

# 전자정부 모바일 표준프레임워크 실행환경(Device API)

**eGovFrame**

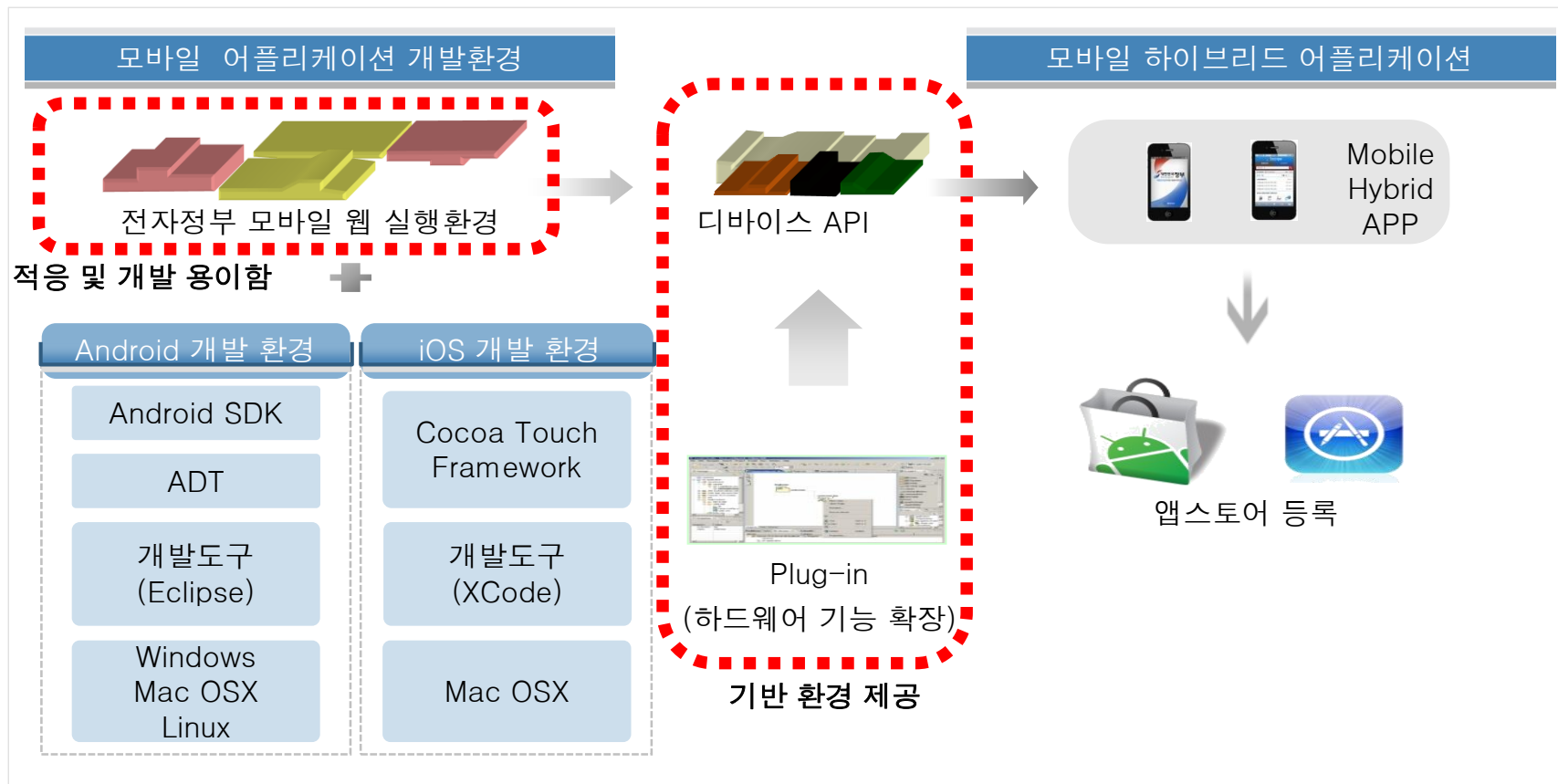


# Contents

1. \_ 모바일 하이브리드 어플리케이션
2. \_ Device API



- 전자정부 디바이스 API 실행환경은 기존 모바일 어플리케이션 개발 환경에서 Plug-in 형태로 구현되어 있는 디바이스 API를 추가하여 모바일 하이브리드 어플리케이션을 구현하는 방식이다.



### □ 네이티브앱

- 경쟁이 치열한 분야의 앱
- 사용자 경험(UX)가 우수해야 할때
- 하드웨어 기능에대한 의존도가 높을때

### □ 하이브리드앱

- 기간이 충분하지 않을때 ( 1개의 소스로 iOS,Android 지원 )
- 사업비가 충분하지 않을때 ( 개발기간 단축 및 웹개발 기술 주로 사용 )
- 화면이 많은경우 대응이 유리 ( 웹개발을 기본으로 하기때문에 분업및 빠른 화면 개발 용이 )
- 웹개발자는 충분한데, 네이티브 대응인력이 충분하지 않을때 ( 기존 웹기술에 HTML5 확장요소 사용)
- 트래픽이 비교적 많지 않을때 ( 웹리소스를 앱에 포함한다음 배포하여 해결 가능 )

### ❑ 모바일 하이브리드 어플리케이션 구현을 위한 하이브리드 프레임워크 기반 제공

- 디바이스 고유기능 호출을 위한 Native Code 및 JavaScript Code 제공

### ❑ HTML 페이지에서 디바이스 고유기능의 호출을 위한 디바이스 API 제공

- 24종의 오픈 소스 디바이스 API 제공
- Accelerator, Compass, Camera, Media, File Read/Write, Device, Vibrator, Network, GPS, Contact 등
- PushNotifications, FileOpener, StreamingMedia, Barcodescanner, WebResourceUpdate, DeviceFileManage, JailbreakDetection, SocketIO, SQLite, UnZip ( DeviceAPI v3.6 이상 )

### ❑ 플러그인 형태의 추가 API 제공

- NPAPI 네이티브 코드와 연계를 위한 JavaScript 형태의 연계 템플릿 API 제공
- 기존의 전자정부 표준프레임워크로 구현 된 웹 서버 어플리케이션과의 연계를 위한 연계 API 제공
- apache.org 제공 플러그인 확장 ( <https://cordova.apache.org/plugins/> )

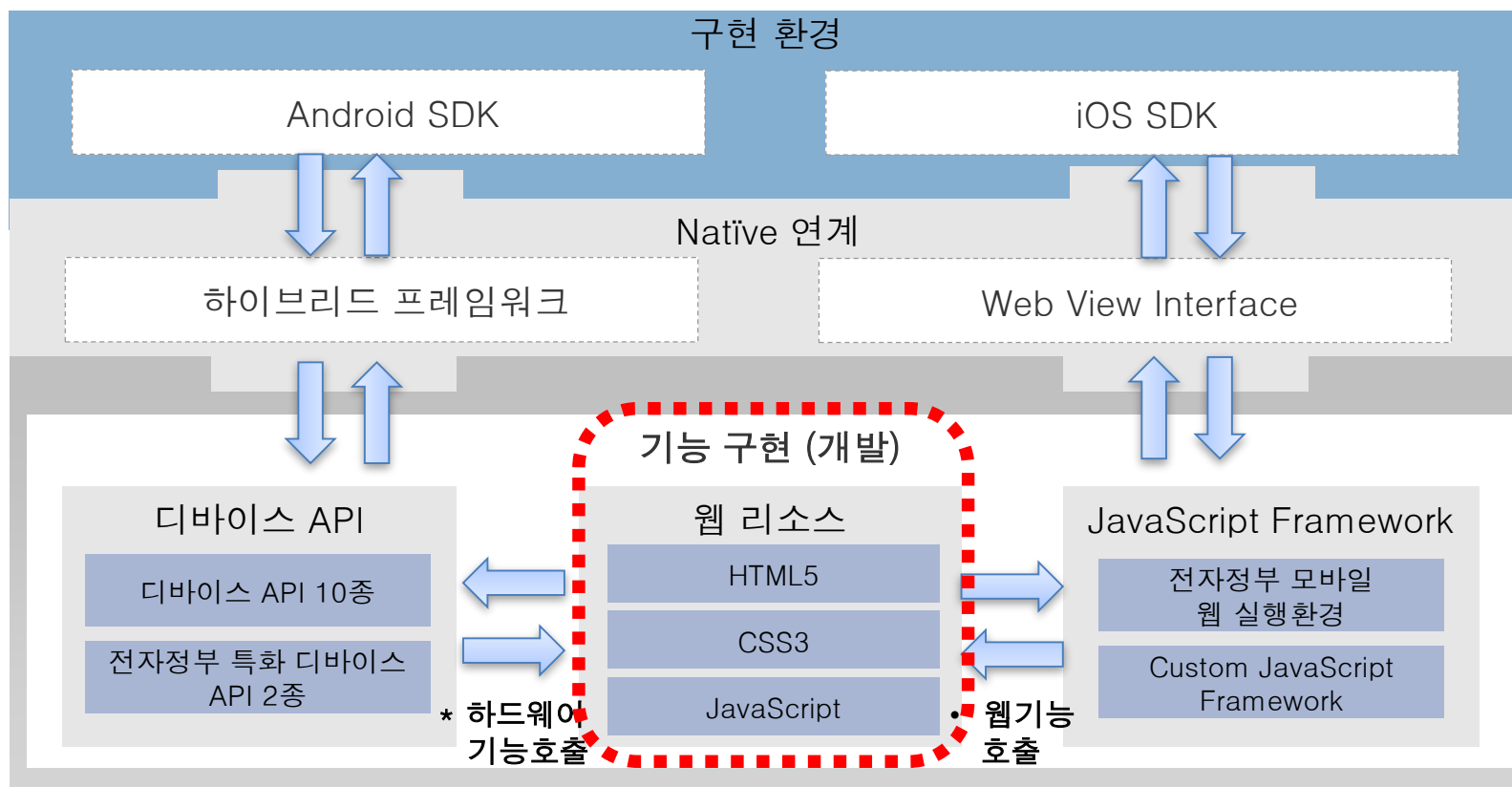
### ❑ 하이브리드 어플리케이션 개발을 위한 오픈 소스 소프트웨어 선정

- 전자정부 모바일 하이브리드 어플리케이션은 PhoneGap(Cordova) 프레임워크를 기반으로 하며, HTML5 기반의 화면 구성을 위하여 전자정부 모바일웹 실행환경을 적용

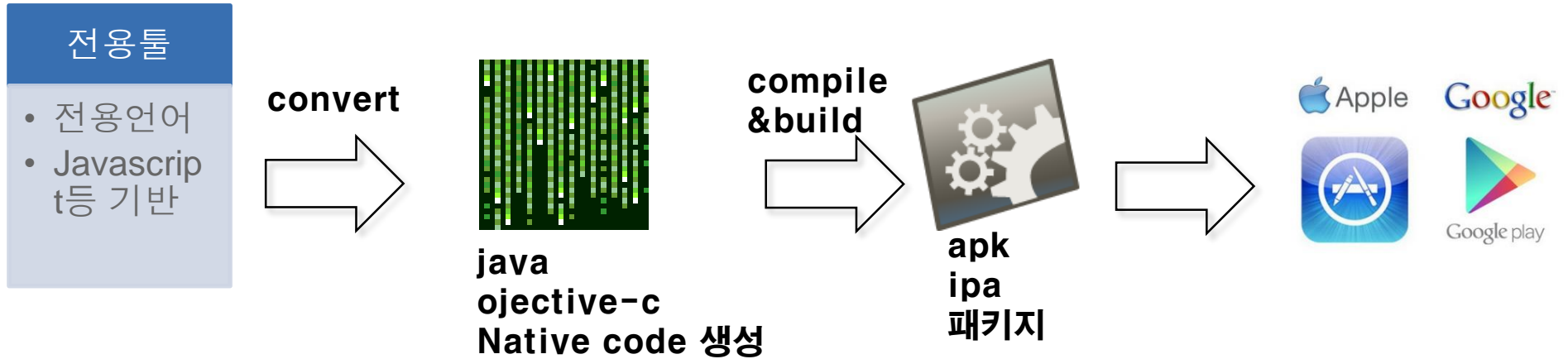
□ 전자정부 디바이스 API 실행환경은 하이브리드 어플리케이션의 구성 기반이 되며, 하이브리드 어플리케이션 구현 및 실행 시 필요한 기본기능을 제공한다.

환경	설명
실행 환경	<ul style="list-style-type: none"> <li>전자정부 모바일 하이브리드 어플리케이션 개발 프레임워크 라이브러리 제공                             <ul style="list-style-type: none"> <li>- Open Source(PhoneGap/Cordoda)</li> <li>- 표준 코드 및 하이브리드 앱 샘플 템플릿 제공</li> </ul> </li> <li>Device API 지원                             <ul style="list-style-type: none"> <li>- 12종의 Open Source Device API</li> <li>- 추가 Device API 10종 제공 ( version 3.6 이상 )</li> </ul> </li> </ul>
모바일 디바이스 API 가이드 프로그램	<ul style="list-style-type: none"> <li>모바일 하이브리드 어플리케이션 개발시 가이드 및 재사용을 위한 디바이스 API 가이드 프로그램 제공                             <ul style="list-style-type: none"> <li>- 총 48종의 디바이스 API 가이드 하이브리드 앱 개발 (안드로이드 24종, iOS 24종)</li> <li>- 디바이스 API 가이드 하이브리드 앱과 통신을 위한 ( MobileApp + HTML/JS/CSS + Restfull Server )</li> </ul> </li> <li>전자정부 표준프레임워크 기반 웹 서버 어플리케이션 개발</li> </ul>
개발 환경	<ul style="list-style-type: none"> <li>플랫폼 별 하이브리드 앱 개발 템플릿 프로젝트 생성 도구 제공</li> </ul>

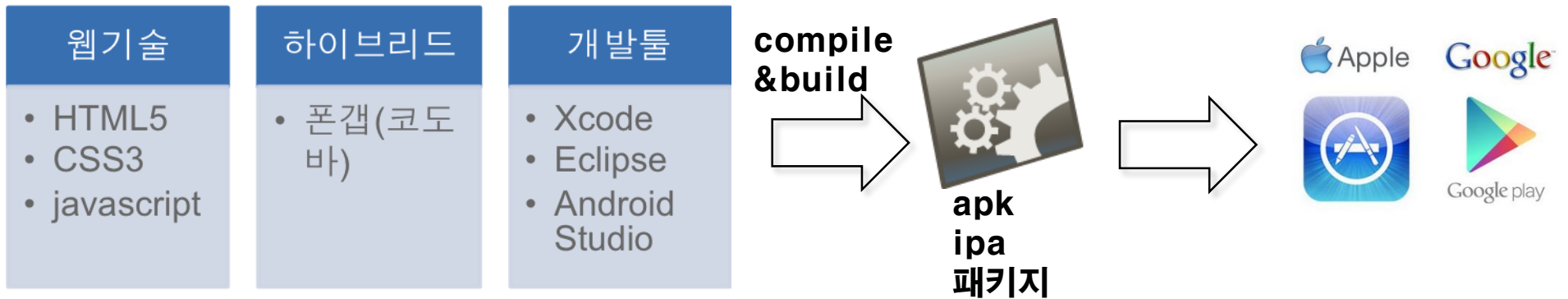
- 디바이스 API 실행환경은 웹 리소스를 통한 디바이스 하이브리드 어플리케이션 구현을 지원하며 플랫폼 별 SDK를 활용하여 구현된 웹 리소스 내의 JavaScript 형태의 Device API와 각 플랫폼 별 Native Code가 하이브리드 프레임워크 및 웹 뷰 인터페이스를 통해 연동되어 실제 디바이스의 고유 기능을 호출할 수 있도록 지원 한다.



### □ 네이티브 코드 생성 방식



### □ 아키텍처 조합 방식 (모바일전자정부 디바이스API)



\* 웹기술을 최대한 활용하여 개발  
(학습요소 최소화 및 빠른 초기개발)



# 모바일 전자정부 디바이스 API(하이브리드)의 장점

## □ 하이브리드 어플리케이션 개발을 위한 오픈 소스 소프트웨어

- 무료 사용
- 기본 환경 및 예제 샘플 제공
- 모바일 프로젝트 수행시 개발표준 수립 용이 및 빠른 초기 개발
- 기존 웹기술의 확장으로 하이브리드앱 개발 가능 (개발자 재교육 최소화)
- 웹기반을 응용하므로 구성원간에 분업에 용이함

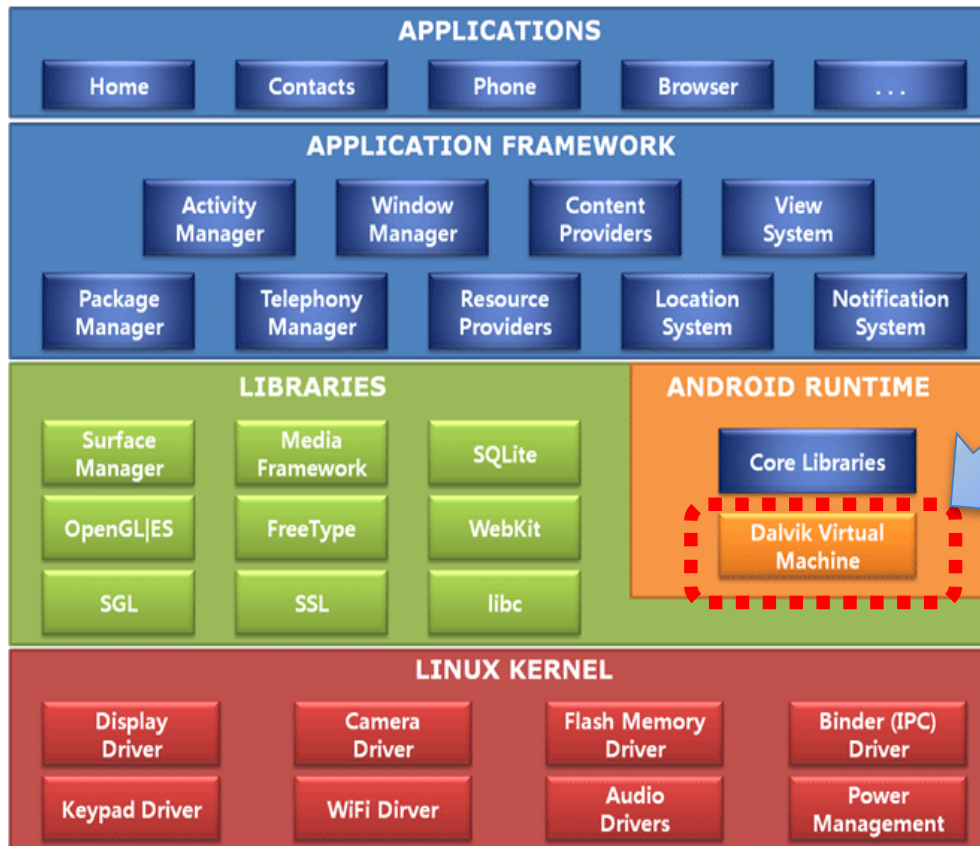
**HTML** : 웹디자이너

**CSS** : 퍼블리셔

**JS 및 Servlet java** : 웹개발자

\* 네이티브앱으로 개발시에는 개발자가 디자인 구현에 어느정도 작업공수를 소비해야함. (개발자 부담 가중)

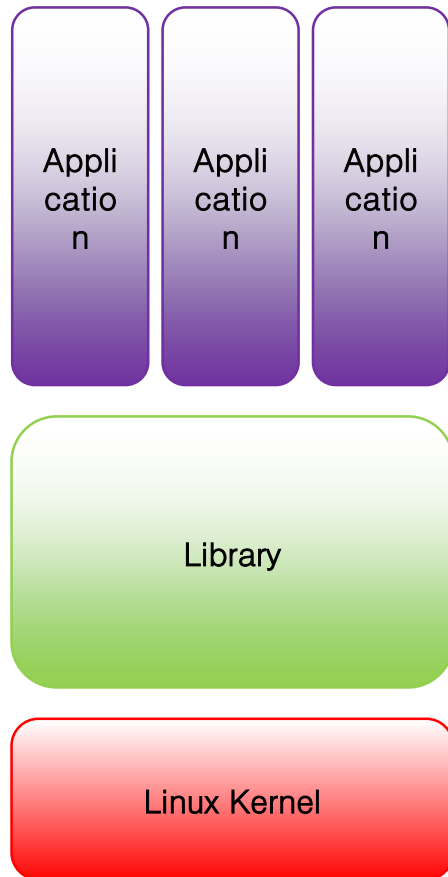
□ Android 운영체제의 구성은 다음과 같다.



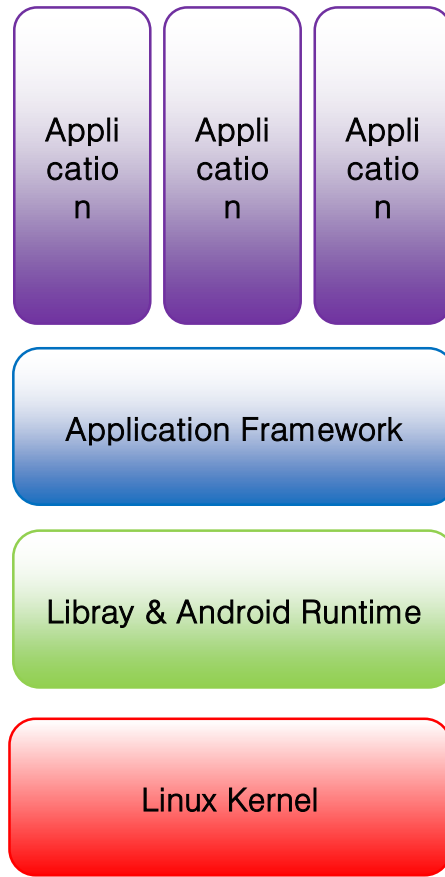
- **Linux kernel**
  - 하드웨어 드라이버, 프로세스와 메모리 관리, 보안, 네트워크, 전력 관리 등의 핵심 서비스 담당.
- **Libraries**
  - Android libc와 SSL 같은 다양한 C/C++ 코어 라이브러리
  - 핸드폰에 사용되는 하드웨어를 지원하기 위해 컴파일되어 핸드폰 공급업체에 의해 핸드폰에 미리 설치됨
- **Android Runtime**
  - Core Libraries, “Dalvik Virtual Machine”으로 구성.
  - 커널위에 존재하며 Dalvik, VM, 코어 라이브러리 등이 포함된다.
- **Application Framework**
  - Android Application을 만드는데 필요한 기능을 지원, App들을 관리하는 역할을 한다.
  - 직접만든 컴포넌트를 통해 확장 시킬 수 있다.
- **Application**
  - 안드로이드 아키텍처 다이어그램의 최상위 계층
  - 안드로이드의 특징 중 하나로 모든 어플리케이션이 동일한 수준으로 실행됨

❑ Linux와 Android OS의 계층구조 차이.

### Linux OS

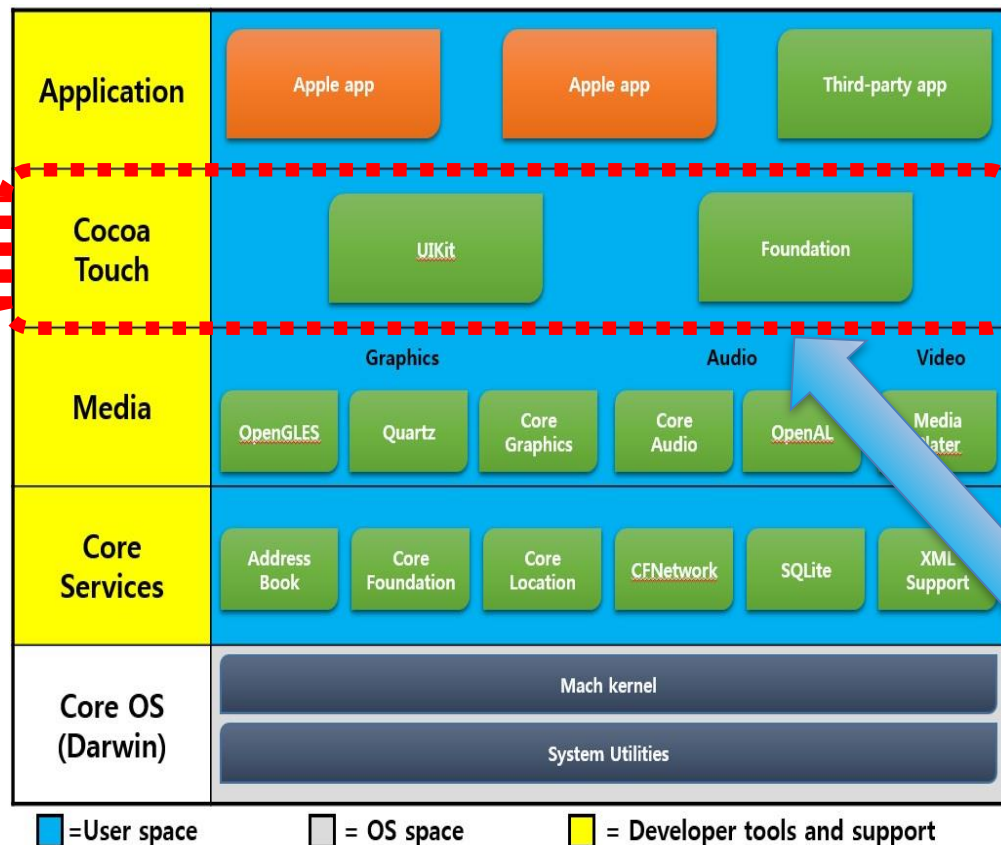


### Android OS



- Linux kernel
  - GPL 2.0 ( 수정시 소스 공개 의무 )
- Libraries
  - 안드로이드의 경우 Apache 라이선스를 유지하기 위해 GNU라이브러리의 경우 재개발 및 대체함.
  - Kernel을 제외한 구글이 배포하는 OS부분은 Apache License로 소스공개의 의무가 없음.
  - 상호 호환되지 않음.
  - C/C++
- Android Runtime
  - Dalvik Virtual Machine (or ART)으로 모바일에 최적화 ( 메모리, CPU, 배터리 사용등 )
- Application Framework
  - Activity Manager, View System, Window Manager, Package Manger, Hardware Services
  - Java
- Application
  - 상호 다른 라이브러리를 사용하므로 호환되지 않음.

□ iOS 운영체제의 구성은 다음과 같다.



### • iOS의 커널

- Mac OS X와 같은 Mach에 기초한다.

### • Core OS와 Core Services 계층

- 매우 기본 적인 iOS의 인터페이스를 가지고 있다. 데이터 타입들, 봉주르 서비스, 네트워크 소켓 등이 있다.

### • Media 계층

- 2D/3D 그리고, 오디오, 비디오 등의 기반 기술을 가지고 있다. OpenGL ES, Quartz Core Audio와 Core Animation이 있다.

### • Cocoa Touch 계층

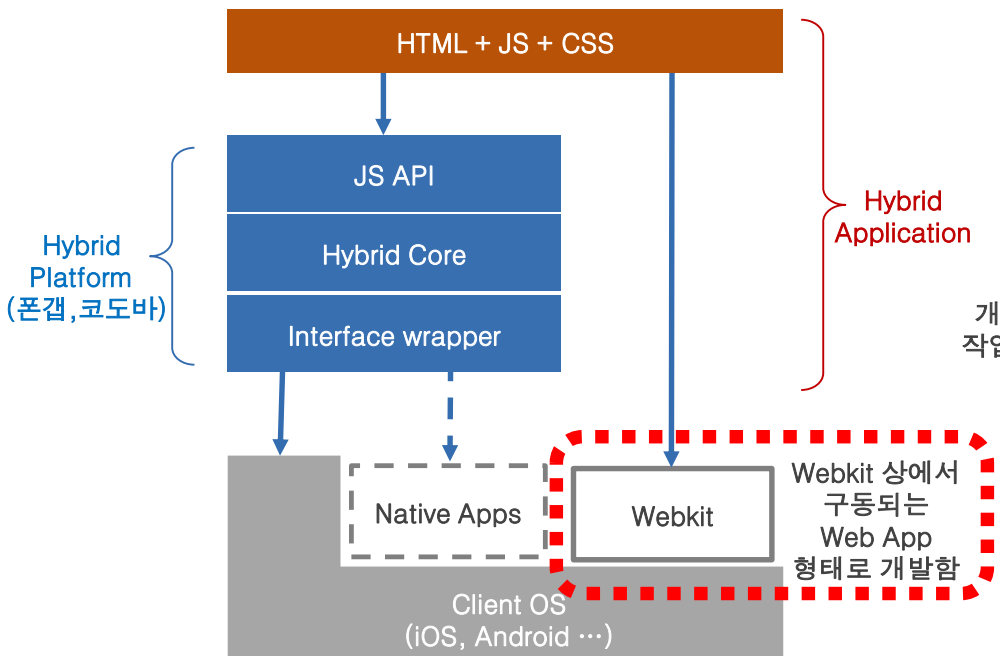
- 모든 기술이 C++ 및 Objective-C를 기본으로 하며, 각종 프레임워크로 응용프로그램을 만들 때 가장 기본적인 인프라를 제공한다.

### • Application

- Android Java와 달리 컴파일되면 이진코드의 기계어 코드가 생성되며 Objective-C로 개발되었으나 최근 간결한 문법과 안정성이 좋은 Swift로도 개발한다.

- 모바일 디바이스 API 실행환경 기반 프레임워크는 하이브리드 어플리케이션을 구현하기 위한 기반 환경으로서 웹 리소스로 구현 된 기능을 네이티브 앱 형태로 래핑하는 역할을 한다.

### Hybrid Application의 구조

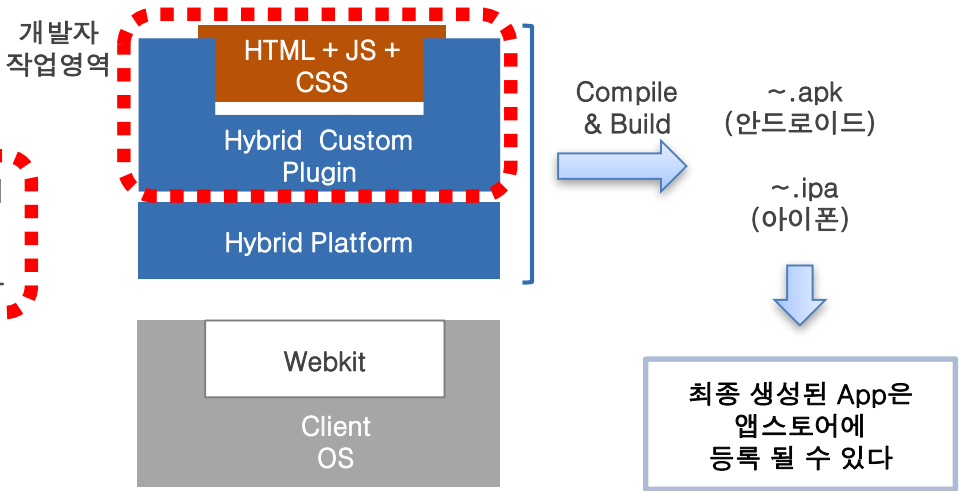


Application 은 Hybrid Platform에서 제공되는 JS를 통해 Native Feature를 이용한다.

### Client OS 상의 Hybrid Application의 Package 형태

Application 개발자는 Web 개발 방식으로 HTML, JS, CSS 를 사용하여 개발한다.

Hybrid Platform으로 최종 Build 된 Hybrid Application은 Native Application과 같은 모습으로 Package 되고, 관리된다.

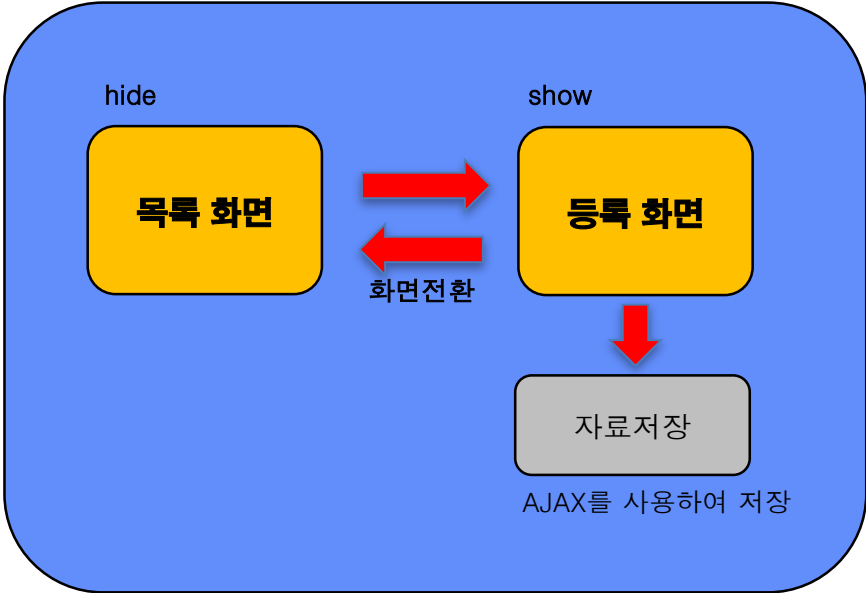


- ❑ Single Page Application은 업무단위로 여러 개의 화면을 포함 하고 한페이지내에서 화면단위로 이동하여 처리된다. (화면이동및 분업이 용이)
- ❑ Round Trip Application은 웹개발에서 고전적인 방식이며 화면 전환시 파라미터를 전달하고 페이지 이동을 한다.

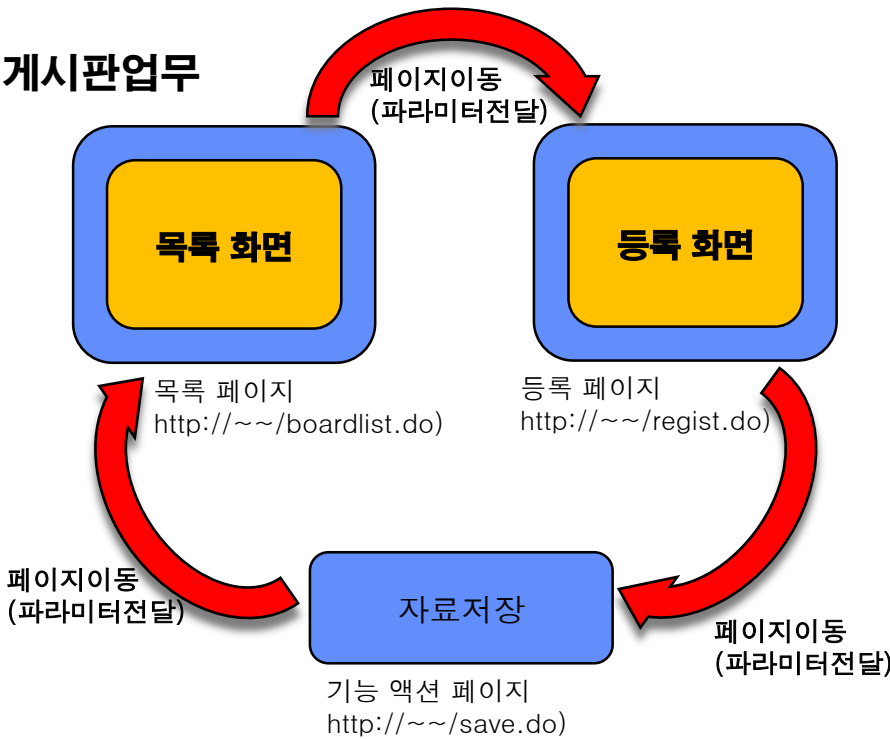
Single Page Application

Round Trip Application

게시판업무 (http://~~/~board.html)



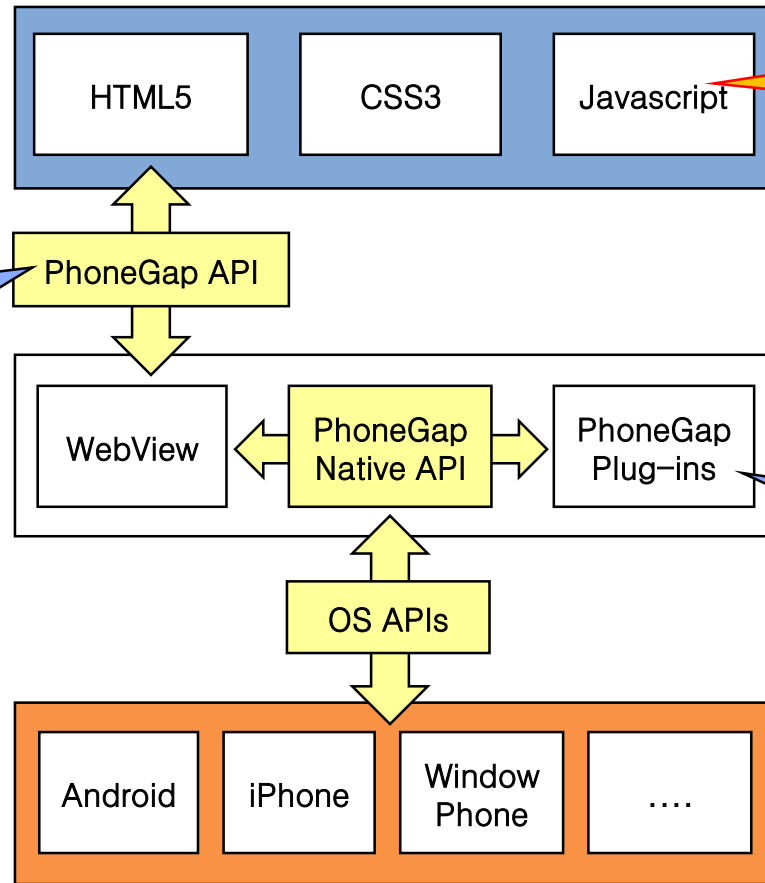
게시판업무



- 디바이스 API 실행환경의 주요 구성요소 인 하이브리드 어플리케이션 프레임워크의 구성은 다음과 같다.

\* 각 언어의 장점을 연결하여 여러 언어를 사용하는것은 현재의 추세

Java , Objective-C등 네이티브언어로 커스토폴마이징 가능

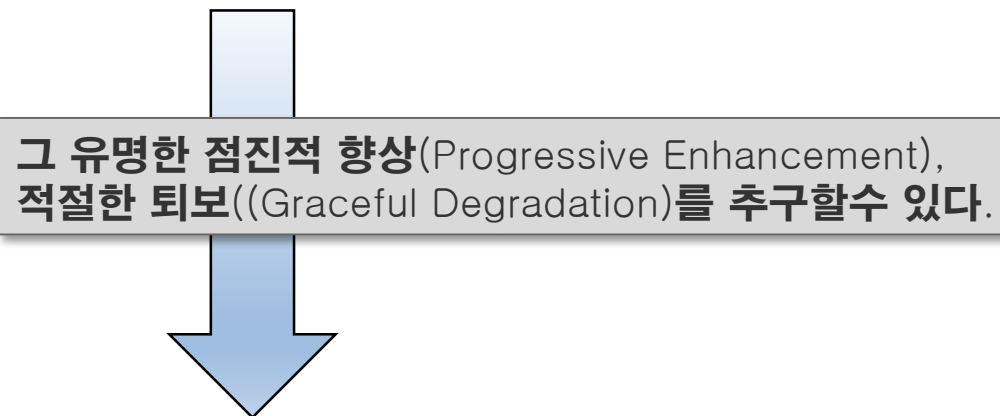


\* Javascript등 주로 웹기술을 사용하여 개발

플러그인으로 기능 확장 (전자정부 플러그인 제공)

- 아이폰 OS 및 안드로이드 OS 외에도 다양한 OS를 지원하여 크로스플랫폼 지원 앱을 개발하는데 유리하다.

- Android
- iOS
- Amazon-FireOS
- BlackBerry10
- FirefoxOS
- Ubuntu
- Mac OS X
- Windows Phone 8.1
- Windows 8
- Windows 10



그 유명한 점진적 향상(Progressive Enhancement),  
적절한 퇴보((Graceful Degradation)를 추구할수 있다.

- 하이브리드 기반을 제공하는 Apache Cordova가 비교적 빠른속도로 발전하고 있음
- Linux OS , Android OS를 이용한 Embedded기기(셋톱박스,발권기,안내기기 등등)에서 HTML5 사용이 늘어나고 있고 기기와 UI Interface 수요가 증가하고 있음.

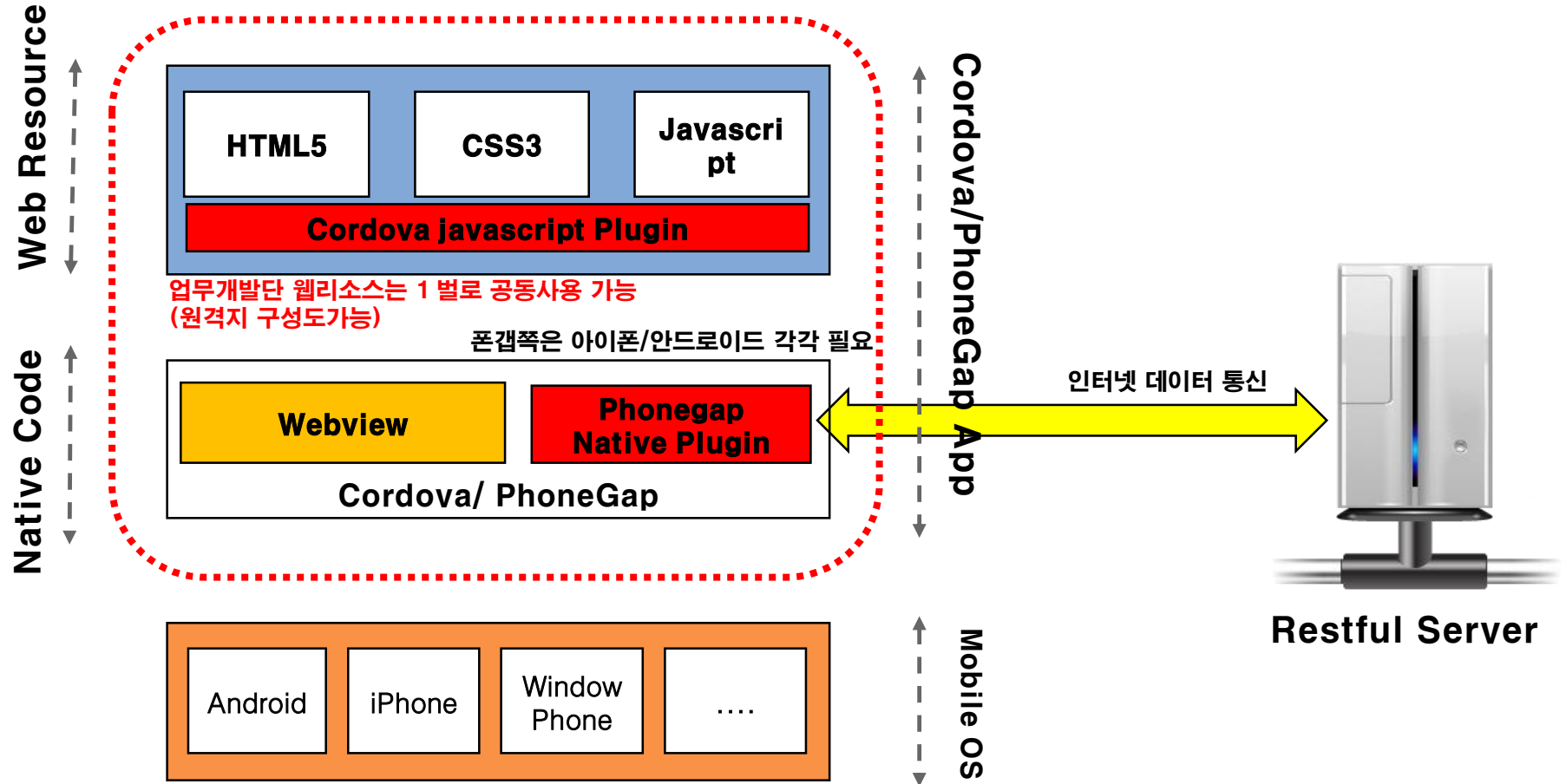


□ 폰갭은 Device API 제어를 위해 미리 구현 한 JavaScript 및 SDK 연계를 위한 Native Code 외에 추가기능을 위해 구현 된 Plug-In을 구성하고 있는 폰갭 Custom Plug-In 과 폰갭 Custom Native Plug-In이 추가된 구조이다.

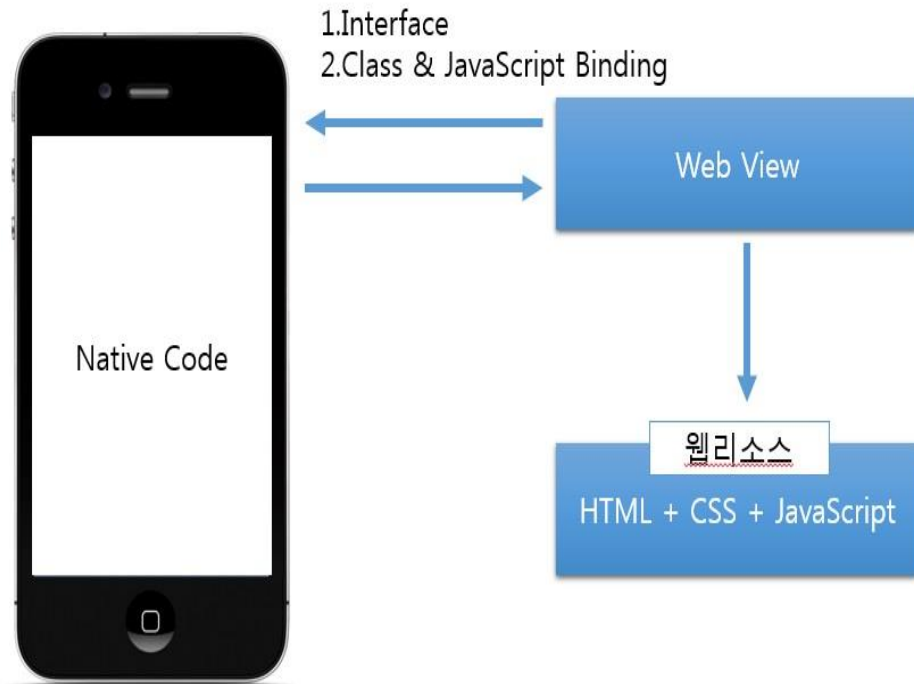
구분	설명
Mobile Web Application	<ul style="list-style-type: none"> <li>HTML, CSS, 사용자 정의 JavaScript</li> </ul>
PhoneGap JavaScript Engine	<ul style="list-style-type: none"> <li>Device API 제어를 위해 JavaScript로 구현되어 제공</li> <li>PhoneGap.js</li> </ul>
PhoneGap Native Engine	<ul style="list-style-type: none"> <li>플랫폼 별 WebView를 상속받아 Mobile Web Application과 연계를 위한 Native Code</li> <li>PhoneGap.jar</li> <li>phoneGap.framework</li> </ul>
PhoneGap Custom Plugin	<ul style="list-style-type: none"> <li>기능 확장을 위해 추가 된 Plug-In을 위한 JavaScript 코드</li> <li>사용자 정의 JavaScript Plug-In 을 포함</li> <li>플랫폼 별로 플러그인을 위한 Native 코드 구성이 상이함에 따라 JavaScript 구성 또한 다를 수 있음</li> </ul>
PhoneGap Custom Native Plugin	<ul style="list-style-type: none"> <li>기능 확장을 위해 추가 된 Plug-In을 위한 플랫폼 Native 코드</li> </ul>

\* Plugin 확장시 jar와 javascript간 1:1 로 구성  
 • egovframe.go.kr DeviceAPI 실행환경 샘플 DeviceAPIGuide\_~~~.zip 예제 참조

- 모바일 전자정부프레임워크중 디바이스 API 전체 아키텍처 구조는 다음과 같다.



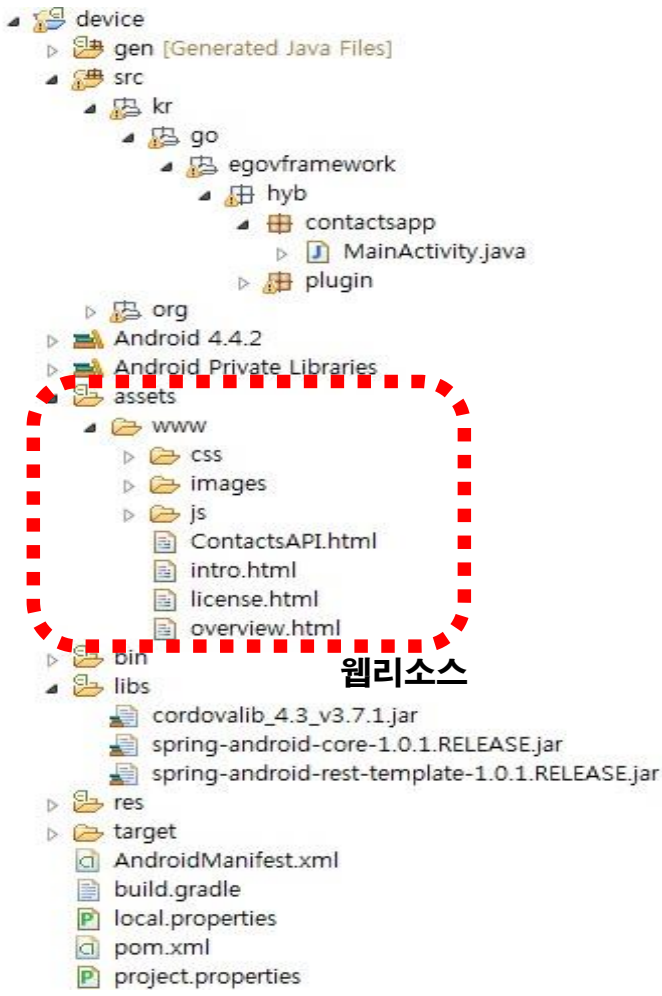
- ❑ 폰갭은 플랫폼 별 SDK내에 내장 되어있는 Web View를 활용하여 하이브리드 앱 실행 시 사용자가 구현 한 HTML 파일을 출력하고 JavaScript 소스를 실행한다.
- ❑ 하이브리드 어플리케이션의 디바이스 API 호출을 위한 브릿지 역할을 해준다.



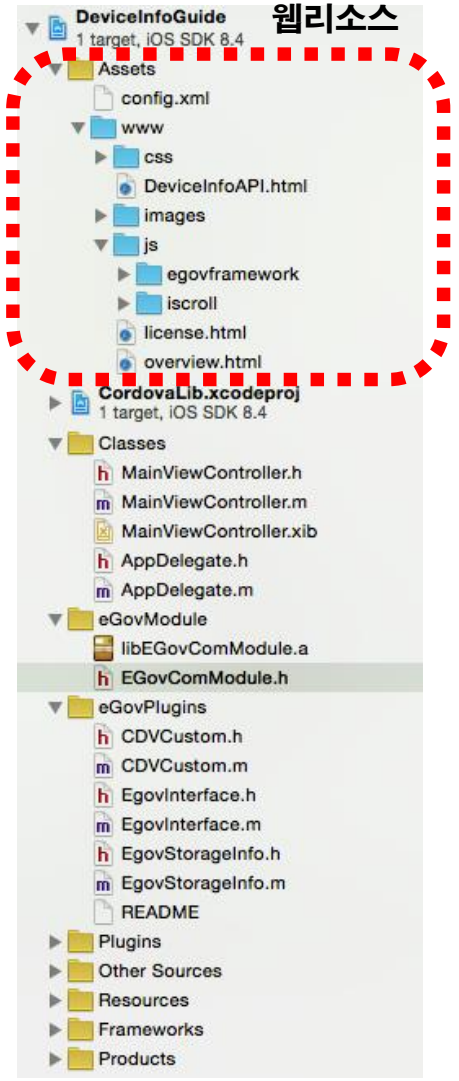
- SDK 로 부터 상속받은 웹 뷰를 생성해서 웹 뷰 내에 소스를 로드 한 후 네이티브API를 사용할 수 있도록 연계
- Interface Module (BrowserControl <-> Device API)
  - iPhone
    - shouldStartLoadWithRequest
    - stringByEvaluationJavaScriptFromString
  - [gap://] protocol catch
  - Android
    - addJavascriptInterface
  - Class & JavaScript Binding

❑ 디바이스 API 실행환경을 적용한 하이브리드 앱 프로젝트를 생성하면 다음과 같이 네이티브 코드, 웹 리소스, 네이티브 라이브러리, 환경 설정 파일로 구성된다.

Android App  
eclipse  
Project



iOS App  
XCode  
Project



- ❑ Android Platform 기반 하이브리드 앱의 메인 클래스 (Activity)는 Phonegap 의 CordovaActivity 클래스를 상속받아 웹 리소스를 loadUrl 형태로 읽어오는 방식이다.

```
package kr.go.egovframework.hyb.cameraapp; MainActivity.java

import android.os.Bundle;
import android.webkit.WebSettings;

import org.apache.cordova.*;

public class MainActivity extends CordovaActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.init();

        super.appView.getSettings().setAppCacheEnabled(false);
        super.appView.getSettings().setCacheMode(WebSettings.LOAD_NO_CACHE);
        super.appView.clearCache(true);
        super.appView.getSettings().setJavaScriptEnabled(true);
        super.appView.getSettings().setDomStorageEnabled(true);

        // Set by <content src="index.html" /> in config.xml
        loadUrl(launchUrl);
    }
}
```

- iOS Platform 기반 하이브리드 앱의 메인 클래스 (MainViewController)는 Phonegap의 CDVViewController를 상속받아 config.xml의 url을 호출하며 주요 코드는 다음과 같다.

```
#import <Cordova/CDVViewController.h>
#import <Cordova/CDVCommandDelegateImpl.h>
#import <Cordova/CDVCommandQueue.h>

@interface MainViewController : CDVViewController

@end

@interface MainCommandDelegate : CDVCommandDelegateImpl
@end

@interface MainCommandQueue : CDVCommandQueue

@end;
```

**MainViewController.h**

```
#import "MainViewController.h"

@implementation MainViewController

- (id)initWithNibName:(NSString*)nibNameOrNil bundle:(NSBundle*)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {

    }
    return self;
}

- (id)init
{
    self = [super init];
    if (self) {

    }
    return self;
}
```

**MainViewController.m**

- 어플리케이션의 주요 로직을 담고 있는 웹 리소스(HTML5)는 HTML, JavaScript, CSS로 구성되며 플랫폼에 비종속적으로 iOS와 Android 플랫폼 간 재사용이 가능하다.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=yes" />
  <title>Network API Guide</title>

  <!-- eGovFrame Common import -->
  <link rel="stylesheet" href="css/egovframework/mb/cmm/jquery.mobile-1.4.5.css" />
  <link rel="stylesheet" href="css/egovframework/mb/cmm/EgovMobile-1.4.5.css" />
  <script type="text/javascript" src="js/egovframework/mb/cmm/jquery-1.11.2.min.js"></script>

  <script type="text/javascript" src="js/egovframework/mb/cmm/jquery.history.js"></script>

  <script type="text/javascript" src="js/egovframework/mb/cmm/jquery.mobile-1.4.5.min.js"></script>
  <script type="text/javascript" src="js/egovframework/mb/cmm/EgovMobile-1.4.5.js"></script>
  <script type="text/javascript" src="js/egovframework/mb/cmm/EgovHybrid.js"></script>

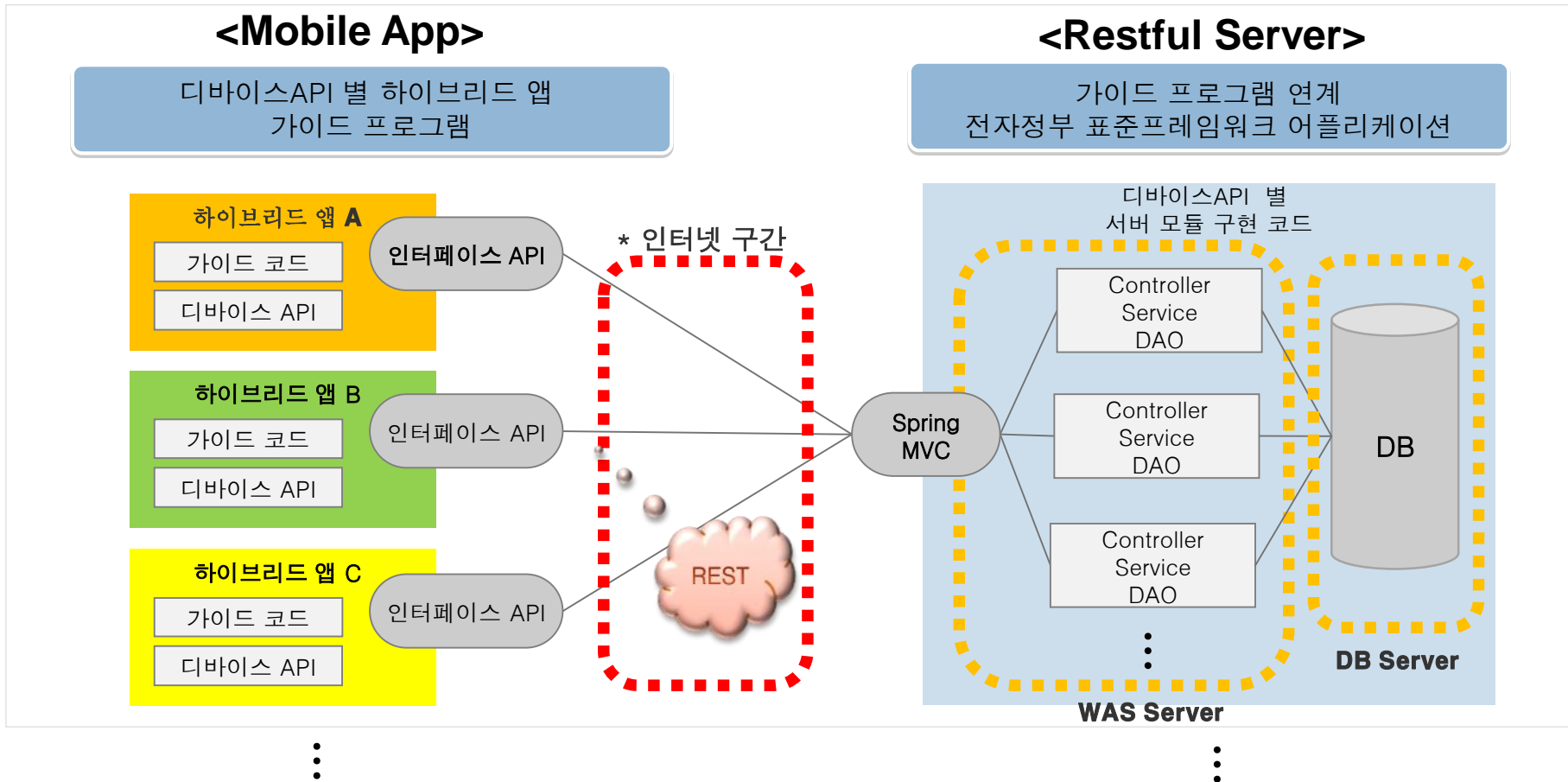
  <link rel="stylesheet" href="css/egovframework/mb/hyb/guide.css" />

  <!-- iScroll.js import -->
  <script type="text/javascript" src="js/iscroll/iscroll.js"></script>

  <!-- Phonegap.js import -->
  <script type="text/javascript" charset="utf-8" src="js/egovframework/mb/cmm/cordova.js"></script>

  <script type="text/javascript">
</script>
</head>
```

- 각 디바이스API별 특성에 따라 서버 모듈 연계 기능을 쉽게 이해하여 활용할 수 있도록 디바이스 API 가이드 프로그램을 개발환경을 통해 제공한다.



\* 교육교재 lab403-DeviceAPI 프로젝트 참조

\* 교육교재 lab403-DeviceAPI-Web 프로젝트 참조



❑ 디바이스 API 실행환경은 추가 플러그인 및 UX 컴포넌트 활용을 위해 폰갭 프레임워크 이외에도 다음과 같은 오픈 소스 라이브러리가 활용 되었다.

<div>Spring For Android</div> <div>(Mobile App)</div>	<ul style="list-style-type: none"> <li>• 전자정부 서버 어플리케이션과 Rest 통신을 위한 안드로이드 용 네이티브 Restful 서비스</li> <li>• 오픈 소스 라이브러리로 Interface Device API(Android) 개발에 사용 되었다.</li> </ul>
<div>ASIHTTPRequest</div> <div>(Mobile App)</div>	<ul style="list-style-type: none"> <li>• 전자정부 서버 어플리케이션과 Rest 통신을 위한 iOS 용 네이티브 Restful 서비스</li> <li>• 오픈 소스 라이브러리로 Interface Device API(iOS) 개발에 사용 되었다.</li> </ul>
<div>jQuery Mobile</div> <div>(Mobile Web)</div>	<ul style="list-style-type: none"> <li>• 전자정부 모바일 웹 표준프레임워크의 코어 프레임워크로서 하이브리드 앱의 UX 개발 시에 사용된다. 네이티브 어플리케이션의 화면 전환 효과 및 각종 버튼 리스트 등을 활용한 개발을 위해 사용된다.</li> </ul>
<div>jQuery</div> <div>(Mobile Web)</div>	<ul style="list-style-type: none"> <li>• jQuery Mobile 프레임워크의 코어 프레임워크로 jQuery Mobile을 사용하기 위해서 필수로 사용된다. Dom Control, Ajax, Restful 서비스 연계 등 다양한 기능을 제공하며 자바스크립트를 이용한 개발을 간단하게 해준다.</li> </ul>
<div>iScroll</div> <div>(Mobile Web)</div>	<ul style="list-style-type: none"> <li>• 어플리케이션의 헤더와 푸터를 고정시킨 채 컨텐츠 내용만 스크롤 해주는 기능을 제공한다. jQuery Mobile에서 제공하지 못하는 UX 효과를 보조하기 위해 사용할 수 있다.</li> </ul>

□ 디바이스 API는 웹 리소스를 통해 디바이스 내의 Native 기능을 호출하기 위하여 디바이스 API 실행환경 내에서 JavaScript 형태로 제공되는 API의 모음이며 각 디바이스 API 별 특징은 다음과 같다.

순번	디바이스 API	하드웨어 기능	설명
1	Accelerator	가속도계	단말기의 가속도계 정보를 제공하는 API(단말기의 움직임 정보를 x, y, z 축의 값으로 제공)
2	GPS	GPS	단말기의 현재 위치에 대한 정보를 제공하는 API
3	Vibrator	진동처리	단말기의 진동 및 알림음 기능을 호출 할 수 있는 API
4	Camera	카메라	단말기의 카메라 촬영 기능을 호출 할 수 있는 API
5	Contact	연락처	단말기의 주소록(연락처) 정보를 조회 및 수정 할 수 있는 API
6	Compass	나침반	단말기의 방향정보를 조회 할 수 있는 API
7	File Reader/File Writer	파일 일기/쓰기	단말기의 내장 저장 장치의 파일을 읽기/쓰기 할 수 있는 API
8	Network	네트워크 정보	단말기의 네트워크 연결 정보를 조회 할 수 있는 API
9	Device	디바이스 정보	단말기의 기본 정보(UUID, 버전 등)을 조회 할 수 있는 API
10	Media	오디오파일	단말기의 오디오 파일을 컨트롤 할 수 있는 API
11	Interface	통신	전자정부 표준프레임워크 기반 웹 서버 어플리케이션과 연계를 지원하는 API
12	NPki	NPki	단말기에 설치 된 npki 모듈을 호출 할 수 있는 API

□ 디바이스 API는 웹 리소스를 통해 디바이스 내의 Native 기능을 호출하기 위하여 디바이스 API 실행환경 내에서 JavaScript 형태로 제공되는 API의 모음이며 각 디바이스 API 별 특징은 다음과 같다.

순번	디바이스 API	하드웨어 기능	설명
13	Push Notifications	푸쉬통보	모바일 앱 사용자에게 다양한 푸시 메시지를 전달할 수 있는 기능을 제공하는 API
14	File Opener	문서뷰어 연결	단말기의 사용 가능한 문서 앱의 연동을 제공하는 API
15	Streaming Media	스트리밍 미디어	멀티미디어(동영상)을 실시간으로 볼수 있도록 내장 미디어 플레이어로 연동하는 기능을 호출할 수 있는 API
16	Barcode scanner	바코드 스캐너	바코드, QR코드등 정보를 확인 할 수 있는기능을 호출 할 수 있는 API
17	WebResource Update	웹리소스 업데이트	웹 리소스의 최신버전 조회 및 버전 업데이트를 진행할 수 있는 기능을 호출 할 수 있는 API
18	Device FileMgmt	파일관리	디바이스 저장소 내의 폴더(디렉토리) 및 파일 관리(이동,삭제,복사) 기능을 호출 할 수 있는 API
19	JailbreakDetection	탈옥및 루팅 체크	디바이스의 루팅 및 탈옥 정보 조회 기능을 호출 할 수 있는 API
20	SocketIO	웹소켓	웹서버의 websocket에 접속하여 양방향 데이터 처리 기능을 사용 할 수 있는 API
21	SQLite	SQLite DB	디바이스 내 독립적인 DB를 사용 할 수 있는 기능을 지원하는 API
22	Unzip	파일 압축 해제	단말기의 파일의 압축과 해제 기능을 지원하는 API

### □ Sample Code

```
<!DOCTYPE html>
<html>
<head>
<title>Acceleration Example</title>

<script type="text/javascript" charset="utf-8" src="cordova.js"></script>
<script type="text/javascript" charset="utf-8">

document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
    navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
}

function onSuccess(acceleration) {
    alert('Acceleration X: ' + acceleration.x + '\n' +
        'Acceleration Y: ' + acceleration.y + '\n' +
        'Acceleration Z: ' + acceleration.z + '\n' +
        'Timestamp: ' + acceleration.timestamp + '\n');
}

function onError() {
    alert('onError!');
}

</script>
</head>
<body>
<h1>Example</h1>
</body>
</html>
```

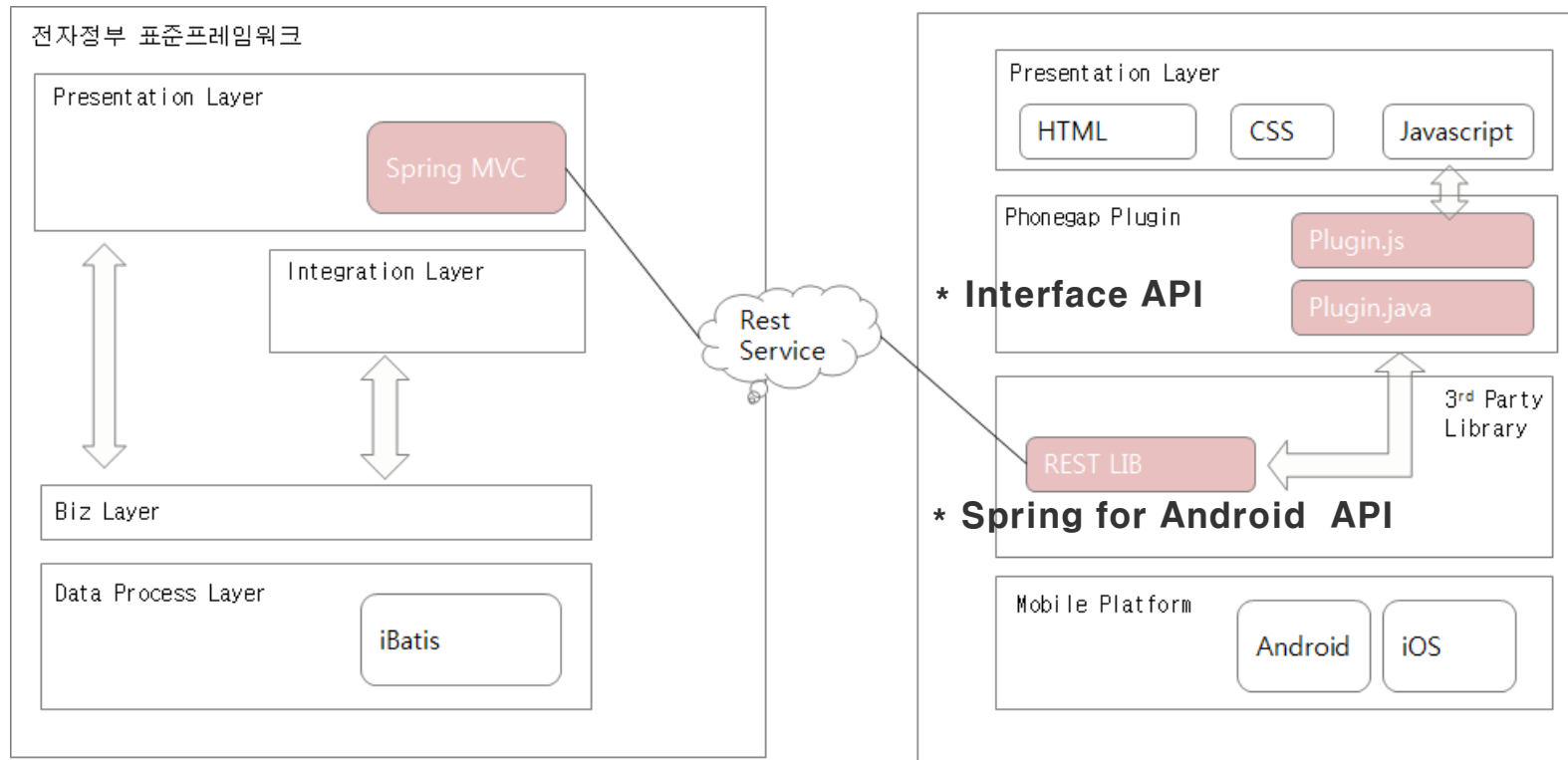
Cordova javascript interface 선언

Cordova 정상 초기화시 device ready Event 발생

Deviceready가 성공하면 navigator.accelerometer 객체가 주입

디바이스의 가속도계로부터 위치이동 변동값 x,y,z를 전달받는다. Cordova가 휴대폰 기기에서 가져와서 전달한다.

- 전자정부 표준 프레임워크 서버 어플리케이션과 모바일 하이브리드 앱은 Loosely Coupling 되어 있으며, Restful 방식으로 웹서비스를 통해 통신한다.



- ① 전자정부 표준 프레임워크 기반의 서버 어플리케이션과 모바일 클라이언트는 웹서비스를 통해 Loosely Coupling 된다.
- ② 웹서비스는 REST Service 프로토콜을 이용하여 클라이언트와 통신한다.
- ③ 웹서비스를 이용하는 모바일 클라이언트는 전자정부 표준프레임워크 REST 서비스 뿐만 아니라, 표준 REST 서비스를 구현한 서버와의 통신이 가능하다.

- ❑ 특정 서버와 HTTP/HTTPS 통신을 하는 static library이다.
- ❑ GET, POST, PUT, DELETE로 특정서버와 통신 파일 upload / download 가 가능하다.

### Spring for Android

Spring for Android 는 안드로이드 어플리케이션 개발을 단순화 하기 위한 Spring framework 의 확장기능 이다.

- ① REST Client 로써 Java 기반의 RestTemplate 사용을 포함
- ② 트위터, 페이스북 같은 소셜 네트워크에 대한 인증을 지원
- ③ JSON, XML, RSS, Atom 에 대한 third-party 라이브러리를 사용할 수 있음
- ④ 최종 1.0.1 release 버전 사용
- ⑤ Maven 지원

### Function List

구분	설명
window.plugins.EgovInterface.get	- HTTP GET Method를 수행한다.
window.plugins.EgovInterface.post	- HTTP POST Method를 수행한다.
window.plugins.EgovInterface.geturl	- 환경설정에서 설정한 SERVER_URL을 얻어온다.

```

window.plugins.EgovInterface.get(url,accept_type, null, function(jsontdata) {
    var data = JSON.parse(jsontdata);
        ...
});

window.plugins.EgovInterface.post(url, accept_type, null, function(jsontdata) {
    var data = JSON.parse(jsontdata);
        ...
});

window.plugins.EgovInterface.geturl(function(url){
        ...
});
    
```

### □ Sample Code

```

window.plugins.EgovInterface.get(url,accept_type, null, function(jsondata) {
    var data = JSON.parse(jsondata);
    var list_html = "";
    var totcnt = data.networkInfoList.length;
    for (var i = 0; i < totcnt; i++) {
        result = data.networkInfoList[i];
        list_html += "<li><h3>UUID : " + result.uuid + "</h3>";
        list_html += "<p><strong>Network Connection Type : " + result.networktype + "</strong></p>";
        list_html += "<p>Availability : " + result.useYn + "</p></li>";

    }
    var theList = $('#theList');
    theList.html(list_html);

    $.mobile.changePage("#networkInfoList", "slide", false, false);
    theList.listview("refresh");
    setTimeout(loadScroll, 200);
});

```

```

var uri = "/nwk/deleteNetworkInfo.do";
//Accept_Type setting
var accept_type = "json";
// http post method call
egovHyb.post(url, accept_type, null, function(jsondata) {
    var data = JSON.parse(jsondata);
    if(data.resultState == "OK"){
        $.mobile.changePage("#networkInfo", { transition: "slide", reverse: true });
    }else{
        $("#alert_dialog").click( function() {
            jAlert('데이터 삭제 중 오류가 발생 했습니다.', '삭제 오류', 'c');
        });
    }
});
});

```

- ❑ 특정 서버와 HTTP/HTTPS 통신을 하는 static library이다.
- ❑ GET, POST, PUT, DELETE로 특정서버와 통신 파일 upload / download 가 가능하다.

### 추가 파일

#### ① EgovInterface Framework

- CFNetwork.framework
- MobileCoreServices.framework
- SystemConfiguration.framework
- Security.framework

#### ② Libraries

EgovInterface API를 연결하는 header 파일

- EgovComModule.h
- libEgovComModule.a

#### ③ EgovInterface

해당 header 파일과 구현부 파일이 있어야만 EgovInterface API와 연동

- EgovInterface.h
- EgovInterface.m

### Function List

구분	설명
initWithURL:delegate	- 초기화와 기본적인 설정을 한다.
setURL	- 통신할 곳의 URL을 설정한다.
setTimeoutSecondsI	- time out 초를 설정한다. Default로 10초로 설정되어 있다.
addPost:key	- 입력하고자 하는 값을 key값의 request 파라미터에 설정한다.
startAsynchronous	- 설정하고 입력한 값으로 비동기 통신을 시작한다.

구분	설명
onNetworkStarted	- 통신이 시작된 후 일어나는 이벤트
onNetworkFailed	- 통신이 실패한 후 일어나는 이벤트
onNetworkFinished :responseString: responseStatusCode:	- 통신이 성공한 후 일어나는 이벤트



### □ Sample Code

```
//생성방법
EGovComModule *m_module = [[[EGovComModule alloc] initWithURL:[NSString stringWithFormat:@"%s/index.do",kSERVER_URL]
delegate:self] autorelease];

//wi-fi 및 네트워크 체크
if (m_module.isWifi) {
    NSLog(@"wi-fi");
} else {
    if (m_module.isNetworkConnected) {
        NSLog(@"3G/4G");
    } else {
        NSLog(@"Network Disconnected");
    }
}

//통신 시작
[m_module startAsynchronous];

//key value로 값을 추가하고자 할 경우
[m_module addPost:@"값" key:@"key"];
//통신 시작 이벤트
- (void)onNetworkStarted
{
}

//통신 실패 이벤트
- (void)onNetworkFailed:(NSError*)error
{
}

//통신 완료 이벤트
- (void)onNetworkFinished:(NSData*)responseData responseString:(NSString*)responseString
responseStatusCode:(NSInteger)responseStatusCode
{
}
```

네이티브 Function을 호출 하기위해 cordova.exec 자바스크립트를 이용한다.

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

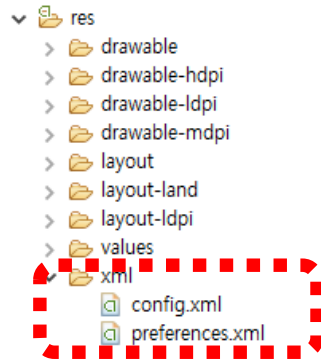
구분	설명
successFunction	- Android 네이티브 함수가 정상 호출 되었을 때, 호출되는 자바스크립트 콜백 함수
failFunction	- Android 네이티브 함수의 호출 결과 에러가 발생했을 때, 호출 되는 자바스크립트 콜백 함수
service	- 플러그인 설정파일인 config.xml에서 설정된 플러그인의 Name
action	- 서비스 구분을 위해 Android 네이티브 함수로 전달되는 파라미터
args	- Android 네이티브 함수 호출에 필요한 파라미터 배열

res/xml/config.xml

```
<feature name="<service_name>">
  <param name="android-package" value="<full_name_including_namespace>"
/>
</feature>
```

\* 예시

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



구분	설명
name	- cordova.exec 함수에 전달되는 플러그인 서비스 Name
value	- cordova.exec 함수에 의해 호출되는 네이티브 패키지명 클래스명

- 플러그인 작성을 위해서는 폰갭 프레임워크에서 제공하는 Plugin 클래스를 상속받아야 한다.
- cordova.exec에 의해 호출되는 메서드인 execute 메서드를 작성해야 하며 PluginResult클래스를 반환한다.

```
public class Echo extends CordovaPlugin {
    @Override
    public boolean execute(String action, JSONArray args, CallbackContext callbackContext) throws JSONException
    {
        if (action.equals("echo")) {
            String message = args.getString(0);
            this.echo(message, callbackContext);
            return true;
        }
        return false;
    }

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

구분	설명
action	- 플러그인 서비스 구분을 위해 사용되는 Action명
args	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터들
callbackId	- 플러그인 실행 완료 후 호출되는 자바스크립트 콜백 Function

네이티브 Function을 호출 하기위해 cordova.exec 자바스크립트를 이용한다.

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

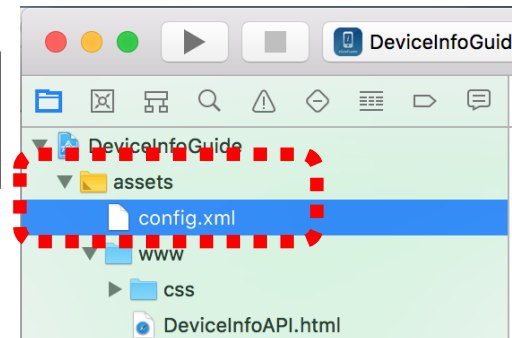
구분	설명
successFunction	- iOS 네이티브 함수가 정상 호출 되었을 때, 호출되는 자바스크립트 콜백 함수
failFunction	- iOS 네이티브 함수의 호출 결과 에러가 발생했을 때, 호출 되는 자바스크립트 콜백 함수
service	- 플러그인 설정파일인 plugin.xml에서 설정된 플러그인의 Name
action	- 서비스 구분을 위해 iOS 네이티브 함수로 전달되는 파라미터
args	- iOS 네이티브 함수 호출에 필요한 파라미터 배열

/assets/config.xml

```
<feature name="<service_name>">
  <param name="ios-package" value="<including_namespace>" />
</feature>
```

\* 예시

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



구분	설명
service_name	- cordova.exe 함수에 전달되는 플러그인 서비스 Name
PluginClassName	- cordova.exe 함수에 의해 호출되는 iOS 네이티브 클래스

플러그인 작성을 위해서는 폰갭 프레임워크에서 제공하는 CDVPlugin 클래스를 상속받아야 한다.

```
#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 (NSEException* exception) {
        pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_JSON_EXCEPTION messageAsString:[exception reason]];
        javascript = [pluginResult toErrorCallbackString:callbackId];
    }

    [self writeJavascript:javascript];
}

@end
```

구분	설명
arguments	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터
options	- 플러그인 호출 자바스크립트에서 넘겨주는 파라미터 중 마지막 배열 값