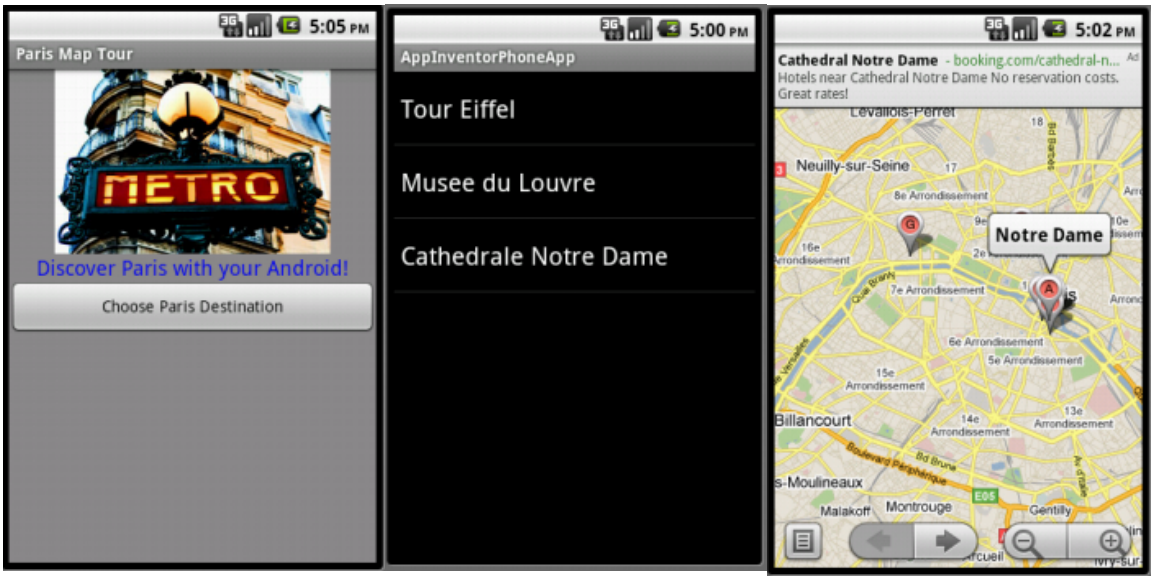
Paris Map Tour

In this chapter, you'll build an app that lets you create your own custom guide for a dream trip to Paris. And since a few of your friends can't join you, we'll create a companion app that lets them take a virtual tour of Paris as well. Creating a full-functioning map might seem really complicated, but App Inventor lets you use of the ActivityStarter component to launch Google maps for each of your virtual locations. First, you'll build an app that launches maps for the Eiffel Tower, the Louvre, and Notre Dame Cathedral with a single click. Then you'll modify the app to create a virtual tour of satellite maps that are also available from Google Maps.

What You'll Learn

This chapter introduces the following App Inventor components and concepts:

- The Activity Starter component for launching other Android apps from your app. You'll use it to launch Google Maps with various parameters.
- The ListPicker component for allowing the user to choose from a list of choices.



Design the Components

Create a new project in App Inventor and call it “ParisMapTour.” The user interface for the app has an Image with a picture of Paris, a Label with some text, a ListPicker component which comes with an associated button, and an ActivityStarter (non-visible) component. You can design the components using the snapshot in Figure 6-1.

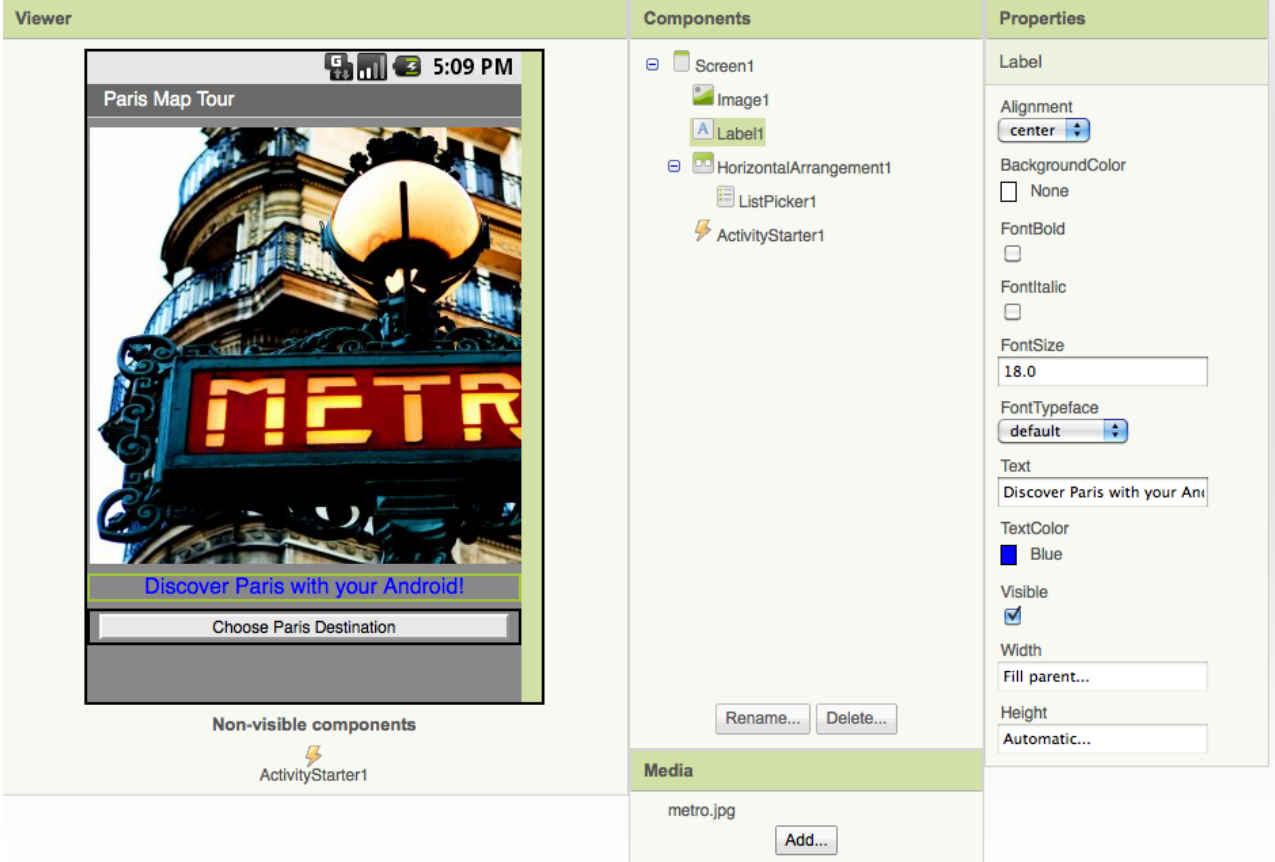


Figure 6-1. The UI for the app in the Component Designer.

The components listed in Table 6-1 were used to create this designer window. Drag each component from the Palette into the Viewer and name it as specified.

Table 6-1. Components for the Paris Map Tour

Component Type	**Palette Group**	**What you'll name it**	**Purpose of Component**
Image	Basic	Image1	Show a static image of Paris map on screen
Label	Basic	Label1	“Discover Paris with your Android”
ListPicker	Basic	ListPicker1	Display the list of destinations for choosing
ActivityStarter	other stuff	ActivityStarter1	Launches the

			maps app when a destination is chosen

Setting the Properties of ActivityStarter

ActivityStarter is a component that lets you launch any Android app: a browser, Google Maps, even another one of your own apps. When another app is launched from your app, the user can click the back button to get back to your app. You'll build Paris Map Tour so that the Maps application is launched to show particular maps based on the user's choice. The user can then hit the back button to return to your app and choose a different destination.

ActivityStarter is a relatively low-level component in that you'll need to set some properties with information familiar to a Java Android SDK programmer, but foreign to the rest of the 99.999% people in the world. For this app, just enter the properties as specified in table 6-2 and **be careful**--even the upper/lower case letters are important.

Table 6-2. ActivityStarter properties for launching Google Maps.

Property	**Value**
Action	android.intent.action.VIEW
ActivityClass	com.google.android.maps.MapsActivity
ActivityPackage	com.google.android.apps.maps

In the Blocks Editor, you'll set one more property, DataUri, which will allow you to launch a specific map in Google Maps. This property must be set in the Blocks Editor instead of the Component Designer because it needs to be dynamic; it will change based on whether the user chooses to visit the Louvre, the Eiffel Tower, or the Notre Dame Cathedral.

We'll get to the Blocks Editor in just a moment, but there's just a couple more details to take care of and you can move on to programming the behavior for your components:

- Download the file metro.jpg from the book site (TK) onto your computer, then choose Add in the Media section to load it into your project.You'll then need to set it as the Picture property of Image1.

- The ListPicker component comes with a button--when the user clicks it, the choices are listed. Set the text of that button by setting the Text property of ListPicker1 to "Choose Paris Destination."

Add Behaviors to the Components

In the Blocks Editor, you'll need to define a list of destinations, and two behaviors:

1. When the app begins, the app loads the destinations into the ListPicker component so the user can choose.

2. When the user chooses a destination from the ListPicker, the Maps application is launched to show a map of that destination. In this first version of the app, you'll just open Maps and tell it to run a search for the chosen destination.

Create a list of destinations

Open the blocks editor and create a **variable** with the list of destinations for the Paris Map Tour using the blocks listed in Table 6-3.

Table 6-3. Blocks for creating a destinations variable.

Block Type	Drawer	Purpose
def variable ("Destinations")	Definitions	create a list of the destinations
make a list	Lists	to add the items to the list
text ("Tour Eiffel")	Text	the first destination
text ("Musee du Louvre")	Text	the second destination
text ("Cathedrale Notre Dame")	Text	the third destination

The **destinations** variable will call the **make a list** function, in which you can plug in the text values for the three destinations in your Tour as shown in Figure 6-2.

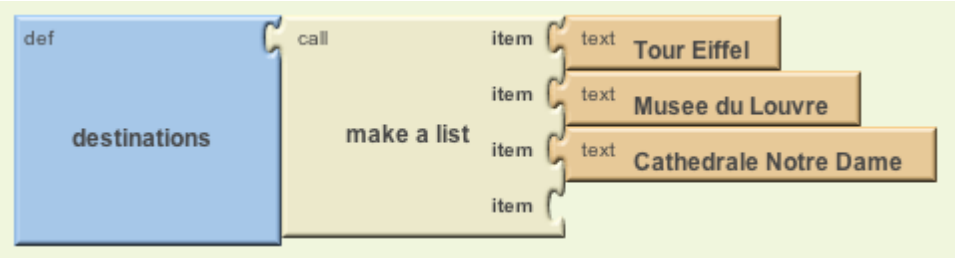


Figure 6-2. Creating a list is easy in App Inventor.

Let the User Choose a Destination

The purpose of the ListPicker component is to display a list of items for the user to choose from. You preload the choices into the ListPicker by setting the property Elements to a list.

For this app, you want to set the ListPicker's Elements property to the destinations list you just created. Because you want this to happen when the app launches, you'll define this behavior in the **Screen1.Initialize** event. You'll need the blocks listed in Table 6-4.

Table 6-4. Blocks for launching the ListPicker when the app starts.

Block Type	Drawer	Purpose
Screen1.Initialize	Screen1	this event is triggered when the app starts
set ListPicker1.Elements to	ListPicker1	set this property to the list you want to appear
global destinations	My Definitions	the list of destinations

How the Blocks Work

Screen1.Initialize is triggered when the app begins. As shown in Figure 6-3, the event handler sets the Elements property of ListPicker so that the three destinations will appear.

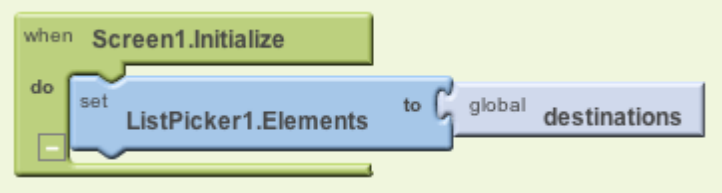


Figure 6-3. Put anything you want to happen when the app starts in a Screen1.Initialize event handler.

Test this behavior. You'll need click on "restart the app on the device" in the blocks editor first. Then on the phone, click the button labeled "Choose Destination". The list picker should appear with the three items.

Opening Maps with a Search

Next, you'll program the app so that when the user chooses one of the destinations, the ActivityStarter launches Google Maps and searches for the selected destination.

When the user chooses an item from the ListPicker component, the **ListPicker.AfterPicking** event is triggered. In the event handler for AfterPicking you need to set the DataUri of the ActivityStarter component so it knows which map to open, then you need to launch Google Maps using **ActivityStarter.StartActivity**. The blocks for this functionality are listed in table 6-5.

Table 6-5. Launching Google Maps with the Activity Starter

Block Type	Drawer	Purpose
ListPicker1.AfterPicking	ListPicker1	this event is triggered when the user chooses from ListPicker
set ActivityStarter1.DataUri to	ActivityStarter1	the DataUri tells Maps which map to open on launch
make text	Text	build the DataUri from two pieces of text
text ("geo:0,0?q=")	Text	the first part of the DataUri expected by Maps
ListPicker1.Selection	ListPicker1	the item the user chose
ActivityStarter1.StartActivity	ActivityStarter1	launch Maps

How the Blocks Work

When the user chooses from the ListPicker, the chosen item is stored in **ListPicker.Selection** and the **AfterPicking** event is triggered. As shown in Figure 6-4, the DataUri property is set to a text object which combines “http://maps.google.com/?q=” with the chosen item. So if the user chose the first item, “Tour Eiffel”, the DataUri would be set to “http://maps.google.com/?q=Tour Eiffel”

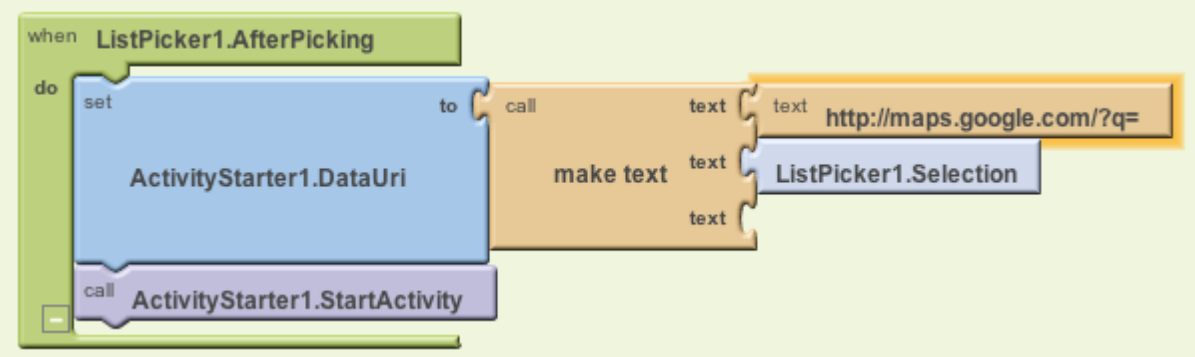


Figure 6-4. Setting the Data URI to launch the selected map.

Since you already set the other properties of the ActivityStarter so that it knows to open Maps, the **ActivityStarter1.StartActivity** block launches the Maps app and invokes the search proscribed by the DataUri.

Test this behavior. Restart the app and click the "Choose Destinations" button

again. When you choose one of the destinations, does a map of that destination appear? Google Maps should also provide a back button to return to your app to again--does that work? (You may have to click the back button a couple of times.)

Set up a virtual tour

Now let's spice the app up so that it opens up some great zoomed-in and street views of the Paris monuments so your friends can follow along while you're away. To do this, you'll first explore Google Maps to obtain the URLs of some specific maps. You'll still use the same Parisian landmarks for the destinations, but when the user chooses one, you'll use the *index* of their choice (or its position in the list) to select and open a specific zoomed or street-view map.

Before going on, you may want to SaveAs your project so you have a copy of the simple map tour you've created so far. That way, if you do anything that causes issues in your app, you can always go back to the version you know works and try again.

Finding the DataUri for specific maps

The first step is to open Google Maps on your computer to find the specific maps you want to launch for each destination:

- On your computer, browse to maps.google.com
- Search for a landmark (e.g., Eiffel Tower)
- Zoom in to the level you desire.
- Choose the type of view you want, e.g., Address, Satellite, or Street view.
- Click on the Link button near the top-right of the Maps window and copy the URL for the map. You'll use this URL or parts of it to launch a map from your App Inventor app.

Using this scheme, table 6-5 shows the URLs you'll use.

Table 6-5. Virtual tour URLs for Google Maps.

Landmark	**Maps URL**
Eiffel Tower	http://maps.google.com/maps?f=q&source=s_q&hl=en&geocode=&q=eiffel+tower&sll=37.0625,-95.677068&sspn=48.909425,72.333984&ie=UTF8&hq=Tour+Eiffel&hnear=Tour+Eiffel,+Quai+Branly,+75007+Paris,+Ile-de-France,+France&ll=48.857942,2.294748&spn=0.001249,0.002207&t=h&z=19
Musee Louvre	http://maps.google.com/maps?

	f=q&source=s_q&hl=en&q=louvre&sll=48.86096,2.335421&sspn=0.002499,0.004415&ie=UTF8&t=h&split=1 &filter=0&rq=1&ev=zi&radius=0.12&hq=louvre&hnear=&ll=48.86096,2.335421&spn=0.002499,0.004415&z=18
Notre Dame (Street View)	http://maps.google.com/maps? f=q&source=s_q&hl=en&q=french+landmarks&sll=48.853252,2.349111&sspn=0.002411,0.004415 &ie=UTF8&t=h&radius=0.12&split=1&filter=0&rq=1&ev=zi&hq=french+landmarks&hnear=&ll=48.853252,2.349111&spn=0,0.004415 &z=18&layer=c&cbll=48.853046,2.348861&panoid=74fLTqeYdgkPYj6KKLlqgQ&cbp=12,63.75,,0,-35.58

To view any of these maps, paste the URLs above into a browser. The first two are zoomed satellite views, while the third is a street view.

You can use these URLs directly to launch the maps you want. You can also define cleaner URLs yourself using the Google Maps protocols defined at mapki.com. For example, you can show the Eiffel Tower map using only the GPS coordinates found in the long URL above and the Maps “geo” protocol:

```
geo:48.857942,2.294748?t=h&z=19
```

With such a DataUri, you’ll get essentially the same map as the map based on the full URL. The GPS coordinates are extracted from the longer URL above. The “t=h” specifies that Maps should show a hybrid map with both satellite and address view, and the z=19 specifies the zoom level. If you’re interested in the details of setting parameters for various types of Maps, check out the documentation at mapki.com.

So you can get used to using both types of URLS, lets use the geo format for the first two DataUri settings, and the full URL for the third.

Defining the dataURIs List

You'll need a list named dataURIs, one for each of the maps you want to show. Create this list as shown in Figure 6-5 so that the items correspond to the items in the destinations list, e.g., the first dataURI should correspond to the first destination (Tour Eiffel).

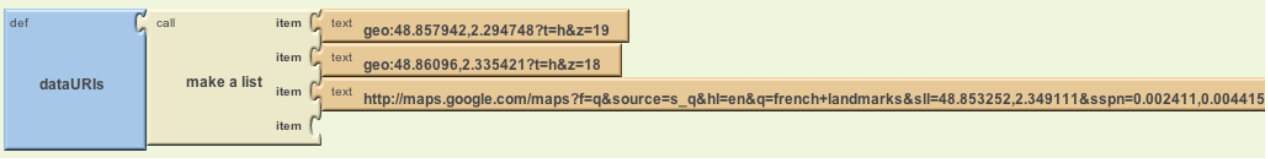


Figure 6-5. The list of maps for your virtual tour.

The first two items shown are DataURIs for the Eiffel Tower and the Louvre. They both use the "geo:" prototcol. The third DataURI is not shown completely as the block is too long for this page. You should copy this URL from the entry for "Notre Dame, Street View" in the table above and place it in a **text** block.

Modify the ListPicker.AfterPicking Behavior

In the first version of this app, the **ListPicker.AfterPicking** behavior set the DataUri to the *concatenation* (or combination) of "http://maps.google.com/?q=" and the destination the user chose from the list, e.g., Tour Eiffel. In this second version, the AfterPicking behavior must be more sophisticated, as the user is choosing from one list (**destinations**) but the DataUri must be selected from another list (**dataURIs**). Specifically, when the user chooses one of the items from the ListPicker, you need to know the *index* of the choice so you can use it to select the correct DataUri from the **dataURIs** list. We'll explain what an index is in a moment, but it helps to have the blocks set up first to better understand the concept. There's quite a few blocks required for this functionality, all of which are listed in Table 6-6.

Table 6-6. Blocks for using one list to pick an item from another.

Block Type	Drawer	Purpose
def variable ("index")	Definitions	This variable will hold the index of the user's choice
number (1)	Math	Initialize the index variable to 1
ListPicker1.AfterPicking	ListPicker1	This event is triggered when the user chooses an item
set index to	My Definitions	Set this variable to the position of the selected item
position in list	Lists	Gets the position (index) of a selected item
ListPicker1.Selection	ListPicker1	The selected item, e.g., "Tour Eiffel". Plug into the thing slot of position in list.

global destinations	My Definitions	Plug this into the list slot of position in list
set ActivityStarter.DataUri	ActivityStarter	Set this before starting activity to open map
select list item	Lists	Select an item from the DataURIs list
global DataURIs	My Definitions	The list of DataURIs
global index	My Definitions	This variable holds the position of the chosen item
ActivityStarter.StartActivity	ActivityStarter	Launch the Maps app

How the Blocks Work

When the user chooses an item from the ListPicker, the **AfterPicking** event is triggered. The chosen item, e.g., "Tour Eiffel", is in **ListPicker.Selection**. The event handler uses the this to find the position of the selected item, or the *index* value, in the **destinations** list. The **index** corresponds to the position of the chosen destination in the list. So if "Tour Eiffel" is chosen, the index will be 1, if "Musee du Louvre" is chosen, it will be 2, and if "Cathedrale Notre Dame" is chosen, the index will be 3.

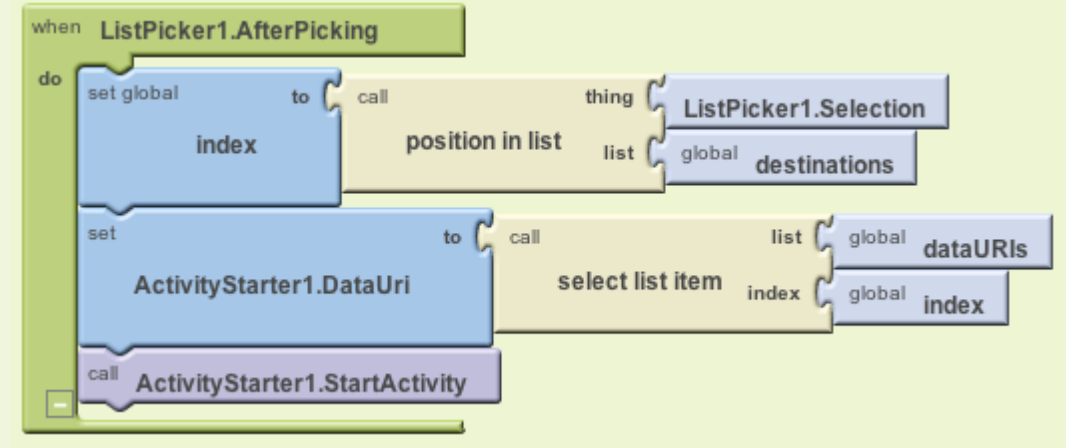


Figure 6-6. Finding the position of an item in a list with the index.

The **index** can then used to select an item from another list, in this case **dataURIs**, and to set that entry as the ActivityStarter's DataUri. Once this is set, the map can be launched with **ActivityStarter.StartActivity**.

Test this behavior. On the phone, click the button labeled "Choose Destination." The list picker should appear with the three items. Choose one of the items and see what map appears.

Variations

- Create a virtual tour of some other exotic destination or of your workplace or school.
- Create a customizable “Virtual Tour” app that lets a user create a guide for a location of their choice by entering the name of each destination along with the URL of a corresponding map. You’ll need to store the data in a TinyWebDB database, and create a “take virtual tour” app that works with the entered data. For an example of how to create a TinyWebDB database, see the MakeQuiz/TakeQuiz app in Chapter XX.

Summary

Here are some of the ideas covered in this chapter:

- The ListPicker component lets the user choose from a list of items. ListPicker's Elements property holds the list, the Selection property holds the selected item, and the **AfterPicking** event is triggered when the user chooses.
- The ActivityStarter component allows your app to launch other apps. This chapter demonstrated its use with the Maps application, but you can launch a browser or any other Android app as well, even another one you created yourself. See the http://appinventor.googlelabs.com/learn/reference/other/activitystarter.html for more information.
- You can launch a particular map in Google Maps by setting the DataUri property. You can find URIs by configuring a particular map in the browser then choosing the link button to find the URI. You can either place such a URI directly into the DataUri of your ActivityStarter, or build your own URI using the protocols defined at mapki.com.
- You can identify the index of text that appears in a list with its position in a list block. With ListPicker, you can use position in list to find the index of the item chosen by the user. This is important if, as in this chapter, you need the index to choose an item from a second, related list.