

Multi Select Dialog Picker XPage Component

Steve Castledine

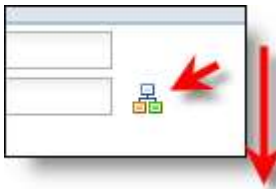
v1.0 March 15th 2010

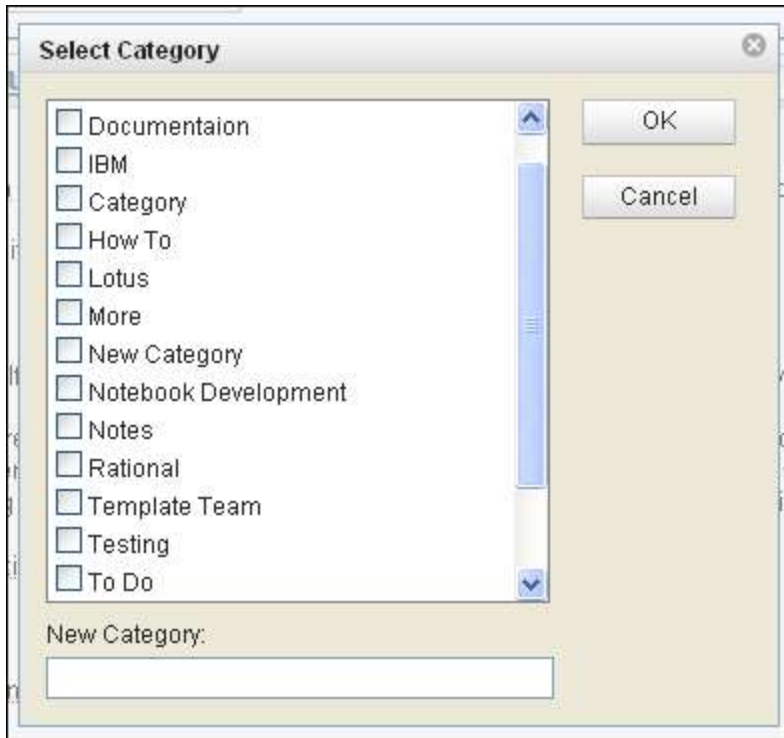
Features

The multi-select dialog picker allows you to specify a Lotus Notes View/column based data source, or via an api any list you may choose and return a list of selected values.

Further features:

- 1) The picker can easily be bound to an existing editable field or values returned via api.
- 2) Allows users to add new values, or via property disallow this function
- 3) Multiple use on one page.
- 4) Data can come from existing database, db on existing server, or db on another server
- 5) Background colors, list height and width, titles etc can be set via properties
- 6) Extends Dijit.dialog so can be hidden, shown etc using usual dojo api
- 7) Self contained – so just one custom control to drag and drop (no code or css files to copy as well).
- 8) Cross browser and XPages Client compatible





Using

Simply copy the control to your application and drag and drop into your application. I would normally add to the bottom.

It will work without changing the defaults but will need a viewname and column specifying for the lookup, or data added via api.

You may want to bind it to an existing editable field, see section below.

As it extends Dojo's dijit.dialog, by default hiding or showing is done like this (from say a button). Hiding is handled automatically within the control. See property "dialogID" for multiple use.

```
dijit.byId('multiSelectDialog1').show();  
dijit.byId('multiSelectDialog1').hide();
```

Attaching to View Column for the selectable List

To use a view column in the existing database, you need to supply the viewname in property "viewName" and the column number in "column".

The other thing to ensure is the data separator property "multipleSeparator" matches that used within your field. The default is a comma.

If the lookup is going to be in a different database, then supply the database in property "dbName". If the database is on another server then also specify the server in property "serverName" in the format "SERVER//OU//O" etc.

Binding to a field

There is a property “input Binding” that you can specify the id of an existing editable field, which in turn you have bound to a field in a data source which when set will automatically retrieve and set values to.

If you leave blank then you can just use the api to set and retrieve values.

Properties

The control has a number of properties allowing you to customize it:

custom	
allowNew	true
backgroundColor	#ECE9D8
column	1
dbName	
dialogID	multiSelectDialog1
dialogTitle	Select Category
inputBinding	categories
listBackgroundColor	#FFFFFF
listHeight	250px
listWidth	250px
multipleSeparator	,
newLabel	New Category:
serverName	
viewName	By Category

allowNew: Defaults to true, setting to false will disable the “add new” field.

backgroundColor: Defaults to Notes standard color. Sets the background color of the dialog.

column: Lookup Column in specified View

dbName: Optional. Used if view lookup is in another database.

dialogID: Default is “multiSelectDialog1”. If you want to use multiple versions on one page you must specify unique id's, so “multiSelectDialog2” etc

dialogTitle: Title at the top of the Dialog. Default is “Title”

inputBinding: The name of the editable field you want to automatically retrieve and populate data with. Leaving blank allows you to use api instead.

listBackgroundColor: Sets the background color of the list area. Default is white.

listHeight: Sets the list height. Default is 250px. You need to specify the units (px etc)

listWidth: Sets the list width. Default is 250px. You need to specify the units (px etc)

multipleSeparator: Specifies the separator used in the field used for the view lookup

newLabel: Label used above the “new entry” field. Default is “New:”

serverName: Optional. For if the view for the selection is on another server.

viewName: View name for lookup

API

You can insert and/or retrieve values from the control using the api. You can also tell it to sort, refresh etc.

You can get the object using: `var control=dijit.byId('multiSelectDialog1')`
(or other id if changed)

`setSelectedValues()` - Sets the selected Values from the bound editable field

`getSelectedValues()` - Gets selected values as a sorted array

`addItem(string)` - Adds string value to existing “selectable” list (needs a refresh to update)

`removeItem(string)` - Searches and removes item of string value if found (needs a refresh to update)

`sort()` - Sorts existing selectable list (needs a refresh to show)

`refreshList()` - Refreshes visible selectable list

Test XPage

As well as the custom control “”, there is a test Xpage “test.xsp” contained within the nsf file which demonstrates the control in use (and multiple use).

