第一個 Core Service 與實機演練

一、撰寫你的第一個核心服務

1. 爲什麼要寫核心服務呢？

- 因爲底層核心服務是 Android 框架裡最接近 Linux/Driver 的部分。爲了充分發揮硬件設備的差異化特性，核心服務是讓上層 Java 應用程序來使用 Driver/HW Device 特色的重要管道。
- 在開機過程中，就可以啓動核心服務(例如漢字輸入法服務等)，讓眾多應用程序來共用之。
- 由於共用，所以能有效降低 Java 應用程序的大小(Size)。

2. 如何撰寫自己的第一個核心服務呢?

要點：

- 核心服務通常在獨立的進程(Process)裡執行。
- 必須提供 IBinder 接口，讓應用程序可以進行跨進程的綁定(Binding)和呼叫。
- 因爲共用，所以必須確保多線程安全(Thread-safe)。
- 以 C++類定義，誕生其對象，透過 SM 之協助，將該對象參考值傳給 IServiceManager::addService()函數，就加入到 Binder Driver 裡了。
- 應用程序可透過 SM 之協助而遠距綁定該核心服務，此時 SM 會回傳 IBinder 接口給應用程序。
- 應用程序可透過 IBinder::transact()函數來與核心服務互傳資料。

將 SQRService 核心服務加入 Binder Driver

此範例功能爲簡單的整數平方(Square)運算，此核心服務命名爲 SQRService。

Step-1：以 C++撰寫 SQRService 類，其完整程序碼爲：

// SQRService.h

```
#include <stdint.h>
#include <sys/types.h>
#include <utils/Parcel.h>

#ifndef ANDROID_MISOO_SQRSERVICE_H
#define ANDROID_MISOO_SQRSERVICE_H
#include <utils.h>
```

```
#include <utils/KeyedVector.h>
#include <ui/SurfaceComposerClient.h>

namespace android {
class SQRService : public BBinder
{
public:
        static    int       instantiate();
        virtual status_t onTransact(uint32_t, const Parcel&, Parcel*, uint32_t);
                                        SQRService();
        virtual                         ~SQRService();
};
};
#endif
```

// SQRService.cpp

```
#include <utils/IServiceManager.h>
#include <utils/IPCThreadState.h>
#include <utils/RefBase.h>
#include <utils/IInterface.h>
#include <utils/Parcel.h>
#include "SQRService.h"

namespace android {
enum {
      SQUARE = IBinder::FIRST_CALL_TRANSACTION,
};

int SQRService::instantiate(){
      LOGE("SQRService instantiate");
      int r = defaultServiceManager()->addService(
                  String16("misoo.sqr"), new SQRService());
      LOGE("SQRService r = %d\n", r);
      return r;
}
SQRService::SQRService(){
      LOGV("SQRService created");
}
SQRService::~SQRService(){
      LOGV("SQRService destroyed");
}
status_t SQRService::onTransact(
      uint32_t code, const Parcel& data, Parcel* reply, uint32_t flags)
{
      switch(code) {
            case SQUARE: {
                  int num = data.readInt32();
                  reply->writeInt32(num * num);
                  LOGE("onTransact::CREATE_NUM.. n=%d\n", num);
```

```
                return NO_ERROR;
            }
            break;
            default:
               LOGE("onTransact::default\n");
               return BBinder::onTransact(code, data, reply, flags);
        }
}
}; // namespace android
```

// Android.mk 檔

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_SRC_FILES:= \
        SQRService.cpp
LOCAL_C_INCLUDES := \
     $(JNI_H_INCLUDE)

LOCAL_SHARED_LIBRARIES := \
        libutils
LOCAL_PRELINK_MODULE := false

LOCAL_MODULE := libSQRS01
include $(BUILD_SHARED_LIBRARY)
```

執行上述 Android.mk 檔，就產出 libSQR01.so 檔案了。

Step-2：以 C++撰寫一個可獨立執行的 addserver.cpp 程序，它的用途是：誕生一個 SQRService 類之對象，然後將該對象參考存入 Binder Driver 裡。其內容爲：

// addserver.cpp

```
#include <sys/types.h>
#include <unistd.h>
#include <grp.h>

#include <utils/IPCThreadState.h>
#include <utils/ProcessState.h>
#include <utils/IServiceManager.h>
#include <utils/Log.h>

#include <private/android_filesystem_config.h>

#include "../libadd/SQRService.h"
//#include <libadd/SQRService.h>

using namespace android;

int main(int argc, char** argv)
{
```

```
    sp<ProcessState> proc(ProcessState::self());
    sp<IServiceManager> sm = defaultServiceManager();
    LOGI("ServiceManager: %p", sm.get());
    SQRService::instantiate();
    ProcessState::self()->startThreadPool();
    IPCThreadState::self()->joinThreadPool();
}
```

編譯並連結此 addserver.cpp，產出 addserver.exe 可執行程序。

Step-3：上述兩個步驟分別產出了 libSQR01.so 類庫和 addserver.exe 可執行程序了。接著將 libSQR01.so 拷貝到 Android 模擬器的/system/lib/裡；也把 addserver.exe 拷貝到/system/bin/裡。

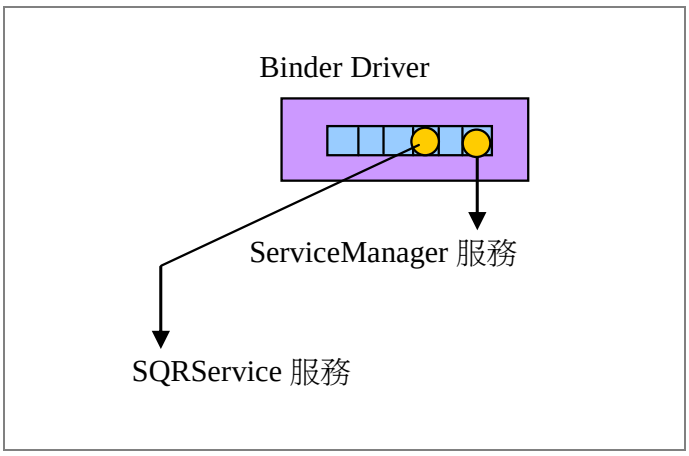
Step-4：執行 addserver.exe，其中的指令：

SQRServer::instantiate();

就執行到 SQRServer 類的 instantiate()函數，其內容爲：

```
int AddService::instantiate() {
    LOGE("AddService instantiate");
    int r = defaultServiceManager()->addService(
        String16("Misoo.sqr"), new SQRService());
    LOGE("AddService r = %d\n", r);
    return r;
}
```

其先執行到 new SQRServer()，就誕生一個 SQRServer 類之對象；接著，呼叫 defaultServiceManager()函數取得 SM 的 IServiceManager 接口，再呼叫 IServiceManager::addServer()將該對象參考存入 Binder Driver 裡，如下圖所示：



寫個 SQR 類來使用 SQRService 核心服務

剛才已經成功地將 SQRService 服務加入到 Binder Driver 裡了。現在，茲寫個 C++應用類來綁定(Bind)此核心服務。

Step-5：以 C++撰寫 SQR 類，其完整程序碼爲：

// SQR.h

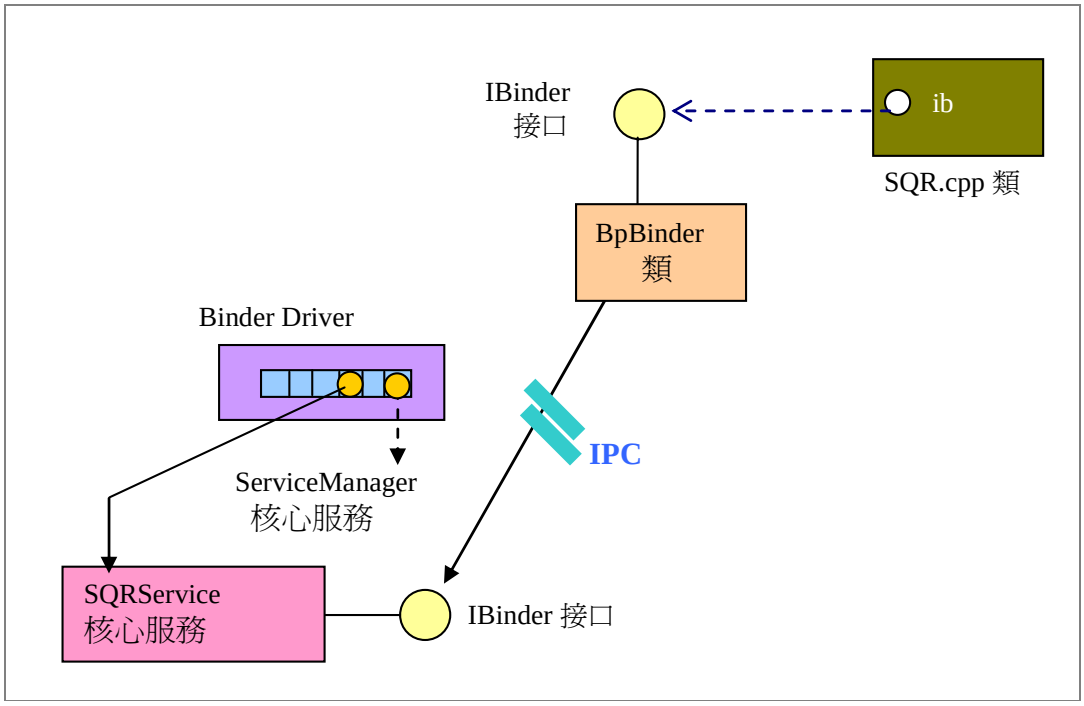
```
#ifndef ANDROID_MISOO_SQR_H
#define ANDROID_MISOO_SQR_H

namespace android {

class SQR {
private:
        const void getAddService();
public:
        SQR();
        int execute(int n);


};
}; //namespace
#endif // ANDROID_MISOO_SQR_H
```

// SQR.cpp

```
#include <utils/IServiceManager.h>
#include <utils/IPCThreadState.h>
#include "SQR.h"

namespace android {
   sp<IBinder> m_ib;

SQR::SQR(){
        getSQRService();
}

const void SQR::getSQRService(){
        sp<IServiceManager> sm = defaultServiceManager();
                        m_ib = sm->getService(String16("misoo.sqr"));
                        LOGE("SQR:getSQRService %p\n",sm.get());
                        if (m_ib == 0)
                                LOGW("SQRService not published, waiting...");
                        return;
}

int SQR::execute(int n) {
         Parcel data, reply;
         data.writeInt32(n);
         LOGE("SQR::execute\n");
         m_ib->transact(0, data, &reply);
         int num = reply.readInt32();
```

```
        return num;
    }
}; //namespace
```

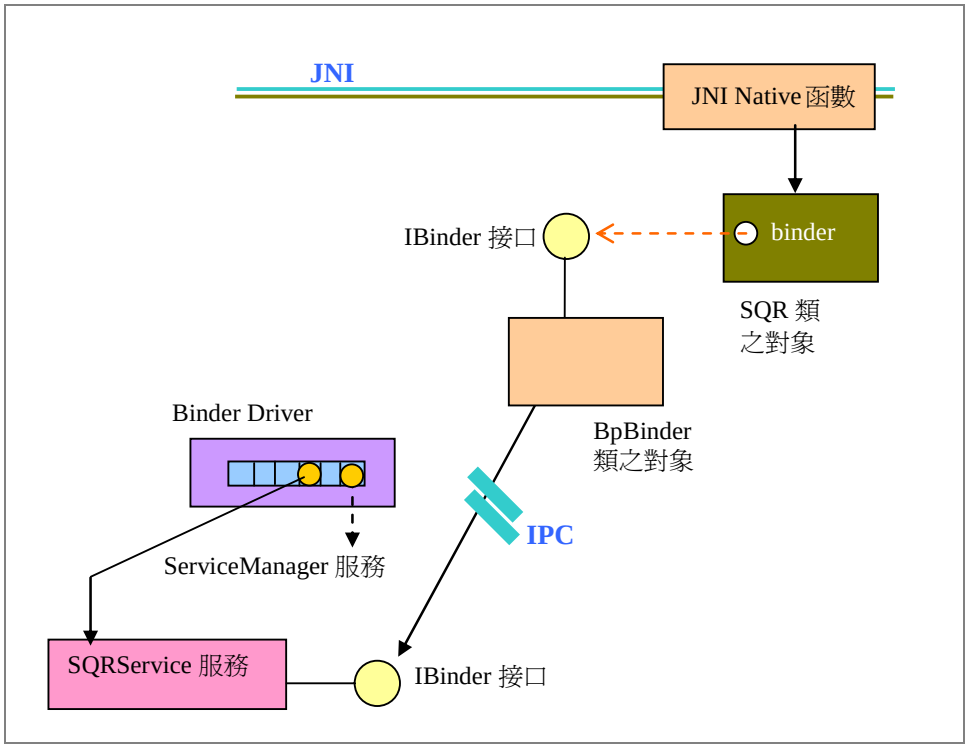
其中的 execute()函數呼叫 getSQRService()函數來取得 ServiceManager 的接口參考，然後要求 ServiceManager 去協助綁定 SQRService 核心服務，完成時 ServiceManager 會回傳 BpBinder 對象的 IBinder 接口參考，然後暫存於 binder 變數裡。如下圖所示：



二、實機演練

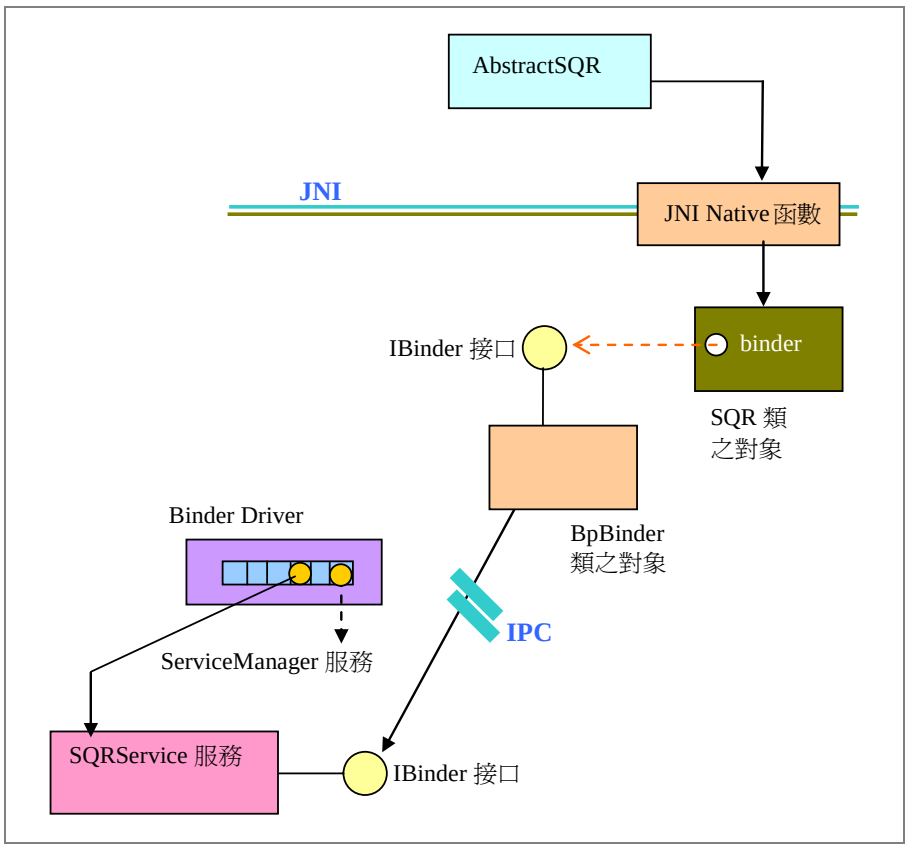
題目(一)：寫個 JNI Native 類來使用 SQR 類之對象

如下圖所示：



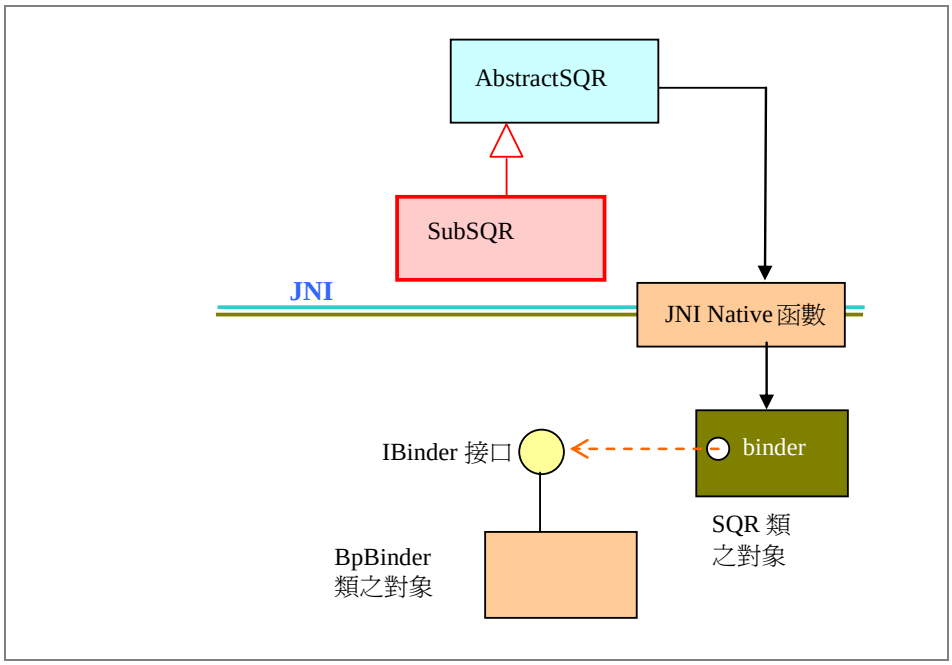
題目(二)：寫個 Java 層的父類(Superclass)來使用 JNI Native 函數

如下圖所示：



題目(三)：寫個 Java 層的子類(Subclass)來繼承 AbstractSQR 父類

如下圖所示：



~~ END ~~