

Cloud Computing Project Report

Student No. x11105089

Course BSHCE

Application Irish Tide Times

Introduction

The aim of the project is to provide an SMS service for retrieving times of high and low tides at specific locations around the country. The service could be used by both professionals and hobbyists who need this information but may not have ready access to mobile internet due to geographic location or weather conditions (anglers, boaters, surfers etc.).

A user sends an SMS to a given Irish phone number with their location code, and the times for first low, first high, second low and second high tides are sent in a reply SMS to the user. If an erroneous location is given, the reply instead gives a list of possible location codes.

Twilio (<https://www.twilio.com/>) is used to provide the phone number and to handle incoming SMS. Twilio was chosen for several reasons. First, the documentation available for Twilio is second to none. Clear example of how to implement the API are given for a variety of different programming languages and frameworks. Twilio also proved to have excellent customer service and were helpfully able to supply a “beta” Irish telephone number which could handle SMS (this was important as it would be unfeasible if users had to sent an SMS to a UK number, incurring a cost). Another reason Twilio was used was that it did not prove to be cost-prohibitive, with both receiving and sending SMS costing a fraction of a cent. A Twilio account also provides useful records of each SMS sent and received, as well as affording flexibility in how incoming SMS are handled (eg. limiting the number of SMS per day and so on).

The second data source used was the tides page from The Irish Times at <http://www.irishtimes.com/weather/tides>. Due to anecdotal evidence this appears to be the canonical source used by anglers. Having reference these tides myself for many years I know them to be accurate. Secondly, using a well-established and reliable website such as the Irish Times’ ensures small chance of the information being unavailable on a given day. The tide information is also presented in plain HTML which proved to be relatively straightforward to parse.

Technical Overview

User Interface Design

The choice was made to keep the user interface simple as the main goal of the site is to explain what the service is, give the phone number required as well as the instructions for using it.

Twitter Bootstrap was effective for presenting this information in a clear and concise manner.

For the headline of the site, the Bootstrap feature “Jumbotron” was used (figure 1).

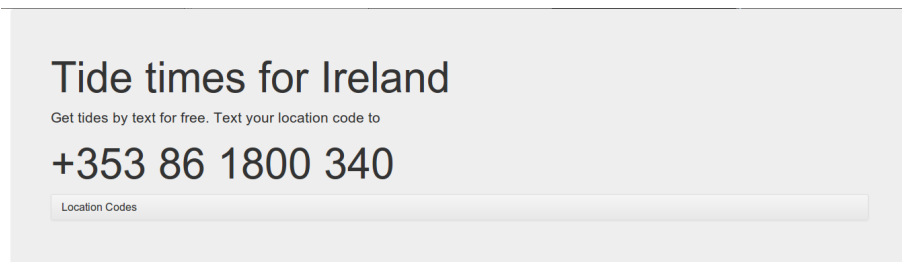


Figure 1: Bootstrap Jumbotron

On the Jumbotron, the overview of the site and the SMS phone number are clearly displayed, forming a “call to action”.

For the list of location codes, Bootstrap’s “Accordion” was used (figure 2). This allowed the location codes to be hidden until a user clicks, keeping the site looking uncluttered. When the user clicks again, the accordion closes. The Accordion was useful for displaying the location codes clearly.

For the About section at the bottom of the page, a Bootstrap column was used to contain the information. Two Bootstrap “glyphicons” were used to highlight the information points (figure 3).

Data Source 1: Twilio

Twilio provides a Python module for Twilio. This is installed simply using

```
pip install twilio
```


This is then imported into a Python file


```
import twilio.twiml
```



Figure 2: Bootstrap Accordion

About

 About this service.

 Done as part of Cloud Computing module for National College of Ireland, 2014. Student no. x11105089

© x11105089

Figure 3: Bootstrap Glyphicons

This gives us the functionality to create TWIML which is a HTML-like language for speaking to Twilio.

Twilio is given the URL for a callback which is accessed when your chosen phone number receives an SMS. This is implemented in my Django application using a view as follows

```
def receive_sms(request):

    location = request.GET.get('Body', '').strip()
    print location

    sorted_locations = sorted(Tide.locations)

    if location == "Dublin":
        location = "Dublin (North Wall)"

    if location in sorted_locations:
        results = Tide.objects.all().filter(location=location,date=date.today())
        if not results:
            tides.models.get_tides()
            results = Tide.objects.all().filter(location=location,date=date.today())

        tide = results[0]

        message = "Tides for %s at location %s\nFirst Low: %s\nFirst High: %s\nSecond Low: %s\nSecond High: %s" % (
            date.today(), location, tide.first_low, tide.first_high, tide.second_low, tide.second_high)
    else:
        message = "Sorry, can't find tides for that location code. Available location codes are: "
        for location in sorted_locations:
            message += "%s,\n" % location

    resp = twilio.twiml.Response()
    resp.message(message)

    return HttpResponseRedirect(resp, content_type='text/xml')
```

This code

- finds the location code given in the SMS
- strips any blank spaces
- checks location code is valid
- finds tides for the given location
- creates a TWIML response
- creates a message for response
- returns the response

Data Source 2: Irish Times website

The URL <http://www.irishtimes.com/weather/tides> is scraped for the purpose of gathering tide data. This is implemented using a Python library called BeautifulSoup. The library is installed using

```
pip install bs4
```

It is imported into a Python file using

```
from bs4 import BeautifulSoup
```

The following code is used to scrape the web page

```
def get_tides():

    url = "http://www.irishtimes.com/weather/tides"

    r = requests.get(url)

    data = r.text

    soup = BeautifulSoup(data)

    for td in soup.find_all('td'):
        if td.getText() in Tide.locations:
            tr = td.parent
            tds = tr.find_all('td')

            location = tds[0].text

            first_low = tds[1].text
            first_high = tds[2].text
            second_low = tds[3].text
            second_high = tds[4].text

            results = Tide.objects.all().filter(location=location,date=date.today())
            if not results.exists():
                tide = Tide()
                tide.date = date.today()
                tide.location = location

                if first_low:
                    tide.first_low = first_low
```

```

else:
    tide.first_low = None

if first_high:
    tide.first_high = first_high
else:
    tide.first_high = None

if second_low:
    tide.second_low = second_low
else:
    tide.second_low = None

if second_high:
    tide.second_high = second_high
else:
    tide.second_high = None

print "Saving tide for %s... :)" % location
tide.save()
else:
    print "Already have tide for %s today... :)" % location

```

The above code converts the HTML into a BeautifulSoup-parseable object and then traverses the relevant elements to gather tide information. Each “tide” is then stored as an object and then persisted to the database using

```

tide.save()

```

Deployment Strategy

Amazon EC2

<http://ec2-54-171-231-83.eu-west-1.compute.amazonaws.com/>

A user account was registered on Amazon AWS using the “free tier”.

Ireland was selected as the region for launching instances of servers.

A key pair was created and downloaded to my local machine.

An instance of Ubuntu 14.04 LTS 64bit was launched.

Port 80 was opened to allow demoing of the site (security groups -> settings and added a rule for port 80 allowing traffic)

SSH was used to access the server using the key which was downloaded

```
ssh -i aws.pem ubuntu@ec2-54-171-231-83.eu-west-1.compute.amazonaws.com
```

The installation instructions from README.md in the application's repository were followed (see below).

Digital Ocean

<http://178.62.76.73/>

Signed up to DigitalOcean.

I created a Droplet (instance) (€5 per month tier) calling it "irish-tide-times" and followed the setup wizard.

I chose London as the Region as this was the closest location.

I chose Ubuntu 14.04 64bit as the image.

I added public ssh key of my laptop to the account to allow me to access via ssh

```
ssh root@178.62.76.73
```

I followed the installation instructions from README.md in my application's repository (see below)

Installation

Install requirements

```
sudo apt-get install git nginx python-setuptools python-dev build-essential
```

Install pip

```
sudo easy_install pip
```

Install the Python requirements from requirements.txt

```
sudo pip install Django==1.5 South>=0.7.5 django-extensions==1.2.0 \
    beautifulsoup4 requests twilio gunicorn
```

Clone the repo

```
git clone https://github.com/sandeel/irish-tide-times.git
cd irish-tide-times
```

Copy the nginx config file found inside the repo

```
sudo cp nginx.conf /etc/nginx/nginx.conf
```

Make the gunicorn script executable

```
cd app
sudo chmod +x run_gunicorn.sh
```

Edit the following line in the wsgi.py file to point to the absolute location of the irish-tide-times folder

```
sys.path.append('/home/ubuntu/irish-tide-times')
```

run the server `./run_gunicorn.sh`