

# Alfredo-Patch

## Introduction

Alfredo-Patch provides two small, but important additions to the SugarCRM framework:

- *Injection of arbitrary Javascript code into HTML generated by SugarCRM.*  
Javascript code may be inserted in HTML globally (for all modules) or on a per-module basis.
- *Execution of “pre-session” initialization scripts.*  
This feature is used to execute some arbitrary code BEFORE SugarCRM does a PHP `session_start()`. A use case for this feature is e.g. the requirement to store custom objects in PHP's `$_SESSION` variable – the classes must be known to PHP *before* the `session_start()` is done, otherwise the unserialization of these objects will miserably fail.

To implement these features, Alfredo-Patch overwrites two of SugarCRM's core files. Agreed, this is DIRRRTY, but sometimes this is the only way to extend SugarCRM's functionality... Alfredo-Patch was designed to keep the dirty part of the Alfredo project in one small, maintainable module. No other module should ever overwrite one of SugarCRM's core files.

This also means that every release of Alfredo-Patch will be tied to EXACTLY one patch version of SugarCRM. However, it is easy to adapt Alfredo-Patch's changes to a new release of SugarCRM, as I'll explain shortly.

## Implementation

### Extensions to SugarView.php

In `include/MVC/View/SugarView.php`, the function `displayJavascript()` is extended to scan some additional locations for include files:

```
//
// Kludge aw@abcona.de 2009-08-02 ff:
// Scan additional locations for Javascript includes
// Each found PHP script should simply echo one or more <script... > tags
//
foreach (glob('custom/application/Ext/javascript/*.php') as $auxJS)
{
    $GLOBALS['log']->debug("(G) Using additional JS from $auxJS");
    require_once($auxJS);
}
foreach (glob('custom/modules/' . $this->module . '/Ext/javascript/*.php') as $auxJS)
{
    $GLOBALS['log']->debug("(M) Using additional JS from $auxJS");
    require_once($auxJS);
}
```

```
foreach (glob('modules/' . $this->module . '/Ext/javascript/*.php') as $auxJS)
{
    $GLOBALS['log']->debug("(M) Using additional JS from $auxJS");
    require_once($auxJS);
}
```

Each `require_once`'d file is supposed to emit one or more `<script>` or `<link>` tags, which are appended to the generated HTML's Javascript section.

Thus, if you want to have some Javascript embedded in your page, follow these two simple rules:

- If you want to have some Javascript included only for *one* module, drop an include file into either...

**modules/<your module>/Ext/javascript/** or  
**custom/modules/<your module>/Ext/javascript/**

- If you want to have some Javascript embedded for *all* modules in SugarCRM, drop an include file into

**custom/application/Ext/javascript/**

## ***A somewhat complex example***

Here is an example for an complex include file taken from the Alfredo Connector project. This file is supposed to live in `modules/abcna_AlfDocuments/Ext/javascript/Alfredo-ext.php` and thus, according to the rules given, a per-module extension.

```
<?php

require_once('modules/abcna_AlfDocuments/include/include-styles.php');
require_once('custom/include/javascript/jquery/1.3/include-all.php');
require_once('modules/abcna_AlfDocuments/include/include-logic.php');

switch ($_REQUEST['action']) {
    case 'DetailView':
        echo "<script type='text/javascript' src='modules/abcna_AlfDocuments/include/DetailView-ext.js'></script>\n";
        break;
    case 'EditView':
        echo "<script type='text/javascript' src='modules/abcna_AlfDocuments/include/EditView-ext.js'></script>\n";
        break;
    case 'index':
        echo "<script type='text/javascript' src='modules/abcna_AlfDocuments/include/index-ext.js'></script>\n";
        break;
    case 'Popup':
        echo "<script type='text/javascript' src='modules/abcna_AlfDocuments/include/Popup-ext.js'></script>\n";
        break;
    default:
        echo "<!-- Alfredo was here (no script for action=" . $_REQUEST['action'] . " -->\n";
}
?>
```

Points worth noting in this example:

- Includes for dependent libraries are wrapped in some other PHP include and `require_once`'d. By using this technique, we can make sure that each Javascript `<script>` tag is only emitted once (as PHP keeps track which files were already required).
- We can dynamically select which code is emitted by evaluating some arbitrary request parameter (but keep in mind that your file may be as simple as a simple “echo ....” oneliner!).

## Extensions to SugarApplication.php

In include/MVC/SugarApplication.php, function startSession() is augmented to contain this block of code:

```
foreach(glob("modules/__autoload/*.php") as $auxClassDef) {
    $log->debug("Preload classdefs from $auxClassDef");
    require_once($auxClassDef);
}
foreach(glob("custom/modules/__autoload/*.php") as $auxClassDef) {
    $log->debug("Preload classdefs from $auxClassDef");
    require_once($auxClassDef);
}
```

By dropping some PHP include in one of the designated locations, any arbitrary action may be executed before SugarCRM eventually does a PHP session\_start().

However, the intended usage pattern is to require custom PHP classes before the session is started, otherwise PHP fails to retrieve such objects from \$\_SESSION.

See for example this include, which pulls in the PHP classes from Alfresco's PHP API:

```
<?php
//
// Preload class defs
//
// Save current include path
$save_include_path = get_include_path();
// Temporarily extend include path
$sugarRoot = realpath(dirname(__FILE__) . "/../..");
set_include_path("$sugarRoot/modules/Alfredo-PHPAPI/include" . PATH_SEPARATOR . get_include_path());
// Includes for Alfresco/PHP
require_once 'Alfresco/Service/NamespaceMap.php';
require_once 'Alfresco/Service/Repository.php';
require_once 'Alfresco/Service/Session.php';
require_once 'Alfresco/Service/SpacesStore.php';
require_once 'Alfresco/Service/Node.php';
// Restore previous include path...
set_include_path($save_include_path);
?>
```

Please note how this script takes care to *not* alter the global include path! Although this is not a strict requirement, it is a good defensive programming strategy to avoid side effects

## **Conclusion**

At a first glance, the usefulness of Alfredo-Patch might not be obvious. However, we made good use of it in the Alfredo project, so you should carefully study the other modules in this project to see why and where Alfredo-Patch might be useful. Enjoy!

## **Copyright**

Alfredo source files and documentation are © abcona e.K ([www.abcona.de](http://www.abcona.de)) and released to the public under the terms of the GNU Public License, Version 3.