Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today

Level: Advanced
Works with: Extended Search
Updated: 01-Aug-2002

by
Brett
Morris

So you want to customize your Extended Search setup to provide for the specific needs of your environment? One way you can accomplish this is by creating custom Web-based templates that use Extended Search Java Beans. Extended Search provides several examples of Web templates that interact with the Extended Search Java Beans to create an interface for searching and retrieving results.

In Part 1 of this two part article, we describe how to create a Web template that interacts with Extended Search Java servlets and Java Beans as well as an explanation of several simple templates. You can download these templates from the **Sandbox**. These templates demonstrate how to retrieve source and category information as well as how to organize and how to display field information in your search results. The following technical explanation is based on Extended Search Release 3.5 and 3.7. Part 1 of the article demonstrates how to use HTML and the custom Extended Search tags to customize the Extended Search Web templates. In Part 2, we'll use JavaServer Page (JSP) tags to customize the Web templates.

This article assumes advanced knowledge of HTML, including forms, and experience with Extended Search. For an architectural overview of Extended Search, please see the *LDD Today* article "**Introduction to Lotus Extended Search**."

Extended Search Web development overview

The Extended Search Web application provides several servlets used to expose various functions of the system. The ESAdmin servlet accesses the Extended Search Web administration client. The Web-based search templates use the JKMSearchController servlet. The JKMSearchController servlet loads the distinct search template interfaces in Extended Search, including:

- JKM (for Java Knowledge Management)
- JSP 1.0 (for JavaServer Pages version 1.0)
- JSP Tag 1.1 (for a JSP tag library approach supported by JSP processors 1.1+)

The JKM interface, also known as an HTML interface, uses JavaScript to support older application servers, like Domino R5. The JSP interfaces are the newer, and preferred, method of interfacing with Extended Search. These Extended Search interfaces support application servers with the JSP 1.0 and/or 1.1+ processors, such as the WebSphere Application Server. Both interfaces use a set of Java Beans that gather and send information to and from Extended Search.

The JKMSearchControllerservlet

Regardless of the interface being used, all Extended Search Web client requests must be sent to the JKMSearchController servlet. This servlet is responsible for handling authentication and for setting up session information, including the required and optional parameters in the request. The JKMSearchController servlet

requires the following parameters:

Parameters	Description
desTemplateFile or desJSPFile (depending on interface)	Specifies the Web template to load. If the request contains the desTemplateFile parameter, then the template uses the JKM interface. If the request contains the desJSPFile parameter, then the template uses a JSP interface.
desClientLocale	Specifies the locale that the Extended Search server uses when running the search and returning results.
AppID	Specifies the Extended Search application to use for this request. The application ID is specified in the Extended Search configuration database.

To pass the parameters, including the Web template, to the servlet, the JKM interface and the JSP tags use URLs. The following are examples of how the JKM interface and JSP tags access JKMSearchController via HTTP.

JKM:
http://localhost/lotuskms/JKMSearchController?desTemplateFile=AllOptions.txt&desClientLocale=enUS&AppID=Demo

JSP:
http://localhost/lotuskms/JKMSearchController?desJSPFile=AllOptions.jsp&desClientLocale=enUS&AppID=Demo

The JKM interface and JKM tag
To create templates using the JKM interface, you must first understand the JKM tags and how the JKMSearchController servlet interprets them. When you submit a request to the JKMSearchController servlet to load a JKM template, the servlet performs a series of replacements. It parses the file, locates each JKM tag, and invokes the appropriate Java Bean to replace the tag with data. The end result is an HTML file that you can view in your Web browser.

So, what is a JKM tag? A JKM tag denotes a location within a text (pseudo-HTML) file that the JKMSearchController servlet interprets when it parses. These tags tell the JKMSSearchController servlet which data (Java) bean to use to retrieve the specified data. A beginning tag with associated attributes, an ending tag, and the replacement variable define each JKM replacement tag:

- <!--<JKM>--> denotes the beginning of a JKM tag.
- $ITEM$ denotes the Extended Search replacement variable.
- <!--</JKM>--> denotes the end of a JKM tag.

There are several attributes for the JKM tag:

Attribute	Description
Type	Specifies the Java Bean that the JKMSearchController servlet should use for this tag.
Occurrence	Specifies an integer value, string, or predefined variable. The value of this tag depends on the Java Bean being used. Its value typically denotes the number of times to execute a loop.
Parent	Explicitly identifies the parent of the current type attribute. It is useful when there is a parent-child relationship within the data (for example, Categories and Sources).

The JKM tag passes the Occurrence and Parent attributes as parameters to the Java Bean being loaded. It is up to the Java Bean to interpret the attribute and to act accordingly. For this reason, these tags can be used differently depending on the needs of the Java Bean and the data requested. Here is an example of a JKM tag using the Type and Occurrence attributes:

```
<!--<JKM TYPE=HttpRequest OCCURRENCE=MyVariable>-->
$ITEM$
<!--</JKM>-->
```


In the example, the JKM tag tells the JKMSearchController servlet to load the JDESHttpRequestBean Java Bean and to look for a variable named MyVariable. The JDESHttpRequestBean Java Bean contains the contents of the HTTP request and looks for a parameter named MyVariable in the request. If it finds this parameter, the Java Bean replaces $ITEM$ with the value of the parameter. If the parameter isn't found in the request, the JDESHttpRequestBean Java Bean replaces $ITEM$ with an empty string. Notice that we passed a type of HttpRequest instead of passing the entire name of the JDESHttpRequestBean Java Bean. This is done because the JKMSearchController servlet supplies JDES at the beginning and Bean at the end of the name you enter for Type. Therefore, a full Java Bean name isn't necessary.

JKM Java Beans
The following JKM Java Beans are used to create the sample template provided with this article, which you can download from the **Sandbox**:

- JDESActionURLBean
- JDESCategoriesBean
- JDESSourcesBean
- JDESHitlistBean
- JDESHitFieldBean

For a detailed list and description of all the JKM Java Beans, refer to your Extended Search programming documentation.

Simple search Web template

Now let's look at an example that uses the JKM interface to interact with Extended Search. To access this example, download the sample templates from the **Sandbox**, and then copy the template files to your Extended Search JKM templates directory.

- In Extended Search 3.5, the default JKM templates directory for WebSphere is /lotuskms/templates/enUS, and for Domino, the default directory is /servlet/templates/enUS.
- In Extended Search 3.7, the default JKM templates directory for WebSphere is /lotuskms/templates/enUS/JKM, and for Domino, the default directory is /servlet/templates/enUS/JKM.

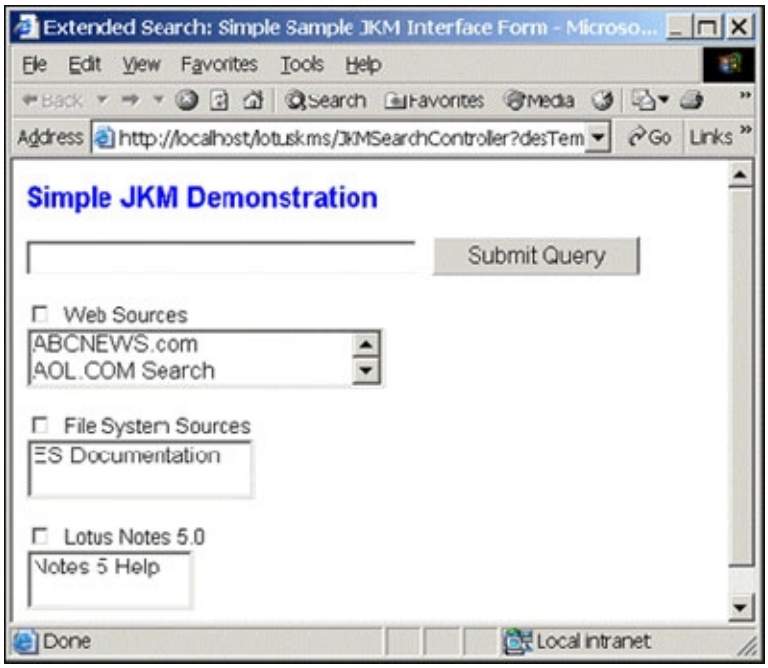
You can invoke this example by using the following URLs:

Domino:
http://<your server name>:<port>/servlet/JKMSearchController?desTemplateFile=simpleSample.txt&desClientLocale=enUS&AppID=<your application>

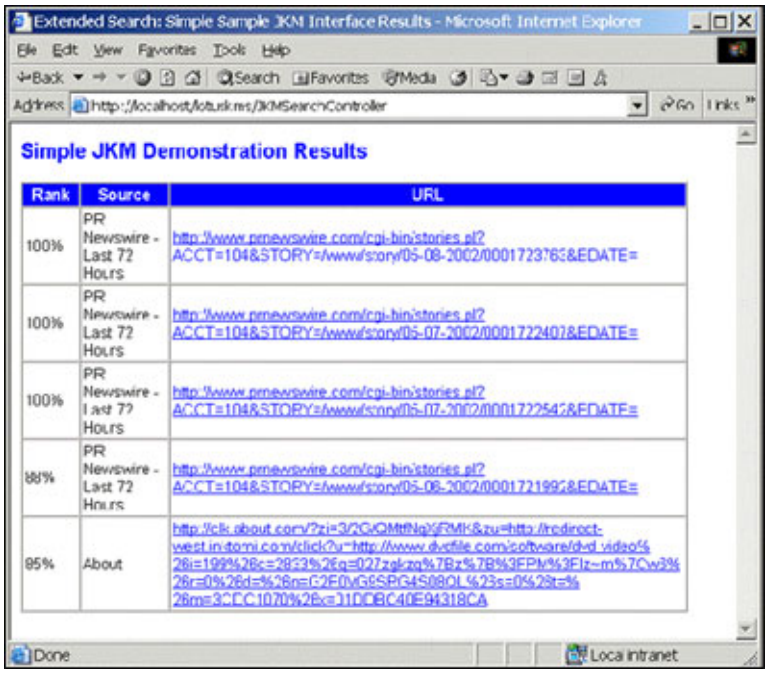
WebSphere:
http://<your server name>:<port>/lotuskms/JKMSearchController?desTemplateFile=simpleSample.txt&desClientLocale=enUS&AppID=<your application>

Replace <your server>:<port> with the hostname and port of your Web server. Likewise, <your application> should be replaced with the name of the application you want to load. In our example, we use Demo as the application name.

After you load the template, you should see the initial screen, which displays a simple search form:



In the example, there is a text box for entering a query string along with checkboxes for each category. There is also a list box underneath each category, containing the sources for the corresponding category. The template allows the user to type a query string, to select one or more categories or data sources to search, and to click the Submit Query button to run the search. When you click this button, JKMSearchController runs the search and returns a new page that contains the search results. For example:



For each document, the list contains the name of the source and the URL of the search result. The results are sorted by rank, which indicates the likely relevance of the document to the query.

Now let's look at the HTML and script used to create this template. We'll start by breaking down the HTML file

simpleSample.txt used to submit the query. First, here it is in its entirety.

```
<HTML>
<HEAD>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
      <TITLE>Extended Search: Simple Sample JKM Interface Form</TITLE>
</HEAD>
<BODY>
<!--<JKM TYPE=ActionURL OCCURRENCE=SearchServlet>-->
<FORM NAME="_QUERY" ACTION="$ITEM$" METHOD="POST">
<!--</JKM>-->

      <INPUT TYPE=HIDDEN NAME="desTemplateFile" VALUE="simpleSampleResults.txt">
      <INPUT TYPE=HIDDEN NAME="AppID" VALUE="Demo">
      <INPUT TYPE=HIDDEN NAME="desClientLocale" VALUE="enUS">

<!-- Title -->
<FONT COLOR="BLUE" SIZE="5"><B>Simple JKM Demonstration</B></FONT><BR><BR>

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

      <!--<JKM TYPE=Categories OCCURRENCE=ALL>-->
      <INPUT TYPE="CHECKBOX" NAME="DESCategory-$ITEM$"    VALUE="$ITEM$">
       $ITEM$<BR>
      <SELECT NAME="DESSources" SIZE="2" MULTIPLE>
      <!--<JKM TYPE=Sources OCCURRENCE=ALL>-->
            <OPTION NAME="DESSource-$ITEM$" VALUE="$ITEM$">$ITEM$</OPTION>
      <!--</JKM>-->
      </SELECT>
      <BR><BR>
      <!--</JKM>-->

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

Remember that a JKM page is a Web page, so it's simply a combination of HTML (used to format text, to provide information from Extended Search, to return results, and so on) and JKM tags used by the JKMSearchController servlet. In our example, we added a few JKM tags to an HTML file to gather information from Extended Search.

The first JKM tag in the page above is used to obtain the action for the form.

```
<!--<JKM TYPE=ActionURL OCCURRENCE=SearchServlet>-->
<FORM NAME="_QUERY" ACTION="$ITEM$" METHOD="POST">
<!--</JKM>-->
```

This JKM tag tells the JKMSearchController servlet to use the JDESActionURLBean Java Bean to obtain the search servlet path and to return the path used for the post, for example, /lotuskms/JKMSearchController, to the JKMSearchController servlet.

The next important thing to note about this page is the hidden input parameters. If you recall, we mentioned that the JKMSearchController servlet requires several parameters in any request that it processes. Therefore, we add these parameters as hidden inputs tags in our HTML file.

```
<INPUT TYPE=HIDDEN NAME="desTemplateFile" VALUE="simpleSampleResults.txt">
<INPUT TYPE=HIDDEN NAME="AppID" VALUE="Demo">
<INPUT TYPE=HIDDEN NAME="desClientLocale" VALUE="enUS">
```

Using hidden input tags ensures that these parameters are in the POST request to the servlet. If these parameters are not present in the request, the JKMSearchController servlet rejects the request and displays an error message in the requesting browser.

The last part of this file includes a set of JKM tags used to gather the category and source information.

```
<!--<JKM TYPE=Categories OCCURRENCE=ALL>-->
<INPUT TYPE="CHECKBOX" NAME="DESCategory-$ITEM$" VALUE="$ITEM$">
 $ITEM$<BR>
<SELECT NAME="DESSources" SIZE="2" MULTIPLE>
<!--<JKM TYPE=Sources OCCURRENCE=ALL>-->
<OPTION VALUE="$ITEM$">$ITEM$</OPTION>
<!--</JKM>-->
</SELECT>
<BR><BR>
<!--</JKM>-->
```

The JKM type Categories uses the JDESCategoriesBean Java Bean to retrieve the categories for the application.

```
<!--<JKM TYPE=Categories OCCURRENCE=ALL>-->
```

In this case, we specified an occurrence of ALL because we want the Java Bean to return every category in the application. This creates a loop that continues until every category is returned. Therefore, if we have three categories, everything inside the JKM tag for categories is executed three times.

Inside the categories loop, we created a checkbox for each category.

```
<INPUT TYPE="CHECKBOX" NAME="DESCategories" VALUE="$ITEM$">
```

The checkboxes are HTML form variables that must have the name DESCategories because that name is used to determine which category or categories are used in the search. If the user selects a checkbox, the request parameter of DESCategories is added to the request. This parameter's value contains the name or names of the categories used in the search. The variable $ITEM$ is replaced by the JKMSearchController servlet with the name of the category. For example, if we have a category named Web Sources, then the checkbox name is DESCategory-Web Sources. If the user selects this checkbox, then the name of the form variable is included in the request to the JKMSearchController servlet. This name is used to determine which category or categories are used for the search.

The next item inside the categories loop is the list box of sources.

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

This form variable is also very important. A request parameter of DESSources is used to determine which source or sources are used in the search. If the user selects an item within this list box, a request parameter of DESSources is present in the request. This parameter contains the name or names of the sources used in the search.

Inside the list box, the JKM tag retrieves the sources.

```
<!--<JKM TYPE=Sources OCCURRENCE=ALL>-->
```

This tag uses the JDESSourcesBean Java Bean to retrieve each source for the category. This bean acts similarly to the categories bean. It loops until every source for this category is returned. By placing this tag inside the categories bean, we create a nested loop, so the sources loop is executed for each category. An option inside the list box (DESSources) is created for each source. The value of this option is the name of the source.

```
<OPTION VALUE="$ITEM$">$ITEM$</OPTION>
```

The end result of this setup is a checkbox for each category with an associated list box for each source under that category.

Retrieving search results

Now we are ready to run the search! After you enter the query string in the text box and select a category or source (or both), click the Submit Query button to execute the search. This button submits the form variables to the specified location (that is, the form action). If you recall, we set the form action to submit requests to the JKMSearchController servlet, which processes our request and loads the specified desTemplateFile, which is a

hidden input parameter.

```
<INPUT TYPE=HIDDEN NAME="desTemplateFile" VALUE="simpleSampleResults.txt">
```

Now the JKMSearchController servlet loads the simpleSampleResults.txt file. Let's take a look at the HTML and tags used in the simpleSampleResults.txt file. Here it is in its entirety:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
<TITLE>Extended Search: Simple Sample JKM Interface Results</TITLE>
</HEAD>
<BODY>
<FONT COLOR="BLUE" SIZE="5"><B>Simple JKM 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>
        <!--<JKM TYPE=HitList OCCURRENCE=ALL>-->
        <TR>
        <TD>
        <!--<JKM TYPE=HitField OCCURRENCE=DocRank>-->
        $ITEM$%
        <!--</JKM>-->
        </TD>
        <TD>
        <!--<JKM TYPE=HitField OCCURRENCE=$SOURCE$>-->
        $ITEM$
        <!--</JKM>-->
        </TD>
        <TD>
        <A HREF="$ITEM$" TARGET=NEW>$ITEM$</A>
        </TD>
        </TR>
        <!--</JKM>-->
    </TABLE>
</BODY>
</HTML>
```

On the results page, the first JKM tag uses the type "HitList."

```
<!--<JKM TYPE=HitList OCCURRENCE=ALL>-->
```

This loads the HitList Java Bean used to run the search and return the document ID for each hit. The HitList Java Bean processes the request variables and tells Extended Search the search options, categories, sources, and query string to use for this search. An occurrence of ALL is used to loop over each item in the hitlist. In this case, we loop over each hit and replace $ITEM$ with the document ID for the hit.

```
<A HREF="$ITEM$" TARGET=NEW>$ITEM$</A>
```

Next, we have two JKM tags using the HitField Java Bean to return hit field information. The first tag uses the HitField Java Bean to obtain the document rank for this hit.

```
<!--<JKM TYPE=HitField OCCURRENCE=DocRank>-->
```

The second tag retrieves the source name for the hit (that is, which source returned this hit).

```
<!--<JKM TYPE=HitField OCCURRENCE=$SOURCE$>-->
```


You must use the HitField Java Bean tag within the body of the HitList Java Bean tag because the HitList bean sets necessary information for the HitField bean within the session. An occurrence of ALL can be used to loop over every field associated with each hit. In our case, however, we only want to display the rank and source names. (Refer to the Extended Search documentation to see the occurrences that can be used for this bean). The end result of this is a table of search results. The results contain the document ID (URL) of each hit as well as the document rank and source name.

How $ITEM$ is used

One of the most difficult things to understand about JKM tags is how $ITEM$ is replaced. In other words, if we use $ITEM$ multiple times throughout a template page, how does the JKMSearchController servlet know the proper replacement string? To illustrate this, let's take a look at an example using the HitList Java Bean tag and the HitField Java Bean tag.

```
<!--<JKM TYPE=HitList OCCURRENCE=ALL>-->
Document ID: $ ITEM $
<!--<JKM TYPE=HitField OCCURRENCE=ALL>-->
&amp;nbsp;&amp;nbsp;Field: $ ITEM $<br>
<!--</JKM>-->
&amp;nbsp;&amp;nbsp;URL: <A HREF="$ITEM$" TARGET=NEW>$ITEM$</A><br><br>
<!--</JKM>-->
```

This example displays the document ID for the hit followed by all of the fields and a URL to access the document. The $ITEM$ replacement variables are on the second, fourth, and sixth lines. The JKMSearchController servlet replaces the $ITEM$ variable on the second and sixth lines with the document ID returned from the HitList Java Bean. However, the $ITEM$ variable on the fourth line is replaced by the field value returned from the HitField Java Bean. The reason for this is that $ITEM$ variables are replaced by the string returned from the last JKM tag opened. The JKMSearchController servlet replaces $ITEM$ with the string from the beginning JKM tag until it finds a closing JKM tag. When it finds a closing JKM tag, it uses the replacement string from the last JKM tag opened. We see that $ITEM$ on the sixth line appears after the closing tag on the fifth line for the HitField Java Bean tag, so it's replaced by the string from the HitList Java Bean tag on the first line. However, the $ITEM$ variable on the fourth line is inside the HitField JKM tag on the third line, so it is replaced by the string returned from the HitField Java Bean.

Conclusion

In Part 1 of this two part article, we briefly discussed how the JKMSearchController loads an Extended Search Web template and the role that the JKMSearchController servlet plays in the Extended Search Web client. We also described how you can use the JKM Interface to create a Web template that interacts with Extended Search. We discussed how to use the JKM tags and how the JKMSearchController interprets the tags and loads the appropriate Java Beans. We also looked at a sample template that uses the JKM interface to setup search criteria, to submit a search, and to display search results. In Part 2 of the article, we will examine how this functionality is exposed through the Extended Search JSP interface. We will look at the JSP tag libraries that Extended Search provides and see how these are used to create a sample template similar to the JKM sample in this article.

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.

