XPages development practices: developing a common Tree View Custom Controls

Use XPages develop a common style of user control Dojo

Level: Intermediate

Zhan Yonghua, Software Engineer, IBM China Software Development Center WPLC
Yang Zhilei, Senior Software Engineer, IBM China Software Development Center WPLC
Zhang Jia, Software Engineer, IBM China Software Development Center WPLC

August 10, 2009

This paper describes how to use Lotus ® Domino Designer 8.5 in XPages develop a common Tree View style custom control used to display Domino view hierarchy. Through this article, the reader can be applied to a new generation of XPages technology, the Web-based Domino applications, the development of flexible, reusable custom controls, and able to understand XPages is to facilitate the application of Dojo toolkit and JavaScript libraries.

Background

This section will be used in this article a brief introduction of related technologies. For more information about the reader a more detailed content, you can visit the IBM Web site or a related organization.

XPages: Domino 8.5 a powerful Web design elements

XPages is the IBM Lotus Domino 8.5 in a powerful new design elements, we can fully use it for our projects present a Web client user interface. XPages Using a new JSF (Java Server Face) rendering engine, and the traditional Domino Web development compared, XPages is undoubtedly a powerful Web design elements.

Domino 8.5 for Eclipse IDE provides an integrated visual development environment. Use XPages development of Web-based Domino applications, get "WYSIWYG" user experience. Each of the XPages a page control object, we can directly visualize a way to modify their properties, and real-time preview; Accordingly, the object supported by the event, XPages also provides a classified guide to help users to edit and manage incident response script.

XPages greater advantage is reflected in the database on the Domino data binding. We can for each page or embedded panel to create a "data source", this data source can be a Domino database, a document or view. Then, through a simple operation can be the data source data, such as a document field value, and dynamic binding to a Web page object. Of course, by writing a script, we can also calculate the data processed and then bind it with page elements.

Just this way, for the development of Agent used to manipulate Domino data users, XPages provides flexible programmability and scalability. Users can integrate the Eclipse IDE to develop their own Java library and then use XPages support server-side JavaScript to invoke the Java function, the client generates the user wants data.

In this article, we introduce how to apply XPages and Domino toolkit Dojo 1.1.1 is included to develop a common XPages custom controls, using Dojo's Tree Widget to show the relationship between the level of the traditional Domino view. In introducing the method of process, I believe the foregoing XPages users can experience the power and flexibility.

Domino View

Domino view is a list of document collections, it is rich in depth the structure and expression makes it a powerful tool for Domino database browsing. Like with the document, Domino view supports both Notes client access methods, and also supports Web browsing. (In Designer open the **"View> AllExpences",** in

the tool bar, select **"Preview in Web browser.")** Default Web view of Web 2.0 has been far from being able to meet the needs of users. Even more inconvenient is that its data source and page format is completely determined by the view already in the database, the user can not do and flexible customization. So a lot of Domino Web application developers will choose to develop their own Web views, in the traditional Domino Web development, this is not an easy task.

Dojo and dijit.Tree

Domino provides a Dojo toolkit, can the server (C: \ IBM \ Lotus) \ Domino \ data \ domino \ js \ directory to find it. The XPages using Dojo toolkit, we can easily XPage page by adding Web 2.0 elements, such as the Widget, animations and so on.

Dojo is a powerful object-oriented, open-source JavaScript toolkit. Dojo for the development of Web applications provides a great deal based on DHTML, CSS, and JavaScript technologies such as client-side components. Dojo project development objective is to address the initial development of DHTML applications encountered in those long-standing historical issues. Such as: cross-browser issues. Therefore, the use of Dojo, easier to make your Web pages with dynamic capabilities, or in any solid support for JavaScript language environment play a role. The Dojo in code quality, operating performance, and documentation support in the development of similar projects can be said to be second to none. In this article will be applied dijit.Tree object is used to display the tree Dojo hierarchy Widget. It uses the data source is JSON format.

JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format. Is easy to read and write. But also easy for machines to parse and generate. It is based on a subset of JavaScript.

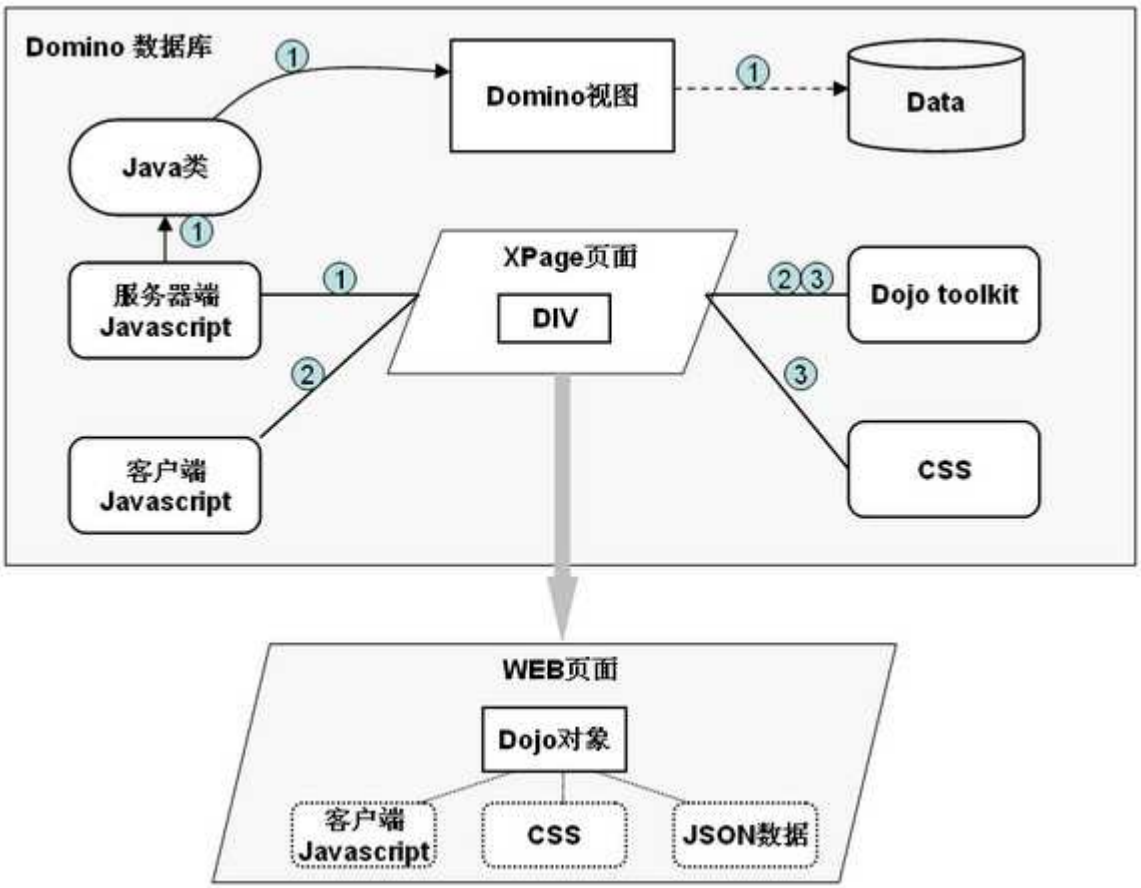
JSON object is an unordered " 'name / value' pairs of" collection. An object with "(" begins with ")" end. Each "name" followed by a ":","' name / value 'pairs "between use", "separated. JSON array is the value (value) of the ordered collection. An array of "[" start "]" end. Between the values use "," separated. JSON value (value) can be enclosed in double quotes string (String), value (number), true, false, null, object (object) or an array (array). These structures can be nested.

Systematic framework and methodology

Before all the steps at the beginning, we need to introduce the first general chapter involved in the XPages elements and their relationships. Such as the server-side / client-side JavaScript, Java classes, CSS, etc. The development and use.

Shown in Figure 1, server-side Domino database design elements and the client run-time Web page elements have been listed out. XPage page DIV objects will be interpreted as the Dojo client-side objects. By the icon, we can clearly see the data about it (1), action (2) and appearance (3) definition.

Figure 1. System Framework



Dojo Objects

The XPage page, we define a DIV object map corresponding Dojo. There are two main ways you can bind an object DIV and Dojo.

First, in the DIV directly using 'dojotype' attribute defines Dojo object type, the following example defines a dijit.Tree object, what we often say Dojo Tree Widget.

Listing 1. Dojo object definition

```
<div dojoType="dijit.Tree" id="mytree" model="continentModel" showRoot="false"> </ div>
```

Listing 2. Dojo object dynamic binding

```
<! - XPage page - "
<div id="piechart" style="width: 250px; height: 250px;"> </ div>
"! - Client-side JavaScript ->
var chart1 = new dojox.charting.Chart2D ( "piechart")
```

Observant readers may find in the above two examples, we directly use the HTML of the <DIV> tags, but did not use XPages in <XP :DIV> label. This is not a random choice. On the first method, we use the 'dojotype' property, which is acceptable <DIV> label attribute definition, but it is not acceptable to <XP :DIV> label attribute definition, XPage error page will appear; right The second method, if we use the words of <XP :DIV> label, XPages engine will put its id attribute value parsing into a similar view: _id1: _id2: piechart form, so that dynamic binding phase of Dojo will be because he can not be found id for the object of piechart bind failed. Interested users can try both cases, their own results, deepen XPages tag in HTML tags and XPage understanding.

So, **when the page in the XPage adding Dojo object, be sure to use the HTML tags.**

Data

In this case the data from Domino view, in addition, data can also be derived from the Domino document or other relational database. Figure 1 shows the (1), indicating the route that we use Domino view data the way: Using Java classes to read Domino view data, and data structures required for the Dojo object JSON Schema format; re-use server-side JavaScript call Java classes obtained these data, and store it in XPage page in the client-side JavaScript variable.

Action

Figure 1 in the icon (2) to indicate the object of the Dojo client-side JavaScript action to respond to the source: Dojo package against the object and user-defined client-side JavaScript. In practical applications, the user is usually through the preparation of client-side JavaScript, to target support for the Dojo client-side user events, such as `onClick onChange` etc. join the movement response.

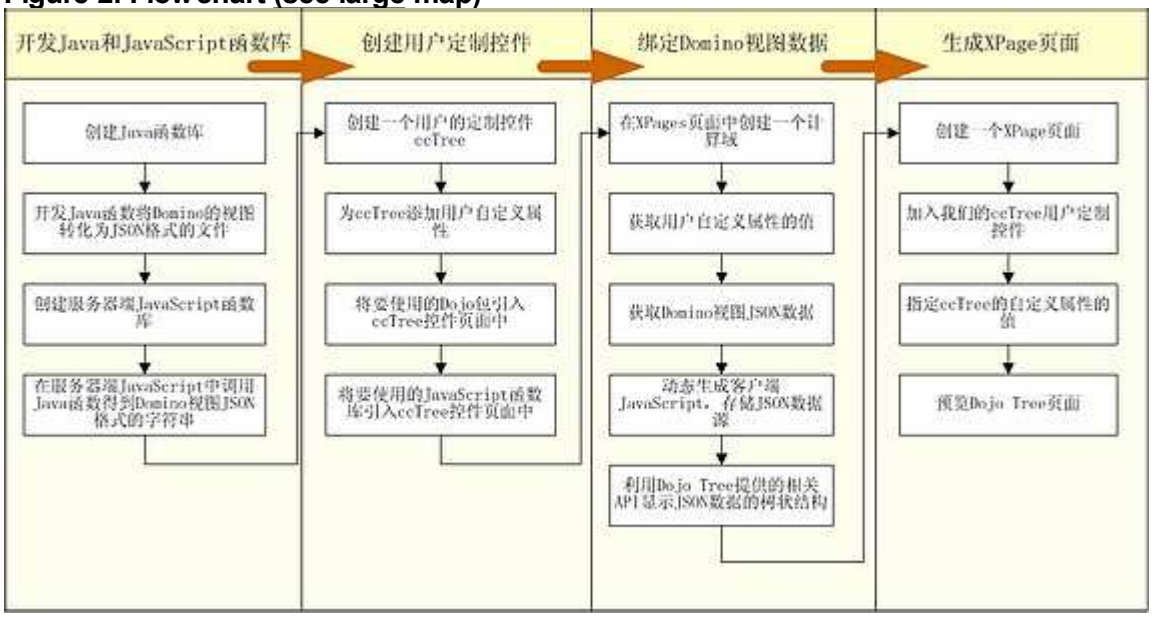
Appearance

Figure 1 in the icon (3), indicating a decision Dojo object format, the source of the appearance: Dojo package on the appearance of the object definition and user-defined CSS. In order to change the default appearance of Dojo defined display format, the user needs to "override" approach to re-define the object's CSS class files.

Domino displayed using the Dojo Tree View

Of this article Part 3 will detail how we extract data from a Domino database, and will bind the data with the Tree view shows up. Let us follow the Figure 2 flowchart examples of steps to build a Dojo Tree Widget for users to customize the control bar.

Figure 2. Flowchart (see large map)



Domino view for JSON data

Domino view has its own JSON data using the URL command parameters, we can see the view of the JSON data on Domino. On the Web browsing from a Domino view, only the final URL in the browser side to add parameters `?ReadViewEntries&outputformat=JSON` you can see its JSON format.

However, Domino view JSON data with Dojo Tree of JSON Schema is not uniform, so we can not directly use the Domino view data as a JSON object Dojo Tree of data sources. The following diagram of the Domino view JSON format shown in Figure 3 (a), if you want to construct Dojo tree, it takes in Figure 3 (b) format JSON.

Figure 3. Domino view JSON and Dojo tree JSON

We can use Java classes and server-side JavaScript to achieve these two different JSON format conversion.

First of all, create a Java class for Domino view data. Domino Designer 8.5 based on the Eclipse platform, you can easily create Java classes. Through the **"window" Open Perspective> Java** "Select Java window.

On the right to choose their own projects tree.nsf, created under the project related to Java class files. In this case, we created two classes `TreeNode` and `OrganJSON` a specific code that can be found in the appendix of this article. Which, `OrganJSON` class `getOrganTree` method to achieve the function of the Domino's view hierarchy into a tree structure, and the return type for the `TreeNode` root node, through the tree's root node, we can traverse all the nodes in the tree; getTreeJSON method to achieve was started by the root of the tree traversal, and finally generate the Dojo Tree able to identify the JSON format string.

Then we in the Designer to create a server-side JavaScript is used to call just to create a Java class method. Through the **"window" Open Perspective> Domino Designer** "Back to the Designer of the work area. In the "code -> Script Library" and select "New Server JavaScript library" to create Tree.jss file. In this file, create a `getJsonString` function, in this method, we need to pass Domino the view name and view the name of a column, this column's name is used in the tree structure shows that in this example you can choose name column can also choose to email column.

Listing 3. JavaScript: getTreeJSON function

```
function getJsonString (view, column) (
    var viewName: String = view;
    var columnName: String = column;
    var organJSON: com.ibm.test.tree.OrganJSON = new com.ibm.test.tree.OrganJSON ();
    var organTree: com.ibm.test.tree.TreeNode = organJSON.getOrganTree (session,
        "tree.nsf", viewName, columnName);
    var organTreeStr = organJSON.getTreeJSON (organTree);
    return organTreeStr;
)
```

Domino view data binding

The next step in the XPage page to add Dojo data sources, we adopted a computational domain to achieve the code is as follows:

Listing 4. JavaScript: using computational domain to obtain a server-side data

```
<xp:text escape="false" id="computedField1">
    <xp:this.value>
        <! [CDATA [# (javascript: var viewName = compositeData.ViewName;
        var columnName = compositeData.ColumnName;
        var json = getJsonString (viewName, columnName);
        var result = "<script language=\"JavaScript\" type=\"text/javascript\">"
        result + = "var jsonStore = new dojo.data.ItemFileReadStore ((data:" + json + "});"
        result + = "</ script>"
        return result ;}]]>
    </ xp: this.value>
</ xp: text>
```

We first get introduced custom control's properties ViewName and ColumnName (relating to custom controls and properties description, refer to Part 4), and then created by calling our previous server-side JavaScript, methods json=getJsonString(viewName, columnName); to generate JSON formatted data source.

With the JSON data source, then we only need to control the page using Dojo Tree provided by the API can be related to the tree structure of our show out. Add the following criteria in the source code of HTML. Where store="jsonStore" is defined in the previous calculation of the domain off jsonStore, which is used in this example the data source. query="{name:'0'}" is read from the data source jsonStore the root node as the root element of the tree structure. <Script> Leaf node is added to the action, when the click leaf node is a alert dialog box appears, showing the contents of the selected node's label.

Listing 5. XPage the Dojo Objects

```
<div dojoType = "dijit.tree.ForestStoreModel"
    jsId = "continentModel" store = "jsonStore" query = "(name: '0 ')">
</ div>

<div dojoType = "dijit.Tree" id = "mytree" model = "continentModel"
    howRoot = "false" openOnClick = "true">
    <script type="dojo/method" event="onClick" args="item">
        alert ( "Execute of node" + jsonStore.getLabel (item) + ",
        population = "+ jsonStore.getValue (item," population "));
    </ script>
</ div>
```

Creating Custom Controls

Creating Custom Controls

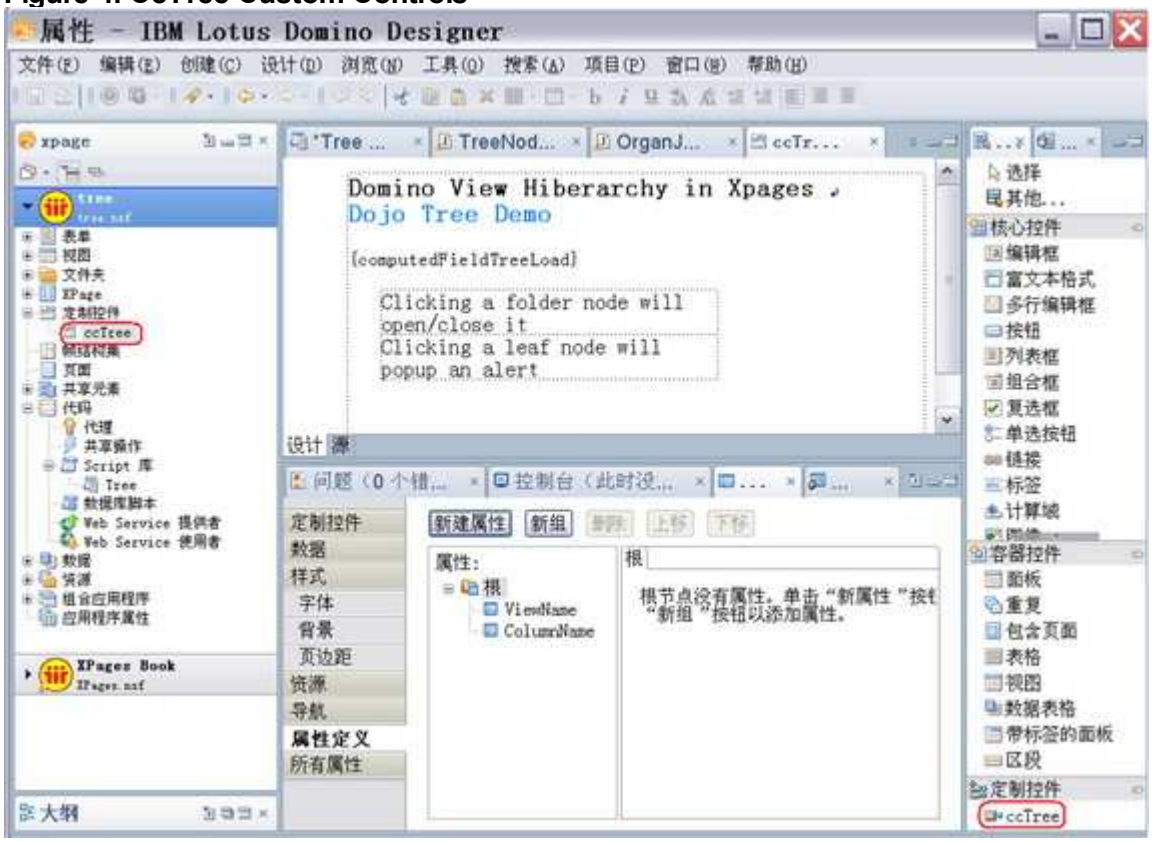
The XPages, the user is a custom control allows the user to define, edit, and pages in different XPage reusable components. It is similar to Domino template design in the "sub-template", or Web page design in the panel. The difference is that it provides more powerful features and customization. For example, a way to support users to drag and drop directly XPages "core control," grouped together; support user-defined control properties; to manage and support users to drag and drop way to reuse and customize the control. If the user-developed Web applications, some parts will be used repeatedly, such as Header, Menu bar,

Navigator, or as we are creating the display Domino View hierarchy common Dojo Tree, then create a reusable " Custom Control "is the best choice.

By right, **"control"** select **"New Custom Controls"** ccTree, shown in Figure 4. In the work area, we can "design" and "source" two page edit it. In the work area below, you can see the control's properties, events. Map the lower right corner, in the custom control panel in the management, ccTree be listed on the inside, while the other Xpage want to create a custom page or other controls, we can drag it directly through the work area re-use it. In other words, custom control to support nested.

Custom control also allows user-defined attributes, use it, by passing a different attribute values can be customized to achieve it. The Figure in the ViewName and ColumnName is what we define as ccTree properties. ViewName represent want to show the name of the Domino view, ColumnName represents the tree structure used to display the column's name.

Figure 4. CcTree Custom Controls



In order to use the Dojo Tree in the XPages elements, the need to use the Dojo package introduced to control through the **"Attributes" resources "Add Script Library"** button, add the use to the `dojo.parser dojo data ItemFileReadStore` and `dijit.Tree` package. Can also choose to edit the source achieved by directly.

Listing 6. XPage adding resources

```
<xp:this.resources>
    <xp:dojoModule name="dojo.data.ItemFileReadStore"> </ xp: dojoModule>
    <xp:dojoModule name="dojo.parser"> </ xp: dojoModule>
    <xp:dojoModule name="dijit.Tree"> </ xp: dojoModule>
</ xp: this.resources>
```

At the same time, in order to be able to use our previous definition of JavaScript libraries, but also it needs to be introduced to control the same through **"Attributes" resources "to add the script library"** to achieve. Of course, you can also choose to pass directly into the source `<xp:this.resources>` have added `<xp:script src="/Tree.jss" clientSide="false"></xp:script>` statement to achieve.

ccTree custom control content of the document please refer to the annex.

Reused in the XPages control ccTree

We created a custom control ccTree is now ready to use. Let's look at how to use it is convenient and flexible bar!

We need to do is just:

1. Create a new XPage page.
2. Be ccTree control from the custom control panel, drawn into the work area (need to switch to the "Design" page to see the implementation drag and drop).
3. Incoming parameter. Switch to the "source" page, find the location of the control by adding Domino view needs to display the column name and the name. The code is as follows, in this case, the database tree.nsf, there is one called Contacts view shows a list of contacts, including Name column is the contact's name.

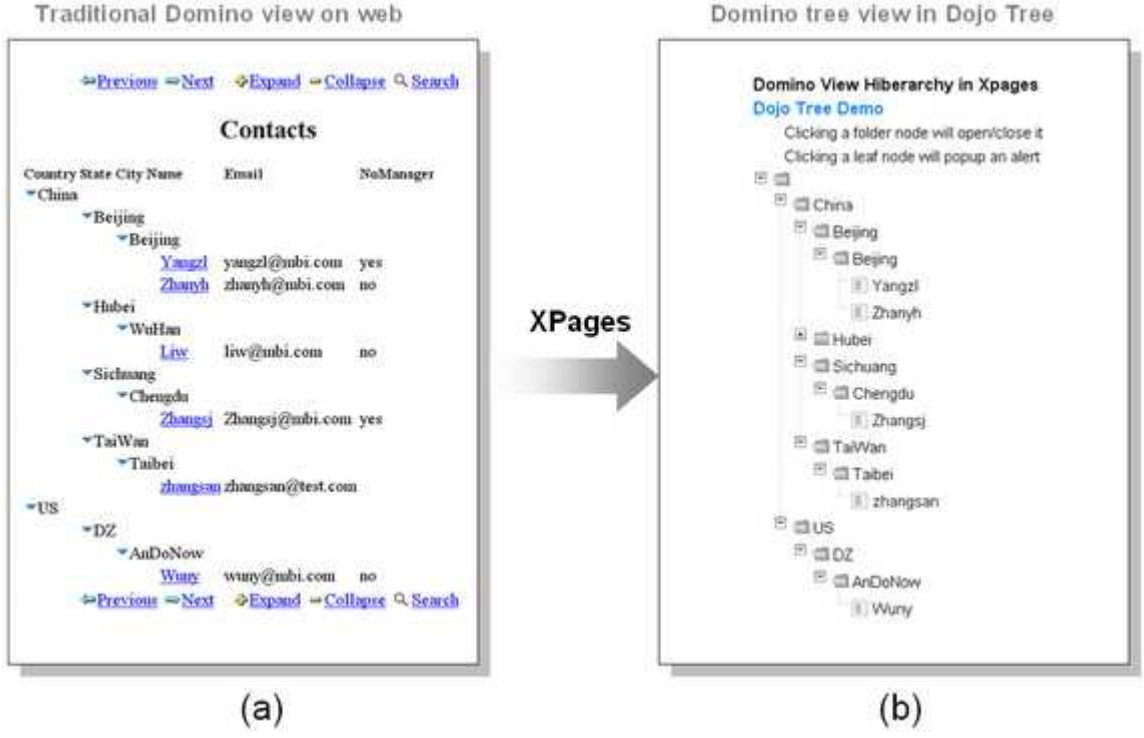
```
<xc:ccTree ViewName="Contacts" ColumnName="Name"> </ xc: ccTree>
```

After the preview of this XPage page, you can see in Figure 5 (b) of the Dojo Tree tree, displaying the Contacts view hierarchy, in the leaf node shows the contact name. This example, if you want a leaf node displays contact e-mail address is very simple! ColumnName in the above code will set the value of Email; if you want to display another view, also very simple to change the above code in the ViewName and the ColumnName value on it.

Example results show

In order to enable readers to more intuitive understanding of this achieved Dojo Tree custom controls, we give in Figure 1 for examples of renderings. Figure 5 (a), is supported by Domino default Web view; Figure 5 (b), is to use custom controls in this article, in the XPages generated Dojo Tree style view level. This Web page is more in line with Web 2.0 user's habits. More importantly, it XPages the form of custom controls to the user, very easy to reuse and customization, you can easily re-used in any XPage page, by passing a small amount of parameters can be used to display any Domino view.

Figure 5. Example: show contact list using Dojo Tree



How to use the article in the source file

In this paper, used in the source file. Include:

1. Java library, Java class files: OrganJSON.java, TreeNode.java
2. Server-side JavaScript library: Tree.jss (code> Script Library> Tree)
3. Custom Controls: ccTree (custom control> ccTree)

How to use these source files:

1. Open Domino Designer 8.5, create a new database tree.nsf (if you want to preview in Designer, you need to tree.nsf copy the Data directory to the Notes).
2. Switch to the Java perspective, the Java class files OrganJSON.java, TreeNode.java copy tree.nsf project directory.
3. Switch back to Desinger work area, custom controls ccTree and the JavaScript function library Tree.jss copy to your database.
4. When you need to join the database ccTree control XPage page in accordance with Section 4.2 generates "XPage page" approach by adding ccTree control, and configure its parameters, is used to display your existing Domino database views.
5. Select **"Design" in the Web browser preview. "**

Deployed to the Domino Server to browse:

1. Tree.nsf copied to the Domino Server's Data directory.
2. From the browser to access http:// (dominoserver host) / tree.nsf / Tree.xsp.

Conclusion

In this paper, a detailed example of how to use XPages develop a common Dojo Tree-style custom control used to display Domino view hierarchy. Through this article, the reader application XPages technology, the Web-based Domino applications, development of flexible, reusable custom controls, and the use and integration of Dojo's JavaScript library.

Download

Description	Name	Size	Download Method
Java class files used in this article: OrganJSON.java	OrganJSON.java	8 KB	**HTTP**
Java class files used in this article: TreeNode.java	TreeNode.java	4 KB	**HTTP**
This paper used server-side JavaScript: Tree.jss	Tree.jss	4 KB	**HTTP**
This paper used XPages custom control: ccTree.xc	ccTree.xc	4 KB	**HTTP**

→ Information about download methods

References

Learn

- See the article "XPage Keys": Learn XPage main functions: XPage basic controls, basic properties, for Ajax and JavaScript support for the operation of the control, XPage event model.
- View tutorial "in the Lotus Domino Designer using XPages the power," XPage understand the basics, as well as how to build a View type of XPage, NotesDocument type of XPage, add XPage front-end and multi-document or delete.
- View tutorial "in the IBM Lotus Domino Web application development using XPages, Themes, and

Mashups": In this tutorial will combine the use of XPages and Themes to create an enterprise-class sales and distribution of tasks to track Lotus Domino Web application.

- See "developerWorks Dojo theme": the topic together with the Dojo-related technical resources.
- See the article "JSON Getting Started Guide": This article will quickly explain the JSON format, and through the code sample demonstrates how to distinguish between the client and server side JSON format for data processing.
- Lotus Domino Designer documentation: Read the White Paper, Redbook and more documentation.
- Best Practices for Building Domino 8 Web Applications: Read IBM Redbook.
- XPages in Domino Designer 8.5: Read Rob McDonagh on XPages's blog post.

Access to products and technologies

- Download trial version of Lotus Domino v8.5, the immediate experience XPage power.
- Download trial version of IBM Lotus Notes 8.5 and Domino Designer 8.5, try to use technology development XPage the first Domino Web 2.0 applications.
- Download trial version of IBM software and experience the powerful DB2 ®, Lotus ®, Rational ®, Tivoli ® and WebSphere ® software.

Discussion

- Participate in discussion forum.
- View developerWorks blog for the latest information.

Author

Zhan Yonghua, in the IBM China Software Development Center WPLC, the current related work in Lotus iNotes for Domino Web development and Web 2.0 with great interest.

Yang Zhilei, and now IBM China Software Development Lab Lotus Development Center and currently engaged in Lotus iNotes for testing. On the Domino Web development, Web 2.0 has extensive experience in related technologies.

Zhang Jia, and now IBM China Software Development Lab Lotus Development Center and currently engaged in Lotus iNotes automated testing. On the Domino Web developers, Dojo has extensive experience in related technologies.

- ○ Too bad! (1)
- ○ Need to increase (2)
- ○ General; can still (3)
- ○ A good article (4)
- ○ Terrific! (5)

Send us your suggestions or participate in discussions with others by sharing your thoughts.

Feedback

Top