Monitoring battery status with CTelephony

CBatteryCheck implementation illustrates how to check and monitor the battery status of S60 3rd Edition devices with CTelephony API.

The implementation is pretty simple, and when using this one only thing to do in calling class is to implement the callback interface and then to construct an instance of the CBatteryCheck.

When constructing CBatteryCheck, the initial battery level is checked with GetBatteryInfo()-function and after this function returns and calls back the RunL, all changes in battery status are monitored by calling NotifyChange for the battery status.

In the callback the aChangeStatus variable has the value of the current percentage of the battery power left, and the aStatus indicates the battery status with the help of enum from CTelephony:

```
enum TBatteryStatus
{
    /**
    The phone software can not determine the phone's current power status.
    */
    EPowerStatusUnknown,

    /**
    The phone is powered by the battery.
    */
    EPoweredByBattery,

    /**
    A battery is connected, but the phone is externally powered.
    */
    EBatteryConnectedButExternallyPowered,

    /**
    No battery is connected.
    */
    ENoBatteryConnected,

    /**
    Power fault.
    */
    EPowerFault
};
```

Link against:

```
LIBRARY etel3rdparty.lib
```

BatteryLevel.cpp

```
#include "BatteryLevel.h"

CBatteryCheck* CBatteryCheck::NewLC(MBatteryObserver& aObserver)
{
    CBatteryCheck* self = new (ELeave) CBatteryCheck(aObserver);
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

CBatteryCheck* CBatteryCheck::NewL(MBatteryObserver& aObserver)
{
    CBatteryCheck* self = CBatteryCheck::NewLC(aObserver);
    CleanupStack::Pop(); // self;
    return self;
}

void CBatteryCheck::ConstructL(void)
{
    iTelephony = CTelephony::NewL();
    GetBatteryInfo();
}

CBatteryCheck::~CBatteryCheck()
{
    Cancel();
    delete iTelephony;
}

CBatteryCheck::CBatteryCheck(MBatteryObserver& aObserver)
: CActive(EPriorityStandard),iObserver(aObserver),iBatteryInfoV1Pckg(iBatteryInfoV1)
{
    CActiveScheduler::Add(this);
}

void CBatteryCheck::GetBatteryInfo()
{
    if(iTelephony && !IsActive())
    {
        iGettingBattery = ETrue;
        iTelephony->GetBatteryInfo(iStatus, iBatteryInfoV1Pckg);
        SetActive();
    }
}

void CBatteryCheck::RunL()
{
    iObserver.BatteryLevelL(iBatteryInfoV1.iChargeLevel,iBatteryInfoV1.iStatus);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    if(iStatus != KErrCancel)
    {
        iTelephony->NotifyChange(iStatus,CTelephony::EBatteryInfoChange,
        iBatteryInfoV1Pckg);
        SetActive();
    }

    iGettingBattery = EFalse;
}

void CBatteryCheck::DoCancel()
{
    if(iGettingBattery)
    iTelephony->CancelAsync(CTelephony::EGetBatteryInfoCancel);
    else
    iTelephony->CancelAsync(CTelephony::EBatteryInfoChangeCancel);
}
```

BatteryLevel.h

```
#include <Etel3rdParty.h>

class MBatteryObserver
{
    public:
        virtual void BatteryLevelL(TUint aChargeStatus, CTelephony::TBatteryStatus aStatus) = 0;
};

class CBatteryCheck : public CActive
{
    public:
        ~CBatteryCheck();
        static CBatteryCheck* NewL(MBatteryObserver& aObserver);
        static CBatteryCheck* NewLC(MBatteryObserver& aObserver);

    private:
        CBatteryCheck(MBatteryObserver& aObserver);
        void ConstructL(void);

    private:
        void GetBatteryInfo();
        void RunL();
        void DoCancel();

    private:
        MBatteryObserver&               iObserver;
        CTelephony*                     iTelephony;
        CTelephony::TBatteryInfoV1      iBatteryInfoV1;
        CTelephony::TBatteryInfoV1Pckg  iBatteryInfoV1Pckg;
        TBool                           iGettingBattery;
};
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.