Creating Map-Based Activities

The MapView provides a compelling User Interface option for presentation of geographical data. One of the most intuitive ways of providing context for a physical location or address is to display it on a map. Using a MapView, you can create Activities that feature an interactive map.

Map Views support annotation using both Overlays and by pinning Views to geographical locations. Map Views offer full programmatic control of the map display, letting you control the zoom, location, and display modes — including the option to display satellite, street, and traffi c views.

In the following sections, you'll see how to use Overlays and the MapController to create dynamic map-based Activities. Unlike online mashups, your map Activities will run natively on the device, allowing you to leverage its hardware and mobility to provide a more customized and personal user experience.

Introducing MapView and MapActivity

This section introduces several classes used to support Android maps:

❑ MapView is the actual Map View (control).

❑ MapActivity is the base class you extend to create a new Activity that can include a Map View. The MapActivity class handles the application life cycle and background service management required for displaying maps. As a result, you can only use a MapView within MapActivity-derived Activities.

❑ Overlay is the class used to annotate your maps. Using Overlays, you can use a Canvas to draw onto any number of layers that are displayed on top of a Map View.

❑ MapController is used to control the map, allowing you to set the center location and zoom levels.

❑ MyLocationOverlay is a special overlay that can be used to display the current position and orientation of the device.

❑ ItemizedOverlays and OverlayItems are used together to let you create a layer of map markers, displayed using drawable with associated text.

Creating a Map-Based Activity

To use maps in your applications, you need to create a new Activity that extends MapActivity. Within it, add a MapView to the layout to display a Google Maps interface element. The Android map library is not a standard package; as an optional API, it must be explicitly included in the application manifest before it can be used. Add the library to your manifest using a uses-library tag within the application node, as shown in the XML snippet below:

```
<uses-library android:name="com.google.android.maps"/>
```

Google Maps downloads the map tiles on demand; as a result, it implicitly requires permission to use the Internet. To see map tiles in your Map View, you need to add a uses-permission tag to your application manifest for android.permission.INTERNET, as shown below:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Once you've added the library and confi gured your permission, you're ready to create your new mapbased Activity.

MapView controls can only be used within an Activity that extends MapActivity. Override the onCreate method to lay out the screen that includes a MapView, and override isRouteDisplayed to return true if the Activity will be displaying routing information (such as traffi c directions).

The following skeleton code shows the framework for creating a new map-based Activity:

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.os.Bundle;
public class MyMapActivity extends MapActivity {
private MapView mapView;
```

```
private MapController mapController;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.map_layout);
mapView = (MapView)findViewById(R.id.map_view);
}
@Override
protected boolean isRouteDisplayed() {
// IMPORTANT: This method must return true if your Activity
// is displaying driving directions. Otherwise return false.
return false;
}
}
```

The corresponding layout fi le used to include the MapView is shown below. Note that you need to include a maps API key in order to use a Map View in your application.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<com.google.android.maps.MapView
android:id="@+id/map_view"

android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:enabled="true"
android:clickable="true"
android:apiKey="mymapapikey"
/>

</LinearLayout>
```

At the time of publication, it was unclear how developers would apply for map keys. Invalid or disabled API keys will result in your MapView **not loading the map image tiles. Until this process is revealed, you can use any text as your API key value.**

Figure 7-5 shows an example of a basic map-based Activity

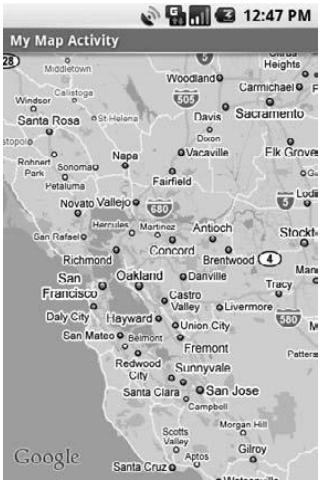

Figure 7-5

Android currently recommends that you include no more than one MapActivity *and one* MapView *in each application.*