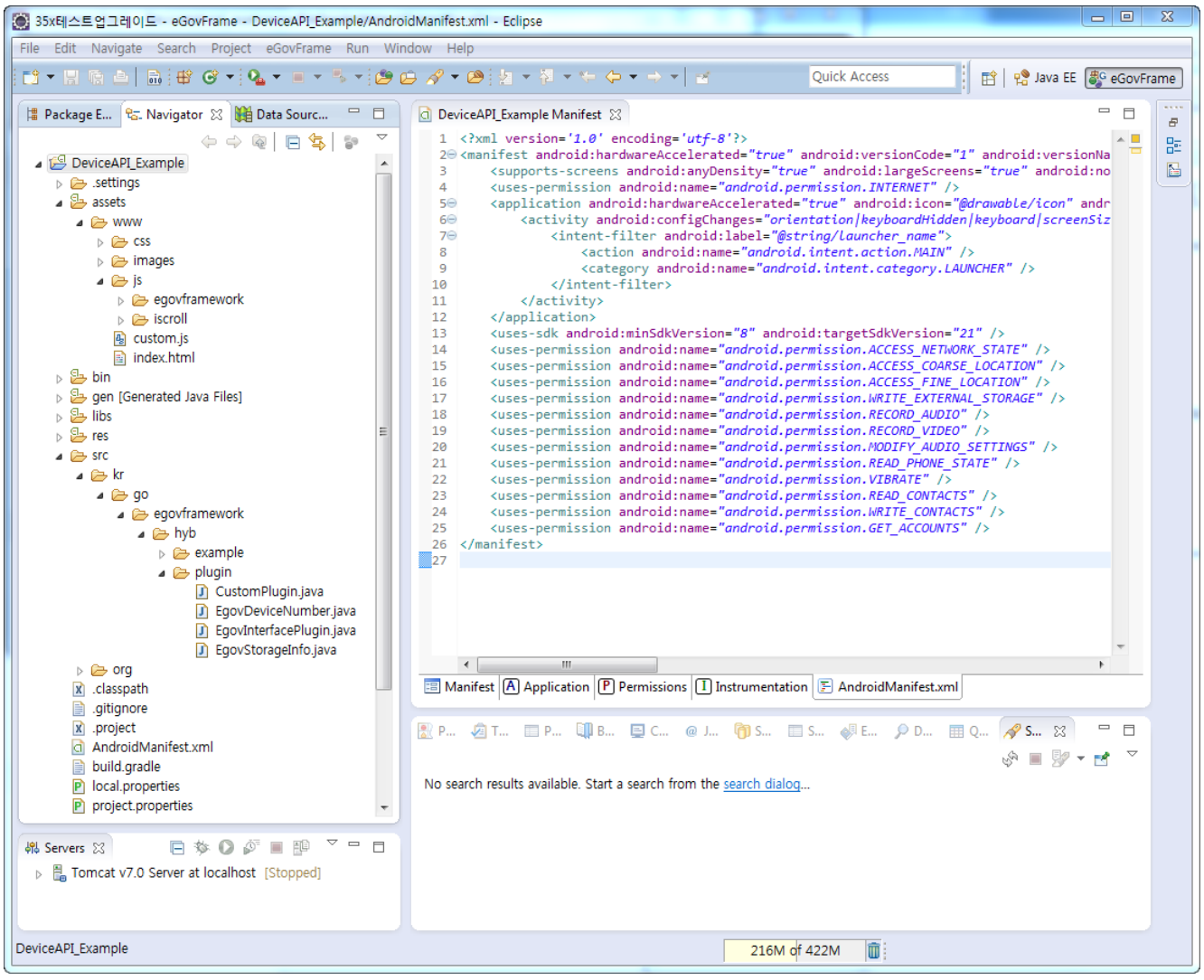
http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:hyb3.5:hrte:%EC%BB%A4%EC%8A%A4%ED%85%80_%ED%94%8C%EB%9F%AC%EA%B7%B8%EC%9D%B8_%EC%98%88%EC%A0%9C

Outline

While using the runtime environment for eGovFramework Device API, you can either add the third-party plug-in or develop customized plug-in for extension. You do not need to develop separate websources for Device API for iOS and Android as they share the same websouce format.

Android Device API

Sample Project Structure



Sources

Category	Type	Title	Remark
Native Source	JAV	kr/go/egovframework/hyb/example/CustomPlu	Native Source for Android Plug-in for PhoneGap

	A	gin.java	Cordova
Web source	JS	assets/www/custom.js	Custom JavaScript for PhoneGap Cordova
Configuration File	X M L	res/xml/config.xml	Configuration File for PhoneGap Cordova

Native

CustomPlugin.java

execute method

- The method Execute is called as interfaced in JavaScript
- In JavaScript, the method Executed is called in the form of cordova.exec(callbackSuccess, callbackFail, “CustomMyPlugin”, “getMessage”, [message]);

```
@Override
public boolean execute(String action, JSONArray args, CallbackContext callbackContext)
throws JSONException {
    if (action.equals(ACTION_ECHO)) {
        String message = args.getString(0);
        this.echo(message, callbackContext);
        return true;
    } else if (action.equals(ACTION_GET_MESSAGE)){
        this.getMessage(callbackContext);
    } else if (action.equals(ACTION_RUN_JAVASCRIPT_FUNCTION)){
        String functionName;
        if (args.length() == 0)
     functionName = "no args";
        else
     functionName = args.getString(0);
        this.runJavaScriptFunction(functionName, callbackContext);
    }
    return false;
}
```

Example of calling WebView JavaScript

- Called out of WebView for one-way processing

```
private void echo(String message, CallbackContext callbackContext) {
    if (message != null && message.length() > 0) {

        AlertDialog.Builder builder = new AlertDialog.Builder(this.cordova.getActivity());
        builder.setMessage(message).setPositiveButton("확인", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {

            }
        });

        AlertDialog dialog = builder.create();
        dialog.show();
```

```
                callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.OK,
message));
                callbackContext.success(message);
            } else {
                callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.ERROR,
"Expected one non-empty string argument."));
                callbackContext.error("Expected one non-empty string argument.");
            }
        }
```

Example of echoing result of WebView JavaScript called

- Called out of WebView for two-way processing

```
    private void getMessage(CallbackContext callbackContext) throws JSONException {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name", "generated out of Android Native");

        callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.OK, jsonObject));
    }
```

Example of calling WebView JavaScript in Native

- Called out of JavaScript Function of WebView in Native

```
    private void runJavaScriptFunction(String functionName, final CallbackContext callbackContext)
throws JSONException {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name", "Calls JavaScript Function in Android Native,
args="+functionName);

        final String javascriptString = "print_message(" + jsonObject.toString() + ")";

        Log.d(this.getClass().getSimpleName(), "=>>> print_message : " + javascriptString);

        this.webView.sendJavascript(javascriptString);

        callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.OK));

    }
```

Web source

custom.js

echo method

- Calls the method Execute out of CustomMyPlugin.java.
- When called, the parameter value is echoed to [message] in the form of Json Structure.
- Also calls webview ⇒ Android native JavaScript Code.

```
function CustomMyPlugin(){}

CustomMyPlugin.prototype.echo = function(message){
    cordova.exec(null, null, "CustomMyPlugin", "echo", [message]);
}
```

getMessage method

- Calls the method getMessage out of CustomMyPlugin.java.
- When called, the parameter value is echoed to [message] in the form of Json Structure.
- When successful, the callback function is processed in the object callbackSuccess function.
- When failed, the callback function is processed in the object callbackFail function.
- Also calls webview ⇒ Android native JavaScript Code ⇒ webview.

```
CustomMyPlugin.prototype.getMessage = function(){
    var callbackSuccess = function(result){
        alert(result.name);
    };

    var callbackFail = function(error){
        alert(error);
    };

    cordova.exec(callbackSuccess, callbackFail, "CustomMyPlugin", "getMessage", []);

}
```

runJavaScript method

- Implements core functions when running CustomMyPlugin.java of out of Native.
- One of the core functions is to call print_message javascript function out of CustomMyPlugin.java.
- Also calls Android Native JavaScript Code ⇒ Webview Function.

```
CustomMyPlugin.prototype.runJavaScript = function(){

        var callbackFail = function(error){
        console.log(error);
        alert(error);
    };

    cordova.exec(null, callbackFail, "CustomMyPlugin", "runJavaScriptFunction", []);

}

function print_message(result){
        console.log("result : "+result);
    alert(result.name);
}
```

Settings

config.xml

- Bound in the form of a JavaScript object with the name as defined in the element Feature.
- The name defined in the element Param refers to the type of the device compatible.
- The value defined in the element Param refers to the native class to be bound.

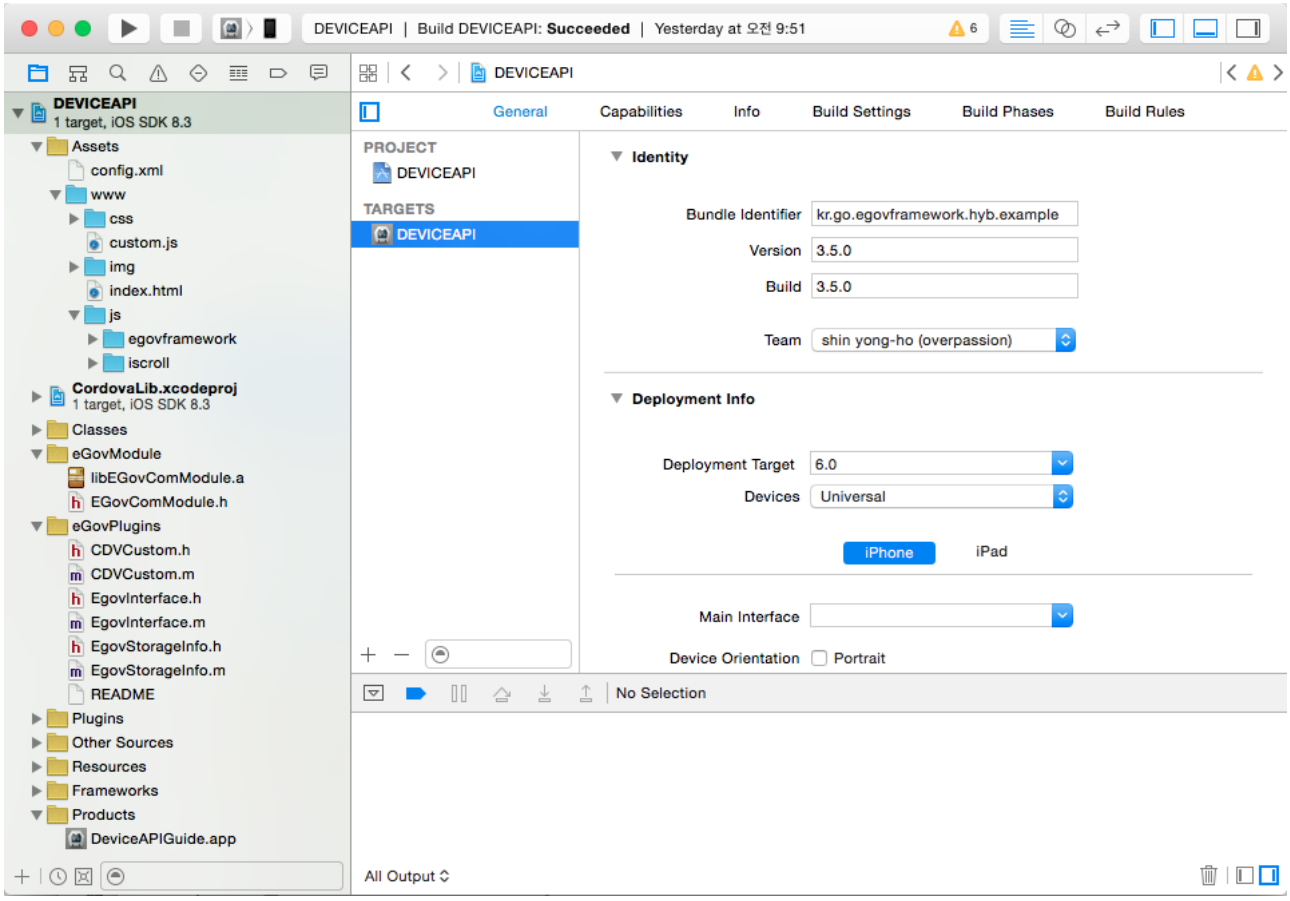
```
<!-- eGovframe DeviceAPI Plug-In-->
<feature name="EgovInterfacePlugin">
<param name="android-package" value="kr.go.egovframework.hyb.plugin.EgovInterfacePlugin" />
</feature>
<feature name="StorageInfoPlugin">
```

<param name="android-package" value="kr.go.egovframework.hyb.plugin.EgovStorageInfo" />
</feature>
<feature name="DeviceNumberPlugin">
<param name="android-package" value="kr.go.egovframework.hyb.plugin.EgovDeviceNumber" />
</feature>

<feature name="CustomMyPlugin">
<param name="android-package" value="kr.go.egovframework.hyb.plugin.CustomPlugin" />
</feature>

iOS Device API

Sample Project Structure



Sources

Category	Type	Title	Remark
Native Source	Objective-C	eGovPlugins/CDVCustom.h	Native Header for iOS Plug-in for PhoneGap Cordova
Native Source	Objective-C	eGovPlugins/CDVCustom.m	Native Source for iOS Plug-in for PhoneGap Cordova
Web source	JS	Assets/www/custom.js	Custom JavaScript for PhoneGap Cordova

Configuration File XML Assets/config.xml Configuration File for PhoneGap Cordova

Native

CDVCustom Class

header file

- Derived out of C/C++, Objective-C requires definition of the methods to be used in the header.
- In Objective-C, an implementation is made by way of inherited CDVPlugin out of PhoneGap Cordova.

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

@interface CDVCustom : CDVPlugin

- (void)echo:(CDVInvokedUrlCommand*)command;
- (void)getMessage:(CDVInvokedUrlCommand *)command;
- (void)runJavasScriptFuncion:(CDVInvokedUrlCommand *)command;

@end
```

Example of calling WebView JavaScript

- Called out of WebView for one-way processing
- In JavaScript, the method Executed is called in the form of cordova.exec(null, null, “CustomMyPlugin”, “echo”, [message]);

```
- (void)echo:(CDVInvokedUrlCommand*)command
{
    NSString *message = [command.arguments objectAtIndex:0];
    [[[UIAlertView alloc] initWithTitle:@"iOS eGovframe" message:message delegate:nil
cancelButtonTitle:@"cancel" otherButtonTitles:@"confirm", nil] show];
}
```

Example of echoing result of WebView JavaScript called

- Called out of WebView for two-way processing
- In JavaScript, the method Executed is called in the form of cordova.exec(callbackSuccess, callbackFail, “CustomMyPlugin”, “getMessage”, []);

```
- (void)runJavasScriptFuncion:(CDVInvokedUrlCommand *)command
{
    NSDictionary *jsonInfo = @{Called JavaScript function out of @"name": @"iOS native"};

    NSError *error;
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonInfo
options:NSJSONWritingPrettyPrinted error:&error];
    CDVPluginResult *pluginResult = nil;

    if (!error) {
        NSString *jsonString = [[NSString alloc] initWithData:jsonData
encoding:NSUTF8StringEncoding];
        NSString *javaScriptString = [NSString stringWithFormat:@"print_message(%@)",
jsonString];
```

```
        [self.webView stringByEvaluatingJavaScriptFromString:javaScriptString];

        pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_OK];
    } else {
        pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_ERROR
messageAsString:[error localizedDescription]];
    }

    [self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];

}
```

Example of calling WebView JavaScript in Native

- Called out of JavaScript Function of WebView in Native
- In JavaScript, the method Executed is called in the form of cordova.exec(null, callbackFail, “CustomMyPlugin”, “runJavasScriptFuncion”, []);

```
    private void runJavaScriptFunction(String functionName, final CallbackContext callbackContext)
throws JSONException {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name", "Calls JavaScript Function in Android Native,
args="+functionName);

        final String javascriptString = "print_message(" + jsonObject.toString() + ")";

        Log.d(this.getClass().getSimpleName(), "=>>> print_message : " + javascriptString);

        this.webView.sendJavascript(javascriptString);

        callbackContext.sendPluginResult(new PluginResult(PluginResult.Status.OK));

    }
```

Web source

custom.js

- iOS shares a single js file with Android. The same applies to html, css and other web sources.

echo method

- Calls the method Execute out of CDVCustom.m.
- When called, the parameter value is echoed to [message] in the form of Json Structure.
- Also calls webview ⇒ iOS native JavaScript Code.

```
function CustomMyPlugin(){}

CustomMyPlugin.prototype.echo = function(message){
    cordova.exec(null, null, "CustomMyPlugin", "echo", [message]);
}
```

getMessage method

- Calls the method getMessage out of CDVCustom.m.
- When called, the parameter value is echoed to [message] in the form of Json Structure.

- When successful, the callback function is processed in the object callbackSuccess function.
- When failed, the callback function is processed in the object callbackFail function.
- Also calls webview ⇒ iOS native JavaScript Code ⇒ webview.

```
CustomMyPlugin.prototype.getMessage = function(){
	var callbackSuccess = function(result){
		alert(result.name);
	};

	var callbackFail = function(error){
		alert(error);
	};

	cordova.exec(callbackSuccess, callbackFail, "CustomMyPlugin", "getMessage", []);

}
```

runJavaScript method

- Implements core functions when running CDVCustom.m out of JavaScript.
- One of the core functions is to call print_message javascript function out of CDVCustom.m.
- Also calls iOS Native JavaScript Code ⇒ Webview Function.

```
CustomMyPlugin.prototype.runJavaScript = function(){

		var callbackFail = function(error){
		console.log(error);
		alert(error);
	};

	cordova.exec(null, callbackFail, "CustomMyPlugin", "runJavaScriptFunction", []);

}

function print_message(result){
		console.log("result : "+result);
	alert(result.name);
}
```

Settings

config.xml

- Bound in the form of a JavaScript object with the name as defined in the element Feature.
- The name defined in the element Param refers to the type of the device compatible.
- The value defined in the element Param refers to the native class to be bound.

```
<!-- eGovframe DeviceAPI Plug-In-->
<feature name="StorageInfoAPI">
<param name="ios-package" value="EgovStorageInfo"/>
</feature>
<feature name="InterfaceAPI">
<param name="ios-package" value="EgovInterface"/>
</feature>

<feature name="CustomMyPlugin">
```

<param name="ios-package" value="CDVCustom" />
</feature>