Archived:Getting schedule and task info 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 get information about schedules and scheduled tasks using the class RScheduler. The method `GetScheduleRefsL()` is used to get names and IDs of schedules that are defined. The filter parameter `TScheduleFilter` of this method can be used to get all schedules (EAllSchedules) or only pending (EPendingSchedules) schedules.

The method `GetTaskRefsL()` is used to get names and IDs of scheduled tasks. This method also has filter parameters `TScheduleFilter` and `TTaskFilter`. Task filters can have values to include all tasks (EAllTasks) or only those tasks that the calling client is associated with (EMyTasks).

The method `GetTaskDataSize()` returns the size of a given task's data member and it can be used to create a descriptor for retrieved task data. The data associated with a given task can be retrieved with the `GetTaskInfoL()` method.

The method `GetScheduleL()` is used to get all the data associated with the schedule by a given ID. There are separate methods for time-based and condition-based schedules with different parameters. Both of them are demonstrated 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 and a time-based and a condition-based schedule with example tasks must be scheduled before this code snippet can be executed. See code snippets Creating and registering a task handler with RScheduler, Archived:Creating persistent and transient schedules with RScheduler, and Archived:Creating a condition-based schedule with RScheduler for more information.

Resource files

.rss

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

.hrh

```
enum TSchedulerExampleIds
    {
```


(C) Copyright Nokia 2014. All rights reserved.

```
  //...
    EGetInfoExample,
 //...
    };
```

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:
    TInt GetConditionBasedScheduleTasksL(CArrayFixFlat<TTaskInfo>& aTaskInfoArray,
                RScheduler& aScheduler, TInt aScheduleId, TBool& aScheduleEnabled);

    TInt GetTimeBasedScheduleTasksL(CArrayFixFlat<TTaskInfo>& aTaskInfoArray,
                RScheduler& aScheduler, TInt aScheduleId, TBool& aScheduleEnabled);

    //...
private:

    RScheduler iScheduler;

    TSchedulerItemRef iPersistentScheduleHandle;
    TSchedulerItemRef iConditionScheduleHandle;
    };

#endif // __SCHEDULEREXAMPLEAPPUI_H__
```

Source file

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

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

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

CSchedulerExampleAppUi::~CSchedulerExampleAppUi()
    {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    //...
    iScheduler.Close();
    }

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 EGetInfoExample:
     {
     TInt err(KErrNone);
     TInt schedulerTaskCount = -1;
     TInt conditionTaskCount = -1;
     TInt timeTaskCount = -1;
     TBool timeTasksEnabled=EFalse;
     TBool conditionTasksEnabled=EFalse;

     CArrayFixFlat<TSchedulerItemRef>* taskItems =
new (ELeave) CArrayFixFlat<TSchedulerItemRef>(1);
     CleanupStack::PushL(taskItems);

     //retrieve data about schedulers tasks
     err = iScheduler.GetTaskRefsL(*taskItems, EAllSchedules, EAllTasks);

     schedulerTaskCount = taskItems->Count();
     for (TInt index=0; index < schedulerTaskCount; index++)
       {
       Text1.Zero();
       Text2.Zero();

       TSchedulerItemRef item = taskItems->At(index);
       TInt handle = item.iHandle;
       TName name = item.iName;

       Text1.Append(_L("TaskItem:"));
       Text1.AppendNum(handle);

       Text2.Append(_L("Name:"));
       Text2.Append(name);

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

     CleanupStack::PopAndDestroy(taskItems);

     CArrayFixFlat<TSchedulerItemRef>* schedulerItems =
new (ELeave) CArrayFixFlat<TSchedulerItemRef>(1);
     CleanupStack::PushL(schedulerItems);
```


(C) Copyright Nokia 2014. All rights reserved.

```
     //retrieve data about schedules
     err = iScheduler.GetScheduleRefsL(*schedulerItems, EAllSchedules);

     TInt schedulerItemCount = schedulerItems->Count();
     for (TInt index=0; index < schedulerItemCount; index++)
       {
       Text1.Zero();
       Text2.Zero();

       TSchedulerItemRef item = schedulerItems->At(index);
       TInt handle = item.iHandle;
       TName name = item.iName;

       Text1.Append(_L("SchedulerItem:"));
       Text1.AppendNum(handle);

       Text2.Append(_L("Name:"));
       Text2.Append(name);

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

     CleanupStack::PopAndDestroy(schedulerItems);

     CArrayFixFlat<TTaskInfo>* timeTasks =
new (ELeave) CArrayFixFlat<TTaskInfo>(3);
     CleanupStack::PushL(timeTasks);
     GetTimeBasedScheduleTasksL(*timeTasks, iScheduler,
 iPersistentScheduleHandle.iHandle,timeTasksEnabled);
     timeTaskCount = timeTasks->Count();

     for (TInt index=0; index < timeTaskCount; index++)
       {
       Text1.Zero();
       Text2.Zero();

       TTaskInfo task = timeTasks->At(index);
       TInt id = task.iTaskId;
       TName name = task.iName;
       Text1.Append(_L("TimeTask ID:"));
       Text1.AppendNum(id);

       Text1.Append(_L(" Name:"));
       Text1.Append(name);

       TInt taskSize = 0;
       err = iScheduler.GetTaskDataSize(task.iTaskId, taskSize);
       Text2.Append(_L("Size:"));
       Text2.AppendNum(taskSize);

       HBufC* taskData = HBufC::NewLC(taskSize);
       TPtr dataPtr = taskData->Des();

       TTsTime nextDue;
       TTaskInfo taskInfo;
       TSchedulerItemRef schedulerItem;
```


(C) Copyright Nokia 2014. All rights reserved.

```
        err = iScheduler.GetTaskInfoL(task.iTaskId,
                                      taskInfo,
                                      dataPtr,
                                      schedulerItem,
                                      nextDue);

        Text2.Append(_L(" TaskData:"));
        Text2.Append(dataPtr);

        CleanupStack::PopAndDestroy(taskData);

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

      timeTasks->Reset();
      CleanupStack::PopAndDestroy(timeTasks);

      CArrayFixFlat<TTaskInfo>* conditionTasks =
new (ELeave) CArrayFixFlat<TTaskInfo>(3);
      CleanupStack::PushL(conditionTasks);
      GetConditionBasedScheduleTasksL(*conditionTasks, iScheduler,
 iConditionScheduleHandle.iHandle,conditionTasksEnabled);
      conditionTaskCount = conditionTasks->Count();

      for (TInt index=0; index < conditionTaskCount; index++)
        {
        //TTaskInfo task = timeTasks->At(index);
        //Do something with a task info...
        }

      conditionTasks->Reset();
      CleanupStack::PopAndDestroy(conditionTasks);
      }

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


//get task info array from schedule when type is EConditionSchedule
TInt CSchedulerExampleAppUi::GetConditionBasedScheduleTasksL(
                              CArrayFixFlat<TTaskInfo>& aTaskInfoArray,
                              RScheduler& aScheduler,
                              TInt aScheduleId,
                              TBool& aScheduleEnabled)
  {
  TInt ret(KErrNone);
  TTsTime time;
  TScheduleState2 state;
  TScheduleType type;

  ret = aScheduler.GetScheduleTypeL(aScheduleId, type);

  if(ret != KErrNone)
```


(C) Copyright Nokia 2014. All rights reserved.

```
    {
    return ret;
    }

  //the given schedule id is not the one of a condition-based schedule
  if(type != EConditionSchedule)
       {
       return KErrGeneral;
       }

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

  aTaskInfoArray.Reset();
  ret = aScheduler.GetScheduleL(aScheduleId,
                    state,
                    *conditions,
                    time,
                    aTaskInfoArray);

  aScheduleEnabled = state.Enabled();

  CleanupStack::PopAndDestroy(conditions);

  return ret;
  }

//get task info array from schedule when type is ETimeSchedule
TInt CSchedulerExampleAppUi::GetTimeBasedScheduleTasksL(
                              CArrayFixFlat<TTaskInfo>& aTaskInfoArray,
                              RScheduler& aScheduler,
                              TInt aScheduleId,
                              TBool& aScheduleEnabled)
  {
  TInt ret(KErrNone);
  TTsTime time;
  TScheduleState2 state;
  TScheduleType type;

  ret = aScheduler.GetScheduleTypeL(aScheduleId, type);

  if(ret != KErrNone)
    {
    return ret;
    }

  if(type != ETimeSchedule)
      {
      //the given schedule id is not the one of a time-based schedule
      return KErrGeneral;
      }

  CArrayFixFlat<TScheduleEntryInfo2>* entries =
new (ELeave) CArrayFixFlat<TScheduleEntryInfo2>(1);
  CleanupStack::PushL(entries);
```


(C) Copyright Nokia 2014. All rights reserved.

```
    aTaskInfoArray.Reset();
    ret = aScheduler.GetScheduleL(aScheduleId,
                      state,
                      *entries,
                      aTaskInfoArray,
                      time);

    aScheduleEnabled = state.Enabled();

    CleanupStack::PopAndDestroy(entries);

    return ret;
    }
```

Postconditions

The IDs and names of schedules and scheduled tasks are displayed to the user with info dialogs.

See also

- 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 count using RScheduler
- Archived:Editing a schedule using RScheduler
- Archived:Deleting schedules and tasks using RScheduler

(C) Copyright Nokia 2014. All rights reserved.