

On the Receiving End of Web services

In this article we will cover integrating live stock updates within the Sugar application. Yahoo provides a web service that generates a comma separated file containing the stock prices of a ticker symbol. We will be utilizing this web service to add a Live Stock View to the Detail View of the Accounts module, but because Sugar 5 has an improved Object Oriented Model we will be adding this as a new Sugar Field Type that can be used in any module.

Account List

Account Name	Ticker Symbol	City	Phone	Team	User
Apple Computer	aapl 175.74(-3.58)			(admin)	admin
Oracle	orcl 23.04(-0.15)			(admin)	admin
Intel	intc 24.26(-0.10)			(admin)	admin
Cisco	cscoc 24.91(+0.24)			(admin)	admin
Cement Company	CX 21.74(+0.05)			(admin)	admin
Ford	f 15.11(+0.01)			(admin)	admin

The Web Service

The web service that yahoo provides will take in one more stock ticker symbols and return the current value in a comma separated file. To access this web service we simply visit the following website

<http://quote.yahoo.com/d/quotes.csv?s=<ticker symbol>&f=s1d1t1c1>

Where "s" is the ticker symbol and "f" is the formatting we want the data returned in. The above formatting string will return the data as SYMBOL, PRICE, LAST DATE UPDATED, LAST TIME UPDATED, DAYS CHANGE. So if we wished to see Apple's stock price, we know the ticker symbol is AAPL. We could simply visit the website

<http://quote.yahoo.com/d/quotes.csv?s=AAPL&f=s1d1t1c1>

Now it's easy to do this manually, but we are going to automate the data retrieval so the Sugar Application can fetch this data when needed. We also want to make sure any module can call on this function. So what we are going to do is create a custom view that is accessible by all modules.

The first thing we need to do is create the following files if they do not already exist.

1. `custom/include/MVC/View/Views/view.ticker.php`

2. `custom/include/MVC/Controller/action_view_map.php`

Now let's open `custom/include/MVC/View/Views/view.ticker.php`. We are going to create a class called ViewTicker that extends SugarView. Make sure our constructor calls on the SugarView constructor. Here is what the code should look like so far.

```
<?php
class ViewTicker extends SugarView {
    function ViewTicker(){
        parent::SugarView();
    }
}
```

Now we are going to add a display function to our custom view that will use our REQUEST parameter containing 1 or more ticker symbols and it will return a JSON encoded result with the current prices of those symbols.

```
<?php
class ViewTicker extends SugarView {
    function ViewTicker(){
        parent::SugarView();
    }
}
/**
 * takes in 1 or more ticker symbols from the REQUEST parameter and
 * echos a JSON encoded associative array containing the SYMBOL and a formatted string of the current price and a formatted string of the date and time
 * @return
 */
function display(){
    //make sure that nothing has been displayed to the screen
    ob_clean();
    //only do the work if one or more ticker symbols are passed up
    //for multiple tickers simply place a space between them so for Apple(AAPL) and Ford(F) it would be AAPL F or in a url it would be AAPL+F
    if(!empty($_REQUEST['ticker'])){
        //ensure that we encode the ticker symbol since we are going to pass it in a url
        $ticker = urlencode($_REQUEST['ticker']);
        //open a file resource to the Yahoo Web Service URL passing in our ticker symbols and the formatting we wish to use
        //the format will return data in a comma separated file with the following columns SYMBOL, PRICE, LAST DATE UPDATED, LAST TIME UPDATED, DAYS CHANGE
        $handle = fopen("http://quote.yahoo.com/d/quotes.csv?s=" . $ticker . "&f=s1d1t1c1", "r");
        //did we get any results
        $found = false;
        //store the results in this array to be JSON encoded
        $result = array();
        //Loop over the result of the data
        while ($data = fgets($handle, 1000, "")) {
            //lets color our strings on if the stock is up(green), down(red), even(black)
            $font = 'black';
            //if there is a + in the day's price change set the color to green
            if(substr_count($data[4], '+') ==1) $font = 'green';
            //if there is a - in the day's price change set the color to red
            if(substr_count($data[4], '-') ==1) $font = 'red';
            //associative array between the SYMBOL and an array containing two strings the first is PRICE the second is DATE TIME
            $result[$data[0]] = array("<span style='color:$font>". $data[1] . " " . $data[4] . "</span>", $data[2] . " " . $data[3]);
            //mark that we have found a result
            $found = true;
        }
        //if nothing was found echo 'N/A' for the service is not available
        if(!$found) echo 'N/A';
        //get our JSON object
        $json = getJSONObj();
        //echo a JSON encoded version of our result data
        echo 'var result=' . $json->encode($result);
        //close our handle
        fclose($handle);
    }
}
```

```

    }
    //close database connections and everything else
    sugar_cleanup(true);
}
}

```

The display function will be called on by the SugarController whenever our view is utilized, but in order to allow our view to be accessed we need to add it to the `custom/include/MVC/Controller/action_view_map.php` so let's open that file and add the following code to it. This will register our ticker view for use.

```

<?php
//if the action passed in the url is ticker then map it to our ticker view.
$action_view_map['ticker'] = 'ticker';

```

Now we can visit the following URL <http://<ugarinstance>/index.php?module=Accounts&action=ticker&ticker=AAPL> and we will see

```
var result=["AAPL":["179.32(0.00)<Vspan>","8V14V2008 4:00pm"]]
```

We can also pass in more than one ticker by adding a "+" between them. Going to <http://<ugarinstance>/index.php?module=Accounts&action=ticker&ticker=AAPL+F> and we will get

```
var result=["AAPL":["179.32(0.00)<Vspan>","8V14V2008 4:00pm"],"F":["5.10(0.00)<Vspan>","8V14V2008 4:01pm"]]
```

The Javascript Class

Now that we can utilize this web service let's create a javascript class that will let us utilize this web service in our HTML pages. By having a javascript class to take advantage of the web service we can have our ticker prices continuously updating using AJAX without the need for the user to refresh the page.

We know at the very least we need the following functions

update- this will be our work horse. It does the AJAX requests to our ticker view
fail - this will be called on by the YAHOO.util.Connect library when our AJAX requests fail
render- this will be called on when our AJAX request succeeds and will take in the results of our request

Since we designed our ticker view to be able to take in one or more stock symbols we should ensure that our javascript class is able to take advantage of this feature. We should also allow for the web page to register symbols so as different areas of the page load different symbols can be registered. So we will need the following functions also:

register- this will add a stock symbol to the list of symbols to be updated
buildTickerString- this function will be utilized to cache the stock symbols as a string so we don't have to rebuild the string every time we call on update.

Also since we want the page to continuously update we are going to need to more functions

setNextUpdate- this will be called on by render to set the next timer that we want to fire off an update
start- this will kick off the whole stock ticker updating process

Create a javascript file called ticker.js and add the following code to it.

```

/**
 * SUGAR.ticker is a utility class for use in conjunction the ticker view.
 * The only function that should be called externally is SUGAR.ticker.register for registering stock symbols
 * On page load auto updating of the stock prices will automatically be started.
 * @author Majed Itani
 */
if(typeof(SUGAR.ticker) == 'undefined'){
    //this is the object we will be building off of
    SUGAR.ticker = {};

    //registered will contain a list of stock symbols we wish to update on the page
    SUGAR.ticker.registered = {};

    //registeredString is a string containa a list of all the stock symbols registered seperated by a "+" sign
    SUGAR.ticker.registeredString = "";

    //shortString is a boolean that when set to true we will only display the stock price
    SUGAR.ticker.shortString = false;

    //autoRefresh is a boolean that when set to true will have the system refresh
    SUGAR.ticker.autoRefresh = true;

    //refreshTime is the number of seconds between refreshes if autoRefresh is set to true
    SUGAR.ticker.refreshTime = 300; //@@300 it should update every 5 minutes

    /**
     * Does an async call to our ticker view
     */
    SUGAR.ticker.update = function(){
        //on a succesful call it will pass the results to our render function
        //on an unsuccessful call it will pass the results to our fail function
        var callback = {success:SUGAR.ticker.render,failure:SUGAR.ticker.fail};
        //since we want all modules to have access to this we will use the Home module since all users should have access to that
        YAHOO.util.Connect.asyncRequest('POST', 'index.php', callback, 'module=Home&action=ticker&ticker=' + SUGAR.ticker.registeredString);
    }

    /**
     * Fail is called on when an aync request is unable to connect to the server
     * @param {Object} o
     */
    SUGAR.ticker.fail = function(o){
        //we do nothing on failures, but we could alert the user
    }

    /**
     * Render will take the results of the request and update the any field with an id with the format <symbol>_stock
     * For example if we were gathering the stock price for Apple(AAPL) we would need an id on the page in the format aapl_stock
     * <span id="aapl_stock"></span>
     * @param {Object} o - the response object from the async request
     */
    SUGAR.ticker.render = function(o){
        //if we get text saying N/A the service is not available so do nothing
        if(o.responseText == 'N/A')return;
        //evaluate the JSON encoded text string this will generate a javascript variable called result that contains the data generated in our ticker view
        eval(o.responseText);
        //loop over the result
        for(stock in result){
            //get the stock symbol in lowercase formatting
            var stockL = stock.toLowerCase();
            //if there is an element on the page with that id we set the innerHTML of that element to the stock price

```

```

        if(document.getElementById(stockL + '_stock')){
            document.getElementById(stockL + '_stock').innerHTML = result[stock][0];
            //if shortString is turned off then we also append the time it was last updated at
            if(!SUGAR.ticker.shortString)document.getElementById(stockL + '_stock').innerHTML += ' @ ' + result[stock][1];
        }
    }
    //if autorefresh is enabled call on setNextUpdate which will prepare us for the next update
    if(SUGAR.ticker.autoRefresh){
        SUGAR.ticker.setNextUpdate();
    }
}

/**
 * setNextUpdate will set a timer to call on update at the defined refresh time
 */
SUGAR.ticker.setNextUpdate = function(){
    //set the timer
    window.setTimeout(function(){
        //call on update
        SUGAR.ticker.update();
        //refreshTime in seconds * 1000 gives us the refresh time in milliseconds
    }, SUGAR.ticker.refreshTime * 1000);
}

/**
 * Start will start the cycle of updating stock prices on the page
 */
SUGAR.ticker.start = function(){
    //build the registered string of stock symbols from the list of registered symbols
    SUGAR.ticker.buildTickerString();
    //if the string is empty then there are no stocks to be updated
    if(SUGAR.ticker.registeredString=='')return false;
    //call on update and start the process
    SUGAR.ticker.update();
}

/**
 * generates a string containing the stock symbols separated by "+"
 *
 * For example if we had Apple(aapl) and Ford(f) it would generate a string containing aapl+f
 */
SUGAR.ticker.buildTickerString = function(){
    //clear the string
    SUGAR.ticker.registeredString = '';
    //loop over the registered symbols
    for(i in SUGAR.ticker.registered){
        //if the string isn't empty we need the '+' to separate symbols
        if(SUGAR.ticker.registeredString != '')SUGAR.ticker.registeredString += '+';
        //add the symbol
        SUGAR.ticker.registeredString += i;
    }
}

/**
 * Registers a stock symbol for updating
 * @param {Object} symbol - the stock symbol
 */
SUGAR.ticker.register = function(symbol){
    SUGAR.ticker.registered[symbol.toLowerCase()] = symbol;
}

//on the load of the page start the stock updating
YAHOO.util.Event.addListener(window, "load", SUGAR.ticker.start);
}

```

The Ticker Sugar Field

Now that we have a javascript class to take advantage of our ticker view, we need to create a Sugar Field that can take advantage of the javascript class. Creating new field types in sugar is very easy.

First let's create the following files

1. include/SugarFields/Ticker/SugarFieldTicker.php

2. include/SugarFields/Ticker/DetailView.tpl

And let's place our ticker.js file in

include/SugarFields/Ticker/ticker.js

Now let's open up the **DetailView.tpl** file and add the following code to the file. This is all in Smarty templating syntax for more information on this syntax go to <http://www.smarty.net/>

```

{**/this will display the ticker symbol**}
{糖糖var key='value'}}

{**/If there is a ticker symbol we'll want to register it**}
{if !empty({糖糖var key='value' string=true})}

    {**/this is the span tag that will be populated with the price the id should be the lower case ticker symbol followed by _stock**}
    <span id='{糖糖var key='value' string=true}lower_stock'></span>

    {**/register the stock symbol with our javascript class**}
    <script>SUGAR.ticker.register('{糖糖var key='value'}');</script>

{**/end the if statement**}
{/if}

```

We still need to load our ticker.js file before any of this can work. Now open up **SugarFieldTicker.php** and add the following code.

```

<?php
require_once('include/SugarFields/Fields/Base/SugarFieldBase.php');
//have our class extend the base class
class SugarFieldTicker extends SugarFieldBase {
    function getDetailViewSmarty($parentFieldArray, $vardef, $displayParams, $tabindex) {
        static $loaded = false;
        //only echo out the line to include the ticker.js file once per page
    }
}

```

```

        if(!$loaded)echo '<script src="include/SugarFields/Fields/Ticker/ticker.js"></script>';
        $loaded = true;
        //call on the setup function
        $this->setup($parentFieldArray, $vardef, $displayParams, $stabinde);
        //render our template
        return $this->fetch('include/SugarFields/Fields/Ticker/DetailView.tpl');
    }
}
?>

```

Making a Ticker Field Tick

We need to convert our ticker_symbol field in the application from a type **varchar** to a type **ticker**. There are two approaches we could take here and it each one has it's own benefits. The first way I am going to show you is to change it in the company SugarObject. This will allow all modules that extend company - including accounts - to take advantage of this functionality. This includes any modules built using module builder that are of type company. **WARNING THIS APPROACH IS NOT UPGRADE SAFE. YOU WILL NEED TO MANUALLY MERGE THIS FILE ON UPGRADES.**

Start by opening the file

`include/SugarObjects/templates/company/vardefs.php`

and change

```

'ticker_symbol' =>
array (
    'name' => 'ticker_symbol',
    'vname' => 'LBL_TICKER_SYMBOL',
    'type' => 'varchar',
    'len' => 10,
    'comment' => 'The stock trading (ticker) symbol for the company',
    'merge_filter' => 'enabled',
),

```

to

```

'ticker_symbol' =>
array (
    'name' => 'ticker_symbol',
    'vname' => 'LBL_TICKER_SYMBOL',
    'type' => 'ticker',
    'dbType' => 'varchar',
    'len' => 10,
    'comment' => 'The stock trading (ticker) symbol for the company',
    'merge_filter' => 'enabled',
),

```

The second approach is upgrade safe, but we have to manually add it to each module we wish to support this functionality.

Start by creating the following file

`custom/Extension/modules/<module>/Ext/Vardefs/ticker.php`

Now open it and add the following code to it.

```

<?php
Dictionary['Account']['fields']['ticker_symbol']['type'] = 'ticker';
Dictionary['Account']['fields']['ticker_symbol']['dbType'] = 'varchar';

```

Either approach you took you will need to run the repair script located in admin->repair->Quick Repair and Rebuild.

Once this is done you should be able to go to an Account Detail View and see something like

Name:	Ford
Website:	
Ticker Symbol:	f5.11(+0.01) @ 8/15/2008 4:00pm
Member of:	

With the above code the stock price should update every 5 minutes.

The ListView Hack

We have it in the detail view, but wouldn't it be great to have it working in the List View where we can track several tickers at the same time.

WARNING NEITHER APPROACH IS UPGRADE SAFE. YOU WILL NEED TO MANUALLY MERGE THIS FILE ON UPGRADES.

The first way I am going to show you will add support to Ticker Fields for any object included ones that are not subclasses of company.

APPROACH 1 - CHANGING ListViewGeneric.tpl

To start open the file

around line 120 you will see code that looks as follows

```

elseif $params.type == 'multienum'
{
    if !empty($rowData.$col)
    {
        {counter name="oCount" assign="oCount" start=0}
        {assign var="vals" value="^,^"|explode:$rowData.$col}
        {foreach from=$vals item=item}
            {counter name="oCount"}
            {sugar_translate label=$params.options select=$item}{if $oCount != count($vals)},{/if}
        {/foreach}
    }{/if}
}
else
{
    {$rowData.$col}
}

```

```
{/if}
```

Change it to the following

```
{elseif $params.type == 'multienum'}
    {if !empty($rowData.$col)}
        {counter name="oCount" assign="oCount" start=0}
        {assign var="vals" value="^,^" |explode:$rowData.$col}
        {foreach from=$vals item=item}
            {counter name="oCount"}
            {sugar_translate label=$params.options select=$item}{if $oCount != count($vals)},{/if}
        {/foreach}
    {/if}
    {elseif $params.type == 'ticker'}
        {if $id == 0}
            <script src="include/SugarFields/Fields/Ticker/ticker.js"></script>
            <script>SUGAR.ticker.shortString=true;</script>
        {/if}
        {$rowData.$col}{if !empty($rowData[$col])}&nbsp; <span id='{ $rowData.$col }_stock'></span><script>SUGAR.ticker.register('{ $rowData.$col }');
```

APPROACH 2 - CHANGING Company.php

Open the file `include/SugarObjects/templates/company/Company.php`

and add the following function to the Company class.

```
function get_list_view_array(){
    $temp_array = parent::get_list_view_array();
    static $loadedJS = false;
    if(!empty($temp_array['TICKER_SYMBOL'])){
        $ticker = $temp_array['TICKER_SYMBOL'];
        $_ticker = strtolower($temp_array['TICKER_SYMBOL']);
        //only want to load the js file once
        if(!$loadedJS){
            $ticker .= '<script src="include/SugarFields/Fields/Ticker/ticker.js"></script><script>SUGAR.ticker.shortString=true;</script>';
            $loadedJS = true;
        }
        //create the span tags we will populate with the data and register the ticker
        $ticker .= "<span id='{$_ticker}_stock'></span><script>SUGAR.ticker.register('{$_ticker}');
```

In 5.0 and 5.1 there is a slight bug in that Accounts still extend Sugar Bean instead of Company. So open up `modules/Accounts/Account.php`

and change it to

```
require_once('include/SugarObjects/templates/company/Company.php');
// Account is used to store account information.
class Account extends Company {
```

USE ONE OR THE OTHER APPROACH DO NOT USE BOTH

now going to your accounts list view we can see something like

Account List						
Select ▾	Delete	Export	Merge Duplicates	Compose Email	Selected: 0	(1 - 6 of 6)
Account Name	Ticker Symbol	City	Phone	Team	User	
Apple Computer	aapl 175.74(-3.58)			(admin)	admin	🗑
Oracle	ord 23.04(-0.15)			(admin)	admin	🗑
Intel	intc 24.26(-0.10)			(admin)	admin	🗑
Cisco	cscs 24.91(+0.24)			(admin)	admin	🗑
Cement Company	CX 21.74(+0.05)			(admin)	admin	🗑
Ford	f5.11(+0.01)			(admin)	admin	🗑

