Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today

Level: Advanced
Works with: Extended Search
Updated: 04-Nov-2002

by
Brett
Morris

How easy is it to customize Lotus Extended Search? With some HTML or JavaServer Pages (JSP) experience, you can create custom Web-based templates for Lotus Extended Search. In **Part 1** of this two part article, we explained how to create custom Extended Search Web templates using HTML and the custom Extended Search tags. We also described the Java servlets and Java Beans used by the search templates. In Part 2, we explain how to build custom Extended Search Web templates using JavaServer Pages (JSP) tags. You can download the templates described in this article from the **Sandbox**. This article assumes knowledge of JavaServer Pages and HTML. It also assumes that you have experience with Lotus Extended Search.

The JSP interface

In Part 1, we used the JKM (Java Knowledge Management) interface to create custom search templates using HTML. The JKM interface uses the JKMSearchController servlet and several Java Beans to support the HTML interface. The HTML interface is supported by Lotus Domino and IBM WebSphere application servers.

In addition to the HTML interface, Extended Search provides Java Beans that support the JSP (JavaServer Pages) interface. This interface lets you use the power of the Java programming language within your Extended Search Web templates. The Extended Search package includes examples that support the older JSP version 1.0 as well as examples that implement techniques introduced by the latest JSP processors (1.1+). The functionality introduced by version 1.1 allows for the use of JSP tag libraries. These libraries simplify the coding and creating of the Web templates and make it easier for nonprogrammers to create Web templates.

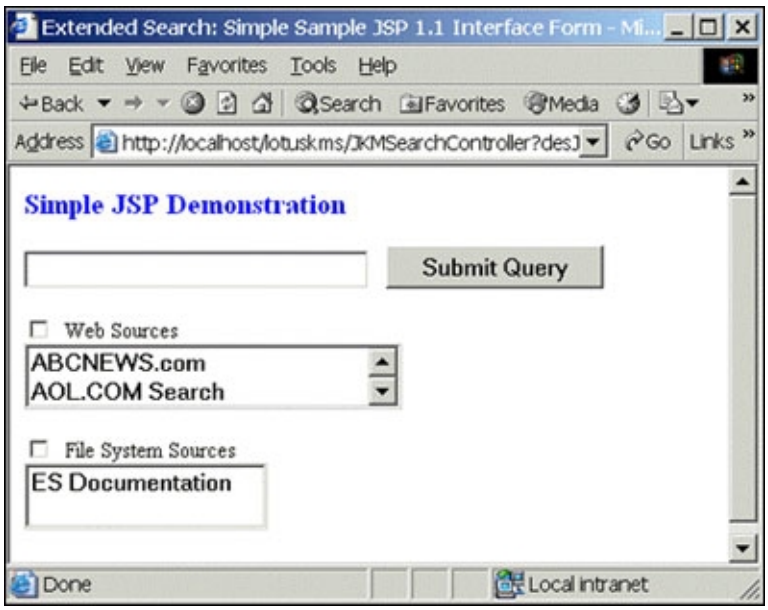
The Extended Search JSP interface requires WebSphere Application Server 3.5 (or later) Advanced edition. Domino 5.0x, which Extended Search 3.5 and 3.7 support, does not support JSPs. Refer to the **Extended Search Installation Guide** for your release to find out which release of WebSphere Application Server you need.

In this article, we will analyze a version of our Web template sample from the previous article that uses the Extended Search JSP Java Beans and tag libraries (JSP 1.1+). A version of the JSP template that does not use the tag libraries (JSP 1.0) is also available for download. To access these templates, you need to extract the files from the downloadable zip file (sampleTemplates.zip) and place them in your Extended Search JSP directory (e.g. /lotuskms/template/enUS/JSP). You can invoke this example by using the following URL:

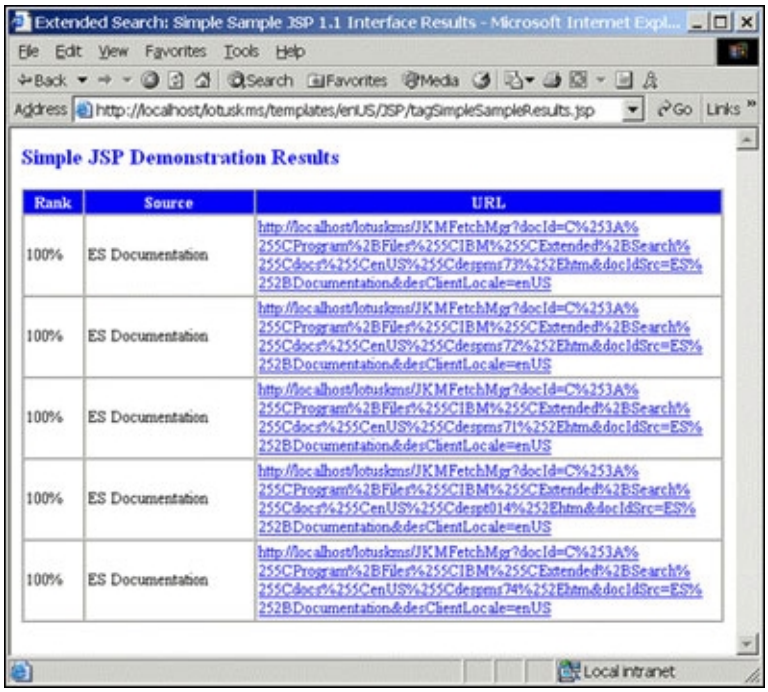
http://<your server name>:<port>/lotuskms/JKMSearchController?desJSPFile=tagSimpleSample.jsp&desClientLocale=enUS&AppID=<your application>

Replace <your server>:<port> with the hostname and port of your Web server. Likewise, replace <your application> with the name of the application you want to load (for example, Demo).

After you load the template, you view the initial screen:



As you can see, this example looks exactly like the example we discussed for the JKM interface in the previous article. There is a text box to enter the query string and a table of categories and sources. The results page is set up in the same way as the JKM example and includes a list of hits ordered by document rank.



Now let's look at the JSP 1.1+ code that was used to create this example.

Creating the simple search form

There are two JSP template pages used for the sample. One page displays the query criteria (query string, categories, and so on), while the other page runs the search and displays the results. Let's look at the query criteria page tagSimpleSample.jsp.


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<%@ page language="Java" %>
<%@ page contentType="text/html; charset=iso-8859-1" %>
<%@ page errorPage="errorpage.jsp" %>
<%@ taglib uri="/estaglib.jar" prefix="lotuskms" %>

<HTML>
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
    <TITLE>Extended Search: Simple Sample JSP 1.1 Interface Form</TITLE>
</HEAD>
<BODY>

    <jsp:useBean id="dir" class="com.ibm.bts.es.srv.beans.jsp.EsHTMLDirectoryBean"
    scope="session"></jsp:useBean>
    <jsp:setProperty name="dir" property="*"/>

    <FORM NAME="_QUERY" ACTION="<%= dir.getJSPURI() %>tagSimpleSampleResults.jsp"
    METHOD="POST">
        <!-- Title -->
        <FONT COLOR="BLUE" SIZE="5"><B>Simple JSP Demonstration</B></FONT><BR><BR>

            <INPUT TYPE="TEXT" NAME="DESQueryString" VALUE="" SIZE="30">
              <INPUT TYPE="SUBMIT" VALUE="Submit Query">
            <BR><BR>

            <% String applID = (String)request.getParameter("AppID"); %>
            <lotuskms:categories appID="<%= applID %>">
            <INPUT TYPE="CHECKBOX" NAME="DESCategoryNames" VALUE="<%= esCategoryName %>">
             <%= esCategoryName %><BR>
            <SELECT NAME="DESDataSourceNames" SIZE="2" MULTIPLE>
            <lotuskms:sources appID="<%= applID %>">
            <OPTION VALUE="<%= esSourceName %>"><%= esSourceName %></OPTION>
            </lotuskms:sources>
            </SELECT>
            <BR><BR>
            </lotuskms:categories>

    </FORM>
</BODY>
</HTML>
```

Invoking the JSP tag library

At the top of the page, the first thing to note is the group of JSP directives. The most important one is the directive specifying the jar file containing the Extended Search tag library definitions. This directive must be on the page if you want to use the JSP tag libraries.

```
<%@ page language="Java" %>
<%@ page contentType="text/html; charset=iso-8859-1" %>
<%@ page errorPage="errorpage.jsp" %>
<%@ taglib uri="/estaglib.jar" prefix="lotuskms" %>
```

Estaglib.jar ships with Lotus Extended Search 3.5 and 3.7. You can find the jar file in the following paths:

- WebSphere 3.5: <WAS_path>\hosts\default_host\lotuskms\web
- WebSphere 4.0: <WAS_path>\installedApps\ExtendedSearch.ear\extendedSearch.war

The contentType directive tells the application server what type of content the page contains and what locale it should use to display the content. If this is not set, then it will use the system default locale and content type. The system default locale will be the locale of the server where the application server is running.

The next thing we see is a call to instantiate EsHTMLDirectoryBean. The ID (nickname) of “dir” allows us to use the bean methods throughout the page by referring to them as “dir.<method name>.”


```
<jsp:useBean id="dir" class="com.ibm.bts.es.srv.beans.jsp.EsHTMLDirectoryBean"
scope="session"></jsp:useBean>
<jsp:setProperty name="dir" property="*"/>
```

The purpose of the EsHTMLDirectoryBean is to remove the need to “hard code” the JSP directory paths and application paths. The bean provides methods for retrieving the client locale, Web root, JSP directory, image directory, and so on. These values are taken from the Extended Search admin.properties file. The admin.properties file is a configuration file that specifies many of the administrative paths and variables for Extended Search. Describing the values in this file is beyond the scope of this article, but if you want to find out more about them, see the **Lotus Extended Search documentation**.

This line shows an example of how the ExHTMLDirectoryBean is used:

```
<FORM NAME="_QUERY" ACTION="<%= dir.getJSPURI() %>tagSimpleSampleResults.jsp" METHOD="POST">
```

The dir.getJSPURI() method returns the relative path to the Extended Search JSP templates, for example, /lotuskms/templates/enUS/JSP/. This sets the forms action to the current path of the JSP file. You may have noticed that we are no longer posting to JKMSearchController as we did in the JKM example in the previous article. The form action is set to the JSP page (tagSimpleSampleResults.jsp) instead of JKMSearchController. This is one of the major differences between the JSP and JKM interfaces. After the Extended Search JSP template is loaded, it no longer relies on the JKMSearchController. Therefore, none of the required JKMSearchController parameters need to be in the request.

The Extended Search JSP tags
The next block of code uses the Extended Search JSP tags. The JSP tags, like the JKM tags, have both a begin tag and an end tag. The JSP begin tag specifies the name of the tag (Java Bean) to use, while the end tag specifies the name of the tag to close.

Begin Tag	<lotuskms:[tag name] [required parameter]>
End Tag	</lotuskms:[tag name]>

Let’s look at how we use these JSP tags to retrieve categories and sources.

```
<% String applID = (String)request.getParameter("AppID"); %>
<lotuskms:categories appID="<%= applID %>">
<INPUT TYPE="CHECKBOX" NAME="DESCategoryNames" VALUE="<%= esCategoryName %>">
 <%= esCategoryName %><BR>
<SELECT NAME="DESDataSourceNames" SIZE="2" MULTIPLE>
<lotuskms:sources appID="<%= applID %>">
<OPTION VALUE="<%= esSourceName %>"><%= esSourceName %></OPTION>
</lotuskms:sources>
</SELECT>
<BR><BR>
</lotuskms:categories>
```

The JSP categories and sources tags function similarly to their JKM counterparts. The categories tag loops over the categories within an application. The sources tag is placed within the categories tag to retrieve the sources associated with each category. The replacement variable esCategoryName is used to get the name of a category within the application. Likewise, a replacement variable of esSourceName is used to get the name of each source within the category. Again, the names of the form variables are very important. Inside the categories tag, we have the checkbox for the category. Its name must be DESCategoryNames because the Java Bean used to retrieve the categories looks for request parameters with the name “DESCategoryNames” to determine which categories to use for the search.

```
<INPUT TYPE="CHECKBOX" NAME="DESCategoryNames" VALUE="<%= esCategoryName %>">
```

Likewise, a name of DESDataSourceNames must be given to sources because the Java Bean references this name as well.

```
<SELECT NAME="DESDataSourceNames" SIZE="2" MULTIPLE>
```


Creating the search results page

Now that we have displayed the categories and sources, you can submit the query. To run the search, you must enter the query string in the text box provided, select at least one category and/or source, and click the Submit Query button. When you click the button, the form submits the request to the location specified in the form action. As we mentioned before, this request is no longer submitted to JKMSearchController (as it was for the JKM interface), but rather to the JSP listed in the action. We chose to submit the request to the tagSimpleSampleResults.jsp page. Let's take a look at the JSP code used to create the tagSimpleSampleResults.jsp page.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<%@ page language="Java" %>
<%@ page contentType="text/html; charset=iso-8859-1" %>
<%@ page errorPage="errorpage.jsp" %>
<%@ taglib uri="/estaglib.jar" prefix="lotuskms" %>

<HTML>
<HEAD>
     <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
     <TITLE>Extended Search: Simple Sample JSP 1.1 Interface Results</TITLE>
</HEAD>

<jsp:useBean id="criteria" class="com.ibm.bts.es.srv.beans.jsp.EsSearchCriteriaBean"
scope="session"></jsp:useBean>
<jsp:setProperty name="criteria" property="*" />

<BODY>
     <%
          // Execute the search and get the results vector.
          Java.util.Vector hitListResults = null;
          try {
     %>
     <lotuskms:runSearch searchCriteria="<%= criteria %>">
     <%     hitListResults = esSearchResults;        %>
     </lotuskms:runSearch>

     <FONT COLOR="BLUE" SIZE="5"><B>Simple JSP Demonstration Results</B></FONT><BR><BR>
     <TABLE WIDTH="640" CELLPADDING="2" CELLSPACING="0" BORDER="1">
          <TR>
          <TD WIDTH="50" STYLE="color: white; background-color: blue; font-weight: bold; text-align:
          center">Rank</TD>
          <TD WIDTH="150" STYLE="color: white; background-color: blue; font-weight: bold; text-align:
          center">Source</TD>
          <TD STYLE="color: white; background-color: blue; font-weight: bold; text-align: center">URL</TD>
          </TR>
          <lotuskms:hitList searchResults="<%= hitListResults %>">
          <TR>
          <TD>
          <lotuskms:hitFields name="DocRank"><%= esHitFieldValue%>
          </lotuskms:hitFields>
          </TD>
          <TD><%= esHitSourceName %></TD>
          <TD><A HREF="<%= esHitURL %>" TARGET="NEW"><%= esHitURL %></A></TD>
          </TR>
          </lotuskms:hitList>
     </TABLE>
     <%
     } catch ( Throwable e ) {
          out.println( e.getMessage() + "<BR>" );
          }
     %>

</BODY>
```



```
</HTML>
```

Again, we see the JSP directives at the top of the page as well as a call to instantiate the EsSearchCriteriaBean.

```
<jsp:useBean id="criteria" class="com.ibm.bts.es.srv.beans.jsp.EsSearchCriteriaBean"
scope="session"></jsp:useBean>
<jsp:setProperty name="criteria" property="*" />
```

This bean, as its name suggests, is used to set up all of the search criteria, such as the query string, number of hits returned, sources, and so on. The set property call is used to set any values found within the request that match the set methods within the bean. In this case, the user selects the query, categories, and sources in the request. This call automatically reads all of these parameters for us!

The next block of code runs the search.

```
<%
      // Execute the search and get the results vector.
      Java.util.Vector hitListResults = null;
      try {
%>
<lotuskms:runSearch searchCriteria="<%= criteria %>">
<%      hitListResults = esSearchResults;         %>
</lotuskms:runSearch>
```

The runSearch tag
The JSP tag runSearch executes the search. A Java vector is created to store the results received from the search. The replacement variable esSearchResults returns a vector of search results from the bean. The bean may throw an exception, so it is important that you surround the runSearch tag with code to catch the exception. An exception can occur if the user does not enter a query string, does not select at least one category or source, and so on. The bean throws an exception containing a message in the appropriate code page (based on the client locale) that indicates what problem occurred.

The final block of code processes the results of the search.

```
<lotuskms:hitList searchResults="<%= hitListResults %>">
<TR>
<TD>
<lotuskms:hitFields name="DocRank"><%= esHitFieldValue%>%</lotuskms:hitFields>
</TD>
<TD><%= esHitSourceName %></TD>
<TD><A HREF="<%= esHitURL %>" TARGET="NEW"><%= esHitURL %></A></TD>
</TR>
</lotuskms:hitList>
```

The hitList and hitFields tags
The hitList tag is used to loop over the search results. You must supply a vector of search results that are set within the tag (searchResults="<%= hitListResults %>"). The variable hitListResults refers to the vector of results that we received from the runSearch tag. Within the hitList tag, the name of the source that returned the hit replaces the replacement variable esHitSourceName, while the document ID of the hit replaces esHitURL.

The hitFields tag retrieves the field for each hit. This tag must be used within the hitList tag. Here, we look for the field named DocRank. The value of this field replaces the replacement variable esHitFieldValue.

The end result is a page that displays a table of search results. The results contain the document ID as well as the document rank and source name.

Summary

In this article, we discussed the JSP interface provided by Extended Search. Lotus Extended Search provides a JSP interface to expose the power of the Java programming language. The samples provided with this article demonstrate how both the JSP and HTML interfaces can be used to retrieve information from Extended Search. There are also samples provided with the Extended Search package, which show more advanced concepts. We encourage you to look at these samples to learn about the additional functionality Extended Search provides.

ABOUT THE AUTHOR

Brett Morris is a developer with the Extended Search team. He has a Bachelor's degree in Computer Science from George Mason University, and he has been a member of the Extended Search Team since January 2000. Brett has been heavily involved with the client template development for Extended Search since the fall of 2000, and he is the author of several of the JSP templates included with the Extended Search package. Brett is also a fourth degree Black belt in American Karate and enjoys competing and training in the martial arts.

