Creating an MDI Parent Form with a Menu Bar

In this exercise, you will create an MDI form in the WinApp project. You will also see how to create a menu bar for the parent form, that will allow you to navigate to all the child forms. To do so, follow these steps:

1. Navigate to Solution Explorer, select the WinApp project, right-click, and select "Add" -> "Windows form". Change the Name value from "Form1.cs" to "ParentForm.cs", and click "Add".
2. Select the newly added ParentForm in the Design View. Select the ParentForm form by clicking the form's title bar, navigate to the Properties window, and set the following properties:
    - Set the "IsMdiContainer" property to True (the default value is False). Notice that the background color of the form has changed to dark gray.
    - Set the Size property's Width to 546 and Height to 411.
3. Drag a MenuStrip control to the ParentForm. In the top-left corner, you should now see a drop-down showing the text "Type Here". Enter the text "Open Forms" in the drop-down. This will be your main, top-level menu.
4. Now under the Open Forms menu, add a submenu by entering the text "Win App".
5. Under the Win App submenu, enter "User Info".
6. Now click the top menu, "Open Forms", and on the right side of it, type "Help". Under the Help menu, enter "Exit".
7. Now, click the top menu, on the right side of Help, type "Windows".
8. Under the Windows menu, add the following options as separate submenus: Cascade, Tile Horizontal, Tile Vertical, and Arrange Icons. These will help in arranging the child forms.
9. Now it's time to attach code to the submenus you have added under the main menu Open Forms. First, you'll add code for the submenu Win App, that basically will open the WinApp form. In the Design View, double-click the "Win App" submenu, that will take you to the Code View. Under the click event, add the following code:

    ```
    WinApp objWA = new WinApp();
    objWA.Show();
    ```

10. Now to associate functionality with the User Info submenu: double-click this submenu, and under the click event add the following code:

```
UserInfo objUI = new UserInfo();
objUI.Show();
```

11. To associate functionality with the Exit submenu located under the Help main menu, double-click "Exit", and under the click event add the following code:

```
Application.Exit();
```

12. Now you have the form-opening code functionality in place, and you are nearly set to run the application. But first, you need to set the ParentForm as the start-up object. To do so, open Program.cs, and modify the "Application.Run(new UserInfo());" statement to the following:

```
Application.Run(new ParentForm());
```

13. Now build the solution, and run the application by pressing F5; the MDI application will open and should look as in Figure 1-1.

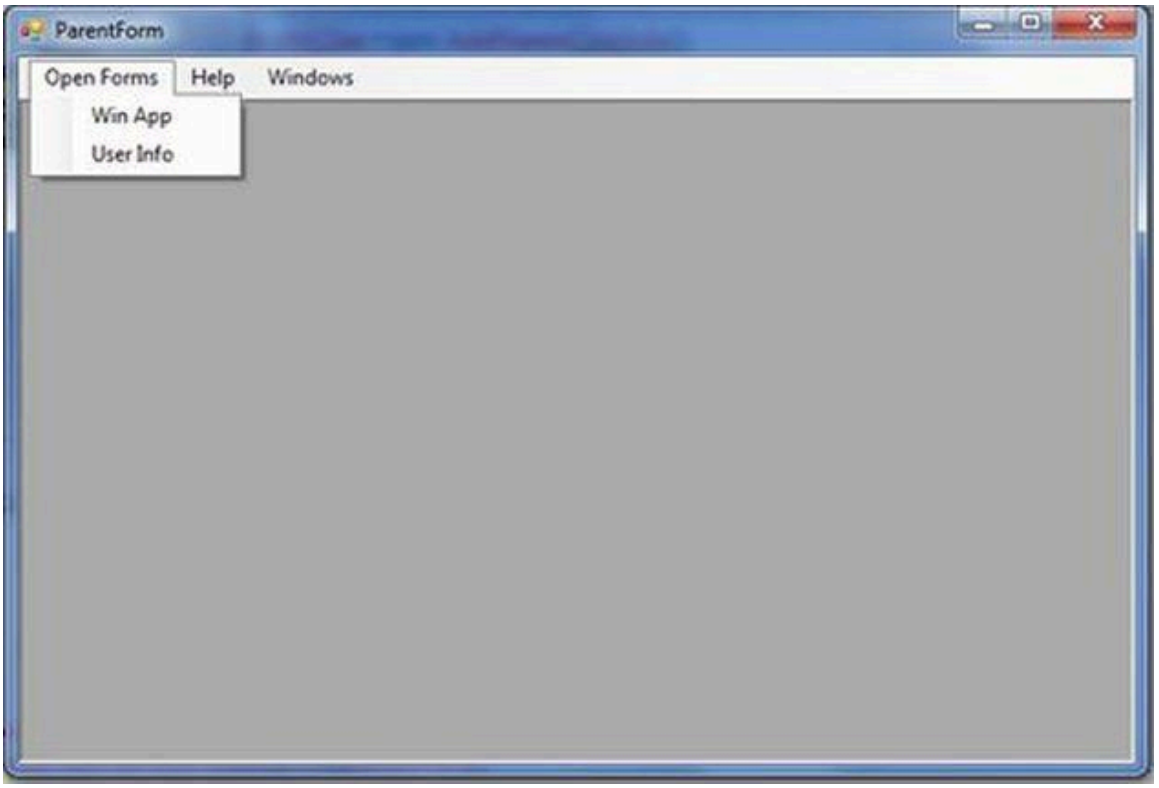

Figure 1-1. Running an MDI form application

14. Now if you click "Win App" and then "User Info" then both the forms will open one by one. These forms can be opened and dragged outside of the MDI form. This is not an expected behavior from a MDI application, as shown in Figure 1-2.

    This issue will be addressed later in this chapter.

Figure 1-2. Running an MDI form application

Opening an MDI Child Form Within an MDI Application

As mentioned and shown in Figure 1-2 above, the problem is that even though the MDI form shows a parent menu, the forms are still able to open outside, and context is moved from a form to another. You can try clicking the title bar of each the open form, and you will see how you can move back and forth with these opened forms.

In this exercise, you will overcome this problem and associate all the forms you created earlier as MDI child forms to the main MDI parent form you created in the previous task.

1. In the project you modified in the previous exercise, you'll first make the WinApp form an MDI child form. To do so, you need to set the MdiParent property of the child form's object to the MDI parent form

itself, but in the Code View. You have already added functionality in the previous task (opening the WinApp form); just before the line where you are calling the Show() method, add the following code (this code can be found under Win App menu click event):

objWA.MdiParent=this;
After adding this line, the code will appear as follows:

```
WinApp objWA = new WinApp();
objWA.MdiParent = this;
objWA.Show();
```

Note: this is a C# language keyword that represents the current instance of the class. In this case, it refers to ParentForm. Because you are writing this code inside ParentForm, you can use this keyword for the same.

Now you will make the UserInfo form an MDI child form. To do so, you need to set the MdiParent property to the name of the MDI parent form but in the Code View. Add the following code as you did in the previous step (this code can be found under the User Info menu click event):

```
objUI.MdiParent=this;
```

After adding this line, the code will appear as follows:

```
UserInfo objUI = new UserInfo();
objUI.MdiParent=this;
objUI.Show();
```

Now build the solution, and run the application by pressing F5; the MDI application will open and should appear as shown in Figure 1-3.

Click "Open Form" -> "Win App"; the WinApp form should open. Again, open the main menu and click "User Info". Both the forms should now be open inside your main MDI parent form application, and unlike before, you will not be able to drag these out of your MDI parent form (as shown in Figure 1-2). Figure 1-3 shows the expected behavior of an MDI application with opened form(s).

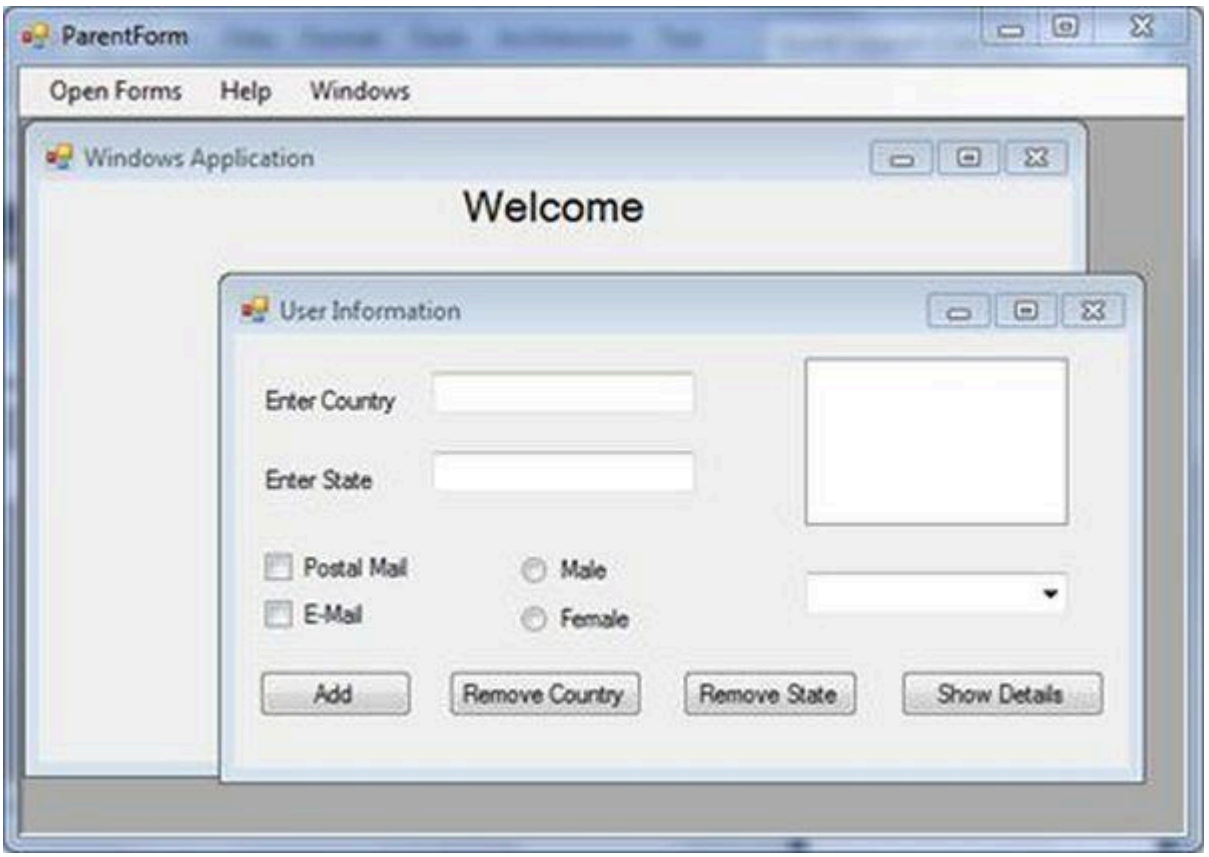

Figure 1-3. Opening child forms inside an MDI form application:

Because both the forms are open inside one MDI parent, it becomes easier to work with them, and they are not draggable outside of the MDI parent. Switch among these forms by clicking their title bars. Once you are done with the forms, close the application by selecting "Help" -> "Exit".