Windows Phone - Integrare un’applicazione nell’hub Pictures

#### di [Matteo Pagani](https://mvp.support.microsoft.com/profile=19D4248B-0204-4B23-AC27-62CD4E60A569)

* 1. 

*Marzo 2012*

Se siete pratici del mondo di Windows Phone, saprete sicuramente che le applicazioni girano isolate l’una dall’altra: non è possibile accedere dalla nostra App ai dati memorizzati da un’altra applicazione. Per questo motivo, sono stati creati i launcher e i chooser, ovvero un modo per interagire con le applicazioni native (People, Pictures, Camera, ecc.) del sistema operativo. Se volete approfondire il discorso, ho pubblicato un po’ di tempo fa sul mio blog personale una serie di post sull’argomento.

A volte però si vorrebbe offrire ai nostri utenti un grado di interazione maggiore: mettiamo caso di aver sviluppato un’applicazione in grado di elaborare le immagini. L’utente sta sfogliando la sua libreria di foto tramite l’applicazione Pictures, quando ne trova una che vorrebbe elaborare con la nostra applicazione: in condizione normali, sarebbe costretto a tornare al menu principale di Windows Phone, lanciare la nostra applicazione e da lì, tramite le funzioni disponibili, andare a selezionare la foto da importare tramite il chooser dedicato.

Come potete immaginare, questo scenario pecca un po’ di usabilità: sarebbe bello se l’utente potesse selezionare la foto da elaborare con la nostra applicazione direttamente dalla sezione Pictures.

E chi dice che non possiamo farlo?

## L’hub delle foto

In Mango abbiamo la possibilità di agganciare la nostra applicazione in due punti dell’hub delle foto:

* + Integrarla nel panorama: se sul vostro device avete installato almeno un’applicazione in grado di elaborare immagini (o meglio, che è stata dichiarata come tale dallo sviluppatore) nell’hub delle foto ci sarà un item nel panorama chiamato **Apps**, nel quale verranno elencate.
  + Integrarla nell’application bar: nella versione precedente di Windows Phone, nel momento in cui selezionavate una foto era a disposizione nel menu contestuale nell’application bar una voce chiamata **Extras**, che mostrava una serie di applicazioni che erano in grado di interagire con la vostra applicazione. Questa feature è rimasta, solo che la voce di menu è stata rinominata in **Apps**.



Come fare per supportare entrambi gli scenari? Le operazioni da effettuare sono due:

* + Istruire la nostra applicazione affinché risulti disponibile tra quelle visualizzate nell’hub delle foto.
  + Gestire l’eventualità che questa venga aperta in seguito all’utilizzo del menu Apps.

## Integrarsi nell’hub delle foto

Rendere disponibile la nostra applicazione tra quelle elencate nell’hub delle foto è molto semplice: è sufficiente aggiungere una dichiarazione nel file di manifest.

Il file di manifest è un file XML, chiamato **WMAppManifest.xml** e rintracciabile sotto la cartella **Properties** del progetto, che definisce una serie di informazioni indispensabili al sistema operativo per capire cosa fa la nostra applicazione: vengono dichiarate ad esempio le capabilities (ovvero le funzionalità del telefono utilizzate), i background agent e così via.

Nel file di manifest esiste una sezione apposta per l’estensibilità della nostra applicazione, chiamata **Extensions**. Di default, non esiste: dobbiamo perciò aggiungerla a mano in un punto qualsiasi, all’interno del nodo **App**.

Ecco la dichiarazione da inserire per rendere disponibile la nostra applicazione tra quelle visualizzate nell’hub Pictures:

* 1. XML
  2. <Extensions>
  3. <Extension ExtensionName="Photos\_Extra\_Hub" ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5632}" TaskID="\_default" />
  4. </Extensions>

Il gioco è fatto! Non ci sono altre operazioni da fare, dato che ci stiamo limitando a creare un nuovo punto di ingresso da cui l’utente può lanciare l’applicazione. Diverso è il discorso se vogliamo agganciarci al menu contestuale Apps, come vedremo tra poco.

## Agganciarsi al menu Apps

Dato che anche agganciarsi al menu Apps significa estendere la nostra applicazione, la modalità è la stessa che abbiamo visto per l’integrazione con l’hub delle foto: dobbiamo inserire una dichiarazione nel file di manifest, nella sezione **Extensions**.

* 1. XML
  2. <Extensions>
  3. <Extension ExtensionName="Photos\_Extra\_Viewer" ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5632}" TaskID="\_default" />
  4. </Extensions>

La differenza è però che questa volta non è finita qui: inserendo questa dichiarazione ci siamo, infatti, limitati ad inserire la nostra applicazione tra quelle disponibili nel menu contestuale Apps. Dobbiamo però gestire il fatto che l’utente possa selezionare una foto e dopodiché scegliere la nostra applicazione dal menu contestuale.

## Recuperare l’immagine selezionata dall’hub delle foto

Per recuperare l’immagine scelta dall’utente e importarla nella nostra applicazione dobbiamo posizionarci nella pagina principale (quella che, per intenderci, viene mostrata all’avvio, tipicamente **MainPage.xaml**) e intercettare l’evento **OnNavigatedTo**. Questo evento, legato al sistema di navigazione di Silverlight / Windows Phone, viene invocato nel momento in cui l’utente si è spostato verso la pagina.

Se conosciamo il funzionamento della navigazione nelle applicazioni Silverlight o Windows Phone, sapremo che non è molto diversa dalla navigazione tra pagine web: quando invochiamo il metodo **Navigate** esposto dal **NavigationService** dobbiamo specificare un oggetto di tipo **Uri** che contiene il percorso della pagina a cui vogliamo portare l’utente (ad esempio, **/Views/View.xaml**). In più, esattamente come nelle pagine web, possiamo passare delle informazioni in query string (ad esempio, **/Views/Views.xaml?ID=5**).

Il menu Apps utilizza proprio questa tecnica per passarci l’informazione relativa all’immagine scelta: nel caso in cui l’applicazione sia stata aperta da tale menu, l’URL della pagina conterrà un parametro in query string chiamato **token** con una stringa univoca che identifica la nostra immagine.

Ecco come possiamo fare, all’avvio della nostra applicazione, a determinare se questa sia stata aperta in maniera tradizionale e, in caso contrario, a salvarci il valore del parametro token.

* 1. C#
  2. protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
  3. {
  4. string token = string.Empty;
  5. NavigationContext.QueryString.TryGetValue("token", out token);
  6. if (!string.IsNullOrEmpty(token))
  7. {
  8. MessageBox.Show(token);
  9. }
  10. else
  11. {
  12. MessageBox.Show("Token not found");
  13. }
  14. }

L’oggetto **NavigationContext** espone diverse informazioni sul contesto attuale della navigazione, tra cui anche tutte le query string che sono state passate nell’URL.

Ecco perciò che utilizziamo il metodo **TryGetValue** per cercare di recuperare il valore del parametro **token**: se questo esiste, la variabile token sarà valorizzata con tale valore, altrimenti sarà vuota.

Quello che facciamo dopo è molto semplice: mostrare a video un messaggio con il token nel caso sia stato trovato o un messaggio di avviso nel caso il parametro non ci sia.

## Come testarlo?

Per testare questa funzionalità abbiamo bisogno di un device reale: l’emulatore infatti è basato su una immagine minimale del sistema operativo, perciò non contiene l’applicazione Pictures che ci servirebbe per testare la funzionalità.

Una volta collegato il nostro device, premiamo **F5** e lanciamo l’applicazione: al primo avvio comparirà il messaggio **Token not found**, dato che l’abbiamo aperta in modo tradizionale.

Ora chiudiamo l’applicazioni, andiamo nell’hub Pictures, selezioniamo una foto qualsiasi e poi dalla application bar selezioniamo la voce Apps: se abbiamo fatto tutto correttamente, nell’elenco comparirà anche la nostra applicazione. Selezioniamola e a questo punto questa si aprirà, mostrando a video il token.

#### Rendiamo le cose più interessanti

Abbiamo appena dimostrato che l’integrazione funziona: mostrare un messaggio a video non è però molto utile. Vediamo perciò di fare qualcosa di più interessante, come mostrare l’immagine selezionata all’interno della nostra applicazione.

Innanzitutto dobbiamo inserire nella nostra pagina un controllo di tipo **Image**, nel quale visualizzare l’immagine:

* 1. XAML
  2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  3. <Image x:Name="ExampleImage" Width="300" HorizontalAlignment="Center" VerticalAlignment="Center">
  4. </Image>
  5. </Grid>

Come facciamo dal token a recuperare lo stream dell’immagine vera e propria, così da poterlo utilizzare ed elaborare a nostro piacimento? Ci viene in nostro aiuto la classe **MediaLibrary**, che non fa parte di Silverlight ma di XNA. Per poterla utilizzare, infatti, dobbiamo aggiungere una reference alla libreria **Microsoft.XNA.Framework**.

* 1. C#
  2. protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
  3. {
  4. string token = string.Empty;
  5. NavigationContext.QueryString.TryGetValue("token", out token);
  6. if (!string.IsNullOrEmpty(token))
  7. {
  8. MediaLibrary library=new MediaLibrary();
  9. Picture picture = library.GetPictureFromToken(token);
  10. BitmapImage image = new BitmapImage();
  11. image.SetSource(picture.GetImage());
  12. ExampleImage.Source = image;
  13. }
  14. base.OnNavigatedTo(e);
  15. }

Una volta istanziata la classe **MediaLibrary**, abbiamo a disposizione il metodo **GetPictureFromToken** che serve proprio per gestire casi come questi: dato un token identificativo di una immagine ci restituisce un oggetto di tipo Picture, il quale contiene tutte le informazioni sulla nostra immagine (dimensioni, nome del file, data in cui è stata scattata, ecc.). In più, espone un metodo chiamato **GetImage** che restituisce l’immagine vera e propria sotto forma di oggetto di tipo **Stream**.

Il gioco è quasi fatto: ora ci basta creare un nuovo oggetto di tipo **BitmapImage**, utilizzarne il metodo **SetSource** (che inizializza l’immagine proprio partendo da un oggetto di tipo **Stream**), dopodichè assegnare l’immagine alla proprietà Source del nostro controllo di tipo **Image**.

Ora premiamo nuovamente F5, rifacciamo il deploy dell’applicazione sul nostro device e ripetiamo l’esperimento di prima: se tutto è andato a buon fine, l’immagine scelta comparirà al centro della nostra pagina.

## In conclusione

Come avete potuto vedere, è piuttosto semplice integrare le nostre applicazioni fotografiche con l’hub delle foto, così da garantire maggiore usabilità e semplicità d’uso.

L’articolo è corredato da un progetto di esempio in cui trovate implementato concretamente quello di cui abbiamo parlato: in più, l’applicazione finale implementa anche il chooser **PhotoChooserTask**, che permette all’utente di andare a selezionare una foto direttamente nell’applicazione stessa. In questo modo, abbiamo supportato entrambi gli scenari possibili che consentono all’utente di importare una foto nella nostra app.

#### di Matteo Pagani ([blog](http://www.qmatteoq.com/)) – Microsoft MVP

* 1. [*Altri articoli di Matteo Pagani nella Libr*](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Matteo%20Pagani)*ary* [](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Sergio%20Govoni)