부트스트랩을 이용한 반응형 웹 및 하이브리드 앱 개발

허능호*, 김형은**, 최병준*, 김영진***, 김용훈*, 서태원*
*고려대학교 컴퓨터학과
**단국대학교 시각디자인학과
***세종대학교 디지털콘텐츠학과
e-mail:hnh0716@korea.ac.kr, thegoku@naver.com, cbj5210@nate.com,
18vs1004@naver.com, newtype94@korea.ac.kr, suhtw@korea.ac.kr

A Study on Development of Responsive Web and Hybrid App using Bootstrap

Neung-Ho Heo*, Hyeong-Eun Kim**, Byung-Jun Choi*, Young-Jin Kim***,
Yong-Hoon Kim*, Tae-Weon Suh*
*Dept of Computer Science and Engineering, Korea University
**Dept of Visual Design, Dankook University
***Dept of Digital Contents, Sejong University

요 약

다양한 크기의 모바일 기기가 등장함에 따라 기존 웹 어플리케이션으로는 개발자가 의도한대로 레이아웃을 구성하기 어려우며 네이티브 어플리케이션은 운영체제별로 각각 개발해야하기 때문에 많은 개발 시간을 필요로 한다는 단점이 있다. 본 논문에서는 부트스트랩을 이용하여 기기의 화면 크기에 가변적인 반응형 웹을 구축하고 모바일 운영체제의 웹뷰를 이용한 하이브리드 앱을 구현함으로써 개발 과정 및 개발 시간 단축을 검증하였다.

1. Introduction

일반적으로 모바일 어플리케이션은 크게 네이티브 어플리케이션, 웹(Web) 어플리케이션, 하이브리드 어플리케이션과 같이 3가지로 나눌 수 있다. 첫째로 네이티브 어플리케이션은 iOS, 안드로이드와 같은 모바일 운영체제가 제공하는 개발 환경에서 ObjectiveC 또는 Java를 사용하여 만들어지는 어플리케이션을 말한다.[1] 또한 HTML과 웹 브라우저를 기반으로 원격의 웹 서버에서 주된 기능을 수행하도록 구성된 것을 웹 어플리케이션이라 하며 하이브리드 어플리케이션은 최신 모바일 기기들 대부분에 적용되어 있는 웹킷(Webkit)의 웹뷰(WebView)를 이용하여 웹 페이지를 구성한 어플리케이션을 말한다. 주요 모바일 어플리케이션의 유형별 구성요소는 (그림 1)에 종합적으로 나타나 있다.[2]

하지만 네이티브 어플리케이션의 경우 개발자가 선택한 모바일 운영체제에서만 동작한다는 단점이 있으며 웹 어플리케이션의 경우 특정 화면 크기에 최적화되어 제작되기 때문에 다양한 크기의 모바일 기기에서는 개발자가 기대한대로 레이아웃이 구성되지 않는 단점이 있다. 따라서 최근에 주목받고 있는 개발 방식은 반응형 웹 프레임워크를 이용하여 웹 어플리케이션을 구축하고 모바일 운영체제가 제공하는 웹뷰를 이용해 하이브리드 앱을 제작하는 것이다. 이를 통해 모바일 기기의 크기, 운영체제의 종류에 관계없이 동일한 레이아웃을 사용자에게 제공할 수 있으며 웹과 앱을 한 번에 개발할 수 있기 때문에 개발 시간을 단축시킬 수 있는 장점이 있다. 현재 가장 잘 알려진 반응형 웹 프론트엔드 프레임워크로는 부트스트랩(Bootstrap)[3]이 있다. 부트스트랩은 오픈 소스로 공개되어 있으며 그리드 레이아웃을 사용하여 기기의 종류 및 크기에 따라 디자인 요소를 자동으로 정렬한다.[4]

(그림 1) 모바일 어플리케이션의 유형별 구성도[2]

본 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 기본연구지원사업 지원을 받아 수행되었음 (NRF-2014R1A1A2059652)

2. 반응형 웹 및 하이브리드 앱 개발

2.1. 반응형 웹 어플리케이션 개발

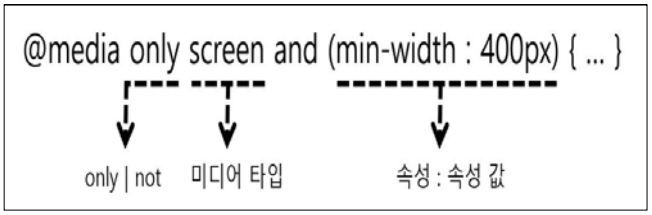
본 논문에서는 Apache 2.2 서버에서 PHP 5.5, MySQL 5.1을 사용해 반응형 웹을 구축하였다. 또한 웹 어플리케이션의 구축 과정에서 (그림 2)에 나타나 있는 CSS3(Cascading Style Sheets3) 미디어 쿼리, 유동형 그리드(Fluid Grid), 유동형 이미지(Flexible Images) 기술 요소를 사용하였다.

기술 요소	설명
캐스케이딩 스타일 시트3 미디어 쿼리	디스플레이의 폭, 높이 등의 정보들을 이용하여 자유자재로 스타일을 바꿀 수 있음.
유동형 그리드	기존의 시스템과 달리 유동적인 그리드를 사용.
유동형 이미지	디스플레이 비율에 기반한 유동형 이미지를 사용.

(그림 2) 반응형 웹의 주요 기술[5]

2.1.1. CSS3 미디어 쿼리

CSS3 미디어 쿼리는 현재 브라우저가 특정 쿼리를 만족하는지 여부에 따라 서로 다른 CSS 스타일을 적용시킬 수 있는 기능이다. CSS3 미디어 쿼리의 문법은 (그림 3)과 같으며 @media를 접두어로 only(미디어 쿼리를 지원하는 브라우저에서만 쿼리 만족 여부를 조사) 혹은 not(제외할 미디어 타입 지정)을 명시해야 한다. 데스크탑 혹은 모바일 기기의 경우 미디어 타입은 'screen'이며 이 후 쿼리를 구성하는 속성(조건)을 입력해야 한다.



(그림 3) 미디어 쿼리 문법[6]

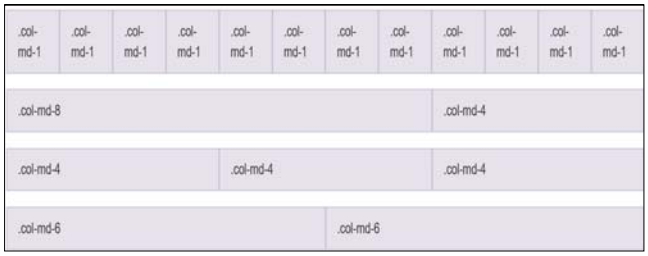
본 논문에서는 브라우저의 현재 가로 픽셀을 조건으로 쿼리를 만들었으며 사용한 코드는 (그림 4)와 같다.

2.1.2. 유동형 그리드

유동형 그리드는 브라우저의 가로 너비를 몇 개의 칼럼(column)으로 나누고 각 칼럼에 요소들을 배치하는 것으로 필요한 경우 칼럼을 묶어 다양한 크기로 요소를 나타낼 수 있다. 본 논문에서는 (그림 5)의 12칼럼 그리드를 사용하였으며 (그림 6)의 유동형 그리드 옵션을 이용하여 기기의 따라 여러 가지 형태로 그리드 시스템을 적용시킬 수 있게 하였다. (그림 7)은 유동형 그리드를 적용한 소스

```
<style>
    /* 991px 가 화면이 바뀌는 기준픽셀*/
    @media screen and (max-width:991px){
        .img-wrapper1 { margin:40px 0px; }
    }
</style>

<style>
    /* 400px 가 화면이 바뀌는 기준픽셀 - 로그텍스트*/
    @media screen and (max-width:400px){
        .reduce-text {font-size :0.8em;}
    }
</style>

<style>
    /* 386px 가 화면이 바뀌는 기준픽셀 - 로고이미지*/
    @media screen and (max-width:386px){
        .reduce-image {width : 100%}
    }
</style>

<style>
    /* 460px 가 화면이 바뀌는 기준픽셀 - 지역바로가기 이미지 */
    @media screen and (max-width:460px){
        .reduce-image2 {width : 60%;}
    }
</style>
```

(그림 4) 미디어 쿼리 소스코드



(그림 5) 12칼럼 그리드[7]

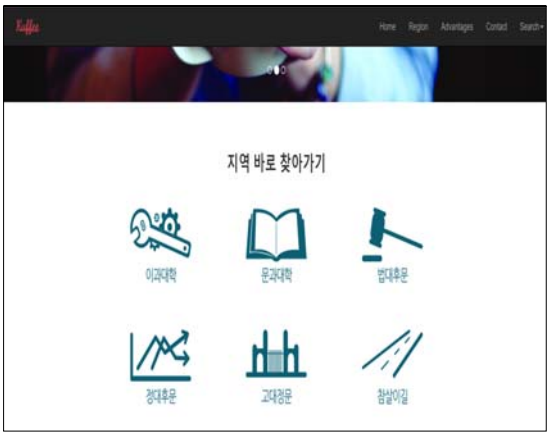
	매우 작은 기기 모바일폰 (<768px)	작은 기기 태블릿 (≥768px)	중간 기기 데스크탑 (≥992px)	큰 기기 데스크탑 (≥1200px)
콘테이너 너비	없음 (auto)	750px	970px	1170px
클래스 접두사	.col-xs-	.col-sm-	.col-md-	.col-lg-

(그림 6) 그리드 시스템 옵션[7]

코드의 예로 <div class = "col-md-4 col-xs-6 …>가 의미하는 것은 (그림 8)과 같이 데스크탑 환경에서는 하나의 요소를 4개의 칼럼(col-md-4)으로 표현하라는 것이고 (그림 9)와 같이 모바일 기기 환경에서는 하나의 요소를 6개의 칼럼(col-xs-6)으로 표현하라는 것이다.

```
<div class="col-md-4 col-xs-6 img-wrapper2 reduce-image reduce-padding"
style="padding-top: 15px; padding-bottom: 0px;">
    <a href="/src/cafe_list.php?region=1"><img class="reduce-image2"
    src="img/icon/1.png"></a>
</div>

<div class="col-md-4 col-xs-6 img-wrapper2 reduce-image reduce-padding"
style="padding-top: 15px; padding-bottom: 0px;">
    <a href="/src/cafe_list.php?region=2"><img class="reduce-image2"
    src="img/icon/2.png"></a>
</div>

<div class="col-md-4 col-xs-6 img-wrapper2 reduce-image reduce-padding"
style="padding-top: 15px; padding-bottom: 0px;">
    <a href="/src/cafe_list.php?region=3"><img class="reduce-image2"
    src="img/icon/3.png"></a>
</div>

<div class="col-md-4 col-xs-6 img-wrapper2 reduce-image reduce-padding"
style="padding-top: 15px; padding-bottom: 0px;">
    <a href="/src/cafe_list.php?region=4"><img class="reduce-image2"
    src="img/icon/4.png"></a>
</div>

<div class="col-md-4 col-xs-6 img-wrapper2 reduce-image reduce-padding"
style="padding-top: 15px; padding-bottom: 0px;">
    <a href="/src/cafe_list.php?region=5"><img class="reduce-image2"
    src="img/icon/5.png"></a>
</div>
```

(그림 7) 그리드 시스템 소스코드



(그림 8) 그리드 시스템 - 데스크탑

(그림 9) 그리드 시스템 - 모바일 기기

2.1.3. 유동형 이미지

유동형 이미지의 크기는 브라우저의 크기가 변경되는 것에 비례하여 변화하며 백분율로 가변 폭을 명시할 수 있다. 즉 (그림 10)의 소스코드에서 style="width:100%"은 요소의 가로 너비와 브라우저의 가로 너비가 동일하게 표현되는 것을 의미한다.

```
<div class="carousel-inner" role="listbox">
    <div class="item active">
        <img class="slide" src="/img/img1.jpg" alt="First slide" style="width:100%">
        <div class="container">
            <div class="carousel-caption"></div>
        </div>
    </div>

    <div class="item">
        <img class="slide" src="/img/img2.jpg" alt="Second slide" style="width:100%">
        <div class="container">
            <div class="carousel-caption"></div>
        </div>
    </div>
</div>
```

(그림 10) 유동형 이미지 소스코드

2.2. 하이브리드 앱 개발

본 논문에서는 웹뷰로 구성된 하이브리드 앱을 제작하여 사용자가 모바일 기기에서 좀 더 편리하게 반응형 웹 페이지에 접근할 수 있도록 하였다. 개발 도구는 안드로이드의 경우 Android Studio 2.1.2, iOS의 경우는 Xcode 7.3을 사용하였다. Android Studio는 구글이 IntelliJ IDEA를 기반으로 만든 통합 개발 도구로 2014년 10월부터 안드로이드의 공식 IDE가 되었으며[8] Xcode는 C, C++, Objective C, ObjectiveC++, Java, AppleScript, Python등을 지원하는 애플의 통합 개발 도구이다.

2.2.1. 안드로이드 웹뷰

안드로이드 웹뷰는 어플리케이션의 레이아웃(xml)을 구성하는 요소 중 하나로 모바일 브라우저와 달리 주소창을 포함하지 않기 때문에 특정 사이트에 특화된 앱을 제작할 때 자주 사용된다. 웹뷰를 사용하기 위해서는 xml에 선언된 웹뷰를 Java 클래스에서 객체로 생성해야하며 객체로 생성된 웹뷰는 어떤 옵션을 선택하느냐에 따라 다른 동작을 수행할 수 있다. 또한 Android Manifest에서 인터넷 액세스와 관련된 권한을 부여해야만 웹뷰를 정상적으로 사용할 수 있다. 웹뷰를 구성하는 소스코드는 (그림 11)과 같다.

```
//Android Manifest

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

// Main Activity

import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

protected void onCreate(Bundle savedInstanceState) {
webView = (WebView) findViewById(R.id.webView);
webView.setWebViewClient(new WebViewClient());
WebSettings set = webView.getSettings();
set.setJavaScriptEnabled(true);

webView.clearHistory();
webView.clearCache(true);
webView.loadUrl("http://kuffee.com");
webView.goBack();
webView.goForward();
webView.reload();
}
```

(그림 11) 안드로이드 웹뷰 소스코드

안드로이드의 웹뷰는 기본적으로 취소키(BackKey)를 눌렀을 경우 어플리케이션이 종료되도록 설정되어 있다. 하지만 본 논문의 하이브리드 앱에서는 사용자가 반응형 웹 페이지의 첫 번째 페이지를 이용 중인 경우에만 취소키를 눌렀을 때 프로그램이 종료되어야 하고 아닌 경우에는 이전 페이지로 이동해야 한다. 따라서 (그림 12)의 소스코드를 활용하여 첫 번째 웹 페이지에서 취소키를 빠르

게 2번 입력하는 경우에만 하이브리드 앱이 종료되도록 하였다.

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) && webView.canGoBack()) {
        webView.goBack();
        return true;
    }
return super.onKeyDown(keyCode, event);
}

public void onBackPressed() {
    if (System.currentTimeMillis() > backKeyPressedTime + 2000) {
        backKeyPressedTime = System.currentTimeMillis();
    }
    else{
        activity.finish();
    }
}
```

(그림 12) BackPressCloseHandler 함수[9]

2.2.2. iOS 웹뷰

iOS의 웹뷰는 안드로이드의 웹뷰와 같은 기능을 수행하며 비슷한 구조를 가지를 가지고 있다. swift를 이용하여 개발한 iOS 웹뷰 소스코드는 (그림 13)과 같다.

```
import UIKit
import WebKit

class ViewController: UIViewController {
    @IBOutlet weak var container: UIView!
    var webView: WKWebView!

    override func viewDidLoad() {
    super.viewDidLoad()
    webView = WKWebView()
    container.addSubview(webView)
    }

    override func viewDidAppear(animated: Bool) {
        let frame = CGRectMake(0, 0,
        container.bounds.width, container.bounds.height)
        webView.frame = frame
        loadRequest("http://kuffee.com")
    }

    func loadRequest(urlStr: String) {
        let url = NSURL(string: urlStr)!
        let request = NSURLRequest(URL: url)
        webView.loadRequest(request)
    }
}
```

(그림 13) iOS 웹뷰 소스코드

3. 결론

다양한 모바일 기기의 화면에 알맞은 웹 어플리케이션을 제작하려면 다수의 레이아웃을 구축해야 하므로 개발에 많은 시간이 소요된다. 하지만 레이아웃을 유동적으로 변하게 하는 반응형 웹을 이용하면 웹 어플리케이션 제작 시간을 현저히 줄일 수 있다. 또한 각각의 모바일 운영체제별로 네이티브 앱을 제작하는 것이 아니라 웹뷰를 활용한 하이브리드 앱 제작을 통해 반응형 웹을 모바일 기기에서 손쉽게 표현할 수 있다. 즉 기존에 웹 어플리케이션, 안드로이드 어플리케이션, iOS 어플리케이션을 각각 디자인하고 제작하던 것을 반응형 웹과 하이브리드 앱을 제작하는 것으로 대체 할 수 있다.

본 논문에서는 부트스트랩을 이용하여 반응형 웹을 구축하고, 웹뷰를 이용해 하이브리드 앱을 제작한 결과 개발 시간이 상당히 단축됨을 확인할 수 있었다. 본 논문의 모든 결과물은 http://kuffee.com 혹은 http://github.com/Shaffron/kuffee에서 확인할 수 있다.

참고문헌

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

[2] 정우진, 오장훈, and 윤동원. "하이브리드 모바일 앱 프레임워크 설계 및 구현." 한국정보통신학회논문지 16.9 (2012): 1990-1996.

[3] http://bootstrapk.com/

[4] http://www.doopedia.co.kr/doopedia/master/master.do?_method=view&MAS_IDX=160404001525749

[5] http://www.doopedia.co.kr/doopedia/master/master.do?_method=view&MAS_IDX=160329001525470

[6] http://www.nextree.co.kr/p8622/

[7] http://bootstrapk.com/css/

[8] https://en.wikipedia.org/wiki/Android_Studio

[9] http://dsnight.tistory.com/14