# Instructions for working on your web site

This assumes you currently have a working web site (called ‘myapp’ located under csi185website) We did some of this in class. Those of you who were not there Monday will need to do all of it. Those of you who were in class and succeeded will be able to skip the first couple of tasks.

If you don’t have the csi185website directory or the functional tests do not work, do the following.

1. In a terminal, type “git clone https://github.com/mstoth/csi185website.git”
2. Verify you can run the functional tests.
   1. In a terminal, go to the directory called csi185website (You should see it in the directory you typed the clone command)
   2. Run the server by typing “python3 manage.py runserver” and minimize the terminal so it’s out of your way.
   3. In another terminal go to the same directory and run your functional tests by typing “python3 functional\_tests.py” [[1]](#footnote-1)

At this point, you have a working web site. It’s called “myapp” and it’s located under csi185website.

If you now run a browser and type in the URL “localhost:8000/index.html” you should see a computer. When you click on the keyboard, you should be taken to another page showing a teletype. Go ahead and try it. Depending on your browser settings, you may or may not see the title of the web site, “The Title of my Web Site”

## Set up your environment

I found it easiest if I do the following. Start the terminal to run the server, cd to the csi185website directory, run the server, then minimize the window. Start another terminal and cd into the csi185website directory and use this window to run your functional tests. Run Sublime and load the index.html file and keep that window accessible. Run Idle and load the functional\_tests.py file. Minimize the Idle shell. Don’t try to run the functional test file in Idle, we use the terminal to do that. Just use the Idle editor to make changes in functional\_tests.py.

So you should now have 3 windows that you move between as you do your work; the terminal to run functional tests, the Sublime window to edit your HTML files, and the Idle editor to edit your functional tests. When you modify your functional tests, you will be using Idle, when you run your tests you will use the terminal, and when you modify your html files you will use Sublime.

## Making your first failure

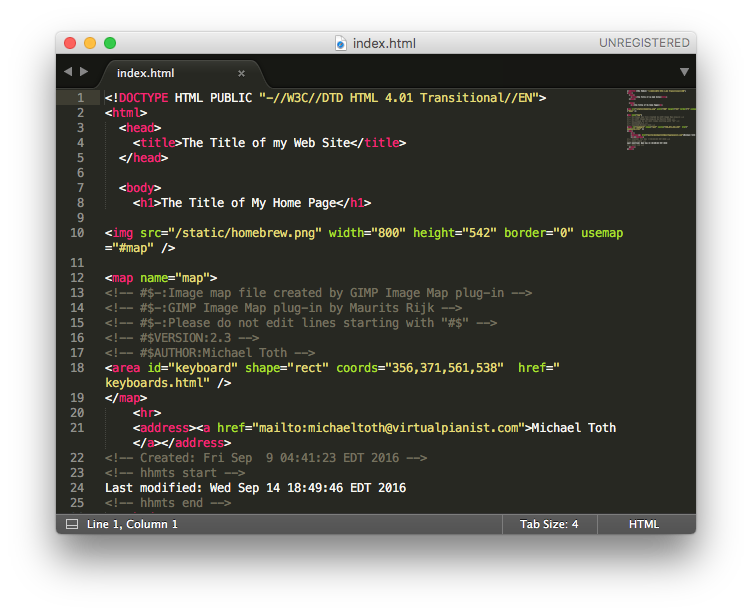
Now we will create our first failure. Using Idle, edit the functional\_tests.py file and change the title of your web site to something you want. Change the following line,

self.assertIn('The Title of my Web Site',self.browser.title)

so it has the title you want (we did this in class but if you weren’t there, you need to do it.)

Once you have done that, re-run your functional tests and verify that you have a failure. [[2]](#footnote-2)

Now we will fix this failure by editing index.html. It’s located under the templates directory under myapp. Use Sublime to edit it.



Change the text between <title> and </title> to match your title you put in functional\_tests.py, save it, then re-run your functional tests. You should now have a passing test again. *Talk to me if you don’t.*

Now the next failure will be the heading. Change the following line in functional\_tests.py

self.assertIn("The Title of My Home Page",h.text)

Change “The Title of My Home Page” to be what you want. Then save it and re-run your functional tests.[[3]](#footnote-3)

Now change index.html to match.

<h1>The Title of My Home Page</h1>

becomes

<h1>The Beginning</h1>

in my case. We are now back to GREEN. This was my goal last Monday.

## Changing the Image

The image is defined by an img tag.[[4]](#footnote-4) In the index.html file, it appears like this.

<img src="/static/homebrew.png" width="800" height="542" border="0" usemap="#map" />

All your images will be stored in the static directory (under myapp) and they will be referenced like ‘/static/<filename>’ as in the example, src="/static/homebrew.png". This is why the HTML class under Lynda is a good thing to do. It explains all this stuff.

After you have decided which image to show on your home page, change your functional tests.

m=self.browser.find\_element\_by\_tag\_name('img')

self.assertIn('homebrew.png',m.get\_attribute('src'))

Change homebrew.jpg to the name of your image file you want to use.

Rerun your functional test and verify you get an error. [[5]](#footnote-5)

Now copy your image file into the static directory and change your index.html file to have the correct image name.

You should now be back in the GREEN condition and when you open localhost:8000/index.html in your browser you should see your picture.

Try it out.

Now that you have changed the image, your mapping is no longer valid. Delete the following lines from your index.html file.

<map name="map">

<!-- #$-:Image map file created by GIMP Image Map plug-in -->

<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->

<!-- #$-:Please do not edit lines starting with "#$" -->

<!-- #$VERSION:2.3 -->

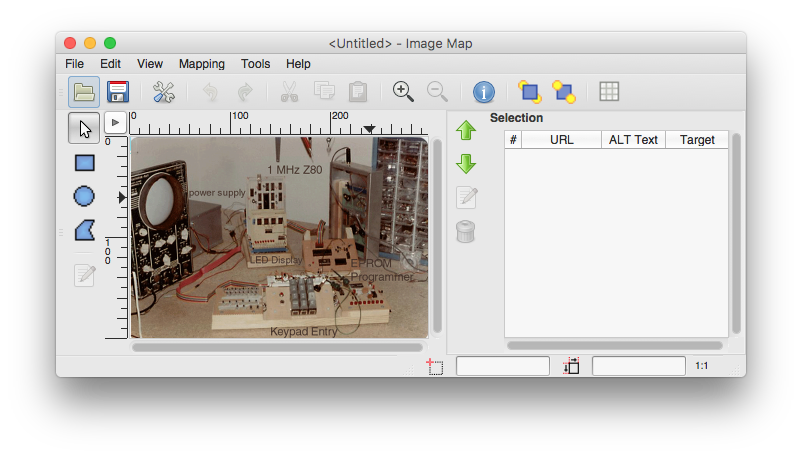
<!-- #$AUTHOR:Michael Toth -->

<area id="keyboard" shape="rect" coords="356,371,561,538" href="keyboards.html" />

</map>

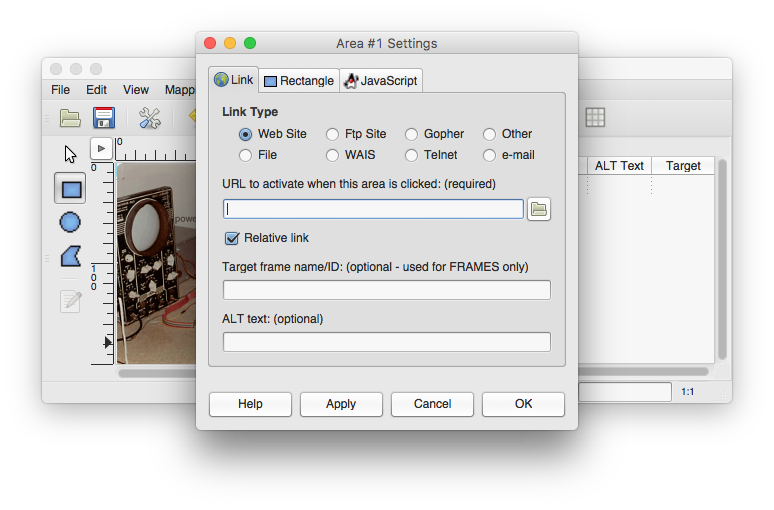
## Mapping an Area of the Picture Using Gimp

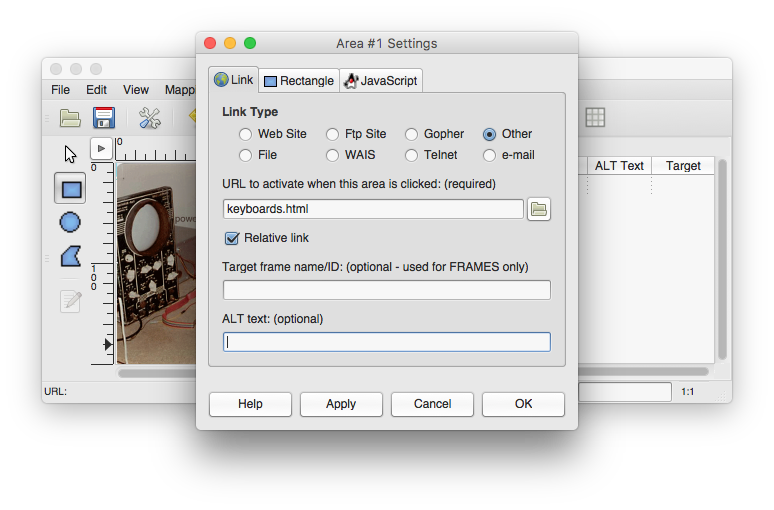
Run Gimp and open your image. Select Filters->Web->Image Map… in the Gimp menus. You should see a new window appear like this.



With your picture, not mine of course. Pick a shape on the left and draw the shape on the image where you want it to react to a mouse click.

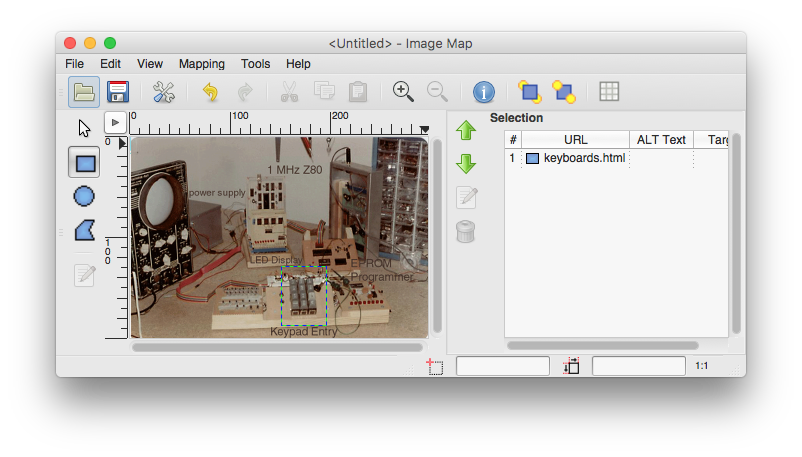
You should see another window appear like this.



Type in the URL you want to go to when you click on the area. This will be a new page on your web site. You can name it anything you want. Just put the name of the file here, nothing in front of it. 

In my case the file is called keyboards.html and my area is around the keyboard of the homebrew computer.

Click Apply then OK.



You should see your area on the right hand side of the window.

Select File->Save and navigate to your templates directory. Save the file as <something>.map where <something> means something to you. No spaces. You now have a map file in your templates directory. In my case it looks like this.

<img src="[homebrew2] (imported)" width="300" height="203" border="0" usemap="#map" />

<map name="map">

<!-- #$-:Image map file created by GIMP Image Map plug-in -->

<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->

<!-- #$-:Please do not edit lines starting with "#$" -->

<!-- #$VERSION:2.3 -->

<!-- #$AUTHOR:Michael Toth -->

<area shape="rect" coords="151,129,196,188" href="keyboards.html" />

</map>

Your file will be different of course but the format should be the same. This gives us a map of your image and you refer to this map in the <img> tag. It also includes an <img> tag as well.

## Defining more than one area with Gimp

When you have more than one area you would like to map, you just draw as many areas in Gimp as you like. Below is the result for 3 areas of my image corresponding to the display, processor, and keyboard.

<img src="[homebrew] (overwritten)" width="500" height="338" border="0" usemap="#map" />

<map name="map">

<!-- #$-:Image map file created by GIMP Image Map plug-in -->

<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->

<!-- #$-:Please do not edit lines starting with "#$" -->

<!-- #$VERSION:2.3 -->

<!-- #$AUTHOR:Toth -->

<area shape="rect" coords="260,223,313,303" href="keyboards.html" />

<area shape="rect" coords="203,131,267,187" href="displays.html" />

<area shape="rect" coords="214,88,241,122" href="processors.html" />

</map>

When we write out tests we will want to click on these areas so the easiest way to do that is to add an ‘id’ to them. Below is an example with the image source corrected and the areas given ids.

<img src="/static/homebrew.png" width="500" height="338" border="0" usemap="#map" />

<map name="map">

<!-- #$-:Image map file created by GIMP Image Map plug-in -->

<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->

<!-- #$-:Please do not edit lines starting with "#$" -->

<!-- #$VERSION:2.3 -->

<!-- #$AUTHOR:Toth -->

<area id=”keyboards” shape="rect" coords="260,223,313,303" href="keyboards.html" />

<area id=”displays” shape="rect" coords="203,131,267,187" href="displays.html" />

<area id=”processors” shape="rect" coords="214,88,241,122" href="processors.html" />

</map>

You will need to edit your .map file accordingly so you can write your tests.

## Testing for the image area map

There is already one area defined with an id of ‘keyboard’ in my example. We test for it with the following line in functional\_tests.py

a=self.browser.find\_element\_by\_id('keyboard')

If the area is not there, you will get an error. Since you have created a map file using Gimp and added your ids for each area as above, you know what to look for. Change the Python statement above to use the id you chose.

At this point, you should have your image but no map in your index.html file. You should have a map file that you modified as above and saved in the templates directory, but have not yet incorporated it into your index.html file. Since we deleted the map in the index.html file, the element with the id is no longer there and we get an error.

My error looks like this.

… omitted text …

raise exception\_class(message, screen, stacktrace) selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate element: {"method":"id","selector":"keyboards"}

… omitted text …

We fix this by inserting the text for the image and map that we created with Gimp.

My index.html file now looks like this.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>The Title of my Web Site</title>

</head>

<body>

<h1>The Title of My Home Page</h1>

<img src="/static/homebrew.png" width="500" height="338" border="0" usemap="#map" />

<map name="map">

<!-- #$-:Image map file created by GIMP Image Map plug-in -->

<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->

<!-- #$-:Please do not edit lines starting with "#$" -->

<!-- #$VERSION:2.3 -->

<!-- #$AUTHOR:Toth -->

<area id=”keyboards” shape="rect" coords="260,223,313,303" href="keyboards.html" />

<area id=”displays” shape="rect" coords="203,131,267,187" href="displays.html" />

<area id=”processors” shape="rect" coords="214,88,241,122" href="processors.html" />

</map>

<hr>

<address><a href="mailto:michaeltoth@virtualpianist.com">Michael Toth</a></address>

<!-- Created: Fri Sep 9 04:41:23 EDT 2016 -->

<!-- hhmts start -->

Last modified: Wed Sep 14 18:49:46 EDT 2016

<!-- hhmts end -->

</body>

</html>

You want to copy this into your index.html file and replace “[homebrew2] (imported)” with the name of your image file “/static/homebrew2.png” (replace my file name with yours of course)

If you find that your image is too small, you can resize it before creating the map using Gimp. Select Image->Scale Image … in the Gimp menu bar.

1. You should see something like this

   hello-mac:csi185website michaeltoth$ python3 functional\_tests.py

   .

   ----------------------------------------------------------------------

   Ran 1 test in 2.581s

   OK

   hello-mac:csi185website michaeltoth$ [↑](#footnote-ref-1)
2. My failure looks like this

   hello-mac:csi185website michaeltoth$ python3 functional\_tests.py

   F

   ======================================================================

   FAIL: test\_home\_page (\_\_main\_\_.NewVisitorTest)

   ----------------------------------------------------------------------

   Traceback (most recent call last):

   File "functional\_tests.py", line 41, in test\_home\_page

   self.assertIn('Computer History',self.browser.title)

   AssertionError: 'Computer History' not found in 'The Title of my Web Site'

   ----------------------------------------------------------------------

   Ran 1 test in 3.322s

   FAILED (failures=1)

   hello-mac:csi185website michaeltoth$ [↑](#footnote-ref-2)
3. My change to self.assertIn("The Beginning",h.text) resulted in this:

   hello-mac:csi185website michaeltoth$ python3 functional\_tests.py

   F

   ======================================================================

   FAIL: test\_home\_page (\_\_main\_\_.NewVisitorTest)

   ----------------------------------------------------------------------

   Traceback (most recent call last):

   File "functional\_tests.py", line 55, in test\_home\_page

   self.assertIn("The Beginning",h.text)

   AssertionError: 'The Beginning' not found in 'The Title of My Home Page'

   ----------------------------------------------------------------------

   Ran 1 test in 2.495s

   FAILED (failures=1)

   hello-mac:csi185website michaeltoth$ [↑](#footnote-ref-3)
4. http://www.w3schools.com/tags/tag\_img.asp [↑](#footnote-ref-4)
5. In my case I get

   hello-mac:csi185website michaeltoth$ python3 functional\_tests.py

   F

   ======================================================================

   FAIL: test\_home\_page (\_\_main\_\_.NewVisitorTest)

   ----------------------------------------------------------------------

   Traceback (most recent call last):

   File "functional\_tests.py", line 48, in test\_home\_page

   self.assertIn('homebrew2.png',m.get\_attribute('src'))

   AssertionError: 'homebrew2.png' not found in 'http://localhost:8000/static/homebrew.png'

   ----------------------------------------------------------------------

   Ran 1 test in 5.566s

   FAILED (failures=1)

   hello-mac:csi185website michaeltoth$ [↑](#footnote-ref-5)