한국컴퓨터정보학회 동계학술대회 논문집 제24권 제1호 (2016. 1)

안드로이드 어플리케이션의 효율적인 개발을 위한 로그 서비스 관리 모델

최윤석O
O동덕여자대학교 컴퓨터학과
e-mail: cooling@dongduk.ac.krO

Log Service Management Model for Effective Developments of Android Applications

Yun-seok ChoiO
ODept. of Computer Science, Dongduk Women's University

● 요 약 ●

본 논문에서는 안드로이드 기반의 효율적인 모바일 소프트웨어 개발을 위한 로그 서비스 관리 모델을 제안한다. 제안한 모델은 관점지향 프로그래밍을 적용하여 로깅 대상의 변경 없이 로그를 획득하며, 획득한 로그를 원격 호출 인터페이스를 사용하여 로깅 관리자에 전달한다. 로그 관리자는 안드로이드 기본 컴포넌트인 서비스 형태로 구성하며, 로깅 대상 어플리케이션 수행에 독립적으로 로그를 관리한다. 제안한 모델은 로깅을 위한 어플리케이션 변경을 최소화하고, 로깅이 어플리케이션 실행에 미치는 영향을 감소시킬 수 있다.

키워드: 모바일 소프트웨어(mobile software), 안드로이드(Android), 로깅(logging), 관점지향 프로그래밍(Aspect-oriented programming)

I. 서론

안드로이드 어플리케이션의 디버깅을 위해서는 로깅의 사용이 필수적이며, 이에 따라 운영체제 수준에서 로그를 활용할 수 있는 방법을 제공하고 있다. 로깅은 횡단관심사로 분류할 수 있으며 관점지향 프로그래밍(AOP: aspect-oriented programming)[1]을 적용할 경우 효과적으로 모듈화 할 수 있다. 그러나 한정된 자원과 이동성을 갖고 있는 모바일 기기의 특성상 AOP를 적용한 로깅도 대상 어플리케이션의 변경과 실행속도 등에 영향을 줄 수 있다. 따라서 로그 수집이 어플리케이션 실행에 주는 영향을 최소화하고 모바일 자원 소비를 최소화할 수 있는 로그 정보의 관리 방법이 필요하다. 이에 본 논문에서는 AOP와 안드로이드 어플리케이션 컴포넌트인 서비스(service)를 활용한 로그 서비스 관리 모델을 제안한다. 제안한 모델은 대상 어플리케이의 변경을 최소화하며, 독립적인 프로세스로 로그를 관리하므로 대상 어플리케이션의 실행에 대한 영향을 감소시킬 수 있다.

II. 관련 연구

1. 안드로이드 어플리케이션 구성요소와 로그

안드로이드 어플리케이션은 프레임워크에서 제공하는 액티비티(activity), 서비스, 브로드캐스트 리시버(broadcast receiver), 컨텐트 프로바이더(content provider), 그리고, 이들 사이의 메시지 전달을 담당하는 인텐트(intent)를 사용하여 구성할 수 있다[2]. 이중 서비스는 백그라운드 서비스 또는 원격호출 서비스 형태로 구성할 수 있으며, 원격 호출 형태로 구현한 서비스는 개별 어플리케이션 사이에서 원격 호출이 가능하다.

안드로이드는 운영체제 차원에서 로그 정보를 제공하며, 로그 정보는 Logcat을 활용하여 개발자가 살펴볼 수 있다. 개발자는 Log 클래스를 활용하여 필요한 곳에 로그 정보 확인 코드를 추가할 수 있다.

3. 안드로이드 관점지향 프로그래밍

안드로이드 어플리케이션 개발 시 관점지향 프로그래밍을 적용할 수 있는 도구로 AspectJ를 활용할 수 있다[3]. AspectJ는 안드로이드 어플리케이션을 개발할 수 있는 통합개발환경인 eclipse의 플러그인 형태로 활용할 수 있으며, 안드로이드 어플리케이션의 클래스와는 별도로 애스펙트라는 모듈 단위로 횡단관심사를 구현한다[4]. (그림 1)은 AspectJ로 구현한 애스펙트의 예를 나타낸다.

```
public aspect EventTracker {

    ArrayList<String> logs = new ArrayList<String>();
    long start, stop;

    // pointcut
    pointcut traceExecutionMethods() :
        ( execution (* mobile..*(..))   || execution (*..new(..)) )
            && !within(EventTracker);

    // advice
    before() : traceExecutionMethods() {
        start = System.nanoTime();
        String logString = getDetailedSignature(thisJoinPoint);
        logs.add(logString.toString());
        stop = System.nanoTime();
        logs.add("total: " + Long.valueOf(stop-start).toString());
    }

}
```

그림 1 애스펙트의 예

Fig. 1. An Example of Aspect

III. 본 론

1. 로그 서비스 관리 모델

로그 서비스 관리 모델은 로그 수집기와 로그 관리자로 구성한다. 로그 수집기는 AOP를 적용하여 어플리케이션의 로깅 대상을 식별하고 로그 정보를 수집하며, 수집한 정보를 원격 호출을 통해 로그 관리자에게 전달한다. 로그 관리자는 독립적인 안드로이드 서비스로 구성한다. (그림 2)는 로그 서비스 관리 모델의 구성을 나타낸다.

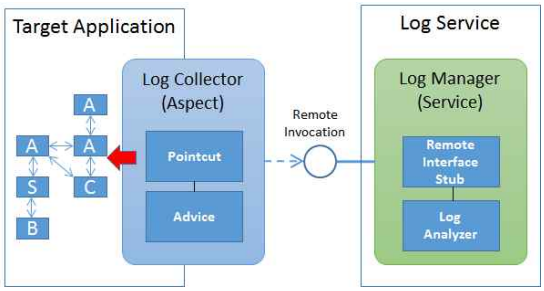

그림 2 로그 서비스 관리 모델

Fig. 2. Log Service Management Model

1.1 로그 수집기

로그 수집기는 로그 수집 부분과 수집한 로그를 로그 관리자에게 전달하기 위한 원격 호출 인터페이스로 구성한다. 로그 수집기는 AOP를 적용하여 애스펙트로 구성하며, 어플리케이션의 로깅 대상을 지정하고 로깅 정보를 수집 후 수집한 정보를 원격 호출 인터페이스를 사용하여 로그 매니저에게 전달한다. 다음 (그림 3)은 로그 수집기의 순차도를 나타낸다.

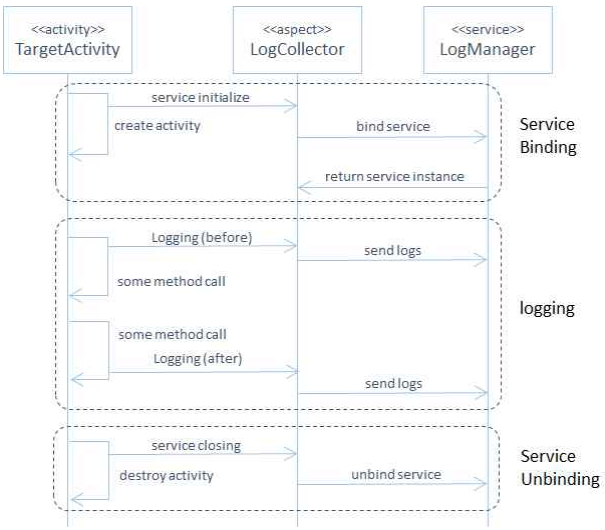

그림 3 로그 수집기 순차도

Fig. 3. Sequence Diagram of Log Collector

애스펙트로 구성한 로그 수집기는 서비스 바인딩, 로깅 수행, 서비스 언바이딩을 수행할 포인트컷을 지정하고, 액티비티 생명주기와 로깅 대상 메소드를 식별하여 결합점으로 지정한다. 안드로이드 어플리케이션의 사용자 인터페이스를 담당하는 컴포넌트인 액티비티는 실행 시 생명주기 메소드인 onCreate을 수행하여 초기화를 수행하며, 종료 시에는 onDestory를 수행한다. 따라서 로그 수집기는 로깅 대상 어플리케이션의 최초 액티비티가 갖고 있는 onCreate 메소드를 결합점으로 지정하여 로그 서비스 바인딩을 수행하고, 액티비티의 onDestroy 메소드 실행 시에는 서비스 언바인딩을 수행한다. 로깅 대상이 되는 액티비티 내의 메소드 실행 시에는 필요에 따라 사전 또는 사후로 로깅을 수행하는 애스펙트 충고(advice)의 결합시점을 지정할 수 있다. 로깅의 경우 메소드 수행 전후의 결과가 주요관심사가 될 수 있으므로 메소드 수행 중의 시점은 로깅에서 제외할 수 있다.

로그 수집기는 원격 호출 인터페이스의 잦은 호출을 줄이기 위해 수집한 로그를 일정 크기의 버퍼에 저장한다. 버퍼가 다 찼을 경우나 로깅을 종료하여야 할 시점에 다다르면 버퍼에 저장한 로그의 집합을 원격 호출 인터페이스를 통해 로그 관리자에게 전송한다. 로그 관리자는 전달받은 로그의 집합을 분리하여 개발자가 활용할 수 있는 형태의 로그로 재구성한다.

1.2 로그 관리자

로그 관리자는 안드로이드 프레임워크에서 제공하는 서비스 컴포넌트를 활용하여 구성한다. 로그 수집기에서 수집한 로그 정보를 원격 호출 인터페이스를 통해 전달 받으며, 이를 처리하여 로그 정보를 구성하는 역할을 담당한다. 로그 관리자는 원격 호출 인터페이스에 대한 스텁과 스텁을 통해 전달 받은 로그를 분석하여 이를 활용 가능한 로그 형태로 분석하는 로그 분석기로 구성한다.

원격 호출 서비스 구현을 위해서는 안드로이드의 인터페이스 정의 언어인 AIDL(Android Interface Definition Language)을 사용하여

원격 호출 인터페이스를 정의한다[5]. AIDL을 사용하여 정의한 인터페이스를 구현하는 스텁은 로그 관리자의 멤버로 선언하여 로그 처리 기능을 구성하며, 원격 인터페이스 구성 시 필요한 onBind() 메소드를 통해 로그 수집기에 해당 객체를 전달한다.

2. 로그 서비스 구현

제안한 로그 관리 서비스 모델을 적용하기 위하여 로그 관리 서비스 어플리케이션을 구현하였다. 구현한 로그 관리 서비스 어플리케이션은 AspectJ를 사용하여 애스펙트로 구현한 로그 수집기와 안드로이드 프레임워크 컴포넌트인 서비스를 상속받아 구현한 로그 관리자로 구성하였다. (그림 4)는 AspectJ로 구현한 LogCollector 애스펙트를 나타낸다.

```
public aspect LogCollector {
    LogServiceUtil logServiceUtil;
    ArrayList<String> logList;

    pointcut initService(Context context)
    : execution (* mobile..MainActivity.onResume(..))
                && target(context);

    before(Context context) : initService(context) {
        logServiceUtil = LogServiceUtil.getInstance(context);
        logList = new ArrayList<String>();
    }

    pointcut unbindService(Context context)
    : execution (* mobile..MainActivity.onBackPressed(..))
                && target(context);

    before(Context context) : unbindService(context) {
        if (logServiceUtil != null)
                logServiceUtil.unbindLogService(context);
    }

    pointcut sendLog() : execution (* mobile..onClick(..));

    before() : sendLog() {
        logServiceUtil.sendLog
            (thisJoinPointStaticPart.toShortString());
    }
}
```

그림 4 Log Collector 애스펙트
Fig. 4. Log Collector Aspect

로그 수집을 담당하는 LogCollector 애스펙트는 수집한 로그를 LogServiceUtil에 전달한 후 이를 로그 관리자인 LogService에 전달한다.

로그 관리자 역할을 수행하는 LogService를 구성하기 위해 AIDL을 사용하여 로그 수집기가 호출 시 사용할 원격 인터페이스를 정의하고, 안드로이드의 프레임워크의 컴포넌트인 서비스를 상속받아 구현하였다. 로그의 영구적인 보전 및 관리를 위해 유틸 클래스로 LogWriter를 추가하였다. (그림 5)는 LogService의 원격 호출 인터페이스를 정의한 AIDL과 안드로이드 서비스로 구현한 LogService를 나타낸다.

```
// Remote Invocation Interface
interface ILogger {
    boolean putLogMessgage(in String str);
}
// Log Service Manager
public class LogService extends Service {
    LogWriter logWriter;

    @Override
    public void onCreate() {
        super.onCreate();
        logWriter = new LogWriter(this);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    ILogger.Stub mBinder = new ILogger.Stub() {
        @Override
        public boolean putLogMessgage(String str)
                throws RemoteException {
            StringTokenizer st
            = new StringTokenizer
                (str, System.getProperty("line.separator"));
            while(st.hasMoreTokens()) {
                String log = "log: " + st.nextToken();
                logWriter.writeLogs(log);
            }
            return true;
        }
    };
}
```

그림 5 AIDL과 LogService
Fig. 5. AIDL & LogService for Log Management

로그 관리 서비스를 LBS 기반의 안드로이드 어플리케이션에 적용하여 로그를 수집할 수 있었으며, 다음 (그림 6)은 수집한 로그 결과의 일부를 나타낸다.

```
log: execution(MainActivity.onClick(..))
log: execution(AddNewLocationActivity.onClick(..))
log: execution(DetailedLocationActivity.onClick(..))
log: execution(DetailedLocationActivity.onClick(..))
log: execution(MainActivity.onClick(..))
```

그림 6 로깅 결과
Fig. 6. Result of Logging

Ⅳ. 결 론

본 연구에서는 AOP와 안드로이드 프레임워크의 컴포넌트인 서비스를 활용한 로그 관리 서비스 모델을 제안하였다. 제안한 모델 상의 로그 수집기는 AOP를 적용하여 애스펙트로 구성하므로 대상 어플리케이션의 변경 없이 로그 정보를 수집할 수 있으며, 로그 관리자는 독립적인 서비스로 구성하므로 로그 수집 및 처리가 대상 어플리케이션 실행에 미치는 영향을 감소시킬 수 있다.

향후에는 안드로이드 어플리케이션의 실행 흐름과 오류를 효과적으로 파악할 수 있도록 로그 수집기를 구성하는 방법에 대한 연구가

필요하며, 단말기 외부에서 로그를 확인할 수 있도록 원격 로깅을 수행하는 방법에 대한 연구가 필요하다.

References

[1] G. Kiczales, J. Irwin, J. Lamping, J. M. Loingtier, C. Lopes, C. Maeda, and A. Mendhekar, "Aspect-Oriented Programming", Proceedings of the European Conference on Object-Oriented Programming(ECOOP'97), Springer-Verlag, Finland, pp.220-242, June 1997.

[2] Android Application Fundamentals, http://developer.android.com

[3] R. Laddad, AspectJ in Action, Manning Publications, 2005.

[4] AJDT: AspectJ, http://www.eclipse.org/aspectj/

[5] Android Interface Definition Language (AIDL), http://developer.android.com/guide/components/aidl.html