IBM

# Using the Dojo widgets XPages

*IBM Lotus Notes / Domino application development technology, "XPages" in high-performance JavaScript library for "Dojo Toolkit" shows you how to use the provided widget*

Ono Makoto Journal ( Onoat@Jp.ibm.com ), Business Partner Technical Enablement Team Wplc, IM & Lotus Development / Software Development Laboratory, IBM

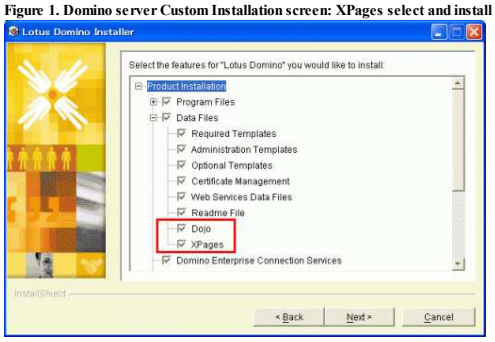
**Summary:** IBM Lotus Notes / Domino version 8.5 and appeared XPages a Web application development technology based on new technology. By using XPages, Lotus Notes / Domino can run on Web2.0 applications to perform efficient development of the style. In order to achieve XPages Web2.0-style pages, the open source Dojo Toolkit JavaScript library that has been adopted. The Dojo Toolkit has a variety of features, user interface among components of "widgets" (hereinafter, Dojo widgets) are rich in type, Web2.0 can be reused as part look combines style and functionality Masu. In this paper, how to use Dojo widgets in XPage, introduced with the sample.

**Date:** February 26, 2010
**Level:** Intermediate
**Please send your Feedback to feel free:** 0 (コメントを追加)

XPages in Dojo Toolkit

The XPages, Lotus Notes / Domino 8.5 (below, Notes Lotus is Notes, Domino Lotus Domino is referred to) which appeared, Notes / Domino running on the Web is a technology for developing applications using the technology. Cited as one of its characteristics, or type-of rich AJAX user interface, with features such as parts of the screen update Web2.0 is that easy to develop applications of the style. Web2.0 applications in implementing this style, XPages open source Dojo Toolkit is a library in JavaScript has been adopted, XPages are used in the internal implementation. As a result, Domino server during installation, Dojo Toolkit has to be installed by default. For custom installation, XPages to take advantage of the features shown in Figure 1 to select the Dojo and XPages, XPages Dojo Toolkit and make sure you also need to install.

**Figure 1. Domino server Custom Installation screen: XPages select and install Dojo**




The Dojo Toolkit installation script and thus dojo.js bootstrap is shown in Listing 1 can be accessed using the URL to. As mentioned earlier, Toolkit Dojo is also used because it is XPages inside, XPage you create a page with this without any explicit dojo.js now they are loaded on the page Masu (Figure 2).

**Listing 1. Domino server URL file dojo.js**

```
/ / If version 8.5 http:// <Dominosabanohosuto名> / domjs/dojo-1.1.1/dojo/dojo.js
/ / Http:// if <Dominosabanohosuto名> version 8.5.1 / domjs/dojo-1.3.2/dojo/dojo.js
```

**Figure 2. XPages page created in HTML Source: dojo.js being loaded by default**

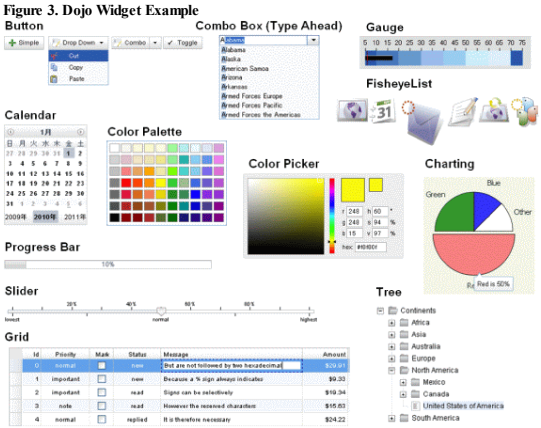
application to ensure compatibility well in advance is important.

---

Dojo Widgets

Dojo Widget, Dojo Toolkit provides the user interface (hereinafter, UI) components for. Button (Button), a combo box (ComboBox) to the basic widgets such as widget Charting chart display, which provides Grid editing tabular data display widgets, make data display widgets such as tree-style high-Tree to the widget, and many are available (Figure 3).

**Figure 3. Dojo Widget Example**




XPages before introducing in the Dojo widget usage, compared to a normal HTML page in how to use Dojo widgets, described in Figure 4, a Calendar widget example. In this example, Calendar Widgets is arranged with an input field, Calendar Widget when you click a date to be set as the date input field.

**Figure 4. Calendar widget example**


Calendar widget not only in general, how to use Dojo widgets Web page in two ways. One used in the HTML markup to declare a static Dojo widgets, and one way is to dynamically generated by JavaScript code. We used the former how to implement the HTML markup declaration. Figure 4 Source HTML Calendar widget will look like Listing 2 below.

**Listing 2. Calendar widget using HTML source example**

```
<html>
<head>
Uijettosanpuru <title> Calendar </ title>
<link rel="stylesheet" type="text/css" href="/domjs/dojo-1.3.2/dojo/resources/dojo.css">
<link rel="stylesheet" type="text/css" href="/domjs/dojo-1.3.2/dijit/themes/tundra/ tundra.css "> <! - b ->
<Script type = "text / javascript" src = "/ domjs/dojo-1.3.2/dojo / dojo.js"
    djConfig = "parseOnLoad: true"> </ script> <! - a ->
<script type="text/javascript">
dojo.require ("dijit._Calendar"); / / <! - c ->
dojo.addOnLoad (function () (/ / <! - f ->
    dojo.connect (dijit.byId ("calendar"), "onValueSelected", function (date) (
        var formattedDate = dojo.date.locale.format (date, (formatLength: "medium", selector: "date"));
        document.getElementById ("selectedDate"). value = formattedDate;
    ));
));
</ Script>
</ Head>
<body class="tundra">
    <div> <! - d, e ->
        <div id="calendar" dojoType="dijit._Calendar" dayWidth="narrow" style="width:200px"/>
    </ Div>
    <div> <input id="selectedDate" type="text"/> </ div>
</ Body>
</ Html>
```

Sample 1. Calendar widget to use

Basics As first presented in Figure 4 the same page Calendar widget example, XPages implemented using. This sample code presented in Listing 2 introduces a ~ f step instructions along. The final version of the sample application will Calendar.xsp.

djConfig = "parseOnLoad: true" attribute added, and the tundra theme loading

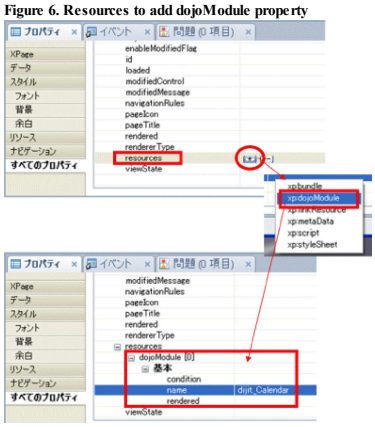
First, the sample code in Listing 2 points a and b are part of the implementation. As mentioned earlier in this paper, dojo.js XPage has not been created in a page loaded by default, djConfig = "parseOnLoad: true" attribute is not included by default. To add this, XPage canvas on the screen corresponds to the part design "XPage" selected open the Properties in Control, "all properties" tab in the **"base"**> "dojoParseOnLoad" property Set to true (Figure 5). The same "all properties" tab in the **"base"**> "dojoTheme" property set to true, tundra page will be loaded into the theme (Figure 5).

**Figure 5. DojoParseOnLoad dojoTheme and property to true**


dojo.require ("dijit._Calendar") add code

The following sample code in Listing 2 point c of dojo.require ("dijit._Calendar") implements a part. This part of the client-side JavaScript, so JavaScript client-side scripting such as the library in dojo.require ("dijit._Calendar") will work to write exactly the code that sets properties without writing code ways. Like the earlier "XPage" of the Property Control "all properties" in the tab, "Basic"> **"resources"** in the property, the "+" button **"xp: dojoModule"** adding, adds dojoModule or the "base"> **"name"** property you want to load Dojo widget class name is "dijit._Calendar" sets (Figure 6). This XPage when viewing this in HTML, dojo.require ("dijit._Calendar") that will be embedded JavaScript statements.

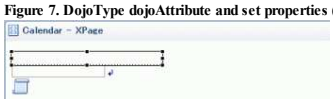
**Figure 6. Resources to add dojoModule property**

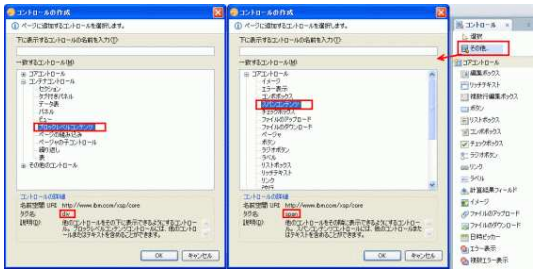



HTML markup dojoType attributes and additional attributes Dojo widgets

The following sample code in Listing 2 points d and e of implementing parts of the HTML markup. View the Calendar section of the widget, as shown in Figure 7, "Panel" control and place. "Panel" control is displayed in  HTML element. This "panel" and select the Properties of control "all properties" and open the tab, "dojo"> **"dojoType"** has a property called. Here, dojoType was specified in the attribute "dijit._Calendar" said the Calendar widget class name (Figure 7). The same "all properties" in the tab, **"dojo"**> "dojoAttributes" property attribute can Dojo widgets. Here dayWidth = "narrow" to add an attribute called, "dojoAttributes" property "+" and click **"dojoAttribute"** added one, **"name"** to "dayWidth" a, "value" to "narrow" set (Figure 7).

**Figure 7. DojoType dojoAttribute and set properties (Version 8.5.1 only)**





Please note that this property dojoAttribute and dojoType, Notes / Domino version 8.5.1 of the newly added property XPages. As a result, can not be available in version 8.5. In version 8.5 instead, <div dojoType="dijit._Calendar dayWidth="narrow"/> just as the HTML tag that embeds XPage. To embed, XPage the "source" and edit embedded (Fig. 9) or using the embedded field, or results (Figure 10) would either be.

**Figure 9. XPage the "source" and edit the HTML directly embedded (version 8.5)**

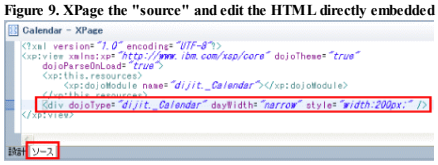
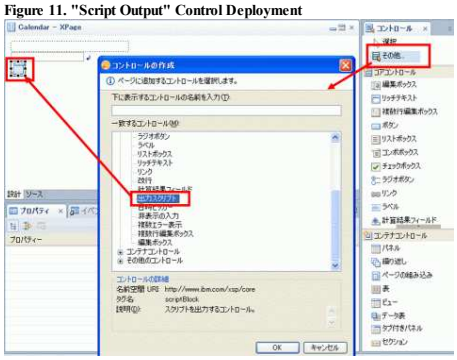


**Figure 10. As a result the field of custom embedded HTML (version 8.5)**

Adding logic using JavaScript dojo.connect

Finally, the point f of the sample code in Listing 2 implements the logic portion dojo.connect by JavaScript. XPage to embed JavaScript <script> of logic elements, "Script output" control is used. "Script Output" control by default "control" is not the view, "More ..." appears when you select the "Create Control" can be selected from the dialog (Figure 11 .)

**Figure 11. "Script Output" Control Deployment**



Arranged script "output" in selected open the Properties control, "all properties" tab in the "Data"> **"value"**

**Listing 3. Dojo.connect output script set in the logic part of the JavaScript example**

```
XSP.addOnLoad (function () {
    dojo.connect (dijit.byId ("# (id: calendar)"), "onValueSelected", function (date) {
        var formattedDate = dojo.date.locale.format (date, (formatLength: "medium", selector: "date"));
        document.getElementById ("# (id: selectedDate)"). value = formattedDate;
    ));
));
```

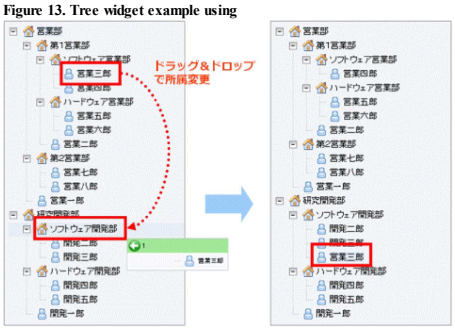
Calendar widget example above is complete. Completed using a Web browser to access XPage, please check the behavior.

---

Sample 2. Tree Using Widgets

Then applied as a Guide, Tree XPages widget shows how to use. The sample presented here, as in Figure 13, Tree using a tree widget to display Domino department employee information stored in the application, drag and the ability to change further in the department drop implements.

**Figure 13. Tree widget example using**




Tree data visualization and other Dojo widget widgets, data retrieval and update often done in AJAX, REST is common for it to provide services on the server side. So in this example, the information services department employees get a tree, and drag and create a drop-time update service department, Tree AJAX widgets to these services to access and use.

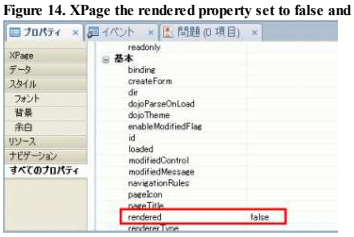
---

Example 2-1. Tree widget to display JSON data returned by the service to create XPage

First, the information department employees a tree, Tree widget that can display the JSON-formatted data service is created as a return. Basically, the Web is accessed from XPage browser to display HTML pages on the return data. But by devising an implementation, HTML and XML to JSON return data that can be returned. Using this method, XPage try to create a service in return JSON data. The final version of the sample application will ListEmployees.xsp.

First, "XPage" Control of the Properties "all properties" tab in the "base"> **"rendered"** property to false (Fig. 14). This setting allows for HTML will not display any output data. White screen is displayed by the browser preview and to do in this situation, HTML can understand that is not returned and no data source view.

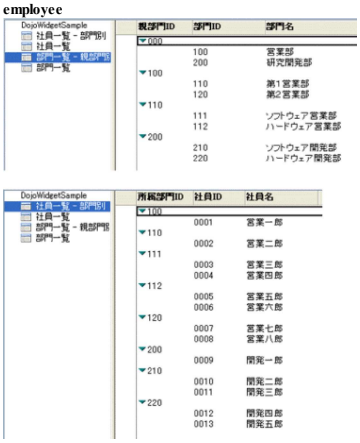
**Figure 14. XPage the rendered property set to false and return data to HTML**




Then return to JSON implementation of the data portion. "XPage" Open the Events view of the control in the event list **"page"**> "afterRenderResponse" In the event the server-side JavaScript in Listing 4 to describe (Fig. 15). XPages the JSF (Java Server Faces) and use that based on the technology, JSF to write the response data have a ResponseWriter to get the object, JSON and export directly as a string data .

**Figure 15. XPage the event afterRenderResponse**

**Figure 16. List view List view and department employees used to generate information department employee**




**Listing 5. JSON data writer to write server-side JavaScript example**

```
/ / Function to export the data tree of all employees function writeEmployeeTree (writer) (
    try (
        writer.append ('(identifier: "id", label: "name", items :[');
        writeEmployeeTreeInDept (writer, "000"); / / top-level category where '000 '
        writer.append ("]}");
    ) Catch (e) (
        print (e);
    )
)

/ / Write function of tree data sector employees under a given function writeEmployeeTreeInDept (writer, deptId) (
    / / Child of the employee's department and a member designated by the department get a Collection view var deptCol = database.get
    var empCol = database.getView ("EmpsByDept"). getAllEntriesByKey (deptId);

    / / Write the data sectors son var deptEntry = deptCol.getFirstEntry ();
    while (deptEntry! = null) (
        var deptDoc = deptEntry.getDocument ();
        var childDeptId = deptDoc.getItemValueString ("deptID");
        var childDeptName = deptDoc.getItemValueString ("deptName");
        writer.append ('(id: "' + childDeptId + '", name: "' + childDeptName + '", type: "dept",');

        / / Write elements of recursive division of child and children writer.append ("children :[");
        writeEmployeeTreeInDept (writer, childDeptId); / / recursive call writer.append ("]}");

        deptEntry = deptCol.getNextEntry (deptEntry);
        if (deptEntry! = null | | empCol.getCount ()> 0) writer.append (",");
    )
    / / Export data belonging employee if (empCol.getCount ()> 0) (
        var empEntry = empCol.getFirstEntry ();
        while (empEntry! = null) (
            var empDoc = empEntry.getDocument ();
            var empID = empDoc.getItemValueString ("employeeID");
            var empName = empDoc.getItemValueString ("employeeName");
            writer.append ('(id: "' + empID + '", name: "' + empName + '", type: "employee "}');

            empEntry = empCol.getNextEntry (empEntry);
            if (empEntry! = null) writer.append (",");
        )
    )
)
```

Implementation of the above service is complete. ListEmployees.xsp to access, as shown in Listing 6 is returned JSON data. (Line breaks and indentation 06 in the list were added manually later for viewing the data, the actual sample data is returned without line breaks or indentation.)

**Listing 6. ListEmployees.xsp actually return JSON data**

```
(
    identifier: "id", label: "name", items: [
        (Id: "100", name: "Sales", type: "dept", children: [
            (Id: "110", name: "First Sales", type: "dept", children: [
                (Id: "111", name: "Software Sales", type: "dept", children: [
                    (Id: "0003", name: "Sales Saburo", type: "employee"),
                    (Id: "0004", name: "Sales Shiro", type: "employee")
                ]),
                (Id: "112", name: "Hardware Sales", type: "dept", children: [
                    (Id: "0005", name: "Sales Goro", type: "employee"),
```

widget here please for more information.

**Listing 7. 2-1 Sample Data API to access services through a created in the Tree widget's HTML source example (abbreviated)**

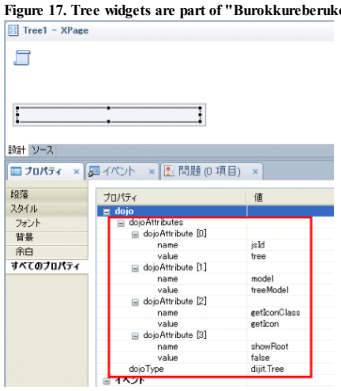
```
<script type="text/javascript">
dojo.require ("dojo.data.ItemFileWriteStore");
dojo.require ("dijit.tree.ForestStoreModel");
dojo.require ("dijit.Tree");

function getIcon (item, opened) (
    try (
        return (! item)? "deptIcon": (treeStore.hasAttribute (item, "type")
                & & "Employee" == treeStore.getValue (item, "type"))? "EmpIcon": "deptIcon";
    ) Catch (e) (
        return "deptIcon";
    )
);
</ Script>

<body class="tundra">
    <div dojoType="dojo.data.ItemFileWriteStore" jsId="treeStore" url="ListEmployees.xsp"/>
    <Div dojoType = "dijit.tree.ForestStoreModel" jsId = "treeModel" store = "treeStore"
            rootId = "root" childrenAttrs = "children" />
    <div class="treeOuter">
        <Div dojoType = "dijit.Tree" jsId = "tree" model = "treeModel"
                getIconClass = "getIcon" showRoot = "false" />
    </ Div>
</ Body>
```

7, the contents of this list XPage how to implement the sample 1 is similar to the case of the Calendar widget. dojo.require part "base"> "resources"> "dojoModule" to the other, JavaScript script "output" controls, dojoType and Dojo widget attributes part of "dojo"> "dojoType" and "dojoAttributes" to Define each. For example, in Listing 7 is the body of the widget Tree  dijit.Tree specified portion of the element, XPage on "Burokkureberukontentsu" shown in Figure 17 can be implemented using the control.

**Figure 17. Tree widgets are part of "Burokkureberukontentsu" setting on the control XPage**




Tree widgets are more visible part of the implementation. Tree1.xsp to preview, you should see a tree in the information department employee, as shown in Figure 13. We can not drag and drop yet.

---

Example 2-3. XPage data modification services to create a

Then, Tree on the widget drag & drop operation during use, the update service department employees XPage Create. The final version of the sample application will UpdateEmployee.xsp.

Implementation of the update service is introduced in the sample 2-1 is similar to the implementation of JSON data retrieval service. The difference is, afterRenderResponse server-side JavaScript is the only event in the implementation. Here, as in Listing 8, a new department employee ID and the ID parameter empId = 0001 & deptId = 100 POST when it has been to implement the ability to update the department employees. Example 2-1 HTTP access JSON data retrieval service is Response data was to generate a return service, this update service, HTTP Request access to services and are receiving treatment.

**Listing 8. XPage afterRenderResponse of server-side data update service that implements the JavaScript event example**

```
var extContext = facesContext.getExternalContext ();

/ / Request Get var request = extContext.getRequest ();
```

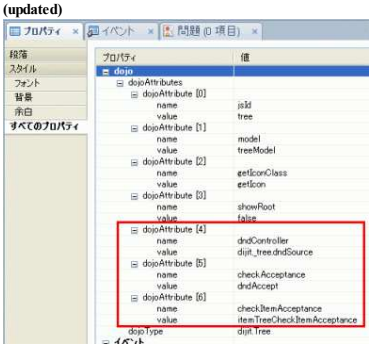
```
);

function itemTreeCheckItemAcceptance (node, source) (
    var type = "";
    var widget = dijit.getEnclosingWidget (node);
    if (widget & & widget.item) (
        type = treeStore.getValue (widget.item, "type");
    )
    return (type == "dept")? true: false;
);
</ Script>

<body class="tundra">
    <div dojoType="dojo.data.ItemFileWriteStore" jsId="treeStore" url="ListEmployees.xsp"/>
    <Div dojoType = "dijit.tree.ForestStoreModel" jsId = "treeModel" store = "treeStore"
            rootId = "root" childrenAttrs = "children" />
    <div class="treeOuter">
        <Div dojoType = "dijit.Tree" jsId = "tree" model = "treeModel"
                getIconClass = "getIcon" showRoot = "false"
                dndController = "dijit._tree.dndSource" checkAcceptance = "dndAccept"
                 checkItemAcceptance = "itemTreeCheckItemAcceptance" />
    </ Div>
</ Body>
```

09 Sample 1 and sample the contents of this list as well as when the 2-2, XPage can be converted into implementation. For example, dijit.Tree  update part of the specified element, XPage on "Burokkureberukontentsu" is updated to control Figure 18.

**Figure 18. Tree widgets are part of "Burokkureberukontentsu" XPage control settings on the (updated)**




Now, Tree node and drag on the company can now move widgets and drop. Finally, when dropped, 2-3 sample data update service should be changed to implement AJAX call. In order to perform during the drop, as in Listing 10 dndController Tree widget with the object of the dojo.connect function onDndDrop in dojo.xhrPost in this data update service and to call the function using Masu.

**Listing 10. DndController of onDndDrop dojo.connect function to invoke the service when data updates have JavaScript drop case**

```
XSP.addOnLoad (function () (
    dojo.connect (tree.dndController, "onDndDrop", function (source, nodes, copy) (
        var empId = null, deptId = null;
        / / Drop the Employee from the Source Node ID Employee Gets var sourceWidget = dijit.getEnclosingWidget (nodes [0]);
        if (sourceWidget & & sourceWidget.item) (
            empId = treeStore.getValue (sourceWidget.item, "id");
        )
        / / Drop the node ID from the Target department Dept obtain var targetWidget = dijit.getEnclosingWidget (this.current);
        if (targetWidget & & targetWidget.item) (
            deptId = treeStore.getValue (targetWidget.item, "id");
        )
        / / POST request to change
        if (empId & & deptId) (
            dojo.xhrPost ((
                url: "UpdateEmployee.xsp",
                postData: "empId =" + empId + "& deptId =" + deptId,
                load: function () (),
                error: function () ()
            ));
        )
    ));
));
```

Or drag & drop implementation of data update is complete. As shown in Figure 13, drag it to another department employees and to move and drop functionality implemented Domino because the information is updated in the application are displayed correctly updated data can then refresh the browser Masu.

- Actual XPages Application Development (developerWorks article)
- Xpages Straight Up (English) (developerWorks article)
- IBM Lotus Notes / Domino 8.5.1 New features in XPages (developerWorks article)
- Lotus Notes and Domino Application Development Wiki (English)

**Dojo Toolkit**

- The Dojo Toolkit : Dojo Toolkit website (English)
- Dojocampus : Dojo Official Documentation (English)
- IBM developerWorks: Wikis - Software Technology Japan - Dojo Toolkit
- Object-oriented start with Dojo Toolkit
- Working with the DataGrid Dojo Toolkit

About the Author

Ono Makoto Journal, the Software Development Laboratory, IBM Japan belong to the development of IM & Lotus, IBM Lotus Notes / Domino, IBM WebSphere Portal, IBM Lotus Forms, IBM Lotus Mashups WPLC and work to develop various products for applications ISV / BP for technical assistance we are like. This time of year is suffering from hay fever every day.