SERVING INTERACTIVE WEB PAGES TO TechSat21 CUSTOMERS

Lance Self
Air Force Research Laboratory
Space Vehicles Directorate
Kirtland AFB, NM
Lance.Self@kirtland.af.mil

ABSTRACT

TechSat21 is an innovative satellite program sponsored by the Air Force Research Laboratory Space Vehicles Directorate and the Air Force Office of Scientific Research. Its mission is to control a cluster of satellites that, when combined, create a "virtual satellite" with which to conduct various experiments in sparse aperture sensing and formation flying. Because TechSat21 customers have a need to view very large data sets that vary from the payload to the satellite state of health[1] a modern viewing method using Java Server Pages and Active Server Pages is being developed to meet these interactive dynamic demands.

KEY WORDS

TechSat21, Java Server Pages (JSP), Active Server Pages (ASP), Extensible Markup Language (XML), Internet Information Service (IIS).

INTRODUCTION

Java Server Pages (JSP) was first introduced by Sun Microsystems in 1998 to provide an easy means to create information rich, dynamic web pages. At the time, Microsoft had already deployed its Active Server Pages (ASP) that was also used for dynamic web page creation. Sun developed JSP as an extension to Java Servlets or "server side" Java applets. Servlets were, however, difficult to create and maintain. With the introduction of JSP, Sun was able to effectively compete with Microsoft's ASP while maintaining the "write once, run anywhere" advantage of Java.

JSP pages use tag based syntax similar to Hypertext Markup Language (HTML). The JSP pages themselves are a combination of scripts, HTML, Extensible Markup Language (XML), and Java, all of which allow web page developers to create and maintain dynamic web content while simultaneously providing access to all Java classes and functions. This combination gives JSP the power of an object-oriented language that encapsulates the logic that generates page logic coupled

with the ease of scripts. The data processing logic resides on the server while the page formatting (HTML, XML) is passed back to the browser thus separating data processing from data display. With this combination JSP can be a useful tool for displaying telemetry data points, via a web page, while separating the data content generation from the user interface.

JSP PROTOCOL

With simple HTML protocol (Figure 1) the client requests a web page from the server. The server simple retrieves the requested page and sends it back to the client browser.

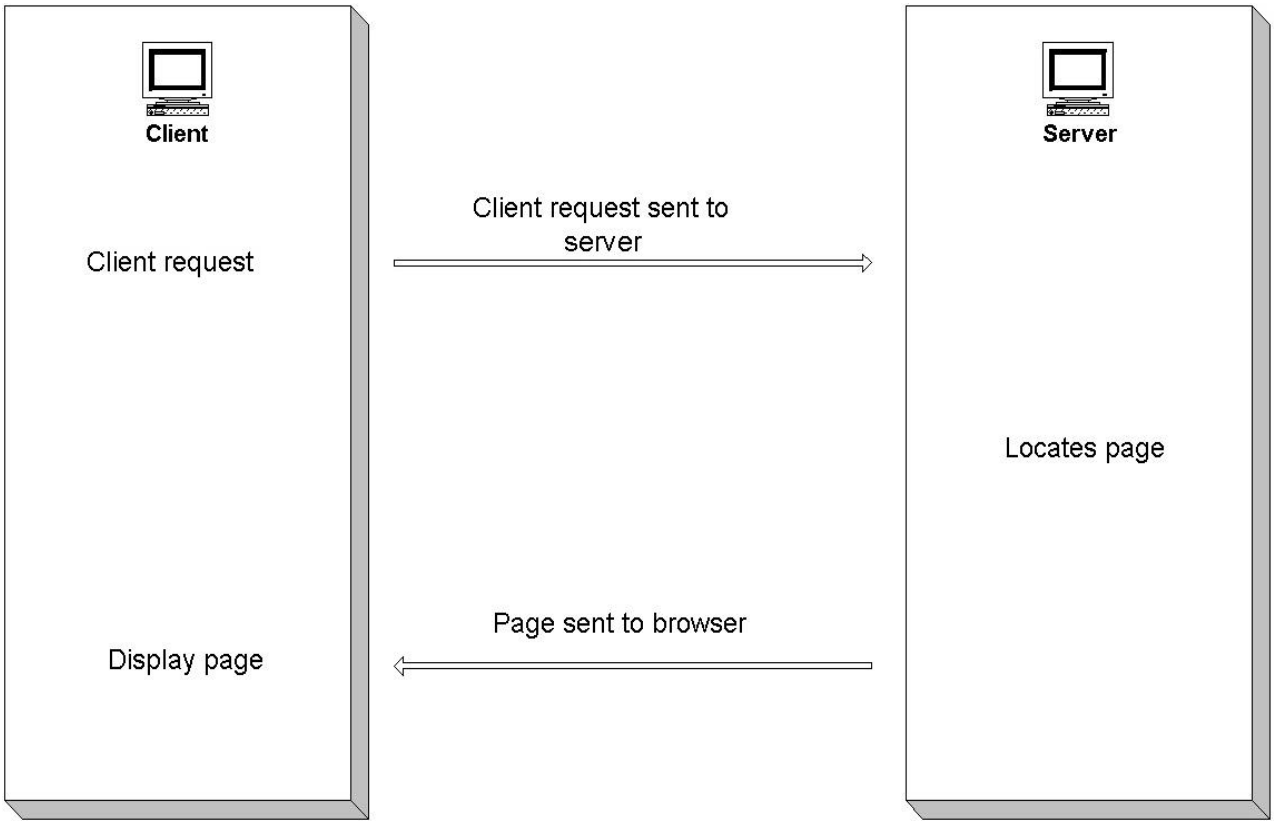

Figure 1

As shown in Figure 2, the JSP protocol resembles the typical client/server protocol. At the browser the user (client) makes an entry into a form. The entry typically triggers a request for information from the server, such as placing an order.

The information request is sent to the server where the server interprets the request as JSP code and locates the JSP source file residing on the server. The JSP engine, Tomcat for example, translates the JSP program into a Java class that is compiled into a Java servlet program. The Java servlet is what actually processes the request and the results of the processing are sent back to the client as HTML, or XML, where it is formatted and displayed.

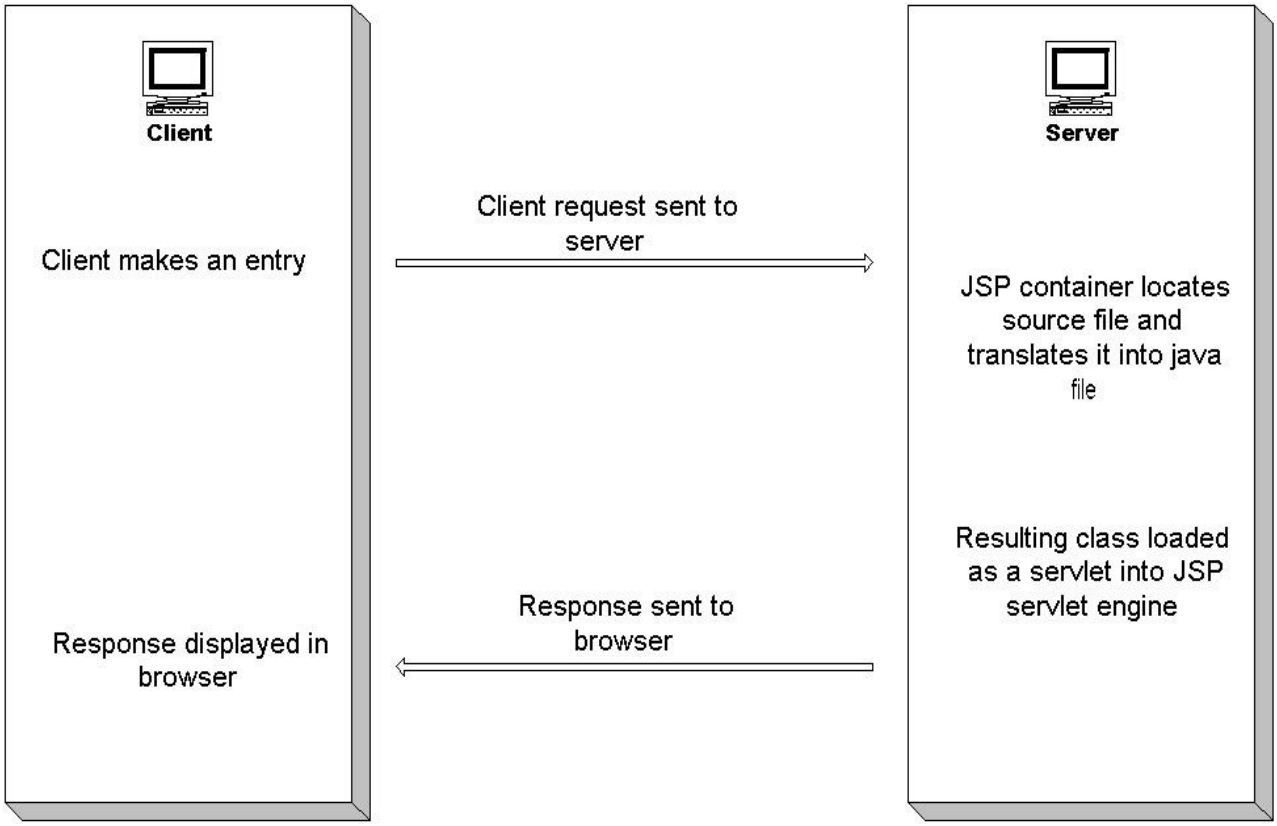

Figure 2

JSP pages look like HTML pages but JSP is compiled into Java servlets when they are instantiated. They are a combination of HTML and JSP that, when combined, give users the dynamic content, e.g., the server alters the file before sending it back to the server. JSP use of Java Servlets means, to the developer, that the full use of Java API's are available giving the client web page more utility and power.

COMPARING MICROSOFT ASP WITH SUN JSP

Both Microsoft ASP and Sun JSP have similar goals, which is to provide dynamic, interactive content to web pages. However, Microsoft and Sun attain these goals in a different manner. Table 1 lists some of these differences.

	Microsoft Active Server Pages	***Sun Java Server Pages***
Web Server	Microsoft IIS or Personal Web Server	Any Web server
Platform	*Microsoft Windows	Windows, Solaris, MacOS, Linux
Components	COM/DCOM	Java Beans, JSP Custom Tags
Memory Leak Protection	No	Yes

Base Language	VBScript, JScript	Java
Custom Tags	No	Yes
Database Access	Yes (Active Data Objects)	Yes (JDBC API)
Code Interpretation	Each Instance	Once
Industry Support	Extensive	Extensive

Table 1

***** *Microsoft is due to release ASP+ which is expected to support across platforms.*

Microsoft uses ActiveX controls and for this reason is somewhat restricted to Microsoft platforms. However, third party products are readily available to port ASP to other platforms. Interestingly, one company, Halcyonsoft (http://www.halcyonsoft.com), ports Microsoft COM components to Java Beans in order to port COM to other platforms.

JSP provides memory clean up through its base language, Java, which has automatic memory clean up. To developers this means they do not have to keep track of the objects created and delete or release the memory those objects use.

JSP is interpreted once to Java byte code and is only re-interpreted when the JSP file is modified. In practical terms this means there is a delay for the first request going from the client to the server. Subsequent requests do not suffer this delay because they go to the servlet byte code, which is already in memory.

Microsoft ASP is optimized for Windows. If the environment you are developing for is a Windows environment then ASP is likely the best use of resources. On the other hand if your environment consists of a variety of platforms and operating systems then JSP should be considered a serious contender to ASP.

JSP SYNTAX

JSP syntax can be broken down into these elements; Directives, Declarations, Expressions, Scriplets, and Standard Actions.

The Directives are:

> "used to set global values such as client declaration, methods to be implemented, output content type, etc. They do not produce any output to the client."[2]

All JSP directives are enclosed within the **<%@ … %>** tag.

The ***page*** directive is used to define attributes that affect the whole page and its' syntax is:

<%@ page *ATTRIBUTE* %>

where *ATTRIBUTE* = language, extends, import, session, buffer, autoFlush, isThreadSafe, info, errorPage, contentType, charset, isErrorPage.[3] For example,

<%@ page import="java.util.*" buffer="12kb"%>

is interpreted to include the "java.util.* class library and sets the page buffering to 12K.

The ***include*** directive is used to include files and its' syntax is:

<%@ include file="Filename" %>

For example, <%@ include file="/example/myfile.html"%>

JSP Declarations allow the developer to define page level variables and are enclosed within the **<%! … %>** tag. Declarations are initialized when the JSP page is initialized.

```
<html>
<body>
<h1> Expression Example </h1>
</html>

<%!    int i = 1;
       private int j;
       public void myMethod() { … }
%>
```

JSP Expressions outputs a value back to the client, the result being converted to a string. Expressions are enclosed within **<%= … %>**. The following snippet would increment the variable 'i'.

```
<html>
<body>
<h1> Expression Example </h1>

<%!    int i = 0; %>
<%     i++; %>
<%=    "My counter is " + i %>

</body>
</html>
```

JSP Scriptlets are blocks of Java code fragments that are executed at request time. Scriptlets are enclosed within **<% … %>** and may contain any valid Java code.

```
<html>
<body>
<h1> Scriptlet Example </h1>

<%     for (int i=0; i<5; i++)
```

```
            {
                System.out.println (" i is: " + i);
            }
        %>
```

</html>

To use JSP you will need to install some software on your system. Because you are creating programs that roughly follow the client/server paradigm you will need a web server. I used Apache version 1.3 for my (local) web server. You will also need a JSP server that recognizes JSP pages. I installed a local JSP server that is available from Tomcat (version 3.2.1). Both pieces of software are open source. Detailed installation instructions can be found at:

http://www.apache.org/

and

http://developer.java.sun.com/developer/onlineTraining/JSPIntro/exercises/SetupTomcat/help.html

Various sources claim that Sun JSP will run on Microsoft Internet Information Service (IIS) thereby eliminating the need to install Apache and Tomcat on Windows systems. To test this claim I downloaded a simple JSP example from Sun. While it ran correctly using Tomcat it did not display correctly when I used IIS as the server.

DATABASE EXAMPLE

One of the more common uses of JSP (or ASP) is to retrieve information stored in a database. Incorporating database information into the web page enhances and creates a meaningful interactive web page. Telemetry data points, stored in a database, can be accessed using finer and finer search mechanisms to return the data that the user is interested in viewing.

TechSat21, a new satellite project for the Air Force Research Laboratory (AFRL), is taking a modern approach towards satisfying customer needs with respect to accessing data. Customers will be viewing data via the AFRL intranet and will be ordering their data sets on CD's or downloading them directly from the web site. Figure 3 is an example of the type of web pages being made available to TechSat21 customers.

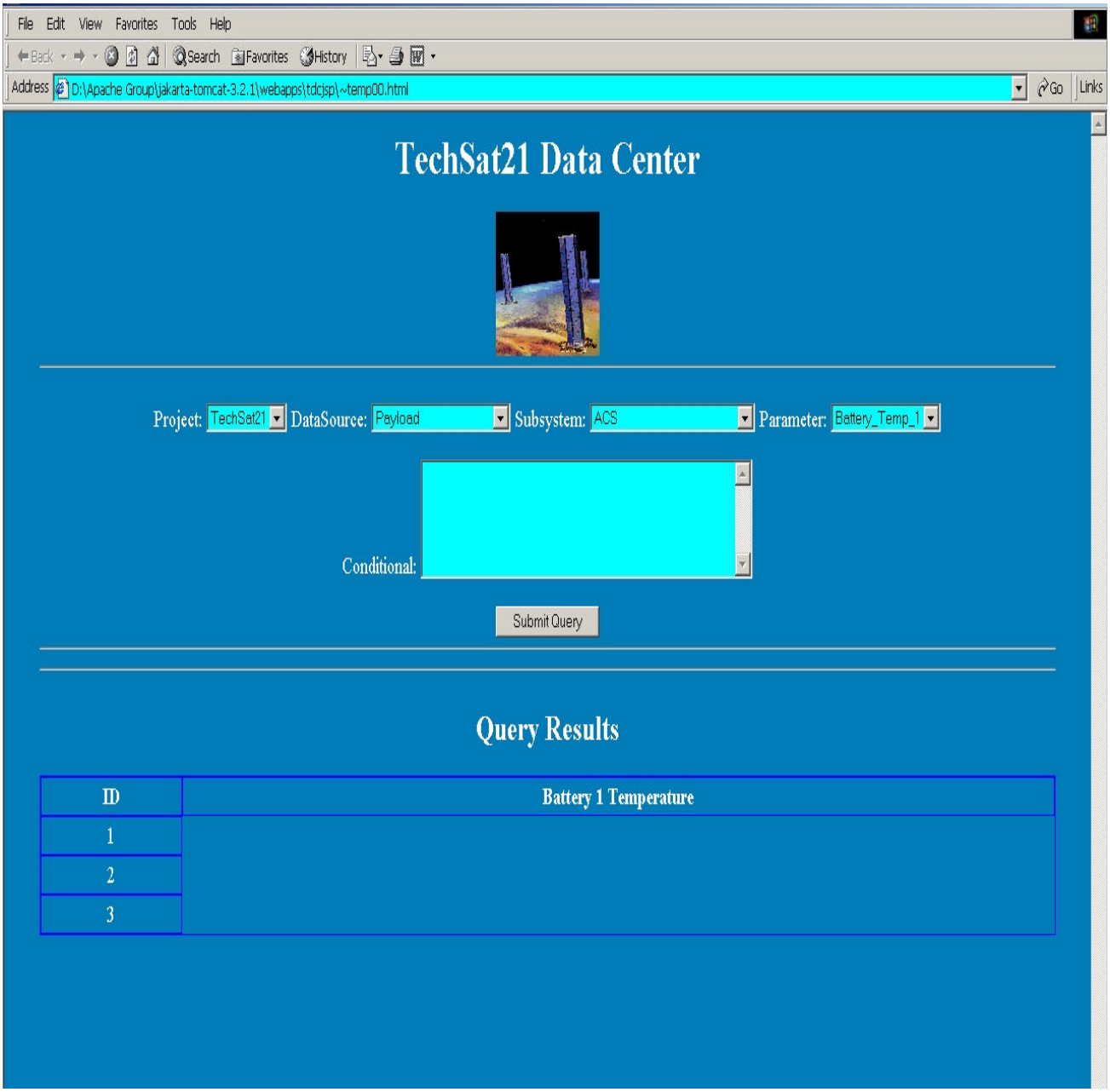

Figure 3

The TechSat21 customer uses the list boxes (Project, DataSource, etc.) to refine the search down to the individual parameter level. Depending upon the search criteria, the user may also enter a conditional statement such as ">30" in the text area. The table at the bottom of the page is where query results are displayed. This table expands and contracts as needed, another feature of an interactive, dynamic web page.

The first step is to create an ODBC connection using the built in *ODBC Administrator* which is a Windows 2000 (also NT) operating system component. Following the prompts, the correct drivers were loaded, and a new data source created, that are tied to the data base file.

This example uses a connection to Microsoft SQL Server 2000 database management system (DBMS). The directive

<%@ page import java.sql.* %>

is part of the JSP file and is used to import the Java SQL library. Once the library is imported this example uses code to connect to the DBMS, which is also built into the JSP page, and in this case uses the JDBC-ODBC bridge to connect Sun JSP to Microsoft SQL Server. When the "Submit" button is pressed a JSP program collects the selected parameters ("TechSat21", "Payload", "ACS", "Battery_Temp_1") that define the SQL query criteria.

Finally, the query is sent to the server where the DBMS resides. The DBMS parses the query, gathers the results, and sends the results to the JSP page where it is formatted and displayed in the table.

The preceding section described one example of using interactive web pages. Other uses such as inventory monitoring, trend analysis, or any application that requires data being updated on a regular, non-real time, basis, and viewed by geographically separated customers, are applications that fit well into the interactive paradigm.

CONCLUSION

Dynamic web pages are the next step in the data display evolution. Customers are asking for some measure of control over what, and how, data is viewed and Java Server Pages takes a step towards that control. With the power of an object-oriented language behind it, JSP is in a position to extend the versatility of existing web pages beyond existing static displays.

REFERENCES

[1] Self, Lance. "TechSat21 Testbed Database." Proceedings of the International Telemetering Conference, Vol. XXXVI, Instrument Society of America, November 2000.

[2] Avedal, Karl, et al. Professional JSP. Birmingham: Wrox Press, 2000.

[3] http://java.sun.com/products/jsp/syntax.pdf