하이브리드 모바일 앱 프레임워크 설계 및 구현

정우진* · 오장훈* · 윤동원**

Design and Implementation of Hybrid Mobile App Framework

Woojin Jung* · Janghoon Oh* · Dongweon Yoon**

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의
연구결과로 수행되었음(NIPA-2012-H0401-12 -2003)

요 약

본 논문에서는 기존 웹앱 기반의 하이브리드 모바일 앱의 실행 성능 및 사용성을 개선하기 위하여 각 모바일 운영체제가 지원하는 네이티브 UI(User Interface) 및 각종 자원을 자바스크립트(JavaScript)를 이용하여 직접 제어할 수 있는 새로운 하이브리드 모바일 앱 프레임워크인 WApplE.js를 설계하고 구현한다. WApplE.js 프레임워크의 전체 소프트웨어 구조 및 레이어별 구성에 대하여 설계 결과를 제시하고, 구현된 하이브리드 앱 프레임워크에서 사용자의 API 호출 및 처리 프로세스에 대한 분석을 수행하며, 기존 프레임워크들과 특징을 비교하여 그 결과를 제시한다.

ABSTRACT

In this paper, in order to improve the execution performance and serviceability of cross-platform applications frameworks based on the existing web applications, we design and implement a new hybrid application framework named as WApplE.js, which enables direct control of native UI(User Interface) of mobile operating systems and various resources via JavaScript. We first present the design results for the overall software structure and the configuration of every layer of WApplE.js, and then analyze the processes for calling and handling APIs in the implemented hybrid application framework. In addition, the results of comparison of features to the existing frameworks are presented.

키워드

하이브리드 모바일 앱 프레임워크, 크로스 플랫폼, 모바일 앱, WApplE.js

Key word

Hybrid Mobile App Framework, Cross-Platform, Mobile App, WApplE.js

* 정회원 : 한양대학교 전자컴퓨터통신공학과
** 정회원 : 한양대학교 융합전자공학부 (교신저자, dwyoon@hanyang.ac.kr)

접수일자 : 2012. 08. 20
심사완료일자 : 2012. 09. 07

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.9.1990

Ⅰ. 서 론

2007년도 미국 애플사의 아이폰 출시는 혁신적인 사용자 인터페이스의 편리성을 바탕으로 사용자들에게 모바일 컴퓨팅에 대한 새로운 인식을 가지게 하고 모바일 패러다임의 커다란 변화를 주도하게 되었다. 하지만, 현재의 모바일 생태계는 2008년도 동사의 앱스토어 서비스 공개를 통하여 그 최초 형태를 갖추게 되었다[1].

이러한 모바일 생태계의 발생 및 확장은 애플의 iOS, 구글의 안드로이드, 그리고 마이크로소프트의 윈도우즈 모바일 등 상호 호환성이 결여된 다수의 모바일 운영체제가 등장함에 따라 특정 운영체제를 위한 별도의 애플리케이션을 개발하여야 하는 부담이 증가하는 것도 현실이다. 개발언어 및 개발환경의 파편화 문제는 개발 비용 뿐 아니라 개발 이후에 유지보수 하는데 들어가는 비용 역시 크게 증가시킨다. 따라서 다양한 모바일 운영체제를 지원하여 최소의 비용으로 애플리케이션을 개발, 유지보수하기 위한 여러 가지 크로스 플랫폼 기술이 개발, 적용되고 있다[2][3].

하이브리드 앱 프레임워크는 HTML5(Hyper Text Markup Language 5)와 HTML5를 지원하는 웹킷(WebKit) 기술 및 그 확장성을 이용하여 웹 기술 기반의 애플리케이션을 각 운영체제의 독자적 개발 언어로 개발된 네이티브 애플리케이션과 동일하게 생성해 내는 프레임워크를 의미하며, 대표적으로 폰갭(PhoneGap)과 앱스프레소(Appspresso)가 있다. 이러한 기존 하이브리드 앱 프레임워크는 표준화된 기술을 사용함으로써 다양한 모바일 운영체제를 동시에 지원하는 효율성 있는 프레임워크 기술인 장점이 있는 반면, 네이티브 애플리케이션에 비하면 웹킷이나 자바스크립트(JavaScript) 엔진 성능을 거쳐야 하므로 실행 속도가 느릴 뿐만 아니라 운영체제가 제공하는 사용자 인터페이스 요소들을 직접 사용하지 못하는 단점이 있다.

즉, 운영체제가 직접 제공하는 사용자 인터페이스 요소들은 운영체제와 긴밀히 연결되어 있어 최대한의 성능을 내도록 되어 있으나 웹 기술을 이용한 사용자 인터페이스는 웹킷의 HTML 렌더링을 거쳐야 하므로 성능의 저하를 피할 수 없다. 따라서 다양한 운영체제를 기반으로 하는 여러 형태의 모바일 디바이스, 스마트 TV 및 자동차 IT 장비 등에 대한 안정적 지원을 위해서는 실행 성능 및 사용성이 개선된 하이브리드 모바일 앱의 개발이 필요하다.

본 논문에서는 HTML5, CSS3(Cascading Style Sheets3), 자바스크립트를 이용해서 네이티브 앱을 개발할 수 있는 새로운 하이브리드 앱 프레임워크인 WApplE.js(Web Application Engine Java Script Framework)를 설계하고 구현한다. WApplE.js는 표준 웹 기술을 그대로 이용할 수 있을 뿐만 아니라 모바일 운영체제가 지원하는 사용자 인터페이스 요소들을 직접 제어하므로 HTML 렌더링을 거칠 필요가 없고, 따라서 기존의 하이브리드 모바일 앱 프레임워크에 비해 빠른 실행이 가능하다.

Ⅱ. 하이브리드 모바일 앱 프레임 워크

2.1. 하이브리드 모바일 앱 프레임워크 개요

일반적으로 모바일에서 구동하는 애플리케이션은 크게 네이티브, 웹 그리고 하이브리드 애플리케이션으로 나눌 수 있다. 그림 1에서는 주요 모바일 애플리케이션 유형별 특징을 나타내고 있다.

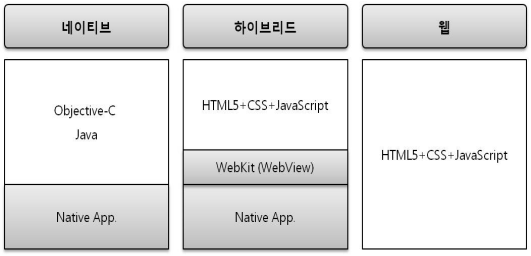

그림 1. 모바일 애플리케이션 유형별 구성도
Fig. 1 Structure diagram of three types of mobile applications

네이티브 애플리케이션은 iOS, 안드로이드와 같은 모바일 운영체제가 제공하는 개발 환경에서 Objective-C 또는 Java와 같은 전용 개발 언어를 사용하여 만들어지는 애플리케이션을 말한다[4][5]. 웹 애플리케이션은 스마트폰에 탑재되어 있는 웹 브라우저를 기반으로 하며 주로 원격의 모바일 웹 서버에서 주된 기능을 수행

한다.

하이브리드 애플리케이션은 최신 모바일 기기들 대부분에 적용되어 있는 웹킷이 제공하는 웹뷰(WebView)를 기반으로 웹 페이지를 구성하는 HTML, CSS, 자바스크립트 등의 기술을 이용하여 네이티브 애플리케이션을 만들어내는데 이 때 모바일의 기능에 대한 접근을 가능케 하는 API(Application Programming Interface)를 제공하고 최종 결과물로 네이티브 애플리케이션을 생성해내는 역할을 하는 프레임워크를 하이브리드 앱 프레임워크라고 한다.

이와 같은 하이브리드 모바일 앱 프레임워크가 주목받는 이유는 보안상 접근하지 못하는 모바일 자원에 접근할 수 있는 기능을 제공하여 웹앱의 기능을 확장할 수 있도록 해 주기 때문이다.

2.2. 주요 하이브리드 앱 프레임워크

최근 다양한 하이브리드 앱 프레임워크가 상용화되고 있으며, 그 중 대표적 웹 기반 프레임워크인 폰갭과 앱스프레소의 구조를 살펴본다.

2.2.1. 폰갭(PhoneGap)

폰갭은 2008년 처음 Nitobi사에 의해 개발되었으며, HTML5, CSS, 자바스크립트와 같은 표준 웹 기술을 이용하여 애플리케이션을 개발하는데 구조적으로 모바일 브라우저를 앱에 내장시키는 형식이다. 그림 2는 폰갭 프레임워크의 구조도를 나타낸다[6].

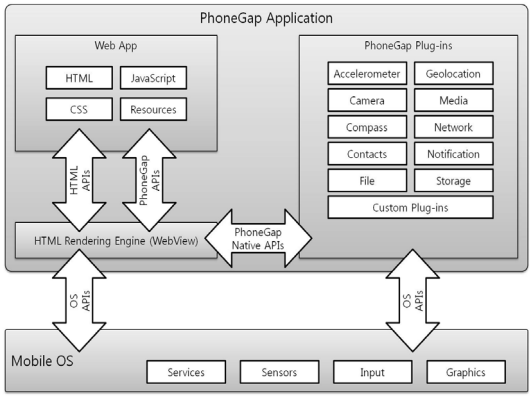

그림 2. 폰갭 프레임워크 구조도
Fig. 2 Structure diagram of PhoneGap framework

폰갭의 구조는 모바일 브라우져가 기본적으로 제공하는 기능만을 활용하면 모바일 디바이스에서 제공하는 기능인 센서 등의 정보를 활용할 수 없으므로 이런 추가적인 정보나 기능을 이용할 수 있도록 하기 위해 폰갭 자체적인 API를 제공한다. 이에 더하여 개발자가 직접 새로운 기능을 추가할 수 있는 플러그인 확장 기능도 제공한다. 폰갭은 하나의 웹앱 소스 코드로 iOS, 안드로이드, 블랙베리, WebOS, 심비안, 삼성 바다 그리고 윈도우즈폰7 운영체제에서 동작하는 애플리케이션을 동시에 생성해 낼 수 있고 이는 현존 하이브리드 앱 프레임워크 중 가장 많은 운영체제 지원이 가능하다[7][8][9].

2.2.2. 앱스프레소(Appspresso)

앱스프레소는 KTH사에서 개발하여 무료로 배포하는 하이브리드 앱 프레임워크이며, 웹킷 기반의 브라우져를 내장하는 형태의 하이브리드 프레임워크로 폰갭과 유사한 구조를 가지며 구성 및 동작 또한 비슷하다. 주요 특징으로는 WAC (Wholesale Application Community)에서 제정한 Waikiki API를 제공한다는 점을 들 수 있다. 브라우져가 제공하는 기본 기능과 함께 Waikiki API를 제공하고, 자체적인 API도 제공하고 있다. 또한 다른 프레임워크와 마찬가지로 개발자가 직접 플러그인을 만들어 적용할 수 있는 확장 기능도 제공한다. 그림 3은 앱스프레소 프레임워크의 구조도를 나타낸다[10].

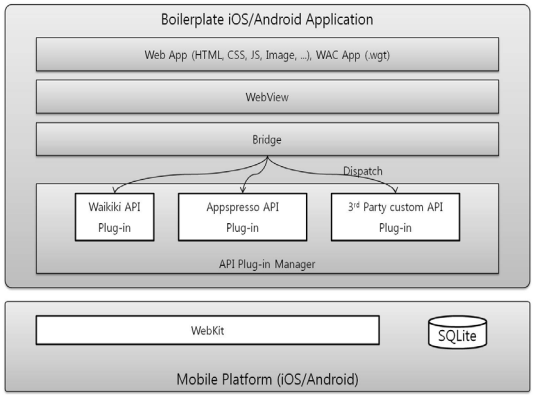

그림 3. 앱스프레소 프레임워크 구조도
Fig. 3 Structure diagram of Appspresso framework

Ⅲ. WApplE.js 프레임워크 설계 및 구현

3.1. WApplE.js 아키텍처

WApplE.js는 웹 기술을 이용해서 네이티브 애플리케이션을 개발할 수 있도록 자바스크립트 API를 제공하는 하이브리드 모바일 앱 프레임워크이다. 타 하이브리드 프레임워크와 동일하게 웹킷에서 제공하는 웹뷰 기반으로 HTML, CSS를 이용하여 UI를 구성할 수 있도록 지원하고, 타 프레임워크에는 없는 고유기능으로 제공하는 윈도우를 통하여 자바스크립트 API 를 이용해 호출할 수 있는 네이티브 UI 요소를 활용할 수 있다. 따라서 그림 4와 같이 두 개의 윈도우가 겹친 형태의 멀티-레이어 윈도우(Multi-layer Windows)로 구성된다. 이러한 구성은 웹 UI와 네이티브 UI를 두 개의 윈도우에 별도 구성 후 중첩시킴으로써 다양한 결과물을 만들어 낼 수 있다.

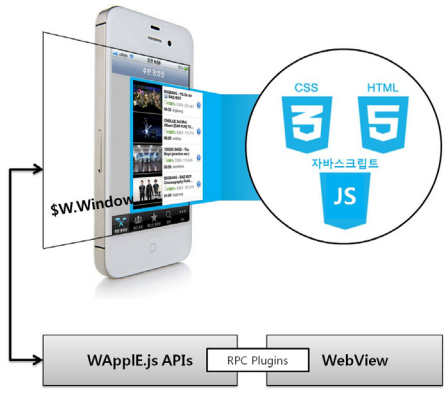

그림 4. WApplE.js의 멀티-레이어 윈도우 시스템
Fig. 4 Multi-layer windows system of WApplE.js

웹 UI는 다양한 라이브러리를 활용할 수 있는 장점이 있으나 웹킷이나 웹뷰의 성능에 따라 속도가 네이티브 애플리케이션에 비해 현저히 떨어질 가능성이 존재한다. 하지만, WApplE.js 프레임워크는 멀티-레이어 윈도우 시스템을 통해 네이티브 UI를 제공하므로 웹 UI에 비하여 빠른 속도의 성능을 구현할 수 있다.

WApplE.js가 지원하는 자바스크립트 API는 네이티브 UI에 대한 자바스크립트를 이용하는 제어가 가능하도록 한다. 또한 단말기에 포함된 위치, 방향 정보 등의 센서 정보 또는 모바일 데이터 베이스와 같은 Device API 인터페이스를 제공하고 WApplE.js 프레임워크에서 지원하지 않는 장치나 추가적인 기능을 제3자가 쉽게 개발하여 추가할 수 있도록 플러그인 기능도 제공한다. WApplE.js의 소프트웨어 구조는 그림 5에서와 같이 크게 두 개 레이어 - 자바스크립트 레이어(JavaScript Layer)와 네이티브 레이어(Native Layer) - 로 구성된다.

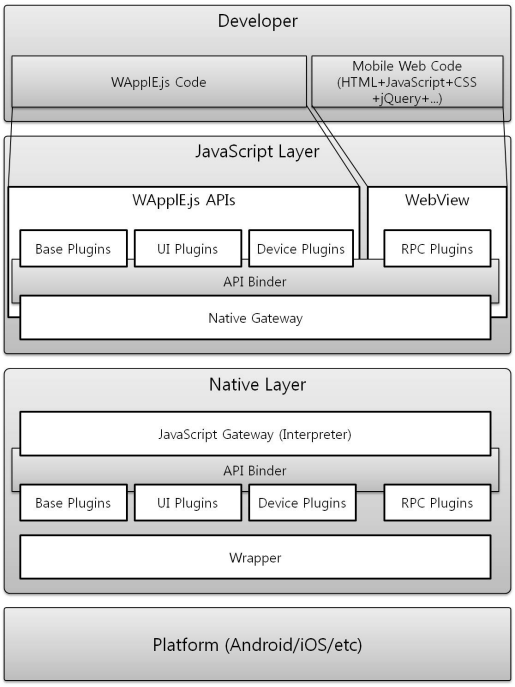

그림 5. WApplE.js 아키텍쳐
Fig. 5 Software architecture of WApplE.js

자바스크립트 레이어는 개발자가 직접 사용 가능한 자바스크립트 API를 제공하는 레이어로서 UI 플러그인, 베이스 플러그인과 디바이스 플러그인으로 구분되며, 개발자에게 제공하는 자바스크립트 API를 제공하는 부분이다. 네이티브 레이어는 자바스크립트 레이어가 제공하는 API의 기능을 실제 각 플랫폼에 포팅되어 기능을 구현하는 레이어이다. 이는 현재 iOS와 안드로이드 플랫폼 버전으로 구현되어 있다.

3.1.1. 자바스크립트 레이어

자바스크립트 레이어(JavaScript Layer)는 개발자에

게 사용 가능한 자바스크립트 API를 제공하는 레이어이다. 이는 크게 UI 플러그인, 베이스 플러그인과 디바이스 플러그인이 WApplE.js 자바스크립트 API를 제공하는 부분이고 프레임워크의 공통 기능을 정의하고 제공하는 코어(Core)와 네이티브 레이어와의 인터페이스 기능을 수행하는 네이티브 게이트웨이 (Native Gateway)로 구성되어 있다.

자바스크립트 레이어의 UI 플러그인은 각 플랫폼에서 제공하는 사용자 인터페이스 컨트롤의 공통 요소를 추출하여 자바스크립트 API로 제공하는 역할을 수행한다. 즉, 각 플랫폼이 제공하는 네이티브 UI 요소를 추상화하고 자바스크립트 API로 제공하는 역할을 수행한다.

베이스 플러그인은 UI 플러그인과는 다르게 화면상에 보여지는 UI 요소가 아니라 데이터베이스, 메시지 로그, 파일 시스템과 같이 단말기에서 제공하는 기능을 호출 할 수 있는 플러그인이다. 마지막으로 디바이스 플러그인은 단말기의 상태 정보, 가속도 센서, 방향 센서등과 같은 하드웨어적인 측정 정보를 제공하는 플러그인이다. 그림 6은 개발자가 UI 플러그인이 제공하는 API를 호출했을 때 자바스크립트의 시퀀스 다이어그램을 제시한다. 이 시퀀스 다이어그램은 베이스 플러그인과 디바이스 플러그인에 공통적으로 적용이 가능하다.

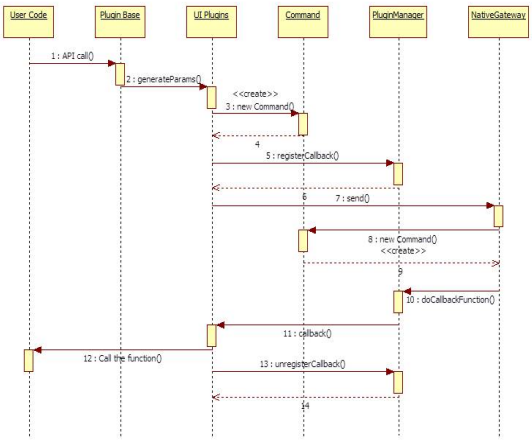

그림 6. UI 플러그인 시퀀스 다이어그램
Fig. 6 Sequence Diagram for UI Plugins

네이티브 게이트웨이는 자바스크립트 레이어와 네이티브 레이어 간 통신을 담당하며, 크게 NativeGateway 클래스와 Command 클래스로 구성된다. NativeGateway 클래스는 자바스크립트 레이어에서 발생한 명령을 네이티브 레이어로 전달 하거나 네이티브 레이어에서 발생한 명령 정보를 수신하고 해석한다. 이러한 역할을 수행하기 위해 NativeGateway 클래스는 내부에 array 형태의 Send Queue와 Receive Queue를 가지고 있으며, 네이티브 게이트웨이와 네이티브 레이어의 자바스크립트 게이트웨이(JavaScript Gateway) 간의 통신에서 명령어 전달에 의한 오버헤드(overhead)를 최소화 하는 완충 역할을 한다.

3.1.2. 네이티브 레이어

네이티브 레이어에서는 자바스크립트 레이어가 개발자에게 제공하는 공통 API가 실제로 사용되는 모바일 플랫폼의 고유 API로 변환되고, 실행된 API 결과가 자바스크립트 레이어로 전달된다. 따라서 네이티브 레이어는 WApplE.js 프레임워크를 적용하고자 하는 모바일 플랫폼에 종속되어 별도로 설계, 구현되어야 한다.

네이티브 레이어는 자바스크립트 레이어와의 통신을 담당하는 자바스크립트 게이트웨이, 자바스크립트 레이어의 UI 플러그인, 베이스 플러그인, 디바이스 플러그인을 실제 모바일 운영체제 기반으로 처리하기 위한 는 UI 플러그인, 베이스 플러그인, 디바이스 플러그인, 그리고 실제 모바일 플랫폼의 API를 호출하는 래퍼(Wrapper) 클래스 모듈로 구성된다.

3.2. WApplE.js의 API 호출 및 처리 프로세스

구현한 WApplE.js에서 모바일 운영체제의 네이티브 기능을 사용하기 위해서는 API를 호출하여 필요한 기능을 수행하도록 하여야 한다. 이렇게 개발자가 실제 WApplE.js를 사용할 때 하이브리드 모바일 앱 프레임워크의 각 부분이 모바일 운영체제와의 연동을 위해 어떠한 절차와 단위 모듈을 거쳐 실행되는지 그 과정을 설명한다.

그림 7은 WApplE.js 프레임워크를 이용하는 개발자가 API를 호출하였을 때 프레임워크의 각 모듈이 어떤 순서에 의해 동작하는 지 단말기 운영체제가 제공하는 기능이 실행되기까지 거치는 절차를 나타내는 흐름도이다.

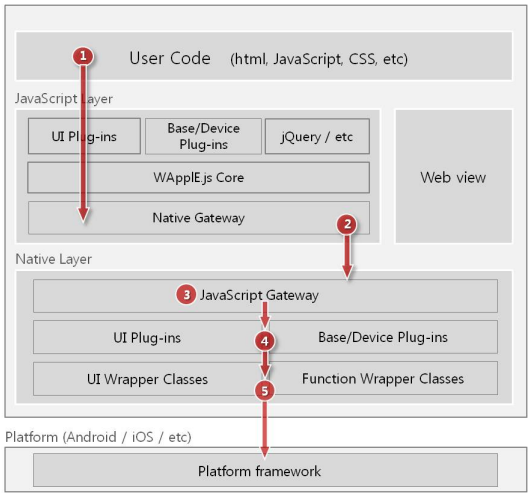

그림 7. WApplE.js 프레임워크 API 호출 및 처리 절차
Fig. 7 Workflow of calling and handling APIs in WApplE.js framework

상기 흐름도에서 숫자로 표시된 순서에 따라 절차를 예를 들어 설명하면 다음과 같다.

① 먼저 사용자가 $W.View와 같이 WApplE.js가 제공하는 API를 호출한다.

② 이 호출은 해당 플러그인, WApplE.js 코어, 그리고 네이티브 게이트웨이를 거친 후 delegate 또는 callback 등의 방법으로 네이티브 레이어로 전달된다.

③ 네이티브 레이어의 자바스크립트 게이트웨이는 사용자가 호출한 $W.View에 대응하는 네이티브 플러그인 레이어의 WPView 클래스를 호출한다.

④ WPView 클래스의 메서드는 래퍼 클래스에 구현되어 있는 기능을 호출한다.

⑤ 이를 통해 실제 모바일 운영체제가 지원하는 API를 호출한다.

제안하는 WAppE.Js와 기존 프레임워크인 폰갭, 앱스프레소간 주요 특징을 비교하면 다음과 같다. WAppE.Js는 실행 성능의 관점에서는 웹킷을 기반으로 하는 폰갭이나 앱스프레소에 비하여 네이티브 UI의 직접적인 제어가 가능하기 때문에 모바일 앱 실행 속도에서 향상된 성능을 보인다. 또한 사용성 측면에서는 폰갭이 지원하지 못하는 WAC에 대한 지원과 앱스프레소에서 제공하지 못하는 데이터베이스 기능인 SQLite를 모두 지원함으로써 다양한 기능을 가진 모바일 앱 개발이 가능함을 확인할 수 있다.

Ⅳ. 결 론

본 논문에서는 웹킷 기반 하이브리드 모바일 앱 프레임워크의 느린 속도를 개선하면서도 웹 표준 기술을 그대로 활용하여 애플리케이션 개발을 지원하는 멀티-레이어 윈도우 특징을 가진 새로운 WApplE.js 프레임워크를 설계 및 구현하였다.

WApplE.js는 자바스크립트 API를 이용해 모바일 운영체제가 제공하는 네이티브 UI를 제어 및 사용할 수 있으며, 베이스 플러그인, 디바이스 플러그인 등을 통해 단말기가 제공하는 센서 정보, 데이터베이스 정보를 포함하는 다양한 정보를 제어하거나 사용할 수 있다. 모든 자바스크립트 API가 플러그인 형태로 구성되어 있고 향후 추가로 필요한 API나 제 3 개발자가 개발한 기능을 손쉽게 통합하고 사용 가능하도록 설계하였다.

구성 측면에서 WApplE.js는 크게 자바스크립트 레이어와 네이티브 레이어로 구성되며, 새로운 모바일 운영체제를 지원하기위해 네이티브 레이어만 새로 개발할 수 있도록 하여 개발 기간이나 투입 자원을 최소화할 수 있는 장점을 가진다. 향후 다양한 모바일 운영체제를 기반으로 하는 모바일 디바이스, 스마트 TV, 자동차 IT 장비와 같은 다양한 정보기기를 위한 N-스크린 애플리케이션 개발을 위한 모바일 앱 프레임워크로 발전이 가능할 것으로 기대된다.

감사의 글

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음(NIPA-2012-H0401-12 -2003)

참고문헌

[1] 곽정호, 조지연, 이용석, 이봉규, “새로운 통신시장 활성화를 위한 모바일 생태계 통신정책,” 인터넷정보학회논문지 제12권 제4호, pp. 93-106, 2011.08.

[2] Feida Lin, Weiguo Ye, “Operating System Battle in the Ecosystem of Smartphone Industry,” *2009 International Symposium on Information Engineering and Electronic Commerce,* pp. 617-621, 2009.

[3] 김영주, 김경주, 유영중, 박성호, “XML 기반 스마트폰 미들웨어 Open API 구현,” 한국정보통신논문지, 제15권, 제4호, pp. 869-876, 2011.04.

[4] Daniel Y. Na, “The What, Why, and How of Mobile Applications,” *Sigma,* Vol.11, Issue 1, pp. 20-26, Oct. 2011.

[5] Gavalas, D., Economou, D., “Development Platforms for Mobile Applications: Status and Trends,” *Software, IEEE,* Vol.38, Issue 1, pp. 77-86, Jan. 2011.

[6] Bryce Curtis, “IBM, PhoneGap and the Enterprise,” *PhoneGap Day 2011,* Jul. 1960. http://www.slideshare.net/drbac/phonegap-day-ibm-phonegap-and-the-enterprise

[7] Thomas Myer,“Beginning PhoneGap,” *John Wiley & Sons*, 2012.

[8] 최재규, “하이브리드 모바일 앱 개발을 위한 폰갭-하이브리드 앱 전성시대 그리고 폰갭 플랫폼,” *마이크로소프트*, pp. 182-187, 2012.03

[9] 최재규, “하이브리드 모바일 앱 개발을 위한 폰갭-폰갭 플랫폼 내부 상세 분석,” *마이크로소프트*, pp. 182-187, 2012.04.

[10] 한기태, “하이브리드앱의 미래, 앱스프레소 1.0,” *H3 Developers Conference 2011,* pp. 14-29, 2011. 11.

저자소개

정우진(Woojin Jeong)

1992년 한양대학교 전자통신공학과 학사
1994년 한양대학교 전자통신공학과 석사
2005년～현재 한양대학교 전자컴퓨터통신공학과 박사과정
1994년～2002년 삼성전자 무선개발실 책임연구원
2002년～현재 (주) 트루모바일 대표이사
※ 관심분야 : 모바일 플랫폼, 임베디드 SW, HTML5

오장훈(Janghoon Oh)

1996년 한양대학교 전자통신공학과 학사
2006년 UC Irvine 전기공학과 석사
2011년~현재 한양대학교 전자컴퓨터통신공학과 박사과정
1996년~2004년 SK텔레콤 매니저
2008년~2009년 팜미디어 NS사업본부장
2009년~2011년 엔씨소프트 차장
※관심분야 : 차세대 이동통신, 임베디드 SW, 자동차 IT

윤동원(Dongweon Yoon)

현재 한양대학교 융합전자공학부 교수
※관심분야 : 통신 이론, 무선 및 이동 통신, 위성 및 우주 통신, IT 융합