

NotesView2

release 1.7 documentation

Last update: 11/20/2007
Jeremy G Hodge
Documentation version 0.1

NotesView 2.1.7 Documentation

Table of Contents

1 – Introduction

- 1.1 - What's new in this release
- 1.2 - Conventions used in this manual
- 1.3 - State of NotesView2

2 – Installing NotesView2 into your application

- 2.1 - Copy the JavaScript libraries into your application
- 2.2 - Copy the LotusScript Agents libraries into your application
- 2.3 - Copy the appropriate CSS files to your application.
- 2.4 - Configuring a Lotus Notes Form (for the web) to use the NotesView2 component

NotesView 2.1.7 Documentation

1.0 Introduction



Color ▲	Type ▼	Model ▼	Price ▼
☐ Ash Red			\$915,090
	☐ Ford		\$86,930
		Echo	\$13,650
		Yaris	\$13,825
		Civic	\$13,920
		Civic	\$14,648
		Mustang	\$15,291
		Civic	\$15,596
	☐ General Motors		\$97,452
		General Motors - Buick	\$59,635

Fig 1 Sample of a Categorize NotesView2 implementation with column totals.

NotesView2 is an open-source AJAX web component written in javascript to display a Lotus Notes view, as designed, in a web browser. NotesView2 can display:

- Standard flat views (no categories)
- Categorized views
- Categorized views with multiple levels in a single column. For example, categorized column formula is
`Value1 + "\\\" Value2`
- Views with column totals
- Views with documents in a response hierarchy

Built-in features include:

- Built-In Action bar for javascript-defined actions. (does not use the actions from the view)
- Expand All / Collapse All actions
- View Refresh Action
- Jump-To view key action
- View Search (limited functionality with response hierarchy)
- Capability to add custom-defined actions
- Ability to save/re-load the view state when navigating away from and returning to a page on which NotesView2 is placed

1.1 What's new in this release

- Full developer documentation.
- Added support for Column Totals - Thanks to Jonathan Roberts for code submission
- CSS definition for
 - `columnTotal` - used for the call contents of a cell that contains a category's total
 - `isCategory` - used for the cell and contents of a category that is a category
 - `CategoryTotal` - used for the cell at the bottom of a view that contains the view-wide total in a column Total
 - `CategoryTotalEmptyCell` - used in the empty cells to the left and right of the `CategoryTotal` cell as defined above
- Added `Adjacent[HTML | Text | Element]` support for firefox - Thanks to Peter Kempf
- Robusted up `saveState` to included expanded/collapsed entries, `selectedPosition` and `start position` Added `resetState()` function to clear the view's current state
- Added `'extendLastColumn'` property - defaults to true to mimic 2.1.6 & previous releases, but if set to false, the last column does not extend the total width - helpful for right aligned last columns
- Included example in Categorized View illustrating how to add a custom action to get the selected documents and process them using an agent in the background using `XMLHttpRequest` and then refreshing the view

NotesView 2.1.7 Documentation

- Added stateTimeout parameter to limit the length of time the state of the view will be saved
- Updated examples to reference view alias instead of view title

1.2 Conventions used in this document

Code samples will be displayed using a mono-type font in an inset box, for example:

```
function loadXMLRequest(url, callback, callbackContext, type, postData){
    if(!type)type="GET";
    if(!postData)postData = "";
    if(callbackContext==null)callbackContext = window;
    var request = null;
    var ok = false;
    try{
        if(window['XMLHttpRequest'])
            var request = new XMLHttpRequest();
        ok = true;
    }else{
        var request = new ActiveXObject("Microsoft.XMLHTTP");
        ok = true;
    }
    }catch(e){
        request = null;
    }
    }

    if(request){
        request.onreadystatechange = function(){
            if(request.readyState==4){
                var xml = (request.status==200) ?
request.responseXML : null;
                callback.call(callbackContext,xml);
            }
        };
        request.open(type,url,true);
        request.send(postData);
    }else{
        callback.call(callbackContext,null);
    }
}
```

Points of interest or notes will be displayed as such:

NOTE

To define a different length of time before the state of the view is to expire, set the stateTimeout property of the NotesView2 object. For example, assuming oView is your NotesView2 object:

```
oView.stateTimeout = { years: 0, months: 0, days: 1, hours: 0, mins: 0, secs: 0};
```

NotesView 2.1.7 Documentation

1.3 - State of NotesView2

NotesView2 v1.6 is the current stable release, with v1.7 currently in a "Release Candidate" stage. After an amount of time in the wild, v1.7 will become the current stable release.

NotesView 2.1.7 Documentation

2.0 - Installing NotesView2 into your application.

The following section describes how to install and configure NotesView2 for use in a Lotus Notes web application.

2.1 Copy the JavaScript libraries into your application

To implement NotesView2 in your application, copy the jsNotesView2.x.x, jsUtility, and NotesView2Lang_en JavaScript libraries from the NotesView2 sample application into your application.

There are 3 versions of the jsNotesView2.x.x and jsUtility libraries to choose from:

Standard	The standard libraries have all the original source code laid out in a human-readable format with comments. (jsNotesView2/jsUtility)
Crunched	The crunched version has had all comments and white-space removed to decrease the script libraries file size to speed delivery. (jsNotesView2_crunched/jsUtility_crunched)
Compressed	The compressed versions are compressed even further using a compression algorithm and decompression scheme delivered to the browser in the script library. (jsNotesView2_compressed/jsUtility_compressed)

All versions of the scripts have been tested for compatibility and you should have the same experience with each version, so pick the version that matches your design specifications.

The jsNotesView2.x.x library contains the code for the NotesView2 component itself. The jsUtility script provides additional functions and extensions to base JavaScript objects that NotesView2 requires. The NotesView2Lang_en script is a script to localize the NotesView2 object to other Languages. The NotesView2Lang_en script is for the English language. To localize NotesView2, make a copy of this script, rename the '_en' to your appropriate language code, and substitute that script library for the NotesView2Lang_en below.

2.2 Copy the LotusScript Agents libraries into your application

Next copy the appropriate agents to your application. There is 1 required you must add to your application, and 2 optional agents that are required if you enable searching of the view. The required agent is the (sleep) agent found in the NotesView2 Sample application. The sleep agent simply provides the ability for the NotesView2 object's script to sleep for a specified number of seconds before continuing execution.

```
Sub Initialize
  Dim session As New NotesSession
  Dim context As NotesDocument

  Set context = session.DocumentContext

  Dim sleeptime
  sleeptime = Evaluate(("{@TextToNumber(@Middle(") + context.Query_String(0) +
  {"; "&sleep="; "&"}))

  Print "Slept for " + Trim$(Str$(sleeptime(0)))

  Sleep (sleeptime(0))
End Sub
```

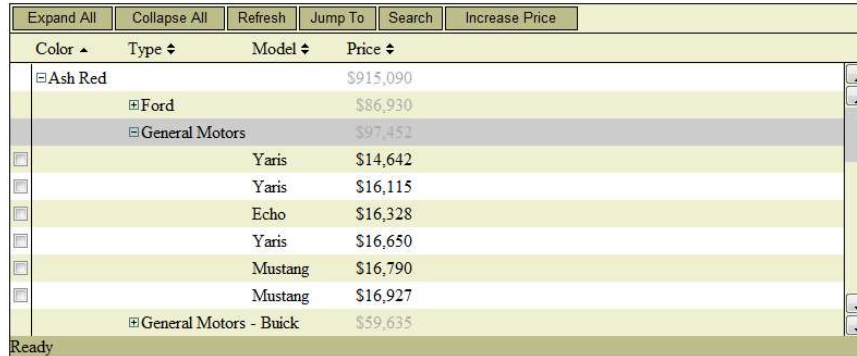
Code 1 The code for the (sleep) agent.

The optional agents are used if you enable searching from the NotesView2 component. If you enable this feature, you must copy these agents into the application. The two agents (agtIsDbIndexed and agtSearchView) report on the status of the full text index for an application (agtIsDbIndexed) and perform the actual search (agtSearchView) and return the results to the NotesView2 component.

NotesView 2.1.7 Documentation

2.3 Copy the appropriate CSS files to your application.

NotesView2 comes with two standard Cascading Style Sheet (CSS) documents that control the look & feel of the NotesView2 component. These CSS files can be customized to create your own look and feel.



Color	Type	Model	Price
[-] Ash Red			\$915,090
	[-] Ford		\$86,930
		[-] General Motors	\$97,452
		Yaris	\$14,642
		Yaris	\$16,115
		Echo	\$16,328
		Yaris	\$16,650
		Mustang	\$16,790
		Mustang	\$16,927
		[-] General Motors - Buick	\$59,635

Fig 2 Example of sample Plain.css



Color	Type	Model	Price
[-] Ash Red			\$915,090
	[-] Ford		\$86,930
		[-] General Motors	\$97,452
		Yaris	\$14,642
		Yaris	\$16,115
		Echo	\$16,328
		Yaris	\$16,650
		Mustang	\$16,790
		Mustang	\$16,927
		[-] General Motors - Buick	\$59,635

Fig 3 Example of sample Pretty.css

Copy your desired style sheet from [Pages] in the Sample application to your application.

2.4 Configuring a Lotus Notes Form (for the web) to use the NotesView2 component

To place a NotesView2 component on a form, first include a reference to the CSS document and JavaScript libraries in the [HTML Head] of the form. Include the following code (For this example, we will include a reference to the pretty.css cascading style sheet and the crunched version of the script libraries):

```
<LINK REL='stylesheet' TYPE='text/css' href='/' + @WebDbName + "/Pretty.css"> +
<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsNotesView2.1.7_crunched">
+
<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsUtility_crunched"> +
<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsNotesView2Lang_en">
```

Code 2 HTML Source for including the CSS and JavaScript

Next, we need to declare some JavaScript variables that we will use later when we initialize the NotesView2 object. Replace the <<YOUR_VIEWS_ALIAS_HERE>> with the alias (or view name if your view has no alias) of the view you wish to display.

```
<script>+
var DB_NAME = '/' + @ReplaceSubstring(@WebDbName;";";"\\") + ";"+
var VIEW_NAME = '<<YOUR_VIEWS_ALIAS_HERE>>'; +
</script>+
```

Code 3 HTML Source for declaring variables for use later.

NotesView 2.1.7 Documentation

NOTE

If you are creating a \$\$ViewTemplate form, you can substitute

```
"var VIEW_NAME = '<<YOUR_VIEWS_ALIAS_HERE>>';"
```

with

```
"var VIEW_NAME = '/' + @ReplaceSubstring(@Subset(@ViewTitle;-1);"':"\"";\"\\\":"\\\\\\") + ";"
```

Your complete [HTML Head] should look similar to the following:

```
"<LINK REL='stylesheet' TYPE='text/css' href='/' + @WebDbName + "/Pretty.css'" +  
"<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsNotesView2.1.7_crunched'"></SCRIPT>" +  
"<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsUtility_crunched'"></SCRIPT>" +  
"<SCRIPT TYPE='text/javascript' SRC='/' + @WebDbName + "/jsNotesView2Lang_en'"></SCRIPT>" +  
"<script>" +  
"var DB_NAME = '/' + @ReplaceSubstring(@WebDbName;"':"\"";"\\'") + ";" +  
"var VIEW_NAME = '<<YOUR_VIEWS_ALIAS_HERE>>';" +  
"</script>"
```

Code 4 Complete [HTML Head] Source for declaring variables for use later.

NOTE

You can also use the "Insert Resource" command in domino designer in the [JS Header] of the form to include the jsNotesView2.1.7_crunched, jsUtility_crunched, and jsNotesView2Lang_en from the script library. If you do, omit lines 2-4 from the example above.

Next we have to add a 'container' (a bit of pass-thru HTML code) on the form where we want the view to appear. In the location where you want the NotesView2 component to appear, include the following code, and mark it as Pass-Thru HTML.

```
<div id="viewHolder"></div>
```

Code 5 Pass-Thru HTML indicating where the NotesView2 component should appear

This creates an empty place holder on the form where the NotesView2 object will render itself later when we tell it to.

NOTE

You can control the width of the NotesView2 object by setting the CSS width style of the viewHolder div. For example, to restrict the width to 400 pixels, change the code to:

```
<div id="viewHolder" style="width: 400px;"></div>
```

The height of the component is controlled by the linesToDisplay property which we'll discuss a little later.

NotesView 2.1.7 Documentation

Now we have to create the actual NotesView2 object, configure it, and tell it to render to the document. To do this, we add code to the form's onLoad event.

- Line 1 Creates a new instance of the NotesView2 object. We pass the DB_NAME and VIEW_NAME parameters that we declared earlier in the [HTML Head]
- Line 4 We want to display the selection margin, so we set hideSelectionMargin to false. To hide the selection margin, set it to true
- Line 5 This line tells the NotesView2 object what to do when a user double clicks a document in the view. openDocument is an event that is called when a user double clicks a view entry that is a document (the event is not dispatched for categories, but is dispatched for parents of response documents). The openDocument event is used by declaring a function that serves as an event handler. The parameter passed to the event handler function is a NotesViewEntry2 object, which represents a ViewEntry in the NotesView2 object.
- Line 6 In this example, we open the document represented by the NotesViewEntry2 object by replacing the current browser window's location with a URL that points to the Database, View, and document's UNID (for example '/NotesView_217.nsf/cars/1167EB12799288208525739300056126?Opendocument')
- Line 8 The actionBarDisplay property instructs the NotesView2 object how to render action buttons.
 - 1 – Icon & Text
 - 2 – Icon Only
 - 3 – Text Only
- Line 9 The linesToShow property controls how many lines will be displayed in the NotesView2 object's
- Line 10 The linesToGet property controls how many view entries at a time NotesView2 will request from the domino server each time it requests information.
- Line 11 The extendLastColumn property, when set to true, tells NotesView2 to extend the last column the entire remaining width of the view if space is available.
- Line 12 thru Line 16 add actions to the action bar of the NotesView2 object. The addAction function takes two parameters, the first is the label of the button, and the second is a callback function that the NotesView2 object will call when the user clicks the button. In this example, the script adds 5 buttons, one for each of the built in functions of the NotesView2 object to Expand All categories, Collapse All categories, Refresh the view, Jump to a specific key in the view, and to turn on the "Search this view" bar.
- Line 17 The render function tells the NotesView2 object to draw itself onto the document. The parameter passed is the HTML element we placed on the Domino form as a 'container' in the previous step.

```
1  oView = new NotesView2(DB_NAME,VIEW_NAME);
2
3  //set some extra parameters (optional)
4  oView.hideSelectionMargin = false;
5  oView.openDocument = function(viewEntry){
6      document.location.href = DB_NAME + "/" + VIEW_NAME + "/" + viewEntry.unid + "?Opendocument";
7  }
8  oView.actionButtonDisplay = 3;
9  oView.linesToShow = 10;
10 oView.linesToGet = 25;
11 oView.extendLastColumn = false;
12 oView.addAction("Expand All", function(){oView.expandAllEntries()});
13 oView.addAction("Collapse All", function(){oView.collapseAllEntries()});
14 oView.addAction("Refresh", function(){oView.refresh()});
15 oView.addAction("Jump To", function(){oView.jumpTo()});
16 oView.addAction("Search", function(){oView.toggleSearchBar()});
17 oView.render(document.getElementById('viewHolder'));
```

Code 6 JavaScript included in the form's onLoad event

That's it!! Launch your form in your favorite browser, and you should be all set!

More documentation to come soon ... including advanced examples for handling selected documents and more.