RewriteCubiUIWithAngular
Rewrite cubi ui with Angularjs
Featured, angularjs

Updated May 12, 2014 by rockys...@gmail.com

Rewrite Cubi UI with Angularjs

This document gives steps to rewrite cubi UI with Angularjs

Why Angularjs

Web Architecture Changes

Openbiz Cubi is a mature web application platform built on top of previous generation web architecture and tools

- Traditional MVC framework. Web server implements Model, Viewer and Controller. Viewer layer is rendered from server side template.
- Limited web service supported. No build-in Restful API support
- Even though jQuery is used on browser side, many UI logic is executed at web server side.

The new web architecture is featured with

- Web server provide web services (e.g. Restful). Thus it can support multiple client including browser, mobile apps.
- Browser owns MVC stack, web server only provider layout. Data is fetched from server side and dynamically rendered on the page.
- Browser side tools are heavily used to make more clean and efficient client code.

Angularjs vs others

Angularjs https://angularjs.org/ gets more popular comparing with other javascript frameworks. See trend at
http://www.google.com/trends/explore?hl=en-US#q=Angularjs,+Backbone.js,+Ember.js&date=today+12-m&cmpt=q

Apply Angularjs on Cubi List Form

Prepare the changes

Things to be prepared before make code change

- Add Restful web service on Cubi app. https://code.google.com/p/openbiz-cubi/wiki/CubiRestService
- Download angularjs and put it under cubi/js/

Data model on both server and browser

It is important to have the concept that Browser contains data model in javascript. Such data model are usually the same data model as web service side.

- On query web service, it may return a list of data objects (we can call it dataset). On browser side, the dataset is stored into $scope.dataset. When we loop through $scope.dataset, each data record is named as dataobj.
- On query web service, it may return one data object. On browser side, the data object is stored into $scope.dataobj.

So on browser side, we have following naming convention that simplify our code.

- 'dataset' as a set of data records
- 'dataobj' as one data record

The code changes

Here we use UserListForm as an example. The form is at /cubi/modules/system/form/UserListForm.xml. The changes will happen on

Restful Web Service

- Add SystemRestService.php and RestService.xml under /cubi/modules/system/websvc/

```
<?php
include_once MODULE_PATH.'/websvc/lib/RestService.php';

class SystemRestService extends RestService
{
        protected $resourceDOMap = array('users'=>'system.do.UserDO');
}
?>
```

Form Metadata

- Form metadata xml file adds DataService and QueryString attributes to replace DataObject and SearchRule attributes
- Add ng-click on elements.

```
<EasyForm Name="UserListForm" Icon="icon_user_list.gif" Class="UserForm" FormType="List" jsClass="jbForm"
Title="User Management" Description="Manage user accounts in the application"
DataService="/system/users" PageSize="10" TemplateEngine="Smarty" TemplateFile="system_right_listform.tpl.html"  Ac
```

```
<NavPanel>
    <Element Name="btn_first"  Class="Button" CssClass="button_gray_navi first" HTMLAttr="ng-click='gotoPage(1)'"/>
    <Element Name="btn_prev" Class="Button" CssClass="button_gray_navi prev" HTMLAttr="ng-click='gotoPage(currentPage-
    <Element Name="txt_page" Class="LabelText" Text="{tx}{{currentPage}} of {{totalPage}}{/tx}"/>
    <Element Name="btn_next" Class="Button" CssClass="button_gray_navi next" HTMLAttr="ng-click='gotoPage(currentPage+
    <Element Name="btn_last" Class="Button" CssClass="button_gray_navi last" HTMLAttr="ng-click='gotoPage(totalPage)'"
</NavPanel>
```

Form class

- Form class keeps only render logic and delete the code that access data objects
- Form class adds code to read DataService and QueryString, then output them to template.

Template

- View template file adds "ng-app" in body tag
- Form template file adds angular Controller in form tag, use ng-... to replace existing logic.

```
<form id='{$form.name}' name='{$form.name}' ng-controller="TableFormController" ng-init="init('{$form.name}','{$for
...
<table>
        <tbody>
                <tr ng-repeat="dataobj in dataset" ng-class-odd="'odd'" ng-class-even="'even'" ng-class="highlightc
                {foreach item=cell key=elems_name from=$dataPanel}
                        <td>{$cell.element}</td>
                {/foreach}
                </tr>
        </tbody>
</table>
```

Javascript

- javascript file adds TableFormController that provides functions for search, paging, sort. Each controller use angular http component to fetch data from web service.

```
function TableFormController($scope, $http, $location) {
        $scope.currentPage = 1;
        $scope.totalPage = 1;
        $scope.sort = "";
        $scope.sorder = "";
        $scope.urlPath = $location.path();

        $scope.init = function(name, dataService) {
                $scope.name = name;
                $scope.dataService = dataService;

                $scope.gotoPage(1);
        }

        $scope.gotoPage = function(page) {
                if (page < 1) return;
                if (page > $scope.totalPage) return;
                $scope.fetchData(page, $scope.sort, $scope.sorder, null);
        }

        $scope.sortRecord = function(field) {
                // if sort on the field, toggle the sort order
                if (field == $scope.sort) {
                        if ($scope.sorder == "") fieldOrder = "asc";
                        else if ($scope.sorder == "asc") fieldOrder = "desc";
                        else fieldOrder = "asc";
                }
                else {
                        fieldOrder = "asc";
                }
                $scope.fetchData(1, field, fieldOrder, null);
        }

        $scope.search = function() {
                // run search with user input on searchPanel
                if (typeof $scope.searchPanel != 'undefined' && $scope.searchPanel != null) {
                        var elemValues = [];
                        for (var key in $scope.searchPanel) {
                                elemValues.push(key+"="+$scope.searchPanel[key]);
                        }
                        var queryString = elemValues.join("&");
                        $scope.fetchData(1, $scope.sort, $scope.sorder, queryString);
                }
        }

        $scope.fetchData = function (page, sortField, sortOrder, queryString) {
                var url = $scope.dataService+'/q?format=json';
                if (page != null) url += '&page='+page;
                if (sortField && sortOrder) url += '&sort='+sortField+'&sorder='+sortOrder;
                if (queryString) url += '&'+queryString;
```

```
                $http.get(url).success(function(responseObj) {
                        $scope.dataset = responseObj.data;
                        $scope.totalPage = responseObj.totalPage;
                        $scope.currentPage = page;
                        $scope.sort = sortField;
                        $scope.sorder = sortOrder;
                });
        }
}
```

Apply Angularjs on Left Menu and Main Tab

Here we apply angular on cubi/modules/menu/widget/ApplicationMenu.xml and cubi/modules/menu/widget/MainTabMenu.xml

Similarly, we need to

- add web service
- change menu metadata xml files
- change menu class MenuWidget.php
- template files
- add javascript for Angular controller

Template change cubi/modules/menu/template/vertical_menu.tpl

```
<div class="menu_title" >
<h2>{$widget.title}</h2>
<p style="float:right;display:block;padding-top:4px;">
<span style="display: block;float: left;"><a class="menu_index_link" href="{$app_index}/system/general_default">{t}Index
<a href="{$app_index}/system/general_default"><img id="system_dashboard" class="btn_dashboard"  src="{$image_url}/spacer.g
</div>
<div ng-controller="LeftMenuController" ng-init="init('{$form.name}','{$form.dataService}','{$form.queryString}')">
<ul class="toplevel {$widget.css} left_menu">
        <li ng-repeat="node in treeNodes">
                <a onclick="show_submenu(this)" ng-class="node.m_Current ? 'current':''">
                        <img ng-class="node.m_IconCSSClass" ng-src="{$image_url}/{literal}{{{/literal}node.m_IconImage{lit
                        {literal}{{{/literal}node.m_Name{literal}}}{/literal}
                </a>
                <ul class="secondlevel module" style="{literal}{{{/literal}node.m_Current ? 'display:block':''{literal}}}{/
                        <li ng-repeat="subitem in node.m_ChildNodes">
                                <a ng-class="subitem.m_Current ? 'current':''" href="{literal}{{{/literal}subitem.m_URL{lit
                                {literal}{{{/literal}subitem.m_Name{literal}}}{/literal}</a>
                        </li>
                </ul>
        </li>
</ul>
</div>
<div class="v_spacer"></div>
```

Javascript LeftMenuController

```
function LeftMenuController($scope, $http) {
        $scope.init = function(name, dataService, queryString) {
                $scope.name = name;
                $scope.dataService = dataService;

                var url = $scope.dataService+'/q?format=json';
                if (queryString) url += '&'+queryString;
                $http.get(url).success(function(responseObj) {
                        $scope.treeNodes = responseObj;
                        $scope.matchTreeNodes();
                });
        }

        $scope.matchTreeNodes = function() {
                // find the current node by matching with application breadcrumb
                bcIds = [];
                for (var k=0; k<breadCrumb.length; k++) {
                        bcIds.push(breadCrumb[k].id);
                }
                for (var i=0; i<$scope.treeNodes.length; i++) {
                        $scope.treeNodes[i].m_Current = bcIds.indexOf($scope.treeNodes[i].m_Id) >= 0 ? 1:0;
                        if ($scope.treeNodes[i].m_Current == 1) {
                                if ($scope.treeNodes[i].m_ChildNodes) {
                                        for (var j=0; j<$scope.treeNodes[i].m_ChildNodes.length; j++) {
                                                $scope.treeNodes[i].m_ChildNodes[j].m_Current = bcIds.indexOf($scope.treeNo
                                                if ($scope.treeNodes[i].m_ChildNodes[j].m_Current == 1) break;
                                        }
                                }
                                console.log($scope.treeNodes[i]);
                                break;
                        }
                }
        }
}
```

Apply Angular on Login form

Now we apply angular on login form which is under cubi/modules/user/form/LoginForm.xml.

Similarly, we need to

- add web service under cubi/modules/user/websvc/userService.xml and cubi/modules/user/websvc/userService.php
- change LoginForm metadata xml files
- change LoginForm class LoginForm.php
- template files login.tpl.html
- add javascript for Angular controller. We can append javascript code in the template file login.tpl.html

LoginForm.xml

```
<EasyForm Name="LoginForm" Class="LoginForm" ...>
   <DataPanel>
      <Element Name="username" FieldName="fld_username" Label="Username" Class="InputText"/>
      <Element Name="password" FieldName="fld_password" Label="Password" Class="Password"/>
...
   <ActionPanel>
      <Element Name="btn_login" Class="Button" Text="Login" CssClass="button_highlight" HTMLAttr="ng-click='login()'">
```

LoginForm.php, remove login method, and add it to userService.php

login.tpl.html

```
...
<form id="{$form.name}" name="{$form.name}" ng-controller="LoginFormController" ng-init="init('{$form.name}','{$form.dataSe
...
<p class="input_row" ng-class="{literal}{{/literal} 'has-error' : errors.username {literal}}{/literal}">
        <label style="width:60px;">{$dataPanel.username.label}</label>
        <span style="width:200px; display:block;float:left;">{$dataPanel.username.element}</span>
        <span class="input_error_msg" ng-show="errors.username">{literal}{{{{/literal} errors.username {literal}}}}{/literal
</p>

<p class="input_row" ng-class="{literal}{{/literal} 'has-error' : errors.password {literal}}{/literal}">
        <label style="width:60px;">{$dataPanel.password.label}</label>
        <span style="width:200px; display:block;float:left;">{$dataPanel.password.element}</span>
        <span class="input_error_msg" ng-show="errors.password">{literal}{{{{/literal} errors.password {literal}}}}{/literal}
</p>
...
<script>
{literal}
function LoginFormController($scope, $http, $window, $timeout) {
        $scope.dataobj = {};

        $scope.init = function(name, dataService) {
                $scope.name = name;
                $scope.dataService = dataService;
        }

        $scope.login = function() {
                console.log($scope.dataobj);
                var url = $scope.dataService+'/?method=login&format=json';
                var postQueryString = "argsJson="+angular.toJson($scope.dataobj);
                $http.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded; charset=UTF-8';
                $http.post(url, postQueryString).success(function(responseObj) {
                        if (responseObj.data.redirect) {
                                // redirect to new page
                                $scope.redirect = responseObj.data.redirect;
                                $window.location.href = $scope.redirect;
                        }
                        else if (responseObj.data.errors) {
                                // display error
                                $scope.errors = responseObj.data.errors;
                        }
                });
        }
}
{/literal}
</script>
```

Get the source

Get the latest angular cubi code from angular branch https://openbiz-cubi.googlecode.com/svn/branches/angular.

You can test it out several angular enable page so far.

- login page
- after login, you land on system dashboard page where main tab and left navigation menu have angular enabled
- click on user icon, you can see user list form has angular enabled