



Developer Notes

Sugar Dashlets and You v0.9

1. Introduction

2. Dashlet Framework

- Overview

- Templating

- Categories

- Dashlet Base Class

- dashlet.js

3. Creating your own Dashlets

- Generic Dashlet Creation

- Custom Dashlets

- Making it Module Installable

- Refreshing the Dashlet Cache

4. Help



Developer Notes

Sugar Dashlets and You v0.9

1. Introduction

Out of the box, SugarCRM 4.5 currently uses Dashlets on the Homepage module. This allows users to customize their views of data in a module. It also provides users with various charts and tools.

The Dashlet Framework allows developers to easily create new Dashlets that are installable into SugarCRM instances with no limitations.

Sugar Dashlets and You v0.9

2. Dashlet Framework

2.1. Overview

- Dashlets use the abstract factory
http://en.wikipedia.org/wiki/Abstract_factory_pattern design pattern. All Dashlets extend the base abstract class Dashlet.php.
- All Dashlets must be contained in the following directories:
modules/moduleName/Dashlets/
OR
custom/modules/moduleName/Dashlets/
- The file names need to match the Dashlet's class name. (eg. modules/Home/Dashlets/JotPad/JotPadDashlet.php contains the dashlet class JotPadDashlet.)
- There must also be a meta data file accompanying each Dashlet. The naming convention is: className.meta.php (eg. modules/Home/Dashlets/JotPad/JotPadDashlet.meta.php)

```
$dashletMeta['JotPadDashlet'] = array('title' => 'LBL_TITLE','description' => 'LBL_TITLE','icon' => 'themes/Sugar/images/Accounts.gif','category' => 'Tools');
```
- The 'title' and 'description' elements will be translated. The values are indexes in \$dashletStrings.
- Language files also have a similar naming convention:
className.locale.lang.php (eg. modules/Home/Dashlets/JotPad/JotpadDashlet.en_us.lang.php)
- Icon files can either be defined in the .meta data file or simply included in the Dashlet Directory (eg. modules/Home/Dashlets/JotPad/JotPadDashlet.icon.png). The system will scan for image files in the corresponding Dashlet directory.

Sugar Dashlets and You v0.9

2.2. Templating

The suggested templating engine for Dashlets is Smarty <http://smarty.php.net> however it is not required.

2.3. Categories

There are 5 categories for Dashlets.

- Module Views – Generic views of data in modules
- Portal – Dashlets that allow access to outside data (rss, webservices, etc)
- Charts – Charts of data
- Tools – Various tools like a notepad, calculator, or even a world clock!
- Miscellaneous - Any other Dashlets

2.4. Dashlets Base Class

- The main Dashlet base class is `include/Dashlets/Dashlet.php`. All Dashlets should extend this class.

```
class Dashlet {  
    // guid of the Dashlet, assigned on creation of the Dashlet  
    var $id;  
    // Title of the Dashlet  
    var $title = 'Generic Dashlet';  
    // true if the Dashlet has configuration options.  
    var $isConfigurable = false;  
    // true if the Dashlet contains javascript  
    var $hasScript = false;  
    // Language strings, must be loaded at the  
    // Dashlet level w/ loadLanguage  
    var $dashletStrings;  
    function Dashlet($id) {  
        $this->id = $id;  
    }  
}
```

- Each dashlet upon creation must be assigned a unique id. This id is used in the html id when the dashlet is displayed. This way multiple Dashlets of the same type can be included on the page.

Sugar Dashlets and You v0.9

```
// Title portion of the dashlet, $text is any text to be put
// in between the title and the Close icon
function getTitle($text) { ... }
// Displays the actual Dashlets
// *Inheriting Dashlet should override this
function display($text = ") { ... }
// Displays the Dashlets options
// *Inheriting Dashlet should override this if isConfigurable
function displayOptions() { ... }
// override if you need to do pre-processing before
// display is called
function process() {
}
// Override this if your dashlet is configurable
// (this is called when the configureDashlet form is shown)
function saveOptions($req) {
}
```

- saveOptions is called when the configuration form is submitted. This function need to filter out the \$_REQUEST array and only return an array of the options that needs to be saved

```
// loads language strings from files and stores them
// $dashletStrings. This will try to load the current language
// file first, then default of en_us
function loadLanguage($dashletClassname, $dashletDirectory) { ... }
// Generic way to store dashlet options into user preferences
function storeOptions($optionsArray) { ... }
// Generic way to load dashlet options from user preferences
function loadOptions() { ... }
```

- Dashlets are stored in the table user_preferences under the name 'dashlets' and the category 'home'. Here is a sample array element of 'dashlets' from the MyOpportunitiesDashlets



Developer Notes

Sugar Dashlets and You v0.9

```
[425f8ca8-663c-dabe-b483-448f6e4fb9a8] => Array
(
  [className] => MyOpportunitiesDashlet
  [fileLocation] =>
  ./modules/Opportunities/Dashlets/MyOpportunitiesDashl
  et/MyOpportunitiesDashle
  t.php
  [options] => Array
  (
    [filters] => Array
    (
    )
    [title] => My Top Open Opportunities
    [myItemsOnly] => true
    [displayRows] => 5
    [displayColumns] => Array
    (
      [0] => name
      [1] => amount
      [2] => team_name
    )
  )
)
```

The 'options' element stores the options for the Dashlet and this is the element that is loaded/stored by storeOptions / loadOptions functions in the base Dashlet class.

2.5. dashlet.js

Dashlet utility functions are located in include/javascript/dashlets.js. This contains the following methods:

```
postForm: function(theForm, callback) { }
```

postForm method is used to post the configuration form via AJAX. The callback will usually be SUGAR.sugarHome.uncoverPage to remove the configuration dialog.



Developer Notes

Sugar Dashlets and You v0.9

```
callMethod: function(dashletId, methodName, postData, refreshAfter,  
callback)  
{ }
```

callMethod is a generic way to call a method in a dashlet class. Use this function to generically call a method within your Dashlet class (php side).

You can refresh your Dashlet after a call and also utilize a callback function. This method can also be used as a way to proxy AJAX calls to webservices that don't exist on the SugarCRM install. (Google Maps Mashups for example.)

Sugar Dashlets and You v0.9

3. Creating Your Own Dashlets

3.1. Generic Dashlet Creation

The simplest Dashlet to create is a Module View. These are customizable Listviews of Dashlets. For this section we will use the MyAccountsDashlet as an example.

```
MyAccountsDashlet.php:
// include the base class
require_once('include/Dashlets/DashletGeneric.php');
// required for a seed bean
require_once('modules/Accounts/Account.php');
class MyAccountsDashlet extends DashletGeneric {
// takes an $id, and $def that contains the options/title/etc.
function MyAccountsDashlet($id, $def = null) {
require_once('MyAccountsDashlet.data.php');
parent::DashletGeneric($id, $def);
global $current_user, $app_strings;
$this->searchFields =
$dashletData['MyAccountsDashlet']['searchFields'];
$this->columns =
$dashletData['MyAccountsDashlet']['columns'];
```

All the meta data for this Dashlet is defined in the constructor. `$searchFields` are the search inputs that can be applied to the view. Defining these here will tell which input fields to generate corresponding filters when the user configures the Dashlet.

`$columns` define the available columns to the user. These contain the visible columns and the columns the user can make visible. These are defined in `MyAccountsDashlet.data.php` so that Studio can modify them easily.

```
// define a default title
if(empty($def['title'])) $this->title = 'My Account Dashlet';
$this->seedBean = new Account();
}
}
```

Sugar Dashlets and You v0.9

A seed bean is also required.

MyAccountsDashlet.data.php:

```
$dashletData['MyAccountsDashlet']['searchFields'] =  
array('date_entered' => array('default' => ''));  
$dashletData['MyAccountsDashlet']['columns'] = array(  
    'name' => array(  
        'width' => '40',  
        'label' => 'LBL_LIST_ACCOUNT_NAME',  
        'link' => true, // is the column clickable  
        'default' => true // is this column displayed by default  
    ),  
    'billing_address_state' => array(  
        'width' => '8',  
        'label' => 'LBL_BILLING_ADDRESS_STATE')  
);
```

This file along with the .meta.php file is enough to create a generic Module View Dashlet.

3.2. Custom Dashlets

Dashlets are more than generic Module Views. They can provide unlimited functionality and integration.

For this section we will use the JotPad Dashlet as an example. The JotPad is a simple note taking Dashlet. A user double clicks on the Dashlet and can enter any text in the Dashlet. When the user clicks outside of the textarea, the text is automatically saved via AJAX.

There are 6 files that define this Dashlet

- JotPadDashlet.php – JotPad Class
- JotPadDashlet.meta.php – meta data about the Dashlet
- JotPadDashlet.tpl – Display Template
- JotPadDashletOptions.tpl – Configuration template
- JotPadDashletScript.tpl - Javascript
- JotPadDashlet.en_us.lang.php – English Language file

Sugar Dashlets and You v0.9

JotPadDashlet.php:

```
// this extends Dashlet instead of DashletGeneric
class JotPadDashlet extends Dashlet {
    var $savedText; // users's saved text
    var $height = '100'; // height of the pad
    function JotPadDashlet($id, $def) {
        $this->loadLanguage('JotPadDashlet'); // load the language strings
        // load default text is none is defined
        if(!empty($def['savedText']))
            $this->savedText = $def['savedText'];
        else
            $this->savedText = $this->dashletStrings['LBL_DEFAULT_TEXT'];
        // set a default height if none is set
        if(!empty($def['height']))
            $this->height = $def['height'];
        // call parent constructor
        parent::Dashlet($id);
        // dashlet is configurable
        $this->isConfigurable = true;
        // dashlet has javascript attached to it
        $this->hasScript = true;
        // if no custom title, use default
        if(empty($def['title']))
            $this->title = $this->dashletStrings['LBL_TITLE'];
        else
            $this->title = $def['title'];
    }
    // Displays the dashlet
    function display() {}
    // Displays the javascript for the dashlet
    function displayScript() {}
    // Displays the configuration form for the dashlet
    function displayOptions() {}
    // called to filter out $_REQUEST object when the
    // user submits the configure dropdown
    function saveOptions($req) {}
    // Used to save text on textarea blur.
    // Accessed via Home/CallMethodDashlet.php
    function saveText() {}
}
```

Sugar Dashlets and You v0.9

JotPadDashletOptions.tpl:

```
<form name='configure_{ $id }' action="index.php" method="post"
onSubmit='return SUGAR.dashlets.postForm("configure_{ $id }",
SUGAR.sugarHome.uncoverPage);'>
```

The important thing to note here is the onSubmit. All configure forms should have this statement to uncover the page to remove the configuration dialog.

IMPORTANT: It is important to separate your javascript into a separate javascript file. This is because Dashlets are dynamically added to a page via AJAX. The HTML included into javascript is not evaluated when dynamically included.

It is important that all javascript functions are include in this script file. Inline javascript (<a href onclick=''' etc) will still function. If the Dashlet has javascript and a user dynamically adds it to the page, the Dashlet will not be accessible until after the user reloads the page.

Therefore it is beneficial to use as many generic methods in dashlet.js as possible (dashlets.callMethod()) specifically!).

JotPadDashletScripts.tpl:

```
{literal}<script>
// since the dashlet can be included multiple times a page,
// don't redefine these functions
if(typeof JotPad == 'undefined') {
JotPad = function() {
return {
blur: function(ta, id) {}, // called when textarea is blurred
edit: function(divObj, id) {}, // called when textarea is dbl clicked
saved: function(data) {}, // callback for saving
}();
}
}</script>{/literal}
```

Please refer to the file for more detail comments.

Sugar Dashlets and You v0.9

3.3. Making it Module Installable

To make a Dashlet Module Installable, you will need to package with the following also included in the manifest.php files.

```
$installdefs = array(  
    'id'=> 'jotpad',  
    'dashlets'=> array(  
        array('name' => 'JotPad',  
            'from' => '<basepath>/JotPad',  
        ),  
    ),  
);
```

More documentation on manifest files can be found in Sugar_Module_Loader_and_Upgrade_Wizard.rtf located here http://www.sugarforge.org/frs/?group_id=20

3.4. Refreshing the Dashlet Cache

To add a Dashlet to your SugarCRM installation, you can use the Module Loader to install your Dashlet Package.

However, for development purposes, to make the Dashlet available to add to the home page you will need to run the Repair Dashlet Link in the Admin Repair Panel at least once after you've created your Dashlet.

This will rebuild the cache file /<cache dir>/dashlets/dashlets.php by scanning the folders /modules/ and /custom/modules/ for Dashlet Files.

4. Help

If you have trouble developing Dashlets or have general questions, please visit the SugarCRM Developers Forums.

<http://www.sugarcrm.com/forums/forumdisplay.php?f=6>