|  |  |
| --- | --- |
| This tutorial shows how to use the *Location Sensor* in combination with Google Maps to display directions from the user’s current location to a selected destination.    **Objectives:** In this lesson you will learn to :     * create an app that   + uses the Global Positioning System (GPS) to obtain the user's current location,   + uses the Google Maps Application Programming Interface (API) to obtain directions from the user's current location to a destination address; * appreciate that Google Maps API is an abstraction that allows apps to leverage existing functionality; * obtain some additional experience working with lists and TinyDB to save user-generated information persistently. | ***[Click here to view the preview](https://www.youtube.com/watch?v=j1eiu8Dy_SI)*** |

## 

# Getting Ready

To get started, [open App Inventor with the My Directions Template](http://ai2.appinventor.mit.edu/?repo=templates.appinventor.mit.edu/trincoll/csp/unit7/templates/MyDirections/MyDirectionsTemplate.asc)  in a separate tab and follow along with these tutorials.

# 

# 

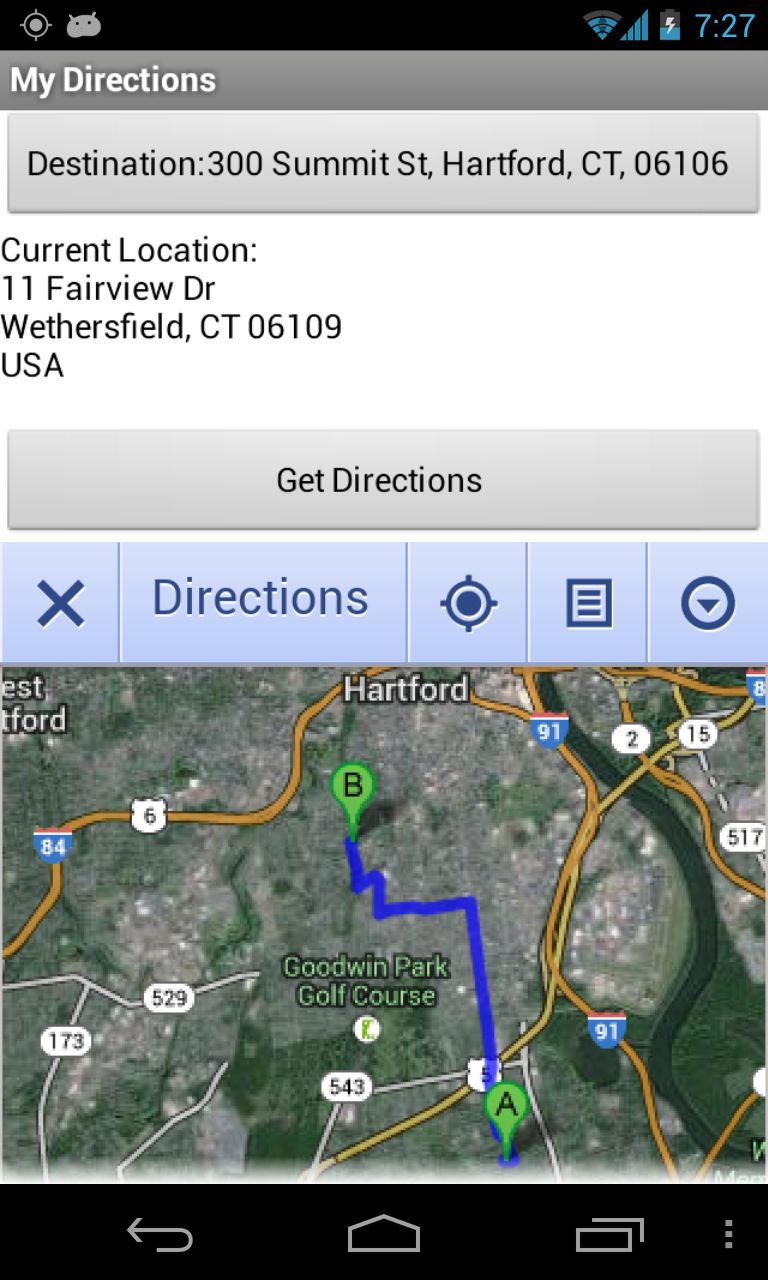
# 

## 

## Basic Idea

The basic idea for this app is to simplify getting directions from one’s current location to a destination that can be selected from a list picker. For example, the app might be populated with the user’s *HOME* and *WORK* addresses. The user would select a destination, click the *‘Get Directions’* button and the app will display a Google Map with the directions.

## My Directions UI

**Setting up the UI**

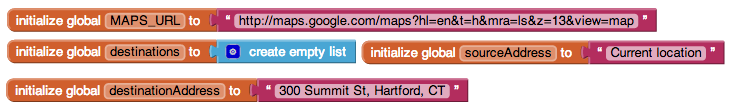
1. The template contains a *ListPicker* and is pre-populated with a list of addresses, such as one’s home and work addresses.
2. Below it is a *Label* that is used to display the user’s current location, which is gotten from the *LocationSensor.*
3. When clicked, the *GetDirections* button will bring up a Google map with the directions from the current address to the selected destination.
4. The map is displayed in a *WebViewer* component. When you use a WebViewer it is best to uncheck the Screen’s *Scrollable* property or the page will grow downward indefinitely.
5. A *Web* component is used for encoding URLs.
6. ***Non-visible Component:*** A *LocationSensor* is used to get the user’s current address.

## The Blocks

### Global Variables

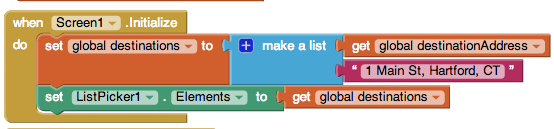
The app uses several global variables:

* *MAPS\_URL* is the root of the Google Maps URL. When the Webviewer calls up the map it will concatenate additional arguments to this string.
* *destinations* is a list of destination addresses. You’ll set these in *Screen.Initialize*. An enhancement might be to allow the user to add destinations that are stored in a *TinyDb* component.
* *destinationAddress* stores a destination address when the user selects one from the list picker.
* *sourceAddress* is the user’s current location, set by the *LocationSensor* whenever the location changes.



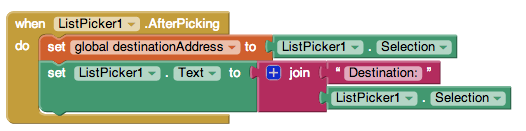
### Initializing the Screen

When the app starts up, the following initializations are performed:

* The *global destinations* list is initialized with some addresses.
* The *ListPicker* is populated with the list of destinations. 

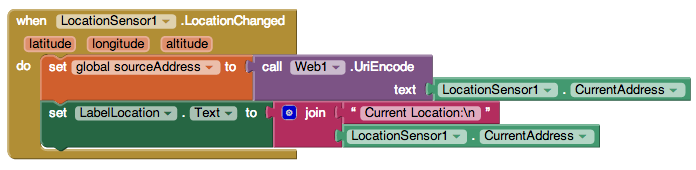
### Picking the Destination

The *ListPicker* is used to select a destination. The selected destination is stored in the *global destinationAddress* variable and is displayed as the *ListPicker’s Text*.



### Updating the Location

Whenever the location changes, the app needs to remember the *CurrentAddress* in its global *sourceAddress* variable and it needs to display the current address in *LabelLocation.*



The *Web.UriEncode* block is used to encode the current street address (LocationSensor1.CurrentAddress) in a form suitable for a Maps URL. In this app, the sole reason for using a Web component is for this encoding (you can also use it to get data from web APIs, but we don’t do that here).

### 

### Displaying the Directions

To display the directions in the *WebViewer* it is necessary to ***construct a URL*** using the information available to the app. The completed URL should look something like this, where the *arguments* have been highlighted:

‘<https://maps.google.com/maps?saddr=330+Summit+Street+Hartford+CT&daddr=11+Fairview+Dr+Wethersfield+CT&hl=en&t=h&mra=ls&z=13>’

The arguments are part of the Google Maps ***Application Programming Interface (API)*,** an ***abstraction*** that lets users and programmers tell Google Maps what to search for and how to display the results:

* saddr -- source address, can also be a lat, long pair.
* daddr -- destination address, can also be a lat, long pair.
* z = zoom level.
* t = type of map (h = hybrid)
* hl = language (en = English)
* mra = ??

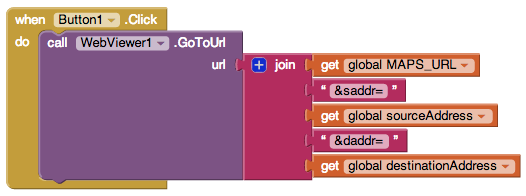
Click here to see a complete list of the [Maps API arguments](https://docs.google.com/document/d/1Rm4aXQRFBQf0i0jWHHqpkvhXrlm8GLcSJJbBqkXJmjA).

### Constructing the Map’s URL

We use the *Text.join* function to put together the different parts of the Map’s URL. The *MAPS\_URL* forms the base of the URL. It contains those elements that won’t vary from search to search:



The rest of the URL is constructed using the variables that are storing the important information and it is then passed to the *WebViewer* when the Button is clicked.



# 

# 

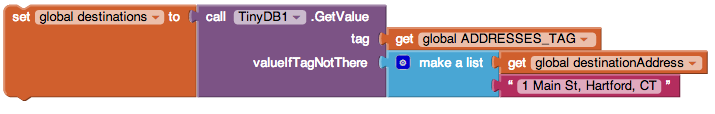
# 

# Enhancements

This completes the walkthrough of the *MyDirections* app. There are a number of enhancements you could implement to improve this app.

1. In its current form, the destination addresses are created by the *programmer*. Add a *Textbox* and a *Button* to the UI to enable the *user* to input their own destination addresses. Addresses entered by the user should be added to the *destinations* list. This will make it possible to share the app with friends.
2. Improve the presentation of the search results by using or modifying one or more additional API arguments in the Map’s URL.
3. **Advanced:** Add a *TinyDb* to the app so that the user’s destinations will persist. Addresses saved in the TinyDb will be there the next time the app is used. HINT: Lists (as well as numbers and strings) can be stored in a TinyDb. So you can store the entire *destinations* list as one element. Determine when and how to store the destinations. Then, determine when and how to retrieve the destinations.

HINT: The destinations should be retrieved from the TinyDb when the app is initialized. Here’s how:



NOTE: This can be tricky to understand. The first time the app is run, there definitely won’t be any ‘addresses’ stored in the *TinyDb*. You can specify how to handle the empty data situation with the *valueIfTagNotThere* parameter. In the example, we set global destinations to a default fixed list so the first time the app runs it will have some sample destinations You could also put a create empty list in valueIfTagNotThere if you wanted the destinations list empty to start.This problem won’t arise once the user has stored some addresses in the TinyDb.

You can also use this app as a starting point for your second programming project, which involves creating a location aware app.

***Nice work! Complete the Self-Check Exercises and Portfolio Reflection Questions as directed by your instructor.***