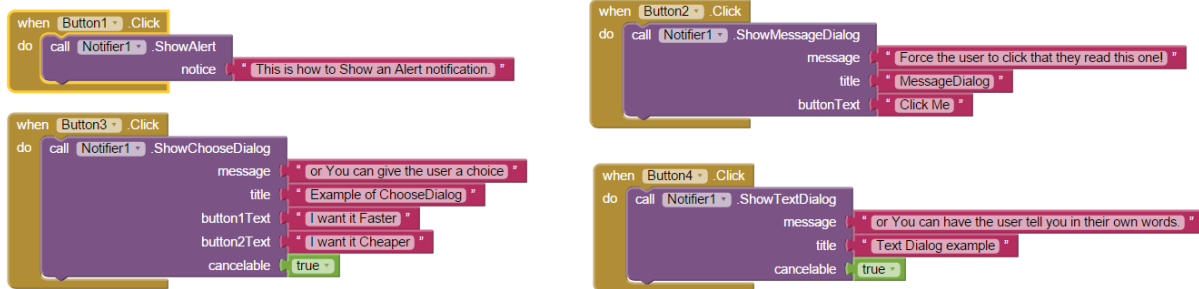


Make This

Chapter 5 - Using Notifier

In this tutorial, you'll learn about the App Inventor **notify** component, which gives you the capability to provide feedback to users and even prompt feedback from them. This tutorial teaches the following skills:

- **Changing the Title of a Screen**
- **Adding an Image to an App**
- **Notifications that Fade Away**
- **Notifications that Require a Click to Dismiss**
- **Notifications that Present a Choice**
- **Notifications that Gather Input from the User**



How Does Notifier Help Me Create More Powerful Apps?

Think about the times you've used an app and needed to fill out a form to supply certain required information:

- **Does everything always go perfectly?** Nope! Perhaps you forgot to fill in some required information in a form. Or maybe you used a word when the required information was supposed to be a number.
- **Do you always know what you need to do next?** Maybe there is an important instruction you need before you continue through the app.
- **How do you know if you accomplished a task or goal?** Did you get some acknowledgement when you completed the form successfully?

Alerts and messages are important in apps to give feedback to (or prompt feedback from) users. The **Notifier component** in App Inventor provides a range of options for generating feedback within your apps. Notifier can generate alert dialogs, messages, and temporary alerts, or create Android log entries.

Note: Before attempting this exercise, complete the Chapter 2 and 3 exercises to familiarize yourself with the App Inventor interface, getting your Android device connected, and starting a new project.

Building the Notifier Example App

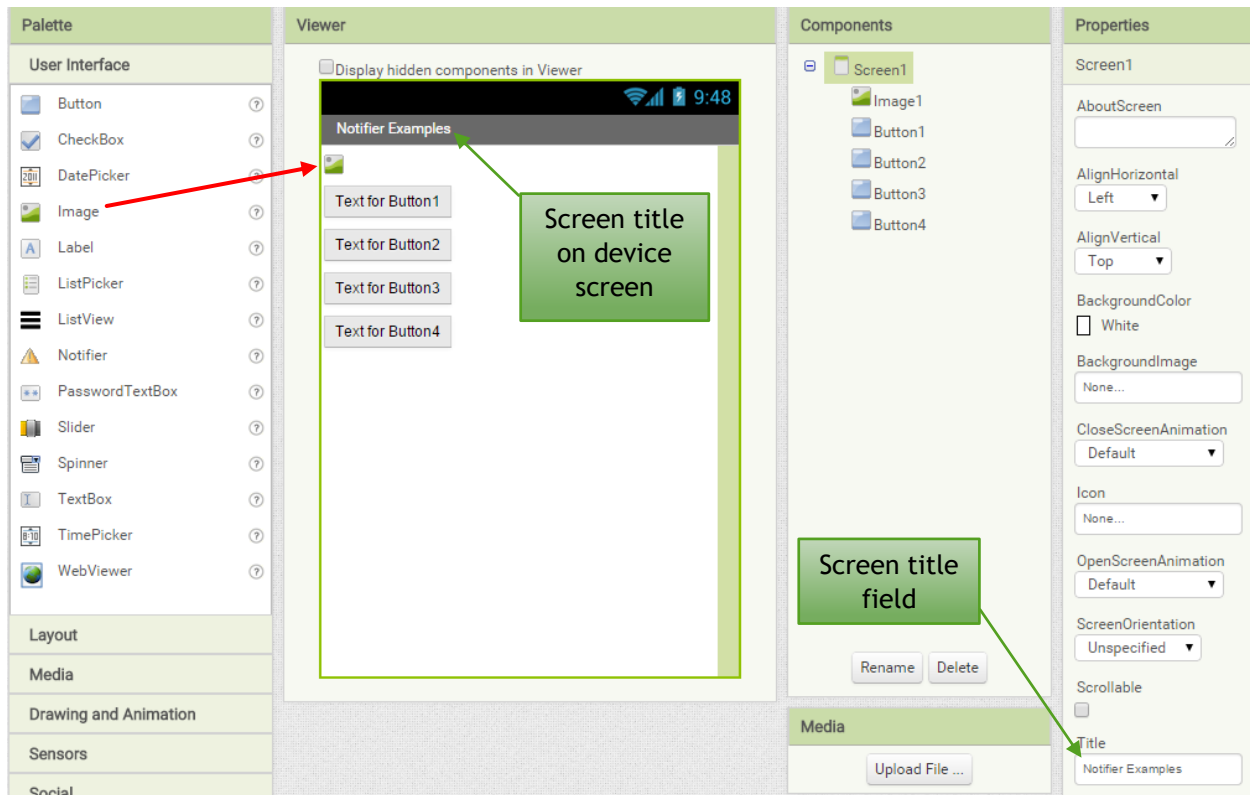
You will add an image and four buttons, then program each button to display a different type of alert. You will also need an image. You can either use the image that is provided for this exercise (**TIA12E_Chpt_5_Background.jpg**) or use one of your own.

1. Navigate to <http://appinventor.mit.edu/explore/>. If necessary, sign in with your Google Account.
2. Start a new project named *NotifierExample*.
3. Connect App Inventor to your Android device.

Part 1 - Constructing the Interface

1. From the **Palette**, under **User Interface**, drag an **image component** and four **button components** into the **Viewer**.
2. In the **Components** window, select **Screen1**.
3. In the **Properties** window, in the **Title field**, replace the default text with *Notifier Examples*. Click anywhere outside the Title field.

Note: Any text you enter in the Title field is displayed at the top of your screen as the screen title. Naming screens helps users understand where they are in an app.

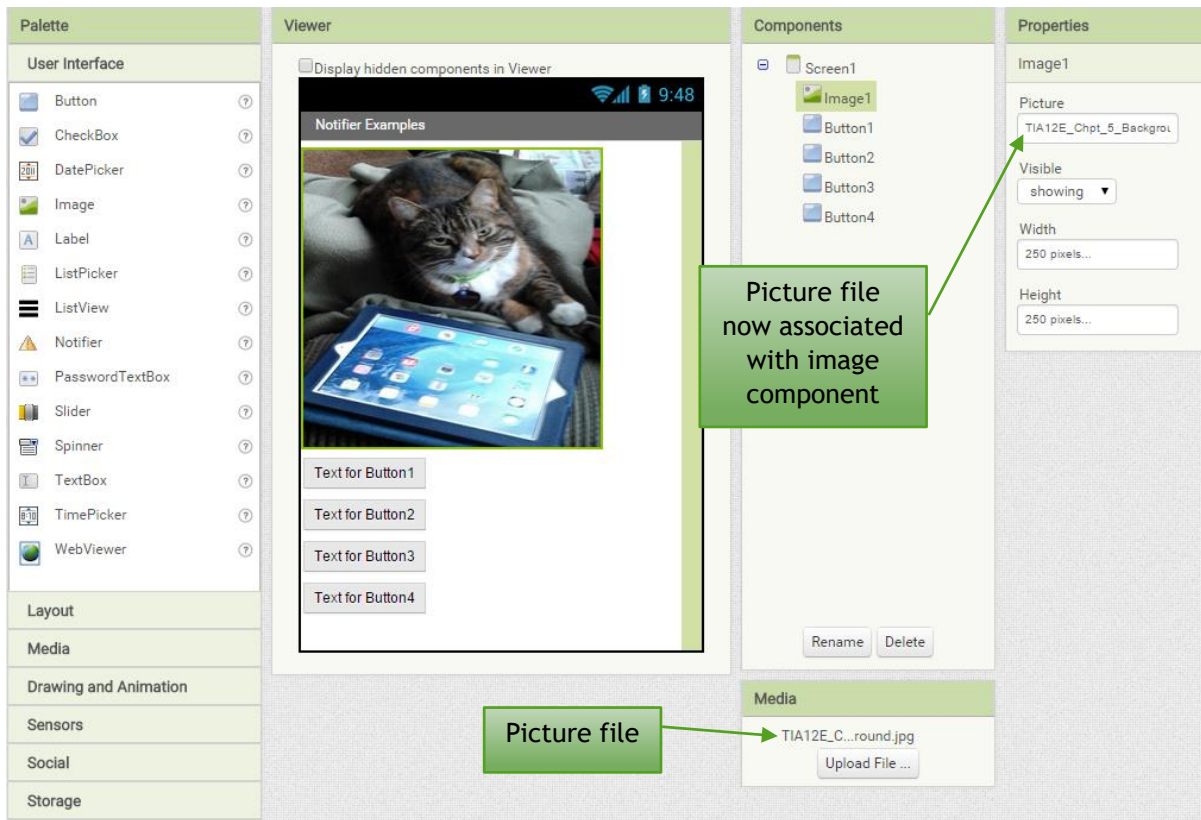


4. In the **Components** window, select **Image1**.

Alert: To associate an image with an image component, it must first be uploaded to App Inventor.

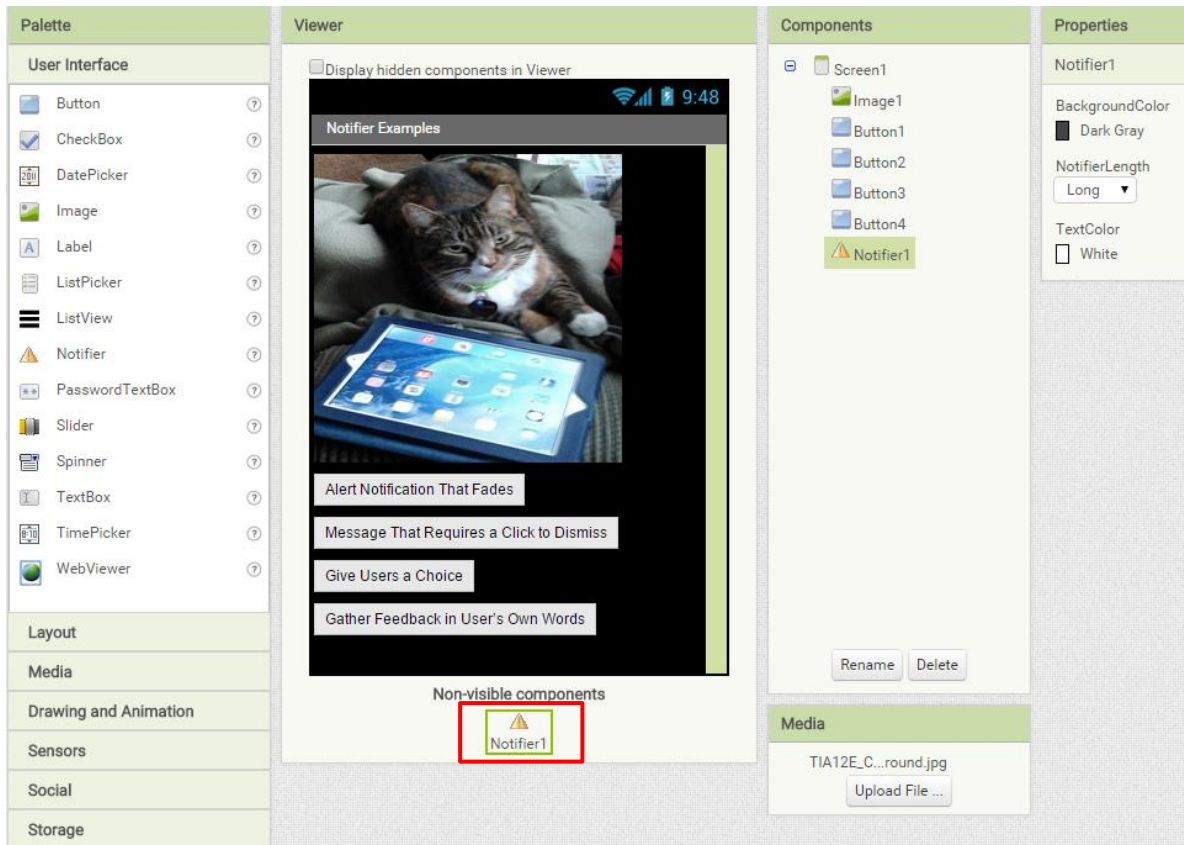
5. In the **Properties** window, click in the **Picture** field to display the dialog box. Click the **Upload File** button. Click the **Choose File** button from the new dialog box that appears. Navigate to the directory where you have the **TIA_Ch5_Background.jpg** file saved and select it. Click **OK**.

Note: Notice that the image is now visible in the **Viewer** on the screen where the image component is located. Notice also that the picture file now appears in the **Media** window and is now available for use in other parts of your app.



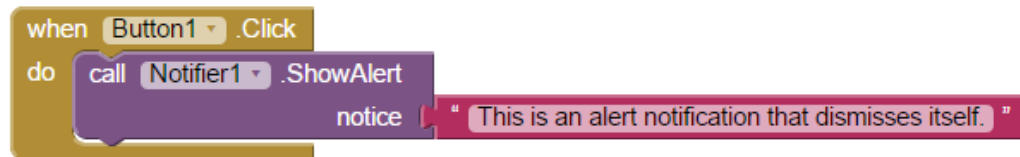
Tip: If the picture is not the size you wanted, you can adjust the size by clicking in the Width and Height fields under properties and either selecting “fill parent” (which should fill your device’s screen) or adjusting the pixel size. Alternately, you could alter the original image, with a program like Paint, to change the size. You would then need to delete the image from App Inventor and upload the revised image file.

6. In the **Properties** window, click **Height** and change the height to *250 pixels*.
7. In the **Components** window, select **Button1**. In the **Properties** window, in the **Text** field, replace the default text with *Alert Notification That Fades*.
8. Select the other buttons one at a time and replace their default text as follows:
 - *Message That Requires a Click to Dismiss*
 - *Give User a Choice*
 - *Gather Feedback in User’s Own Words*
9. In the **Components** window, select **Screen1**. In the **Properties** window, change the **background color** to black (to improve readability).
10. From the **Palette**, under **User Interface**, click and drag a **Notifier** component onto the **Viewer**. Your screen should look like this:





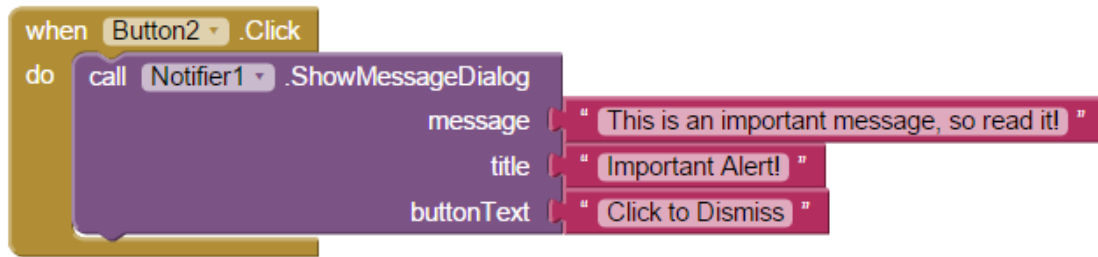
Part 2 - Programming the Buttons for Alerts (with Notifier)



1. Switch to the **Blocks** view.
2. In the **Blocks palette**, select **Button1**, and select the `when Button1 .Click` block from the drawer. Repeat this for Button2, Button3 and Button 4 respectively.
3. In the **Blocks palette**, select **Notifier1**, and select the `call Notifier1 .ShowAlert` block from the drawer. Drag it inside the `when Button1 .Click` block.
4. In the **Blocks palette**, select **Text**, and click and drag a `" "` block onto the **Viewer** and insert it in the open socket on the `call Notifier1 .ShowAlert` block. Insert this text in the `" "` block: *This is an alert notification that dismisses itself.*

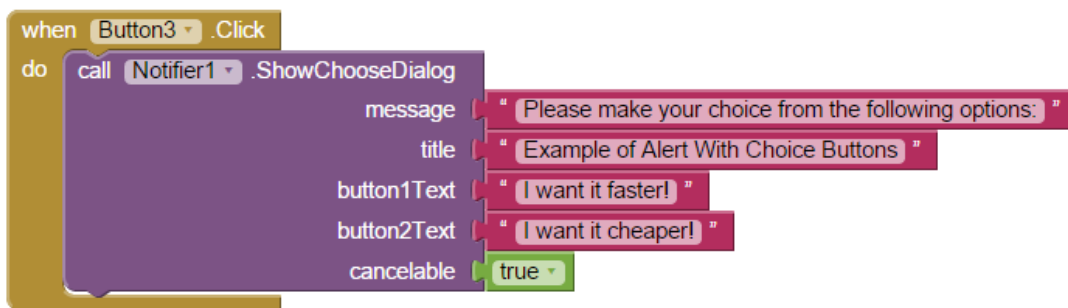


5. In the **Blocks palette**, select **Notifier1**, and select the `call Notifier1 .ShowMessageDialog` block from the drawer. Drag it inside the `when Button2 .Click` block.

6. In the **Blocks palette**, select **Text**, and click and drag three  blocks (or drag out one block and right-click on it to duplicate it) onto the **Viewer** and insert each one in an open socket on the `call Notifier1 .ShowMessageDialog` block.
7. Starting with the first  block, insert text in each one as follows:
 - *This is an important message, so read it!*
 - *Important Alert!*
 - *Click to Dismiss*

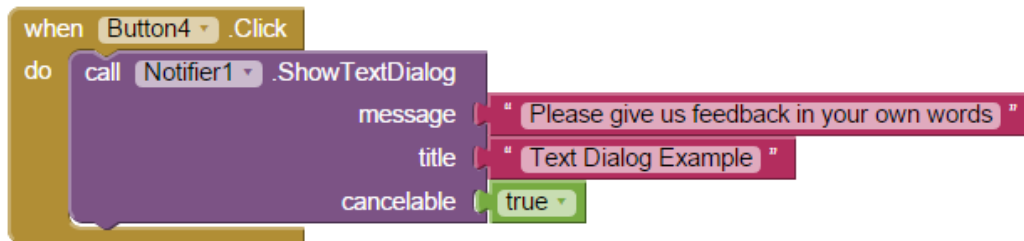


8. In the **Blocks palette**, select **Notifier1**, and select the `call Notifier1 .ShowChooseDialog` block from the drawer. Drag it inside the `when Button3 .Click` block.
9. In the **Blocks palette**, select **Text**, and click and drag four  blocks onto the **Viewer** and insert each one in an open socket on the `call Notifier1 .ShowChooseDialog` block.
10. Starting with the first  block, insert text in each one as follows:
 - *Please make your choice from the following options:*
 - *Example of Alert With Choice Buttons*
 - *I want it faster!*
 - *I want it cheaper!*

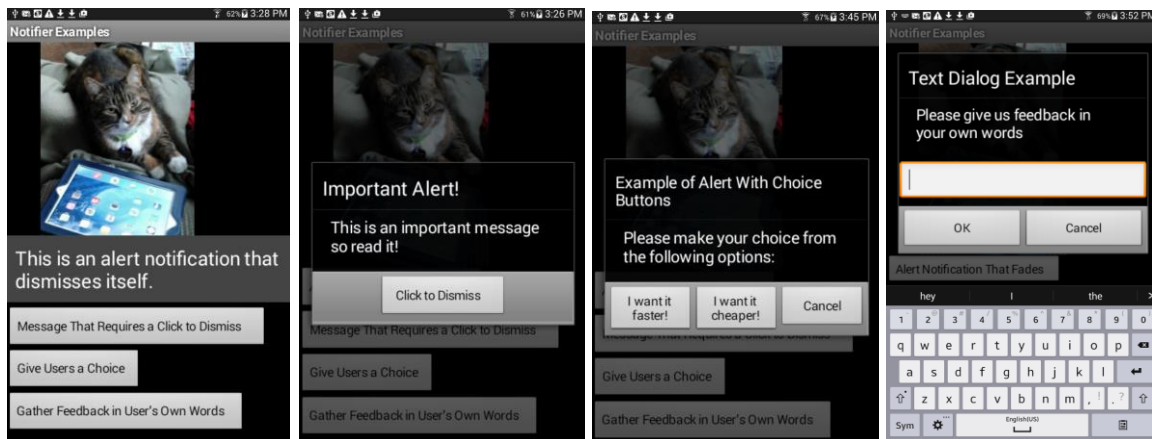


Tip: Notice that the last socket is filled in (by default) with a `true` logic block. If the parameter *cancelable* is true, then a cancel button will be displayed with the two choice buttons. If you don't want the cancel button to appear (the user is forced to choose one of the two buttons), change the *cancelable* parameter to false.

11. In the **Blocks palette**, select **Notifier1**, and select the `call Notifier1 .ShowTextDialog` block from the drawer. Drag it inside the `when Button4 .Click` block.
12. In the **Blocks palette**, select **Text**, and click and drag two `" "` blocks onto the **Viewer** and insert each one in an open socket on the `call Notifier1 .ShowTextDialog` block.
13. Starting with the first `" "` block, insert text in each one as follows:
 - *Please give us feedback in your own words*
 - *Text Dialog Example*



Now test each button to view the examples of each type of alert (notification). They should look like the images below:



Now that you've worked through the project once, go back and modify the components to better suit your own preferences, or try one of the ***Extensions to This Project*** below.

Extensions to This Project

1. Change the text color of a portion of the Notifier text to highlight it. (Hint: Aside from putting plain text in a `" "` block, you can use HTML code.)
2. Modify the app so after Button3 is pressed and the user makes a choice, the value of the choice is stored in a label component.
3. Using the `call Notifier1 .ShowTextDialog` block, modify the app so that the user is prompted to enter a specific password to continue.

Resources

- [User Interface Components: Notifier](#)
- [App Inventor Code Snippets](#)
- [User Guide for App Inventor 2](#)
- [Guide to Understanding Blocks](#)

MIT App Inventor is a blocks-based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices. Google's Mark Friedman and MIT Professor Hal Abelson co-led the development of App Inventor while Hal was on sabbatical at Google. App Inventor runs as a Web service administered by staff at MIT's Center for Mobile Learning - a collaboration of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) and the MIT Media Lab. MIT App Inventor supports a worldwide community of nearly 3 million users representing 195 countries worldwide. App Inventor is an open-source tool that seeks to make both programming and app creation accessible to a wide range of audiences. App Inventor is the property of the Massachusetts Institute of Technology (MIT) and the work licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). For more information on App Inventor, go to [MIT App Inventor About Us page](#).