To: Localdev

From: Don Bellenger

Subj: Some guidance on the code structure of Calendar, Library, and FileExchange

Date: 12/08/2017

This document provides a road map for someone looking at code in the ***Calendar***, ***FileExchange***, and ***Library*** ***Modules***. I would expect that the next developer would make changes (there are “a million ways to do things”), but here is guidance for the “as-built” status.

These three modules were developed from scratch, and refactored in the PMSESCB environment, and share some structure in common. These approaches have stood the test of time, and share common code in several places. They are derived from earlier work by Chad and Robert, and expanded by using newer/better techniques available in coldbox/commandbox. Here is a summary or techniques/structure in no particular order.

***There is common code*** shared by calendar, fileexchange, and library (“calfillib”) in particular locations.

Here are the code locations:

* Handlers/aroundhandler.cfc – This is a key replacement of the responseBean.cfc approach used in other modules.
* Includes/css/calfillib/\*
* Includes/css/uiSecuritySelection.css
* Includes/js/calfillib/\* In particular, includes/js/calfillib/calfillib.js include JS routines shared among the modules
* Includes/js/uiSecuritySelection.js – used in all pmsescb modules
* Includes/js/modal.js – used in all pmsescb modules. However, this file contains the important createRestfulActionModal (), which is used instead of createActionModal() used in other modules. This is because the new routine used the response packet defined in aroundhandler.cfc
* Includes/helpers/calfillibBeanHelper.cfc has common routines included in a module bean like modules/library/models/beans/libraryBean.cfc
* Includes/helpers/calfillibHandlerHelper.cfc has common routines included in a handler like modules/library/handlers/req.cfc
* Includes/helpers/
* Models/beans/response.cfc This is a replacement for models/beans/responseBean.cfc. It extends the “standard” coldbox restful bean in pms-es.modules.cbrestbasehandler.models.Response
* Models/m\_calfillib.cfc. Contains useful CF functions
* Model/m\_functions.cfc, which is included in models/m\_common.cfc. Note that m\_common.cfc is built into every module in pmsescb. Contains useful CF functions

This code is brought in to the calfillib modules in standard coldbox (CB) ways:

* moduleConfig.cfc has lines like binder.map("calFilLib").to("pms-es.models.m\_calfillib");
* layouts/layoutpart1 has lines like <link rel="stylesheet" type="text/css" href="includes/css/calfillib/table.css" media="all">
* layouts/layoutpart2 has lines like <script type="text/javascript" src="includes/js/uiSecuritySelection.js"></script>

***JS code structure***. These files are brought in via the layout files in each module

There are multiple JS files for each module. The first one in the list in the layout file contains a JS global variable, and a JS routine – glalrt. Use glalrt instead of alert within the JS routines, to control whether or not the alert shows by changing gllevel between 1 and 0.

var gllevel = 0;

function glalrt (s){

if ( gllevel != 0 ) {

alert (s);

}

} // glalrt

It is unwieldy to have all the JS for a module in one JS file. So, there is loosely one JS file per major area of the module. For example, in library there is library.js, lbhome.js, document.js, lbbrowse.js, lbfolder.ls, lbsearch.js, version.js.

Library.js has the only $(document).ready function. It also has utility routines used throughout the modules.

Lbhome.js has initHomeView(), which is called from $(document).ready(). This is the typical way the home page in a module in kicked off.

Lbbrowse.js, lbsearch.js and lbhome.js contain routines related to the entry points to the library module from Navigation (Home, Search, and Browse).

Document.js version.js, and lbfolder.js have routines related to crud for the major objects in the module (document, version, and folder).

***JS Code Characteristics***

Use createRestfulActionModal instead of CreateActionModal

***Handlers and Models***

The handlers and models are almost a restful Coldbox structure. Pushing them over the line to a full restful approach would probably depend on simultaneously converting the JS to an organized structure, like vue.js.

***Handlers***

* In each module, all the handlers extend **a *base handler***, which extends pms-es.handlers.aroundHandler, which in turn extends coldbox.system.EventHandler. This gives all the handlers access to the common custom aroundHandler. Each base handler has a ***preHandler(), an onError(), and a onMissingAction()*** routine, designed to get the modules a consistent look and feel. The ***PreHandler ()*** performs vital functions, including ensuring that the oModule bean is instantiated when a user navigates into the module (see below). See the Prehandler code in modules/calendar/handlers/clbasehandler.cfc for the required code.
* Almost all methods in a handler are called from JS.
* Most use the around handler to return a standard packet to JS. We are using the response packet defined in aroundhandler.cfc, except where we are dealing with file upload and download, which use the old response packet format. It is vital that the list of handlers using the around handler be controlled accurately in the standard Coldbox variables this.aroundHandler\_only, or this.aroundHandler\_except, which are in the base handler.
* Handler code is kept small with the notable exception of things related to security (user, cog, program), which use the uiSecuritySelction scheme that is pmsescb-wide.
* Req.cfc is becoming obsolete, but contains routines designed to be called from JS with synchronous calls to get CF variables into the JS environment. This is being replaced by storing session JS values in form variables, to keep everything in the JS environment accessible without going to the back end.

***Beans***

Each module has its own bean, which is created when the user navigates into the module. It is session-scoped. So, once a user is within the module, the handlers have access to prc.oUser, from navigation, and prc.oModule, which is a module-specific session-scoped bean. The code location for the beans is

Calendar: modules/calendar/models/bean/calendarbean.cfc

Fileexchange: modules/fileexchange/models/bean/feBean.cfc

Library: modules/library/models/bean/libraryBean.cfc

Each bean includes the Includes/helpers/calfillibBeanHelper.cfc mentioned above. Each oModule bean has routines that perform similar functions; however, the implementations are different because the table and field names vary among modules.

As an example of the use of the bean, one routine common in all these beans is setsSearchFormVars ( formvars ) , which saves the values for the search form variables which the user has entered in this module (e.g. filename :a, tags: contract, research). The next time the user sees the search form, it will be prepopulated by a call to getsSearchFormVars (). The data has been saved within the session-scoped bean.

The code which instantiates the bean is in the base handler of the relevant module.

Beans occasionally make use of the USERPREFERENCE Oracle table, to save users’ preferences between sessions. There is a simple API implemented in common.getSimplePreference() and common.setSimplePreference(), where a developer can define variables to be preserved between sessions. There is potentially a record in USERPREFERENCE for each value for each user.

***Models***

Usually a Handler calls a Model to prepare a return. Models typically call another “DAO” model, if there is extensive interaction with the database. Functions in the DAO model typically return queries or arrays to the first model, which then makes HTML. This approach would change if converting to vue.js.

Models all extend a ***base model***, which contains the properties defining all the items that wirebox connects. This includes the log file, the “files” connection for uploading/downloading, “securityRolesandPrivs” for security, as well as a reference to all the other models in the module.

In order to avoid putting all the routines into a single model, or pair of models, the code is structured in a way similar to the JS routine code structure. For example,

Document.js calls handler document.cfc calls model m\_document.cfc calls m\_documentDAO.cfc

Version.js calls handler version.cfc calls model m\_version.cfc calls m\_version\_DAO.cfc

Etc.

In order to enforce standard variable names and general consistency within the methods of each model, and to simplify code, there are two small files that are included within the module method. It is expected that the method be called from the handler with prc.oUser in the argument string. prc.oUser is set in Navigation as the user logs in, and navigates the top menu structure.

The first of the two included files is

Library/models/***ousertovars.cfm*** , which contains lines like

local.vProgramID = \_oUser.getProject().getProgramId() ;

Similarly, many DAO model files build queries using variables defined in

Library/models/***setwhereforroles.cfm***, which depends on ***ousertovars.cfm*** , and which contains lines like

local.excludeDeniedPhrase = " NOT EXISTS (SELECT lbDocId

FROM lbDocUserDeny

WHERE lbDocUserDeny.lbDocId = d.lbDocId

AND userID = #theUserId# ) ";

Use of these two files greatly simplifies the code within the models.