HowTo: Graphing with Pygal

by Elijah Voigt for cs290

Pygal is a Python framework which can be used to programmatically generate graphs (*.svg* images) on the server-side of a web-application. This HowTo will walk you through the steps required to create and embed a graph with *pygal*. With this knowledge you could create an application with this framework to graph data in beautiful, interactive, and customizale ways.

Before We Begin...

This HowTo makes a few assumptions that you should be aware of going forward:

1. We will be using Linux.
2. We will be using Python.

If you are currently using a *nix operating system (Linux, Unix, Mac OSX) then you can follow most of the code examples as they are. If you are using Windows you have two options going forward:

A. Install a Linux Virtual Machine.

B. Do a lot of Googling.

Setting up a virtual machine is very simple, see the **Links** section below for a few options to get a virtual machine setup. I encourage people to use Vagrant for this because it is simple and allows you to get straight to work in your virtual machine.

If you go the route of setting up Pygal on a Windows machine you will have to investigate how to setup and use Pygal on your own. Unfortunately there isn't much documentation on this process and it is outside the scope of this HowTo. That said the python, json, and html code are all the same once you get your libraries and programs installed properly.

As far as python experience goes you should check out the Python website and search online for python learning resources, there are many. Python is very simple as far as programming languages go and is very powerful. If you know C or Java or PHP you won't have trouble picking up Python, I guarentee it.

Getting Started

First make sure you have *python*, *pip* (the python package manager), and *virtualenv* installed on your system.

Once you have done that we will setup a Python Virtual Environment.:

```
# Every command pre-pended by a `$` should be run in your terminal
$ mkdir pygal_testing
$ cd pygal_testing
$ virtualenv pygal
Using base prefix '/usr'
New python executable in pygal/bin/python3
Also creating executable in pygal/bin/python
Installing setuptools, pip...done.
$ source pygal/bin/activate
(pygal)$ pip install pygal
*output*
```

Now you are setup to run Pygal in your python scripts! The rest of these code examples can be copy-pasted into the *pygal_testing* directory as this is where we will be doing all of our work. To get back into your Pygal Virtal Environment run the *source pygal/bin/activate* in this directory.

Data

In order to make graphs you need some data. For our purposes we will be making a histogram of the following data set:

```
{"Chrome": ["0","0","0","0","0","0","0","3.9","10.8","23.8","35.3"],
 "Firefox": ["0","0","0","16.6","25","31","36.4","45.5","46.3","42.8","37.1"],
 "Others": ["14.2","15.4","15.3","8.9","9","10.4","8.9","5.8","6.7","6.8","7.5"]
 }
```

This is just simple json. Python supports many data formats but for our testing purposes we will be using the above data in a file called *data.json*.

Generating Graphs

To generate a graph you will need to run code that looks like the following:

```
#File name: pygal_example.py
import Pygal   # Used to graph our dataset
import json     # Used to load and parse json data

# Open data.json and convert string input to floating point numbers for pygal.
with open('data.json', 'r') as f:
    data = { str(key) : [float(item) for item in value] for key, value in \
            json.load(f).items()}

bar_chart = pygal.Bar()      # Create a Pygalbar graph object
bar_chart.title = "Browser usage evolution (in %)"   # Give the graph a title
bar_chart.x_labels = map(str, range(2002, 2013))     # Give the graph labels
bar_chart.add("Firefox", data["Firefox"])    # Add Firefox data to the graph
bar_chart.add("Chrome", data["Chrome"])      # Add Chrome data to the graph
bar_chart.add("Others", data["Others"])      # Add 'Others' data to the graph
bar_chart.render_to_file("graph.svg")        # Render the graph to a local file
```

The way in which you get/parse your input (in this case it is json, but it could be xml, yaml, etc) will differ depending on your setup. In this example we assume you have a static data.json file.

Let's break down that the above code is doing:

- First this script imports the Pygaland json libraries. These will be used to graph data and parse data respectively.
- Next the script imports data.json, parses the input, and converts strings to integers so Pygalcan actually graph your data.
- The *bar_chart* object is created, this is our PygalBar Chart which we will modify.
- A title and axis are added to the graph.
- Data is added to the graph The format for this is *char_obj.add("Label", data)*
- Finally the graph is saved to *graph.svg*

The above script isn't perfect though! As a challenge I encourage you to re-factor the above code to either:

A. Fetch remote json (from a server)

B. Add an arbitrary number of elements to the data set (replace the explicit bar_char.add("Firefox") with a more extendible solution).

C. Try creating a different graph with the same data set (check the *links* for pygal's site which has descriptions, examples, and code-snippets for each of the graphs that *Pygal* supports).

Using Graphs in your Website

To test the new *svg* file you've created you can open it with a web-browser.

Note: If you are running everything in virtual machine with Vagrant you should have a shared directory between the virtual machine and your computer. Read the documentation to find where this is on the VM and check your Vagrantfile to make sure the shared directory is setup. If not you can upload your *svg* to access.engr.oregonstate.edu or shell.onid.oregonstate.edu and view it via your public_html personal webserver setup.

This is an example of how to embed an svg generated by Pygalin your website:

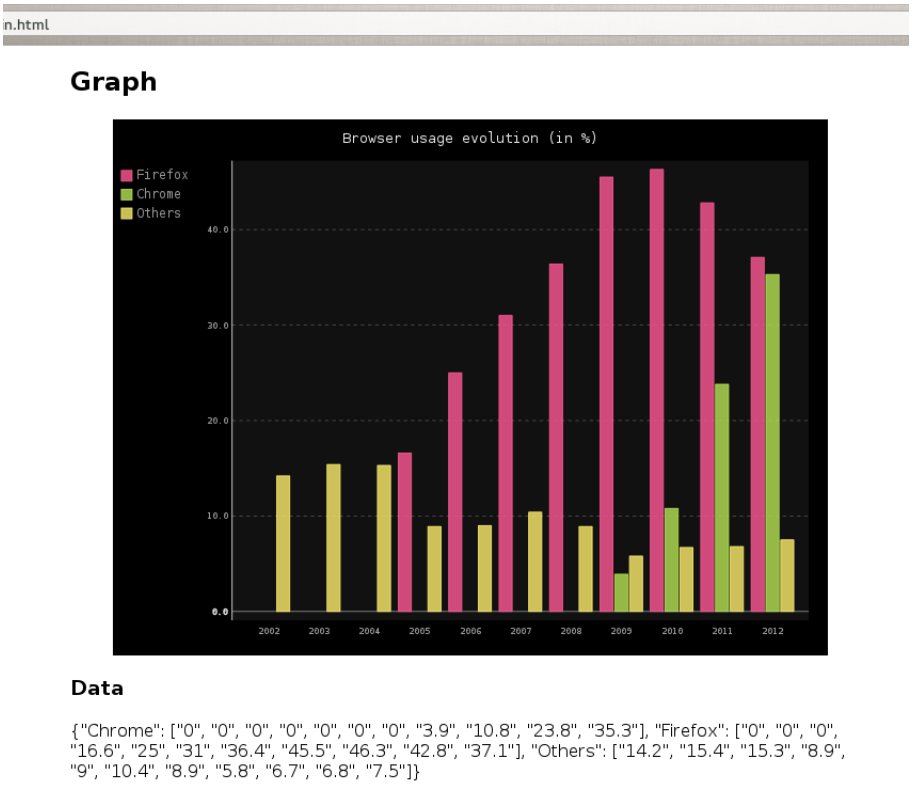
```
<body>
<figure>
  <embed type="image/svg+xml" src="./graph.svg" />
</figure>
</body>
```

You'll want to use to <*figure*> tags and not <image> tags to get the interactive functionalit pygal graphs provide.

If you would like to investigate alternative ways of embedding pugal's results on your web-page *pygal.org* has documentation on this.

Images

This is our end result:



Links

All of the above information can be found by following these urls:

- **Pygal: http://pygal.org/**
    - You can also find all of the Pygaldocumentation, first steps, and tons of other useful documentation here.
- **Python: https://www.python.org/**
    - Python is a powerful and easy to learn programming language.
- **Vagrant: http://www.vagrantbox.es/**
    - Vagrant is a simple tool to spin up and destroy virtual machines. It is used by many developers and is a powerful tool once you get it set up.
- **Virtualbox: https://www.virtualbox.org/**
    - Virtualbox is the go-to free and open source virtualmachine tool.

Further Reading

- **D3.js: http://d3js.org/**
    - Similar to pygal, D3 is a very robust graphing library written in javascript.
- **Pygal Documentation: http://pygal.org/documentation/**
    - Pygal includes tons of customizations and features you'll be interested in if you liked this HowTo. Read through the documentation for more info.