Archived:Prompting the user to select a Bluetooth device using RNotifier

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Overview

This code snippet describes the Bluetooth device discovery mechanism provided by the S60 UI platform. The `RNotifier` class can be used to perform device discoveries where user interaction is required. By using the `TBTDeviceSelectionParams` class it is also possible to restrict the search to a more specific subset of devices.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY bluetooth.lib
LIBRARY btextnotifiers.lib
LIBRARY btdevice.lib
```

Header file

```
#ifndef BTDEVICEFINDER_H_
#define BTDEVICEFINDER_H_

#include <e32base.h>
#include <BTExtNotifiers.h> //TBTDeviceSelectionParamsPckg,TBTDeviceResponseParamsPckg
     class MBTDeviceFinderNotify
         {
         public:
                 virtual void BTDeviceSelected() = 0;
         };

     class CBTDeviceFinder: public CActive
         {
         public:
                 static CBTDeviceFinder* NewL(const TInt aPriority,
                                              MBTDeviceFinderNotify& aNotify);
                 ~CBTDeviceFinder();
         public:
                 void DiscoverDevicesL();
                 void SetSelectionParams(const TBTDeviceSelectionParams& aFilter);
                 const TBTDeviceResponseParams& ResponseParams();
         protected:
                 void RunL();
                 void DoCancel();
         private:
                 CBTDeviceFinder(const TInt aPriority,MBTDeviceFinderNotify& aNotify);
                 void ConstructL(void);
         private:
             RNotifier iSelectorNotifier;
             MBTDeviceFinderNotify&        iNotify;
             TBTDeviceSelectionParamsPckg iSelectParamsBuf;
             TBTDeviceResponseParamsPckg iResponseParamsBuf;
             TBool iConnected;
         };
#endif /*BTDEVICEFINDER_H_*/
```

Source file

```
#include "BTDeviceFinder.h"

CBTDeviceFinder::CBTDeviceFinder(const TInt aPriority,
                 MBTDeviceFinderNotify& aNotify)
:CActive(aPriority),iNotify(aNotify),iConnected(EFalse)
{
}

CBTDeviceFinder::~CBTDeviceFinder()
{
  Cancel();
  if (iConnected)
    {
    iSelectorNotifier.CancelNotifier( KDeviceSelectionNotifierUid );
    iSelectorNotifier.Close();
    }
}

CBTDeviceFinder* CBTDeviceFinder::NewL(const TInt aPriority,
                                 MBTDeviceFinderNotify& aNotify)
{
    CBTDeviceFinder* self = new (ELeave) CBTDeviceFinder(aPriority,aNotify);
```

 (C) Copyright Nokia 2013. All rights reserved.

```
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop();
    return self;
}

void CBTDeviceFinder::ConstructL(void)
{
  CActiveScheduler::Add(this);
}

void CBTDeviceFinder::DoCancel()
{
  iSelectorNotifier.CancelNotifier( KDeviceSelectionNotifierUid );
}

void CBTDeviceFinder::RunL()
{
  //the user has selected some device
  if(iStatus.Int() == KErrNone)
        {
        iNotify.BTDeviceSelected();
        }
  else //an error has occurred
        {
        User::Leave(iStatus.Int());
        }
}

void CBTDeviceFinder::DiscoverDevicesL()
{
  if (!iConnected)
        {
        User::LeaveIfError(iSelectorNotifier.Connect());
        iConnected = ETrue;
        }

  iSelectorNotifier.StartNotifierAndGetResponse(iStatus,
                  KDeviceSelectionNotifierUid,
                  iSelectParamsBuf,
                  iResponseParamsBuf);
  SetActive();
}

void CBTDeviceFinder::SetSelectionParams(const TBTDeviceSelectionParams& aFilter)
{
  iSelectParamsBuf = aFilter;
}

const TBTDeviceResponseParams& CBTDeviceFinder::ResponseParams()
{
  return iResponseParamsBuf();
}
```

Using CBTDeviceFinder class

- In the header file:

```
class CBTDeviceFinder;

class CTestingAppUi : public CAknAppUi, MBTDeviceFinderNotify
  {
public:
  //...
  //from MBTDeviceFinderNotify
  void BTDeviceSelected();
  //...
private:
  //...
  CBTDeviceFinder* iFinder;
  };
```

- In the source file:

```
#include "btdevicefinder.h"
#include <btdevice.h> //TBTDeviceClass, TBTDeviceName
#include <bttypes.h> //TBTDevAddr

void CTestingAppUi::ConstructL()
  {
  CActive::TPriority priority(CActive::EPriorityUserInput);
  iFinder = CBTDeviceFinder::NewL(priority, *this);
  }

CTestingAppUi::~CTestingAppUi()
  {
  delete iFinder;
  }

void CTestingAppUi::SelectDeviceL()
  {
  /* search only within a specific subset of devices
  TBTDeviceSelectionParams selectionFilter;
  TBTDeviceClass deviceFilter(EMajorServiceObjectTransfer, EMajorDevicePhone,
        EMinorDevicePhoneUnclassified |
         EMinorDevicePhoneCellular |
        EMinorDevicePhoneCordless |
         EMinorDevicePhoneSmartPhone |
         EMinorDevicePhoneWiredModem |
        EMinorDevicePhoneCommonISDNAccess );

  selectionFilter.SetDeviceClass(deviceFilter);
  iFinder->SetSelectionParams(selectionFilter);
  */

  iFinder->DiscoverDevicesL();
  }

void CTestingAppUi::BTDeviceSelected()
  {
  //Do something when BT device is selected
  //e.g. get details of the device the user selected
```

 (C) Copyright Nokia 2013. All rights reserved.

```
    //TBTDeviceResponseParams response = iFinder->ResponseParams();
    //TBTDevAddr deviceAddress = response.BDAddr();
    //TBTDeviceName deviceName = response.DeviceName();
    //TBTDeviceClass deviceClass = response.DeviceClass();
    }
```

Postconditions

The Bluetooth device selection dialog is shown to the user through the notifier framework.

 (C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.