**CBMVC Framework Guide**

1. **Life cycle:**

Launch app ==> home view ==> home controller

**2. How to setup a controller:**

Your controller must be in /Resources/app/controller, and the corresponding view must be in Resources/app/views.

A controller and its corresponding view should have identical name.

For example:

/Resources/app/controllers/home.js

/Resources/app/views/home.js

Moreover, you need to add the controller's name into CB.Launch when launch the app

In app.js, it should look like this:

//load the controllers, the main controller must be the last one

Var controllers = ['home','login','mainFrame'];

CB.Launch(controllers);

**3. How to write a controller:**

please refer to the object \_\_exports.

If you declare \_\_exports.some\_function, then the controller will have some\_function method.

We also have a life-cycle for a controller, you can declare these methods:

\_\_exports.viewLoaded = function(e) { ... };

\_\_exports.viewWillAppear = function(e) { ... };

\_\_exports.viewDidAppear = function(e) { ... };

\_\_exports.viewWillDisappear = function(e) { ... };

\_\_exports.viewDidDisappear = function(e) { ... };

The methods' names are self-explanatory.

**4. How to switch between controllers:**

*a. Go to next view :*   
 You can call the following method to switch next view, and set the animation:

CB.pushController = function(controller, animate);

For example:

//switch to the login view and use ‘left’ animation

CB.pushController(CB.controllers.login,’left’);

*b. Go to home view :*   
 You can call the following method to switch home view, and set the animation:

For example:

//go to the home view and use ‘right’ animation

CB.Launch(null, null, 'right');

//or just refresh a view without animation

CB.Launch('home', true, 'none');

*c. Back to previous view :*   
 You can call the following method to back the previous view:

For example:

//just back to previous view

CB.popController();

*d. About the animation:*  
  
none without animation  
left or don't set it, for move to left animation  
right: for move to right animation, default move to left  
up: for move to up animation, default move to left  
down: for move to down animation, default move to left

**5. How to handle data and event:**

You can add the layout element(etc. view button) in the view page, and add the event listener in controller.

For example:

***Home view (/Resources/app/view/home.js):***

//create the base view

var view = Ti.UI.createView({

width : '100%,

height : '100%',

backgroundImage : CB.ApplicationDirectory + 'images/bg.png'

});

//create the button with the view

view.memberLoginBtn = Ti.UI.createButton({

left : 0,

top : '74%',

width : '100%',

height : '12%',

title : ' Member Login',

color : 'red',

textAlign : 'left',

font : {

fontSize : '18dp',

fontWeight : 'bold'

},

backgroundImage : CB.ApplicationDirectory + 'images/memLogin.png'

});

//add the button into the view

view.add(view.memberLoginBtn);

***Home controller (/Resources/app/controller/home.js):***

//the init events must be added to \_\_exports.viewLoaded

\_\_exports.viewLoaded = function() {

this.view.memberLoginBtn.addEventListener('click', function() {

//pass data to next controller

CB.controllers.login.data = 'test';

CB.pushController(CB.controllers.login);

});

};

***Get data within login controller (/Resources/app/controller/login.js):***

//Must be use viewWillAppear or viewDidAppear to get data and show it

\_\_exports.viewWillAppear = function(e) {

//check the data whether is available

if(e != undefined && e.data != undefined){

//set the data to view's element

this.view.loginLabel.text = e.data;

}

}

**6. How to set styles to the element:**

You can set an element’s style with ***CB.Styles*** namespace, add the style’s code in

/Resources/app/base/styles.js

For the better, create a difference namespace for each view’s style:

For example, add a style for home view:

***Styles (/Resources/app/base/style.js):***

(function() {

/\*\*

\* home view's styles

\*/

CB.Styles.home = {

baseView : {

width : CB.Styles.screenWidth,

height : CB.Styles.screenHeight,

backgroundImage : CB.ApplicationDirectory + 'images/bg.png'

},

goBtn : {

top : '50%',

height : '6%',

font : {

fontSize : '17dp',

fontWeight : 'bold'

},

title: 'Go Next'

},

switchLangBtn : {

top : '30%',

height : '6%',

font : {

fontSize : '17dp',

fontWeight : 'bold'

},

title: CB.Util.L('switchLang')

}

}

})();

In the home view , can use styles to create the elements now:

***Home view (/Resources/app/view/home.js):***

\_\_exports = (function() {

//set the views's name

var viewName = 'home';

//create element and set style with CB.Styles

var view = Ti.UI.createView(CB.Styles[viewName].baseView);

view.name = viewName; //just for add the refresh debug button

view.goBtn = Ti.UI.createButton(CB.Styles[view.name].goBtn);

view.add(view.goBtn);

view.switchLangBtn = Ti.UI.createButton(CB.Styles[view.name].switchLangBtn);

view.add(view.switchLangBtn);

return view;

})();

Also you can add some logic for handle different style(just like debug mode or cross platform logic), please refer to the ***/Resources/app/view/home.js*** file.

**7. How to debug the app:**

You can use ***CB.Debug*** for echo the debug message, there are 3 methods for debugger:

***Echo a debug message string:***

/\*\*

\* General echo the debug message

\* @param {String} s, echo which debug message

\* @param {int} line, the line of echo message

\* @param {String} page, the page which debug message show

\* @param {String} type, debug type, support Titanium debug type:

\* info: display message with [INFO] style in console

\* warn: display message with [WARN] style in console (default)

\* error: display message with [ERROR] style in console

\*/

CB.Debug.echo(s, line, page, type)

//for example:

CB.Debug.echo('Debug message, testing',33,'login controller',’warn’);

//or just show the debug message in line 33

CB.Debug.echo('Debug message, testing',33);

***Dump an object(JSON format)***

/\*\*

\* General dump the object

\* @param {Object} o, dump object

\* @param {int} line, the line of debug object

\* @param {String} page, the page which debug message show

\* @param {String} type, debug type, support Titanium debug type:

\* info: display message with [INFO] style in console

\* warn: display message with [WARN] style in console (default)

\* error: display message with [ERROR] style in console

\*/

function dump(o, line, page, type)

//for example:

CB.Debug.dump(this.view,10,'login controller','info');

//or just dump the object in line 10

CB.Debug.dump(this.view,33);

***Add a refresh function to the top bar***

/\*\*

\* Add a refresh function to top bar, then you can click the top bar to refresh the page

\* just for review the layout changed and testing without re-launch the app.

\* @param {Object} coreObj

\* @param {Object} view

\*/

function addRefreshBtn(coreObj, view)

//for example, add the button before view appear

\_\_exports.viewWillAppear = function(e) {

//add a refresh button for testing layout

CB.Debug.addRefreshBtn(CB, e.view);

};

**8. How to use ajax request:**

You need to set the remote server url for ajax in the following aip.js file at first:

***API (/Resources/app/base/api.js):***

/\*\*

\* set the remote API

\*/

CB.API = {

server : 'http://www.abc.com/'

};

(function(){

CB.Platform.extend(CB.API, {

//set each api function's url

login : CB.API.server + 'api/login.aspx'

info : CB.API.server + 'api/info.aspx',

});

})();

Get remote API data and redirect to next view:

***Call getRemoteData function (/Resources/app/base/common.js):***

/\*\*

\* Get date with remote API function

\* @param {String} api, the API's name

\* @param {Object} controller, which controller need to show after got data

\* @param {Boolean} saveData, save response data to local storage or just pass data to next view

\* 'true', save in local storage

\* 'false', just pass data to controller.model to next view (default)

\* @param {String} animate

\*/ getRemoteData : function(api, controller, animate)

//get the user info with info api and redirect to service category view

CB.Common.getRemoteData('info', CB.controllers.serviceCategory);

after that, you can access the user info data within service category controller :

***Handle Ajax Callback (/Resources/app/controllers/serviceCategory.js):***

//get user info within controller in \_\_exports.viewWillAppear event

\_\_exports.viewWillAppear = function(e) {

//get data from controller.model

if(e != undefined && e.model != undefined){

e.view.barTitle.text = CB.Util.L('hi') + e.model.first\_name + ' ' + e.model.last\_name;

}

//or get date from local storage with api name

var userInfo = CB.Util.loadObject('info');

if(userInfo != null){

e.view.barTitle.text = CB.Util.L('hi') + userInfo.first\_name + ' ' + userInfo.last\_name;

}

...

**9. How to localization:**

The framework can support multiple language, and can be effective immediately after changed the language:  
  
 1. Create a language xml file under **/Resources/app/languages/** folder, and please follow the following file name format :

lang.xml

//for example, English should be use

/Resources/app/languages/en.xml

//Chinese should be use

/Resources/app/languages/zf.xml

and in the xml language file, must be use the following format:

***/Resources/app/languages/en.xml***

<?xml version="1.0" encoding="utf-8"?>

<root>

<string id="logo">Welcome to Coder Blog!</string>

<string id="switchLang">Switch to Chinese</string>

</root>

After that, you can use the following to display the language with the key:

CB.Util.L('logo');

2. Switch language:

You can just call the following method to switch the language, after you do that, need to call :

CB.Util.switchLang('en');

after you do that, need to call the following method to refresh the page, so that you can see the changed language:

//refresh the page after changed the language

CB.Launch('home', true, ’down’);

3. Also you can set the default language with first launch the app:

***/Resources/app.js***

//set the default language with the app

CB.DefaultLang = 'en';