스마트폰을 위한 웹콘텐츠의 효율적 접근 방법에 관한 연구

김승리 양정진
가톨릭대학교
Akenasmi@gmail.com, jungjin@catholic.ac.kr

Efficient Approaches to Web Contents for Smart Phones

SeungLee Kim JungJin Yang
School of Computer Science and Information Engineering
The Catholic University of Korea

요 약

스마트폰은 특정공간에 존재해야지만 네트워크에 접속할 수 있다는 데스크탑 컴퓨터의 한계를 극복하였다. 자신이 원하는 정보를 장소에 구애 받지 않고 매우 쉽게 얻어 낼 수 있게 되었고, 자신과 관계된 사회구성원과 항상 연결될 수 있게 되었다. 반면 웹 서비스나 웹콘텐츠에서의 데이터의 수정, 변화는 불가피해졌다. 본 논문에서는 스마트폰의 어플리케이션에 웹 컨텐츠를 어떻게 가져 오는것이 성능면에서, 효율적인면에서, 프로그래머의 편이성면에서 유용할 것인지 연구하고자 한다.

1. 서 론

안드로이드폰, 아이폰에서는 현재 존재하고 있는 웹페이지의 데이터를 편집하여 정보를 사용자에게 제공하는 어플리케이션이 아주 많다. 대표적으로 YouTube, smartSMS 등등이 있다. 전체적으로 보면 아이폰에서는 3G 데이터를, wifi를 이용하여 웹콘텐츠에 접속하여 사용자가 필요하는 데이터를 획득하는 어플리케이션이 약 90%에 이른다. 이들의 서비스를 위해서 프로그래머는 필수적으로 현재의 데스크탑 컴퓨터를 위한 네트워크상의 데이터(XML)를 수정하여, 직접적으로 전달거나, 기존의 웹 컨텐츠(HTML)를 수정하여 서버에 접속하지 않고, 데이터를 가져오는 형태를 가질 수 있다.

본연구에서는 이런 웹콘텐츠를 가져오기 위해서 어떤 방법을 사용할수 있는지에 대해서 논하려고 한다. 첫째, 모든 웹브라우저에서 통용되고 있는 JavaScript를 이용하는 방법과 둘째, Hpple Pasing을 이용한 방법(Hpple의 선택 이유는 다음과 같다. JavaScript와 같이 웹페이지의 모든 데이터가 필요하다는점,Hpple는 DOM(Document Object Model)형태의 XML파싱을 이용한다. JavaScript와 비슷한 구현 방법을 갖는다는점,XML파싱에서 비교적 평균적인 퍼포먼스를 낸다는 점이다.[1]) 두가지를 사용하여 데이터를 가져올 것이다. 또한 이 두가지 기법의 성능을 비교, 어느것이 효율적인 방법인지에 대해서도 알아 볼것이다[5].

2. 관련 연구

① MacOS X와 아이폰 SDK

아이폰 SDK는 MacOS X 10.5.3 이상의 버전에서 만 설치할 수 있다. 그리고 아이폰 SDK를 사용할 때 가급적 인 텔 CPU를 탑재한 매킨토시에서 개발하기를 권장하고 있다. 아 이폰 애플리케이션 개발에 주로 사용되는 프로그래밍 언어는 Objective-C 언어이다. Objective-C는 C 언어 기반에 Smalltalk 의 객체지향 기능을 도입한 프로그래밍 언어로서 애플의 OS에서는 공통적으로 사용할 수 있는 언어이다. 그리고 아이폰 개발을 위해 사용되는 Objective-C는 표준적인 언어이기에 꼭 MacOS에서만 컴파일되는 것은 아니다. 그래서 Unix/Linux에서도 프로그래밍하고 컴파일할 수는 있지만, 개발하면서 필요한 디버깅용 아이폰 에뮬레이터가 정상적으로 동작하지 않는 점 때 문에 매킨토시에서만 개발할 수밖에 없다. 그렇지 않더라도 Cocoa는 애플에 종속적인 라이선스를 갖고 있어서 특별한 일이 없는 한 아이폰 에뮬레이터는 다른 OS에서 구동될 일이 없을 것 이다. 아이폰 SDK를 설치하면 개발용 IDE로서 같이 설치되는 Xcode가 있다. Xcode는 MacOS의 개발에도 사용되는 유용한 개발도구인데, 아이폰OS가 발표되면서 개발자를 위한 아이폰 SDK에 함께 포함시켜 설치에 용의하도록 지원하게 되었다. 이 처럼 개발 플랫폼의 표준화로 기존에 매킨토시용 프로그램을 개 발하던 사람들에게는 별다른 거부감 없이 아이폰 개발에도 참여 할 수 있다는 게 장점이라 할 수 있다.

② 웹키트(WebKit)

웹키트(WebKit)는 웹 브라우저를 만드는 데 기반을 제공하는 오픈 소스 응용 프로그램 프레임워크이다. 웹키트는 원래 맥 오에스 텐의 사파리 웹 브라우저 엔진으로 사용하기 위해 컨커러 브라우저의 KHTML 소프트웨어 라이브러리에서 가져온 것이었다. 이 프레임워크는 이제 옴니웹, 시라, iCab, 어도비 통합 런타임, 휴대 전화(아이폰 포함), 노키아의 Series 60 브라우저, 구글의 안드로이드 플랫폼에 사용되고 있다. 웹키트의 구성은 웹코어(WebCore),자바스크립트코어(JavaScriptCore),드로세라(Drosera)로 이루어져 있고, 웹코어는 HTML, SVG를 위한 레이아웃, 랜더링, DOM 라이브러리이다[2].

자바스크립트코어는 웹키트 기능을 위한 자바스크립트 엔진을 제공하며 Mac OSX 안의 다른 환경에서 이러한 종류의 스크립팅을 제공한다. 드로세라는 웹키트 순수 빌드에 포함되어 있는 자바스크립트 디버거이다.

③ JavaScript

자바스크립트(JavaScript)는 객체 기반의 스크립트 프로그래밍 언어이다. 이 언어는 웹 사이트에서의 사용으로 많이 알려졌지만, 다른 응용프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있다.

④ Hpple

웹페이지의 데이터를 DOM(Document Object Model)형태로 데이터를 가져올수 있는, 소스 자체가 JavaScript와 매우 유사한 라이브러리 이다. HTML파일과 XML파일을 Parsing할 수 있고, Tag의 Content와 Name, Attribute에 쉽게 접근할수 있다. Objective-C의 libxml2.2의 라이브러리를 사용하였으며, 사용자가 원하는 태그에 키를 통하여 접근 할수 있게 설계 되어있다[5].

3. 실험

① 실험설계

Hpple와 JavaScript의 성능을 비교하기 위해서 첫째 접속에서부터, 데이터처리, 데이터 획득까지의 속도, 둘째, 데이터를 얻기 위해서 필요한 메모리의 양을 기준으로 선정하였다.

본 논문의 실험은 XML 데이터 Parsing, HTML 웹페이지의 데이터 Parsing을 다루었다. XML 데이터 Parsing은 구글의 날씨 API로 하였고, HTML 웹페이지 데이터 Parsing은 본 저자의 도서관인중앙도서관(Library.catholic.ac.kr)로 선정하였다. 실험의 환경적인 요인은 아래표(표 1)와 같다.

제원	성능
CPU	INTEL CORE 2 DUO 2.8GHZ
RAM	DDR3 2G
O/S	MAC OSX LEOPARD 10.5.8
IDE	XCODE 3.1.3

(표 1) 실험 성능 제원표

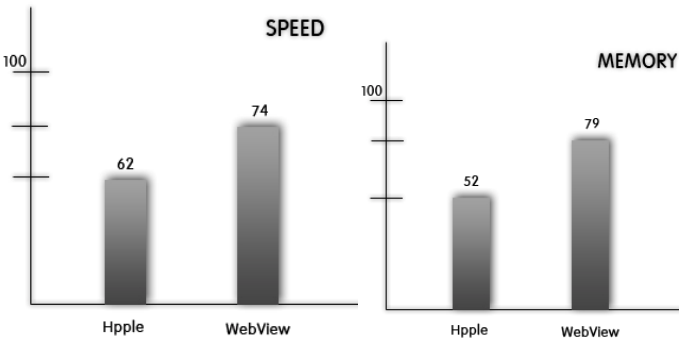
(1.1) XML 데이터 처리

```
NSData *htmlData = [[NSString
stringWithContentsOfURL:[NSURL URLWithString:
@"http://www.google.co.kr/ig/api?weather=incheo
n"]] dataUsingEncoding:NSUTF8StringEncoding];
        NSArray *elements=
            [xpathParser search:@"//humidity"];
                objectAtIndex:0];

        NSMutableDictionary * test =
           [NSMutableDictionary dictionary];
                test = [element attributes];
                mLabel3.text =
                    [test objectForKey:@"data"];
```

(소스 1) Hpple를 이용한 XML데이터 Parsing

상위는 Hpple를 이용한 XML 데이터 Parsing이다. 구글에서 제공하는 인천지역의 날씨에 대한 정보를 담은 XML파일을 Parsing하는 소스코드이다. 반면에 WebView의 JavaScript를 이용한 Parsing은 HTML DOM형태의 데이터만 처리 할수 있기 때문에[4], 위의 코드 처럼 XML파일 자체를 Parsing해 올 수 없었다. 약간의 수정이 필요하였는데, 상단과 하단에 각각 <HTML>태그와 </HTML>태그를 붙여서 데이터 자체를 바꾸어 주어야 했다. 데스크탑용 웹브라우저인 Safari나 인터넷 익스플러러자체는 XML에 대한 자체 Parsing이 가능했으나, iPhone의 웹브라우저인 UIWebView는 직접적으로 Parsing할수 있는 기능을 제한한 것으로 보인다.



(표 2) XML에서의 성능

(표2)는 Parsing 속도와 Memory의 차지 용량 비율에서 Hpple의 Parsing이 더 유용한것을 알려준다. 전체적으로 WebView 자체 클래스의 크기가 크기 때문에 처리해야할 과정이 많아서 성능의 저하가 생기고[2], 자체 기능이 JavaScript외에 여러가지가 있기 때문에 메모리의 필요 공간의 비율이 Hpple이가 더욱 효율적이라는 것을 알수 있다.

(1.2) HTML 데이터 처리

(1.2.1) Hpple

```
NSData *htmlData =
        [[NSString
stringWithContentsOfURL:[NSURL URLWithString:
@"http://library.catholic.ac.kr/jsp/sea/main/
SearchMainController.jsp?menuid=100000000103&
websysdiv=TOT"]]
dataUsingEncoding:NSUnicodeStringEncoding];
        TFHpple   *xpathParser   =   [[TFHpple
alloc] initWithHTMLData:htmlData];
        NSArray *elements  =
   [xpathParser
          search:@"//input[@id='DISP01']"];
        TFHppleElement *element =
               [elements objectAtIndex:5];
        NSMutableDictionary * test =
            [NSMutableDictionary dictionary];
        test = [element attributes];
        NSString *testtext =
              [test objectForKey:@"value"];
        mLabel3.text = testtext;
        NSLog(@"%@",testtext);
```

(소스 2) Hpple을 사용한 HTML Parsing

상위 소스코드(소스 1)는 Hpple가 HTML 데이터를 Parsing하여 처리하는 과정이다. Parsing의 방법은 비교적 쉽다. 해당 데이터의 위치를 URLWithString에 할당받고 XpathParser를 이용해 해당 데이터를 계층적 트리 구조로 만든다. 완성된 트리 구조를 검색하여 원하는 데이터에 접근하여 배열형태로 필요한 데이터만을 할당받아 TFHppleElement를 사용하여 Attribute, Name, Content로 분화한다. 함수를 이용하여 자신이 원하는 형태의 데이터를 가져올수 있게 된다.

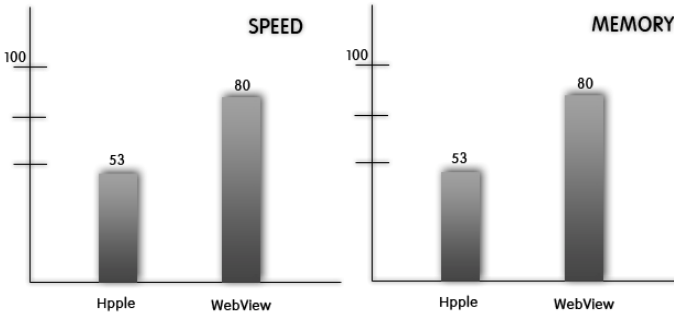
(1.2.2) WebView

(소스 3)는 웹뷰의 JavaScript를 사용하여 원하는 데이터를 가져오는 것을 나타낸다. 소스의 흐름은 다음과 같다. 접근하고자 하는 페이지의 URL을 NSURLRequest에 할당을 한다. UIWebView클래스를 이용하여 만들어진 웹뷰에 URL을 입력하여 웹페이지를 로딩한다. 웹뷰의 로딩이 끝나면, webViewDidFinishLoad 함수가 호출되고 JavaScript를 이용하여 속성인 ID가 "DISP01이라는 값을 가진 태그에 접근한다. 접근한 태그의 속성 "value"를 참조하여 NSString에 원하는 데이터를 가져오게 된다[2].

```
(void)viewdid  {
webView.delegate = self;
NSString *urlAddress =
@"http://library.catholic.ac.kr/jsp/sea/main/Sear
chMainController.jsp?menuid=100000000103&websysdi
v=TOT";
        NSURL *url =
            [NSURL URLWithString:urlAddress];
          NSURLRequest       *requestObj       =
          [NSURLRequest requestWithURL:url];

        [webView loadRequest:requestObj];
}
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UII
nterfaceOrientation)interfaceOrientation {
                return (interfaceOrientation ==
               UIInterfaceOrientationPortrait);
}
-(void)webViewDidFinishLoad:(UIWebView   *)webView
{
        NSString *Test =
 [webView stringByEvaluatingJavaScriptFromString:
    @"document.getElementById('DISP01').value"];
        NSLog(Test);
}
```

(소스 3) WebView를 이용한 HTML Parsing



(표 3) HTML에서의 성능

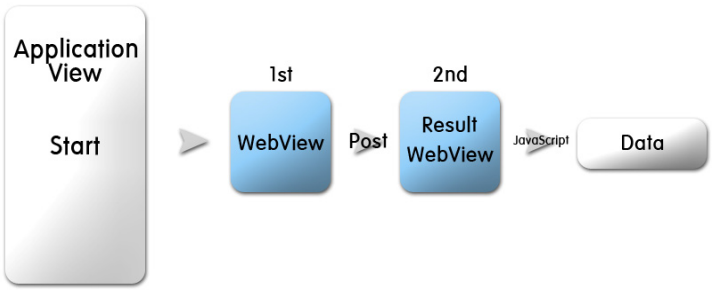
위의 데이터 처리 결과를 보면 HTML파일 Parsing에

서는 Hpple의 성능이 월등하게 좋다는 것을 알 수 있다. Hpple는 WebView에 비해서 기능상의 제한이 많다. 예를 들자면 WebView의 특징인 그래픽 적인 요소(이미지)와 연산기능(앞으로가기, 뒤로가기, 재 로딩)이 클래스 안에 구현 되어 있기 때문에 Hpple에 비해서 필요한 메모리의 양이 많아진다.스피드도 마찬가지이다. 페이지를 시각적으로 처리 해야 하므로 그림 파일등의 데이터를 다운로드 해야하는 시간이 상대적으로 오래 걸린다.

(1.3) Post처리

Post처리는 서비스 사용자가 서버에 임의의 단어를 질문하는 형태로 데이터를 전달하여 서버에게서 그에 대한 답을 받는 서비스 형태이다. 예를 들어, 구글 검색과 같은 기능이 Post처리라고 할수 있다.

이 데이터처리의 구현은 (그림 1.3.1)과 같은 구조로 진행된다.



(그림 1) Post처리의 흐름도

(1.3.1) Application View Start 부분

```
- (void)viewDidLoad {
NSString *urlAddress =
@"http://library.catholic.ac.kr/jsp/sea/main/SearchMainCon
troller.jsp?menuid=100000000101&websysdiv=TOT";
NSURL *url = [NSURL URLWithString:urlAddress];
//URL Requst Object
NSURLRequest *requestObj =
    [NSURLRequest requestWithURL:url];
        [webView loadRequest:requestObj];
```

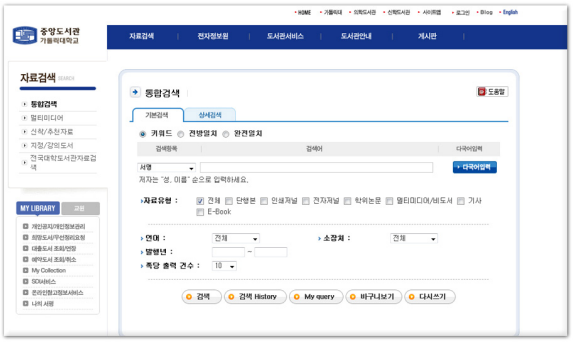
(소스 4) Applicatioin Start

어플리케이션의 시작부분이다. 어플리케이션이 실행되면 자동적으로 (void)viewDidLoad함수가 실행되고 내부의 구현부분이 처리된다. 처리는 해당 주소를 NSString 타입으로 할당하고, 해당 URL의 리퀘스트를 받기 위해서 NSURLRequest 타입에 할당한다. 웹페이지에서 받은 HTML데이터를 WebView에 할당하고 WebView의 페이지를 갱신한다.

(1.3.2) 1st WebView 부분

```
NSString *searchkey = @"마시멜로 이야기";
NSString *testkey =
 [NSString stringWithFormat:
    @"document.main.searchKeyword[0].value =
                                '%@'", searchkey];
[ webView stringByEvaluatingJavaScriptFromString: testkey];
[ webView stringByEvaluatingJavaScriptFromString:
                                @"search()"];
```

(소스 5) 검색부분



(그림 2) 검색 웹페이지

해당 웹페이지(그림 2)의 서버에 쿼리를 보내기 위해서 검색 쿼리 단어를 NSString타입에 할당한다. Post를 이용하여 쿼리를 보낼것이기 때문에, 해당 Form의 TextBox에 할당받은 검색어를 자바 스크립트를 이용하여 적용시킨다. JavaScript를 이용하여 search()함수를 실행하고 그에 대한 결과를 WebView에 받는다.

(1.3.3) 2nd WebView 부분

(그림 3) 검색 결과

```
NSString result;
NSString *lengthTable =
    [ webView stringByEvaluatingJavaScriptFromString:
                @"document.all.length"];
NSInteger Translength = [lengthTable integerValue];
if (Translength >= 225) {
result = [ webView stringByEvaluatingJavaScriptFromString:
                @"document.all[58].innerHTML"];
}
else {
result = [ webView stringByEvaluatingJavaScriptFromString:
                @"document.all[58].innerHTML"];
}
```

(소스 6) 검색 결과에서 데이터를 가져오는 소스

웹뷰에 완성된 데이터를 가져오기 위한 부분이다. 검색이 완료된 테이블(그림 3)에 있는 데이터를 가져오기 위해서는 자바스크립트의 getElementbyName을 이용해서 해당 데이터를 직접 가져오는 방법과 document.all을 이용하여 각 태그의 위치를 파악을 하고 해당 데이터를 가져오는 방법이 있다. 두가지 방법을 사용하는데 있어서 몇가지 조건이 있다. 전자는 웹페이지 프로그래머가 일일이 태그에 Name을 지정해서 만들었을 경우에 가능해 진다. 이렇게 구성된 페이지라면 JavaScript로 데이터를 가져오는 것이 더욱 쉬워진다. 후자는 자신이 가져올 데이터의 위치를 찾아야 한다. 또한 완성되는 페이지의 형태가 동적으로 변경될 경우 그에 따른 후처리를 필요로 한다.

(1.3.4) Hpple

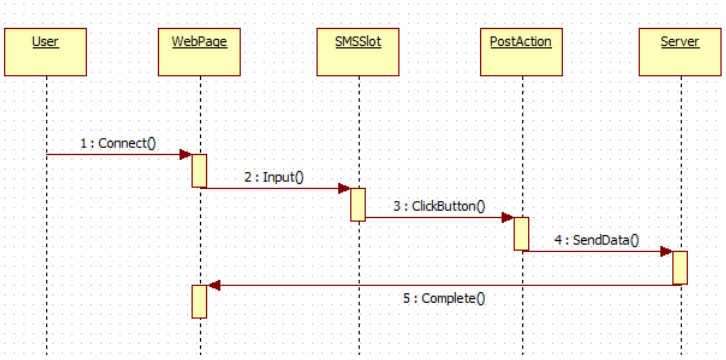
Hpple는 정적 페이지를 Parsing하는 방법이다. 이 특성상 hpple자체로는 post를 보내서 서버에서 피드백을 받아 올수 없다. Post를 사용하기 위해서 NSHTTPRequest를 사용할수 있다.

3. 사례

① SmartSMS

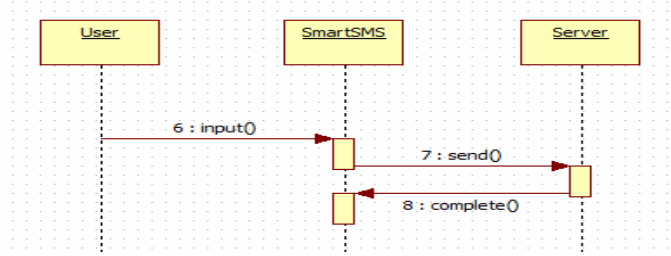
(그림 4) SmartSMS

SmartSMS(그림 4)는 Post를 이용하여 웹페이지와 상호 소통하는 형태의 아이폰 어플리케이션이다. 이 어플리케이션은 인터넷을 통해서 자신이 원하는 핸드폰에 SMS를 보낼수 있게 한다. 이 형태는 WebView를 통하여 Post를 보내는 형태로 구현할 수 있다.



(표 4) 웹페이지에서 SMS를 보내는 흐름도

데스크탑 컴퓨터로 웹페이지의 서비스를 이용하여 SMS 를 보낼 경우이다.(표 4) 사용자가 웹페이지에 접속하고 웹페이지의 서비스인 SMSSlot 에 자신이 보내려는 메시지와 상대방의 번호를 입력한다. Send 버튼을 눌러 서버에 데이터를 전송한다. 서버의 처리가 완료되면 서버는 웹페이지에 전송이 완료 되었다는 메시지를 표시해 준다.



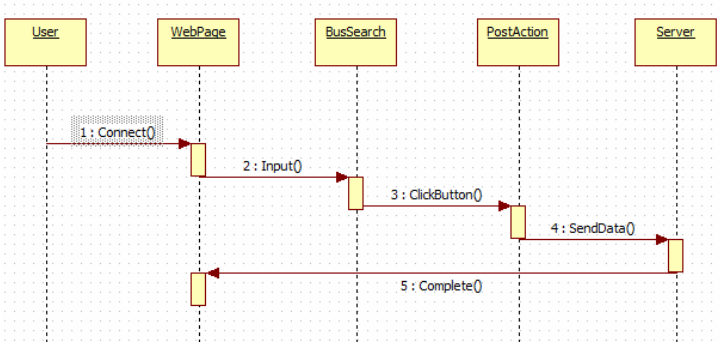
(표 5) 아이폰에서 SmartSMS 를 사용하여 SMS 를 보내는 흐름도

아이폰의 SmartSMS 를 이용하여 SMS 보낼 경우이다.(표 4.2) 웹페이지에 접속할 필요도 없고, 사용자가 SMSSlot 과 같은 기능이 있는 곳을 찾을 필요성도 없다. 웹페이지 데이터를 획득하여 핸드폰에서 문자를 보내는 것과 비슷하게 사용자 인터페이스를 만든다. 사용자는 자신이 보내려고 하는 메시지와 전화번호만 입력하면 상대방에게 SMS 를 보낼 수 있다.

② SeoulBus

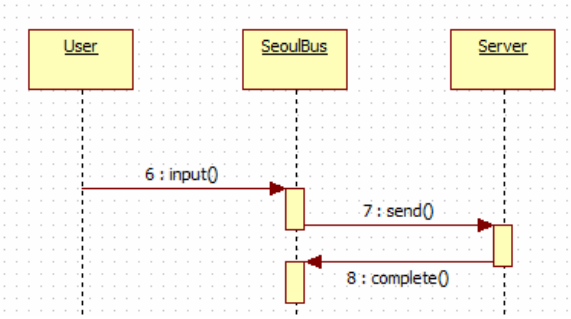
(그림 5) 서울버스

Seoul Bus는(그림 5) 자신이 원하는 정류장에서 다음 버스의 도착 시간이 얼마나 남았는지에 대해서 편리하게 보여주는 아이폰 어플리케이션이다. 해당 버스의 노선이나, 정류장의 위치를 선택하면 현재 버스가 어디에 있고, 버스의 도착시간을 알려준다.



(표 5) 웹페이지에서 버스 서비스를 이용할 경우

데스크탑 컴퓨터로 웹페이지의 서비스를 이용하여 버스의 실시간 정보를 획득할 때이다.(표 5) SMS와 상당히 비슷하다. 사용자가 웹페이지에 접속하고 웹페이지의 버스 검색 페이지를 찾는다. 검색페이지에 자신이 원하는 노선을 입력하고 버튼을 눌러 Post를 서버에 보낸다. 서버가 해당 노선의 데이터를 웹페이지에 전송하는 형태이다.



(표 6) 아이폰에서 버스를 검색할 경우

아이폰에서 SeoulBus어플리케이션을 이용하여 해당 노선을 검색할 때이다.(표 6) 매우 간결해 진것을 알수 있다. 사용자는 자신이 원하는 노선만을 사용자 유저인터페이스에 맞게 디자인된 어플리케이션에 입력만 하면 된다.

3. 결론

본논문에서는 Hpple Parsing과 WebView의 JavaScript를 이용한 웹페이지 데이터 처리에 관하여, 속도 및 필요 메모리양, 편리성에 대해서 비교 및 분석을 해보았다. 작고 제한적인 기기인, 스마트폰에서의 데이터 사용이 높아지는 만큼 데스크탑 컴퓨터에 맞춰진 웹의 정보를 효율적이고 작은 데이터의 양으로 처리하는 것이 중요해 진 것은 자명하다. 분석 결과 JavaScript를 이용한 데이터처리는 Post 및 데이터 흐름을 동적으로 처리할 수 있는 강력한 기능을 갖추었으나[2], WebView의 특성상 여러 복합적인 기능을 갖으므로 필요한 메모리의 양이 Hpple에 비해 비대하였으며, 처리 속도 또한 상대적으로 느렸다. 이에 반해 Hpple는 웹페이지 데이터 처리를 동적으로 처리 하지는 못하지만, XML데이터 처리 및 RSS데이터를 간결하게 처리할수 있고, 필요 메모리양이 상대적으로 적었다. 이러한 퍼포먼스뿐만 아니라 편리성에 대해서도 분석해 보았다. JavaScript는 웹페이지 태그에 대해서 객체적접근(Document.Title)이 가능하였고, 한줄의 소스코드이면 자신이 필요로 하는 데이터에 접근할 수 있었다. 비주얼적인 요소를 WebView 내에 가지고 있기 때문에, 디버깅을 하는데 있어서 편리함을 가졌다.

정리하자면, WebView의 JavaScript를 이용한 파싱은 SmartSMS와 같은 웹페이지 데이터를 직접적으로 활용해야하고 Post를 보내 서버에게서 피드백을 받아 데이터를 처리하는 경우에 유용할 것이다. Hpple를 이용한 Parsing은 빠르고, 필요한 메모리가 작은 만큼 대규모의 XML파일이나, Query를 전달하여 XML형식으로 데이터를 받는 정적인 형태이거나, 웹페이지의 정보가 확정되어있는 경우(날씨정보와 같이, 서비스 제공자가 단방향으로 데이터를 전달하는 형태)에 적합하다고 할수 있다.

4. 참고문헌

[1] How To Choose The Best XML Parser for Your iPhone Project
http://www.raywenderlich.com/553/how-to-chose-the-best-xml-parser-for-your-iphone-project
[2] iPhone OS Reference Library
http://developer.apple.com/iphone/library/navigation/index.html
[3] W3C, Document Object Model (DOM) Level 1 Specification
http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001
[4] JavaScript Tutorial
http://www.w3schools.com/js/default.asp
[5] Hpple, An XML/HTML parser for Objective-C
http://github.com/topfunky/hpple