Hello Purr

This chapter gets you started building apps. It presents the key elements of App Inventor: the Designer *and the* Blocks Editor*, and it leads you through the basic steps of creating your first app. When you're done, you'll be ready to build apps on your own.*

A first program with a new computer system typically prints the message “Hello World” to show that everything is connected correctly. This tradition goes back to the 1970s and work on the C programming language at Bell Labs by Brian Kernighan (who is now a visiting scholar at Google working on the App Inventor team!). With App Inventor, even the simplest apps do more than just show messages: they play sounds and react when you touch the phone. So we’re going to get started right away with something more exciting: your first app will be *Hello Purr*, a picture of a kitty that meows when you touch it, and purrs when you shake it.

Figure 1.1. The HelloPurr app

What You’ll Learn

The chapter covers the following topics:

- You build apps by selecting components and then telling them what to do and when to do it.
- You use the Designer to select components. Some components are visible on the phone screen and some aren't.
- You can add media (sounds and images) to apps by uploading them from your computer.
- You use the Blocks Editor to assemble blocks that define the components' behavior
- App Inventor’s *Instant App Construction* lets you see see how apps will look and behave on the phone step-by-step, even as you are building them.

The App Inventor Environment

You can setup App Inventor using the instructions at http://appinventor.googlelabs.com/learn/

setup/. App Inventor runs primarily through the browser but you need to download some software to your computer's desktop and change some settings on your phone. Typically you can get setup in just a few minutes, though sometimes there are issues with setting up device drivers for particular Android phones. If you have any phone issues, we suggest you get started using the Android emulator that comes packaged with the App Inventor download.

The App Inventor programming environment has three key parts:

1. The *Component Designer* runs in your browser window. You use the Designer to select components for your app and specify their properties.
2. The *Blocks Editor* runs in a separate window from the Designer. You use the Blocks Editor to create behaviors for the components.
3. The phone where you can actually run and test your app as you are developing it. If you don't have an Android phone handy, you can test the apps you build using an Android emulator that comes integrated with the system.

Figure 1-2 shows the App Inventor programming environment with the HelloPurr app loaded and being tested in the emulator:

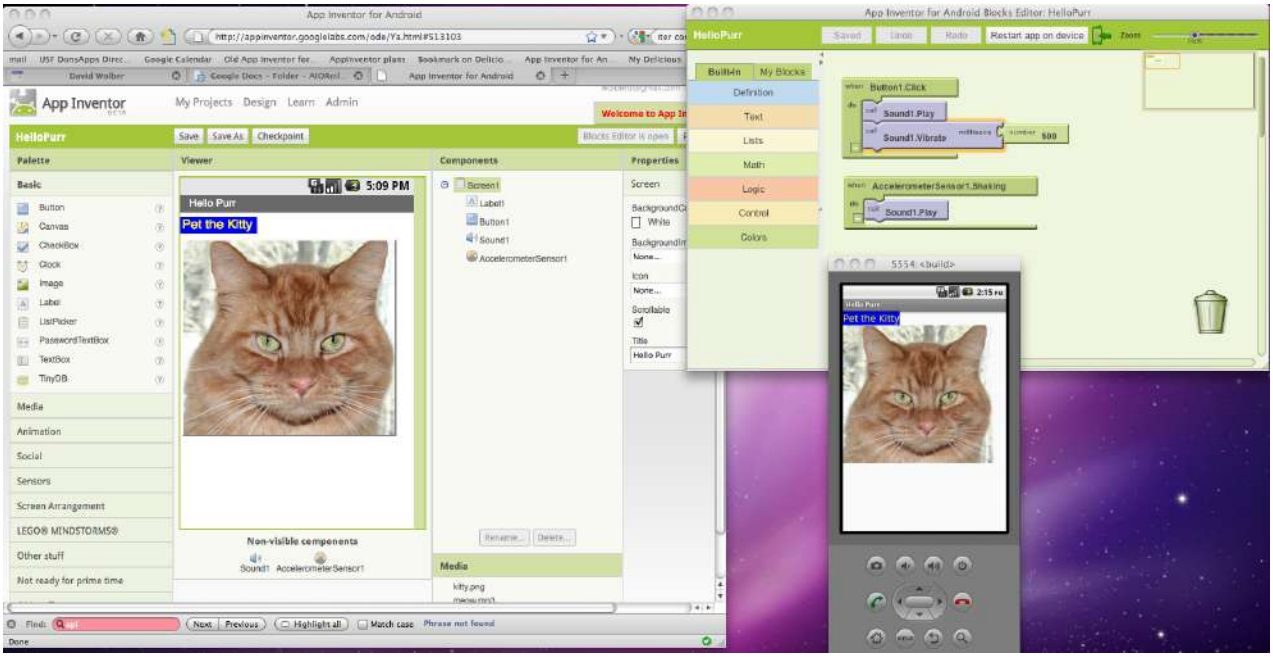

Figure 1-2. The Component Designer, Blocks Editor, and Emulator

You start App Inventor by browsing to http://appinventor.googlelabs.com. If this is the first time you've used App Inventor, you'll see the Projects page, which will be mostly blank, because you don't have any projects created yet. To create a project, click on New at the top left of the page, enter the project name HelloPurr (one word with no spaces), and click OK.

The first window that appears is the Component Designer. When it appears , click on the Open

Blocks Editor in the menu at the top-right. The Blocks Editor comes up in a separate window with the aid of a tool called Java Web Start. It usually takes about 30 seconds.

If everything is okay, you'll be asked to verify some things and then the dialog shown in figure 1-3 will appear:

Figure 1-3. You can choose whether to test on a phone or emulator

If you have an Android phone and a USB cable, plug the phone into the computer and select "Use a phone". Otherwise, click on "Use the emulator" and you can test on that.

If all is well, you should see something like Figure 1-2 (without the emulator if you connected to a phone). If you're having problems here, review the setup instructions at http://appinventor.googlelabs.com/learn/setup/.

Design the Components

The first tool you'll use is the *Component Designer*. Components are the elements you combine to make apps, like ingredients in a recipe. Some components are very simple, like a Label component, which just shows text on the screen, or a Button component that you tap to initiate an action. Other components are more elaborate: a drawing Canvas that can hold still images or animations, an accelerometer (motion) sensor that works like a Wii controller and detects when you move or shake the phone, components that make or send text messages, components that play music and video, components that get information from Web sites, and so on.

When you open the Designer, it will appear as in figure 1-4:

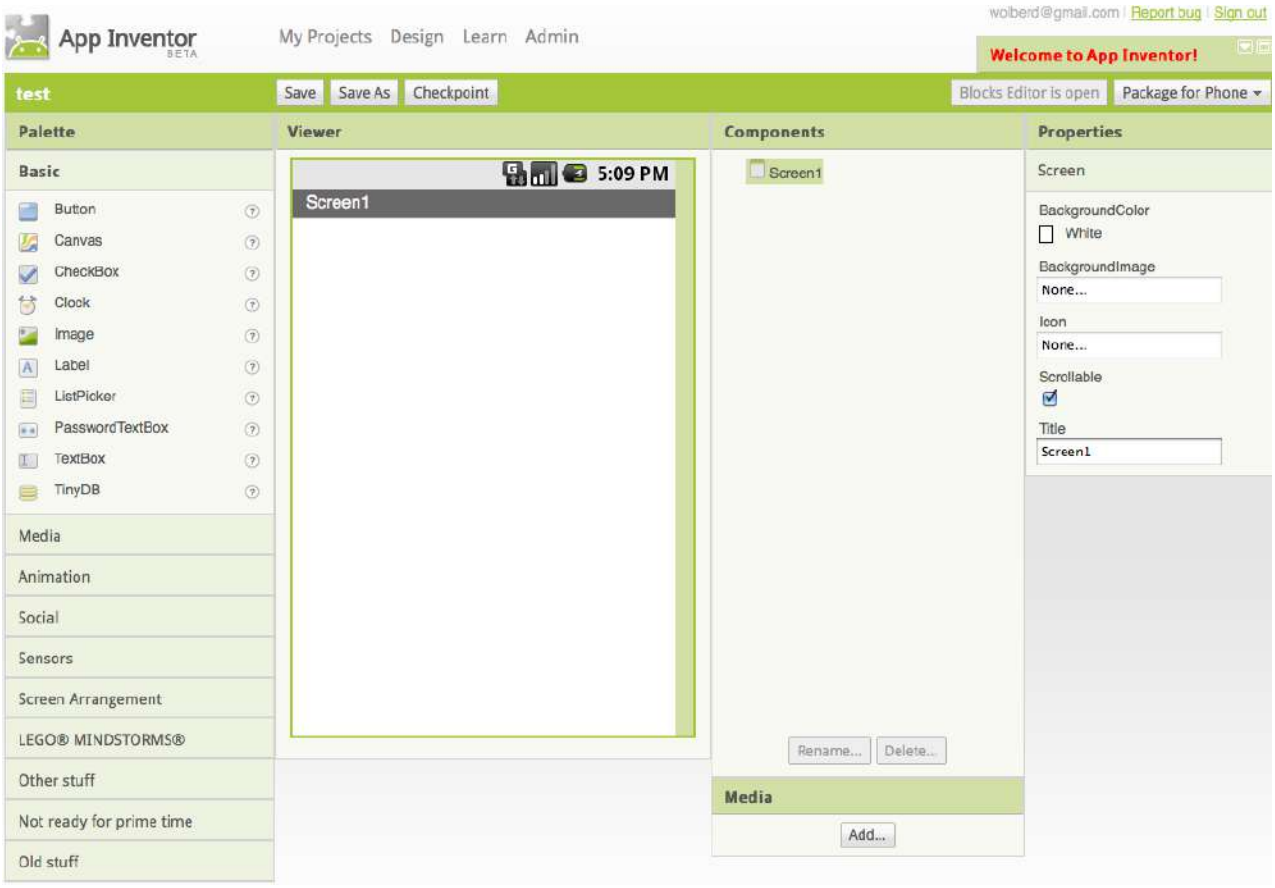

Figure 1-4. The App Inventor Component Designer

The Designer is divided into several areas:

- Toward the center is a white area called the *Viewer*. This is where you place components and arrange them. The viewer shows only a rough indication of how it will look. For example, a line of text might break at a different place in your app than what you see in the Viewer. To see how your app will really appear, you'll need to load it either on your phone or in the emulator that comes with App Inventor.
- To the left of the Viewer is the *Palette*, which is a list of components that you can select from. The Palette is divided into sections; currently only the Basic components are visible, but you can see components in other sections of the Palette by clicking on the headers labeled Media, Animation, etc.
- To the right of the Viewer is the *Components list*, which lists the components in your project. Any Component that you drag into the viewer will show up in this list. Currently, the project at this point has only one component listed: `Screen1`, which represents the phone screen itself.
- Under the Components list is an area that shows the *Media* (pictures and sound) in the project. This project doesn't have any media yet, but you'll be adding some soon.
- At the far right is a section that shows the *Properties* of components; when you click on a Component in the Viewer, you see its Properties detailed here. Properties are details

about each component that you can change: for a Label component you might see properties related to color, text, font, and so on. Right now it's showing the properties of the screen, which include a background color, a background image, and a title.

For the HelloPurr app, you'll need two visible components, the label that says "Pet the kitty" and a button with an image of a cat in it. You'll also need a non-visible Sound component that knows how to play sounds, even "meow" sounds, and an Accelerometer component for detecting when the phone is being shaken. Don't worry, we'll walk you through each one of these step-by-step!

Make the label

The first component to add is a Label:

1. Go to the Palette, click on Label (at about 5 down in the list of components) and drag it to the Viewer. You'll see a rectangular shape appear on the Viewer, with the words *Text for Label1*.
2. Look at the Properties box at the right of the screen. It shows the properties of the label. There's a property called Text about halfway down, with a box for the label's text. Change the text to read *Pet the Kitty* and press return. You'll see the text change in the Viewer.
3. Change the BackgroundColor of the label by clicking on the box, which currently reads *None*, to select a color from the list that appears. Select Blue. Also change the TextColor of the label to Yellow. Finally change the FontSize to 20.

The Designer should now appear as in figure 1-5:

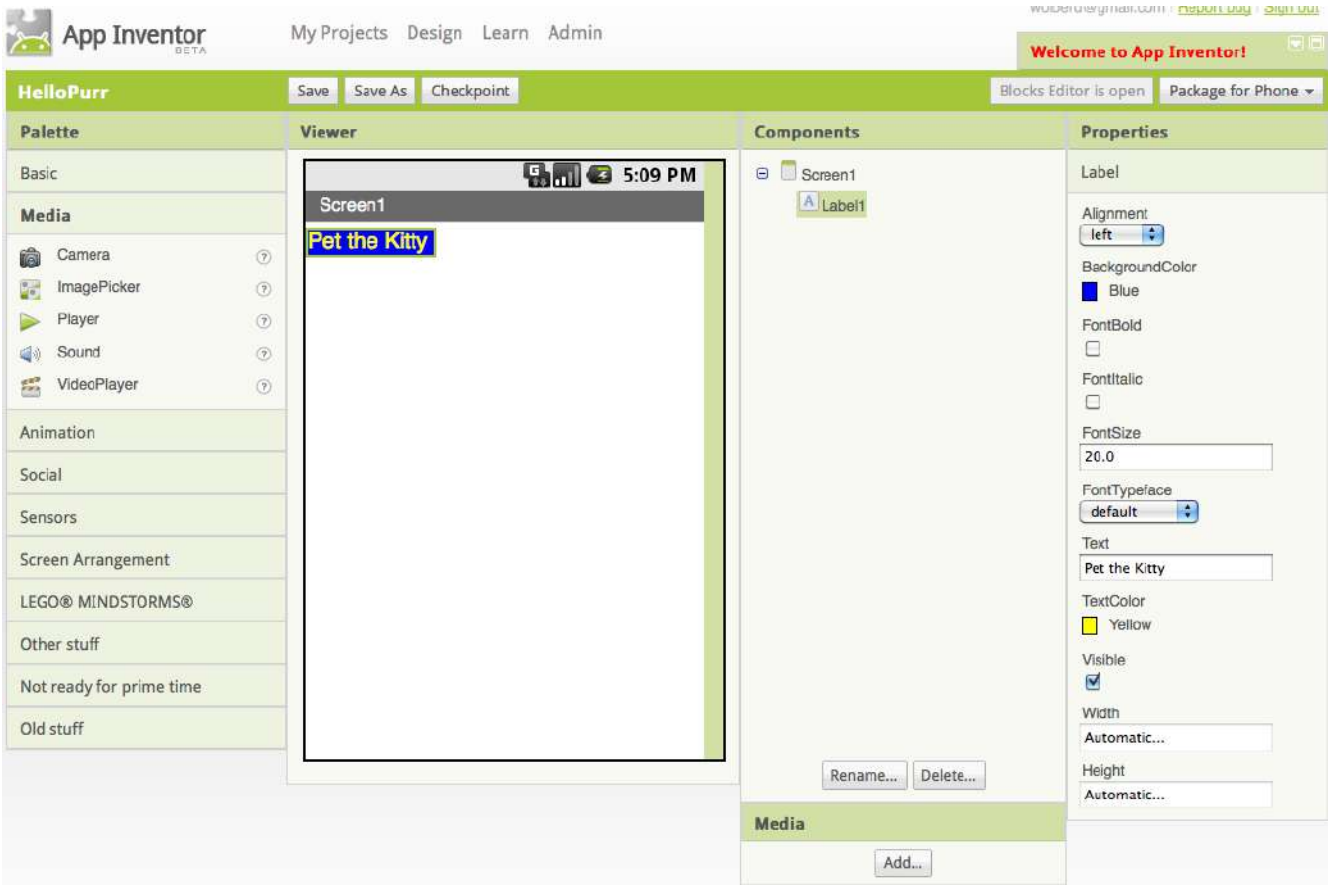

Figure 1-5. The app now has a label.

If your phone is connected, you'll see the label appear on the phone as well. In App Inventor, you build the application on the phone right along with picking the components in the Designer. That way, you can see right away how your application will look. This is called *Instant App Construction,* and it also applies to the behaviors you create for the components, as you'll see below.

Add the button

The kitty for Hello Purr is implemented as a Button component. To make it, go to the Palette in the Designer and click on Button (at the top of the list of components). Drag it on to the Viewer, placing it below the label. You'll see a rectangular button shape appear on the Viewer. After about 10 seconds, the button should appear on the phone. You can go ahead and tap the phone button--do you think anything will happen? That's because your app hasn't told the button to do anything yet.

Now we've got a button that we'll be able to use to trigger the sound effect when someone clicks on it, but we really want it to look like the picture of the kitty, and not a plain old button. To make the button look like the kitty:

1. First, you need to download a picture of the kitty and save it on your computer desktop. You can it from the site for this book at http://oreilly.com/appinventor/hellopurr. The picture is the file called “kitty.png.” While you’re there, you can also download the sound file we need, “meow.mp3.” The kitty.png file is a standard image format similar to .jpg and .gif: all of these file types will work in App Inventor, as will most standard sound files like .mpg.
2. The Properties box should be showing the properties of the button. If it isn't, click on the image of the button in the Viewer to expose the button's properties on the right. In the Properties box, click on the area under `Image` (currently *None...*). A box appears with a button marked *Add...* ,
3. Click on *Add...* and you'll see *Upload file....* Click on *Choose File* and browse to select the kitty.png file you downloaded to your computer earlier, and click OK.
4. You'll see a yellow message at the top of the screen: "Uploading kitty.png to the AppInventor server". After about 30 seconds the message and the upload box will disappear, and kitty.png should be listed as the image property for the button. You'll also see this listed in the Media area of the Designer window. And if you look at the phone, you'll see the kitty picture is now showing-- the button now looks like a kitty.
5. You may have also noticed on your phone that the kitty picture has the words "Text for button 1" showing over it. You probably don't want that in your app, so go ahead and change the `text` property of `Button1` to something like "Pet me" or just delete the text to make it blank.

Now the Designer should appear as in figure 1-6:

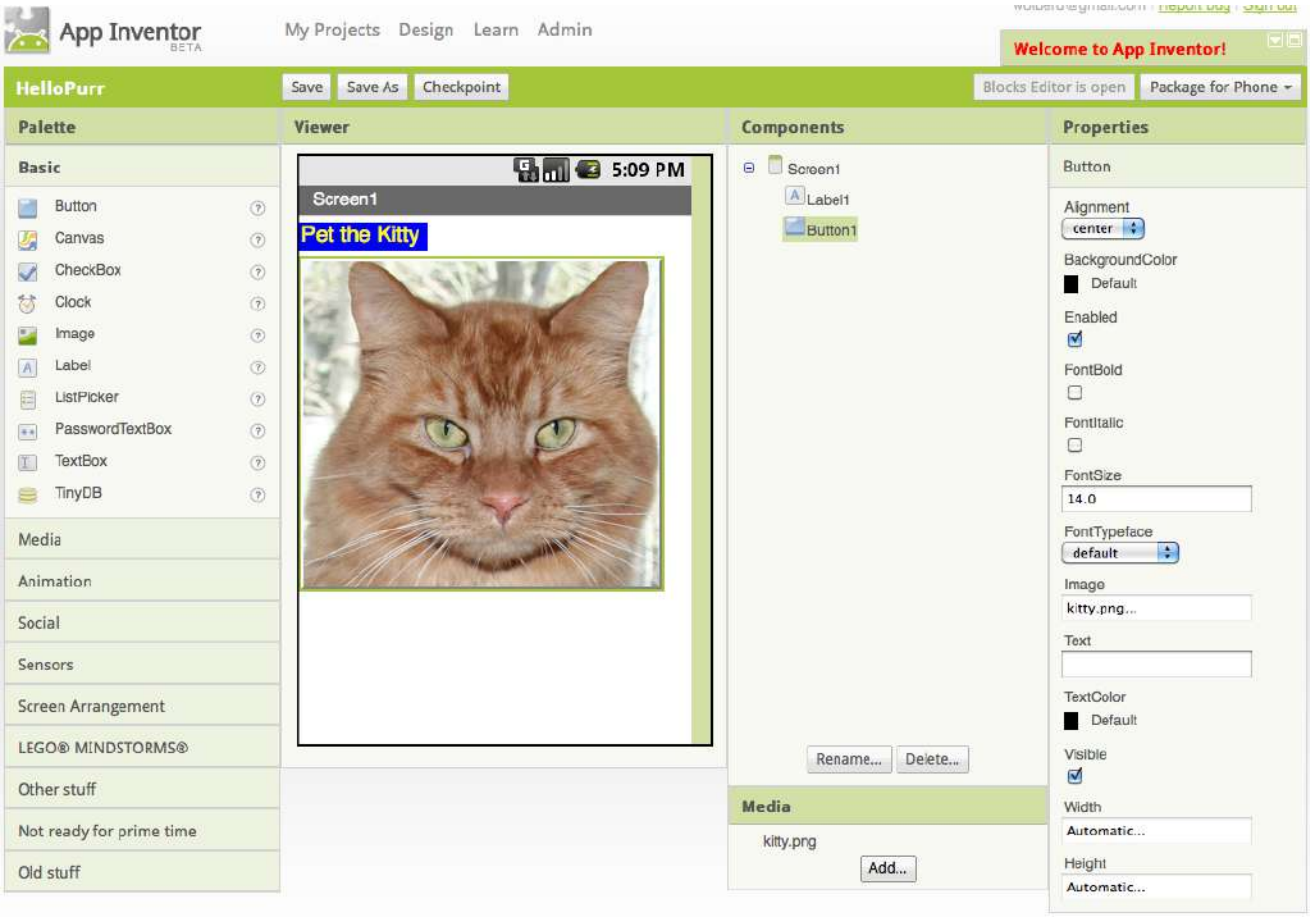

Figure 1-6. The app with a label and a button with an image on it.

Add the meow sound

In your app, the kitty will meow when you tap the button. For this, you'll need to add the meow sound and program the button behavior to play that sound when the button gets clicked:

1. If you haven't downloaded the "meow.mp3" file to your computer's desktop, do so now at http://oreilly.com/appinventor/helloPurr.
2. Go to the Palette at the left of the browser window and click on the header marked Media to expand the Media section of the palette of components. Drag out a Sound component and place it in the Viewer. Wherever you drop it, it will appear in the area at the bottom of the Viewer marked Non-visible components. Non-visible components are objects that do things for you but don't appear in the user interface of the app.

3. Click on Sound1 to show its properties. Set its Source to "meow.mp3". You'll need to follow the same steps to upload this file from your computer as you did to get the kitty picture. When you're done, you should see both "kitty.png" and "meow.mp3" listed in the Media section of the Designer.

You should now have the components depicted in Table 1-1:

Component Type	Palette Group	Name of component	Purpose
Button	Basic	Button1	Press to make the kitty meow
Label	Basic	Label1	Shows the text “Pet the Kitty”
Sound	Media	Sound1	Play the meow sound

Table 1-1. The components you’ve added to the HelloPurr app.

Add Behaviors to the Components

You’ve just added a Label, Image, and Sound as the building blocks for your first app. Now let’s make the kitty meow when you tap the button. You do this with the Blocks Editor. If you’re the Blocks Editor isn’t yet open, click on Open the Blocks Editor in the top-right of the Component Designer.

Look at the Blocks Editor window. This is where you tell the components what to do, and when to do it. You're going to tell the kitty button to play a sound when the user taps it. If components are ingredients in a recipe, you can think of blocks as the cooking instructions.

Make it meow

At the top left of the window, you'll see buttons labeled *Built-In* and *My Blocks*. Click on My Blocks, and you'll see a column that includes a drawer for each component you created in the Designer: Button1, Label1, Screen1, and Sound1. Click the drawer for the button. The drawer opens showing a selection of pieces that you can use to tell the button what to do, starting with “**Button1.Click**” at the top, as shown in figure 1-7:

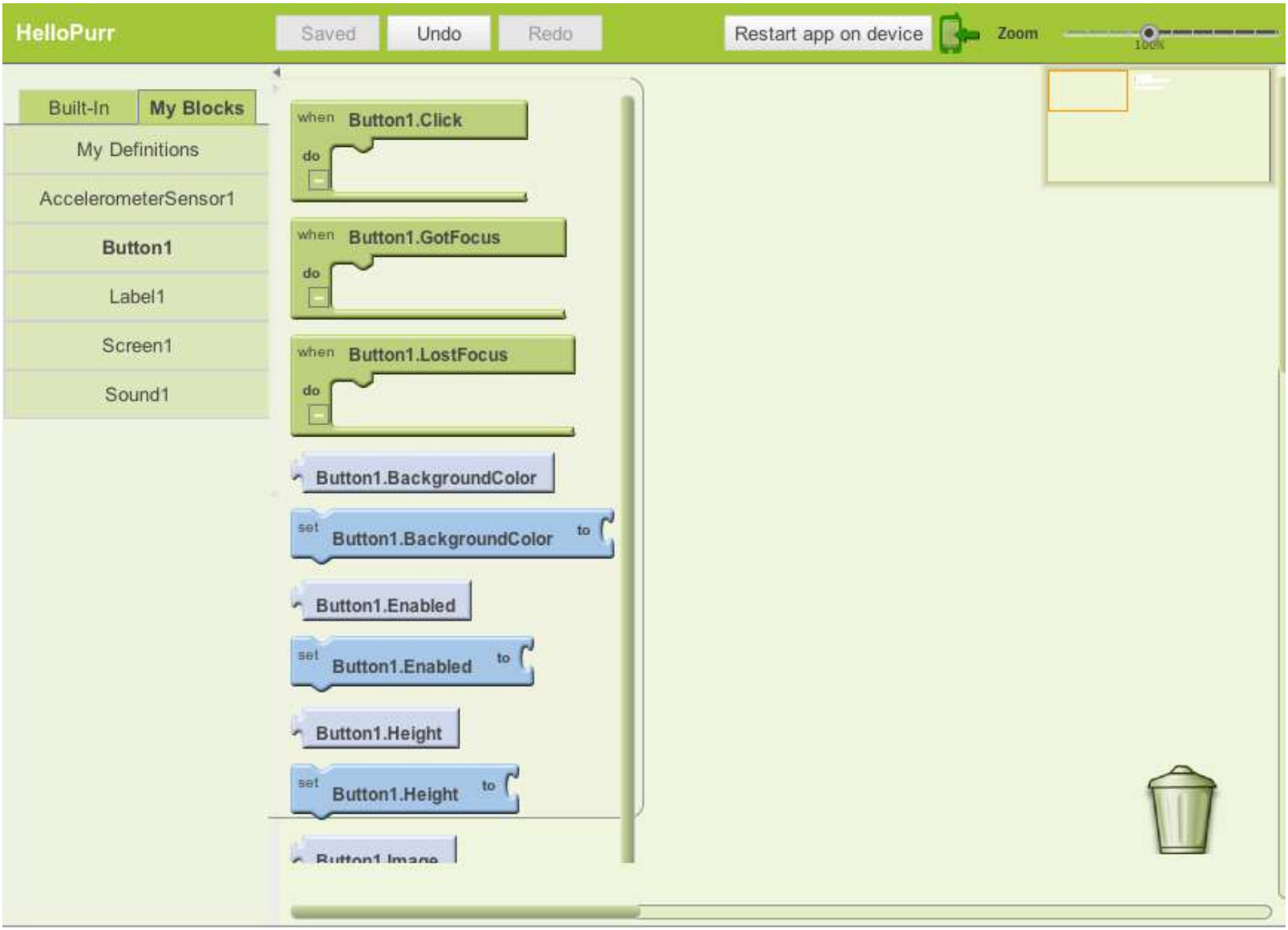

Figure 1-7. Clicking on Button1 shows the component's event handlers and property blocks.

Click on the block labeled **Button1.Click** and drag it out into the workspace. When you're looking for the block, notice that the word "when" is smaller than **Button1.Click**. Blocks with the word "when" in them are called *event handlers*: they specify what components should do when some particular event happens. In this case, the event is the user of the app clicking the button, as shown in Figure 1-8. Next, we'll put some blocks in to show what should happen in response.

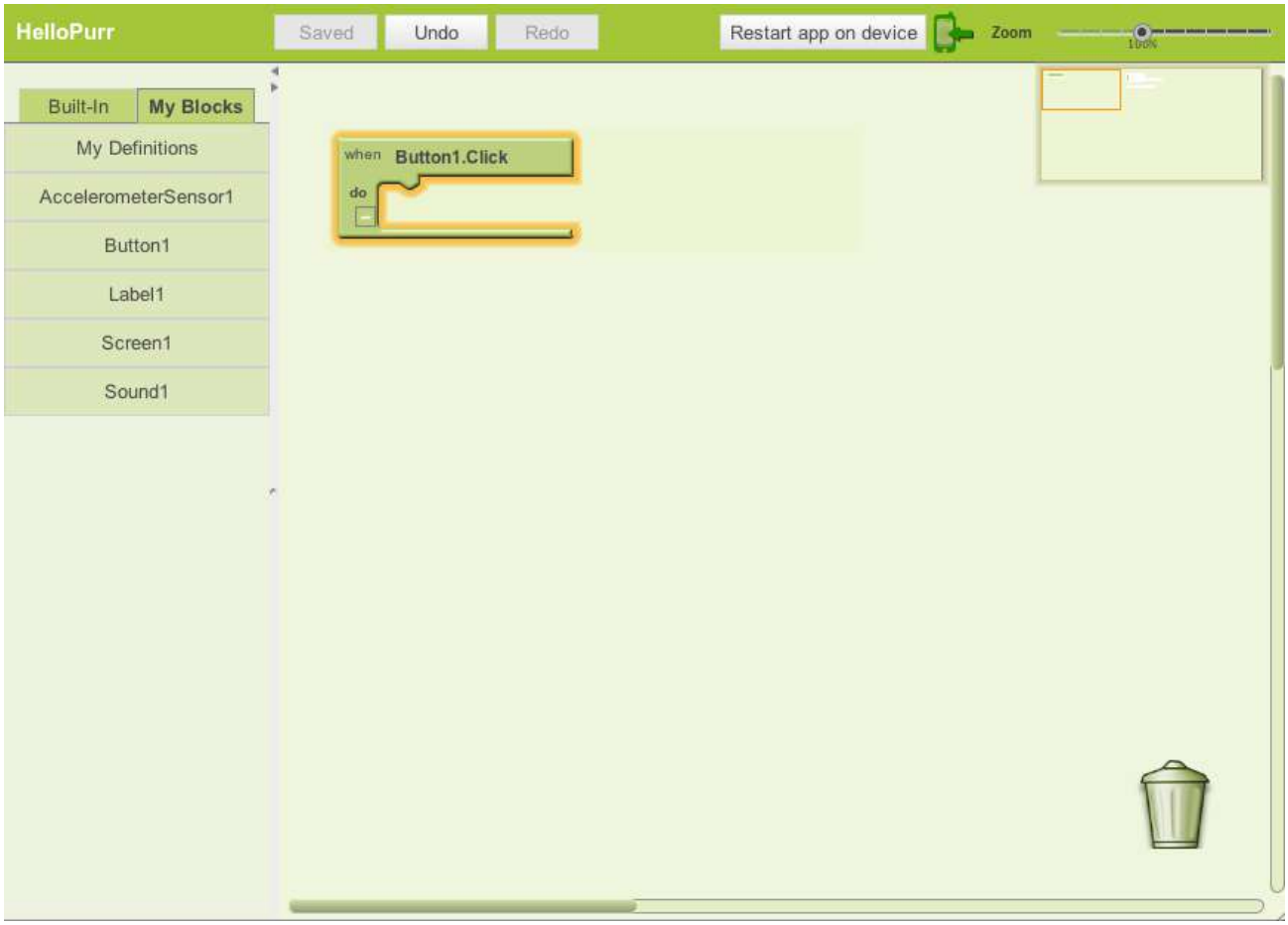

Figure 1-8. The Button1.Click block is for specifying a response to a click.

Click on Sound1 in My Blocks to open the drawer for the sound component, and drag out the **call Sound1.Play** block, fitting it into the gap marked do in the **Button1.Click** block. (Remember that earlier, we set the property for `Sound1` to be the meow sound file you'd downloaded to your computer.) Blocks with the word *call* make components do things. The two blocks should snap together to form a unit, as shown in figure 1-9, and you should hear a snapping sound.

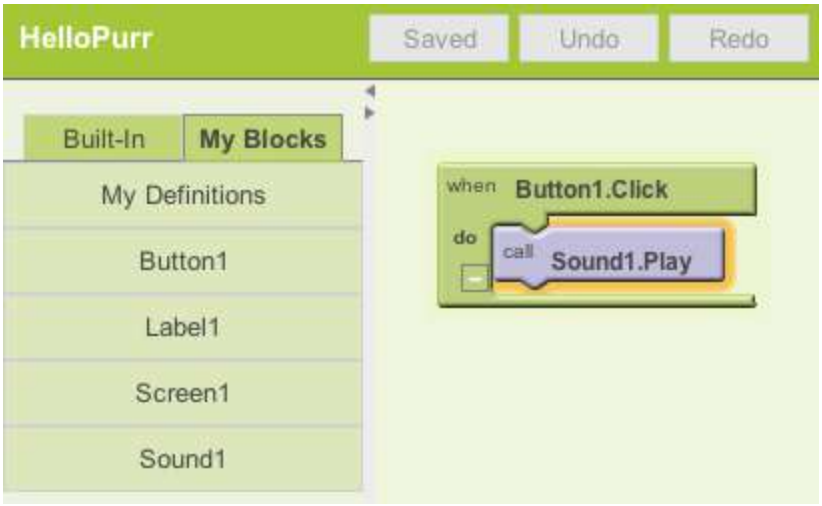

Figure 1-9. Now when you click the button, the meow sound will play.

Unlike traditional programming code (which often looks like a jumbled mess of gobbledygook "words"), Blocks in App Inventor spell out the behaviors you are trying to create. In this case, we are essentially saying "Hey App Inventor, when someone clicks on the kitty button, play the meow sound."

Finally, let's check to make sure everything is working properly--it's important to test your app each time you add something new. Tap the button on the phone (or click it using the emulator). You should hear the kitty meow. Congratulations, your first app is running!

Add a purr

Now we're going to make the kitty purr as well as meow when you tap the button. We"ll simulate the purr by making the phone vibrate.This is easy to do because the Sound component that you used to play the meow sound can make the phone vibrate as well--you don't need to do anything different in the Designer, you can just add a second behavior to the button click in the Blocks Editor:

1. Go to the Blocks Editor and click Sound1 in My Blocks to open the drawer.

2. Select **call Sound1.Vibrate** and drag it in under the **call Sound1.Play** block in the **Button1.Click** slot. The block should click into place as shown in figure 1-10. If it doesn't, try dragging it so that the little dip on the top of **call Sound1.Vibrate** touches the little bump on the bottom of **call Sound1.Play**.

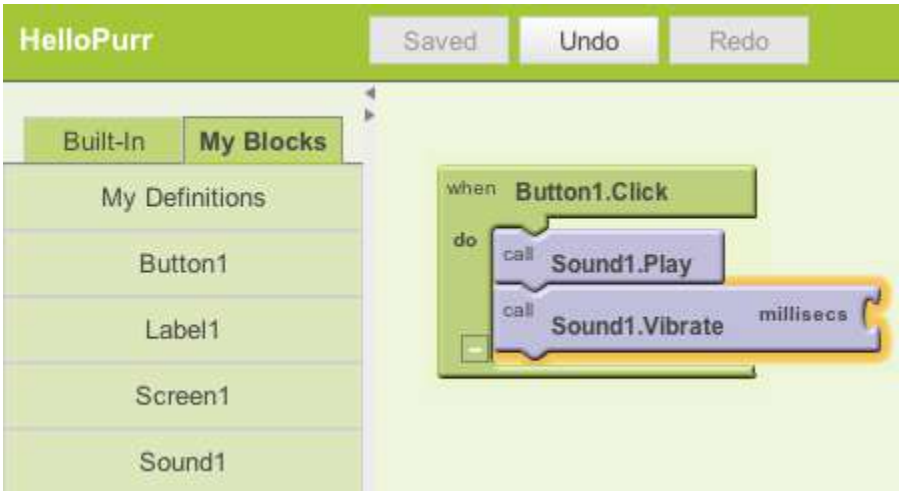

Figure 1-10. Play the sound and vibrate on the Click event

3. You've likely noticed that the Vibrate block says "millisecs" in the top right. You need tell the Vibrate block how long it should vibrate for. You can specify this time in thousandths of a second (milliseconds), which is pretty common for many programming languages. So, to make the phone vibrate for half a second, put in a value of 500 milliseconds. Click in an empty spot on the screen, and you'll see a menu pop up.

4. Click on the green Math button in the menu that pops up, as shown in figure 1-11. You should get a list, with 123 as the first item. 123 indicates a block that represents a number.

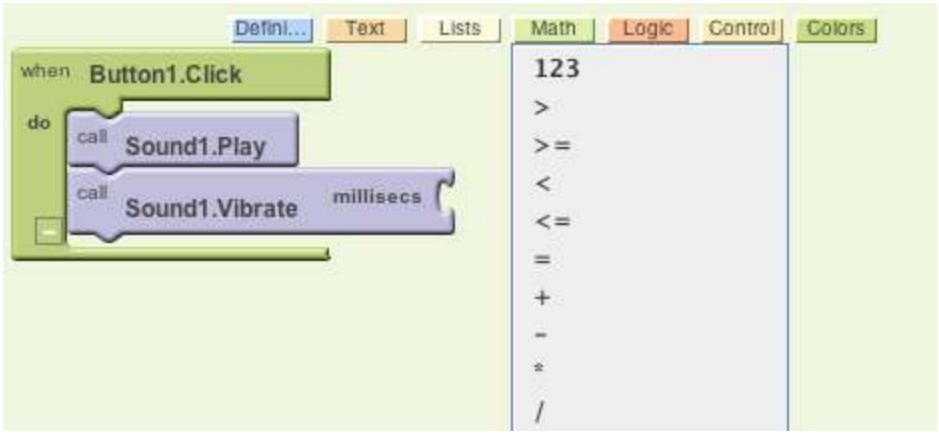

Figure 1-11. Opening the Math Drawer.

5. Click on the 123 and you'll get a green block with the number 123, as shown in figure 1-12.

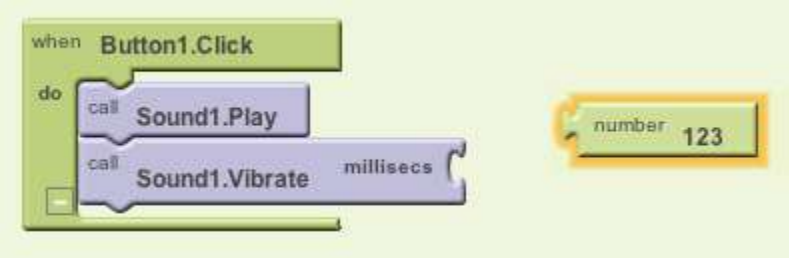

Figure 1-12. Choose a number block (123 is default value).

6. Change the 123 to 500 by clicking on it and typing a new value, as shown in figure 1-13.

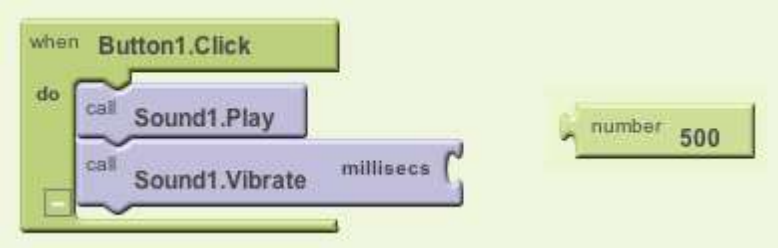

Figure 1-13. Change the value to 500.

7. Plug the 500 number block into the socket at the right of **call Sound1.Vibrate**, as shown in figure 1-14:

Figure 1-14. Plug the 500 into the milliseconds slot.

Try it! Tap the button on the phone and you'll feel the purr for half a second.

Shaking the phone

Now let's add a final feature that taps into a cool feature of Android phones: have the kitty meow when you shake the phone. To do this, you'll use a component called an *AccelerometerSensor* that can sense when you shake or move the phone around.

1. In the Designer, expand the Sensors area in the Palette components list and drag out an AccelerometerSensor. Wherever you place it in the Viewer, it will move to the Non-visible components section at the bottom of the Viewer.
2. Go to the Blocks Editor. There should be a new drawer for AccelerometerSensor1. Open it and drag out the **AccelerometerSensor1.Shaking** block - it should be the second block in the list.
3. Just as you did with the sound and the button click, drag out a **call Sound1.Play** block and fit it into the gap in **AccelerometerSensor1.Shaking**. Try it out by shaking the phone.

Figure 1-15 shows the blocks for the completed HelloPurr app::

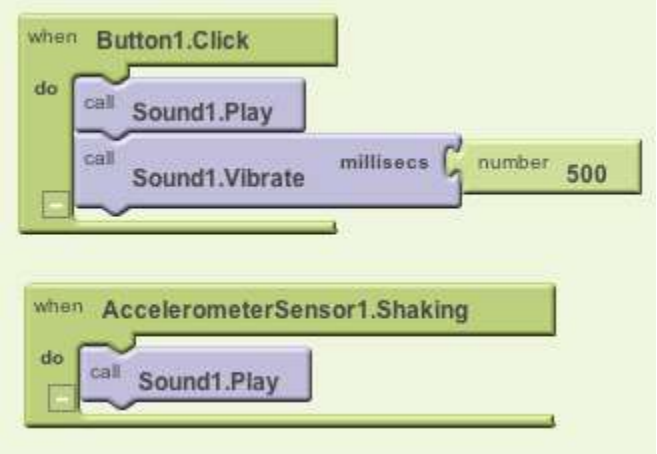

Figure 1-15. The blocks for HelloPurr.

Packaging the app for the phone

App Inventor is a cloud computing tool, meaning your app is stored on Google's servers as you work. So if you close App Inventor, your app will be there when you return.

If your phone is connected to your computer, you've been taking advantage of App Inventor's *instant app construction*, which lets you test your apps as you build them (this is also called *live testing*). Live testing only works while your phone is connected. If you disconnect, the app running on the phone will stop, and you won't find an icon for it anywhere: it was never truly installed.

You can package up and install the completed app so that it works when it's not connected to the computer. First, make sure your phone allows apps to be downloaded from places other than the Android Market. Typically, you do this by going to Settings | Applications on your phone and checking "Unknown sources". Then go to the Designer, click Package for Phone, and select Download to Connected Phone. You should see the messages Saving and then Packaging, a process that takes 45 seconds to a minute. After the Packaging message disappears, continue to wait for another 10-15 seconds while the finished app is downloaded to the phone. You'll see a notice that confirms the download when everything is complete.

Look at the apps available on your phone and you'll see a new app called HelloPurr. That's the app you just built. You run it just like any other app. To test it, make sure that you run your new app not the App Inventor Phone application. You can now unplug the phone, even reboot it, and kill all applications, and your new packaged application will still be there. Your packaged app is separate from the project on App Inventor. You can do more work on the project in AppInventor, connecting the phone with the USB cable as before. That won't change the packaged app: it's a finished version. Of course, you can keep modifying your project in AppInventor, package the result, and download the new self-contained version to replace the one on the phone.

So package your Hello Purr app and show it to your friends! Two things you might notice as you play with it:

1. As you shake the phone, the meows will sound strange, as if they are echoing. That's because the accelerometer sensor is triggering the shaking event many times a second, and so the meows are overlapping. If you look at the sound component in the designer, you'll see a property called Minimum interval. That determines how close together successive sounds can start. It's currently set at one-half second (500 milliseconds), which is less than the duration of a single meow. By playing with the minimum interval, you can change how much the meows overlap.
2. If you run the packaged app and walk around with the phone in your pocket, your phone will meow every time you move suddenly — something you might find embarrassing. Android apps are typically designed to keep running even when you're not looking at them. Your app with the accelerometer and the meow just keeps going. To really quit the app, bring up HelloPurr and press the phone's menu button. You'll be offered an option to stop the application.

Sharing the App

You can share your app in a couple of ways. To share the executable app, click on Package for Phone and choose Show Barcode. In 30 seconds or so a barcode will appear that you or anyone can scan to their phone in order to install your app. You can email this barcode to people or put it on the web. Just let your “customers” know they need to allow “unknown sources” in their phone’s settings and (at least at this time) be registered App Inventor users.

To share the source code (blocks), click on My Projects, check the app you want to share (e.g. HelloPurr) and select More Actions | Download Source. The file created on your computer will have a .zip extension. You can email this file to someone, and they can open App Inventor, choose More Actions | Upload Source, and select the .zip file. This will give them their own complete copy of your app which they can edit and customize.

Summary

- You build apps by selecting components in the Designer and then telling them what to do and when to do it in the Blocks Editor.
- Some components are visible and some aren't. The visible ones appear in the user-interface of the app. The non-visible ones do things like play sounds.
- You define components’ behavior by assembling blocks in the Blocks Editor. You first drag out an event handler like Button1.Click, then place command blocks like Sound.Play within it. Whatever blocks are within Button1.Click will be performed when the user clicks the button.
- Some commands need extra information to make them work. An example is Vibrate,

which needs to know how many milliseconds to vibrate. These values are called arguments.
- Numbers are represented as number blocks. You can plug these into commands that take numbers as arguments.
- App Inventor has sensor components. The AccelerometerSensor can detect when the phone is moved.
- You can package the apps you build and download them to the phone where they run independently of App Inventor.

*** CUTS below here *****

Part 2
This section continues your introduction to the basics of App Inventor. You'll extend your Hello Purr app to make the kitty purr, and also meow when you shake the phone instead of clicking the button. Lastly, you'll learn how to install the app on your phone.
The tutorial covers the following ??? and concepts:
- Some commands take arguments
- Numbers are represented as number blocks.
- App Inventor has sensor components.
- You can package the apps you build and download them to a phone.

Restarting
If you took a break after finishing part 1, you should go back to appinventor.googlelabs.com in your browser, connect your phone to your computer, start the Blocks Editor, and click Connect to phone. App Inventor remembers the last project you were working on. Check that your app is showing on the phone and that the kitty still meows when you "pet" it.

this was under 'save your app' right after part 1
To finish, try one more thing: Unplug your phone from the computer and then plug the phone back in. On your computer, go to the Blocks Editor and click Connect to Phone. After about 30 seconds, the phone app restarts with your work restored. App Inventor remembered your project and shows it on the phone. Even if you log out and come back later, your project will be saved.

When an app is initially built, it works only when the phone is connected to the computer, using some magic from App Inventor. In HelloPurr, Part 2, you'll teach the kitty new tricks and also learn how to get the app working as a self-contained application that works even when the phone is not connected

...If you're reading through this chapter to see what App Inventor is like, you can just continue on. But for learning how to build things, its often better to follow along and build as you read.