Journal of Digital Contents Society Vol. 11 No. 3 Sep. 2010(pp. 323-330)

스마트폰상에서의 웹 응용프로그램 개발 효율성 분석

이고은*, 이종우**

요 약

스마트폰과 앱스토어 열풍은 응용프로그램 개발을 과거 PC응용에 이어 다시 한 번 활성화시키고 있다. 하지만 현재 스마트폰 응용프로그램 개발 시에는 다양한 플랫폼 환경 설정 고려와 이종 기기의 호환성 문제점 등의 한계가 있어 자연스럽게 모바일 웹 응용프로그램 개발이 대안으로 부각되고 있다. 모바일 웹 응용프로그램 형태의 하나인 하이브리드용 웹 응용프로그램은 스마트폰에 내장된 웹킷 엔진을 이용하기 때문에 간단하게 개발될 수 있다는 장점이 있다. 스마트폰의 웹킷 탑재로 웹 응용프로그램 개발자는 HTML과 자바스크립트, CSS만으로도 쉽고 빠르게 개발할 수 있을 뿐만 아니라 다양한 모바일 기기에 독립적인 서비스를 제공할 수 있다. 본 논문에서는 기존에 복잡하게 개발했던 네이티브 응용프로그램 대신 웹킷을 이용하여 단순하고 간결하게 개인이 운영할 수 있는 앱스토어를 안드로이드 환경에서 구현하였다. 기존 네이티브 응용프로그램에서는 복잡하게 구현했던 회원 등록 및 로그인 환경을 OpenID를 이용하여 간결하게 할 수 있었다. 이를 통해 스마트폰용 네이티브 응용이 아닌 웹 응용으로도 성능이 뒤지지 않으면서 쉽게 응용을 개발할 수 있음을 확인하였다. 아울러 웹 응용프로그램 개발 시 장점을 네이티브 응용 개발과 비교함으로써 그 효율성을 보였다.

Analyzing Effectiveness of the Web Application Development in Smartphone

Go Eun Lee*, Jongwoo Lee**

Abstract

Due to the widespread smartphones and application stores, mobile application developments are now booming again in the same manner of PC history. The smartphone software development process, however, gives much inconvenience to developers because of the heterogeneous platform APIs and hardware incompatibility between different devices. To make clear these problems, mobile web applications are being accepted as an alternative to the native smartphone applications. Mobile web applications can be developed more simply and easily than native applications by using webkit engine's html, Java script, and CSS. Additionally developers can provide a platform-independent applications since web applications are going to run on web browsers. In this paper, we develop a personal applications store running on an android phone's browsers. We can accomplish this very simply by using webkit's various APIs such as OpenID. We can find out by implementing a real web application that development of web applications can surpass the native one in cost and time period without much loss of performance.

Keywords : smartphone, web application, webkit, android, OpenID

※ 제일저자(First Author) : 이고은
접수일:2010년 08월 29일, 수정일:2010년 09월 25일, 완료일:2010년 09월 28일
* 숙명여자대학교 멀티미디어과학과
goen0906@hotmail.com
** 숙명여자대학교 멀티미디어과학과(교신저자)
▣ 본 연구는 숙명여자대학교 2009학년도 교내 연구비 지원에 의해 수행되었음

1. 서론

스마트폰은 인터넷을 본격적으로 활용하는 모바일 인터넷 기기라는 점에서 기존의 휴대폰과 차별된다. 스마트폰 경쟁에서 다양한 응용프로그램(이하 응용)의 이용여부가 스마트폰 선택에 있어 가장 중요한 요인이 되고 있다. 2008년 7월

출시되어 각종 엔터테인먼트, 유틸리티, 교육, 게임 등의 응용을 제공하고 있는 애플(Apple)의 앱스토어(Appstore)는 2010년 1월까지 사용자가 내려 받은 누적 건수가 30억 이상이며 이에 따른 애플사의 매출은 61억 달러에 달한다고 애플사가 발표했다[1]. 애플사의 선전에 따라 다양한 업체들이 모바일 응용 경쟁에 본격적으로 뛰어들고 있다. 이는 기존 이동통신사 중심의 폐쇄적인 모바일 응용 개발 및 유통구조가 마켓 플레이스를 중심으로 개방적인 환경으로 변하고 있음을 의미한다. 특히 아이폰(iPhone)과 안드로이드(Android) 플랫폼용 사용자 개발 응용의 인기가 높아지면서 모바일 응용분야도 새롭게 조명받고 있다.

[2]는 안드로이드폰에서 웹킷(webkit)을 이용한 클라이언트/서버 구현방법을 제시하였다. 웹킷에서 제공하는 API를 이용하면 프로그램의 길이가 몇 백 줄에서 단 3~4줄로 짧아지며, 웹 환경에서 서비스를 이용하면서 스마트폰의 내장된 부가기능까지 사용할 수 있는 장점이 있다. 이렇게 네이티브(native) 응용처럼 실행되면서 웹 브라우저에서도 사용하는 응용을 하이브리드(hybrid) 웹 응용이라고 한다. 본 연구진은 [2]에서 제안한 방식이 기존 소켓 방식보다 간편하게 사용할 수 있다는 점에 착안하여 안드로이드폰 환경에서 웹킷을 이용한 하이브리드 웹 응용형태의 소형 앱 스토어를 구현하였으며, 이를 토대로 기존 소켓으로 구현한 네이티브 응용과 본 논문에서 구현한 웹킷 응용을 개발비용 측면에서 비교 분석하였다.

본 논문의 구성은 다음과 같다 1장의 서론에 이어 2장에서는 본 논문에서 제안하는 개인 마켓 실행 환경인 모바일 브라우저에 대해 간략하게 설명하며, 개인마켓구현에 사용한 웹킷에 대해 살펴본다. 3장에서는 OpenID를 이용하여 개인 마켓을 구현하며 4장에서는 본 논문에 제안한 방식과 기존 소켓 방식을 개발 비용측면에서 비교 분석한다. 마지막으로 5장에서는 결론을 제시한다.

2. 관련 연구

이번 절에서는 개인마켓 실행 환경인 모바일 브라우저에 대해 알아보고, 하이브리드 웹 응용의 핵심인 웹킷 엔진에 대해 설명한다. 웹킷의 효용성에 대한 이해를 돕기 위해 웹킷의 렌더링 방식에 대해 살펴본다. 웹킷의 렌더링 방식을 이해하면 3장의 개인 마켓 구현에 사용하는 API가 웹 페이지를 어떠한 방법으로 모바일 화면에 최적화시켜 보여주는지 쉽게 이해할 수 있다.

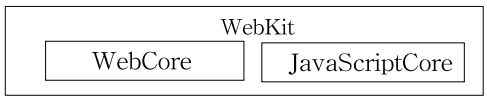
2.1 모바일 브라우저

아이폰은 스마트폰 시장에 큰 열풍을 일으키고 있다. 미국 시장조사기관 NPD그룹의 2008년 11월 발표에 의하면 $200정도 되는 3G 아이폰이 690만대 팔리면서 시장판매율 1위로 선정됐다[3]. 아이폰이 시장 점유율 1위를 기록하게 된 배경은 Wi-Fi나 무선인터넷을 통해서 내장된 사파리 모바일 웹브라우저[4]로 PC급 풀 브라우징이 실현되었기 때문이다. 아이폰 열풍에 맞서 안드로이드 스마트폰 G1[5]이 2007년 개발되었다. 안드로이드폰의 장점은 구글의 서비스를 PC처럼 거의 그대로 사용할 수 있다는 점이다. Gmail이나 구글 캘린더, 구글 맵스, 구글 어스 등 다양한 구글웨어들을 PC에서 사용하듯 안드로이드폰에서 사용할 수 있다. 이러한 서비스는 구글에서 만든 웹 브라우저인 크롬(Chorme)[6]의 모바일 버전이 탑재되었기 때문에 가능하게 되었다. 또한 크롬은 사파리와 동일한 웹킷 엔진을 사용하고 있어서 아이폰과 비슷한 수준의 풀 브라우징이 가능하다.

웹킷은 안정적인 인터넷 연결과 함께 뛰어난 모바일 웹 환경을 제공하는 웹 응용 개발 API이다. 아이폰이 매우 짧은 시간에 각광받는 모바일 웹 클라이언트로 성장한 것도 웹킷 엔진을 아이폰 플랫폼에 탑재했기 때문이다[7]. 다음은 웹킷에 대한 전반적인 구조와 렌더링 방식을 알아본다.

2.2 웹킷

웹킷은 KDE(K Desktop Enviroment)의 KHTML[8] 소스를 기반으로 애플, 노키아, 어도비, 구글 등에 의해 공동개발중인 오픈소스 기반의 브라우저 응용프로그램 엔진이다. 웹킷의 가장 큰 장점은 AJAX[9]를 지원한다는 것이다. AJAX지원 이전 기존의 웹 응용은 브라우저에서 웹서버에 데이터를 요청하고 되돌려 받을 때 이미 받은 적이 있는 데이터를 또 전송 받음에 따라 많은 대역폭의 낭비를 유발한다. 반면 AJAX 응용은 필요한 데이터만을 웹 서버에 요청하여 받은 후 클라이언트에서 데이터에 대한 캐싱(Caching)처리를 할 수 있어서 웹 서버의 부하를 줄여 줄뿐만 아니라 처리 속도도 빨라진다. 또한 HTML 4.01, CSS 2.0, JavaScript 1.5를 충실히 지원하고 XMLHttpRequest 객체도 지원한다. 그림 1은 웹킷의 구성요소를 보이고 있다. WebCore는 HTML 페이지 해석 및 렌더링을 담당하는 핵심이고, JavaScriptCore는 자바스크립트 해석을 담당한다.



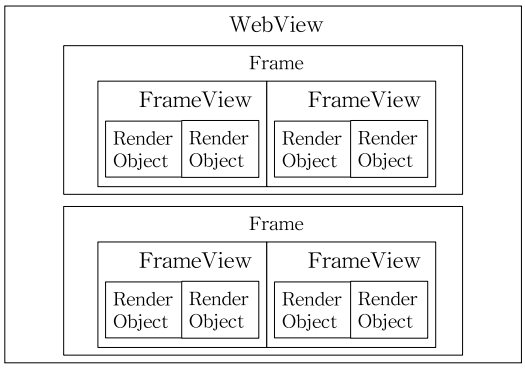
(그림1) 웹킷 구성 요소

2.3 웹킷 렌더링

웹킷은 메모리를 효율적으로 사용하고 디바이스에 임베디드 하기에도 좋으며, 전체적으로 간결

성을 유지하고 있어 새로운 브라우저 개발에서 베이스 코드로 사용하기에 적합하다. 웹킷 렌더링 클래스는 C+RTTI(RunTime Type Information)기법을 사용하여 오브젝트를 그린다. RTTI란 C++ 컴파일러 내에 포함되어 있는 기능으로서, 오브젝트의 유형을 실행 시에 결정할 수 있도록 허락하는 방식이다. 이 방식을 이용하면 프로그램 개발 시 사용하는 코드의 양이 많이 줄어들고 간단해지는 장점이 있다.

그림 2는 웹킷 렌더링 클래스 구조를 보여준다. 전체 구조는 WebView라는 최상위 클래스가 있다. 이 클래스는 웹 전체를 제어하는 클래스이다. 하위 클래스로 Frame은 웹의 페이지 내부 레이아웃을 분할하는 역할을 하는데 여러 FrameView로 구성되어있다.



(그림 2) 웹킷 렌더링 구조

RenderObject는 각 오브젝트인 텍스트, 이미지, 테이블, 등의 베이스 클래스로 공통의 인터페이스를 정해놓은 클래스이다. 웹 화면에 보여줄 문서나 이미지 정보가 도착하면 RenderObject를 상속받아 화면에 최적화 형태로 크기를 변경하여 최종 웹 콘텐츠를 그리게 된다.

다음 장에서는 앞서 살펴본 웹킷 렌더링 엔진을 이용하여 안드로이드환경에서 하이브리드 웹 응용을 구현해 본다.

3. 웹킷을 이용한 웹 응용 구현

앱스토어의 열풍과 관련 어플리케이션이 개발이 인기를 모으면서 복잡한 등록과정, 다양한 플랫폼, 다운로드의 복잡함 등 단점과 한계가 노출돼 자연스럽게 모바일 웹 응용 개발이 대안으로 부각되고 있다. 모바일 웹 응용 프로그램은 단일한 환경에서 HTML과 자바스크립트, CSS 만으로 응용을 개발할 수 있고 기존 웹 개발자들이나 웹 디자이너들도 쉽게 제작할 수 있는 등 제작비가 상대적으로 저렴하고 개발속도도 빠르다는 것이 장점이다.

3.1 하이브리드 웹 응용

모바일 응용 프로그램 종류는 내장형, 웹, 하이브리드로 분류할 수 있다. 내장형 응용 프로그램은 각 단말기에서 사용할 수 있는 단독 응용으로 해당 단말기의 기능을 효과적으로 활용할 수 있다는 장점이 있지만, 많은 단말기를 지원해야 할 경우 각각 별도로 개발해야 한다는 문제점과 응용의 재활용 및 업그레이드가 불편하다는 단점이 있다. 웹 응용의 경우 별도의 설치 없이도 계속 업그레이드된 기능을 활용할 수 있고, Open API등을 통해 손쉽게 매시업 할 수 있다는 장점을 갖는 반면, 단말기기의 모든 기능을 다 활용할 수는 없고, 모바일 기기에 탑재된 브라우저 성능에 따라 표현의 한계가 있다는 단점이 있다. 이에 두 응용 프로그램의 장점을 가질 수 있도록 하며, 보다 쉽고 빠르게 응용 프로그램 개발을 할 수 있도록 하기 위해 표 1과 같이 내장형과 웹 응용 프로그램을 합성한 하이브리드 웹 응용 프로그램이 등장하고 있다[10].

<표 1 >모바일 응용의 유형별 기능 비교

	내장형 APP	웹 APP	하이브리드 APP
그래픽 성능	상	하	상
앱스토어 판매	가능	불가능	가능
오프라인 지원	가능	일부 가능	가능
매시업	불가능	가능	가능
단말기 기능이용	용이	불가능	가능
SW 업데이트	재설치	사용 중 수정	부분 재설치
UI 제작·표현	상	하	중

본문에서 안드로이드를 실험환경으로 선택한 것은 안드로이드 SDK는 어떠한 컴퓨터에서도 설치가 가능하기 때문이다. 본 논문에서는 안드로이드폰 상에서 하이브리드 형태로 개인용 안드로이드 앱스토어를 구현하였다.

3.2 안드로이드용 개인 마켓 구현

안드로이드용 응용을 개발할 때 장점은 구글 혹은 다른 웹사이트에서 제공하는 오픈 API 서비스를 이용할 수 있다는 것이다. 애플에서도 오픈 API를 이용할 수 있는 라이브러리를 제공하고 있지만 오픈 소스가 아니기 때문에 응용프로그램 개발 시 한계가 있다. 이번 장에서는 구글에서 제공하는 OpenID[11]인증 기법을 이용하여 '개인용 안드로이드 마켓' 응용을 구현한다.

3.2.1 OpenID 다이얼로그를 이용한 인증

OpenID는 개인 인증을 위한 사용자 중심 분산형 공개 표준 기술이다. OpenID에서 ID는 웹 사이트처럼 하나의 URL로 표현되며 누구나 인터넷상에서 자신을 식별하게 해준다. URL은 웹 아키텍쳐의 가장 핵심이기 때문에, 사용자 중심 개인 인증을 위한 단단한 토대를 제공한다. OpenID의 장점은 웹 사이트 가입 시 필요한 개인정보를 번거롭게 반복적으로 입력하는 회원가입 과정이 없어도 된다는 점이다. 사용자가 OpenID 로그인을 제공하는 웹사이트에 로그인을 하려면 일반적인 사이트에서 아이디와 비밀번호를 입력해야 하는 것과는 달리, 자신의 OpenID를 입력하면 된다.

OpenID를 작동방식은 다음과 같다. 입력 받은 ID를 실제 URL 주소(http://...) OpenID로 변형한 뒤 그 URL이 가리키는 웹 페이지를 요청하고 HTML link 태그를 통해서 ID제공 서버가 http://openid-provider.org/openid-auth.php임을 알아낸다. 여기서 ID제공 서버란 OpenID URL 등록과 OpenID 인증을 제공하는 서버를 말한다.

안드로이드 SDK에서 제공하는 Dialog & Activity 관련 클래스를 이용하여 OpenID 입력 상자를 구현해 보자. 표 2는 OpenID 입력 상자 구현에 사용된 Alert Dialog 및 Activity 클래스의 주요 멤버함수들을 보이고 있다.

<표 2>Dialog 및 Activity 클래스함수 정의

method	description
setIcon (int iconId) setIcon (Drawableicon)	Activity를 위한 Window 리턴
setTitle(int titleId) setTitle(Char title)	title 설정
setPositiveButton (int textId, OnClickListener( ))	OK 버튼이 눌려 졌을 때 처리
showDialog(int id)	화면에 dialog 표시

그림 3은 이들 메소드를 이용하여 OpenID 다이얼로그를 구현한 수도 코드를 보이고 있다.

ID/PW 다이얼로그상자에서 사용자에게 OpenID와 비밀번호를 입력받은 뒤 https://www.google.com/accounts/ClientLogin에서 로그인 인증을 확인 받으면 되는데, 그림 4는 Open ID 인증을 위한 소스 코드다. 그림 5는 안드로이드 에뮬레이터 상에서 실행시킨 OpenID 다이얼로그 화면을 보이고 있다.

```
public void AuthDlg(){
ADlg 이름을 가진 다이얼로그 생성
ADlg.setTitle("Authentication");
        //ADlg 타이틀 정보 저장
editEmail =
(EditText)qView.findViewById(R.id.email);
 //이메일 입력받을 Text 박스 만들기
editPw =
(EditText)qView.findViewById(R.id.passwd);
 //패스워드 입력받을 Text 박스 만들기
 ADlg.setPositiveButton("Authenticate",
                newOnClickListener(){
ADlg 다이얼로그의 ok버튼 설정 함수
public void onClick(DialogInterface dialog,
                int which){
// ADlg 다이얼로그의 확인 버튼 클릭 후
  정보 전달 함수
GetAuth(email, editPw.getText().toString());
 // OpenID 인증 위한 정보 전달 함수 }    });
ADlg.show();  }
```

(그림 3) OpenId 다이얼로그 수도 코드

```
HTTPRequestHelperhelper= new
HTPRequestHelper((responseHandler);
helper.performPost(HTTPRequestHelper.MI
ME_FORM_ENCODED,
 https://www.google.com/accounts/ClientLog
 in", null, null, null, params);
```

(그림 4) OpenId 인증 소스 코드

(그림 5) 개인 앱스토어 메인 화면

Toast는 안드로이드 SDK에서 제공하는 팝업 객체로 OpenID 인증이 성공했는지 실패했는지 사용자에게 알려주기 위해 사용한다. Toast는 일정 시간 후 자동으로 없어지므로 기존 작업하던 것에 전혀 영향을 주지 않는 장점이 있다. 그림 6은 Toast 메시지 코드이며, 그림 7의 하단 네모박스가 에뮬레이터에서 보이는 Toast 팝업창이다.

```
Tost toast =
Toast.makeText(ct,"Authentication uccess",
Toast.LENGTH_SHORT);
 //팝업 객체 toast 는 'Athentication'제목으로
 짧은 시간 안에 나타났다 사라진다.
 toast.show(); //실제 화면에 toast 보이기
```

(그림 6) Toast 메시지 코드

(그림 7) Toast 메시지 화면

3.2.2 웹뷰(webview)

웹 페이지를 모바일용 웹 브라우저에서 보기 위해서는 웹킷 렌더링 엔진을 사용하여 andoird.webkit.webview 메소드를 사용하면 된다. 웹뷰는 웹킷 렌더링 엔진을 구현하고 있는 클래스로써 이를 사용하기 위해서는 AndroidMainfest.xml에 인터넷 접속 및 웹 페이지 로드를 위한 설정을 그림 8과 같이 추가해야 한다.

```
<uses-permission android:name=
            "android.permission.INTERNET">
```

(그림 8) AndroidMainfest.xml 권한설정

표 3은 개인 앱스토어 구현에 사용된 웹뷰 메소드와 그 기능을 요약하고 있다.

<표 3> 웹뷰 메소드 및 기능요약

method	description
loadUrl(String url)	주어진 url로드
onDownloadStart (url, userAgent, contentDisposition, mimetype, contentLength)	웹 서버에서 제공하는 파일 다운

그림 9는 표 3에 있는 메소드를 이용하여 화면에 웹 페이지를 표시하고 파일을 다운로드하기 위한 수도 코드를 보이고 있다. 그림 9를 실행시키면 웹킷 엔진에 의해 렌더링 되어 그림 10과 같은 화면이 나온다. OnDownloadListener() 함수를 이용하여 서버에서 제공하는 안드로이드 응용 프로그램 및 음악, 그림을 다운받을 수 있다.

```
webView = 변수선언;
webView.loadUrl(웹서버 주소 설정 );
webView.setDownloadListener()
{ onDownloadStart (내려받을 파일 주소,
                   파일 타입 설정,
    내려 받을 파일 저장하는 위치 설정)}}
```

(그림 9) 화면 로딩 및 다운로드 수도 코드

(그림 10) 웹뷰 메소드를 이용한 화면

4. 네이티브 응용과 하이브리드 웹 응용 개발비용 평가

일반적인 안드로이드용 모바일 웹 응용은 서버와 클라이언트로 개발된다. 서버는 다양한 언어를 이용하여 개발되지만 클라이언트는 안드로이드 SDK로 개발해야 한다. 이번 장에서는 소켓과 웹킷을 이용하여 개인마켓을 구현할 때 꼭 필요한 과정을 개발비용 측면에서 비교 분석한다.

4.1 서버연결 과정

소켓기반 서버 연결은 소켓 생성, 닫기, 재생성, 폐기 함수를 기본으로 구성된다. 표 4는 소켓 및 웹킷 서버 연결과정을 비교한 것이다. 소켓 연결에서 onCreate()는 setSocket() 함수를 호출하여 서버에 연결한다. 하지만 웹킷 API인 loadUrl()을 이용하면 소켓보다 더 간결하게 서버에 연결 할 수 있다.

<표 4> 소켓과 웹킷 서버연결과정 비교

소켓서버연결과정	private String ip = "xxx.xxx.xxx.xxx";
	private int port = 9999; // PORT번호
	protected void onStop() {
	super.onStop();
	try { socket.close(); }
	catch (IOException e)
	{ e.printStackTrace();} }
	
	public void onCreate() {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);
	mHandler = new Handler();
	try { setSocket(ip, port); }
	catch (IOException e1) {
	e1.printStackTrace();}
	checkUpdate.start();
	final EditText et =
	(EditText) findViewById(R.id.EditText01);
	Button btn =

	(Button) findViewById(R.id.Button01);
	final TextView tv = (TextView)
	findViewById(R.id.TextView01);
	OnClickListener(newOnClickListener() {
	public void onClick(View v) {
	if(et.getText().toString()!=null||
	!et.getText().toString().equals(""))
	PrintWriter out =
	new PrintWriter(networkWriter, true);
	String return_msg=.getText().toString()
	out.println(return_msg); }}}); }
	
	private ThreadcheckUpdate =newThread()
	public void run() {
	try { String line;
	Log.w("ChattingStart", "Start Thread");
	while (true) {
	Log.w("Chatting is running",
	"chatting is running");
	line = networkReader.readLine();
	html = line';
	Handler.post(showUpdate);}}
	catch (Exception e) {}} };
	private Runnable showUpdate = new
	Runnable() {
	public void run() {
	Toast.makeText(NewClient.this,
	"Coming word: " + html,
	Toast.LENGTH_SHORT).show(); }
	};
	public void setSocket(String ip,int port)
	throws IOException {
	try { socket = new Socket(ip, port);
	networkWriter =
	new BufferedWriter(new Output
	Stream
	Writer(socket.getOutputStream()));
	networkReader =
	new BufferedReader(new InputStream
	Reader(socket.getInputStream()));
	} catch (IOException e) {
	System.out.println(e);
	e.printStackTrace();}}

웹킷	public void onCreate(Bundle icicle)icicle) {
	super.onCreate(icicle);
	setContentView(R.layout.main);
	변수=(WebView)findViewById(R.id.webkit);
	변수.loadUrl("주소"); }

4.2 파일 다운로드 과정

서버연결 후에는 사용자가 원하는 파일을 내려 받도록 설정해야 한다. 소켓에서는 inputStream()과 fileOutputStream()을 이용한다. 가장 중요한 부분은 안드로이드에서 제공하는 openFileOutput()을 이용해야 편하게 outputStream()을 얻을 수 있다. 표 5는 소켓과 웹킷을 이용한 파일 다운로드 과정의 코드를 보이고 있다. 소켓 파일 다운로드 과정에서는 파일의 끝까지 내려 받고 있는지 확인하는 코드를 삽입해야한다. 그렇지 않으면 계속해서 파일을 전송하게 되므로 서버에 과부하가 생기게 된다. 웹킷에서 제공하는 onDownloadSta rt()를 사용하면 아주 간단하게 구현 할 수 있다.

<표 5>소켓,웹킷 파일다운로드 과정 비교

소켓	InputStream inputStream = null;
	FileOutputStream fileOutputStream = null;
	byte[] buf = new byte[100];
	try { inputStream = new URL
	("http://xxx.xxx.xxx.xx/target.dat").openStream(
	
	String file_name = "result.dat";
	fileOutputStream = penFileOutput(file_name,
	MODE_WORLD_READABLE);
	int cnt = 0;
	while((cnt = inputStream.read(buf)) != -1) {
	fileOutputStream.write(buf , cnt);
	fileOutputStream.flush();}}
	catch (MalformedURLException e) {}
	catch (IOException e) {} finally
	{
	try {
	if(inputStream != null)
	inputStream.close();
	if(fileOutputStream != null)
	fileOutputStream.close();
	} catch(IOException ie) {} }

웹킷	public void onDownloadStart
	(String url, String userAgent,
	contentDispositionString mimetype,
	long contentLength) {
	if(contentDisposition==null
	l|!contentDispositionnu.regionMatches
	(true,0, "attachment", 0, 10)) {
	Intent intent =
	newIntent(Intent.ACTION_VIEW);
	intent.setDataAndType(Uri.parse(url),
	mimetype); }

4.3 인증과정

3.1절에서 언급했듯이 안드로이드 폰은 Open API를 제약 없이 사용할 수 있다. 구글은 Open ID API를 제공하기 때문에 간편하게 회원 가입 및 로그인 서비스를 구현할 수 있다. 기존 웹 서버에서는 JSP, PHP, ASP 등으로 회원가입 및 로그인 서비스를 개발하였고 회원정보를 저장하기 위해 MySql과 같은 DB서버까지 구축해야 한다. 표 6은 기존 웹 서비스에서 PHP로 구현한 회원 가입 및 로그인 소스와 OpenID 이용한 소스를 비교하고 있다.

소켓을 이용한 응용 개발은 경험이 풍부한 개발자라면 쉽게 사용할 수 있지만 개발 초보자가

<표 6> 인증과정 비교

기존 웹서비스 회원가입인증 및 로그인확인과정	//회원가입확인 php
	if(!$name){ echo("<script>
	window.alert('이름을 입력해 주세요..')
	history.go(-1) </script>");exit‘ }
	if(!$id){ echo("<script>
	window.alert('아이디를 입력해 주세요..')
	history.go(-1)</script>");exit; }
	if(!$passwd){ echo("<script>
	window.alert('패스워드를입력해주세요..')
	history.go(-1)</script>");exit; }
	elseif($passwd!=$passwd_confirm){
	echo("<script>
	indow.alert('패스워드확인이틀렸습니다..')
	history.go(-1)</script>");exit; }
	if(!$sex){ echo("<script>
	window.alert('성별을 입력해 주세요..')
	history.go(-1)</script>");exit;}
	if((!$birthday_y)||(!$birthday_m)||(!$birthday_d)||(!$birthday_cal)){
	echo("<script> window.alert('생년월일을 입력해 주세요..')
	history.go(-1)</script>");exi;}
	$connect=mysql_connect('localhost','windowdeco','im7yulun')ordie('접속 실패');
	mysql_select_db('windowdeco',$connect)ordie('데이타베이스가 없음.');
	//주소를하나로합침
	$address=$zipcode.$home_address1.' '.$home_address2;
	//전화번호를 하나로 합침
	$telephone=$telephone1.$telephone2.$telephone3
	$mobilephone=$mobilephone1.$mobilephone2.$mobilephone3;
	$sql="insertintomember(name,id,password,sex,birthday_y,birthday_m,birthday_d,
	birthday_cal,academy,job,job_addr,telephone,
	mobilephone,email,address)
	values('$name','$id','$passwd','$sex','$birthday_y','$birthday_m','$birthday_d',
	'$birthday_cal','$academy','$job','$job_addr','$telephone','$mobilephone','$email','$address')"; mysql_query($sql,$connect);
	mysql_close($connect);

	//로그인 검사 php
	function ($name,$value,$expire,$path='/'){
	if(headers_sent()){
	$cookie=$name.'='.urlencode($value).':';
	if($expire)$cookie.='expire='.gmdate('D,dMYH:i:s',$expire).'GMT';
	echo'<scriptlanguage="javascript">document.cookie="'.$cookie.'";</script>';}
	else{setcookie($name,$value,$expire,$path);}
	}
	if(!$id){echo("<script>
	window.alert('아이디를 입력해 주세요.')
	history.go(-1) </script>");exit; }

	if(!$passwd){ echo("<script>
	window.alert('패스워드를 입력.')
	history.go(-1) </script>");exit; }
	include"../function.php";
	//DB접속
	$connect=dbconn();
	//member테이블에서 사용자 정보 조회
	$sql="select*frommemberwhereid='$id'";
	$result=mysql_query($sql,$connect);
	//대상회원정보 조회 |
	result_data=mysql_fetch_array($result);
	//입력된 아이디가 존재하지 않는경우
	if(!$result_data[name]){ echo("<script>
	window.alert('등록되지 않음.')
	history.go(-1) </script>");exit; }
	//아이디존재패스워드가 일치않는 경우
	else{
	$original_passwd=$result_data[password];
	if($original_passwd!=$passwd){
	echo("<script>
	window.alert('잘못된 패스워드 입니다.')
	history.go(-1) </script>");exit; }
	//아이디와 패스워드가 모두 유효한 경우
	else{
	$passcode=passwd('$result_data[password]'); $name=$result_data[name];
	mysql_close($connect);
	echo("<metahttp-equiv='Refresh'content='0;URL=../index.php?name=$name'>"); } } ?>

OpenID	HTTPRequestHelperhelper=new HTTPRequestHelper((responseHandler); helper.performPost(HTTPRequestHelper, MIME_FORM_ENCODED, "https://www.google.com/accounts /ClientLogin", null, null, null, params);

이용하기에 어려운 면이 있다. 이러한 단점은 앞서 구현에서 이용했던 웹킷을 이용하면 손쉽게 해결할 수 있다. 코드 길이를 비교해 봐도 웹킷을 이용하여 구현하는 방법이 간결하면서 훨씬 쉽다. 또한 웹킷이 제공하는 터치 인식, 화면 크기 조절, 제스처 인식 등의 다양한 기능들을 사용할 수 있어서 강력한 서비스를 구현할 수 있다[12].

4.4 하이브리드 응용 서비스 개발 사례

스마트폰 열풍으로 은행에서는 스마트폰 뱅킹 서비스를 시행 중에 있다. 스마트 뱅킹 서비스의 특징 중 하나는 웹과 응용을 적절히 혼용한 하이브리드 방식의 뱅킹 서비스라는 점이다. 모바일 웹은 다양한 정보를 보여줄 수 있다는 장점이 있는 반면, 이동통신사 통신망 상에서 거래 속도가 늦고 데이터 통화량이 많이 발생하여 요금 부담이 생길 수 있다는 단점도 있다. 반면 네이티브 응용 방식은 스마트폰 뱅킹 설치 초기에 프로그램을 다운로드 받고나면 금융거래 전용

통신에서만 활용되기 때문에 업무 처리 속도가 빠르고 데이터 통화량이 적다는 장점이 있다. 반면 다양한 정보를 보여주기에는 UI 구성이 쉽지 않기 때문에 네이티브 앱은 계좌이체나 상품 가입 등 일부 중요 거래 업무 중심으로 사용된다. 따라서 하이브리드 응용 서비스가 스마트 뱅킹 서비스에서 많이 사용되고 있다. 하이브리드 스마트 뱅킹 앱은 모바일 웹 서비스 보다 데이터 통화량에서 유리하면서도 모바일 웹처럼 다양한 정보를 보여주기 쉽다. 아울러 초기 응용 서비스 설치 시 내려 받게 되는 데이터 양과 서버와의 접속 빈도 등을 고려할 때 통신 트래픽도 그리 많지 않아 통신비 부담도 그리 크지 않은 편이다. 게다가 통신사에서 데이터사용 요금에 대한 다양한 요금제가 나오고 있기 때문에 사용자들의 부담은 점차 줄어들 것으로 전망한다[13].

5. 결과 및 향후 연구

지난해 말부터 불어 닥친 스마트폰 열풍에 힘입어 응용 개발에 대한 관심이 집중되고 있다. 특히 스마트폰 사용자가 직접 응용을 만들어 사용하고 판매 할 수 있어 개발 초보자들도 다양한 응용개발에 나서고 있다. 기존 네이티브 응용은 스마트폰이 가진 부가적인 기능을 사용하여 참신하고 기발한 응용을 선보였지만. 오프라인의 한계성으로 발 빠르게 변하는 스마트폰 응용시장에 능동적으로 대처하기 힘들다. 본 논문에서는 네이티브 응용의 단점을 보안한 하이브리드 웹 응용개발을 제안했다. 웹 브라우저 상에서 실행시킬 수 있어 실시간 SW업데이트가 가능하며 웹킷 엔진 채용으로 스마트폰 환경에서도 제약없는 웹 표현이 가능하다. 또한 스마트폰 내장 기능을 자유롭게 사용할 수 있는 점이 강점이다. 그리고 하이브리드 웹 응용을 300~400줄에 달하는 복잡한 소켓 방식으로 구현하는 방법 이외에도 웹킷을 이용한 간편하고 빠르게 3~4줄로 압축하여 구현할 수 있어, 본 논문에서는 웹킷을 이용한 하이브리드 웹 응용의 효율성을 제시하였다. 스마트폰 사용자가 증가하면서 사용자에게 필요한 맞춤형 앱 개발이 활성화 되고 있다. 응용 개발은 프로그램 전문가뿐만 아니라 새로운 아이디어를 가지고 있다면 누구든지 만들어 활용 할 수 있도록 개발자 영역을 넓혀 갈 것이다. 앞으로 하이브리드 웹 응용은 많은 스마트폰 내부기능을 웹상에서도 효과적으로 이용할 수 있도록 개발되어야 한다.

참 고 문 헌

[1] 김민식,정현준 “휴대폰 산업의 탈추격형 대응전략”,정보통신정책제 22권 1호.

[2] 이고은, “구글 안드로이드폰 Personal 오픈 마켓을 위한 클라이언트/서버 통신 모듈 설계”,한국멀티미디어학회, 2009

[3] 앱스토어, “http://www.apple.com”

[4] 사파리 모바일 브라우저, “http://developer.apple.com”

[5] G1폰 홈페이지 ,“ http://www.htc.com”

[6] 김종대, “모바일 시장에 부는 기회의 바람,앱스토아”, LGERI리포트, 2008.8.19

[7] KHTML,"http://www.konqureor.org"

[8] AJAX, "http://www.ajax.org"

[9] 리토마이어, Professional android application development, 제이펍, 2009.

[10] 전종훈, 이승윤,“차세대 모바일 웹 애플리케이션 표준화 동향”, 전자통신동향분석 제25권 , 2010.2

[11] OpenID, "http://ko.wikipedia.org/wiki/OpenID"

[12] 모바일 컨텐츠 이야기 , http://mobizen.pe.kr/844

[13] 스마트폰 뱅킹, 은행마다 속도 다른 이유는? http://www.ciobiz.co.kr/news/articleView.html?idxno=3005

이 고 은

1989년～1995년: 포항공과대학교 전자계산학과 (학사, 석사)

1995년～1999년: 포항공과대학교 컴퓨터공학과 (공학박사)

1999년～2005년: (주)알티캐스트 방송서버 개발, 품질보증 팀장

이 종 우

1990년 : 서울대학교 컴퓨터공학과 (학사)

1992년 : 서울대학교 컴퓨터공학과 대학원(석사)

1996년 : 서울대학교 컴퓨터공학과 대학원(박사)

1996년～1998년 : 현대전자㈜ 정보시스템사업본부 과장

1998년～1999년 : 현대정보기술㈜ 책임연구원

1999년～2002년 : 한림대학교 정보통신공학부 조교수

2002년～2003년 : 광운대학교 컴퓨터공학부 조교수

2003년～2004년 : 아이닉스소프트㈜ 개발이사

2004년～현 재 : 숙명여자대학교 정보과학부 멀티미디어과학전공 부교수

2008년 : 뉴욕주립대 스토니브룩 Research Scholar

관심분야 : Mobile System Software, Storage Systems, Computational Finance, Cluster Computing, Parallel and Distributed Operating Systems, Embedded System Software