# Notes for Development:

# How Tos:

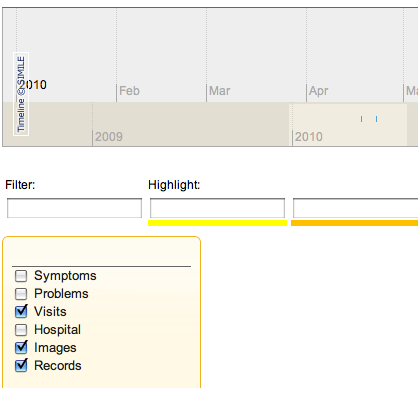
## To Add Events to the Timeline

The Events for the timeline are generated from the PatientListEventResource object under restlet package.

It calls a method:

Event.*retrieveEvents(…)*

Which takes the list of event types (‘Hospital’ ,’Records’,’Visits’,…) etc,.. as input. (Look at the list on the Timeline object.) and returns an ArrayList of Event objects.



The JSON is then generated for each Event and the result is returned in a JSON format that the timeline object can use.

To add the functionality to bring up a new Visit record:

Look at the following:

<http://simile.mit.edu/wiki/Timeline/Own_Handler_Instead_Of_Bubble>

Timeline.DurationEventPainter.prototype.\_showBubble = function(x, y, evt) { alert (evt.getDescription ()); }

Inside of medCafe.timeline.js

Is a method:

fillInfoBubbleCustom(evt, div, this.\_params.theme, this.\_band.getLabeller());

It is this method that should be used to generate the code for clicking on the link and bringing up the related data (whether image etc,) inside of medCafe.

## Overall Functionality

Two main javascript files that deal with generic medCafe functionality:

medCafe.js

medCafeTabs.js

Between these two in the majority of non widget specific medCafe functionality. i.e. deals with adding of a new tab, drag and dropping of tabs, saving of tabs (very complex.).

Most other widgets have their own javascript file following this convention:

medCafe.<widgetname>.js

e.g. medCafe.allergies.js

How Tabs get Loaded:

In index.jsp, there is a call to loadWidgetData for both patient specific and general widgets

This calls loadWidgetData in medCafe.widget.js

loadWidgetData makes a call to widgets-listJSON.jsp which lists all the widgets based on the WidgetList.xml document (located under config.)

Brings back the following JSON as an example:

{“widgets”:[

…

{"id":1,"repository":"OurVista","name":"Timeline","image":"images/timeline.png","method":"","clickUrl":"timelineJSON.jsp","params":"","type":"Timeline"}

…]

}

This is then turned into html using the template,

listWidgets.zm (and associated v2js\_listWidgets javascript).

Resulting in html of the following format:

<div class=”imageContain”>

<img custom:repository="OurVista" custom:params="" custom:id="1" custom:type="Timeline" custom:url="timelineJSON.jsp" alt="Timeline" src="images/timeline.png">

</div>

These are the major parameters used for further processing.

* **Repository** : tells the widget wether this is local, OpenVista, or hData.
* **Type**: the type of widget, very important parameter for further processing.
* **Url**: where to get associated data for this widget (usually , but not always JSON data).
* **Src**: The image to display for the widget.

Then for each “imageContain” class a ‘startWidgetDrag” function is associated with the mouseDown event (touchmove for case of iPad).

This calls ‘startWidgetDrag’ in medCafe.js.

This clones the image object and allows fr dragging to anywhere on screen. (Mainly holdover from when the widget listing was in its own iFrame.)

This then allows for dragging of the cloned image object, until a drop event is detected on the ‘droppable area’. This is an area inside the tabs\_template.jsp object.

A “widgetContent” class.

(This is why we always need at least an empty tab- so as to be able to register a ‘dropped’ event of a widget).

Inside of tabs\_template.jsp is a method that triggers on dropping of an image object (maybe need to check that this is of type imageContain class?)

This method beginning:

$(".widget-content").droppable({

drop: function(event, ui)

{…

Checks if this is a valid object to drop, i.e. is an imageContain object, with associated meta data embedded in the html (see above).

The code then parses out the html data embedded to extract all the meta data needed for further processing. E.g. widget type and url, etc,.

It then looks, using the repository parameter for the associated patient id, from that repository. Every repository will probably have its own id for patients.

Then a check is carried out to see if this tab is empty or contains a widget.

If it does contain a widget, then a new tab will be created.

If it doesn’t then the current tab will be used for the new widget.

The most common case is that widget has content:

This calls the function:

createLink(patientId,link, text, type ,params, repository, repPatientId);

CreateLink in medCafeTab.js is the major method for creation of any new Widget.

medCafeTabs.CreateLink function :

Creates a new Tab, calling addTab method.

The addTab method, uses the label and the type of widget to create a new empty tab object. It then loads a new tabs-template.jsp object into this tab. And then selects the new tab to have focus.

After the new tab object has been added, and loaded with the tabs-template.jsp data, the next step is to create the widget specific data.

This is done through calling createWidgetContent function on medCafe.js

# To Do List

Annotate Images – zooming shapes, moving image (On Annotate):

To bring up this functionality, click on a Images, and click a document, then when the document is brought up in a new tab, click the Annotate button. (This is the only widget that is currently in iFrame.)

Tag user specified shapes with associated Notes. (Probably need to use a right click menu).

The image won’t drag when I have added the capability to draw on shapes on the image. Due to the fact that the mousedown event on the image has overwritten the mousedown on drag for the underlying canvas.

I have tried various methods, without success, so added navigation buttons at the bottom. Unfortunately the shapes do not always stay in sync with the underlying image.

Tag Clouds?

## Test with iPad

It’s been a while since I last tested with the iPad.

## Image

Images don’t display unless the Image tab has focus. Same for Timeline widget.

## Timeline

Currently the First Visit band is hard coded. This information needs to be retrieved using first date from Mary’s OpenVista Visits.

Click on an appointment in the Timeline, and bring up details of the visit. (This info from Mary.), see above on example of how to do this.

Keyboard keeps popping up on iPad when click Timeline. Annoying.

## Modal Windows

Since moving from iFrames, the modal windows do not always work well. Needs testing.

All modal windows are called Editor Tab.

## Associate Patient

We need a mechanism to add a new patient to our system, when they exist currently in VistA. Dependant upon Patient cache functionality. (More documentation here.)

## Add an Image to database

Need a mechanism to add an image to our system, so that the image can be tagged with meta data.

## Listing of hData and OpenVista Data

As these are not patient specific, they shouldn’t be saved when we save patient data.

## Templates

Need to add template functionality. So that bring up a range of widgets at start for a new patient.

## Vital Signs Charts

Use the vital signs data from Mary to create charts.

## Other Possible OpenVista Data

* Lab Visits.
* Lab results/ labs pending?

## Header Information

Make sure that the header info is populated. (Vitals, Patient data, Problem list, etc,.)

## Allow for Entering of Problem List data

## Licenses

Update Ray on the list of licenses of plugins that we are currently using.

## iPad

Sometimes a Tab will appear with a concatenated list of titles. Sporadic though. Hard to recreate.