Windows® Phone

Hands-On Lab

Adding Reminders to Your Application

CONTENTS

Overview

Some Windows® Phone applications need to schedule a notification for a user; the typical scenarios here are an alarm or a reminder for an upcoming event.

One of the many new features of Windows Phone Codenamed Mango is the ability to schedule a notification. There are two types of scheduled notifications: Alarm and Reminder. An alarm allows you to specify a sound when the notification fires, and a reminder allows you to specify a URI that will be actionable when the user clicks on the Reminder. This means, for example, that if the reminder is for a meeting, the URI can be a deep link to this meeting. When the user clicks the reminder, your application launches and navigates the user to the page the URI specified in the reminder.

This lab shows how to add reminders from your application using the new Windows Phone Codenamed Mango API.

Objectives

This lab provides instructions to help you achieve the following:

- Add reminders to your Windows Phone application

Prerequisites

The following prerequisites are required for you to gain most you can from this hands-on lab:

- Microsoft Visual Studio 2010 or Microsoft Visual C# Express 2010, and the Windows® Phone Developer Tools available at http://go.microsoft.com/?linkid=9772716
- Knowledge regarding application development for the Windows Phone 7

Lab Structure

This lab includes a single exercise. The following tasks are contained in the exercise:

1. Adding support for reminder management
2. Managing reminders through the user interface

Estimated completion time

Completing this lab should take from 30 to 45 minutes.

Exercise

This lab is based on the Tidy application. The Tidy application allows users to manage projects containing tasks where each task has a due date. Users can add a reminder for each task to alert them to its impending due date (or for any other reason). This lab starts with a copy of the application that has no reminder-related capabilities. You can find this version in the lab installation folder under **Sources\Begin**. After performing the lab, you should end up with a version that is functionally identical to the one contained in **Sources\End**.

Task 1 – Adding support for reminder management

While the Windows Phone Codenamed Mango API for working with reminders is not complex, it can still be cumbersome in the Tidy application's context that ties reminders to specific tasks. In this task, we add a set of methods to assist us with managing reminders in the application. These methods will demonstrate how to use the new **Microsoft.Phone.Scheduler.ScheduledActionService** class in order to manage reminders.

1. Open the starter solution file, **Todo.sln**, located at the lab installation folder under **Source\Begin**.

2. Examine the solution and locate the **Todo** project. Expand the project's **Misc** folder and open **Utils.cs**. This file contains several utility classes, including the currently empty **RemindersHelper** static class, which you will work on in the following steps.

3. Add the following method to the **RemindersHelper** class:

   C#

   ```
   public static Reminder GetExistingReminder(Guid taskID)
   {
       Reminder reminder =
            (Reminder)ScheduledActionService.Find(taskID.ToString());
       return reminder;
   }
   ```

   In the Tidy Application, each task has a unique Guid identifier. The Guid identifies the reminder associated with that task. You can retrieve the reminder associated with a specific task by using the **ScheduledActionService**'s **Find** method with a string representation of the task's identifier.

4. Add an additional method that checks whether or not a certain task has a reminder set up:

   C#

   ```
   public static bool HasReminder(Guid taskID)
   {
   ```

```
    if (ScheduledActionService.Find(taskID.ToString()) != null)
        return true;
    else
        return false;
}
```

5. Add a method for removing a task's reminder if it exists:

C#

```
public static void RemoveReminder(Guid taskID)
{
    if (HasReminder(taskID))
        ScheduledActionService.Remove(taskID.ToString());
}
```

Note again, how the **ScheduledActionService** identifies reminders using a unique name.

6. Finally, add the only method still missing, the one that defines a new reminder and associates it with a task:

C#

```
public static Reminder AddReminder(Task task, DateTime date,
bool removeOld = true, string body = null)
{
    Guid taskID = new Guid(task.Id.ToString());
    if (HasReminder(taskID))
    {
        if (removeOld)
            RemoveReminder(taskID);
        else
        return null;
    }

    try
    {
        Reminder reminder = new Reminder(taskID.ToString());
        reminder.Title = task.Title;
        reminder.RecurrenceType =
        Microsoft.Phone.Scheduler.RecurrenceInterval.None;
        reminder.NavigationUri = UIConstants.MakeReminderUri(task);
        reminder.Content = body;
        reminder.BeginTime = date;
        ScheduledActionService.Add(reminder);

        return reminder;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message);
        return null;
    }
}
```

Before creating the new reminder, the above code first tries to remove any existing reminder associated with the task, unless the caller specifies otherwise. The method then defines a new reminder, bearing a title derived from the task's own title. The reminder is set to occur only once, at a time specified by the caller and will contain a text message specified by the caller as well. Once clicked, the reminder will navigate to the task viewing page with a query string argument that indicates the application was accessed from a reminder ("From=Reminder").

Task 2 – Managing reminders through the user interface

In the previous task, you laid the groundwork for managing task reminders. In this task, you will alter the application's interface to allow adding, removing, and updating task reminders.

1. Compile and run the application.
2. Create a new task or click on an existing one.
3. In the task view, you should see a toggle button for adding or removing a task's reminder. The button has no effect currently:

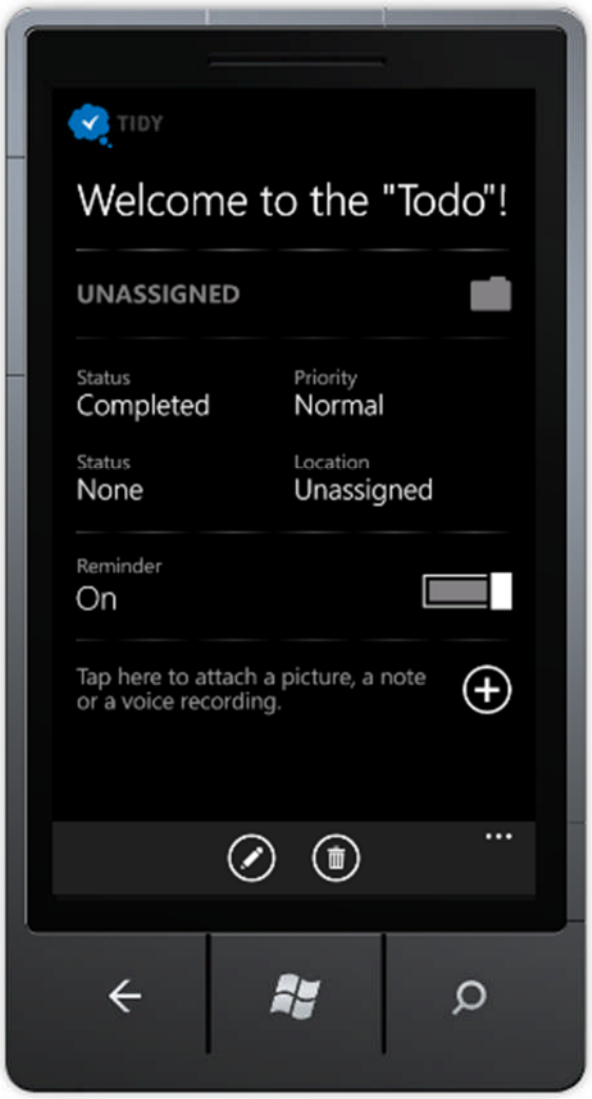


Figure 1
The task view with non functional reminder UI

The reminder managing UI (most of which is currently hidden) is already implemented and all that is left is to wire it up properly. We do this by altering the task editing ViewModel.

4. Examine the **Todo** project.
5. Expand the project's **ViewModels** folder and open **TaskEditViewModel.cs**.
   This file implements the ViewModel used by the task editing view. We will fill out some placeholders contained in the file in order allow managing reminders.
6. Navigate to the **HasSystemReminder** in the **TaskEditViewModel** class and change it to the following:

C#
```
public bool HasSystemReminder
```

```
{
    get
    {
        return RemindersHelpers.HasReminder(task.Id);
    }
}
```

The above getter simply uses the **RemindersHelper** we implemented to return whether or not the task viewed has a reminder.

7. Find the **IsReminderSettingsEnabled** property and change its setter to contain the following code:

C#

```
if (isReminderSettingEnabled != value)
{
    isReminderSettingEnabled = value;
    base.OnPropertyChanged("IsReminderSettingEnabled");
    if (!isReminderSettingEnabled)
    {
        RemindersHelpers.RemoveReminder(task.Id);
        ReminderDate = null;
        ReminderTime = null;
        ReminderContent = string.Empty;
        OnPropertyChanged("HasSystemReminder");
    }
}
```

This setter cleans the current task's reminder if the toggle switch is set to off.

8. Find the **InitializeReminder** method and change it using the following snippet:

C#

```
void InitializeReminder()
{
    //Cache the reminder to prevent too many lookups since a lot of fields
    //need it
    if (reminder == null && notSearchedForReminder)
    {
        reminder = RemindersHelpers.GetExistingReminder(task.Id);
        notSearchedForReminder = false;
    }

    if (reminder != null)
    {
        reminderContent = reminder.Content;
        reminderTime = reminder.BeginTime;
        reminderDate = reminder.BeginTime;
        IsReminderSettingEnabled = true;
    }
}
```

The above method gets the task's existing reminder and initializes some fields accordingly. This allows the page to display an existing reminder's data.

9. The UI will allow the user to set a reminder's text, date and time. We need a method to take the information provided by the user and use it to create a reminder for the current task, or update its existing reminder. Change the **CheckAndSaveReminder** method using the following code:

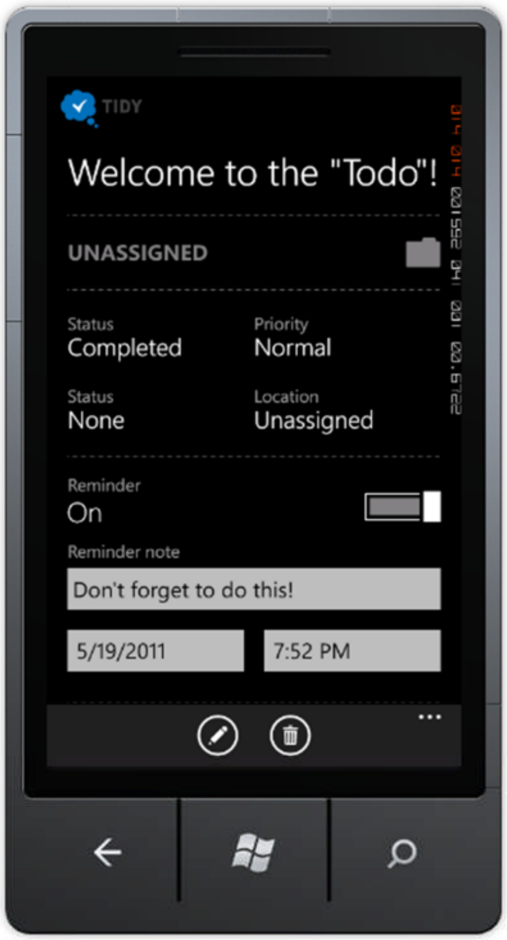
C#

```
void CheckAndSaveReminder()
{
    if ( HasValidReminder)
    {
        DateTime fullDate =
            new DateTime ( ReminderDate.Value.Year,
                           ReminderDate.Value.Month,
                           ReminderDate.Value.Day,
                           ReminderTime.Value.Hour,
                           ReminderTime.Value.Minute,
                               ReminderTime.Value.Second );


        if (DateTime.Now.AddMinutes(3) > fullDate)
        {
            MessageBox.Show(ApplicationStrings.ReminderNotFarEnough);
            return;
        }

        RemindersHelpers.AddReminder(task, fullDate, true, ReminderContent);
        OnPropertyChanged("HasValidReminder");
        OnPropertyChanged("HasSystemReminder");
    }
}
```

10. Compile and launch the application. You should now be able to add reminders to your tasks, and then access the task by clicking its reminder once it pops up:



Welcome to the "Todo"!

UNASSIGNED

Status
Completed

Priority
Normal

Status
None

Location
Unassigned

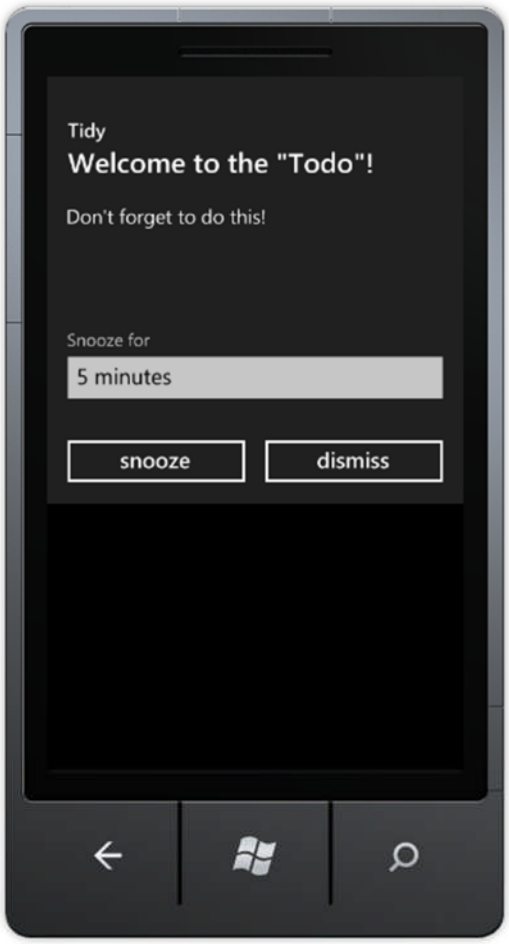
Reminder
On

Reminder note
Don't forget to do this!

5/19/2011

7:52 PM

Figure 2
Adding a reminder to a task


Tidy

Welcome to the "Todo"!

Don't forget to do this!

Figure 3

The reminder after popping up

11. This completes the exercise and the lab.

Summary

This lab has shown you how to use the new Windows Phone Codenamed Mango APIs to add reminders from your code and how reminders can lead to a specific page in your application when the user interacts with them.