String name)
  - Enumeration getInitParameterNames()
  - String getServletName ()

- Can also access ServletContext

Demo for using ServletConfig

- Consider an html form “first.html” which accepts the user name



- A servlet “Second.java” takes in this parameter and displays it on the web page

Demo for using ServletConfig (Contd.).

- Let us take a look at Second.java servlet code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Second extends HttpServlet {

    String homeName;
    ServletConfig config;

    public void init() { //get the initialization parameters
        //Returns this servlet's ServletConfig object
        config = getServletConfig();

        /*Returns a String containing the value of the named initialization
   parameter,
         or null if the parameter does not exist. */
        homeName = config.getInitParameter("homeName");
    }
```

Demo for using ServletConfig (Contd.).

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        String urname = req.getParameter("name");

        out.println("<h2>" + homeName + "</h2>");
        out.println("<hr>");
        out.println("Hello!  " + urname);
    }
}
```

Demo for using ServletConfig (Contd.).

Web.xml

```
<web-app>
    <servlet>
        <servlet-name>Second</servlet-name>
        <servlet-class>Second</servlet-class>
        <init-param>
            <param-name>homeName</param-name>
            <param-value>Welcome to www.simple.com</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>Second</servlet-name>
        <url-pattern>/Second</url-pattern>
    </servlet-mapping>
</web-app>
```

The init parameter values are configured in the web.xml deployment descriptor file

Database Example for using ServletConfig

- The init() method can also be used to perform set up operation such as setting up a database connection

```
public class DBConfigParamServlet extends HttpServlet {
    Connection con;
    PreparedStatement st;
    Statement stmt;
    ResultSet rs;
    ServletConfig config;
    public void init() {
        config = getServletConfig(); //Returns this servlet's ServletConfig object
        String driver = config.getInitParameter("driverName");
        String url = config.getInitParameter("urlName");
        try {
            Class.forName(driver);
            con = DriverManager.getConnection(url, "scott", "tiger");
            System.out.println("Connected by using init parameters..");
        } catch (Exception e) {    System.out.println("Error in connection..");  }
    }
   ……doGet()…{} }
```

Database Example for using … (Contd.).

```
<web-app>
    <servlet>
     <servlet-name>DBConfigParamServlet</servlet-name>
     <servlet-class>DBConfigParamServlet</servlet-class>
     <init-param>
          <param-name>driverName</param-name>
          <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
     </init-param>
     <init-param>
          <param-name>urlName</param-name>
          <param-value>Jdbc:Odbc:vdsn2</param-value>
     </init-param>
   </servlet>
   <servlet-mapping>
     <servlet-name>DBConfigParamServlet</servlet-name>
     <url-pattern>/booksconfig.show</url-pattern>
   </servlet-mapping>
</web-app>
```

Discussion

- *What is the advantage of setting up a database connection by reading init parameter values?*

ServletContext interface

- Allows a servlet to communicate with the servlet container
- Access container-managed resources, dispatch requests, write to logs
- Defines a set of methods that a servlet uses to communicate with its servlet container

  For example, to get the MIME type of a file, dispatch requests, or write to a log file
- There is one context per "web application" per Java Virtual Machine
- ServletContext object is contained within ServletConfig object, which the servlet container provides the servlet when the servlet is initialized

Servlet Context Methods

- Resources such as index.html can be accessed through web server or by servlet
  - Servlet uses *request.getContextPath()* to identify its context path, for example: /app
  - Servlet uses *getResource()* and *getResourceAsStream(request.getContextPath() + "/index.html")*

- To retrieve context-wide initialization parameters, servlet uses *getInitParameter()* and *getInitParameterNames()*

- To access a range of information about the local environment, shared with other servlets in same servlet context, servlet uses *getAttribute() ,setAttribute(), removeAttribute(), getAttributeNames()*

Demo for using ServletContext

- Use of ServletContext for web application initialization: Suppose there is a need to include a contact email of webmaster or an admin on few web pages of a website

- In the web.xml:

```
<servlet>
    <servlet-name>ContextParamServlet</servlet-name>
    <servlet-class>ContextParamServlet</servlet-class>
</servlet>
<context-param>
    <param-name>Email</param-name>
    <param-value>webmaster@simple.com</param-value>
</context-param>
```

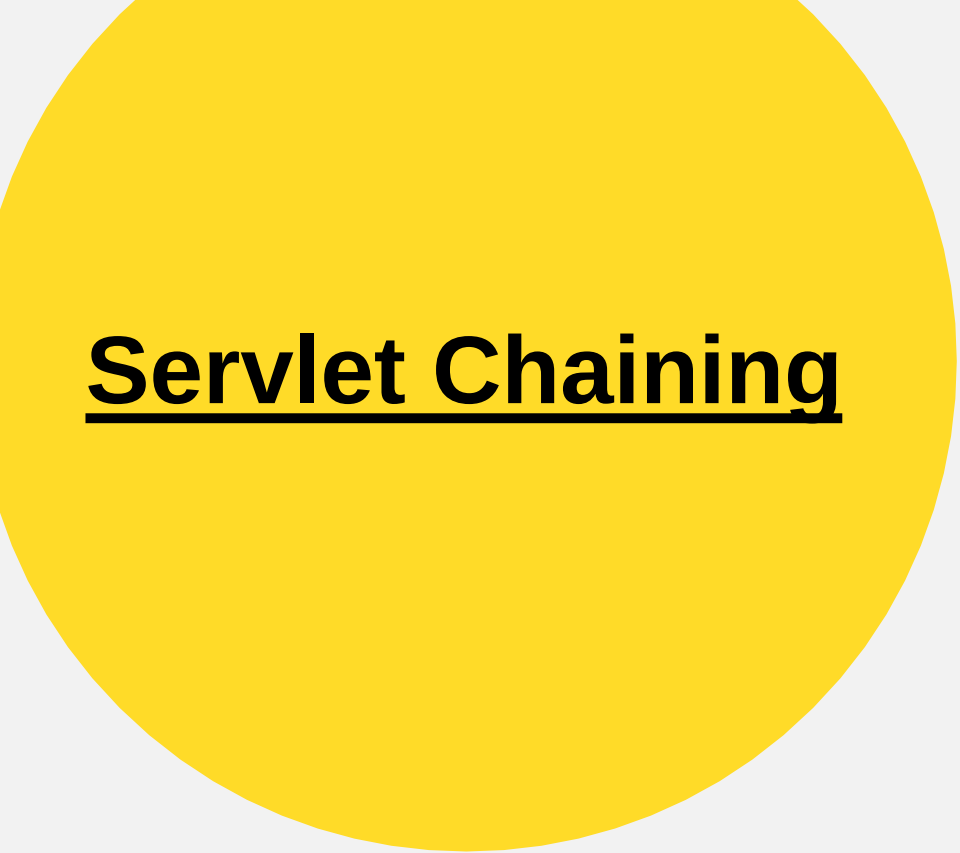
- In the servlet code:

```
ServletContext context = getServletContext();
out.println(context.getInitParameter("Email"));
```

Checkpoint

- *In which tag is the <context-param> tag defined in web.xml file?*

Servlet Chaining



Servlet Chaining: RequestDispatcher Interface

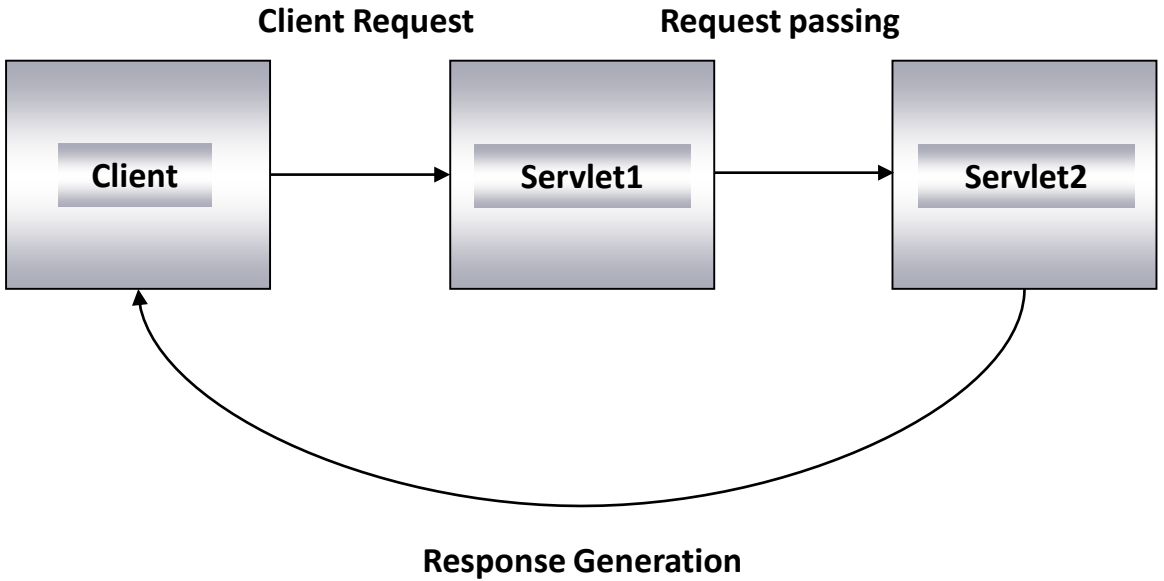
- Used in order to FORWARD or INCLUDE a request from one servlet to another
- Servlet/JSP the RequestDispatcher interface provides two methods.
    1. RequestDispatcher.forward(request,response)
    2. RequestDispatcher.include(request,response)

- Both these methods take ServletRequest and ServletResponse object as an argument

Servlet Chaining: forward (request,response)

```
ServletContext ctx=getServletContext();
RequestDispatcher
  dis=ctx.getRequestDispatcher("/servlet/AnotherServlet");
dis.forward(request,response);
```

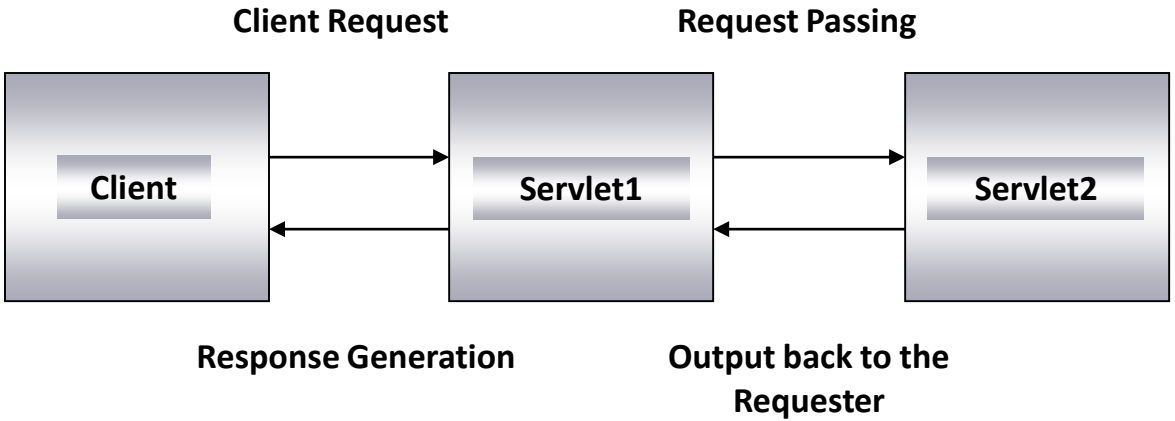
Servlet Chaining: forward (request, response)



Servlet Chaining: include (request, response)

```
ServletContext ctx=getServletContext();
RequestDispatcher
  dis=ctx.getRequestDispatcher("/servlet/AnotherServlet");
dis.include(request,response);
```

Servlet Chaining: include (request, response)



Demo for Servlet Chaining

- This example demonstrates chaining in servlets where output of the first servlet act as a input to the second servlet.
  - ***first.html*** – a form containing a text field and a submit button
  - ***FirstServlet.java*** - accepts user name and forwards it to SecondServlet
  - ***SecondServlet.java*** - Extracts the username value which is set in FirstServlet

Summary

In this module, you were able to:

- Use ServletConfig and ServletContext object in web applications
- Create web applications that implement Servlet Chaining
- Develop web applications that use Cookies
- Implement Session tracking in web applications

References

1. Oracle (2006).Understanding and using Servlet Sessions. Retrieved April 26, 2012, from, http://docs.oracle.com/cd/B31017_01/web.1013/b28959/sessions.htm

2. Tutorial Point (2012). JSP - Cookies Handling. Retrieved April 26, 2012, from, http://www.tutorialspoint.com/jsp/jsp_cookies_handling.htm

3. JavaTPoint (2012). ServletConfig Interface. Retrieved April 26, 2012, from, http://www.javatpoint.com/sonoojaiswal/servletconfig

Thank You