WIPI 기반 동적 API 추가 및 갱신 모듈 구현

Implementation of Dynamic API Addition and Update Module based on WIPI

박재선, 김광현
광주대학교 정보통신학과

Park Jae-Sun, Kim Gwang-Hyun
Dept. of Information and Communications, Gwangju University

요약

이동통신사의 다양한 플랫폼은 콘텐츠 개발회사의 다중개발에 의한 경제적 손실을 발생시키고 경쟁력 약화로 이어져 다양한 콘텐츠 개발에 집중하지 못하고 있는 문제점을 안고 있다. 이를 해결하기 위한 방안으로 기존의 무선인터넷 플랫폼 환경을 WIPI(Wireless Internet Platform for Interoperability)라는 모바일 표준 플랫폼으로 표준화하게 되었다. WIPI 표준 플랫폼은 다운로드에 의한 DLL(Dynamic Linking Library)을 지원함에 따라 무선망을 통해 동적으로 API (Application Programming Interface)를 추가/갱신하는 기능을 제공한다. 본 논문에서는 메모리 사용을 최소화 및 최적화하여 성능을 개선한 콘텐츠 뷰어를 개발하고, 이를 통해 동적 API 추가 및 갱신 모듈을 구현함으로써 효율적인 WIPI 콘텐츠 제작 환경을 제공하고자 한다.

I. 서 론

WIPI(Wireless Internet Platform for Interoperability)는 상호 연동성을 위한 무선인터넷 플랫폼이며, 무선 인터넷을 통해 다운로드 된 응용 프로그램을 이동통신 단말기에 탑재시켜 실행시키기 위한 환경을 제공하는데 필요한 모바일 표준 플랫폼 규격이다[1]. 기존 이동통신 사업자들의 각기 다른 무선인터넷 플랫폼을 사용으로 인해 나타날 수 있는 응용 프로그램 실행 환경의 혼재로 발생될 수 있는 콘텐츠 업체들의 개발 부담과 상호호환이 불가능한 상황에서 발생하는 중복투자의 문제, 그리고 소비자들은 이동통신사별 콘텐츠가 한정되어 이용 가능한 서비스의 종류가 제한되는 불합리한 문제들을 해결하기 위해 WIPI 개발이 진행되었다. WIPI는 이동통신사, 모바일 플랫폼 개발 업체, 단말기 제조사, 콘텐츠 개발 업체들로부터 요구사항을 받아 들여 2001년 5월 정보통신부, 이동통신 3사, 한국전자통신연구원(ETRI) 등이 주관이 돼 한국무선인터넷표준화포럼(KWIS)이 출범했으며, 2002년 3월 WIPI 1.0 버전이 확정됐고, 이어 2004년 2월 WIPI 2.0버전이 KWIS의 공식 표준규격으로 채택되었다[2].

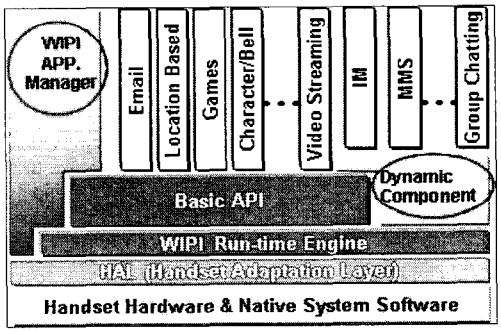
모바일 표준 플랫폼은 기존의 이동통신사에서 서비스되고 있는 GVM, Brew, SKVM과 같은 플랫폼에서는 지원하지 못했던 C/C++, Java 언어를 위한 규격을 모두 지원함으로서 다양한 콘텐츠 개발자들을 수용할 수 있도록 하였고, 바이너리 형태로 플랫폼에 로딩된 어플리케이션으로 수행 속도의 향상이 가능하며, 다중 어플리케이션이 동시에 수행되는 환경을 제공한다. 또한 DLL 규격을 통한 동적 API(Application Programing Interface) 추가 및 관리 기능이 지원됨에 따라 차별화된 API(DLL)의 확장이 가능하고 멀티미디어 코덱이나 보안 모듈 등을 DLL

로 개발하거나, 사용자의 취향에 따른 모든 UI(User Interface) 컴포넌트를 동적으로 변경하는 것들이 가능하다. 플랫폼 자체의 보안 기능을 추가함으로써 일반 어플리케이션으로부터 단말기 시스템과 디렉토리 접근 등을 막을 수 있는 등의 기술적인 특징들을 가지고 있다[1].

본 논문에서는 모바일 플랫폼에 제공하는 기술적인 특징 중 동적 API 추가 및 관리 기능을 이용하여 Java 언어와 KTF에서 제공하는 WIPI 에뮬레이터 WIPI1.2SDK 환경에서 사용자의 취향에 따른 UI 컴포넌트를 동적으로 변경하는 API를 구현하였다.

Ⅱ. 모바일 표준 플랫폼의 개요 및 특징

1. 모바일 표준 플랫폼 구조



▶▶ 그림 1. 모바일 표준 플랫폼 구조

모바일 플랫폼의 구조는 [그림 1]과 같다.

최하단 계층에 있는 Handset Hardware는 단말기의 하드웨어를 말하는 것이며, Native System Software는 단말기의 기본 소프트웨어를 말하는 것으로 단말기를 구동하고 있는 OS와 통신 기능 및 각종 디바이스 드라이버가 포함된다. HAL(Handset Adaptation Layer)은 단말기 제조사를 위한 API를 정의한 것으로 하단의 Native System Software와 플랫폼을 연결해 주는 역할을 한다. HAL이 단말기에 포팅 되면 바로 모바일 표준 플랫폼 실행 엔진을 탑재할 수 있다. WIPI Run Time Engine은 WIPI를 구동시키는 계층으로 Basic API와 HAL API를 연결시켜 주는 기능을 담당하기도 한다. WAM(WIPI Application Manager)은 Clet과 Jlet을 관리하는 프로그램으로 응용프로그램의 정보를 살펴보고 다운로드하여 설치하며, 해당 응용프로그램을 실행하고 삭제한다. 또한 보안 기능을 수행하여 사용자에게 각종 편리한 기능을 제공하며 DLL 관리 기능을 지원한다. Dynamic Component는 WAM에 의해 추가되거나 갱신되는 API들로 컴포넌트를 의미한다[2][3].

2. 동적 API 추가 및 관리 기능

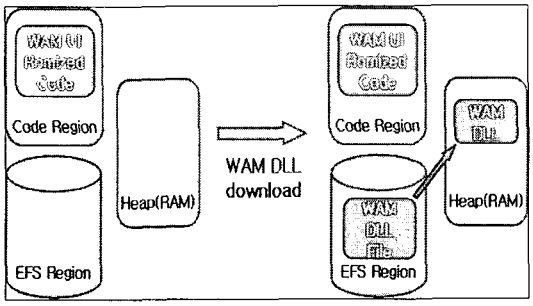
WIPI 2.0 플랫폼의 주요 특징을 보면 CLDC1.1/MIDP 2.0 지원이 선택에서 필수 규격으로 변경되고, 보안이 향상되었으며, 향상된 기능의 단말기 지원을 위한 새로운 API가 추가되었다. 또한 SMS API가 선택 규격에서 필수 규격으로 변경되었으며, VGI(Vector Graphic Interface)가 선택 규격으로 추가되고, API 추가/갱신 기능(DLL)이 필수 규격으로 변경되었다[2][3][4].

WIPI 2.0 특징 중 동적 API 추가 및 관리 기능은 "Ba -sic API Set, Extended API Set, Application manager 및 application 구동 환경에 대해서 기술적으로 가능한 한 무선망을 통해 API를 추가하거나 API 및 각종 manager, 구동 환경을 업그레이드할 수 있어야 한다."는 이동통신사의 요구사항에 따라 동적 API를 추가하거나 기존에 탑재된 API를 교체하는 DLL이 필수 기능으로 채택되었다. DLL은 인터페이스라는 외부와 통신하는 통로를 가지는데 인터페이스란 "함수와 변수로 이루어진 그룹에 이름, 버전을 부여하여 관리하는 단위"를 말하는 것으로 이 인터페이스는 API를 추가/갱신하는데 있어서 기본 단위가 된다[2].

모바일 표준 플랫폼은 API를 무선망을 통해서 추가/갱신할 수 있는데 이 때 추가/갱신된 API는 지속적으로 유지되어야 하며, 이를 지원하기 위하여 플랫폼은 API 추가/갱신에 따른 버전 관리 및 설치/삭제

기능을 가져야 하고 API 보안수준 정책을 동일하게 적용해야만 한다[5].

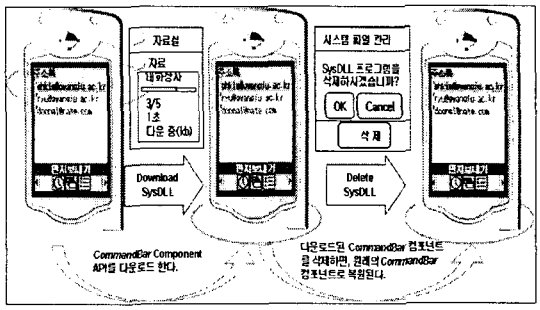
[그림 2]는 API 동적 갱신의 내부 처리 구조를 보여주고 있다. 새로운 DLL이 다운로드 되어 EFS (Encrypting File System) 영역에 저장되고, 플랫폼 구동 시 EFS 영역에 있는 시스템 DLL 파일들을 RAM에 로딩하면 플랫폼이 WAM을 수행하기 위해 동적 링킹 시에 RAM 영역에 갱신된 API가 있다면 먼저 참조하여 수행하게 된다.



▶▶ 그림 2. API 동적 갱신의 내부 처리 구조

동적 API 추가 및 관리 기능은 동적으로 API의 추가/갱신을 지원함으로써 차별화된 API의 확장이 가능하고 멀티미디어 코덱이나 보안 모듈 등을 동적 라이브러리 형태로 개발 가능하며 사용자의 취향에 따라 모든 UI 컴포넌트를 동적으로 변경할 수 있고 플랫폼이 출시된 이후 버그 패치 혹은 기능 추가가 가능하다는 장점을 가지고 있다[1][5]. 추가 또는 갱신할 API에 해당되는 DLL을 설치하고, 삭제하는 기능은 WAM에서 담당하는데, API 호출이 발생할 경우 WAM_Engine API는 ROM의 API를 그대로 사용하고, WAM_UI API는 사용자가 다운로드 받은 WAM_UI API를 호출하게 되며, 사용자가 다운로드 받은 Extended API는 플랫폼에는 없었지만 새롭게 추가된 것처럼 동작한다.

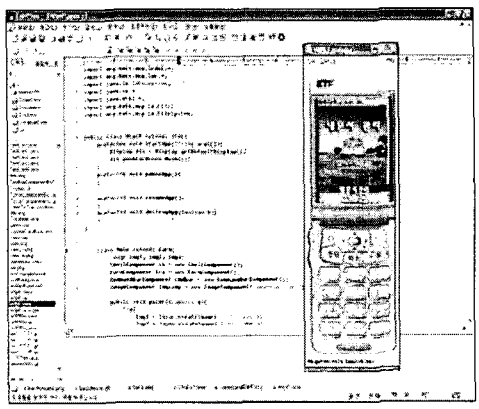
본 논문에서 구현한 [그림 3]은 플랫폼이 동작중인 상태에서 API가 다운로드 되어 설치되고, 플랫폼의 API가 동적으로 변경된다. 새로운 GUI 컴포넌트 (Command-Bar) API를 다운로드하여 동적으로 컴포넌트를 갱신하고 이후 삭제하는 모습을 보여주고 있다.



▶▶ 그림 3. DLL을 통한 API 동적 추가/갱신

Ⅲ. 실험 환경 구축 및 구현

동적 API 추가 및 갱신 모듈을 구현하기 위해서는 먼저 WIPI 에뮬레이터 중 KTF에서 제공하는 WIPI 에뮬레이터 WIPI1.2SDK를 사용하였고, 프로그램 소스편집을 위한 텍스트 편집기는 EditPlus를 사용하였다.



▶▶ 그림 4. 동적 API 추가 및 갱신 모듈 실험 환경

메인 화면, 뷰어 다운로드, 내문서 관리, 이미지 관리의 4가지 메뉴를 구성하였고, 컴포넌트로는

ShellComponent, FormComponent, Command-Bar-Component, ImageCom po -nent, DialogComponent, ProgressComponent, LabelCom -ponent, TextFieldComponent, TextBoxComponent, ButtonComponent를 이용하였다.

▶▶ 그림 5. 동적 API 추가 및 갱신 모듈 구현 화면

메인 화면에는 메인 이미지를 출력하고, 뷰어 다운로드 메뉴는 다운로드 받을 프로그램을 나타내고, 다운로드 선택 시 다운로드 진행 상태 출력 후 뷰어 다운로드 메인으로 이동하도록 하였다. 내문서 관리 메뉴는 새 파일을 작성하고 작성한 파일에 대해 수정과 삭제가 가능하며, 완료시 메인으로 이동하도록 하였다. 이미지 관리 메뉴는 이미지 리스트를 출력하고, 선택 시 이미지를 출력 후 메인 화면으로 이동하도록 하였다.

[그림 6]은 본 논문에서 구현한 내문서 관리 메뉴의 파일을 작성하고, 폴더인지 파일인지를 구분하여 이미지를 보여주는 소스를 보여주고 있다.

```
public void noteForm(String btnOption){
  if (btnOption.equals("")){
    form.removeAllComponents();
    formCmd.removeAllComponents();

    form = new FormComponent();
    formCmd = new FormComponent(false);
    list = new ListComponent(ListComponent.SELECT_IMPLICIT);
    try{
      imgFolder = Image.createImage("/folder.bmp");
      imgFile = Image.createImage("/doc.bmp");
    }catch(IOException e){
      System.out.println("이미지로딩실패");
    }
    try{
      Vector folder = FileSystem.list("/"+selDir,fMode);
      if (folder.size() > 0){
        for(int i = 0 ; i < folder.size() ; i++ ){
          String filename = (String) folder.elementAt(i);
          if(FileSystem.isDirectory(selDir+filename,fMode))
            list.addComponent(new ListItemComponent(filename,"folder.bmp"));
        }
      for(int i = 0 ; i < folder.size() ; i++ ){
        String filename = (String) folder.elementAt(i);
        if(FileSystem.isFile(selDir+filename,fMode))
        list.addComponent(new ListItemComponent(filename,"doc.bmp"));
      }
        list.setActionListener(this, list);
        list.setChangeListener(this, null);
        form.addComponent(list);
      }else{
        form.addComponent(new LabelComponent("폴더 없음"));
      }
    }catch(Exception e){
      System.out.println("이미지로딩실패");
    }
  }
  if (btnOption.equals("")){
    btnMain = new ButtonComponent("메인", null);
    btnCre = new ButtonComponent("새파일 작성", null);
    btnMain.setActionListener(this, "메인");
    btnCre.setActionListener(this,"Cmove");
    formCmd.addComponent(btnMain);
    formCmd.setGab(5);
    formCmd.addComponent(btnCre);
    formCmd.setGab(5);
    if(selDir.length() > 0 ){
      btnUp = new ButtonComponent("상위", null);
      btnUp.setActionListener(this,"/");
      formCmd.addComponent(btnUp);
```

▶▶ 그림 6. 동적 API 추가 및 갱신 모듈 소스

Ⅳ. 결론 및 향후 연구 과제

WIPI 2.0 플랫폼은 동적 API를 추가하거나 기존에 탑재된 API를 교체하는 DLL 기능이 추가되었다. 이로 인해, 멀티미디어 코덱이나 보안 모듈 등을 동적 라이브러리로 개발하여 추가 삭제하는 과정이 용이해졌다.

본 논문에서는 효율적인 WIPI 콘텐츠 제작 환경을 제공하기 위한 동적 API 기능의 추가 및 갱신 모듈을 구현하였다. KTF에서 제공되는 WIPI 에뮬레이터를 이용하여 실험을 진행하였고, JAVA를 이용하여 GUI 환경에서의 동적 API의 다운로드, 설치, 삭제 등의 모듈을 추가하였으며, 메인화면, 뷰어 다운로드, 내문서 관리, 이미지 관리 등의 총 4가지 메뉴를 구성하여 실험하였다.

향후과제로는 단순한 이미지 환경에서 벗어나 3D 효과를 갖는 모듈 구현이 필요하다. 또한, 보다 효율적이며 편리한 WIPI 콘텐츠 개발을 통해 사용자에게 다양하고 유용한 서비스 제공이 요구된다.

▌참 고 문 헌▐

[1] 권영주 "WIPI 2.0, 공식 표준규격 채택의 의미", 정보통신정책, 제16권, 제5호, pp.55-59, 2004.
[2] 이상윤, 이환군, 김우식, 이재호, 김선자, 김홍남 "무선 인터넷 표준 플랫폼 WIPI 2.0의 표준화 동향", 전자통신동향분석, 제19권, 5호(통권 89호), pp.143-149, 2004.
[3] 강상원, 임석진, 심양섭, 모바일 플랫폼 천하통일 위피 프로그래밍, pp.79-81, 제우미디어, 서울, 2004.
[4] 김석구, 김한규, 안종현, 위피 모바일 프로그래밍, 영진닷컴, 2005.
[5] 한국 무선 인터넷 표준화 포럼, 모바일 표준 플랫폼 규격 2.0.1, http://www.wipi.or.kr
[6] KTF Mobile Application Center, http://developer.magicn.com
[7] MobileJAVA Developer Community, http://www.mobilejava.co.kr