

Make This

Chapter 4 - Using Activity Starter to Start an App

In this tutorial, you'll learn about the App Inventor **activity starter** component, which gives you the capability to combine applications by having one app start up another app. This means you can take advantage of apps developed by others (such as Google Maps) and include them as features launched from within your app. This tutorial teaches the following skills:

- **Launching Google Maps with the activity starter**
- **Passing a place name (entered by the user) to Google Maps to generate a search**
- **Updating a label**

How Does Activity Starter Help Me Create More Powerful Apps?

Think about the apps that you use. There are many “features” that are actually other apps. For instance, suppose you are checking out the app for a venue where your favorite band is playing. Since you've never been to this venue before, you'd like to see where it is and obtain directions. So you click the button called Get Directions within the app and suddenly you are in the Google Maps app. The designers of the app called another app (Google Maps) from within their app.

The way to launch apps from within an app you develop in App Inventor is by using the Activity Starter. Think about how much more powerful your apps can become if you can launch apps already on an Android device such as Camera or Maps...or for that matter, any app at all that is installed on the device!

With proper design, you can have your app pass values to an app when launching it (which you will do in this tutorial). You can also launch another application and retrieve information from it that you can use in your app, assuming that the other application is designed to return information. (Currently, Activity Starter can only pass and receive text values...so don't get too creative!)

To start another app with Activity Starter, you must communicate control information to the Android operating system. This is accomplished by configuring various properties of Activity Starter using blocks. For a detailed explanation on how to obtain this information for various apps and for block designs, check out these links:

- [Using the Activity Starter](#)
- [App Inventor Code Snippets](#)

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 Where in the World App

This app features a very simple layout. All you need is a button (to launch an outside app, like Google Maps), a text box (for the user to enter the location) and the activity starter component (non-visible). You will then extend the app by storing the value of the place searched in a label field.

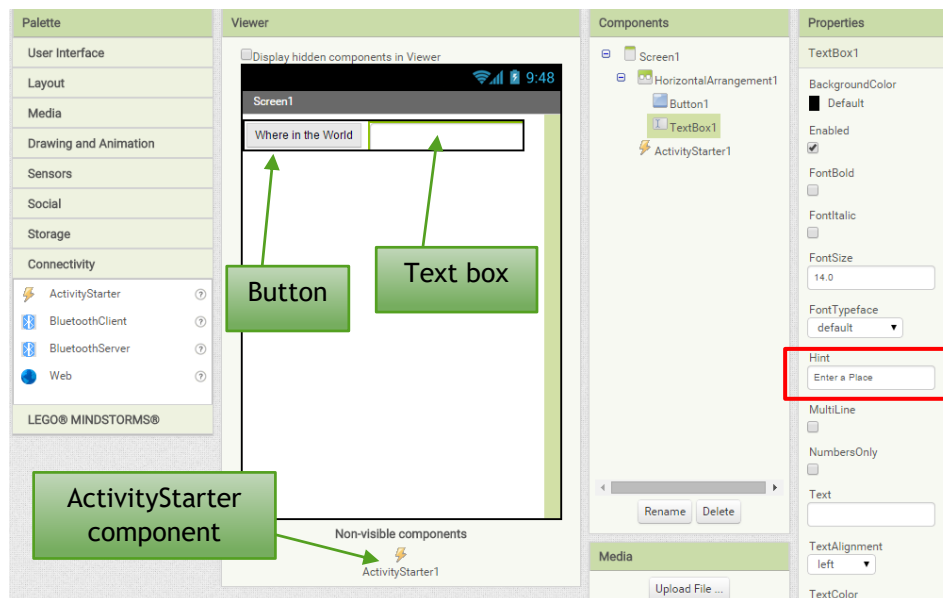
1. Navigate to <http://appinventor.mit.edu/explore/>. If necessary, sign in with your Google Account.
2. Start a new project named *WhereintheWorld*.
3. Connect App Inventor to your Android device.

Part 1 - Launching Google Maps

1. From the **Palette** in the **Layout** menu, drag the **HorizontalArrangement** into the **Viewer** window. Then, from the **User Interface** menu, drag the **Button** and the **TextBox** inside the HorizontalArrangement component.
2. In the **Components** window, select **Button1**. In the **Properties** window, under **Text**, replace the text shown with *Where in the World*.
3. In the **Components** window, select **TextBox1**. In the **Properties** window, under **Hint**, replace the text shown with *Enter a Place*.

Tip: Any text entered in the Hint field will show up in the TextBox when there is no data present in the TextBox. Hints help users understand how to interact with the app.

4. From the **Palette** in the **Connectivity** menu, click and drag the **ActivityStarter** component into the **Viewer** window.



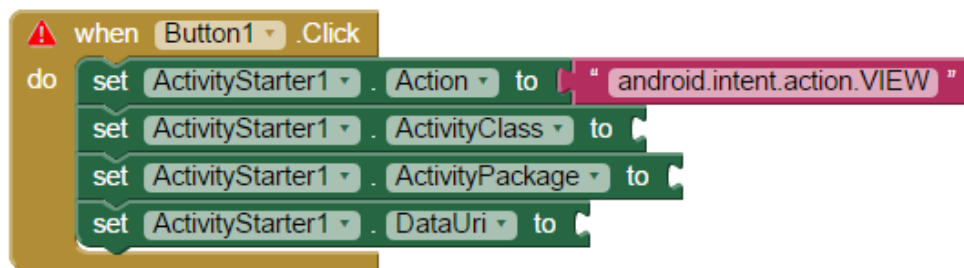
Note: The ActivityStarter component is a non-visible component. Therefore, it appears at the bottom of the Viewer window.





5. Switch to the **Blocks** view.
6. In the **Blocks palette**, select **Button1**, and select the **when Button1 .Click** block from the drawer.
7. In the **Blocks palette**, select **ActivityStarter1**, and select the **set ActivityStarter1 . Action** block from the drawer. Drag it inside the **when Button1 .Click** block.
8. In the **Blocks palette**, select **Text**, and click and drag a **" "** block onto the **Viewer** and insert it in the open socket on the **set ActivityStarter1 . Action** block. Insert this text in the **" "** block: *android.intent.action.VIEW*




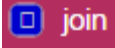

Note: Intent is a programming class in the Android programming language. An intent describes an action that is going to be performed. In this case, the *action.VIEW* piece of the code indicates that you want to see an app you are going to launch with the ActivityStarter.

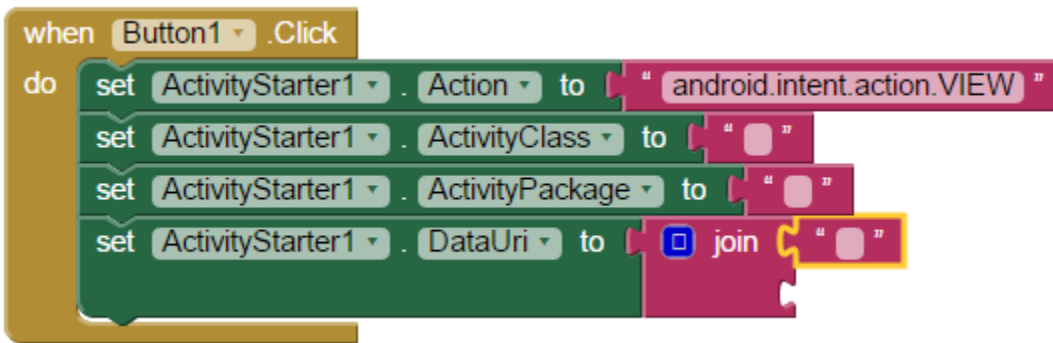
9. Drag out three more **set ActivityStarter1 . Action** blocks and insert them in the **when Button1 .Click** block underneath the first **set ActivityStarter1 . Action** block.
10. In the second **set ActivityStarter1 . Action** block, click the down arrow next to **Action** and from the dropdown list displayed choose **ActivityClass**. For the third and fourth blocks, select **ActivityPackage** and **DataUri**, respectively. The blocks should now look like this:




Alert: The red warning triangle  appears in a block when there may be something wrong with your block. Click the  to display a warning message for guidance. For instance, in the block above, the  appears because there are empty sockets in some of the blocks. Click the  again to dismiss the message.

Drag out two more  blocks and insert them in the open sockets next to ActivityClass and ActivityPackage.

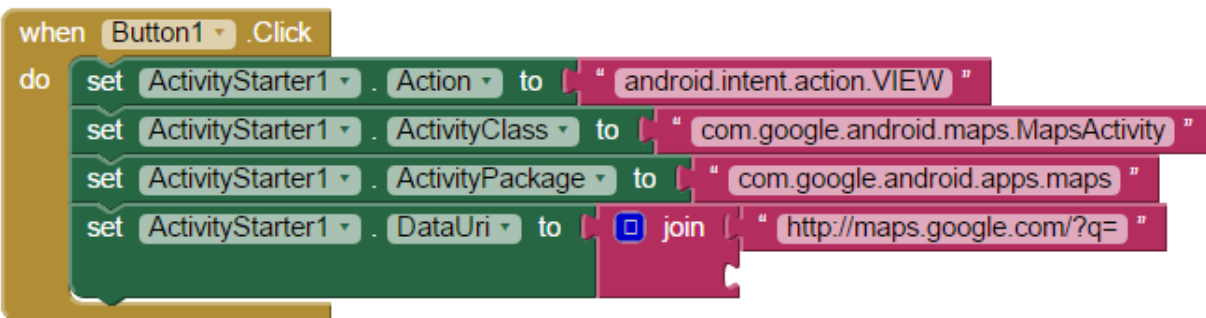
11. From the **Blocks palette**, select **Text**, then drag out a  block and insert it in the DataUri open socket. Drag out one more  blocks and insert it in the first open socket on the join block. Your blocks should look like this:



12. Fill in the  blocks from top to bottom with the following text:

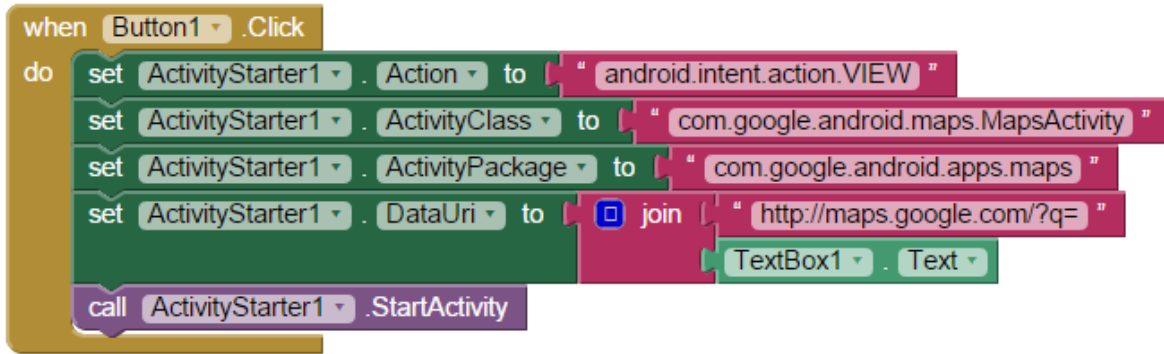
- com.google.android.maps.MapActivity
- com.google.android.apps.maps
- http://maps.google.com/?q=

Your blocks should look like this:



Tip: To launch an app with the activity starter, you must know the appropriate ActivityClass and ActivityPackage to use. At [Using the Activity Starter \(App Inventor 2\)](#) there are listings for the classes and packages of common apps as well as instructions on how to determine these for other apps.

13. In the **Blocks palette**, select **TextBox1**, then drag a **TextBox1 . Text** block from the drawer and insert it into the open socket in the DataUri join block.
14. In the Blocks palette, select **ActivityStarter1**, then drag a **call ActivityStarter1 . StartActivity** block from the drawer and insert it at the bottom of the **when Button1 . Click** block. The completed block should look like this:



Note: DataUri is text that is to be passed to the activity being launched (in this case Google Maps). For Google Maps to display the location entered by the user, our app must pass a URL to Google Maps. Notice this is very similar to the URL and text string manipulation you did in the Chapter 3 exercise.

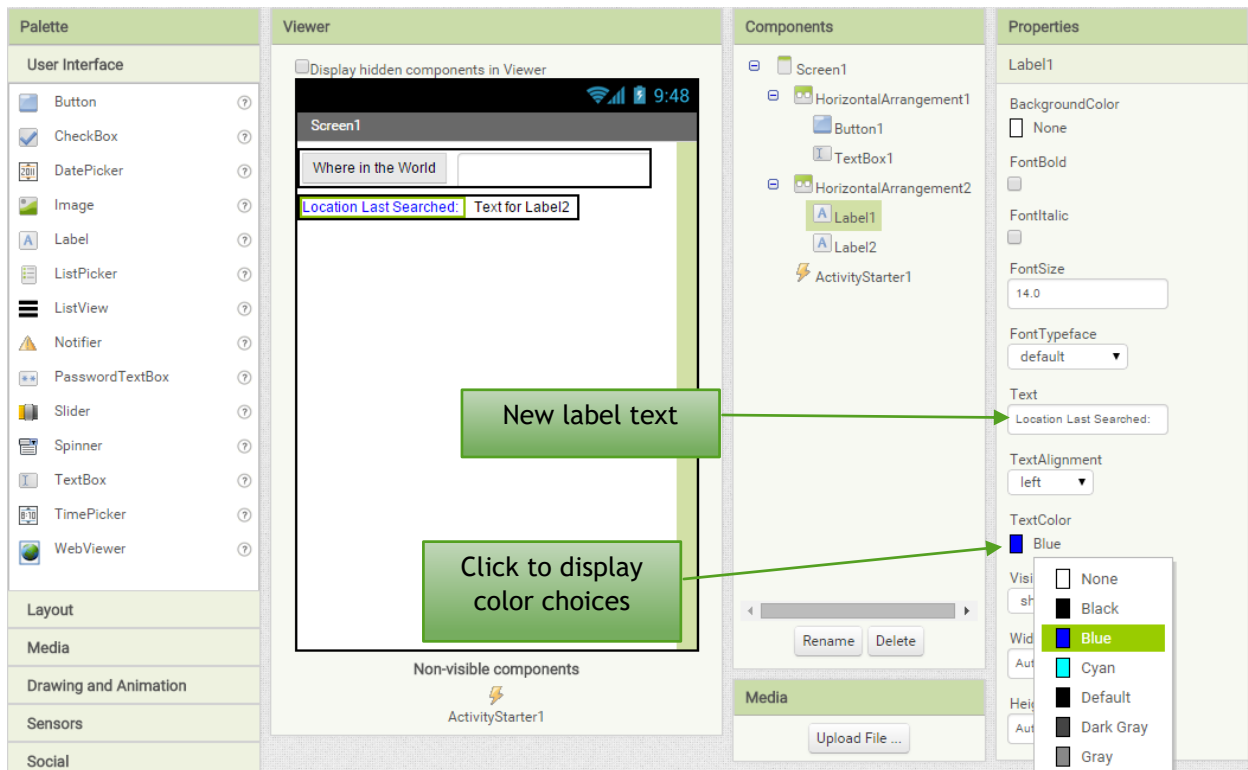
Now test out your app by entering a place, clicking the Where in the World button, and ensuring that Google Maps launches and shows you the location that you entered.

Alert: At the time of publication, this app will not work properly in the emulator. You will receive error messages or be stuck in an endless loop.

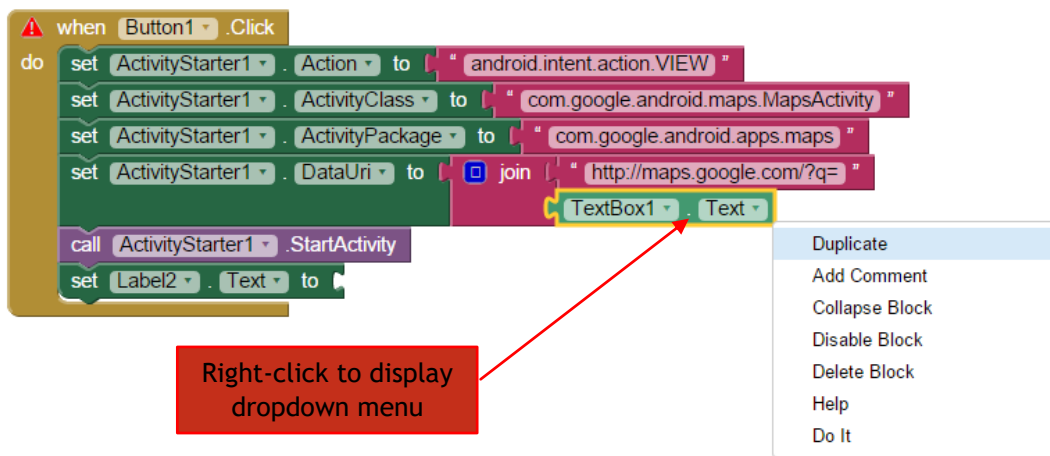
Part 2 - Using Labels

It is often useful to update fields in apps based on user actions that have taken place. An easy way to accomplish this in App Inventor is to use labels. Labels are components that are used to display text. In this app, it might be useful for the user to know what location was previously displayed using Google Maps.

15. Switch to the **Designer** view.
16. In the **Palette**, under **Layout**, click and drag a second **HorizontalArrangement** component on the **Viewer** beneath the existing one.
17. In the **Palette**, under **User Interface**, click and drag two **labels** and place them both in the **Horizontal Arrangement** component you just added.
18. In the **Components** window, select **Label1**. Then in the **Properties** window under the **Text** field, insert the following text: *Location Last Searched:*
19. In the **Properties** window under the **TextColor** field, click the **Black square** and select **Blue** from the dropdown list.

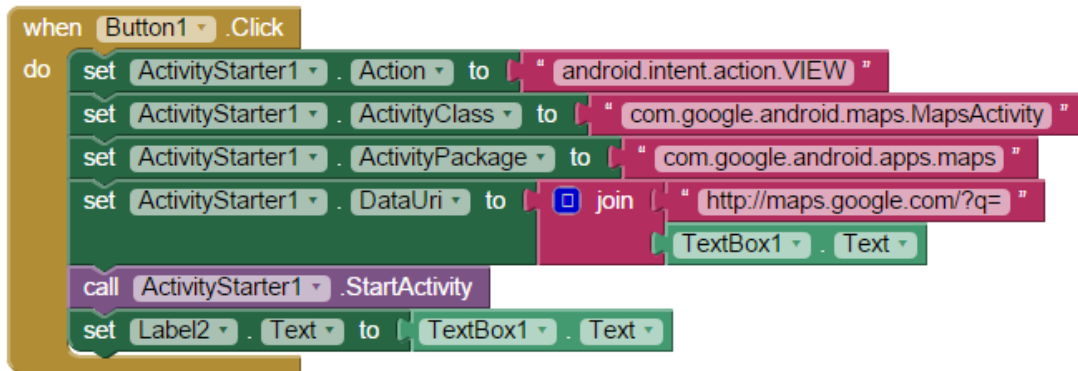


20. In the **Components** window, select **Label2**. Then in the **Properties** window under the **Text** field, delete the default text shown. Then change the **TextColor** to **Red**.
21. Switch to the **Blocks** view.
22. In the **Blocks palette**, select **Label2**, then click and drag a **set Label2 . Text** block onto the Viewer and insert it into the **when Button1 . Click** block underneath the **call ActivityStarter1 . StartActivity** block.
23. Right-click on the **TextBox1 . Text** block in the viewer and select **Duplicate** from the dropdown menu to create another **TextBox1 . Text** block.



24. Select the new block and plug it into the empty socket on the **set Label2 . Text** block.

Your finished block should look like this:



Now test your app to ensure the Label2 field is updated with the search term used.

Alert: At the time of publication, this app will not work properly in the emulator. You will receive error messages or be stuck in an endless loop.

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. Modify the app so that it will keep track of the last five places searched.
2. Add a button to the app that will allow you to launch the e-mail app and send an e-mail to a default address. Review the section of [Using the Activity Starter](#) entitled *Start the mailer with pre-addressed message* for guidance.
3. Use activity starter to launch a YouTube video that is related to the place entered by the user.

Resources

- [Using the Activity Starter](#)
- [App Inventor Code Snippets](#)
- [User Guide for App Inventor 2](#)
- [Guide to Understanding Blocks](#)
- [MIT App Inventor Support Forum](#)
- [Beginner Tutorials \(with videos\)](#)

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](#).