Simple Carpool Application using SAP NetWeaver Portal, KM, XML Forms, and Google Maps

Applies to:

SAP NetWeaver Portal 6.x\7.x, Knowledge Management (KM), and Google Maps. For more information, visit the Portal and Collaboration homepage.

Summary

The scope of this paper is to show how to creatively use Portal Knowledge Management (KM) along with XML forms to build small carpool application.

Author: Harman Shahi

Created on: 01 April 2009

Author Bio

Harman Shahi is a SAP Development Consultant. His expertise includes working on enterprise-level projects based on SAP Enterprise Portals, NetWeaver, Web DynPro, Java\J2EE, Adobe Document Services, RFC, Web Page Composer (WPC), and SAP CRM. h_shahi@yahoo.com

Table of Contents

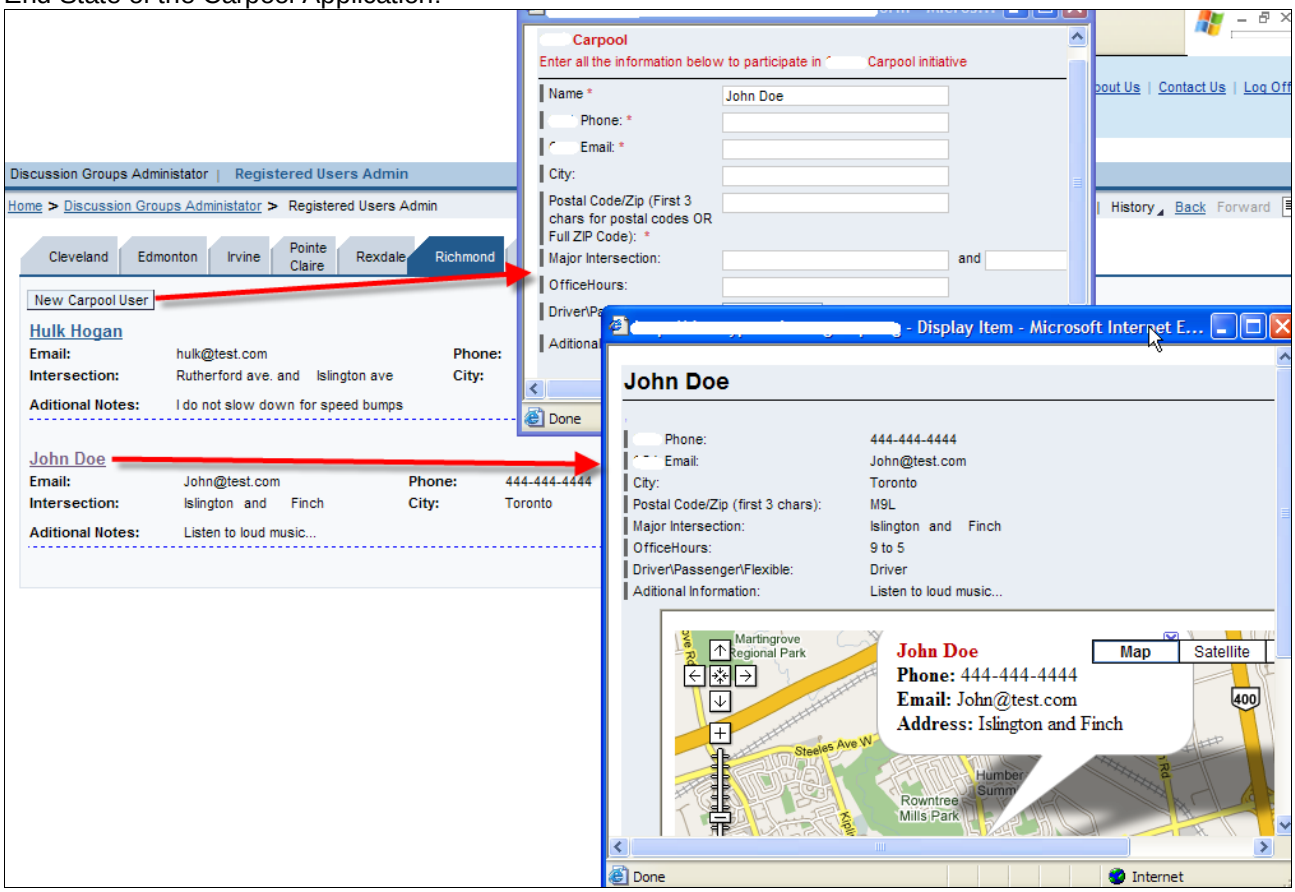
Using KM and XML forms to build a small carpool application

The scope of this document is to show how SAP Portal's Knowledge Management (KM) along with XML forms can be creatively used to build a small carpool application. **The tools and techniques used are out of the box, and can be used to fulfill other similar business requirements.** This paper also shows how to integrate Google Maps into your XML forms.

In Order to achieve this we will:

1. Create XML form to capture user data.
2. Setup of Layout set to display data in tabular manner. Also allow users to add themselves to specific category.
3. Creating a simple HTML file and portal iView to display user information along with their location on Google map.
4. Test

End State of the Carpool Application:



Prerequisites

You must have Content Admin and Content Management role in the portal.

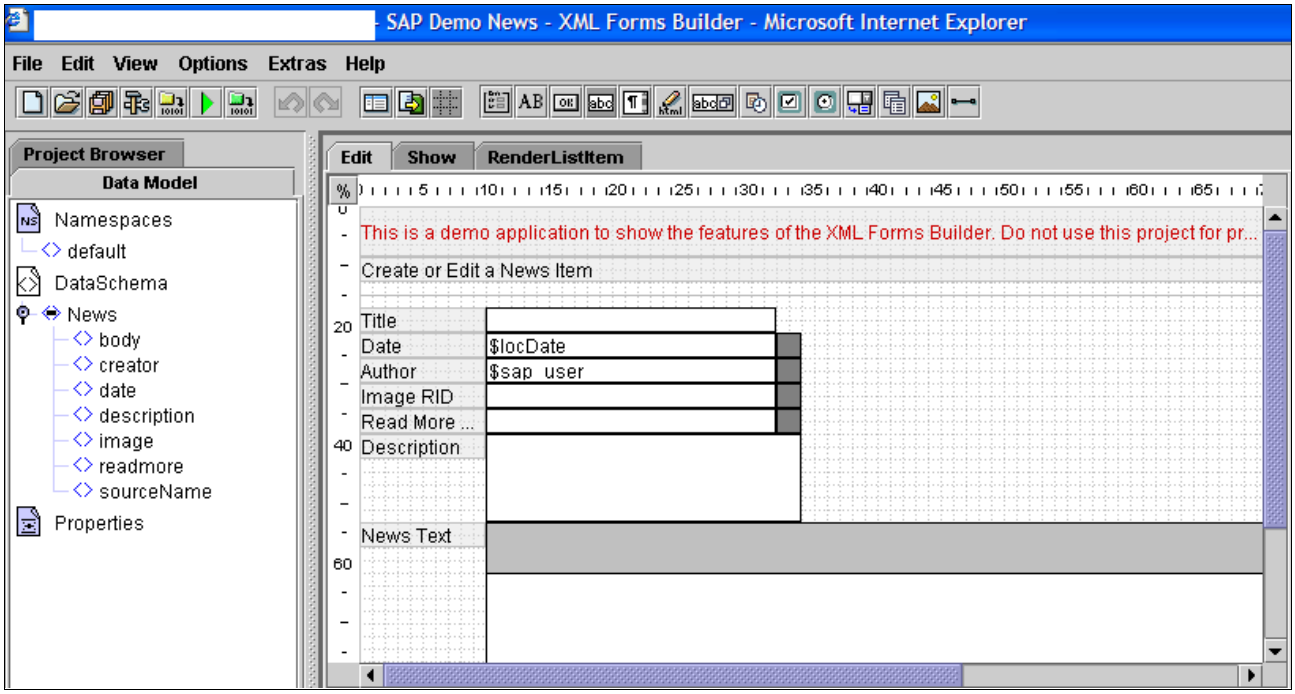
XML Form

Create XML Form for User Input

- **Login** to the portal as a **portal administrator**.
- Go to **Content Management > Form Builder**. This should open the **XML Form Builders** tool in a new browser window. You may have to install a small java plug-in for the first time.

To cut down the XML form build time, we will make a copy of an existing SAP XML form, and work from there: To do this:

- Open the form **XML Demo News**
- Go to **File > Save As**, and give it a name of your choice. In our example, the form name is **Carpool.**

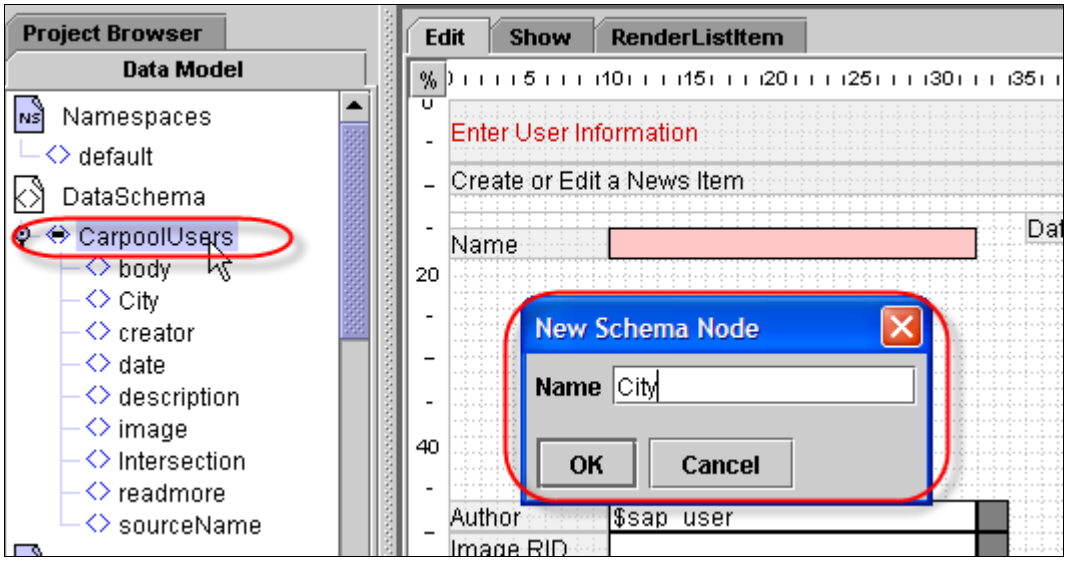


Editing XML Form for User Input

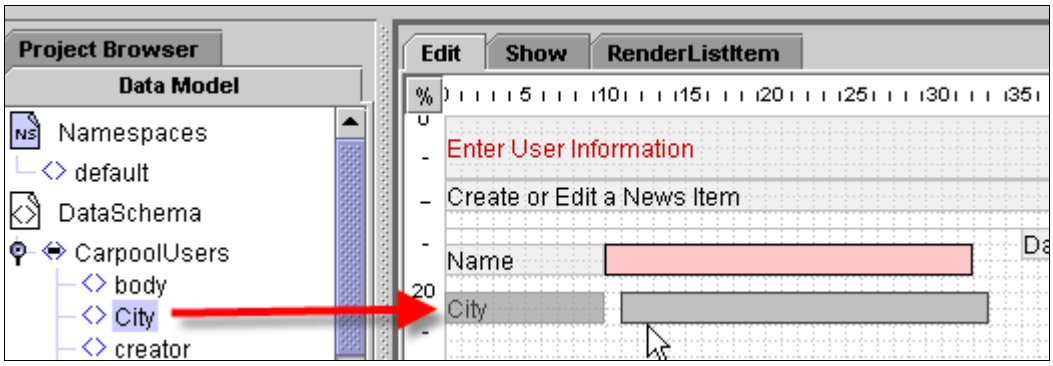
Now we will use our knowledge of XML forms to edit this form to your specific needs.

Next, we will run through a complete process of **adding <u>one</u> input control onto the XML form**, and then you can use this same process to add your necessary controls.

- **Add** a new Schema Node by clicking on your **Schema** (i.e. CarpoolUsers), and select **New Child Node**.
- Give it the name **City,** and click **OK.**



- **Drag** the attribute **City** from **DataSchema** node onto the **form**:



Now using the same drag and drop process, you can add the City node to your **Show** and **RenderList** Item areas of the form.

If you are not familiar with the different purposes of the **3 tabs (Edit, Show, and RenderListItem)**, then please do some reading on the XML form builder tool. I will only try to explain them in one paragraph.

Edit form:

You use this form to create or change individual documents (for example, single news items). Typically, this form contains input fields in which you can create content (for example, the title or text for a news item).

RenderListItem form:

You use this form to define the layout of a **list entry** in a folder in the flexible user interface. Typically, you display some of the elements of a document, such as the title and summary of a news item, on this form. You call up the Show form if you want to display the entire document.

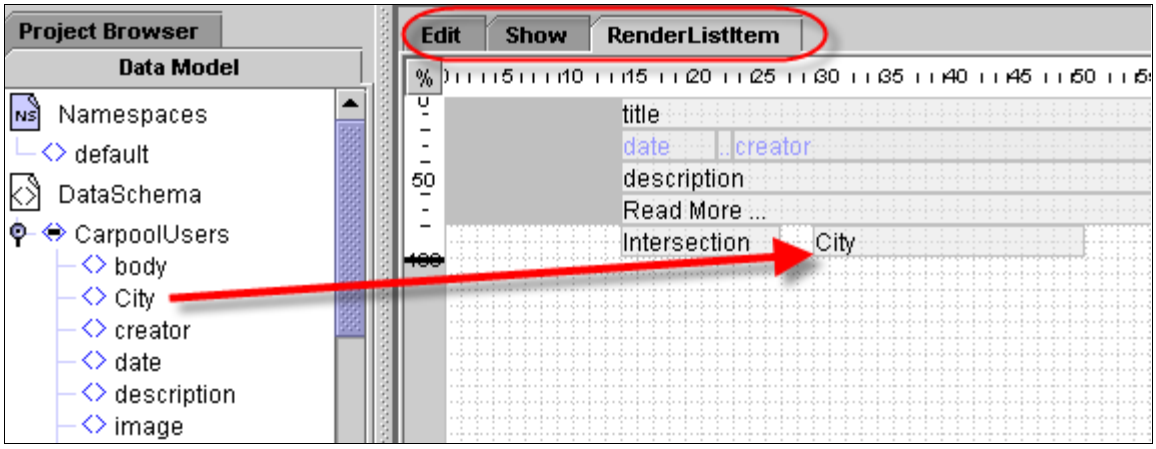
Show form:

You use this form to display a single document. Typically, the Show form displays all elements of a document (for example, the title, author, and all the text contained in a news item). Do not use entry controls on the Show form.

Link to SAP Documentation explaining XML forms:

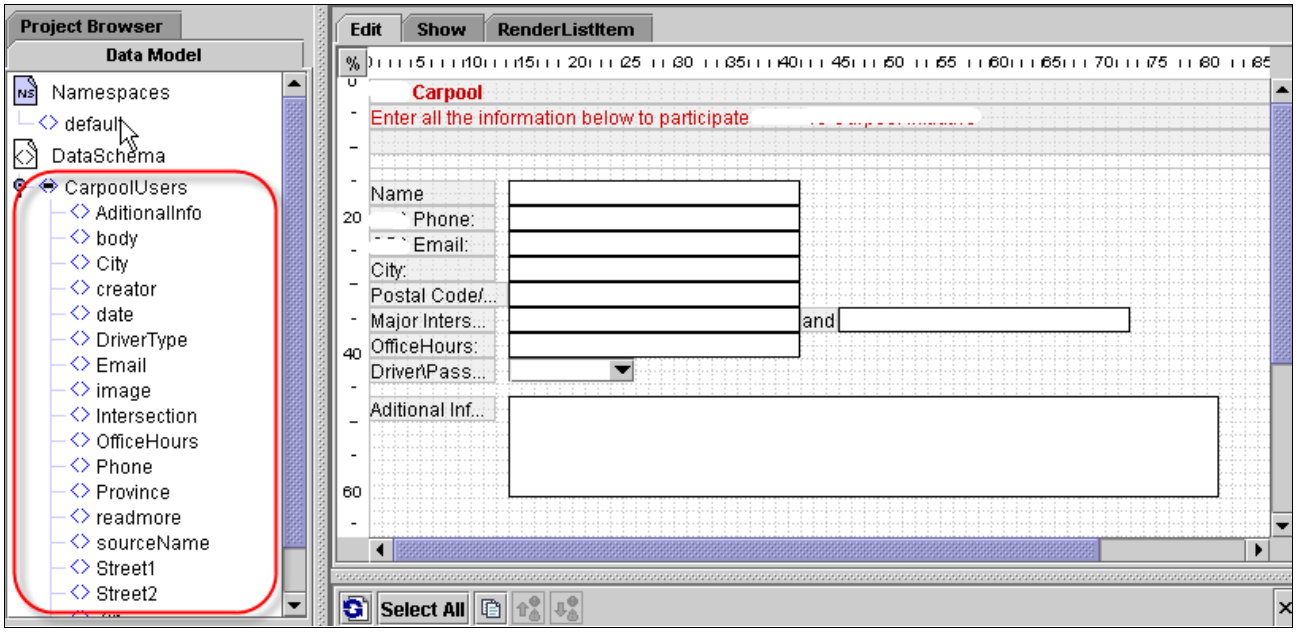
http://help.sap.com/saphelp_nw70/helpdata/EN/fd/bd016254946048a4d7a17c2aad7600/frameset.htm

- Below you can see that I **added** the **City** attribute to the RenderListItem form; this means that I want to display user's city attribute when all the forms are shown in a **list** format.

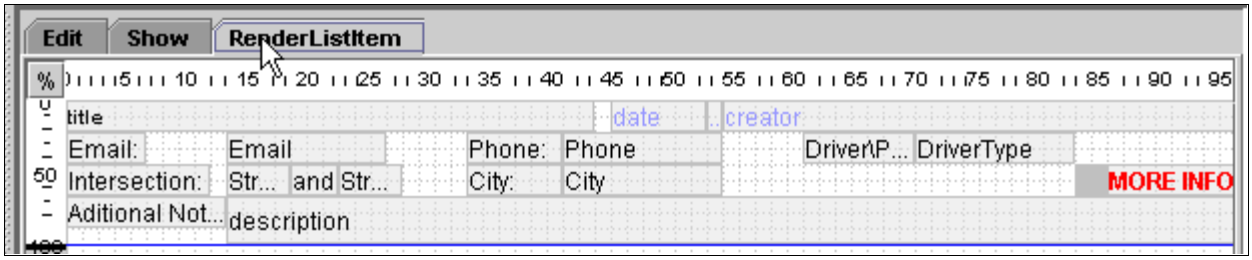


- Below are some screenshot displaying the **completed forms,** and the final view of the **DataSchema.**

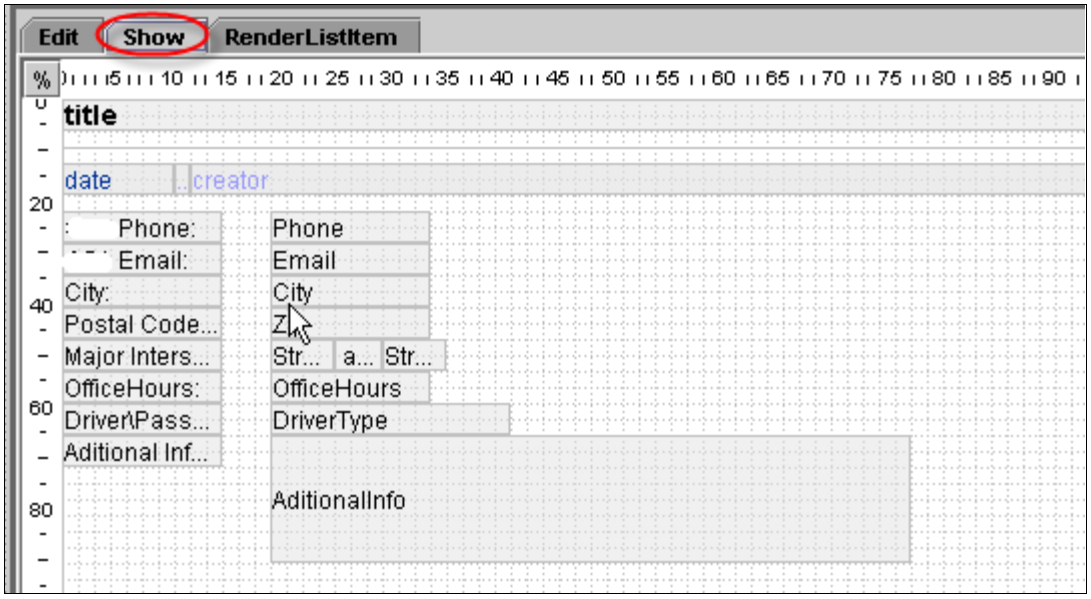
Edit Form:



RenderListItem Form:



Show Form:



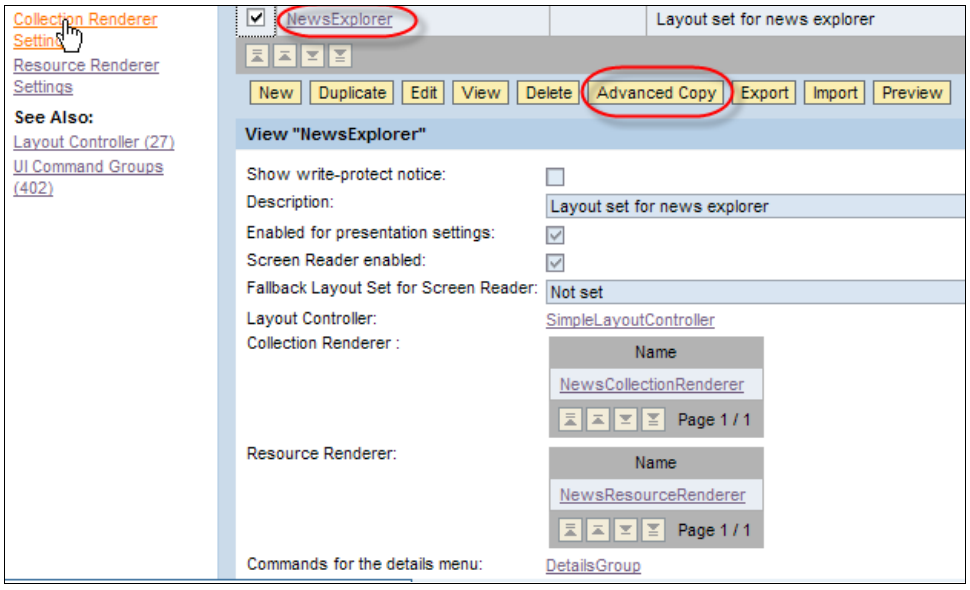
- **Save** your XML form.
- **Preview** your XML form.
- **Generate** your XML form.

Layout Sets:

In order to categorize the users in different offices (cities) we decided to create a New Tab Layout set.

Create New Layout Set:

- Go to **System Administrator > System Configuration > Knowledge Management > Content Management > User Interface > Settings > Layout Set**
- Search for "**NewsExplorer**" Layout Set.
- Create **Advanced Copy** of the "**NewsExplorer**" Layout set.

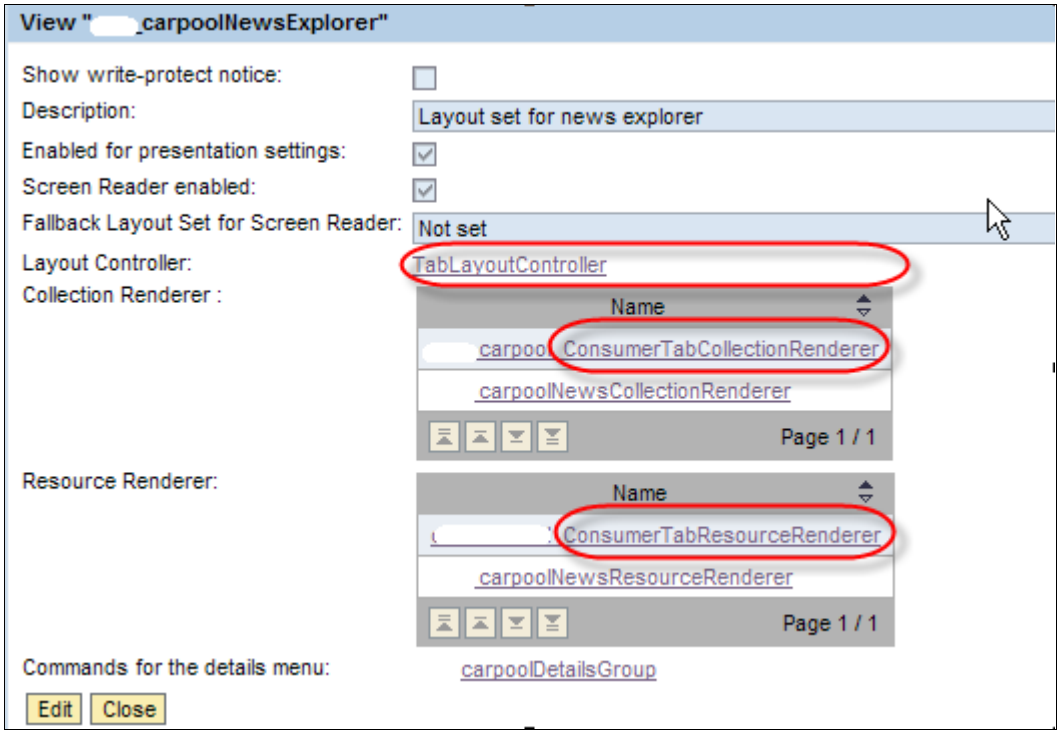


- Open the Newly copied Layout Set, and add the following:
  - Add 'ConsumerTabCollectionRenderer" to Collection Renderer

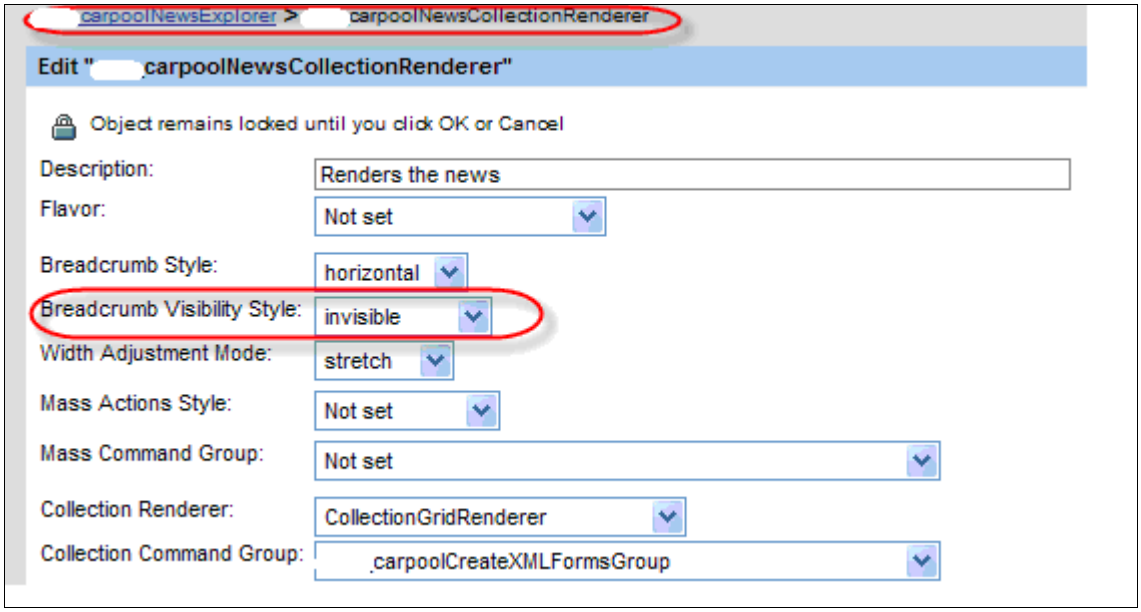
- Add “ConsumerTabResourceRenderer” to Resource Renderer

- Change the **Layout Controller** to **TabLayoutController**

Note: If you decide to make further changes to the look and feel of your application, then it is recommended that you make copies of the ConsumerTabCollectionRenderer and ConsumerTabResourceRenderer, and add the copies to your layout set. This gives you flexibility to edit without worrying about messing up out of the box content.



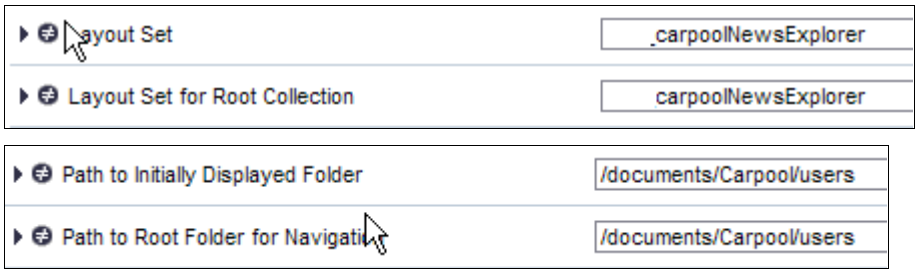
- Make the following setting to take **breadcrumbs** out from inside the tabs (This makes the breadcrumbs disappear, but the “New User” button stays)



- **Save** the LayoutSet.

Creating Portal iView

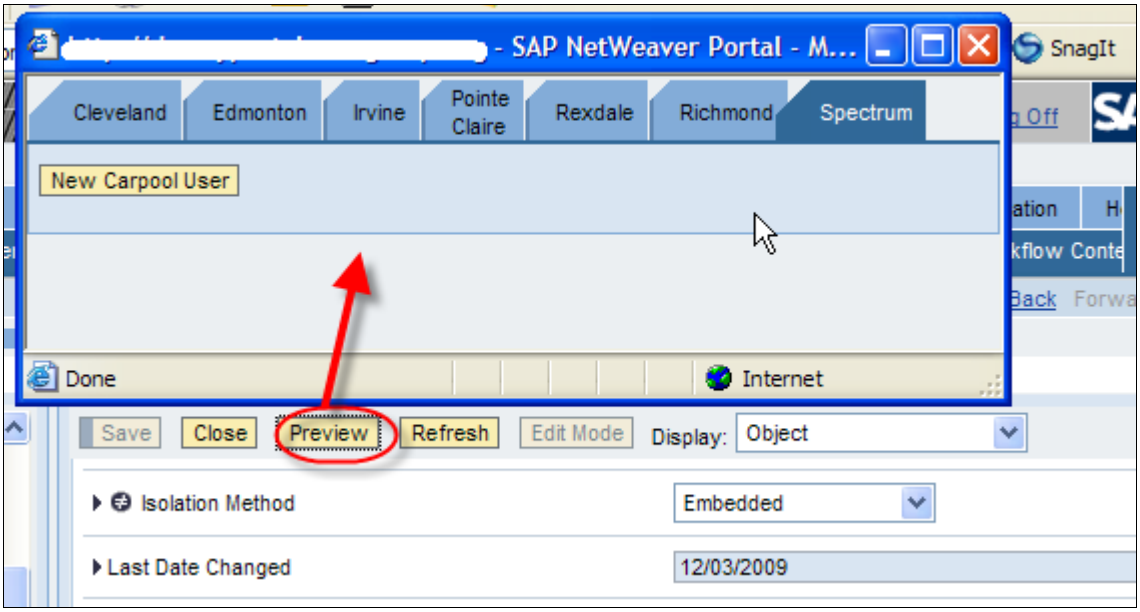
- You can **create your new iView** one of the following ways:
  1. Make a **copy** of the SAP's own News iView, and change the ID and Name, **OR**
  2. Create **new iView**, and select **KM News iView template**.
- After creating the iView, Set the property **Layout Set** to "<Name of your newly created layout set>.



Note: The Path to Initial Displayed Folder is the KM Location where you want to store all XML forms. The way our Layout Set is setup, it renders each folder as a Tab

- **Preview** the iView to make sure your layout set is in effect. (see below)

Note: The tabs that you see (Cleveland, Edmonton, Irvine…) are actually folders within the /documents/Carpool/users location.



Now you guide the users to add them in the appropriate Tab (area). Clicking on the "New Carpool User" button should open up your XML form, and after the user clicks **Save**, the record should be created within the respected Tab.

Adding Google Map in the Show Form (Optional)

Since this application deals with looking up intersections, we thought it would be useful to pull up user's location on Google Map. Unfortunately, **this is not a built in feature in XML forms** for now… So we manually updated the Show Form to add a little iFrame to display the Google Map.

Drawback: The biggest drawback of this approach is that your code will vanish once the project is generated. So this will have to be documented well (how to not overwrite the Show Form when publishing the XML Form). **Again, this step is optional, and can (and should) be skipped if Map is not absolutely necessary.**

Adding Google MAP:

- Sign up for a Google Map Key for you website (http://code.google.com/apis/maps/signup.html)
- Create an HTML File in KM (see **Appendix A** for the HTML file Code, the name of our html file is user_info_and_map.htm)
- Navigate to the KM folder **"/etc/xmlforms/<your form id folder>"**. Under this folder download the file **"<your form id>Show.xsl"** to the local machine. Open the downloaded file and find the section "**MAIN ITEM TEMPLATE**". Using your HTML knowledge place the following code

The code below places an iFrame object in the Show form. The Show form passes the following parameters to the iFrame object: Zip, dname, Email, Phone, username. The **iFrame is displaying** the HTML file (user_info_and_map.htm). In the HTML file we catch all these parameters and Display the Google map.

Code to display Google Map in an iFrame:

```
<!-- START: CUSTOM CODE TO DISPLAY GOOGLE MAP -->
  <!-- START: CUSTOM CODE TO DISPLAY GOOGLE MAP -->
<tr><td colspan="8">
<center>
 <IFRAME border="0" width="565px" height="365px" align="center">
  <xsl:attribute name="alt" />
  <xsl:attribute name="src">
<KM path of your html file location>?Street1=<xsl:value-of select="Street1" />&amp;amp;Street2=<xsl:value-of select="Street2" />&amp;amp;Zip=<xsl:value-of select="Zip"/>&amp;amp;dname=<xsl:value-of select="creator"/>&amp;amp;Email=<xsl:value-of select="Email"/>&amp;amp;Phone=<xsl:value-of select="Phone"/>&amp;amp;userName=<xsl:call-template name="printPropValues">
                    <xsl:with-param select="'default:displayname'" name="propId"/>
                    <xsl:with-param select="'no'" name="disableOutputEscaping"/>
                    <xsl:with-param select="'true'" name="useDisplayName"/>
                  </xsl:call-template>
</xsl:attribute>
</IFRAME>
</center>
</td></tr>
<!-- END: CUSTOM CODE TO DISPLAY GOOGLE MAP -->
```

The code embedded in Show Form. Next image displays the Google Map code embedded within the XML file.

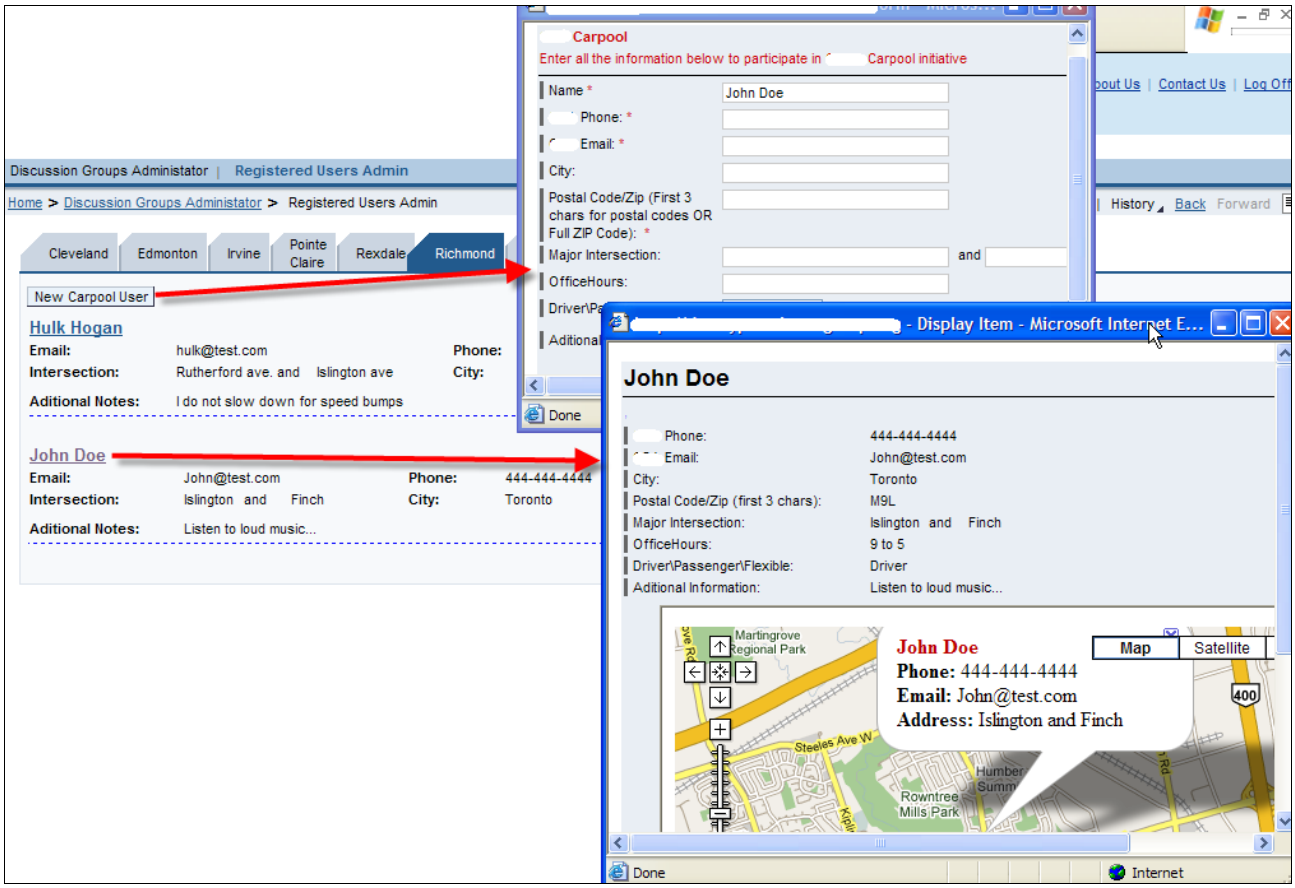
```
        </tr>


  <!-- START: CSA CUSTOM CODE TO DISPLAY GOOGLE MAP -->
<tr><td colspan="8">
<center>
 <IFRAME border="0" width="565px" height="365px" align="center">
  <xsl:attribute name="alt" />
  <xsl:attribute name="src">
/irj/go/km/docs/csag/car_pool/google_map/user_info_and_map.htm?Streetl=<xsl:value-of select="Streetl" />&amp;Street2
                          <xsl:with-param select="'default:displayname'" name="propId"/>
                          <xsl:with-param select="'no'" name="disableOutputEscaping"/>
                          <xsl:with-param select="'true'" name="useDisplayName"/>
                        </xsl:call-template>
</xsl:attribute>
</IFRAME>
</center>
</td></tr>
<!-- END: CSA CUSTOM CODE TO DISPLAY GOOGLE MAP -->


      </table>
    </xsl:template>
    <!--
         MAIN ITEM TEMPLATE
    -->
```

Testing the application

- **Preview** the Carpool Application iView.
- Click **New Carpool User** to add a user
- After the user is added, **click on the username** to see full user details including user's You should see a something like the following:



Appendix A – Code for Google Maps HTML file

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <title>Google Maps</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=<ENTER YOUR
GOOGLE MAPS KEY HERE>" type="text/javascript"></script>
  </head>
  <body onunload="GUnload()">
    <div id="map" style="width: 540px; height: 330px"></div>
    <noscript><b>JavaScript must be enabled in order for you to use Google Maps.</b>
      However, it seems JavaScript is either disabled or not supported by your
browser.
      To view Google Maps, enable JavaScript by changing your browser options, and
then
      try again.
    </noscript>
<script language="javascript" type="text/javascript">
function gup( name )
{
  name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
  var regexS = "[\\?&]"+name+"=([^&#]*)";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
    return "";
  else
    return results[1];
}
    function showAddress(address, winInfo, Zip) {
      if (geocoder) {
        geocoder.getLatLng(
          address,
          function(point) {
            if (!point) {
              //alert("Exact address could not be located, so displaying the
Zip\\\Postal Code area.");
                  showZip(Zip);
            } else {
              map.setCenter(point, 13);
              var marker = new GMarker(point);
              map.addOverlay(marker);
              marker.openInfoWindowHtml(winInfo);
            }
          }
        );
      }
    }
// New Function to display the Zip code area if exact address cannot be located.
    function showZip(Zip) {
   winInfo = '<font color=#bb0000><b>ZIP\\\Postal Code: ' +Zip+ '</b></font><br>';
      if (geocoder) {
```

```
        geocoder.getLatLng(
          Zip,
          function(point) {
            if (!point) {
              alert(Zip + " not found");
            } else {
              map.setCenter(point, 13);
              var marker = new GMarker(point);
              map.addOverlay(marker);
              marker.openInfoWindowHtml(winInfo);
            }
          }
        );
      }
    }
</script>
    <script type="text/javascript">
    //<![CDATA[
var Street1 = gup( 'Street1' );
Street1=Street1.replace(/%20/g,'  ');
//alert(Street1);
var Street2 = gup( 'Street2' );
Street2=Street2.replace(/%20/g,'  ');
//alert('Street2: ' + Street2);
var Zip = gup( 'Zip' );
Zip=Zip.replace(/%20/g,'  ');
//alert('ZIP: ' + Zip);
var dname = gup( 'dname' );
//alert('dname: ' + dname);
var Email = gup( 'Email' );
//alert('Email: ' + Email);
var Phone = gup( 'Phone' );
Phone=Phone.replace(/%20/g,'  ');
//alert('Phone: ' + Phone);
var userName = gup( 'userName' );
userName=userName.replace(/%20/g,'  ');
//alert('userName: ' + userName);
var geocoder = null;
    if (GBrowserIsCompatible()) {
      // A function to create the marker and set up the event window
      // Dont try to unroll this function. It has to be here for the function closure
      // Each instance of the function preserves the contends of a different instance
      // of the "marker" and "html" variables which will be needed later when the
event triggers.
      function createMarker(point,html) {
        var marker = new GMarker(point);
        GEvent.addListener(marker, "click", function() {
          marker.openInfoWindowHtml(html);
        });
        return marker;
      }
      // Display the map, with some controls and set the initial location
      var map = new GMap2(document.getElementById("map"));
      map.addControl(new GLargeMapControl());
      map.addControl(new GMapTypeControl());
    //  map.setCenter(new GLatLng(43.907787,-79.359741),8);
```

```
geocoder = new GClientGeocoder();
   // Set up three markers with info windows
     var addr = Street1 + ' and ' + Street2 + ', loc: ' + Zip;
   //  showAddress(addr);
   var winInfo = '<b><font color=#bb0000>' +userName+ '</font></b><br><b>Phone: </b>'
+ Phone + '<br><b>Email:</b> ' + Email + '<br><b>Address: </b>' + Street1 + ' and ' +
Street2 ;
   showAddress(addr, winInfo, Zip);
    }
    // display a warning if the browser was not compatible
    else {
      alert("Sorry, the Google Maps API is not compatible with this browser");
    }
    </script>
  </body>
</html>
```

Related Content

XML Form Builder

http://help.sap.com/saphelp_nw70/helpdata/en/8f/fe743c74fa6449e10000000a11402f/frameset.htm

Layout Sets

http://help.sap.com/saphelp_nw70/helpdata/en/30/504a1e7f0e354bbf9adedf1a29f3ec/content.htm

Google Maps

http://code.google.com/apis/maps/signup.html

For more information, visit the Portal and Collaboration homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.