Implementing drag and drop in XPages using Dojo

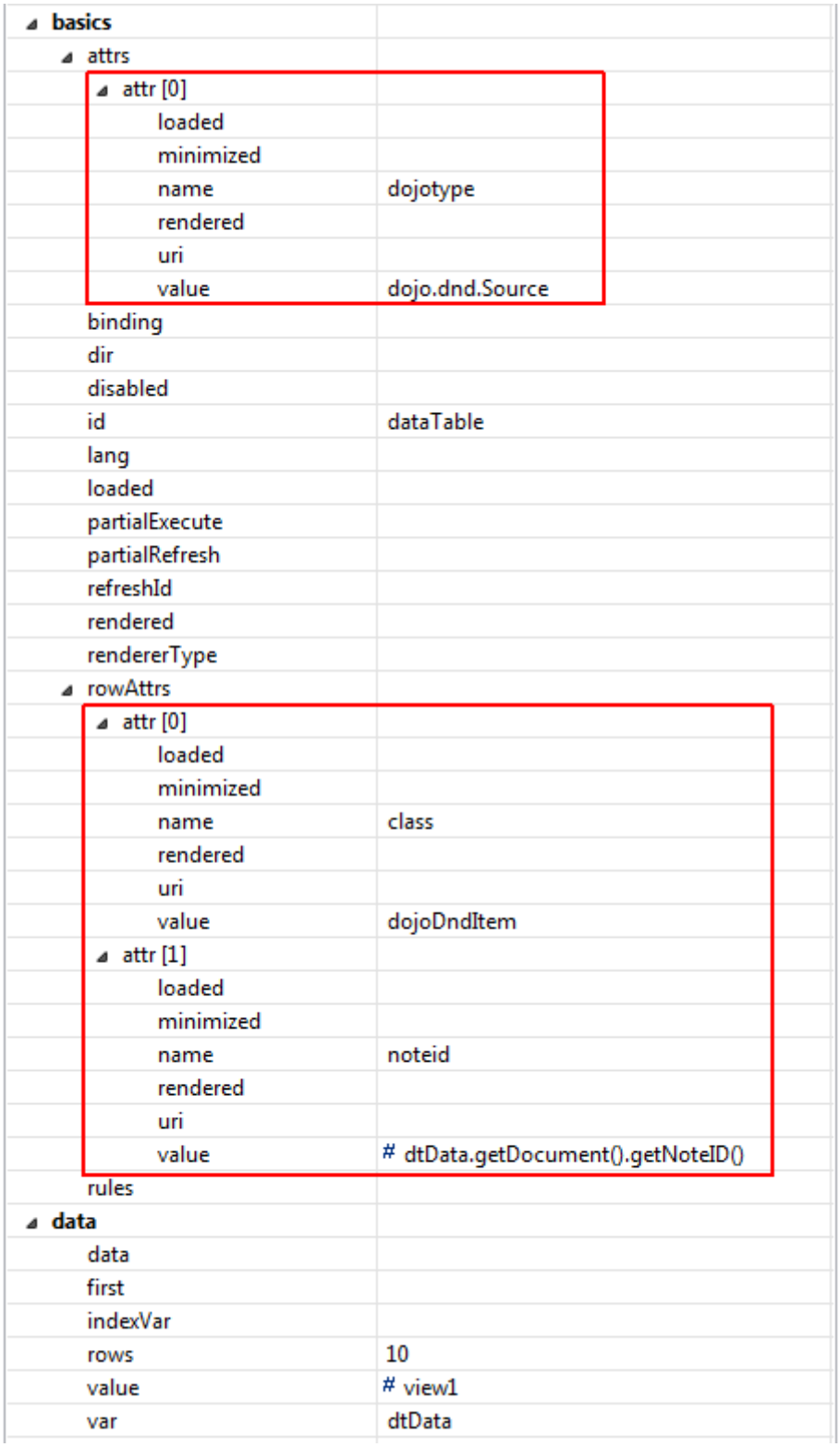
August 6, 2013

Drag and drop can very useful for users and in case of Notes/Domino there are operations like moving documents to folder or trash which can utilize this technique. In this article I would be implementing drag and drop in XPage using Dojo.

In any drag and drop operation there are two elements – first the element which would be dragged and other are the containers where it would be dropped. Dojo provides *dojo.dnd.Source* to define element which would be dragged and *dojo.dnd.Target* as the containers.

Before we start we need to load the Dojo module *dojo.dnd.Source* and its appropriate style sheet. This can be done with below code:

Now lets create our draggable elements. For this we use the Dojo class *dojo.dnd.Source* and assign the *class* attribute as *dojoDndItem* to each of draggable items. I would be using a data table control in this example. In data table we can use the *attrs* property to set *dojotype* to *dojo.dnd.Source* and *rowAttrs* to set the *class* to *dojoDndItem* for each of the individual rows. This would make each of the rows draggable. Also we would be adding another attribute of *noteid* in *rowAttrs* and assigning them note IDs of the documents displayed in each row. These note IDs would be used to process the dragged documents (explained later in the article). So our data table properties would look like this:



And our data table source code would look like this:

```
                    Name



                    Address
```

This is how the rendered HTML for data table looks like. We would use the attribute *noteid* later in our code.

```
▼<table id="view:_id1:dataTable" class="xspDataTable dojoDndSource dojoDndTarget dojoDndContainer" dojotype="dojo.dnd.Source">
  ▶<thead>…</thead>
  ▼<tbody>
    ▼<tr class="dojoDndItem" noteid="F12" id="dojoUnique1">
       <td class="xspColumn">Adams, James</td>
       <td class="xspColumn">1549 Cambridge Drive</td>
     </tr>
    ▶<tr class="dojoDndItem" noteid="F76" id="dojoUnique2">…</tr>
    ▶<tr class="dojoDndItem" noteid="AFE" id="dojoUnique3">…</tr>
    ▶<tr class="dojoDndItem" noteid="100E" id="dojoUnique4">…</tr>
    ▶<tr class="dojoDndItem" noteid="C8E" id="dojoUnique5">…</tr>
    ▶<tr class="dojoDndItem" noteid="1016" id="dojoUnique6">…</tr>
```

Now we need to create a container where elements would be dropped. For this we use the Dojo class *dojo.dnd.Target*. I would be creating a deletion functionality where dragging documents in the container would delete them. We can use a *<xp:div>* (with an image of trash inside it) to create our container, like this for example:

```

```

After this we need to catch the drop event which would be fired when drop operation is performed in *dojo.dnd.Target*. For this we need to write JavaScript code for the event *onDndDrop*.

```
var divTrash;
function init() {
        divTrash = new dojo.dnd.Target("#{id:divTrash}");
        dojo.connect(divTrash, "onDndDrop", targetDrop); // Connect the onDndDrop event to target
}
```

The above code would call the function *targetDrop* (we would define it later in this article) every time a drop operation is performed. We would be calling *init()* function on load of the page.

```
XSP.addOnLoad(init);
```

In our function *targetDrop* we need to do a few things:

1. Check if this is the right handler for this drop event
2. When we drop elements in container they start showing up in it so we need to remove them
3. Collect all the note IDs of the dropped elements
4. Pass the note IDs to server event handler which would process it

```
function targetDrop(source, nodes, copy, target) {
	// To verify that this is the right handler for drop event
	if (dojo.dnd.manager().target !== this) {
		return;
	}

	// Remove the dropped elements
	divTrash.selectAll();
	divTrash.deleteSelectedNodes();

	// Get all the note IDs and put them in an array
	var notesIDs = new Array();
	dojo.forEach(nodes, function (node, i) {
		notesIDs.push(dojo.attr(node, "noteid"));
	});

	// We would define the code to call server event handler later in the article
}
```

Once we have the note IDs of the dropped documents we need to execute server side JavaScript code which would delete those documents. For this I would be using the technique described in this article which allows us to create event handlers in XPage and execute them via client side JavaScript.

Below is the function we would use to execute server event handler:

```
XSP.executeOnServer = function () {
	// Source: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=%2Fcom.ibm.designer.domino.

	// the event handler id to be executed is the first argument, and is required
	if (!arguments[0])
		return false;
	var functionName = arguments[0];

	// OPTIONAL - The Client Side ID that is partially refreshed after executing the event handler
	var refreshId = (arguments[1]) ? arguments[1] : "@none";
	var form = (arguments[1]) ? this.findForm(arguments[1]) : dojo.query('form')[0];

	// catch all in case dojo element has moved object outside of form...
	if (!form)
		form = dojo.query('form')[0];

	// OPTIONAL - Options object containing onStart, onComplete and onError functions for the call to the
	// handler and subsequent partial refresh
	var options = (arguments[2]) ? arguments[2] : {};

	// OPTIONAL - Value to submit in $$xspsubmitvalue. can be retrieved using context.getSubmittedValue()
	var submitValue = (arguments[3]) ? arguments[3] : '';
```

```
        // Set the ID in $$xspsubmitid of the event handler to execute
        dojo.query('[name="$$xspsubmitid"]')[0].value = functionName;
        dojo.query('[name="$$xspsubmitvalue"]')[0].value = submitValue;
        this._partialRefresh("post", form, refreshId, options);
}
```

Now we create our event handler which would delete the documents which were dropped and partially refresh our data table. We would get the note IDs using *context.getSubmittedValue()*.

To execute our event handler we would call the function *XSP.executeOnServer* passing the ID of event handler, ID of data table and note IDs that need to be processed.

```
XSP.executeOnServer("#{id:ehDeleteDocs}", // Event handler to be executed
        "#{id:dataTable}",  // Data table to be partially refreshed
        {}, // onStart, onError and onComplete events
        notesIDs.join(",") // Value to be submitted so that SSJS code can process it
);
```

And we are done! Now if we drop the documents from data table into our container it would delete them and partially refresh the data table.

We can enhance the UI by over riding existing CSS classes provided by Dojo like mouse over.

```
.dojoDndItemOver {
        background-color: #eeeeee;
}

.dojoDndItemSelected {
        background-color: #ffffC6;
}

.dojoDndItemAnchor {
        background-color: #ffffC6;
}
```

Also to give a visual indication to user that drop is performed successfully we can animate the trash icon in our target.

```
// Animate the trash icon shrink-enlarge
var propsShrink = {
        width: {start: "94", end: "80", unit: "px"},
```

```
        height: {start: "94", end: "80", unit: "px"}
};
var propsEnlarge = {
        width: {start: "80", end: "94", unit: "px"},
        height: {start: "80", end: "94", unit: "px"}
};
dojo.anim("#{id:trashIcon}", propsShrink, 250);
dojo.anim("#{id:trashIcon}", propsEnlarge, 250);
```

The above code animates the trash icon shriking and then resizing it to its original size.

DOWNLOAD DEMO DATABASE

The documents in this database have been taken from the View Picklist Custom Control on OpenNTF.

August 6, 2013 [http://www.pipalia.co.uk/implementing-dojo-drag-and-drop-in-xpages/] | by Naveen Maurya | in Dojo, Domino Designer, Drag and Drop, JavaScript, Server side JavaScript, SSJS, xpages .

Comment