Archived:Editing a schedule using RScheduler

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 shows how to create, disable, enable, and edit a condition-based schedule using the class `RScheduler`. When schedules are created, they are enabled by default. By disabling a schedule the scheduled tasks are not executed even if they become due. The `RScheduler` methods `EnableSchedule()` and `DisableSchedule()` are used to enable/disable the example schedule. There are separate methods for time-based and condition-based schedules with different parameters for editing schedules. The condition-based `EditSchedule()` is used for editing the schedule in this example.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY schsvr.lib
```

Preconditions

ExampleTaskHandler.exe must be created before this code snippet can be executed. See Creating and registering a task handler with RScheduler.

Resource files

.rss

```
RESOURCE MENU_PANE r_schedulerexample_menu
    {
    items =
        {
        //...
        MENU_ITEM {command = EEditExample;             txt = "EditExample";},
        //...
        MENU_ITEM {command = EAknSoftkeyExit;          txt = "Exit";}
        };
    }
```

.hrh

```
enum TSchedulerExampleIds
    {
   //...
    EEditExample,
    //...
    };
```

Header file

```
#ifndef __SCHEDULEREXAMPLEAPPUI_H__
#define __SCHEDULEREXAMPLEAPPUI_H__

#include <csch_cli.h> // RScheduler
#include <schinfo.h>  // TSchedulerItemRef, TTaskInfo...

class CSchedulerExampleAppUi : public CAknAppUi
    {
    //...
public:
    void HandleCommandL(TInt aCommand);
    //...
private:
    void CreateConditionVariableL();

    void SetConditionVariableL(TInt aKeyValue);

    void DeleteConditionVariableL();

    TInt EnableOrDisableSchedule(RScheduler& aScheduler, TInt aScheduleId, TBool aDoEnable);

    TInt CreateInitialConditionScheduleWithTaskL(
                                            TSchedulerItemRef& aSchedulerItem,
                                            RScheduler& aScheduler,
                                            TInt aKeyId, TInt aExpectedValue,
```


(C) Copyright Nokia 2013. All rights reserved.

```
                                        const TTsTime& aDefaultRunTime
                                        );



    //...
private:

    RScheduler iScheduler;
};

#endif // __SCHEDULEREXAMPLEAPPUI_H__
```

Source file

```
#include <e32property.h>

const TUid KExampleUid = TUid::Uid(0xE0060BBB); //UID3 from .mmp file
const TInt KExampleKey = 1;

void CSchedulerExampleAppUi::ConstructL()
    {
    //...
    User::LeaveIfError(iScheduler.Connect());

    _LIT(KExampleTaskHandlerExe, "ExampleTaskHandler.exe");
    TFileName exampleHandler(KExampleTaskHandlerExe);

    User::LeaveIfError(iScheduler.Register(exampleHandler, CActive::EPriorityStandard));
    }

CSchedulerExampleAppUi::~CSchedulerExampleAppUi()
    {

    //...
    iScheduler.Close();

    DeleteConditionVariableL();
    }

void CSchedulerExampleAppUi::HandleCommandL(TInt aCommand)
  {
  TBuf<100> Text1; //first line of dialog text
  TBuf<100> Text2; //second line of dialog text

  switch(aCommand)
    {
    case EEikCmdExit:
    case EAknSoftkeyExit:
      Exit();
      break;
//...
    case EEditExample:
      {
      TInt err(KErrNone);

      CreateConditionVariableL();
      TSchedulerItemRef tempRef;

      TTsTime defaultTime;
      TTime time;
      time.HomeTime();
      time = time.operator+(TTimeIntervalMonths(1));
      defaultTime.SetLocalTime( time );

      err = CreateInitialConditionScheduleWithTaskL(tempRef, iScheduler,
KExampleKey, 1, defaultTime );

      //disable the schedule
      err = EnableOrDisableSchedule(iScheduler, tempRef.iHandle, EFalse);

      //set condition variable to trigger the example task
      SetConditionVariableL(1);

      //NOTE: nothing happens because the schedule is disabled!
      //User::After(1000*1000*2);

      //set condition variable not to trigger the example task
      SetConditionVariableL(2);

      //enable the schedule again
      err = EnableOrDisableSchedule(iScheduler, tempRef.iHandle, ETrue);

      if(err == KErrNone)
        Text1.Append(_L("Schedule is enabled!"));

      CArrayFixFlat<TTaskSchedulerCondition>* conditions =
new (ELeave) CArrayFixFlat<TTaskSchedulerCondition>(1);

      CleanupStack::PushL(conditions);
      TTaskSchedulerCondition newCondition;
      newCondition.iCategory    = KExampleUid;
      newCondition.iKey         = KExampleKey;
      newCondition.iState       = 2;
      newCondition.iType        = TTaskSchedulerCondition::EEquals;
      conditions->AppendL(newCondition);

      //edit condition schedule to trigger the example task when
      //condition variable equals to state 2
      err = iScheduler.EditSchedule(tempRef.iHandle, *conditions, defaultTime);

      if(err == KErrNone)
        Text2.Append(_L("Schedule is edited!"));

      CEikonEnv::Static()->InfoWinL(Text1, Text2);

      //Task Scheduler launches the example task...

      CleanupStack::PopAndDestroy(conditions);
      }
      break;
```


(C) Copyright Nokia 2013. All rights reserved.

```
    default:
      //Panic(ESchedulerExampleUi);
      break;
      }
  }


void CSchedulerExampleAppUi::CreateConditionVariableL()
  {
  TInt ret = RProperty::Define(KExampleUid, KExampleKey, RProperty::EInt);

  if (ret != KErrAlreadyExists)
    {
    User::LeaveIfError(ret);
    }
  }

void CSchedulerExampleAppUi::SetConditionVariableL(TInt aKeyValue)
  {
  TInt ret = RProperty::Set(KExampleUid, KExampleKey, aKeyValue);

  User::LeaveIfError(ret);
  }

void CSchedulerExampleAppUi::DeleteConditionVariableL()
  {
  TInt ret = RProperty::Delete(KExampleUid, KExampleKey);

  if (ret != KErrNotFound)
    {
    User::LeaveIfError(ret);
    }
  }

TInt CSchedulerExampleAppUi::EnableOrDisableSchedule(
                                        RScheduler& aScheduler,
                                        TInt aScheduleId,
                                        TBool aDoEnable
                                        )
  {
  TInt ret(KErrNone);

  if(aDoEnable)
    {
    ret = aScheduler.EnableSchedule(aScheduleId);
    }
  else
    {
    ret = aScheduler.DisableSchedule(aScheduleId);
    }

  return ret;
  }

TInt CSchedulerExampleAppUi::CreateInitialConditionScheduleWithTaskL(
                        TSchedulerItemRef& aSchedulerItem,
                        RScheduler& aScheduler,
                        TInt aKeyId, TInt aTriggerValue,
                        const TTsTime& aDefaultRunTime
                      )
    {
    TInt ret(KErrNone);

    aSchedulerItem.iName = _L("Edit Schedule Example");

    CArrayFixFlat<TTaskSchedulerCondition>* conditions =
new (ELeave) CArrayFixFlat<TTaskSchedulerCondition>(1);
    CleanupStack::PushL(conditions);

    TTaskSchedulerCondition condition;
    condition.iCategory     = KExampleUid;
    condition.iKey          = aKeyId;
    condition.iState        = aTriggerValue;
    condition.iType         = TTaskSchedulerCondition::EEquals;
    conditions->AppendL(condition);

    ret = aScheduler.CreatePersistentSchedule(aSchedulerItem, *conditions, aDefaultRunTime);

    if (ret == KErrNone)
      {
      TTaskInfo taskInfo;
      taskInfo.iTaskId = 0;
      taskInfo.iName = _L("ExampleTask4");
      taskInfo.iPriority = 2;
      taskInfo.iRepeat = 0;
      HBufC* data = _L("ExampleData4").AllocLC();

      // schedule the example task
      ret = aScheduler.ScheduleTask(taskInfo, *data, aSchedulerItem.iHandle);
      CleanupStack::PopAndDestroy(); // data
      }
    else
      {
      //Creating persistent schedule failed, do something...
      }

    CleanupStack::PopAndDestroy(conditions);

    return ret;
    }
```

Postconditions

The condition-based schedule with one example task has been created and the task execution is triggered using the Publish and Subscribe mechanism after the schedule is enabled and edited.

See also

(C) Copyright Nokia 2013. All rights reserved.

- Creating and registering a task handler with RScheduler
- Archived:Creating persistent and transient schedules with RScheduler
- Archived:Creating a condition-based schedule with RScheduler
- Archived:Getting schedule and task info using RScheduler
- Archived:Getting schedule and task count using RScheduler
- Archived:Deleting schedules and tasks using RScheduler

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.