APPLYING INTERACTIVE WEB PAGES

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

ABSTRACT

Visitors to web pages are, in most cases, restricted to viewing information the page designer has anticipated they will be interested in viewing. Many times this is adequate, but there are instances where the visitor wants the information they view to be based on selections they choose. The Air Force Research Laboratory (AFRL) Space Vehicles Directorate anticipates selected customers will have a need to view very large data sets that vary from the satellite payload to the satellite state of health[1], and will require controlling what they view in an "ad hoc" manner. In response, AFRL is using Java Server Pages developed within the data center to bring interactive and dynamic web page content to these customers.

KEY WORDS

Java Server Pages (JSP), Extensible Markup Language (XML).

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. This model separates data processing from data display making JSP a useful tool for displaying telemetry data points.

JSP PROTOCOL

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

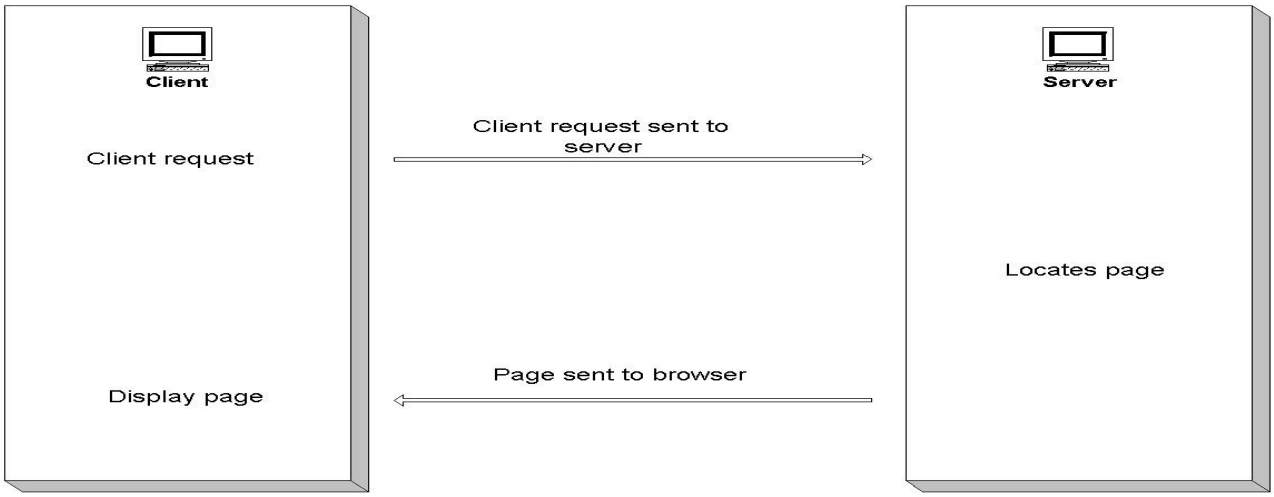

Figure 1

The JSP protocol (Figure 2) starts off resembling the typical client/server protocol. At the browser (client) the user makes an entry into a form. The entry typically triggers a request for information from the server. 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 source 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.

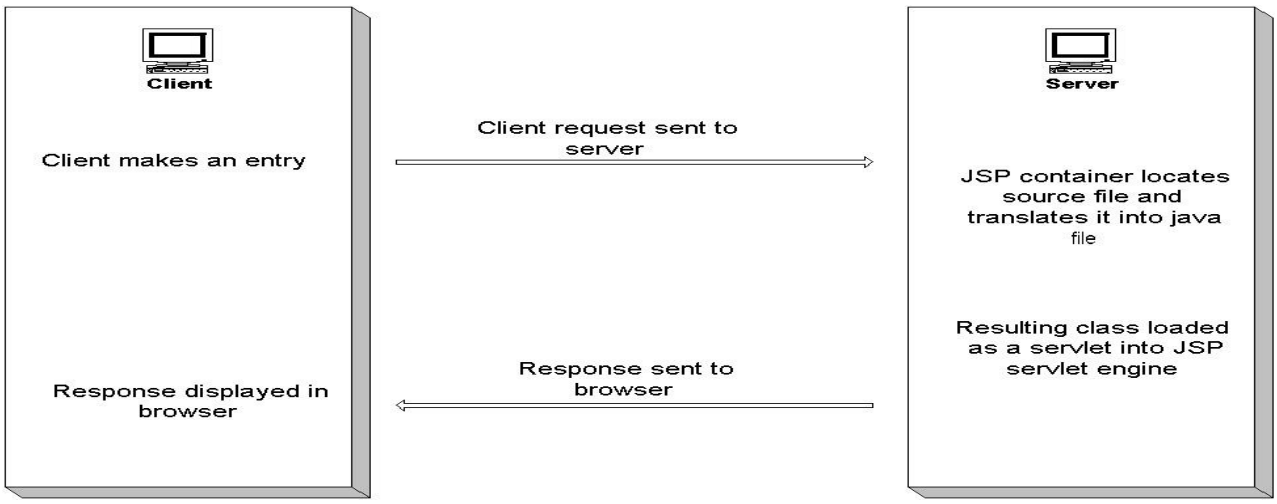

Figure 2

JSP pages are compiled into Java servlets when they are instantiated. They are a combination of HTML, XML, and JSP that give users the dynamic content. For the developer, full use of Java API's are available that gives client web pages data processing utility and power.

DATABASE EXAMPLE

Avedal[2] discusses two models in creating dynamic GUI's, the "Co-Branding" model that customizes the look of the GUI and the "Web Portal" model that customizes the content of the web page. The Co-Branding model is likened to Windows NT or 2000 users being able to customize the wallpaper or screen saver when they logon to their desktop, and will not be used at AFRL. The latter model, however, is being used at AFRL to develop interactive web pages for data center customers.

The Web-Portal model can be either cookie or database driven. We have opted to make our web pages database driven, but the concept is identical in either case. The idea behind this model is to create web pages whose **content is altered** by selections made by the web customer, thus giving customers meaningful information which they request and not what the web page designer *anticipates* they will request.

Using the Portal approach, a request is sent to the database server (see Figure 2) consisting of what the user wants to query. JSP instantiates the Java program and the database is queried. Figure 3 is sample code from a related project that executes a SQL "Select" function. The results of the query are returned from the database management system (DBMS), via the ResultSet function, and sent back to the web page where it is formatted for display.

This example embeds the SQL query within the Java source file which is not an optimal approach. Future development will have the customer building the SQL query on the web page, and that query will be sent to the DBMS.

```
// Load the JDBC driver
  LoadJDBC loadjdbc = new LoadJDBC();
  loadjdbc.LoadDriver();
   ResultSet rs = null;
// Grab the entries in the GUI
   while (entries != NULL)
   {
  String entry 1 = First Entry.getText();
   }
// Build the SQL statement
  String the_query = " SELECT * FROM My_Table;"
// Execute the query
  Statement statement = con.createStatement();
// Results of the query
  rs = statement.executeQuery (the_query);
```

Figure 3

Figure 4 is a design of the type of web pages being planned for data center customers and illustrates the "drilling down" search mechanism.

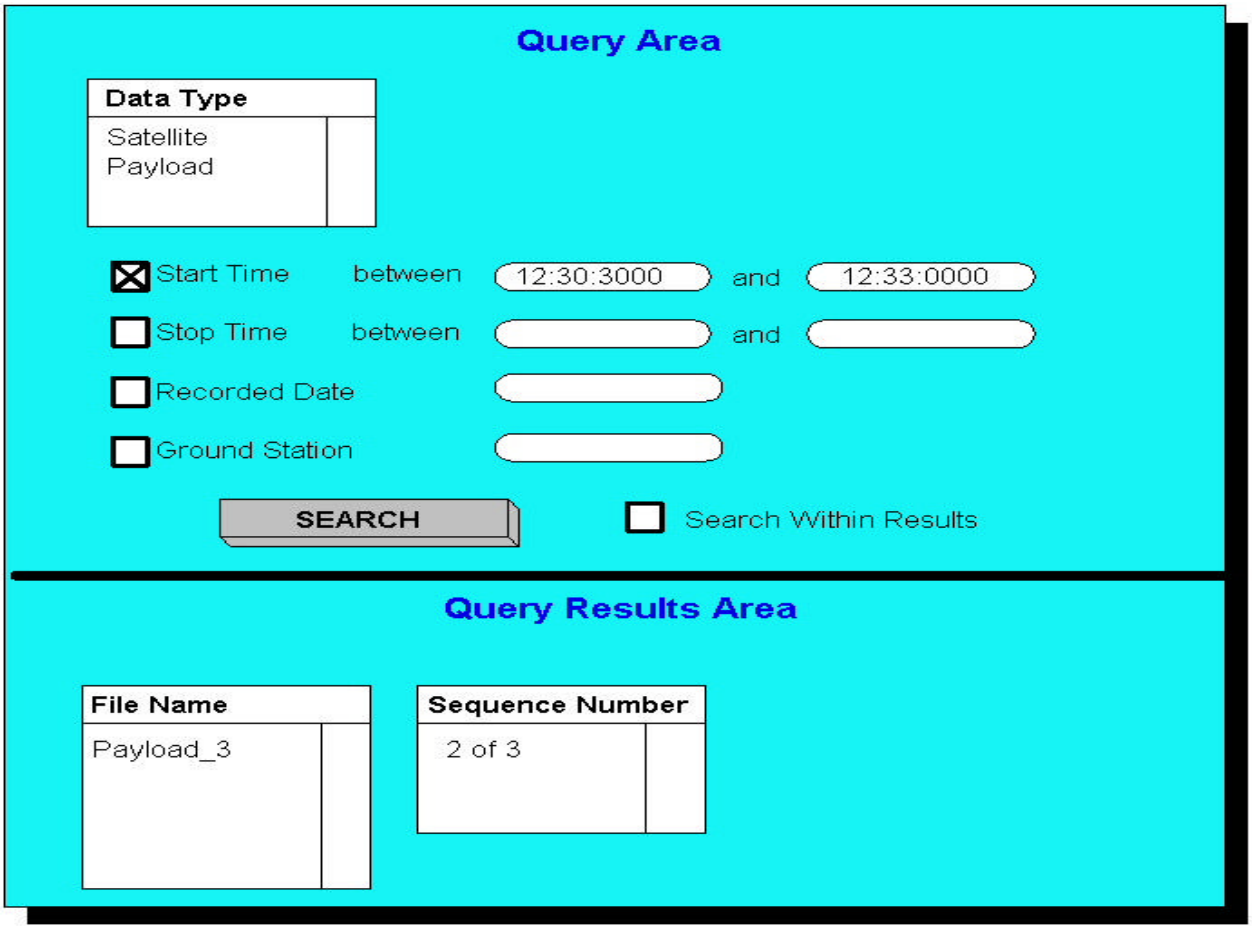

Figure 4

Customers currently have two types of data they can retrieve from the data center. If they select "Payload" the area directly below the Data Type list box is populated with "Payload" options. In this example, the customer wishes to search the database using "Start Time" between "12:30:3000 and 12:33:0000." After pressing the "SEARCH" button the request is sent to the DBMS and the results, Payload_3 and 2 of 3, are returned and displayed at the bottom of the web page in the "File Name" and "Sequence Number" list boxes respectively. These results that were returned, as a result of the query, are table entries in the "Catalog" table within the data center DBMS (Table 1).

File Name	Recorded Date	Ground Station	Start Time	Stop Time
Satellite_2	01/23/2000	Hawaii	10:15:0000	10:17:4000
Payload_3	01/23/2000	Florida	12:27:0000	12:32:0000
Payload_4	01/23/2000	Florida	12:32:0000	12:37:0000

Table 1: Example Catalog table entries

If the Start Time had been between "10:15:0000 and 12:34:0000" then the query would have returned three file names that would be listed in the File Name list box ("Satellite_2, Payload_3, Payload_4").

The user can refine the search by checking the “Search Within Results” check box, changing the time span, and searching again. These new results would replace the “Payload_3” and “2 of 3” currently being displayed.

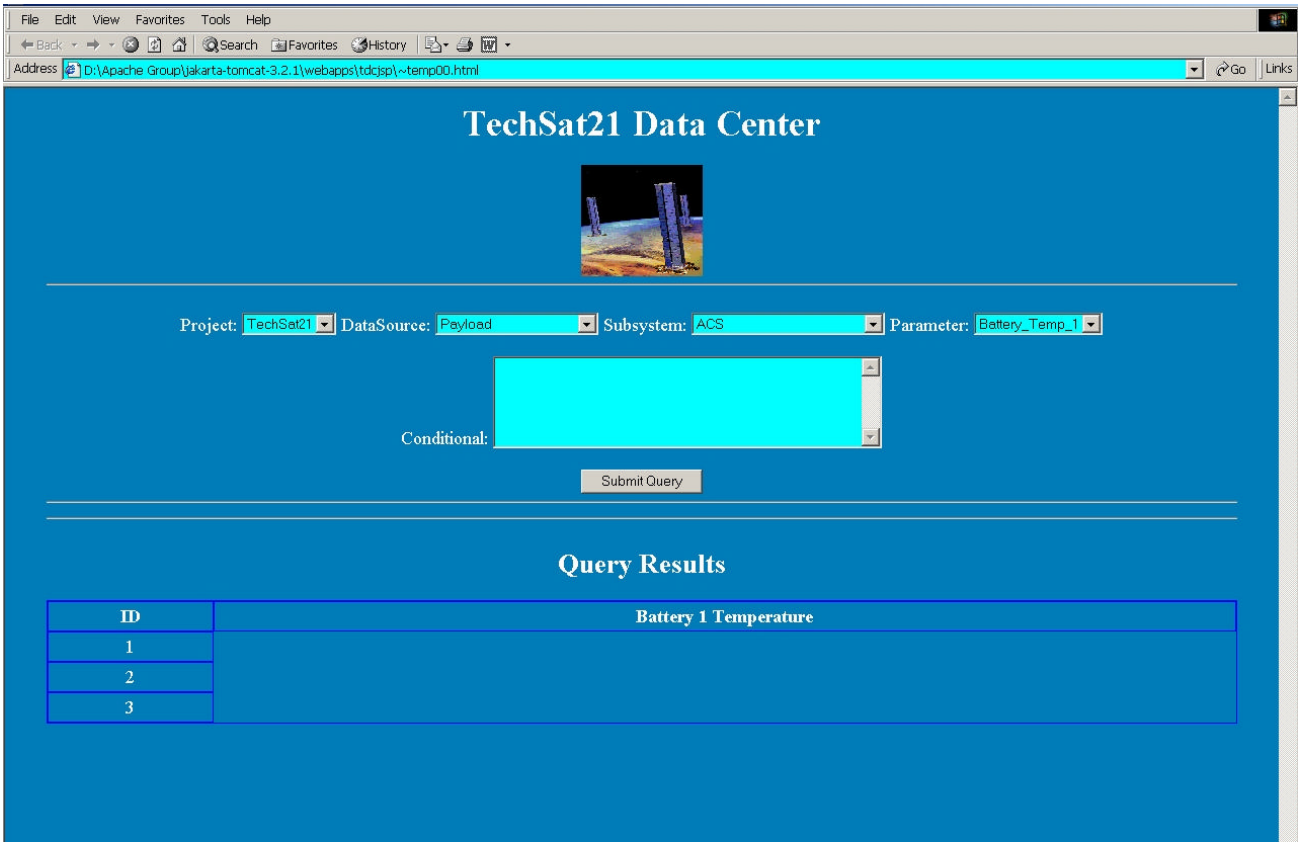

Figure 5

Figure 5 illustrates a slightly different approach to applying the Portal model. Where the previous example returned “File Names” that matched the query, this web page returns the *contents* of the file that match the query. It does this by building, displaying, and populating a series of list boxes that are dependent on the previous list box selection the customer makes. The **DataSource** list box, for example, displays “Payload.” As a result of “Payload” being selected, the **Subsystem** list box displays “ACS.” If another **DataSource** were selected, then the **Subsystem** list box would be populated with an entirely different list of choices. The choices presented to the visitor would come from a different table within the database. Just as in the previous example, the contents of the list box are dynamic and interactive, and give the visitor control over the web page.

Similar approaches can be used regardless of the application. Telemetry data points, for example, stored in a database, can be accessed using finer and finer search mechanisms to return the data set that the user is interested in viewing. Both examples illustrate a specific satellite related application of dynamic, 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, all fit into this interactive paradigm.

CONCLUSION

Dynamic web pages are the next step in the web page evolution. Customers are asking for some measure of control how data is presented to them and control over what they see. Java Server Pages delivers the power of an object-oriented language coupled with the ability to extend the versatility of existing web pages beyond 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.