Chameleon AJAX workshop

by William Bronsema, DM Solutions Group, Ottawa, Canada *and*
Bart van den Eijnden, OSGIS, Utrecht, The Netherlands

Prerequisites

- MS4W PHP4 Base Installer v1.5.5
- Chameleon 2.4 MS4W package

Installation

This workshop is available as an MS4W package at:

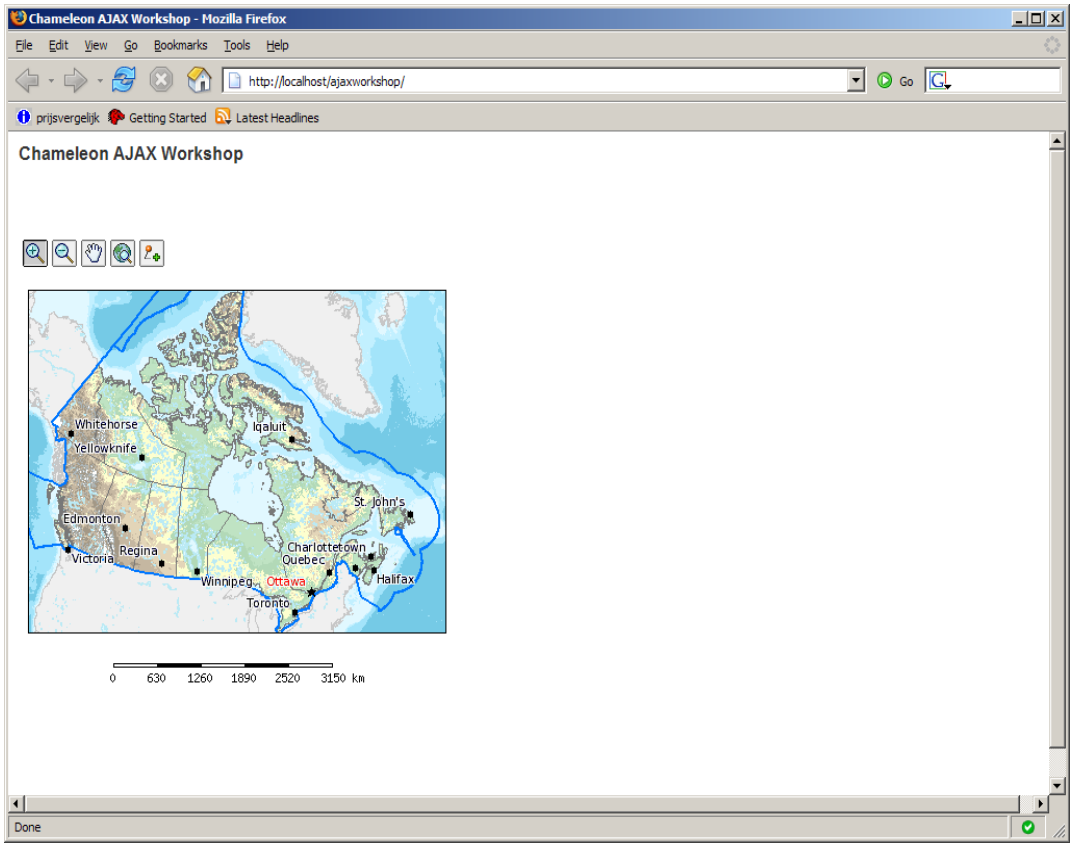
http://www.osgis.nl/lausanne/ajaxworkshop.zip

Install the workshop MS4W package. This requires an Apache restart.

Verify your installation using:

http://localhost/ajaxworkshop/

The result should be:



Purpose
The purpose of this workshop is to create a widget with the functionality for adding a point dynamically to the map, without submitting the page. It will be done using an AJAX call (although no XML is involved). We are going to use the MapNotes widget partly for this.

First we are going to verify that adding a map note using the MapNotes widget submits the form.

Exercise 1
The application currently uses the normal Chameleon mode, in which the form in the page is submitted all the time. How can we prove that the application is submitting the form all the time? Use javascript to add this proof to the html page, and then verify that adding a map note will submit the form.

Exercise 2
Now we are going to change the application to use the Chameleon JSAPI mode instead of the submit mode. Study the Chameleon samples directory on how to do this. Tip: you need to study the files *sample_enhanced_jsapi.phtml* and *sample_enhanced_jsapi.html*.
Verify using the solution from Exercise 1 that the page does not submit anymore using zoom in, zoom out, pan, and zoom to full extent. Note: the MapNotes widget has no JSAPI interface so it will still submit the page. Also verify this.

Exercise 3
We are going to do an experiment with the AJAX code which is part of Chameleon. The code is present in *chameleon\htdocs\common\ajax\xhr.js*.

The main function in the xhr script is call:

```
// u -> url
// o -> object (can be null) to invoke function on
// f -> callback function
// p -> optional argument to specify POST
function call(u,o,f)
{
```

An example could be:

```
call('/chameleon/widgets/MapTips/maptips.php?sid=[$gszSessId$]&minx=' +
goCWCJSAPI.oMap.minx +
    '&miny='+goCWCJSAPI.oMap.miny + '&maxx=' + goCWCJSAPI.oMap.maxx +
    '&maxy='+goCWCJSAPI.oMap.maxy, null, processAjax);
```

When the server returns with the answer, processAjax will process the result, an example could be:

```
function processAjax(szResults)
{
  if (document.all)
    document.all.maptips_imagemap.innerHTML= "'"+szResults+"'";
}
```

(a)

Add an include to xhr.js to your HTML template.

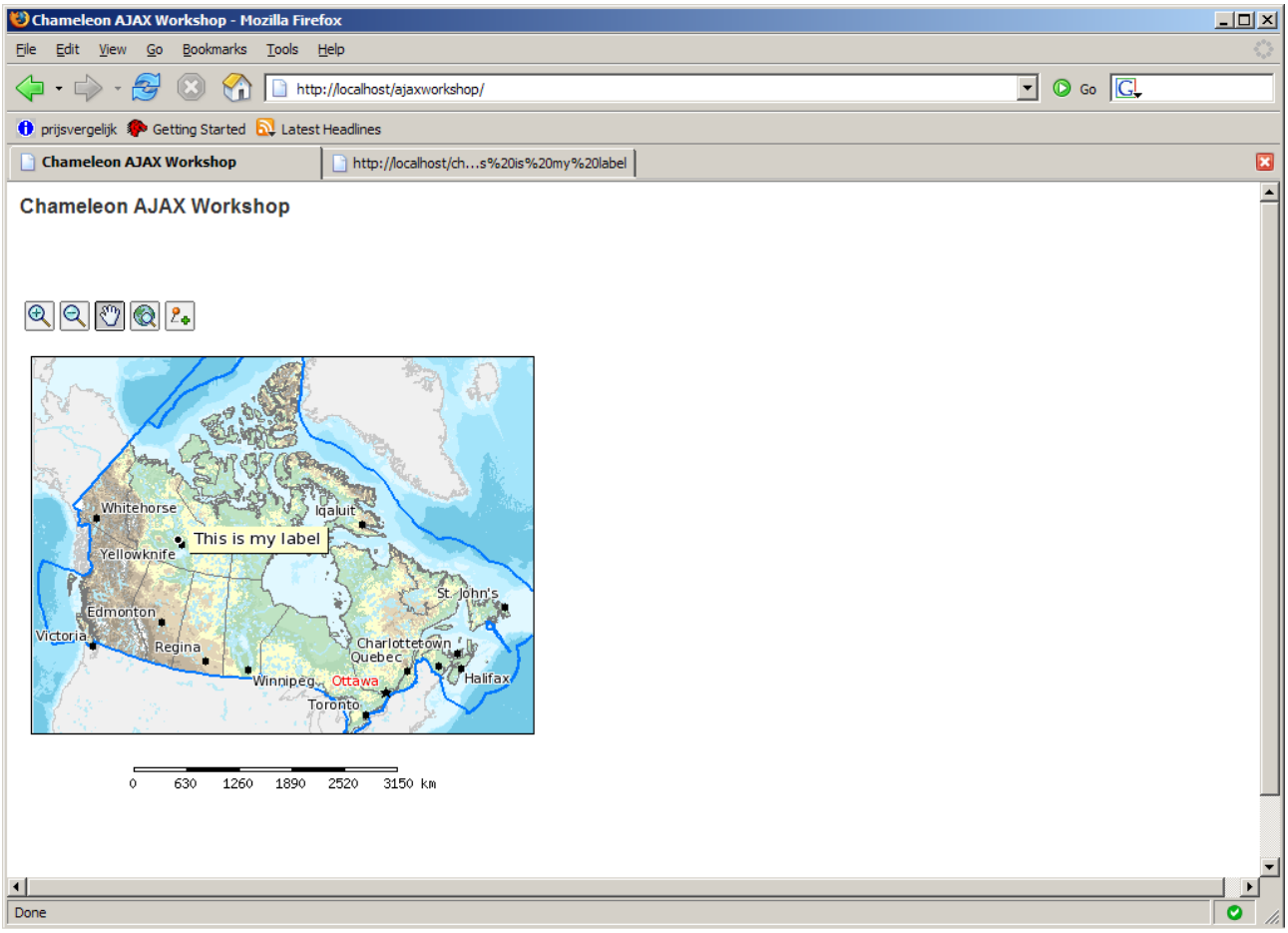
Creating a temporary shapefile in the user's session, and adding that to the session's MAP file is done mostly in properties.php. So we need to understand the interface of properties.php, and do the same thing using an AJAX call.

An example call to properties.php is:

http://localhost/chameleon/widgets/MapNotes/properties.php?sid=44eee8d2f0abf&layer=MapNotesAnnotation&font=sans&fontsize=10&fontcolor=0%200%200&fontoutlinecolor=255%20255%20255&symbol=circle&symbolsize=8&symbolcolor=0%200%200&symboloutlinecolor=255%20255%20255&marquee=1&offsetx=6&offsety=0&labelposition=CR&mode=ADD&mouse_x=105&mouse_y=142&update=1&label=This%20is%20my%20label

This adds a point to the map, at a position of X: 105 and Y: 142 (in pixels, 0,0 is top left part of the map). The label is “This is my label”.

Test this call in your application by changing the sid (session id). View the HTML source of your application, and look for the session id. Then change it in the URL above and execute the URL in your browser. Pan the map a little bit and see if the point appears:

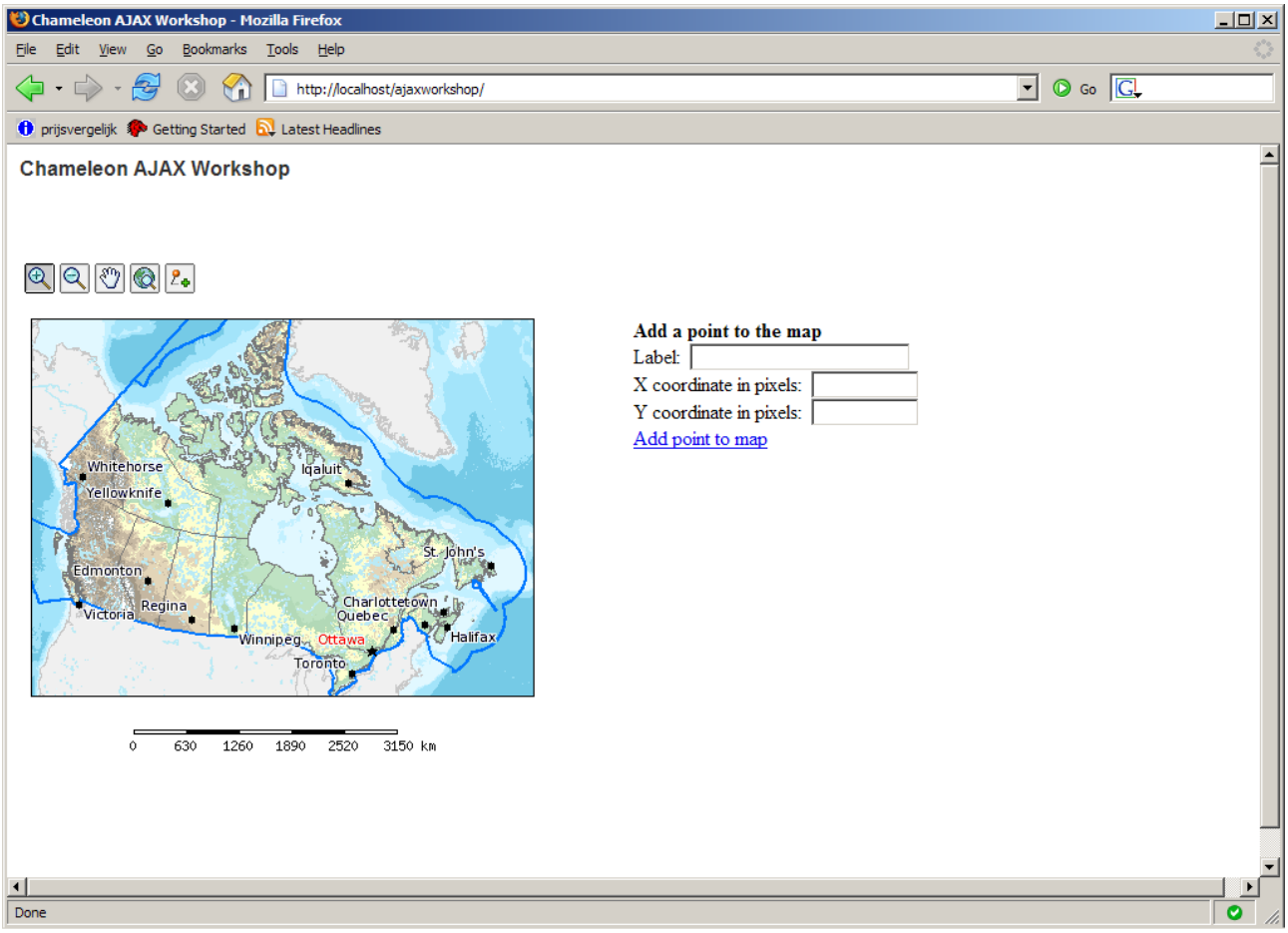


Now we should integrate this logic into the application using AJAX.

(b)

Add three input fields to your HTML template, one for the label, one for the X value (in pixels) and one for the Y value (in pixels). Also update your css so that this part shows right of the map.

Add a link which can call the AJAX function and add the point.
It should look something like:

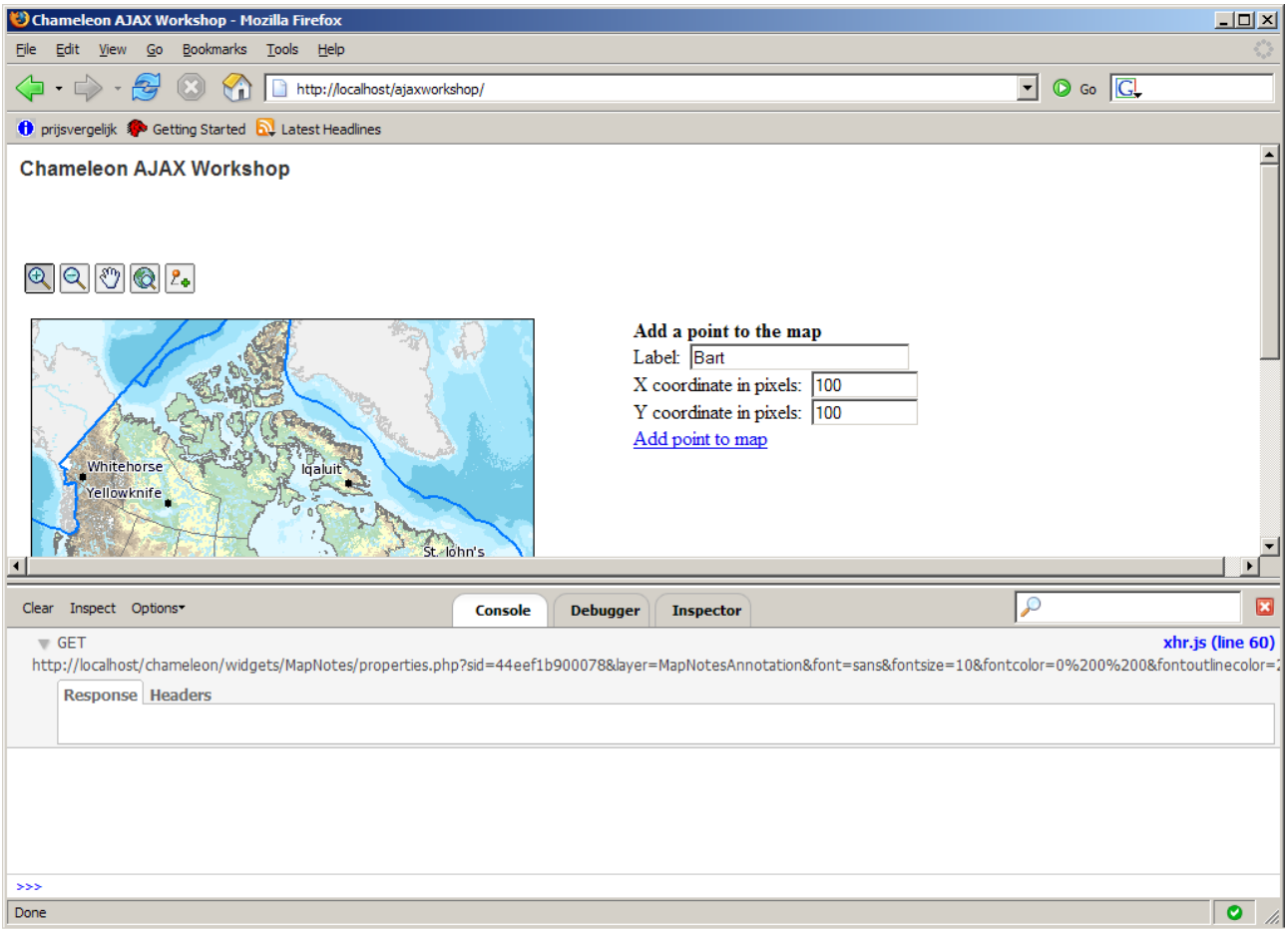


(c)

Now implement the actual javascript function which will be called when the link is clicked.

We need the session id to be available in the HTML template. Chameleon has a mechanism for this which involves changing the index.phtml file. After the CWCInitialize call you can call the function setVar to set the session id. In the HTML template you can use it by [$variable_name$].

Tip1, we can track AJAX calls in Firefox using the Firebug extension:



Tip 2: use the call function to call properties.php. In the callback function, make sure the map is refreshed using goCWCJSAPI.CallServer.

Tip 3: you need to make a little change to the properties.php script so that it gives back a small piece of xml, with an empty response it won't work (i.e. the callback function won't be called), like e.g.:

```
// manage layer
manageLayer( $oMapSession->oMap, $_FORM['layer'], $szFileName );

// save the map state
$_SESSION['gszCurrentState'] = $oMapSession->saveState();

echo '<?xml version="1.0" encoding="UTF-8"?><result>success</result>';
```

(d)
Now verify that the page does not submit by using Exercise 1.

(e)
Add the waitimage so that the user gets the right feedback.

Exercise 4

Now that everything is working, convert the code to a widget!

Tip, start with the foss4g widget template provided.

Answers

Exercise 1

There are several ways but one if them is to add a javascript function to the page which will present an alert. Then add the function to the onSubmit of the form.

Edit ajaxworkshop.html and add the javascript function:

```
    <script type="text/javascript">
    function myOnLoad()
    {
      CWC2OnLoadFunction()
    }
    function mySubmit(form)
    {
      alert('submitting form');
      return true;
    }
    </script>
```

Then at the end of the page, add a piece of javascript which will set the onsubmit handler:

```
        </table>
      </div>
      <script type="text/javascript">
        document.mainform.onSubmit = mySubmit(this);
      </script>
    </form>
  </body>
</html>
```

Start up the application again using a CTRL + “Reload current page” and verify that the alert shows up, it also shows up when we zoom in, add a map note etc.

Exercise 2

Edit index.phtml and set the minimum maturity level of the application the ALPHA, since JSAPI is still considered ALPHA.

```
$oApp->CWCInitialize( $szTemplate, $szMapFile  );
$oApp->mnMinimumMaturityLevel = MATURITY_ALPHA;
$oApp->CWCExecute();
```

Edit ajaxworkshop.html and add:

```
    <cwc2 type="SharedResource" name="WaitImage">
      <waitimage language="en-CA" waitimage="images/spinner.gif"
waitimagewidth="216" waitimageheight="50"/>
    </cwc2>
    <cwc2 type="cwcjsapi" debug="false"/>
    <cwc2 type="SharedResource" name="CWCJSAPI"/>

    <script type="text/javascript">
```

Exercise 3

(a)

Adding the xhr.js to the HTML template:

```
<cwc2 type="SharedResource" name="CWCJSAPI"/>
<script src="../chameleon/common/ajax/xhr.js" type="text/javascript"></script>
<script type="text/javascript">
```

(b)

Adding a div to the HTML template which contains the input fields:

```
      <div id="ReferenceLayer" name="ReferenceLayer">
        <table width="100%" border="0" cellspacing="4" cellpadding="0">
          <tr>
            <td colspan="3" align="center">
              <cwc2 type="Scalebar" units="KILOMETERS" visible="true"
width="250" height="3" />
            </td>
          </tr>
        </table>
      </div>
      <div id="AddPointLayer" name="AddPointLayer">
        <b>Add a point to the map</b><br/>
        Label:  <input type="text" name="label" size="25"><br/>
        X coordinate in pixels:  <input type="text"
name="x_coordinate" size="10"><br/>
        Y coordinate in pixels:  <input type="text"
name="y_coordinate" size="10"><br/>
        <a href="javascript:AddPoint();">Add point to map</a>
      </div>
```

Change to the css:

```
#AddPointLayer
{
  padding:0px;
  margin:0px;
  position:absolute;
  left:500px;
  top:139px;
}
```

(c)

Making sure the session id is available in the HTML template. Edit the index.phtml file:

```
$oApp->CWCInitialize( $szTemplate, $szMapFile  );
$oApp->setVar( 'gszSessId', 'sid='.session_id() );
$oApp->mnMinimumMaturityLevel = MATURITY_ALPHA;
```

Btw, alternatively you could use the javascript variable:

```
goCWCJSAPI.sid = "44eef1b900078";
```

Implementing the AddPoint javascript function:

```
    function AddPoint()
    {
      call('../chameleon/widgets/MapNotes/properties.php?[$gszSessId$]&layer=Map
NotesAnnotation&font=sans&fontsize=10&fontcolor=0%200%200&fontoutlinecolor=255%2
0255%20255&symbol=circle&symbolsize=8&symbolcolor=0%200%200&symboloutlinecolor=2
55%20255%20255&marquee=1&offsetx=6&offsety=0&labelposition=CR&mode=ADD&update=1&
mouse_x=' +
        document.mainform.x_coordinate.value +
'&mouse_y='+document.mainform.y_coordinate.value + '&label=' +
document.mainform.label.value, null, processAjax);
    }
```

Implementing the processAjax function which will update the map so we don't need to pan the map to get the update in:

```
    function processAjax(szResults)
    {
      //alert(szResults);
      goCWCJSAPI.CallServer("goCWCJSAPI.MapExtentsUpdated()",aHiddenVars);
    }
```

(e)
Add javascript to show the

```
    function AddPoint()
    {
      CWCDHTML_ShowLayer("ActivityLayer");
      call('../chameleon/widgets/MapNotes/properties.php?[$gszSessId$]&layer=Map
NotesAnnotation&font=sans&fontsize=10&fontcolor=0%200%200&fontoutlinecolor=255%2
0255%20255&symbol=circle&symbolsize=8&symbolcolor=0%200%200&symboloutlinecolor=2
55%20255%20255&marquee=1&offsetx=6&offsety=0&labelposition=CR&mode=ADD&update=1&
mouse_x=' +
        document.mainform.x_coordinate.value +
'&mouse_y='+document.mainform.y_coordinate.value + '&label=' +
document.mainform.label.value, null, processAjax);
    }

    function processAjax(szResults)
    {
      //alert(szResults);
      goCWCJSAPI.CallServer("goCWCJSAPI.MapExtentsUpdated()",aHiddenVars);
      CWCDHTML_HideLayer("ActivityLayer");
    }
```

Exercise 4

1) Copy the “foss4g” widget folder found in the “foss4g_widget_template” to the “chameleon/widgets/” folder.
2) Add the xhr.js include line to the “GetJavascriptIncludeFunctions” function of the widget:

```
function GetJavascriptIncludeFunctions()
{
    // init
    $aReturn = array();

    // include the ajax code
    $szVar = "xhr.js";
    $aReturn[$szVar] = '<script src="../chameleon/common/ajax/xhr.js"
                                        type="text/javascript"></script>';
    // return
    return $aReturn;
}
```

3) Remove the include from the template.
4) Copy the “AddPoint” and “processAjax” functions to the “GetJavascriptFunctions” function:

```
function GetJavascriptFunctions()
  {
      // init
      $aReturn = array();

      // create the add point function
      $szJsFunctionName = "AddPoint";
      $szSessionId = session_id();
      $szFunction = <<<EOT
/**
  ____________________________________________________________________________
  |
  | Add Point
  |___________________________________________________________________________

 **/
function {$szJsFunctionName}()
{
      CWCDHTML_ShowLayer("ActivityLayer");
      call('../chameleon/widgets/MapNotes/properties.php?sid={$szSessionId}&laye
r=MapNotesAnnotation&font=sans&fontsize=10&fontcolor=0%200%200&fontoutlinecolor=
255%20255%20255&symbol=circle&symbolsize=8&symbolcolor=0%200%200&symboloutlineco
lor=255%20255%20255&marquee=1&offsetx=6&offsety=0&labelposition=CR&mode=ADD&upda
te=1&mouse_x=' +
        document.mainform.x_coordinate.value +
'&mouse_y='+document.mainform.y_coordinate.value + '&label=' +
document.mainform.label.value, null, processAjax);
}
EOT;
      $aReturn[$szJsFunctionName] = $szFunction;

      // create the processAjax function
      $szJsFunctionName = "processAjax";
      $szFunction = <<<EOT
/**
  ____________________________________________________________________________
  |
  | Process Ajax
  |___________________________________________________________________________

 **/
function {$szJsFunctionName}()
```

```
{
      //alert(szResults);
      goCWCJSAPI.CallServer("goCWCJSAPI.MapExtentsUpdated()",aHiddenVars);
      CWCDHTML_HideLayer("ActivityLayer");
}
EOT;
        $aReturn[$szJsFunctionName] = $szFunction;

        // return the array of functions
        return $aReturn;
    }

    /**
     _____________________________________________________________________________
     |
     | DrawPublish function
     |_____________________________________________________________________________

     **/
    function DrawPublish()
    {
        // only draw if visible
        if (!$this->mbVisible)
        {
            return "<!-- foss4g widget hidden -->";
        }

        // draw
        $szResult = <<<EOT
        <div id="AddPointLayer" name="AddPointLayer">
        <b>Add a point to the map</b><br/>
        Label:  <input type="text" name="label" size="25"><br/>
        X coordinate in pixels:  <input type="text"
name="x_coordinate" size="10"><br/>
        Y coordinate in pixels:  <input type="text"
name="y_coordinate" size="10"><br/>
        <a href="javascript:AddPoint();">Add point to map</a>
        </div>
EOT;

        // return
        return $szResult;
    }
```

5) Remove the functions from the template.
6) Add the code to the “DrawPublish” function:

```
    function DrawPublish()
    {
        // only draw if visible
        if (!$this->mbVisible)
        {
            return "<!-- foss4g widget hidden -->";
        }

        // init
        $szResult = '';

        // draw
        $szResult = <<<EOT
        <div id="AddPointLayer" name="AddPointLayer">
        <b>Add a point to the map</b><br/>
        Label:  <input type="text" name="label" size="25"><br/>
```

```
        X coordinate in pixels:  <input type="text"
name="x_coordinate" size="10"><br/>
        Y coordinate in pixels:  <input type="text"
name="y_coordinate" size="10"><br/>
        <a href="javascript:AddPoint();">Add point to map</a>
        </div>
EOT;

        // return
        return $szResult;
    }
```

7) Remove the code from the template and replace it with the widget definition:

```
      <cwc2 type="foss4g" visible="true"></cwc2>
```