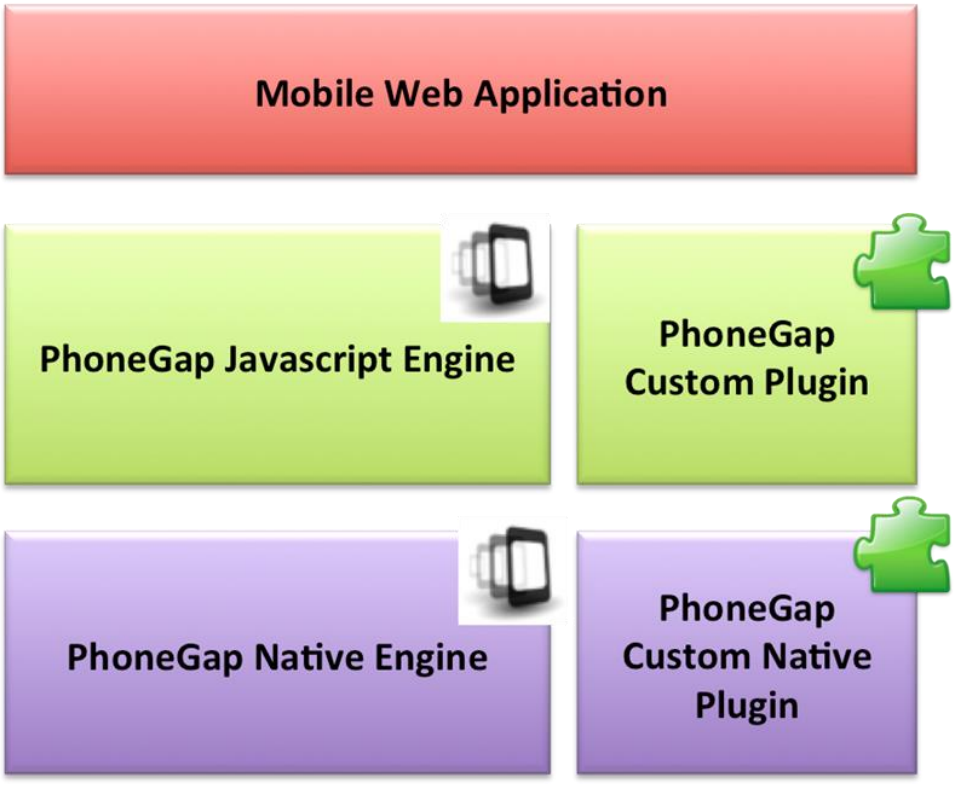
http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:hyb3.5:hrte:device_api_plugin

Outline

PhoneGap Framework supports Device API Plug-in for use of the functions unique to Native in Hybrid Application. PhoneGap Framework is made of JavaScript Engine, Native Engine and Customized Plug-in, the latter of which is intended to develop Device API Plug-in. PhoneGap is compatible with a variety of APIs developed based on the API PhoneGap Plug-in, where the user has a right to expand the functions required.



Android

Description

This Section describes how Android platform plug-in is to be designed. An Android platform plug-in comprises Native Java Class and JavaScript Function to call the Java Classes. To set the plug-in Java Class, the developer can use the Method Execute to return the Class PluginResult, inheriting the Class Plugin of PhoneGap Framework. See the following for how to prepare coding:

Plug-in Coding Practices

Mapping Plug-in

The JavaScript Function cordova.exe is used to call the Native Functions in Android, as follows:

```
exec(<successFunction>, <failFunction>, <service>, <action>, [<args>]);
```

Note that the foregoing function calls Native Functions of Android, as requested by web-view. Refer to the following argument roles:

- successFunction : Callback Function for JavaScript when Native Functions of Android is called.
- failFunction : JavaScript Callback Function when errors occur while Native Functions of Android is called.
- service : Name of the plug-in configured in plugin.xml
- action : Parameter transferred to the Native Function of Android for service classification
- args : Parameter array required for calling of Native Function of Android

Register the plug-in that you desire to use by adding strings in plugin.xml in res/xml, as follows:

<pluginname="<service_name>" value="<full_name_including_namespace>"/>

- name : Name of plug-in service transferred to cordova.exe
- value : Native Class of Android called by the function cordova.exe

Coding in Java Plug-in for Android

Coding in Java Plug-in for Android requires inheritance of the class Plug-in. Refer to the following for how to code in Java Plug-in for Android. Note that the Method “execute” is called by cordova.exe, followed by return of the Class PluginResult:

```
importorg.apache.cordova.api.CordovaPlugin;
importorg.apache.cordova.api.PluginResult;
importorg.json.JSONArray;
importorg.json.JSONException;
importorg.json.JSONObject;

/**
 * This class echoes a string called from JavaScript.
 */
publicclass Echo extends CordovaPlugin {
    @Override
publicboolean execute(String action, JSONArray args, CallbackContext callbackContext)throws
JSONException {
if(action.equals("echo")){
String message = args.getString(0);
this.echo(message, callbackContext);
returntrue;
}
returnfalse;
}

privatevoid echo(String message, CallbackContext callbackContext){
if(message !=null&& message.length()> 0){
                callbackContext.success(message);
}else{
                callbackContext.error("Expected one non-empty string argument.");
}
}
}
```

See the following for how parameters for the Method “execute” are used:

- action : Name of action used for categorization of plug-in services
- args : Parameters echoed by JavaScript calling plug-in
- callbackContext: Context Object echoing upon execution of plug-in

iOS

Description

This Section describes how iOS platform plug-in is to be designed. An iOS platform plug-in comprises Objective-C Class for Native Functions and JavaScript functions intended to call methods. To prepare for the plug-in classes you need to get CDVPlugin Class inherited out of PhoneGap Framework, followed by development of Methods. Note, however, that the iOS platform has no specific name of Method assigned, contrary to the Android platform. See the following for how to prepare coding:

Plug-in Coding Practices

Mapping Plug-in

The JavaScript Function cordova.exe is used to call the Native Functions in iOS, as follows:

```
exec(<successFunction>, <failFunction>, <service>, <action>, [<args>]);
```

Note that the foregoing function calls Native Functions of iOS, as requested by web-view. Refer to the following argument roles:

- successFunction : Callback Function for JavaScript when Native Functions of iOS is called.
- failFunction : JavaScript Callback Function when errors occur while Native Functions of iOS is called.
- service : Name of the plug-in configured in Cordova.plist, the configuration file for PhoneGap.
- action : Name of the method of iOS Native Plug-in Class to be called by JavaScript.
- args : Parameter array required for calling of Native Function of iOS

Register the plug-in that you desire to use by adding strings in Cordova.plist, as follows:

```
<key>service_name</key>
<string>PluginClassName</string>
```

- service_name: Name of plug-in service transferred to cordova.exe
- PluginClassName: Native Class of iOS called by the function cordova.exe

Coding in Java Plug-in for iOS

Coding in Java Plug-in for Android requires inheritance of the class CDVPlugin Refer to the following for how to code in Java Plug-in for Android.

```
#import <Cordova/CDVPlugin.h>

@interface Echo : CDVPlugin

-(void) echo:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options;

@end
```

```
/********* Echo.m Cordova Plugin Implementation *******/

#import "Echo.h"
#import <Cordova/CDVPluginResult.h>

@implementation Echo

-(void) echo:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options
{
	NSString* callbackId =[arguments objectAtIndex:0];

	CDVPluginResult* pluginResult = nil;
	NSString* javaScript = nil;

	@try {
		NSString* echo =[arguments objectAtIndex:1];

if(echo != nil &&[echo length]>0){
			pluginResult =[CDVPluginResult resultWithStatus:CDVCommandStatus_OK
messageAsString:echo];
			javaScript =[pluginResult toSuccessCallbackString:callbackId];
}else{
			pluginResult =[CDVPluginResult resultWithStatus:CDVCommandStatus_ERROR];
			javaScript =[pluginResult toErrorCallbackString:callbackId];
}
} @catch (NSException* exception){
		pluginResult =[CDVPluginResult
resultWithStatus:CDVCommandStatus_JSON_EXCEPTION messageAsString:[exception reason]];
		javaScript =[pluginResult toErrorCallbackString:callbackId];
}

[self writeJavascript:javaScript];
}

@end
```

See the following for how parameters for the Methods intended to implement the plug-in classes:

- arguments : Parameters echoed by JavaScripts intended to call plug-in. Value 0 represents the name of Callback Function for JavaScript.
- options : The Value ending an array of parameters echoed by Javascripts intended to call plug-in. Use of non-deprecated parameters is recommended.