called *Tasks*. When an application requires information from the phone, such as the phone number of a contact or a photo from the camera, it needs to launch an appropriate Task.

Windows Phone Tasks are all structured essentially the same way. However, they can be placed into two groups referred to as *launchers* and *choosers*. As you can imagine, a *launcher* is a Task that launches another application on the phone. It might be to send an e-mail, send a text message, or display a web page within the browser. Essentially, a launcher doesn't return any data to the application. In fact, when an application triggers a launcher, it should be aware that the user may not return to the application. A *chooser*, on the other hand, is a Task that the application can expect to return with a piece of information — for example, requesting that the user take a photo or select a phone number. Table 8-1 lists all the Windows Phone Tasks, including the type of any data returned.

TABLE 8-1: Windows Phone Tasks

CHOOSERS	DESCRIPTION	RETURN TYPE
`CameraCaptureTask`	Opens camera application to take a photo.	`PhotoResult`
`PhotoChooserTask`	Selects an image from your Picture Gallery.	`PhotoResult`
`EmailAddressChooserTask`	Selects an e-mail address from your Contacts List.	`EmailResult`
`PhoneNumberChooserTask`	Selects a phone number from your Contacts List.	`PhoneNumberResult`
`SaveEmailAddressTask`	Saves an e-mail address to an existing or new contact.	
`SavePhoneNumberTask`	Saves a phone number to an existing or new contact.	

LAUNCHERS	DESCRIPTION
`EmailComposeTask`	Composes a new e-mail.
`PhoneCallTask`	Initiates a phone call to a specified number.
`SmsComposeTask`	Composes a new text message.
`SearchTask`	Launches Bing Search with a specified search term.
`WebBrowserTask`	Launches Internet Explorer browsing to a specific URL.
`MarketplaceDetailTask`	Launches Marketplace with the details of a specific application.
`MarketplaceHubTask`	Launches Marketplace at one of the three hubs: Applications, Music or Podcasts.
`MarketplaceReviewTask`	Launches Marketplace to provide a review of the current application.
`MarketplaceSearchTask`	Launches Marketplace and performs a search for content.
`MediaPlayerLauncher`	Launches Media Player.

The general pattern for invoking a Task is to create an instance of the Task, set any necessary properties, and then call its `Show` method. For example, you can invoke the Task to request the user to select an e-mail address using the following two lines:

```
EmailAddressChooserTask addressTask = new EmailAddressChooserTask();
this.addressTask.Completed += addressTask_Completed;
addressTask.Show();

void addressTask_Completed(object sender, EmailResult e){...}
```

In the case of choosers, you will also need to attach an event handler to the `Completed` event. When the chooser application closes, the `Completed` event is invoked, and any return value can be accessed from the event arguments. As mentioned earlier, there is no information returned from a launcher so there is no method to indicate that the launcher has completed.

Where Did My Application Go?

Before going through each of the different Tasks it's important to understand how Windows Phone applications behave when they are placed into the background. This was covered in Chapter 6 in the context of the navigation system but is equally applicable when working with Tasks. If you recall, when your application goes into the background, the Deactivated event is raised and then the application is marked as "Eligible for Termination." At this point it is highly likely that your application will be terminated. This is true even if your application invokes a chooser task that is to return data.

The following code creates an instance of the EmailAddressChooserTask within the EmailAddressButton_Click method. When this method is invoked, the EmailAddressChooserTask will be create, the event handler wired up, and the appropriate chooser displayed. The last part of this process takes the focus away from the application, putting it into the background and making it "Eligible for Termination."

```
private void EmailAddressButton_Click(object sender, RoutedEventArgs e){
    EmailAddressChooserTask addressTask = new EmailAddressChooserTask();
    addressTask.Completed += addressTask_Completed;
    addressTask.Show();
}
```

So what happens if the application does get terminated when the chooser is displayed? More importantly, what happens when the user has selected the contact's e-mail address that they want returned to the application. As you saw in Chapter 6 the application is restarted and the page that the application was on is navigated to. This is where you will run into issues with defining the chooser task inside a method scope (as in the previous code snippet). Because the instance is only created, and the event handler wired up, within the scope of a method, there is no way for the system to know to invoke the `addressTask_Completed` method with the results of the chooser task.

The correct way to work with chooser tasks is to create the chooser as an instance level variable. In the following code the `EmailAddressChooserTask` is instantiated during the construction of the `MainPage` and the event handler for the `Completed` event is wired up at the end of the constru