Windows Phone 7.5 - i nuovi launcher e chooser

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

* 1. 

*Marzo 2012*

[La versione 7.1 dell’SDK di Windows Phone](http://go.microsoft.com/?linkid=9772716) (quella che consente di sviluppare applicazioni per Mango, ormai conosciuto con il nome ufficiale di Windows Phone 7.5) non ha introdotto solamente nuove API e nuove feature (come il Fast Application Switching o i background agents), ma ha anche espanso le funzionalità già esistenti.

E’ il caso dei launcher e chooser: oltre a quelli già presenti nella versione 7.0, ne abbiamo infatti di nuovi, che aprono nuovi scenari per le nostre applicazioni.

Ma facciamo un passo indietro e vediamo brevemente cosa sono i launcher e i chooser: le applicazioni Windows Phone girano in una sorta di sandbox, ovvero isolate l’una dall’altra. Questo significa che non hanno modo di accedere l’una ai dati dell’altra: questo vale ovviamente anche per le applicazioni native.

Per dare perciò la possibilità agli sviluppatori di interagire con il sistema sono stati creati i launcher e i chooser, ovvero dei meccanismi per demandare al sistema operativo stesso l’interazione con le applicazioni native. Questo significa che quando useremo una di queste API la nostra applicazione verrà sospesa, lasciando il controllo a quella nativa.

Che differenza c’è tra queste due tipologie? I launcher vengono utilizzati quando non ci sono informazioni di ritorno da restituire alla nostra applicazione, ma vogliamo semplicemente demandare un’operazione al sistema (ad esempio, la visualizzazione di una mappa); i chooser, invece, servono nel caso in cui è necessario importare dei dati da un’applicazione nativa verso la nostra (ad esempio, una foto o un contatto).

Già dalla versione 7.0 il numero di launcher e chooser disponibili era piuttosto nutrito: grazie ad essi era possibile importare foto dall’hub Pictures, contatti dall’hub People, spedire mail o sms, lanciare una ricerca sul Marketplace o su Bing, ecc.

Tutti i launcher e chooser condividono la medesima struttura: si crea una nuova istanza di una delle classi facenti parte del namespace **Microsoft.Phone.Tasks**, si valorizzano una o più proprietà che servono per definire le informazioni richieste dal launcher / chooser e si chiama il metodo **Show**, che lo esegue.

Nel caso dei chooser solitamente occorre registrarsi anche ad un evento asincrono **Completed**, che viene scatenato nel momento in cui il chooser ha terminato il suo compito e dobbiamo recuperarne i dati restituiti.

Nel resto dell’articolo ci concentreremo solo sui nuovi launcher e chooser introdotti in Mango.

## I launcher

## Bing Maps Direction Task

L’applicazione Maps di Windows Phone, come saprete, offre la possibilità di calcolare il percorso da percorrere per raggiungere un determinato luogo.

Questo launcher vi permette di lanciare una ricerca di questo tipo dalla vostra applicazione, semplicemente specificando il punto di partenza e quello di arrivo. Vediamo un esempio:

* 1. C#
  2. BingMapsDirectionsTask task = new BingMapsDirectionsTask
  3. {
  4. Start = new LabeledMapLocation(StartAddress.Text, null),
  5. End = new LabeledMapLocation(EndAddress.Text, null)
  6. };
  7. task.Show();

La classe **BingMapsDirectionsTask** espone due proprietà, **Start** e **End**, che rappresentano rispettivamente il punto di partenza e il punto di arrivo del nostro percorso. Entrambe le proprietà sono di tipo **LabeledMapLocation** e, in fase di inizializzazione, possiamo specificare due parametri:

* + L’indirizzo, sotto forma di stringa.
  + Le coordinate geografiche, sotto forma di oggetto di tipo **GeoCoordinate**.



Nessuno dei due parametri è obbligatorio: possiamo lanciare il calcolo del percorso solo specificando l’indirizzo (e impostando a **null** le coordinate, come nell’esempio) oppure solo le coordinate geografiche (lasciando vuoto il primo parametro).

### Bing Maps Task

Se abbiamo la necessità di mostrare informazioni di tipo geografico il controllo Map di Windows Phone può sicuramente dare un grande valore aggiunto alle nostre applicazioni. A volte, però, l’implementazione di tale controllo può essere troppo onerosa se, ad esempio, dobbiamo solo mostrare uno specifico luogo sulla mappa.

In queste situazioni ci viene in aiuto il **BingMapsTask**, che permette di demandare questa operazione all’applicazione Maps nativa: date le coordinate del luogo che vogliamo mostrare, l’utente verrà portato direttamente in Bing Maps per visualizzarlo.

Vediamo un esempio:

* 1. C#
  2. BingMapsTask task = new BingMapsTask
  3. {
  4. SearchTerm = Address.Text,
  5. ZoomLevel = 10
  6. };
  7. task.Show();

Questo launcher supporta tre proprietà:

* + **SearchTerm** rappresenta la parola chiave di ricerca, ad esempio un indirizzo o il nome di una città
  + **ZoomLevel** rappresenta il livello di zoom che vogliamo utilizzare di default.
  + **Center**: se conosciamo le coordinate precise del luogo che vogliamo mostrare, possiamo valorizzare questa proprietà, che è di tipo **GeoCoordinate**. All’apertura, la mappa verrà centrata su questa posizione. Attenzione che se avete valorizzato anche la proprietà **SearchTerm** la proprietà **Center** sarà praticamente ignorata: la mappa verrà centrata nel punto definito, ma si sposterà subito dopo nel luogo risultante dalla ricerca.



### ConnectionSettingsTask

Se avete già installato Mango sul vostro device vi sarete imbattuti in applicazioni come Network Dashboard o Connectivity Shortcuts, che offrono la possibilità di avere direttamente in home delle tile per accedere velocemente alle varie impostazioni di rete (Wi-Fi, Bluetooth, ecc.)

Queste applicazioni sono basate proprio su questo task, che permette da un’applicazione di aprire direttamente una specifica schermata dell’hub Impostazioni relativa alle connessioni.

Vediamo un esempio:

* 1. C#
  2. ConnectionSettingsTask task = new ConnectionSettingsTask();
  3. if (WiFi.IsChecked.Value)
  4. {
  5. task.ConnectionSettingsType = ConnectionSettingsType.WiFi;
  6. }
  7. if (Cellular.IsChecked.Value)
  8. {
  9. task.ConnectionSettingsType = ConnectionSettingsType.Cellular;
  10. }
  11. if (AirplaneMode.IsChecked.Value)
  12. {
  13. task.ConnectionSettingsType = ConnectionSettingsType.AirplaneMode;
  14. }
  15. if (Bluetooth.IsChecked.Value)
  16. {
  17. task.ConnectionSettingsType = ConnectionSettingsType.Bluetooth;
  18. }
  19. task.Show();

Il launcher espone solamente una proprietà, chiamata **ConnectionSettingsType**, che è un emumeratore, il quale contiene un valore per ogni tipo di connessione disponibile.

* + **WiFi**: per accedere alle impostazioni della connessione wireless.
  + **Cellular**: per accedere alle impostazioni della connessione dati su rete cellulare.
  + **AirplaneMode**: per attivare o disattivare la modalità aereo (la quale spegne in automatico tutti i tipi di connessione del telefono).
  + **Bluetooth**: per accedere alle impostazioni del bluetooth



## I chooser

### SaveRingtoneTask

Windows Phone 7.5 ha introdotto la tanto attesa possibilità di utilizzare suonerie personalizzate. Ma non solo! Grazie a questo chooser, potremo consentire alla nostra applicazione di salvare dei file audio direttamente tra le suonerie di sistema.

Vediamo un esempio:

* 1. C#
  2. SaveRingtoneTask task = new SaveRingtoneTask
  3. {
  4. Source = new Uri("appdata:/Assets/Ringtones/Ringtone01.wma"),
  5. DisplayName = "Ringtone 1"
  6. };
  7. task.Completed += (obj, args) =>
  8. {
  9. if (args.TaskResult == TaskResult.OK)
  10. {

MessageBox.Show("La suoneria è stata salvata!");

* 1. }
  2. else
  3. {
  4. MessageBox.Show("La suoneria non è stata salvata");
  5. }
  6. };
  7. task.Show();

Questo chooser supporta le proprietà:

* + **Source**: è l’URI della suoneria da salvare. Possiamo utilizzare, come nell’esempio, il prefisso **appdata**:/ per identificare un percorso all’interno del progetto, oppure isostore:/ nel caso il file sia salvato nell’isolated storage.
  + **DisplayName**: il nome della suoneria, che verrà visualizzato nell’elenco della sezione Impostazioni.



Chiamando il metodo **Show** l’utente avrà comunque la possibilità di cambiarne il nome e di impostarla come suoneria predefinita. Una volta che l’operazione è completata, torneremo alla nostra applicazione e verrà scatenato l’evento **Completed**. L’unica informazione restituita sarà l’esito dell’operazione, sotto forma di enumeratore di tipo TaskResult e valorizzato nella proprietà **TaskResult**.

Nell’esempio, ci limitamo a mostrare un semplice messaggio di conferma in base all’esito dell’operazione.

**Importante**! Affinchè un file audio sia utilizzabile come suoneria devono essere soddisfatte le seguenti condizioni:

* + Deve essere un file di tipo MP3 o WMA
  + Deve durare meno di 40 secondi
  + Deve avere una dimensione inferiore a 1 MB
  + Non deve essere protetto da sistemi di DRM

#### 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)