안드로이드 시스템 서비스 구현하기

1. AIDL 컴파일러에 의해서 서비스 인터페이스를 자동으로 생성해주는 aidl 파일 작성
/frameworks/base/core/java/android/app/ISimpleSystemService.aidl

```
package android.app;

/**
*
*
*{@hide}
*/

interface ISimpleSystemService{
        void serviceMethod();
}
```

2. 위의 aidl 파일을 인지하고 AIDL 컴파일러에 의해 컴파일 될 수 있도록 추가해준다.
LOCA_SRC_FILES += ₩ 부분에 다른 aidl 파일들과 같은 형식으로 작성한다.
/frameworks/base/Android.mk

```
LOCAL_SRC_FILES += \
    core/java/android/accessibilityservice/IAccessibilityServiceConnection.aidl \
    core/java/android/accessibilityservice/IAccessibilityServiceClient.aidl \
    core/java/android/accounts/IAccountManager.aidl \
    core/java/android/accounts/IAccountManagerResponse.aidl \
    core/java/android/accounts/IAccountAuthenticator.aidl \
    core/java/android/accounts/IAccountAuthenticatorResponse.aidl \
    core/java/android/app/IActivityController.aidl \
    core/java/android/app/IActivityPendingResult.aidl \
    core/java/android/app/IAlarmManager.aidl \
    core/java/android/app/IHelloWorld.aidl \
    core/java/android/app/ISimpleSystemService.aidl \
```

3. aidl 을 통해서 자동으로 만들어진 ISimpleSystemService.Stub 클래스를 상속받아 구현한다.

/frameworks/base/services/java/com/android/server/SimpleSystemService.java

```
package com.android.server;

import android.app.ISimpleSystemService;
import android.content.Context;
import android.os.RemoteException;
import android.util.Slog;

public class SimpleSystemService extends ISimpleSystemService.Stub{
	static final String TAG = "SimpleSystemService";
	Context mContext;
	public static int i=0;
	static DoSomething thread;

	public SimpleSystemService(Context context){
		mContext = context;
		Slog.i(TAG, "SimpleSystemService is start");

		thread = new DoSomething();
		thread.start();
	}

	public void serviceMethod() throws RemoteException{
		Slog.i(TAG, "call serviceMethod()");
		Slog.i(TAG, i+"");
	}
}
class DoSomething extends Thread{
	@Override
	public void run() {
		// TODO Auto-generated method stub
		while(true){
			SimpleSystemService.i++;
			try {
				Slog.i(SimpleSystemService.TAG,
						SimpleSystemService.i+"");
				Thread.sleep(1000);
			} catch (Exception e) {
				// TODO: handle exception
				e.printStackTrace();
			}
		}
	}
}
```

4. 작성한 서비스를 SystemServer 에 등록시켜 줘야 한다.

Service 를 ServiceManager 가 Map 형식으로 관리한다.
키 값은
/frameworks/base/services/java/com/android/server/SystemService.java

```
// Bring up services needed for UI.
        if (factoryTest != SystemServer.FACTORY_TEST_LOW_LEVEL) {
            try {
                    SimpleSystemService simpleSystemService =
                                                        new SimpleSystemService(context);
                    ServiceManager.addService(Context.SIMPLE_SYSTEM_SERVICE,
                                                        (IBinder)simpleSystemService);
            } catch (Throwable e) {
                    // TODO: handle exception
                    Slog.e(TAG, "Fail..");
            }
```

5. Context.SIMPLE_SYSTEM_SERVICE 문자열을 추가해준다.

/frameworks/base/core/java/android/content/Context.java

```
        public static final String SIMPLE_SYSTEM_SERVICE = "simplesystemservice";
```

6. service 를 구현했으면 service 를 개발자가 사용할 수 있도록 Manager 를 구현한다.

Manager 는 Service 인스턴스를 만들어서 해당하는 메소드를 호출시켜주면 된다.
/frameworks/base/core/java/android/app/SimpleSystemServiceManager.java

```
package android.app;

import android.os.RemoteException;

public class SimpleSystemServiceManager {
        private final ISimpleSystemService mService;

        SimpleSystemServiceManager(ISimpleSystemService service){
                super();
                mService = service;
        }
        public void serviceMethod(){
                try {
                        mService.serviceMethod();
                } catch (RemoteException e) {
                        // TODO: handle exception
                }
        }
}
```

7. 개발자가 getSystemService 를 통해서 이 서비스를 이용할 수 있도록 해준다.

Service 를 가지고 있는 Manager 를 return 한다.

/frameworks/base/core/java/android/app/ContextImple.java

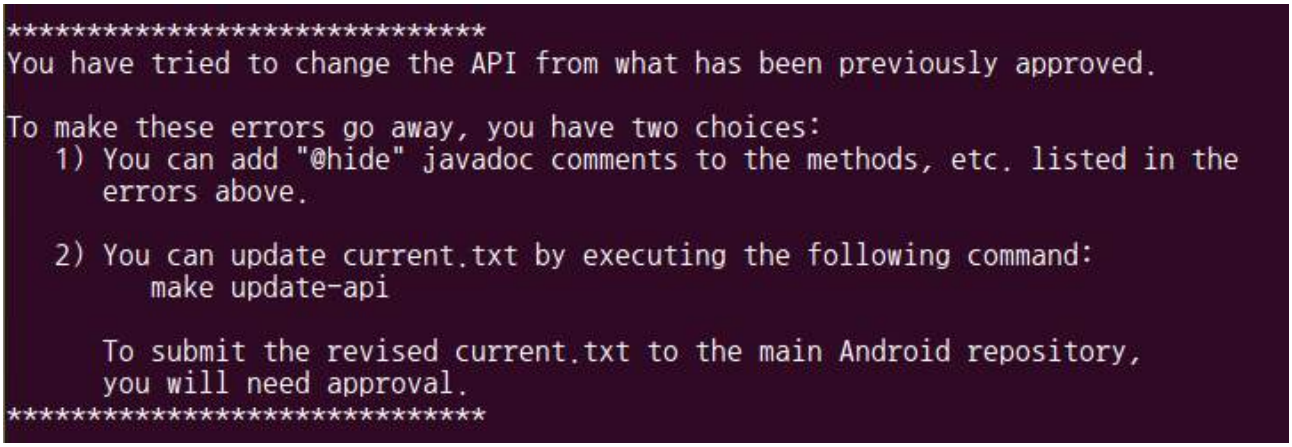
```
static {
    registerService(ALARM_SERVICE, new ServiceFetcher() {
        public Object createService(ContextImpl ctx) {
            IBinder b = ServiceManager.getService(ALARM_SERVICE);
            IAlarmManager service = IAlarmManager.Stub.asInterface(b);
            return new AlarmManager(service, ctx);
        }});

    registerService(HELLO_SERVICE, new ServiceFetcher() {
        public Object createService(ContextImpl ctx){
                IBinder b = ServiceManager.getService(HELLO_SERVICE);
                IHelloWorld service = IHelloWorld.Stub.asInterface(b);
                Log.d("mymymymymymy", "hello world service manager is loaded");
                return new HelloWorldManager(service);
        }
    });

    registerService(SIMPLE_SYSTEM_SERVICE, new ServiceFetcher(){
        public Object createService(ContextImpl ctx){
                IBinder b = ServiceManager.getService(SIMPLE_SYSTEM_SERVICE);
                ISimpleSystemService service = ISimpleSystemService.Stub.asInterface(b);
                return new SimpleSystemServiceManager(service);
        }
    });
```

8. 서비스를 작성했으면 빌드 한다.

빌드를 할 때 SDK API 가 변경된것을 알리는 메시지가 나온다.



Make update-api 를 통해서 api 를 업데이트 해주고 다시 make 로 빌드를 진행하면 된다.

9. 개발자가 사용할 때는 Manager 를 getSystemService 로 얻어서 사용한다.

```
SimpleSystemServiceManager simpleSystemService =
                                (SimpleSystemServiceManager)getSystemService("simplesystemservice");
simpleSystemService.serviceMethod();
```

시스템 서비스에서 parameter 사용하기

---

http://developer.android.com/guide/components/aidl.html

Primitive Type 변수들은 특별한 처리 없이 사용할 수 있다.

다른 변수들은 in 이라는 키워드를 통해서 받는 parameter 라는 것을 명시해주어야 한다.

```
package android.app;

/**
*
*
*{@hide}
*/

interface ISimpleSystemService{
	void serviceMethod();
	int getInteger();
	void setInteger(int integer);
	String getString();
	void setString(in String string);
	MyObject getMyObject();
	void setMyObject(in MyObject myObject);
}
```

사용자가 정의한 객체를 전달하기 위해서는
해당 객체에 대한 인터페이스를 정의하는 aidl 파일을 작성해야한다.

/frameworks/base/core/java/android/app/MyObject.aidl

```
package android.app;

parcelable MyObject;
```

그리고 객체를 작성해주면 된다. 객체를 작성할 때는 parcelable 을 implements 해줘야 한다.
/frameworks/base/core/java/android/app/MyObject.java

```
package kr.ac.mju.hmcl.systemservice;

import android.os.Parcel;
import android.os.Parcelable;


public class MyObject implements Parcelable{

	public int a;

	public MyObject(){}

	public MyObject(Parcel in){
		readFromParcel(in);
	}

	public static final Parcelable.Creator<MyObject> CREATOR =
						new Parcelable.Creator<MyObject>(){
		public MyObject createFromParcel(Parcel in){
			return new MyObject(in);
		}

		@Override
		public MyObject[] newArray(int size) {
			return new MyObject[size];
		}
	};

	@Override
	public void writeToParcel(Parcel dest, int flags) {
		// TODO Auto-generated method stub
		dest.writeInt(a);
	}

	public void readFromParcel(Parcel in){
		a = in.readInt();
```

```
	}

	@Override
	public int describeContents() {
		// TODO Auto-generated method stub
		return 0;
	}
}
```

/frameworks/base/services/java/com/android/server/SimpleSystemService.java

```
package com.android.server;

import android.app.ISimpleSystemService;
import android.app.MyObject;
import android.content.Context;
import android.os.RemoteException;
import android.util.Slog;

public class SimpleSystemService extends ISimpleSystemService.Stub{
	static final String TAG = "SimpleSystemService";
	Context mContext;
	public static int i=0;
	private int integer;
	private String string;
	static DoSomething thread;
	private MyObject myObject;

	public SimpleSystemService(Context context){
		mContext = context;
		Slog.i(TAG, "SimpleSystemService is start");

		thread = new DoSomething();
		thread.start();
	}

	public void serviceMethod() throws RemoteException{
		Slog.i(TAG, "call serviceMethod()");
		Slog.i(TAG, i+"");
	}

	public int getInteger() throws RemoteException{
		Slog.i(TAG, "call getInteger");
		return integer;
	}

	public void setInteger(int integer) throws RemoteException{
		Slog.i(TAG, "call setInteger");
		this.integer = integer;
	}

	public String getString() throws RemoteException{
		Slog.i(TAG, "call getString");
		return string;
	}

	public void setString(String string) throws RemoteException{
		Slog.i(TAG, "call setString");
```

```
                this.string = string;
        }

        public void setMyObject(MyObject myObject) throws RemoteException{
                Slog.i(TAG, "call setMyObject");
                this.myObject = myObject;

        }

        @Override
        public MyObject getMyObject() throws RemoteException {
                // TODO Auto-generated method stub
                return myObject;
        }
}

class DoSomething extends Thread{
        @Override
        public void run() {
                // TODO Auto-generated method stub
                while(true){
                        SimpleSystemService.i++;
                        try {
                                Slog.i(SimpleSystemService.TAG, SimpleSystemService.i+"");
                                Thread.sleep(1000);
                        } catch (Exception e) {
                                // TODO: handle exception
                                e.printStackTrace();
                        }
                }
        }
}
```

/frameworks/base/core/java/android/app/SimpleSystemServiceMnager.java

```
package android.app;

import android.os.RemoteException;
import android.app.MyObject;

public class SimpleSystemServiceManager {
	private final ISimpleSystemService mService;

	SimpleSystemServiceManager(ISimpleSystemService service){
		super();
		mService = service;
	}

	public void serviceMethod(){
		try {
			mService.serviceMethod();
		} catch (RemoteException e) {
			// TODO: handle exception
		}
	}

	public int getInteger(){
		try{
			return mService.getInteger();
		} catch (RemoteException e){
			return 0;
		}
	}

	public void setInteger(int integer){
		try {
			mService.setInteger(integer);
		} catch (RemoteException e) {
			// TODO: handle exception
		}
	}

	public String getString(){
		try {
			return mService.getString();
		} catch (RemoteException e) {
			// TODO: handle exception
			return null;
		}
	}

	public void setString(String string){
```

```
                try {
                        mService.setString(string);
                } catch (RemoteException e) {
                        // TODO: handle exception
                }
        }

        public void setMyObject(MyObject myObject){
                try {
                        mService.setMyObject(myObject);
                } catch (RemoteException e) {
                        // TODO: handle exception
                }
        }

        public MyObject getMyObject(){
                try {
                        return mService.getMyObject();
                } catch (Exception e) {
                        return null;
                }
        }
}
```

Primitive Type 변수를 사용한 결과

D	07-17 00:34:58.599	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	SystemService has 24
I	07-17 00:34:58.599	363	608	system_process	SimpleSystemService	call getInteger
I	07-17 00:34:58.749	363	607	system_process	SimpleSystemService	call getInteger
I	07-17 00:34:58.749	363	375	system_process	SimpleSystemService	call setInteger
D	07-17 00:34:58.759	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	SystemService has 25
I	07-17 00:34:58.759	363	409	system_process	SimpleSystemService	call getInteger

String Type 을 사용한 결과

I	07-17 00:34:26.299	363	409	system_process	SimpleSystemService	call getString
D	07-17 00:34:26.319	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	SystemService has nullaaaaa
I	07-17 00:34:26.399	363	393	system_process	SimpleSystemService	85
I	07-17 00:34:26.469	363	615	system_process	SimpleSystemService	call getString
I	07-17 00:34:26.469	363	374	system_process	SimpleSystemService	call setString
D	07-17 00:34:26.469	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	SystemService has nullaaaaaa

객체를 사용한 결과

D	07-17 00:36:47.169	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	MyObject has 11
I	07-17 00:36:47.169	363	543	system_process	SimpleSystemService	call setMyObject
D	07-17 00:36:47.309	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	MyObject has 12
I	07-17 00:36:47.309	363	409	system_process	SimpleSystemService	call setMyObject
D	07-17 00:36:47.459	1080	1080	kr.ac.mju.hmcl.systemservice	SystemService	MyObject has 13
I	07-17 00:36:47.459	363	542	system_process	SimpleSystemService	call setMyObject