논문 2011-06-29

안드로이드 애플리케이션 코드 재사용성을 높이기 위한 인텐트 메커니즘의 확장

(Extended Intent Mechanism for Increasing Code Reusability of Android Applications)

안 수 정, 김 병 호, 김 진 천*
(Sujeong An, Byungho Kim, Jinchun Kim)

Abstract : A novel ecosystem surrounding developing, publishing and using of smartphone applications is driving a new paradigm in software industry. Thousands of applications are newly published on appstores everyday. However more than 97% of them happen to be downloaded less than 1,000 times and resultingly disappeared out of user's interesting. It means that so many efforts and time of developers are vanished. In this paper, we proposed a new architecture to increase code reusability of Android applications so that the time and efforts to develop new applications can be shortened. The proposed architecture, an extended Intent mechanism, supports sharing of Android components among the applications registered in different servers as well as in the same Android device. We designed a new Intent mechanism by extending the PackageManager service and by adopting a new class for ServerPackageManager service.

Keywords : Android, Code reusability, Mobile applications, Intent, Java

1. 서 론

앱스토어를 통해 스마트폰 애플리케이션을 개발하고 배포하고 사용하는 새로운 에코시스템이 소프트웨어 산업의 패러다임을 바꾸고 있다. 소프트웨어 산업의 중심이 과거 ERP나 패키지 소프트웨어와 같은 대규모 소프트웨어 위주에서 소규모 스마트폰 애플리케이션 중심으로 이동되고 있다. 2011년 4월 현재 애플 앱스토어에 등록된 애플리케이션의 개수는 45만 여개에 달하며 월 평균 1,300여개의 애플리케이션이 신규로 등록되고 있다 [1]. 반면에 애드몹 분석에 따르면 2009년 5월 기준으로 애플 앱스토어에서 1,000번 이상 다운로드된 애플리케이션의 개수는 1,063개로써 등록된 전체 애플리케이션의 2.8%에 불과하다 [2-3]. 애플 앱스토어 이외에도 안드로이드 등 다른 스마트폰 운영체제의 앱스토어들까지 포함하면 매월 출시되는 수 천 개의 애플리케이션들의 95% 이상이 제대로 사용되지도 못하고 사장된다는 것으로 볼 수 있으며 이는 곧 수많은 개발자들의 노력과 시간이 낭비되고 있다는 것을 의미한다.

본 논문에서는 소프트웨어의 재사용성을 높여 스마트폰 애플리케이션 개발자의 노력과 시간의 낭비를 줄일 수 있는 방안에 대해 연구하고 그 해결책을 제시한다. 모바일 플랫폼에서 소프트웨어 재사용성을 높이기 위한 연구 분야를 크게 나누어 보면 1) 모바일 운영체제, 2) 소프트웨어 공학, 3) 프로그래밍 언어 분야가 있다. 소프트웨어 공학 측면에서는 소프트웨어 재사용성을 근본적으로 향상시킬 수 있는 새로운 패러다임에 대한 연구가 가능하며, 프로그래밍 언어 측면에서는 실행 중 코드 공유를 위한 동적 바인딩 문제에 관한 연구가 필요하고, 모바일 운영체제 측면에서는 이러한 동적 바인딩 및 코드 공유를 지원하는 아키텍처에 대한 연구가 가능하다.

본 논문에서는 운영체제 측면에서 동적 코드 공유를 통해 코드 재사용성을 높이는 방안을 연구하

* 김진천(Corresponding Author)
논문접수 : 2011. 05. 31., 수정일: 2011. 07. 11.,
채택확정 : 2011. 08. 12.
안수정, 김병호, 김진천 : 경성대학교 컴퓨터학부

고, 안드로이드 운영체제의 인텐트(Intent) 메커니즘을 확장하였다. 제안하는 인텐트 메커니즘의 확장을 통해 동일 기기 내의 애플리케이션 간은 물론 다른 서버에 있는 애플리케이션과도 동적 코드 공유를 가능하게 함으로써 코드 재사용성을 높일 수 있다.

II. 안드로이드 인텐트

1. 안드로이드 프레임워크

안드로이드 운영체제에서 하나의 애플리케이션은 그림 1과 같이 다수의 컴포넌트들로 이루어진 APK 패키지로 구성된다. 컴포넌트에는 Activity, Content Provider, Service, Broadcast Receiver의 4가지 종류가 있으며, 특히 Activity는 단위 동작을 수행하는 프로그램 모듈로써 일반적으로 하나의 안드로이드 애플리케이션은 다수의 Activity들로 구성된다 [4].

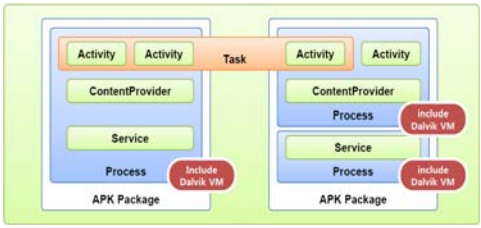

그림 1. 안드로이드 태스크 모델
Fig. 1. Task model in android

Activity는 응용프로그램에서 하나의 화면을 지칭하며 사용자에게 View와 이벤트 응답으로 이루어진 인터페이스를 제공한다. 인텐트 리시버는 일종의 이벤트 핸들러로써 외부에서 전달받은 인텐트를 처리하는 역할을 한다 [5][6].

안드로이드 프레임워크의 컴포넌트 시작 과정은 그림 2와 같다.

달빅 가상 머신을 구동한 후 시스템 서버에 의해 필요한 네이티브 서비스의 실행과 안드로이드 프레임워크의 서비스들을 시작한다. 이때 시작되는 서비스로는 안드로이드 애플리케이션의 액티비티를 관리하는 액티비티 매니저와 애플리케이션을 설치하거나 관리하는 패키지 매니저 등이 있다 [7][8].

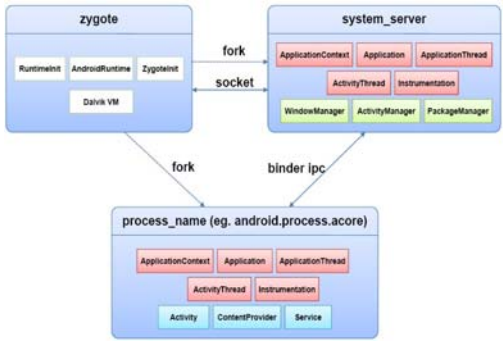

그림 2. 컴포넌트의 시작 과정
Fig. 2. Component start-up

2. 안드로이드의 동적 코드 공유

리눅스 기반 모바일 운영체제인 구글 안드로이드는 서로 다른 애플리케이션 간의 동적 코드 공유를 지원한다. 태스크는 안드로이드 애플리케이션의 실제 수행 단위로써 주목할 점은 태스크 모델을 통해 한 애플리케이션이 타 애플리케이션의 Activity를 가져다 사용할 수 있다는 것이다. 즉, 애플리케이션 개발자는 일부 컴포넌트를 직접 개발하지 않고 다른 애플리케이션에 포함되어 있는 컴포넌트를 대신 사용하도록 지정할 수 있으며, 안드로이드 운영체제는 애플리케이션 실행 시에 해당 컴포넌트를 타 애플리케이션으로부터 가져다 쓸 수 있도록 지원한다.

AndroidManifest.xml 파일은 안드로이드 애플리케이션에 포함되어 그 애플리케이션의 전체 구성을 기술하는데 사용된다. 그림 3의 애플리케이션은 하나의 Activity 컴포넌트로 구성되어 있고 인텐트 필터를 통해 해당 Activity를 공유시키는 예이다 [4].

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.kandroid.helloandroid"
      android:versionCode="1"
      android:versionName="1.0.0">

  <application android:icon="@drawable/icon"
           android:label="@string/app_name">

    <activity android:name=".HelloAndroid"
          android:label="@string/app_name">

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>
  </application>
</manifest>
```

그림 3. 인텐트 필터 예
Fig 3. Example of intent filter

그림 4는 안드로이드 애플리케이션 코드의 예로써 필요한 Activity 컴포넌트를 직접 구현하지 않고 인텐트를 통해 타 애플리케이션으로부터 가져다 쓰도록 지정하는 것을 보여준다. 즉, setAction() 메소드를 사용하여 필요한 Activity의 속성을 지정하고, 이 속성은 타 애플리케이션에 있는 적합한 Activity를 찾는데 사용된다.

그림 4. 인텐트 사용 예
Fig. 4. Example of using intent

3. 인텐트

안드로이드 운영체제에서는 애플리케이션 간의 메시지 전달 메커니즘으로 인텐트를 사용한다. 안드로이드의 4개 컴포넌트들 중 Activity, Broadcast Receiver, Service는 인텐트를 사용하여 활성화되며 이 때 메시지 전달에 사용되는 객체가 인텐트 객체이다. 인텐트 객체는 컴포넌트의 이름, 수행할 액션, 액션을 수행할 컴포넌트의 카테고리, 해당 컴포넌트에서 사용할 데이터들을 포함하고 있는 정보의 묶음이다.

인텐트는 그림 5와 같이 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)로 나뉜다. 명시적 인텐트는 컴포넌트 호출 시 사용할 액티비티나 서비스의 이름을 명시하여 해당 컴포넌트를 실행한다. 반면에 암시적 인텐트는 실행하고자 하는 액티비티나 서비스의 이름을 포함하지 않고 특정 액션 값만을 가지고 컴포넌트를 실행한다. 암시적 인텐트에 포함된 액션을 전달받은 안드로이드 프레임워크는 해당 액션을 처리하는 가장 적합한 컴포넌트를 검색한 다음 이를 실행한다. 즉, 특정 컴포넌트 이름을 명시하지 않고 필요한 컴포넌트의 명세 조건만을 전달하는 방식으로 이 명세 조건은 AndroidManifest.xml에 기술된 인텐트 필터 정보와의 비교를 통해 이 메시지를 받아 처리할 수 있는 컴포넌트를 찾게 되는데 이 과정을 인텐트 해석(Routing Intent)이라고 한다.

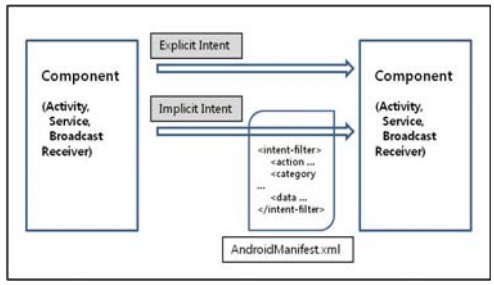

그림 5. 명시적 인텐트/암시적 인텐트
Fig. 5. Explicit intent/implicit intent

인텐트는 같은 애플리케이션 내의 컴포넌트뿐만 아니라 동일 기기에 등록된 다른 애플리케이션에도 전달된다. 따라서 프로그램 개발자는 인텐트를 통해 자신이 만든 컴포넌트뿐만 아니라 다른 사람이 만든 컴포넌트들도 사용할 수 있어 코드 재사용이 가능하다.

4. 인텐트 필터

매니페스트는 안드로이드 운영체제가 해당 애플리케이션 코드를 실행할 때 알아야 하는 정보를 사전에 안드로이드 시스템에 제공할 목적으로 사용된다. 즉, 안드로이드 운영체제가 외부로부터의 컴포넌트 호출 요청을 처리하기 위해서는 해당 컴포넌트들이 처리할 수 있는 기능을 미리 알고 있어야 하기 때문에 애플리케이션은 AndroidManifest.xml의 인텐트 필터를 통해 공유할 컴포넌트의 사양을 명시한다.

컴포넌트 사양은 인텐트의 action, category, data 필터를 통해 기술된다. 하나의 컴포넌트가 여러 개의 인텐트 필터를 가질 수 있는데 각각의 인텐트 필터는 암시적 인텐트를 처리하기 위한 것이다. 암시적 인텐트는 여러 필터 중 하나라도 일치하면 해당 컴포넌트에게 메시지가 전달되어 컴포넌트가 실행된다.

5. 인텐트 해석

인텐트 해석은 어떤 컴포넌트를 실행할 것인가를 선택하는 과정이다. 호출하는 컴포넌트의 인텐트 메시지에 기술된 action, category, data 필터 정보를 분석하여 적합한 수신 컴포넌트를 찾는데 수신 컴포넌트에 관한 정보는 AndroidManifest.xml를 통해 등록된 인텐트 필터를 참조한다. 인텐트 해석은 호출하는 인텐트 객체의 내용과 등록되어 있는 컴포넌트의 인텐트 필터 내용을 비교하여 해당 인텐

트를 처리할 가장 적절한 컴포넌트를 찾아내는 과정이다. Activity 컴포넌트의 경우 암시적 인텐트 해석으로 선택된 Activity 컴포넌트가 2개 이상일 경우 createChooser() 메소드를 통해 사용자가 선택하게 할 수 있다 [9].

인텐트 필터에 사용되는 필터는 다음과 같다.

Action : 컴포넌트의 기능을 기술하는 필터이다. 호출하는 컴포넌트 인텐트 객체에 기술된 액션은 수신 컴포넌트 인텐트 필터에 나열된 액션들에 의해 걸러진다. 수신 컴포넌트의 인텐트 필터와 매칭되는 액션이 없거나 인텐트 필터 내에 아무런 액션이 명시되어 있지 않으면 해당 인텐트 요청은 차단된다. 반면에 호출하는 인텐트에 액션이 지정되어 있지 않고 수신 인텐트 필터에 적어도 하나의 액션이 포함되어 있다면 해당 인텐트 요청은 통과된다.

인텐트 클래스에는 20개의 기본 액션들이 정의되어 있으며 개발자는 기본 액션 이외에 자신만의 액션 문자열을 정의하여 사용할 수 있다.

Category : 컴포넌트의 종류에 대한 추가적인 정보를 포함하고 있다. 인텐트가 카테고리 필터를 통과하기 위해서는 호출 인텐트 객체와 수신 인텐트 필터의 카테고리가 서로 일치하여야 한다. 호출 인텐트 객체의 카테고리가 지정되어 있지 않더라도 수신 인텐트 필터 카테고리가 DEFAULT이면 그 인텐트는 통과된다. 기본적인 카테고리들은 인텐트 클래스에 미리 정의되어 있으며 개발자가 자신만의 카테고리를 정의하여 사용할 수 있다.

Data : 수신 컴포넌트가 처리할 데이터를 기술한다. 수신 인텐트 필터는 호출 인텐트 객체의 URI와 MIME 형식을 비교하여 일치하는 것을 통과시킨다. URI 요소로는 schema, host, port, path가 있다. MIME Type은 비교할 데이터의 종류를 나타내며 특정 요구가 없으면 비워둘 수 있다.

Extra : 부가적인 정보를 전달하는데 사용된다. 예를 들어 컴포넌트를 호출하거나 메시지를 보낼 때 데이터를 URI 형식이 아닌 다른 방식으로 전달할 때 사용된다.

Ⅲ. 인텐트 메커니즘의 확장

1. 기존 인텐트 메커니즘

인텐트를 통한 컴포넌트 활성화는 안드로이드 애플리케이션 프레임워크 계층의 액티비티 매니저와 패키지 매니저가 담당한다. 이 두 서비스는 시스템 서비스로써 시스템에 상주한다.

액티비티 매니저는 컴포넌트가 인텐트를 호출할 때 필요한 startActivity(), startService(), sendBroadcast() 메소드를 제공하고 있으며 동작중인 컴포넌트로부터 인텐트 요청을 받으면 패키지 매니저에 해당 인텐트를 처리할 수 있는 컴포넌트가 무엇인지 물어본다. 패키지 매니저는 애플리케이션들이 설치될 때 각 애플리케이션의 APK 파일에 포함되어 있는 매니페스트 파일로부터 패키지와 관련된 권한 정보와 인텐트 필터 정보를 파싱하고 관리하기 때문에 인텐트 해석 과정에서 실행 가능한 컴포넌트를 찾아 실행 시킬 수 있다. 이때 실행 가능한 컴포넌트가 여러 개 있을 경우 사용자에게 실행시킬 최종 컴포넌트를 선택하게 할 수 있다.

2. 확장 인텐트 메커니즘

기존 안드로이드 인텐트 메커니즘은 메시지를 주고받는 애플리케이션들이 동일한 안드로이드 시스템에 설치되어 있음을 전제로 한다. 공유될 컴포넌트들을 관리하는 패키지 매니저가 애플리케이션의 매니페스트 파일의 인텐트 필터로부터 공유 가능한 컴포넌트들의 목록을 얻을 수 있는 방안은 애플리케이션이 시스템에 설치될 때이기 때문이다.

Activity 컴포넌트의 action 필터의 경우 기본적으로 정의되어 있는 필터값들이 ACTION_VIEW, ACTION_DIAL, ACTION_CALL, ACTION_SEND 등 20개에 이르지만 실제로 주로 사용되는 액션은 애플리케이션 첫 화면을 호출하는 ACTION_MAIN과 키보드 입력을 위한 노트패드 컴포넌트를 호출하는 ACTION_EDIT의 2가지뿐이다. 이들은 안드로이드 시스템에 기본적으로 포함되어 있는 Activity 컴포넌트들로써 시스템에 상주하고 있어 언제든 호출될 수 있기 때문이다.

반면에 일반 애플리케이션에 포함된 컴포넌트들은 해당 애플리케이션이 기기에 설치되어 있다는 보장이 없기 때문에 다른 애플리케이션들이 이 애플리케이션에 포함된 컴포넌트를 활용하기가 쉽지 않다. 결론적으로 안드로이드 시스템의 인텐트 메커니즘은 다른 애플리케이션의 컴포넌트를 동적으로 활용할 수 있어 코드 재사용의 기반을 제공하고 있기는 하지만 공유 가능한 범위가 동일한 안드로이

드 기기에 탑재된 애플리케이션들로 한정되어 있기 때문에 실질적으로는 항상 설치되어 있는 시스템 컴포넌트들만 공유되고 있는 것이 현실이다.

본 논문에서는 코드 공유를 보다 늘리기 위해 공유 가능한 애플리케이션들의 범위를 동일한 기기 뿐만 아니라 외부 서버 영역까지 넓힌 인텐트 메커니즘의 확장을 제안한다.

그림 6은 확장된 인텐트 메커니즘을 사용하는 애플리케이션의 구조이다. 패키지 매니저는 인텐트 해석 시에 기기에 설치된 인텐트 필터 정보뿐만 아니라 서버에 저장된 인텐트 필터 정보들도 사용한다.

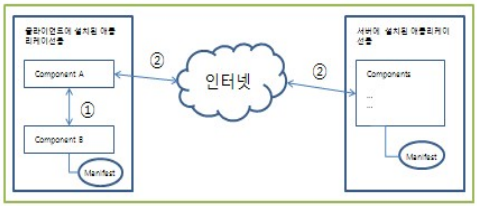

그림 6. 암시적 인텐트 사용 애플리케이션의 구조
Fig. 6. Application architecture for implicit intent

서버로는 다양한 애플리케이션들이 등록되어 있는 안드로이드 마켓이 해당된다. 그림 7과 같이 기존 인텐트 메커니즘에서는 패키지 매니저가 액티비티 매니저로부터 수신된 인텐트 객체를 해석할 때 기기내에 등록된 인텐트 필터 정보만을 사용하지만 확장 인텐트 메커니즘에서는 기기내 인텐트 필터에 찾고자 하는 컴포넌트가 없을 경우 서버의 인텐트 필터 정보까지 검색한다. 이를 위해 서버는 ServerPackageManagerService를 구동하고 각 기기에서 요청되는 인텐트 객체의 해석 서비스를 제공한다.

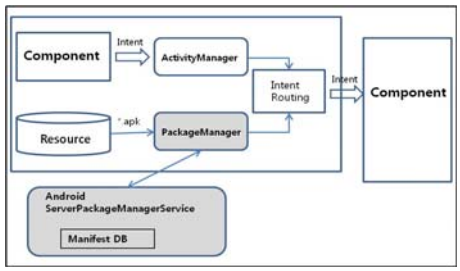

그림 7. 인텐트 메커니즘의 확장
Fig. 7. Extended intent mechanism

서버 패키지 매니저는 애플리케이션이 안드로이드마켓, 즉 서버에 등록될 때 APK 파일의 매니페스트 파일로부터 인텐트 필터 정보를 파싱하여 관리하고 기기로부터 수신된 인텐트 요청에 매칭되는 컴포넌트가 있을 경우 해당 컴포넌트를 기기로 전송한다.

그림 8은 PackageManager에서 Activity 인텐트 처리를 담당하는 queryIntentActivities() 메소드[10]를 확장한 코드이다.

```
@Override
public List<ResolveInfo> queryIntentActivities(Intent intent,
int flags) {
  try {
      return mPM.queryIntentActivities(intent, intent
              .resolveTypeIfNeeded(mContext
              .getContentResolver()), flags);
      } catch (RemoteException e) {
          return mPM.queryIntentActivities(serverIntent,
intent
                  .resolveTypeIfNeeded(mContext
                  .getContentResolver()), flags);
      } catch (ServerRuntimeException e) {
              throw new
                  RuntimeException("Package manager has
died", e);
      }
 }
```

그림 8. 확장된 패키지 매니저
Fig. 8. Extended package manager

확장된 메소드는 요청된 인텐트를 검색하여 해당 인텐트를 처리할 수 있는 컴포넌트들을 ResolveInfo 리스트로 돌려준다. 확장된 PackageManager는 1차적으로 기기 내에서 컴포넌트를 검색하고, 매칭되는 컴포넌트가 없으면 2차적으로 서버의 ServerPackageManagerService에 인텐트를 전달하여 해당 컴포넌트를 찾는다.

본 절에서는 확장 인텐트 매커니즘 구현을 설명하는데 Activity 컴포넌트에 대해서만 예를 들었지만 Broadcast Receiver 및 Service 컴포넌트에 대해서도 동일한 방식으로 적용될 수 있다.

3. 매니페스트 파일 설정

암시적 인텐트를 이용한 코드 재사용이 가능한 시스템은 동일한 시스템에 암시적 인텐트를 받아 처리할 수 있는 애플리케이션이 설치되어 있어야 하고 매니페스트에 해당 인텐트 필터가 정의되어 있어야 된다.

그러나 그림 9와 같이 암시적 인텐트를 처리할 수 있는 서버 애플리케이션의 명세와 매니페스트의 수도 코드를 이용하면 동일 기기내에 설치된 컴포넌트뿐만 아니라 서버에 설치된 공유 가능한 컴포넌트들의 목록을 얻을 수 있다.

```
• 암시적 인텐트를 처리할 수 있는 서버 애플리케이션측 명세 : 애플리케이션에서 지원 가능한 액션명, 카테고리 설정

• 매니페스트 파일
<intent-filter>
   <action android:name="서버에 등록된 컴포넌트의 액션명"/>
   <category android:name
="android.intent.category.DEFAULT" />
</intent-flter>
```

그림 9. 서버측 수도 코드
Fig. 9. Server-side pseudocode

IV. 결 론

본 논문에서는 안드로이드 스마트폰 애플리케이션 개발에서 컴포넌트 단위의 코드 재사용성을 높이기 위한 방안을 제안하고 안드로이드 운영체제를 확장하여 구현하였다. 제안한 방안은 기존 안드로이드 운영체제에서 애플리케이션간의 컴포넌트 재사용을 지원하는 인텐트 메커니즘을 동일 기기내의 애플리케이션뿐만 아니라 외부 서버에 있는 애플리케이션 간에도 컴포넌트 공유가 가능하도록 확장하였다. 이를 위해 PackageManager의 queryIntentActivities () 메소드를 확장하였다.

안드로이드마켓 서버에 등록된 다양한 애플리케이션에 포함된 컴포넌트들을 공유하여 프로그램 개발시 핵심 코드 이외의 나머지 부분을 실행중에 동적으로 불러다 쓰게 함으로써 개발 시간과 노력을 절약할 수 있다.

향후 연구방향으로는 암시적 인텐트를 보낼 때 액션 필터가 일치되어야만 해당 컴포넌트를 실행시킬 수 있는데 외부 서버에 있는 액션의 필터값을 모르더라도 명세 비교만으로 적합한 컴포넌트를 찾아낼 수 있는 필터 메커니즘을 개발하여 보다 높은 수준의 코드 재사용을 가능하게 하는 것이다.

또한 외부 서버에 있는 컴포넌트들을 공유할 때 악의적 코드가 포함된 컴포넌트들을 걸러낼 수 있는 보안 문제와 외부 서버 접속으로 인한 전송 지연 문제를 해결하고자 한다.

참고문헌

[1] admob, http://metrics.admob.com/wp-content/uploads/2009/06/admob-mobile-metrics-report-may-2009.pdf
[2] admob, AdMob Mobile Metrics Report, May. 2009.
[3] Android, http://www.android.com
[4] Android developers, http://developer.android.com
[5] http://www.kandroid.org
[6] 송형주, 김태연, 박지훈, 이백, 임기영, 인사이드 안드로이드, 위키북스, 2010.
[7] 배성호, 김우생, “안드로이드 기반 모바일 정보 공유시스템,” 대한전자공학회논문지-C1, Vol.46, No.2, pp. 58-64, 2009.
[8] 김경환, 하주호, 원종필, 이의성, 김주민, 손진호, “안드로이드 플랫폼 상에서 동기화가 고려된 통합 커서의 설계 및 구현,” 대한임베디드공학회 논문집, Vol.6, No.3, pp. 190-200, 2011.
[9] 리토 마이어, Professional Android2 Application Development, 제이펍, 2010.
[10] http://www.java2s.com/Open-Source/Android/ndroid-core/platform-frameworks-base/android/app/ContextImpl.java.htm

저 자 소 개

안 수 정 (Sujeong An)

1992년 : 경성대학교 물리학과 학사.
1996년 : 경성대학교 산업정보학과 석사.
2009년~현재, 경성대학교 컴퓨터공학과 박사과정.

관심분야 : 센서네트워크, 모바일 OS.
Email : won0@ks.ac.kr

김 병 호 (Byungho Kim)

1990년 : 연세대학교
전산과학과 학사.
1997년 : KAIST
전산학과 석박사.
2007년~현재, 경성대학교
컴퓨터학부 조교수.

관심분야 : 센서네트워크, 모바일 OS.
Email : bkim@ksu.ac.kr

김 진 천 (Jinchun Kim)

1983년 : 한양대학교
전기공학과 학사.
1985년 : 미시간주립대학교
전자 및 시스템공학과 석사.
1996년 : KAIST
전산학과 박사.

1988~1996년 : 삼성종합기술원 선임연구원.
1996년~현재, 경성대학교 컴퓨터학부 교수.
관심분야 : HCI, 멀티미디어통신, 센서네트워크.
Email : jckim@ks.ac.kr