JavaBean/Backing Bean

Search NSF with Links on form

Xpages

System Requirements:
Download Domino Designer 8.5.2 Environment (DDE)
http://www.ibm.com/developerworks/downloads/ls/dominodesigner/

Introduction:
Build a connection to a Lotus Notes Domino Database located on C Drive using a JavaBean, Backing Bean. Once connection is established, grab view data and load to Xpages form. Search for unique documents using links on the page.

Disclaimer:
Information contained in the following is presented as is. This tutorial assumes you have basic programming knowledge. All tutorials are based on an Eclipse/Eclipse-based software. Should you need to familiarize yourself with a certain Eclipse environment, prior to continuing this tutorial, please stop now and see our Tutorials page...

Build Results into Xpages
At this point we assume Domino Designer 8.5.2 is downloaded/installed and a new JavaBean, Backing Bean, and the Xpage are already created. Copy and paste below code to your environment for each code samples.

Keyword.java

[CODE]

```
/**
 * Copyright 2012 Dököll Solutions, Inc.
   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
 *
 * @AppName: Docu.nsf
```

```
 * Program: Keyword.java
 * Created: 2011.02.10.9.21.AM
 * JavaBean Class to load view data to Xpages
 *
 * Modified: 2011.11.27.7.24.PM
 * Adding actual data from a view
 *
 * Modified to add additional field 'BrowserURL'
 * 2012.03.01.12.30.PM
 */
package com.dokoll.solutions.inc.JavaBeans;



/**
 * @author Dököll Solutions, Inc.
 * @version: 2011.02.10.9.17.AM
 */
public class Keyword {

	private String Category;
	private String Topic;
	private String BrowserURL;

	// getter/setter methods

	/**
	 * @return the topicName
	 */
	public String getTopic() {
		return Topic;
	}

	/**
	 * @param topicName
	 *            the topicName to set
	 */
	public void setTopic(String topic) {
		this.Topic = topic;
	}

	/**
	 * @param categoryName
	 *            the categoryName to set
	 */
	public void setCategory(String category) {
		this.Category = category;
	}

	/**
	 * @return the categoryName
	 */
	public String getCategory() {
		return Category;
	}

	/**
	 * @return the browserURL
	 */
```

```
	public String getBrowserURL() {
		return BrowserURL;
	}

	/**
	 * @param browserURL
	 *            the browserURL to set
	 */
	public void setBrowserURL(String browserURL) {
		BrowserURL = browserURL;
	}

	//call this method from the backing bean
	public Keyword(String Category, String Topic, String BrowserURL) {

		this.Category = Category;
		this.Topic = Topic;
		this.BrowserURL = BrowserURL;

	}

}
```

Backing Bean it!

Your JavaBean has been created, and we assume it compiles okay. You are now ready to plug in a Backing Bean to load data to Xpages. Copy and paste below to your environment, areas of interest have been highlighted for your convenience.

RetrieveNewNoticeBackingBean.java

```
/**
 * Copyright 2012 Dököll Solutions, Inc.
   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.

 * Program: RetrieveNewNoticeBackingBean.java
 * Created: 2011.01.28.3.31.PM
 * New Retrieval Bean for Xpages

 * Modified: 2011.01.31.2.22.PM
 * Modified to add Debug variables

 * Modified to add URL reference, additional field 'BrowserURL', UniversalID call
 * 2012.03.01.12.37.PM
 * Modified to Add Server calls to ensure the right server is loaded
 * this also helps in the case the app is viewed in the Client side of Notes
```

```
 * TO DO: Add: ClientType to ensure where users are coming from
 * 2012.03.02.3.49.PM
 *
 */
package com.osc.its.bulletin.board.JavaBeans;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;

import lotus.domino.View;
import lotus.domino.local.Database;
import lotus.domino.local.Document;
import com.dokoll.solutions.inc.JavaBeans.Keyword;

/**
 * @author Dököll Solutions, Inc.
 * @version 2011.01.28.3.31.PM
 *
 */
public class RetrieveNewNoticeBackingBean {

	// declare variables
	final String DB_LOAD = "Docu.nsf/";
	// Added Server calls
	// 2012.03.02.3.44.PM
	// declare variable for localhost Server
	final String LOCAL_LOAD = "http://localhost/";
	final String ViewName = "By Category";


		// Get Keyword collection
	public Keyword[] getKeywords() {

		// Declare Keyword Array
		Keyword[] keywords = null;

		try {
			// 2011.02.10.9.37.AM
			// get the current database being used
			Database database = (Database) FacesContext.getCurrentInstance()
				.getApplication().getVariableResolver().resolveVariable(
					FacesContext.getCurrentInstance(), "database");


			// Call HTTP Servlet Class to render a Server Name
			// 2012.03.02.4.00.PM
	HttpServletRequest httpServletRequest = (HttpServletRequest) FacesContext
			.getCurrentInstance().getExternalContext().getRequest();
			// declare String to hold the ServerName
			String DB_SERVER = httpServletRequest.getServerName();

			// Get the full URL and join it with the current document to grab
			// from Notes back-end
			// 2012.03.02.8.47.PM
	HttpServletRequest req = (HttpServletRequest) FacesContext
			.getCurrentInstance().getExternalContext().getRequest();
			String baseURL = req.getRequestURL().toString().replace(
			req.getRequestURI().substring(1), req.getContextPath());
```

```
                // Above does not work in Notes Client, we get below error
                // Error 404: ProxyServlet:
                // //xsp/Docu.nsf/By%20Category/464A7E820B12313A85257849005B8BFE
                // 2012.03.02.9.05.PM
                // TO DO: Call the right server via facesContext and tack that
                // onto the URL... pull up folders as well (hasNext)
                // 2012.03.02.4.08.PM

                // Grab a collection of all Keyword documents
                View view = database.getView(ViewName);
                System.out.println("View Obtained..." + view);

                Document sDoc;
                Document ndoc;

                sDoc = (Document) view.getFirstDocument();
                System.out.println("Document Obtained..." + sDoc);

                // Begin initialization of the Keywords Array
                // load documents count
                keywords = new Keyword[view.getEntryCount()];

                // Run through Keyword document collection
                int docount = 0;
                while (sDoc != null) {
                Keyword keyword = new Keyword(null, null, null);
                // render document values to variables of
                // Keyword object
                keyword.setCategory(sDoc.getItemValueString("Categories"));
                keyword.setTopic(sDoc.getItemValueString("Subject"));
                // TO DO: Alternatively, add baseURL variable in the place of
                // DB_LOAD and LOCAL_LOAD.  This works best in a web Browser...
          keyword.setBrowserURL(LOCAL_LOAD+DB_LOAD+view+"/"+sDoc.getUniversalID()
                                                          +"?OpenDocument");
                // Send Keyword object into Keywords Array
                keywords[docount] = keyword;
                // increment counts
                docount += 1;
                // load next doc
                ndoc = (Document) view.getNextDocument(sDoc);
                sDoc = ndoc;
          }

                // return the Array
                return keywords;

          } catch (Exception e) {
                e.printStackTrace();
          }

          return null;

     }

}
```

Build Xpage file

Full code added below, jump ahead if necessary, but do read the steps, we will be using a link

component and throw an URL underneath to allow unique documents to be viewed. Areas of interest have been highlighted for your convenience.

Steps:

1. Go to DDE's Database Navigator to your upper left
2. Double-click XPages, then click New XPage button
3. Code is generated, drag a link component from Core Controls Window
4. Select the link, Go to Properties Tab below to modify/add items
5. Your code should look bellow sample, (image at your choosing)...

Code to be modified

```
<xp:link escape="true" text="Link" id="link1" title="Testing Links"
      value="http://dokollsolutionsinc.com">
      <xp:image id="image1" url="/act_scheduler.gif"></xp:image>
</xp:link>
```

1. Go to the Backing Bean, try to understand how the URL is created
2. Make a note of the variable 'browserURL' to see how it connects to the Xpages form

```
// declare variables
final String DB_LOAD = "Docu.nsf/";

final String LOCAL_LOAD = "http://localhost/";
...
...
view = database.getView(ViewName);
Document sDoc;
...
...
setBrowserURL(LOCAL_LOAD+DB_LOAD+view+"/"+sDoc.getUniversalID()
                                                    +"?OpenDocument");
```

TIP:

browserURL variable was declared in the Keyword JavaBean:

```
Keyword keyword = new Keyword(null, null, null);
```

Modify test Xpage, remove information entered earlier

1. Copy dataSource and field name from Topic column '***#{categoryTable.topic}***'
2. Replace Link in **text="*Link*"** with the dataSource
3. Copy ***#{javascript:categoryTable.browserURL}*** dataSource
4. Replace **value="http://dokollsolutionsinc.com"**

*Your code should look as below...

Code modified

```
<xp:link escape="true" text="#{categoryTable.topic}" id="link2"
      value="#{javascript:categoryTable.browserURL}"/>
```

See original code for added information.

xpbycategory.xsp

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">

      <xp:dataTable rows="15" id="dataTable1" var="categoryTable"
            value="#{javascript:RetrieveNewNoticeBackingBean.getKeywords()}"
            style="width:80.0%;background-color:rgb(220,237,237)">
            <xp:this.facets>
                  <xp:pager partialRefresh="true"
      layout="FirstImage PreviousImage SeparatorPage Group NextImage LastImage"
                        xp:key="header" id="pager1" for="dataTable1"
      style="color:rgb(0,64,64);font-weight:bold;background-position:top center" />

<xp:pager partialRefresh="true"
layout="FirstImage PreviousImage SeparatorPage Group NextImage LastImage"
            xp:key="footer" id="pager2" for="dataTable1" title="By Category"
      style="color:rgb(0,64,64);font-weight:bold;background-position:top center" />
            </xp:this.facets>
            <xp:column id="column3">
                  <xp:this.facets>
                        <xp:label id="label1" xp:key="header"
style="font-weight:bold;font-family:Arial Black;font-size:14pt;background-
color:rgb(220,237,237);color:rgb(0,64,0)" for="image1" value="File" />
                  </xp:this.facets>
      <xp:link escape="true" id="link1"
      style="text-align:center;background-position:center center;width:88.0px">
                        <xp:this.value>
                              <![CDATA[#{javascript:categoryTable.browserURL}]]>
                        </xp:this.value>
                        <xp:image id="image1" url="/search.gif" alt="View File"
                              style="align:center" />
                  </xp:link>
            </xp:column>
            <xp:column id="column1">
                  <xp:this.facets>
<xp:label value="Subject" id="topic1" xp:key="header" style="font-
weight:bold;font-family:Arial Black;font-size:14pt;background-
color:rgb(220,237,237);color:rgb(0,64,0)" />
                  </xp:this.facets>
<xp:text escape="true" id="computedField3"
value="#{categoryTable.topic}"
                        style="color:rgb(64,0,0)" />
            </xp:column>
            <xp:column id="column2">
                  <xp:this.facets>
      <xp:label value="Category" id="category1" xp:key="header" style="font-
weight:bold;font-family:Arial Black;font-size:14pt;background-
color:rgb(220,237,237);color:rgb(0,64,0)" />

                  </xp:this.facets>
                  <xp:text escape="true" id="computedField2"
value="#{categoryTable.category}"
                        style="color:rgb(64,0,0)" />
            </xp:column>
      </xp:dataTable>
</xp:view>
```

Conclusion:
You can now have users click on an image from the Xpage to load unique documents or have a link for them to access/search unique documents.

Added info: You will need to reference `RetrieveNewNoticeBackingBean.java` in your faces-config.xml file.

Questions, comments, please post a brief message on our Contact form on the main site.

Thank you for coming...