

Windows
Phone

Launchers and
Choosers

Interacting with and Extending Phone
Services

Launchers and Choosers

- Windows Phone execution model isolates every application in its own sandbox
 - Apps cannot directly access information stores such as contacts
 - Cannot directly invoke other applications such as phone or messaging
- Launchers and Choosers allow applications indirect access to these useful phone features
- Launcher and Chooser APIs invoke distinct built-in applications that **replace** the currently running application

Windows Phone Jump Start

What are Launchers and Choosers?

- Launcher
 - Launches one of the built-in applications through which a user completes a task
 - No data is returned to calling application
 - Example: PhoneCallTask
- Chooser
 - Launches one of the built-in applications through which a user completes a task, and which returns some data to calling application
 - When caller completes, calling application is activated and supplied with the Chooser result
 - Example: PhotoChooserTask

3Windows Phone Jump Start

Launchers & Choosers

Launchers

- EmailComposeTask
- MarketplaceDetailTask
- MarketplaceHubTask
- MarketplaceReviewTask
- MarketplaceSearchTask
- MediaPlayerLauncher
- PhoneCallTask
- SearchTask
- SMSComposeTask
- WebBrowserTask

Choosers

- CameraCaptureTask
- EmailAddressChooserTask
- PhoneNumberChooserTask
- PhotoChooserTask
- SaveEmailAddressTask
- SavePhoneNumberTask

Programming Launchers

- ```
1. // Launches the Email application: displays a new email message
2. EmailComposeTask emailComposeTask = new EmailComposeTask();
3. emailComposeTask.To = "user@example.com";
4. emailComposeTask.Body = "Email message body";
5. emailComposeTask.Cc = "user2@example.com";
6. emailComposeTask.Subject = "Email subject";
7. emailComposeTask.Show();

1. // Launches the Windows Phone Marketplace client application
// which then shows the search results based on search terms
2. MarketplaceSearchTask marketplaceSearchTask = new
MarketplaceSearchTask();
3. marketplaceSearchTask.SearchTerms = "accelerometer xna";
4. marketplaceSearchTask.Show();
```

5 Windows Phone Jump Start

---

---

---

---

---

---

---

## Choosers and the Execution Model

- Important! When you launch a Launcher or Chooser, your app is terminated
  - When task completes, your app may be reactivated and a new app instance created
- To use a chooser
  - Chooser object must be declared with class scope within the [PhoneApplicationPage](#)
  - In the [PhoneApplicationPage](#) constructor, create an instance of the Chooser class you wish to use
  - Also in the [PhoneApplicationPage](#) constructor, assign a delegate for the Chooser's [Completed](#) event
  - Implement the event handler for the [Completed](#) event
  - Call the Chooser's Show method to invoke the Chooser

6 Windows Phone Jump Start

---

---

---

---

---

---

---

# Programming Choosers

```
1. public partial class MainPage : PhoneApplicationPage
2. {
3. PhotoChooserTask photoChooserTask; // Declare the PhotoChooserTask object with page scope.
4.
5. public MainPage()
6. {
7. InitializeComponent();
8. // Initialize the PhotoChooserTask and assign the Completed handler
9. photoChooserTask = new PhotoChooserTask();
10. photoChooserTask.Completed += new
11. EventHandler<PhotoResult>(photoChooserTask_Completed);
12. }
13.
14. private void button1_Click(object sender, RoutedEventArgs e)
15. {
16. photoChooserTask.Show();
17. }
18.
19. void photoChooserTask_Completed(object sender, PhotoResult e)
20. {
21. if (e.TaskResult == TaskResult.OK)
22. {
23. BitmapImage bmp = new BitmapImage();
24. bmp.SetSource(e.ChosenPhoto);
25. myImage.Source = bmp;
26. }
27. }
28. }
```

Windows Phone Jump Start

---

---

---

---

---

---

---

---

# Using the Camera

```
1. CameraCaptureTask cameraCaptureTask; // Declare the CameraCaptureTask object with page scope.
2.
3. public MainPage()
4. {
5. InitializeComponent();
6. // Initialize the CameraCaptureTask and assign the Completed handler
7. cameraCaptureTask = new CameraCaptureTask();
8. cameraCaptureTask.Completed += new EventHandler<PhotoResult>(cameraCaptureTask_Completed);
9. }
10.
11. private void button1_Click(object sender, RoutedEventArgs e)
12. {
13. cameraCaptureTask.Show();
14. }
15.
16. void cameraCaptureTask_Completed(object sender, PhotoResult e)
17. {
18. if (e.TaskResult == TaskResult.OK) {
19. BitmapImage bmp = new BitmapImage();
20. bmp.SetSource(e.ChosenPhoto);
21. myImage.Source = bmp;
22. }
23. }
```

Windows Phone Jump Start

---

---

---

---

---

---

---

---

# Launchers and Choosers

# Demo

---

---

---

---

---

---

---

---

## Lab 2 – Complete After Session

### Launchers and Choosers

- <http://channel9.msdn.com/learn/courses/WP7TrainingKit/WP7Silverlight/LaunchersAndChoosersWP7Lab/>
- Complete both exercises

Post your questions in the BTL Forum – if you have any questions or need support while doing your labs

---

---

---

---

---

---

---

## Hub Extensibility

- Windows Phone aims to offer an integrated experience to users
  - Apps to do with music should be accessible from the Music hub
  - Apps to do with photos should be accessible from the Photo hub
- Photo extras application extensibility allows developers to integrate their photo altering applications seamlessly into the Windows Phone built-in photo application
- Music + Videos hub can be extended with custom applications that meet qualification requirements
  - Use the MediaHistory and MediaHistoryItem classes
  - Pass through Marketplace certification and have received required security capabilities

11

Windows Phone Jump Start

---

---

---

---

---

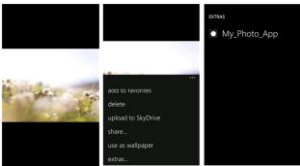
---

---

## Creating a Photo Extras App

- Include file Extras.xml in your solution

```
<Extras>
 <PhotosExtrasApplication>
 <Enabled>true</Enabled>
 <StorageFolder>Incoming</StorageFolder>
 </PhotosExtrasApplication>
</Extras>
```



12

Windows Phone Jump Start

---

---

---

---

---

---

---

## Receiving Photo

```
1. protected override void OnNavigatedTo(NavigationEventArgs e)
2. {
3. IDictionary<string, string> queryStrings = this.NavigationContext.QueryString;
4. string filename = string.Empty;
5.
6. if (queryStrings.Count > 0 && queryStrings.ContainsKey("file"))
7. {
8. filename = this.NavigationContext.QueryString["file"]; // Get filename of photo
9. // append the folder name (incoming) to the file path
10. filename = System.IO.Path.Combine("Incoming", filename);
11. //The following opens the file in the isolated storage folder.
12. IsolatedStorageFile isoStore = IsolatedStorageFile.GetUserStoreForApplication();
13. IsolatedStorageFileStream isoFileStream =
14. isoStore.OpenFile(filename, FileMode.Open, FileAccess.Read);
15.
16. WriteableBitmap picLibraryImage = PictureDecoder.DecodeJpeg(isoFileStream);
17. retrievePic.Source = picLibraryImage;
18. }
19. }
```

13

Windows Phone Jump Start

---

---

---

---

---

---

---

---



Windows®  
Phone

© 2010 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

---

---

---

---

---

---

---

---