TECHNICAL REPORT: IDB3

Sochi 2014 Winter Olympics

by “4 Is Better Than 6”

Members:

- Austin Hooper
- Ardhimas Kamdani
- Nickson Dalimarta

INTRODUCTION

Description:

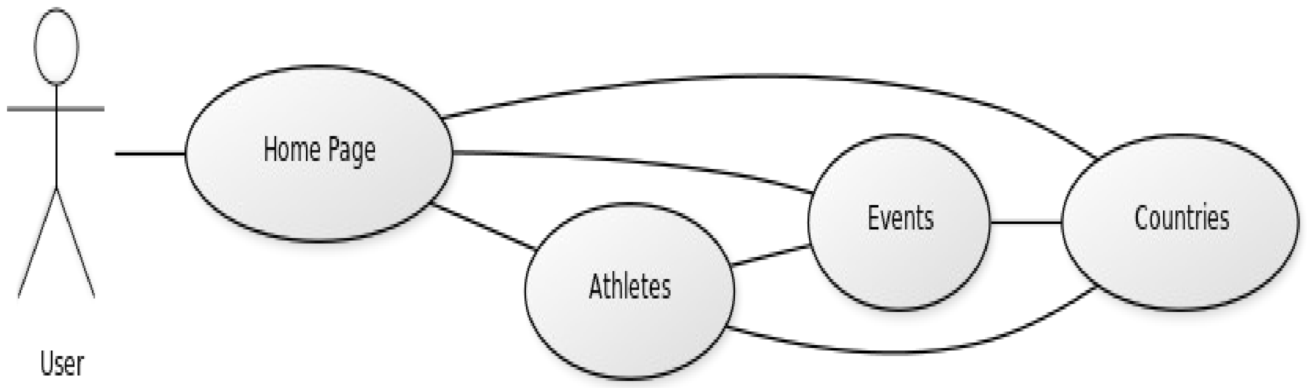
Have you ever found yourself wanting to know the events that went on during the 2014 Sochi Winter Olympics? Or the athletes that took part in said games? How about how well your favorite country did in said events? Then we have a site that is right up your alley!

Use cases (Phase 1) :

* From the Home Page a user can use the navigation bar to be directed to any Athlete, Event, and Country page.

* Navigation Bar exists in all pages. In phase 1, because of limited pages, a user may go to any page by making use of the navigation bar.

Use Case Diagram



DESIGN

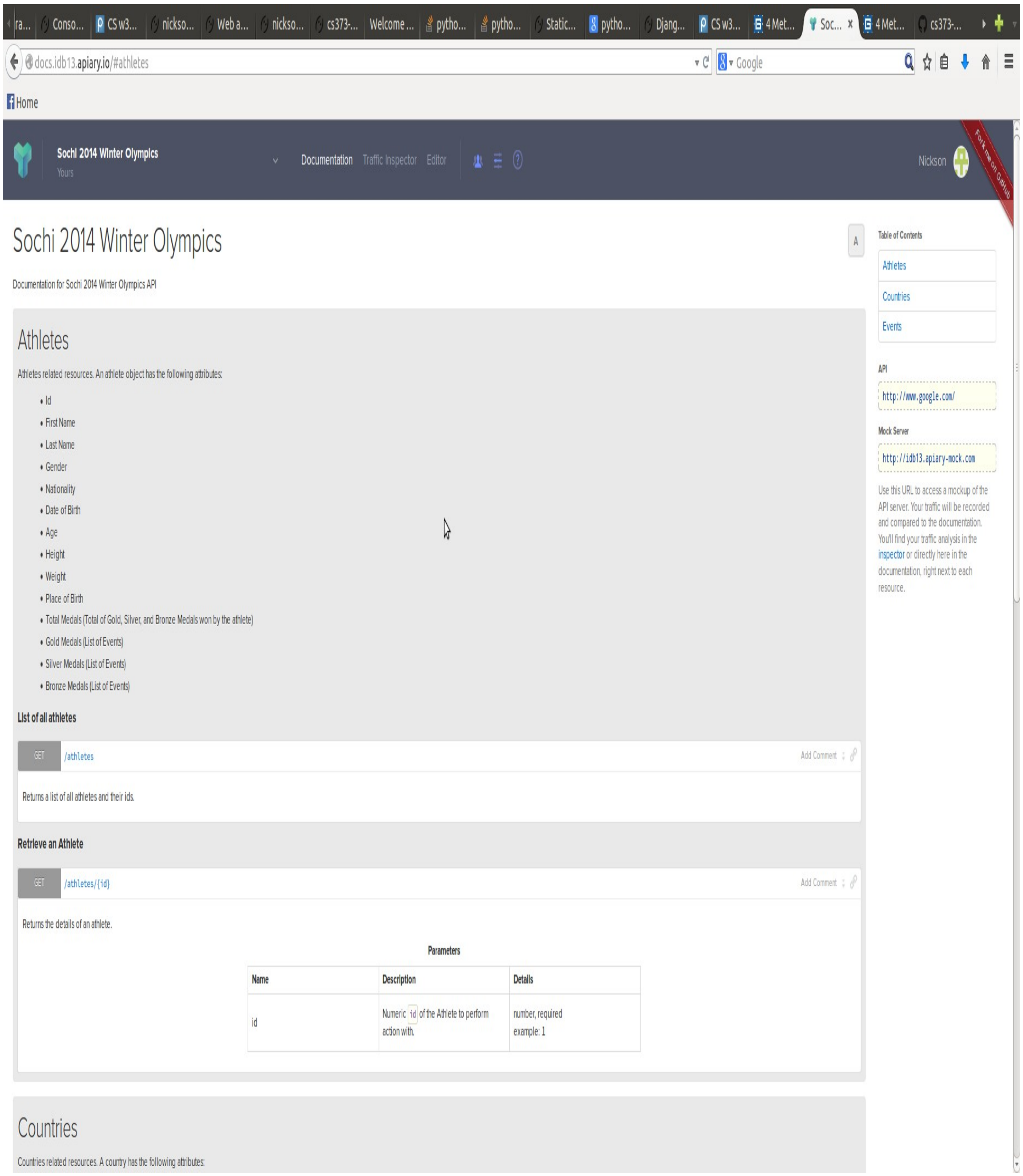
RESTful API:

API exists for retrieving information of all available athletes/ events/ countries. Due to the limited functionality, there are only 2 GET methods for each model. One to retrieve a list of all available datas for a model. Each element is a key-value pair of the model's id and name, separated by an equal sign. The other method is used retrieve the details of an object of a specified id (the only parameter of this method). A user who would like to get the information on Ireen Wust, for example, may first use the first method to get this list

```
"ID=Athletes":["1=Meryl Davis","2=Tobias Arit","3=Ireen Wust"]
```

He can then use the second method to get these details

```
{

    "Id":3,
    "First Name":"Ireen",
    "Last Name":"Wust",
    "Gender":"Female",
    "Nationality":"Netherlands",
    "Date of Birth":"1 April 1986",
    "Age":28,
    "Height":"1.68m (5'6")",
    "Weight":"63 kg (139 lbs)",
    "Place of birth":"Goirle, Netherlands",
    "Total Medals":5,
    "Gold Medals":["Speed Skating:Ladies' 3000m", "Speed Skating:Ladies' Team
      Pursuit"]
    "Silver Medals":["Speed Skating:Ladies' 1000m", "Speed Skating:Ladies' 1500m",
      "Speed Skating:Ladies' 5000m"]
    "Bronze Medals":[]
}
```



Sochi 2014 Winter Olympics

Documentation for Sochi 2014 Winter Olympics API

Athletes

Athletes related resources. An athlete object has the following attributes:

- Id
- First Name
- Last Name
- Gender
- Nationality
- Date of Birth
- Age
- Height
- Weight
- Place of Birth
- Total Medals (Total of Gold, Silver, and Bronze Medals won by the athlete)
- Gold Medals (List of Events)
- Silver Medals (List of Events)
- Bronze Medals (List of Events)

List of all athletes

GET /athletes

Returns a list of all athletes and their ids.

Retrieve an Athlete

GET /athletes/{id}

Returns the details of an athlete.

Parameters

Name	Description	Details
id	Numeric `id` of the Athlete to perform action with.	number, required example: 1

Countries

Countries related resources. A country has the following attributes:

Table of Contents

- Athletes
- Countries
- Events

API

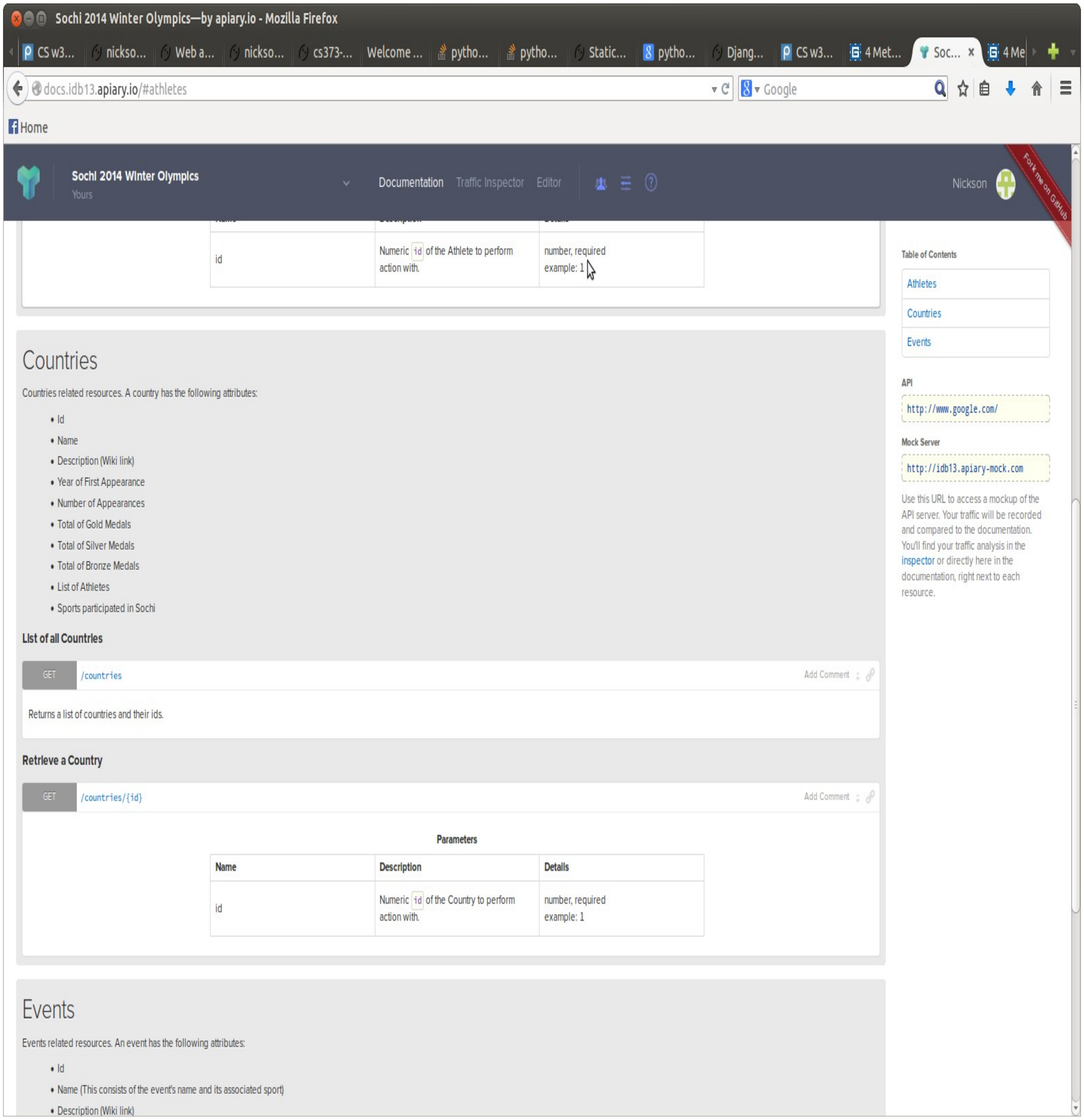
`http://www.google.com/`

Mock Server

`http://idb13.apiary-mock.com`

Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.

API documentation for athletes in Apiary


Sochi 2014 Winter Olympics—by apiary.io - Mozilla Firefox
docs.idb13.apiary.io/#athletes
Home
Sochi 2014 Winter Olympics
Yours
Documentation
Traffic Inspector
Editor
Nickson
Fork me on GitHub

id	Numeric id of the Athlete to perform action with.	number, required example: 1

Table of Contents

- Athletes
- Countries
- Events

API

`http://www.google.com/`

Mock Server

`http://idb13.apiary-mock.com`

Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.

Countries

Countries related resources. A country has the following attributes:

- Id
- Name
- Description (Wiki link)
- Year of First Appearance
- Number of Appearances
- Total of Gold Medals
- Total of Silver Medals
- Total of Bronze Medals
- List of Athletes
- Sports participated in Sochi

List of all Countries

GET `/countries` Add Comment

Returns a list of countries and their ids.

Retrieve a Country

GET `/countries/{id}` Add Comment

Parameters

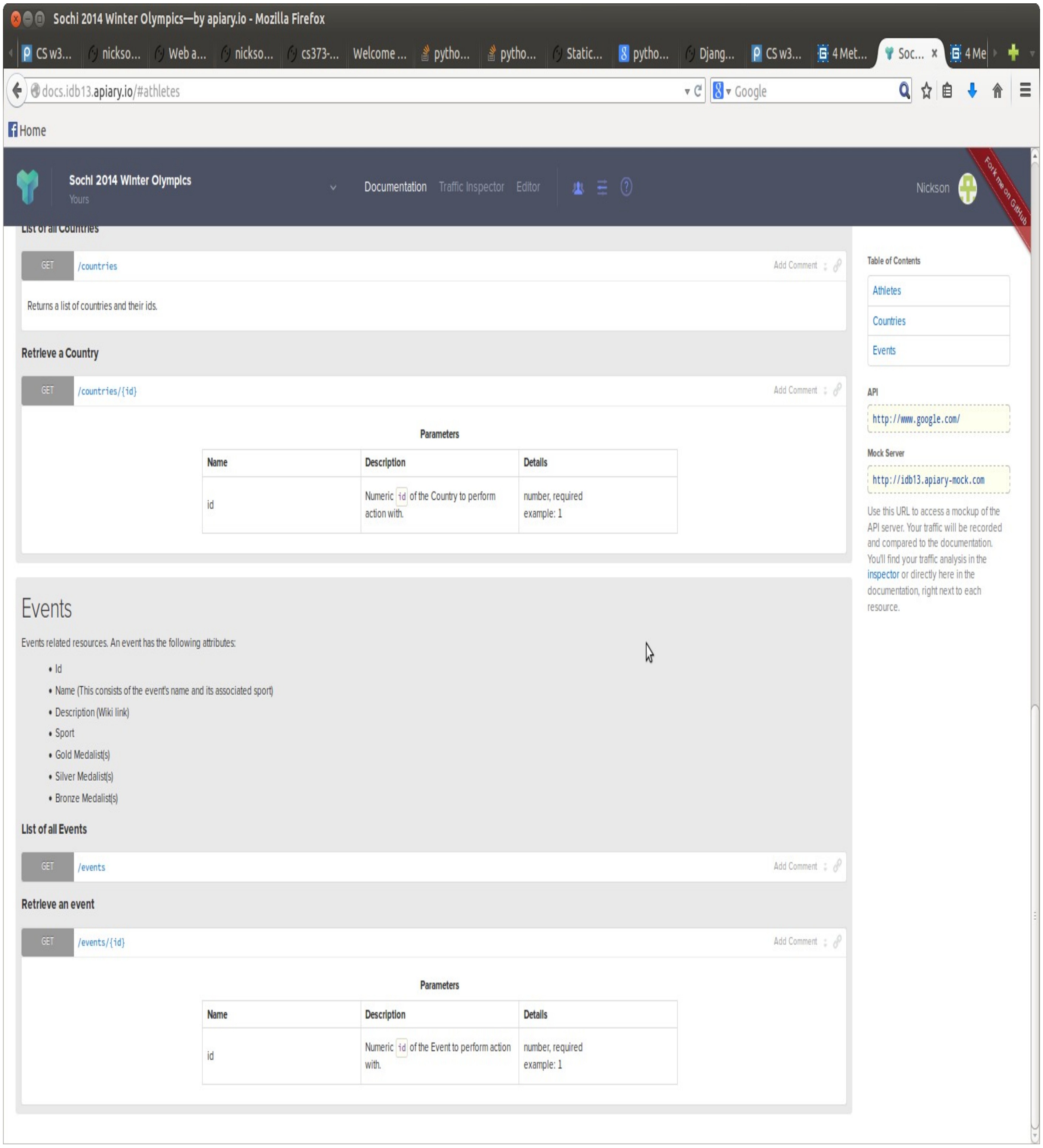
Name	Description	Details
id	Numeric id of the Country to perform action with.	number, required example: 1

Events

Events related resources. An event has the following attributes:

- Id
- Name (This consists of the event's name and its associated sport)
- Description (Wiki link)

API documentation for countries in Apiary


Sochi 2014 Winter Olympics—by apiary.io - Mozilla Firefox

docs.idb13.apiary.io/#athletes

Sochi 2014 Winter Olympics

Documentation Traffic Inspector Editor

List of all Countries

GET /countries

Returns a list of countries and their ids.

Retrieve a Country

GET /countries/{id}

Parameters

Name	Description	Details
id	Numeric id of the Country to perform action with.	number, required example: 1

Events

Events related resources. An event has the following attributes:

- Id
- Name (This consists of the event's name and its associated sport)
- Description (Wiki link)
- Sport
- Gold Medalist(s)
- Silver Medalist(s)
- Bronze Medalist(s)

List of all Events

GET /events

Retrieve an event

GET /events/{id}

Parameters

Name	Description	Details
id	Numeric id of the Event to perform action with.	number, required example: 1

Table of Contents

- Athletes
- Countries
- Events

API

http://www.google.com/

Mock Server

http://idb13.apiary-mock.com

Use this URL to access a mockup of the API server. Your traffic will be recorded and compared to the documentation. You'll find your traffic analysis in the inspector or directly here in the documentation, right next to each resource.

API documentation for events in Apiary

Django Models & Attributes:

1. <u>ATHLETES</u>

   * First Name

     - Primary Key
     - Athlete's first name
     - String

   * Last Name

     - Primary Key
     - Athlete's last name
     - String

   * Country

     - Athlete's Country
     - String

   * Gender

     - Athlete's Gender
     - String

   * Birthdate

     - Primary Key
     - Athlete's Date of birth

    - Date

* Height

    - Athlete's Height

    - String

* Weight

    - Athlete's Weight

    - String

* Picture

    - Athlete's Picture

    - Name of image file with extension

* Video

    - YouTube video related to the athlete

    - YouTube link in https://www.youtube.com/v/* format

* Gold Medals

    - List of events the athlete won the gold medal from

    - Foreign Key

* Silver Medals: Events

    - List of events the athlete won the silver medal from

    - Foreign Key

* Bronze Medals: Events

- List of events the athlete won the bronze medal from
- Foreign Key

FirstName, LastName, and DateofBirth are chosen as the primary keys to prevent conflicts. It is highly unlikely that participating athletes will have the same name and date of birth.

2. COUNTRIES

* Name

- Primary Key
- Name of the coutnry
- String

* Athlete List

- List of participating athletes from the country
- String

* Description

- Brief description of the country
- String

* Gold Medals Won

- Total of gold medals won by the country
- Dynamically calculated from the medals won by the athletes

- Integer

* Silver Medals Won
    - Total of silver medals won by the country
    - Dynamically calculated from the medals won by the athletes
* Bronze Medals Won
    - Total of bronze medals won by the country
    - Dynamically calculated from the medals won by the athletes
* Country Code
    - 3 letter International Olympic Committee country code
    - String
* Coordx
    - X-Coordinate of country used in Google Maps
    - Float
* Coordy
    - Y-Coordinate of country used in Google Maps
    - Float
* Coordz
    - Zoom value used in Google Maps
    - Int

3. <u>EVENTS</u>

* Name
    - Primary Key
    - Name of the event
    - String
* Sport
    - Primary Key
    - Name of the associated sport
    - String
* Description
    - A brief description of the event
    - String
* Icon
    - Icon that represents the sport of the event
    - Name of icon file with extension
* Gold Medalist(s)
    - The name of the gold medalist(s). Can be multiple athletes.
    - Foreign Key
* Silver Medalist(s)
    - The name of the silver medalist(s). Can be multiple athletes.

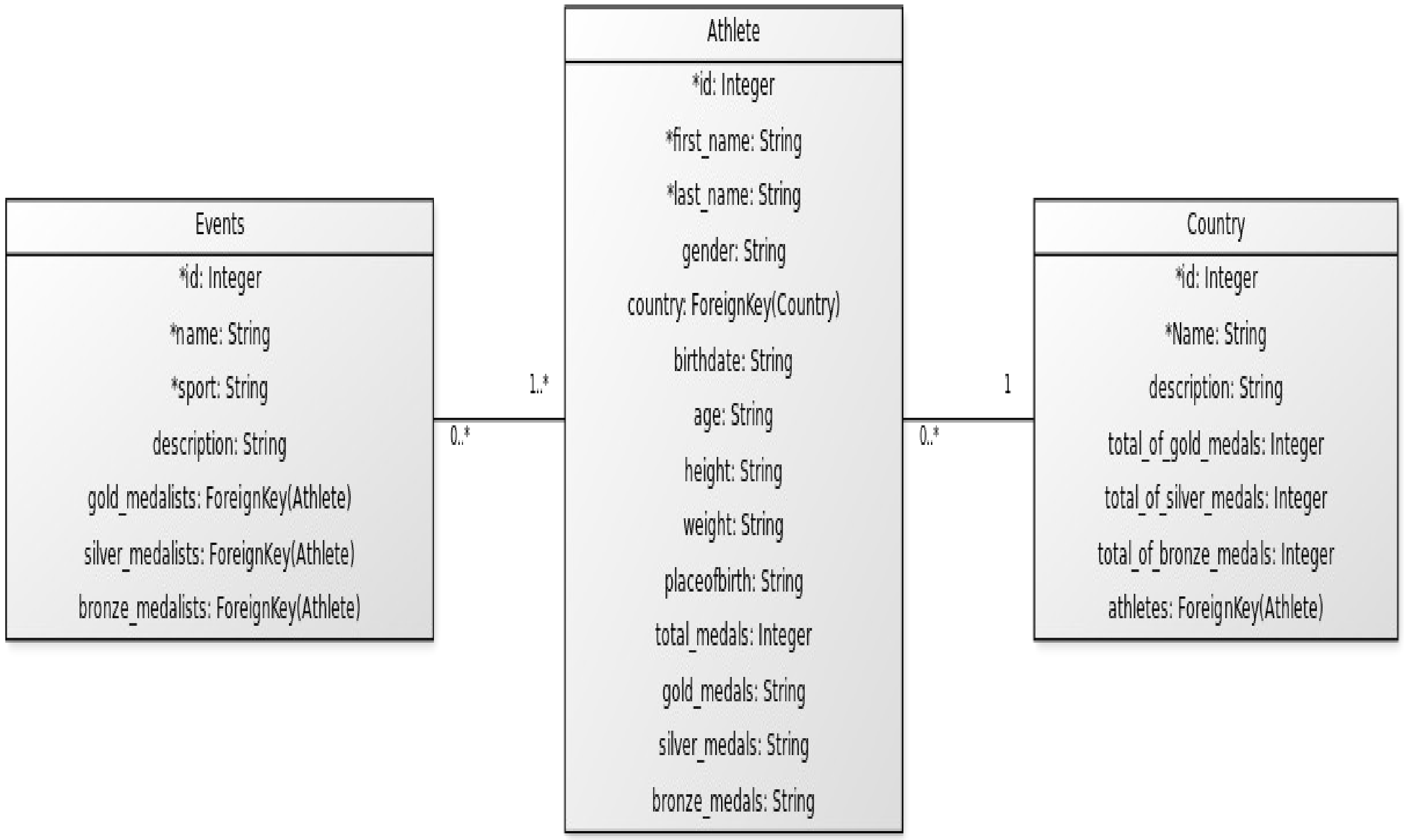
    - Foreign Key

* Bronze Medalist(s)

    - The name of the bronze medalist(s). Can be multiple athletes.

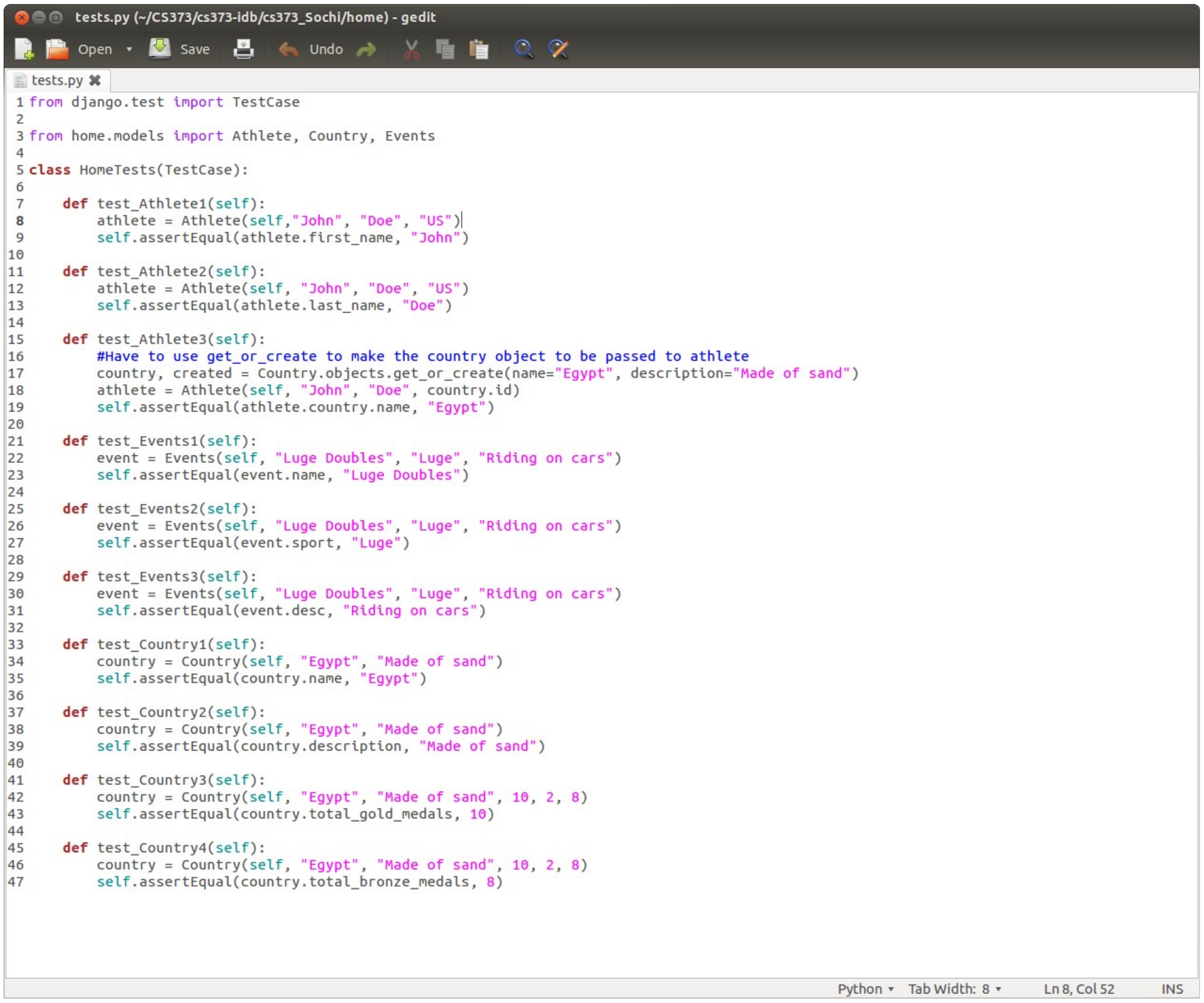
    - Foreign Key

Django Models Diagram



TESTS

Django Models:

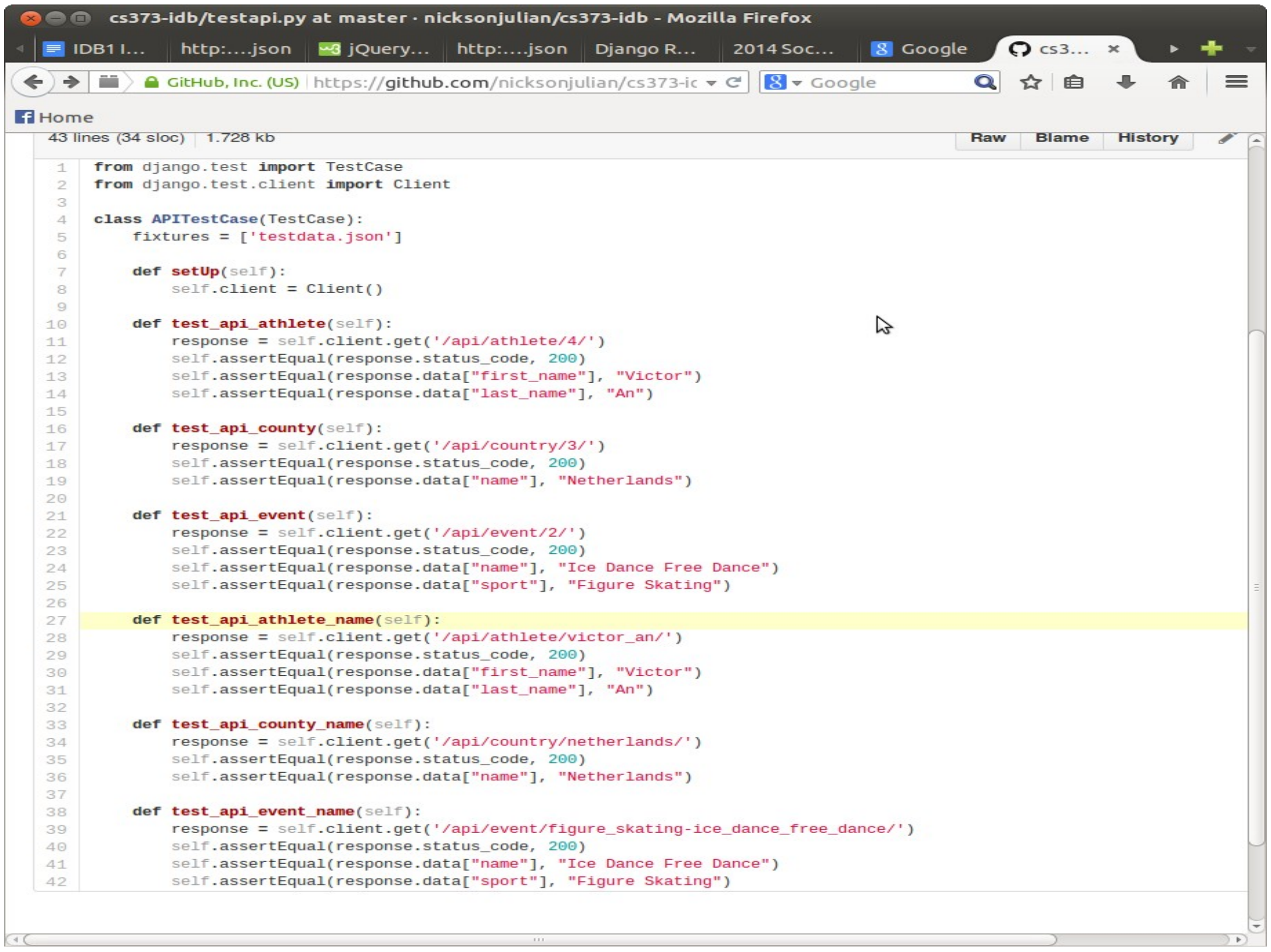


```python
from django.test import TestCase

from home.models import Athlete, Country, Events

class HomeTests(TestCase):

    def test_Athlete1(self):
        athlete = Athlete(self,"John", "Doe", "US")
        self.assertEqual(athlete.first_name, "John")

    def test_Athlete2(self):
        athlete = Athlete(self, "John", "Doe", "US")
        self.assertEqual(athlete.last_name, "Doe")

    def test_Athlete3(self):
        #Have to use get_or_create to make the country object to be passed to athlete
        country, created = Country.objects.get_or_create(name="Egypt", description="Made of sand")
        athlete = Athlete(self, "John", "Doe", country.id)
        self.assertEqual(athlete.country.name, "Egypt")

    def test_Events1(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.name, "Luge Doubles")

    def test_Events2(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.sport, "Luge")

    def test_Events3(self):
        event = Events(self, "Luge Doubles", "Luge", "Riding on cars")
        self.assertEqual(event.desc, "Riding on cars")

    def test_Country1(self):
        country = Country(self, "Egypt", "Made of sand")
        self.assertEqual(country.name, "Egypt")

    def test_Country2(self):
        country = Country(self, "Egypt", "Made of sand")
        self.assertEqual(country.description, "Made of sand")

    def test_Country3(self):
        country = Country(self, "Egypt", "Made of sand", 10, 2, 8)
        self.assertEqual(country.total_gold_medals, 10)

    def test_Country4(self):
        country = Country(self, "Egypt", "Made of sand", 10, 2, 8)
        self.assertEqual(country.total_bronze_medals, 8)
```

Simple tests that test if objects were created successfully.

Added for IDB2 is the tests to check the usage of these objects. The way we do the test is we check if the attributes of the objects created have the right values.

API:



cs373-idb/testapi.py at master · nicksonjulian/cs373-idb - Mozilla Firefox

https://github.com/nicksonjulian/cs373-ic

43 lines (34 sloc) 1.728 kb

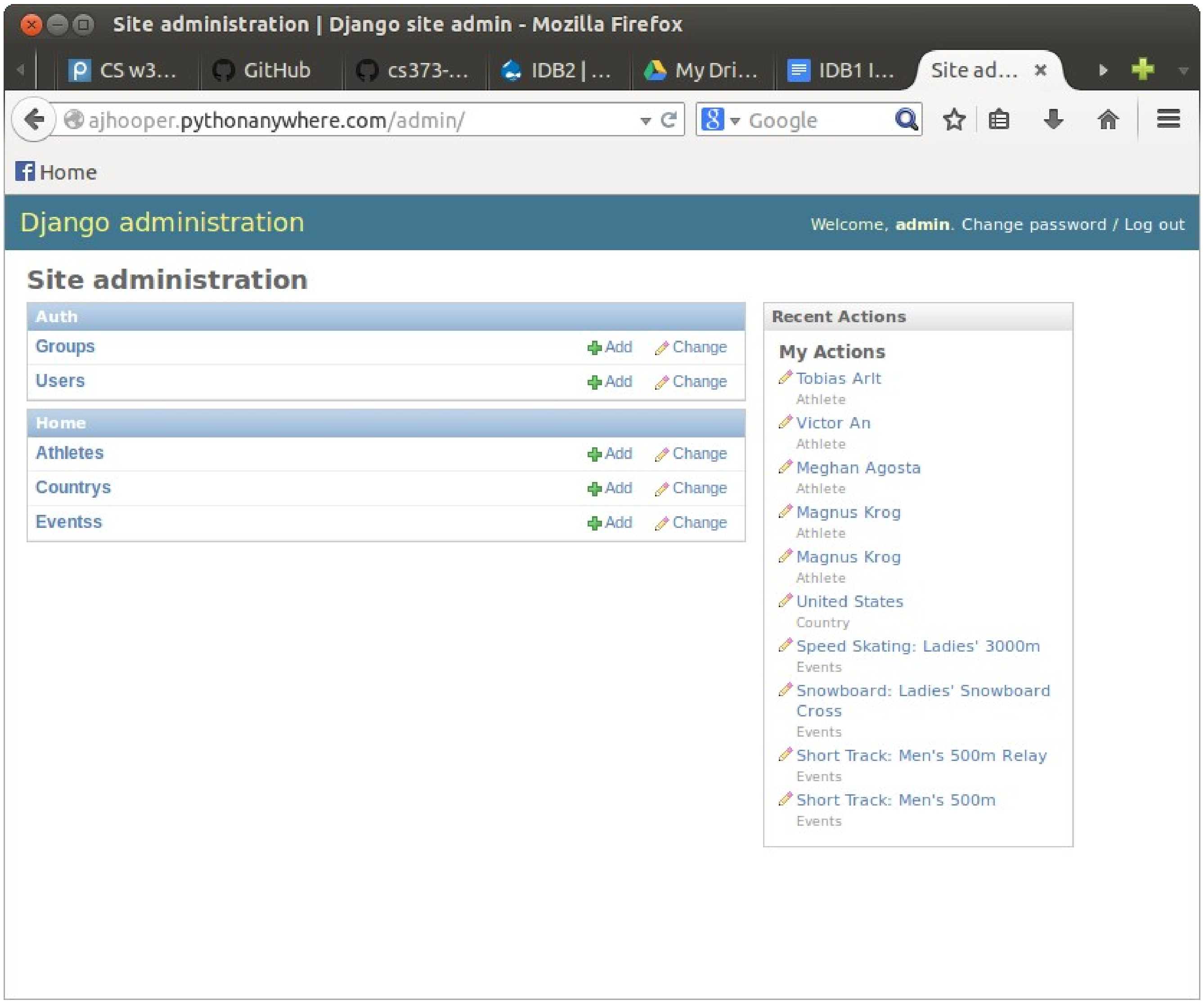
```python
from django.test import TestCase
from django.test.client import Client

class APITestCase(TestCase):
    fixtures = ['testdata.json']

    def setUp(self):
        self.client = Client()

    def test_api_athlete(self):
        response = self.client.get('/api/athlete/4/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["first_name"], "Victor")
        self.assertEqual(response.data["last_name"], "An")

    def test_api_county(self):
        response = self.client.get('/api/country/3/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["name"], "Netherlands")

    def test_api_event(self):
        response = self.client.get('/api/event/2/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["name"], "Ice Dance Free Dance")
        self.assertEqual(response.data["sport"], "Figure Skating")

    def test_api_athlete_name(self):
        response = self.client.get('/api/athlete/victor_an/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["first_name"], "Victor")
        self.assertEqual(response.data["last_name"], "An")

    def test_api_county_name(self):
        response = self.client.get('/api/country/netherlands/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["name"], "Netherlands")

    def test_api_event_name(self):
        response = self.client.get('/api/event/figure_skating-ice_dance_free_dance/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data["name"], "Ice Dance Free Dance")
        self.assertEqual(response.data["sport"], "Figure Skating")
```

tests for API.

Tests for API check if API returns the correct values. As an example, test_api_athlete checks if the name of the person with id 4 is Victor Ann. The test_api_athlete does the same thing, but by using the full name of the person instead of the id.

DATABASE

Fortunately Django handles the sql database for us and we do not have to write any sql queries to manage our database. The database can be accessed by going to the admin page. The picture following is the look of the admin page.


Site administration | Django site admin - Mozilla Firefox
CS w3...
GitHub
cs373-...
IDB2 | ...
My Dri...
IDB1 I...
Site ad...
ajhooper.pythonanywhere.com/admin/
Google
Home

Django administration

Welcome, **admin**. Change password / Log out

Site administration

Auth		
Groups	Add	Change
Users	Add	Change

Home		
Athletes	Add	Change
Countrys	Add	Change
Eventss	Add	Change

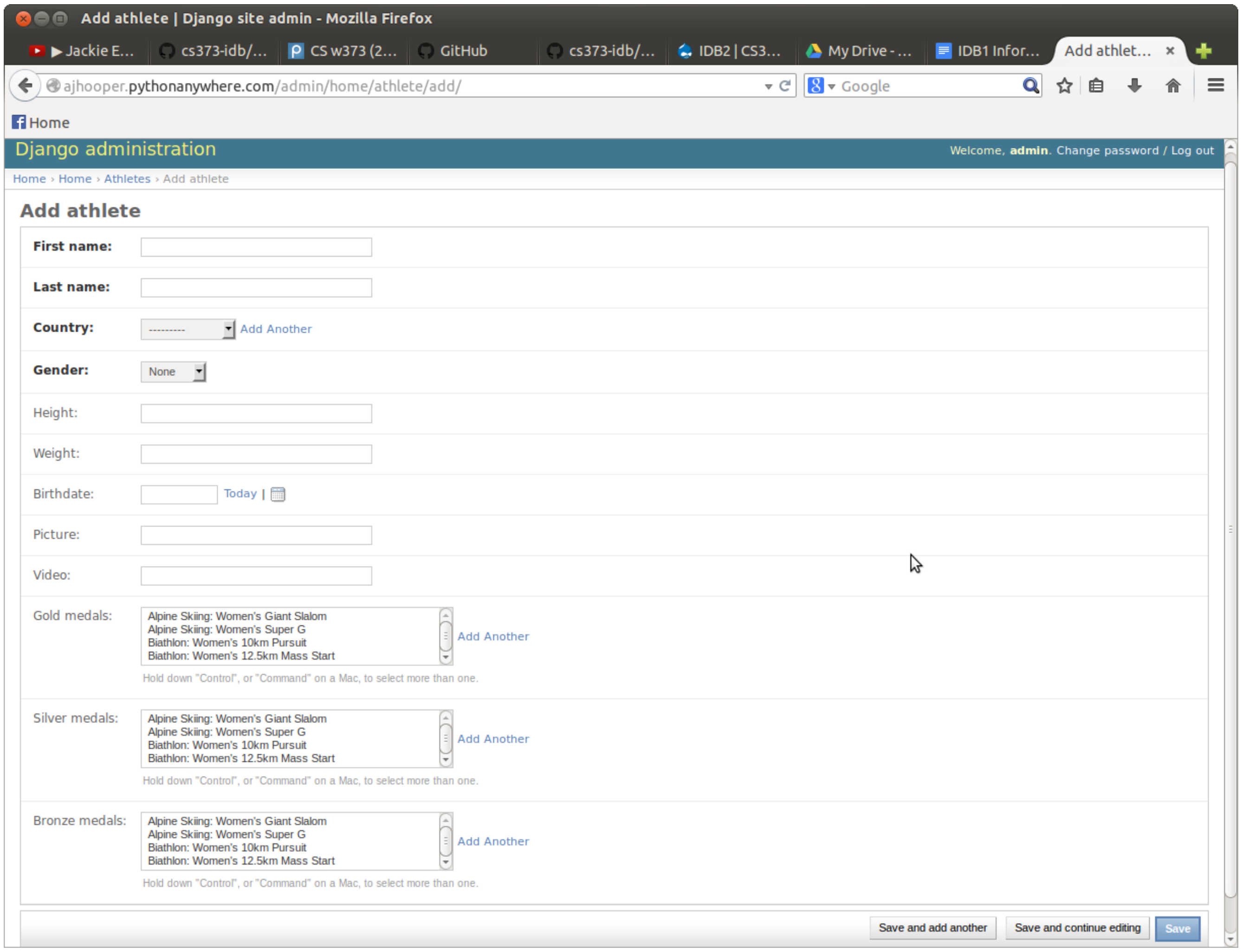
Recent Actions

My Actions

- Tobias Arlt — Athlete
- Victor An — Athlete
- Meghan Agosta — Athlete
- Magnus Krog — Athlete
- Magnus Krog — Athlete
- United States — Country
- Speed Skating: Ladies' 3000m — Events
- Snowboard: Ladies' Snowboard Cross — Events
- Short Track: Men's 500m Relay — Events
- Short Track: Men's 500m — Events

Admin page of our website.

We can easily add a new data to our database by clicking add which will direct us to a form with all the attributes of a particular model. We can just fill in the form to add the desired data.



Django administration

Welcome, **admin**. Change password / Log out

Home › Home › Athletes › Add athlete

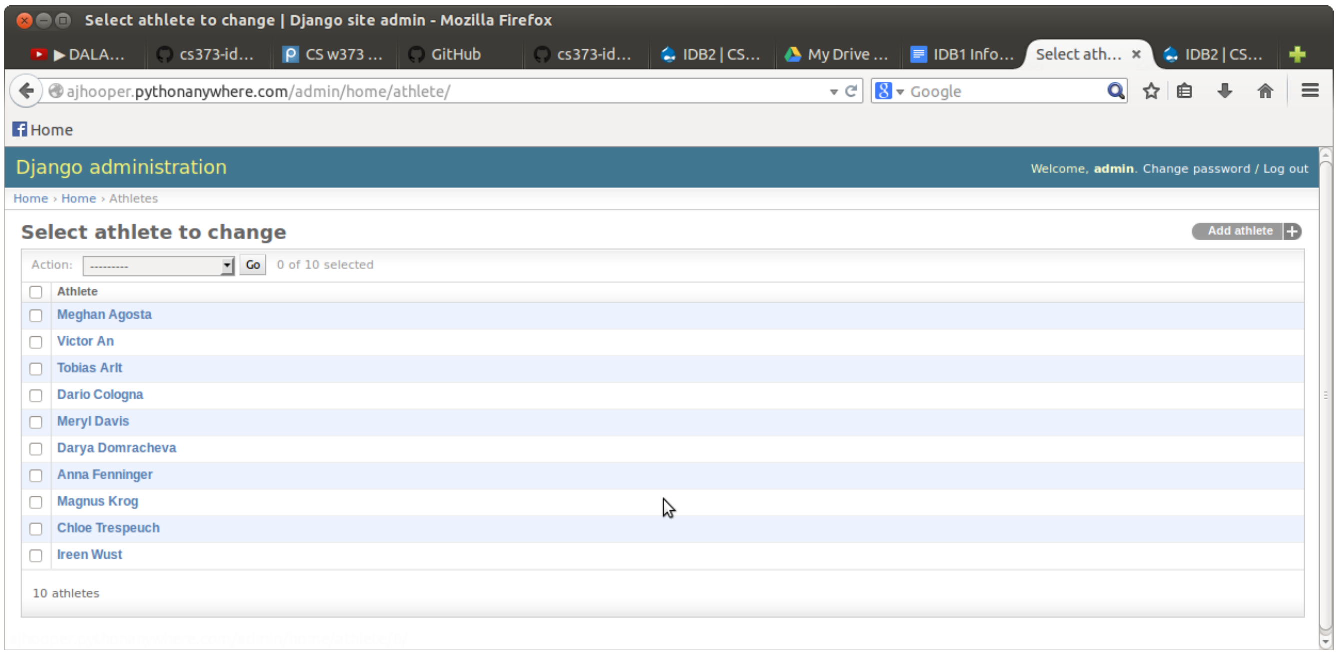
Add athlete

First name:	
Last name:	
Country:	--------- Add Another
Gender:	None
Height:	
Weight:	
Birthdate:	Today |
Picture:	
Video:	
Gold medals:	Alpine Skiing: Women's Giant Slalom Alpine Skiing: Women's Super G Biathlon: Women's 10km Pursuit Biathlon: Women's 12.5km Mass Start Add Another Hold down "Control", or "Command" on a Mac, to select more than one.
Silver medals:	Alpine Skiing: Women's Giant Slalom Alpine Skiing: Women's Super G Biathlon: Women's 10km Pursuit Biathlon: Women's 12.5km Mass Start Add Another Hold down "Control", or "Command" on a Mac, to select more than one.
Bronze medals:	Alpine Skiing: Women's Giant Slalom Alpine Skiing: Women's Super G Biathlon: Women's 10km Pursuit Biathlon: Women's 12.5km Mass Start Add Another Hold down "Control", or "Command" on a Mac, to select more than one.

Add form of athlete.

One important thing to notice is that Django automatically creates a drop-down list for foreign keys. This list is automatically updated whenever there is a change in the relevant databases. In this case, all Events, which are foreign keys for are listed in the drop-down list for Gold, Silver, and Bronze medals.

For updating the existing datas, we can click Change which directs us to the list of existing datas for a particular model.



Select athlete to change | Django site admin - Mozilla Firefox
ajhooper.pythonanywhere.com/admin/home/athlete/
Django administration
Welcome, admin. Change password / Log out
Home › Home › Athletes
Add athlete
Action: --------- Go

Select athlete to change

0 of 10 selected

Athlete
Meghan Agosta
Victor An
Tobias Arlt
Dario Cologna
Meryl Davis
Darya Domracheva
Anna Fenninger
Magnus Krog
Chloe Trespeuch
Ireen Wust

10 athletes

List of existing athletes.

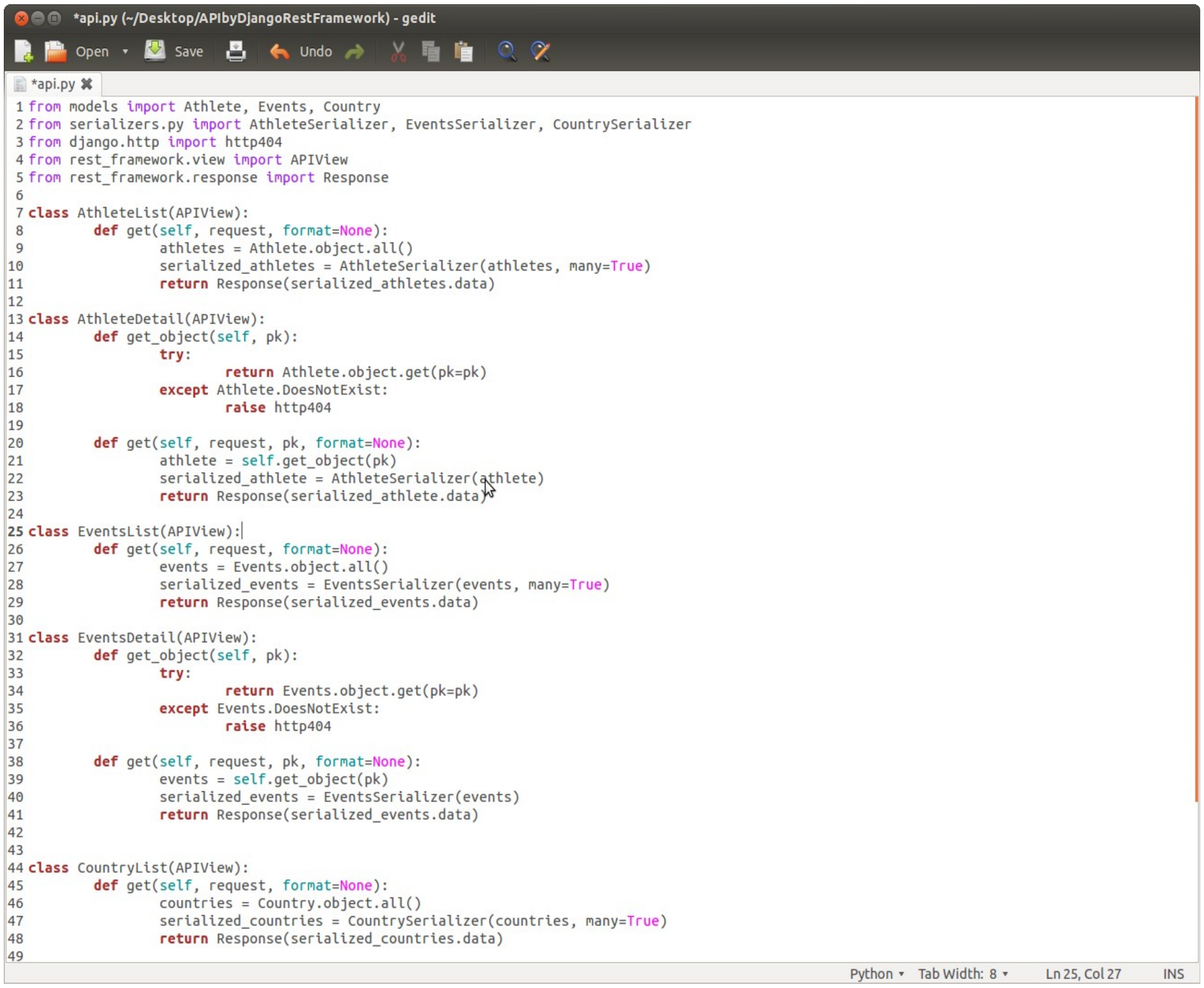
To update an athlete, say Ireen Wust, we just have to click her name which directs us to a filled form. We can then change the attributes and click save to update.

<u>**API (Django REST framework or Tastypie?)**</u>

There were two options to the software that we can use to build the API. One was the Django Rest Framework, the other one is Tastypie. We decided to create both versions and see which one suited better for our site.

A. <u>Django REST framework</u>

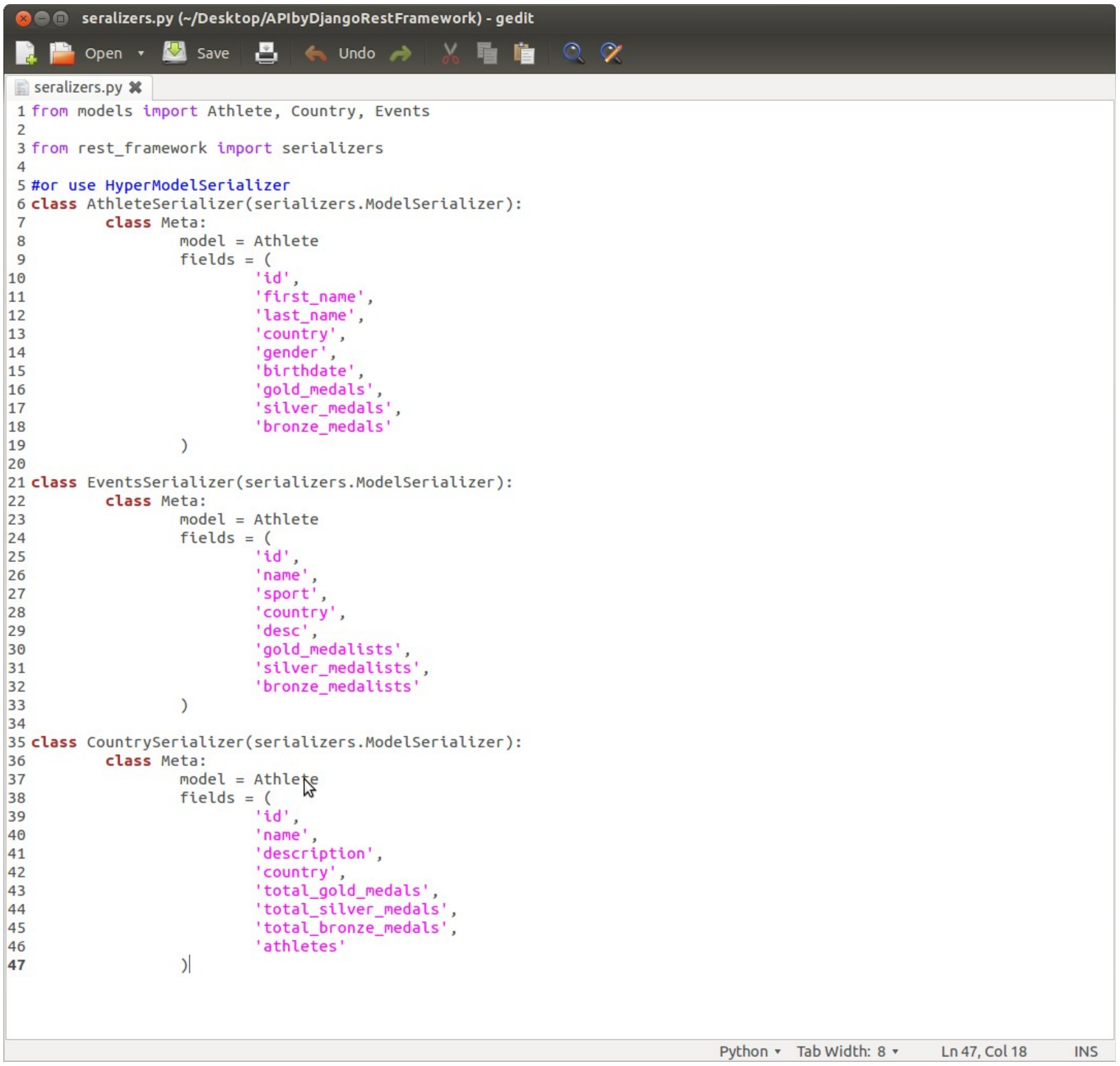
The core file is api.py in which we decribe the kind of APIs that we want as described in DESIGN section for RESTful API (list and details).



```python
from models import Athlete, Events, Country
from serializers.py import AthleteSerializer, EventsSerializer, CountrySerializer
from django.http import http404
from rest_framework.view import APIView
from rest_framework.response import Response

class AthleteList(APIView):
        def get(self, request, format=None):
                athletes = Athlete.object.all()
                serialized_athletes = AthleteSerializer(athletes, many=True)
                return Response(serialized_athletes.data)

class AthleteDetail(APIView):
        def get_object(self, pk):
                try:
                        return Athlete.object.get(pk=pk)
                except Athlete.DoesNotExist:
                        raise http404

        def get(self, request, pk, format=None):
                athlete = self.get_object(pk)
                serialized_athlete = AthleteSerializer(athlete)
                return Response(serialized_athlete.data)

class EventsList(APIView):
        def get(self, request, format=None):
                events = Events.object.all()
                serialized_events = EventsSerializer(events, many=True)
                return Response(serialized_events.data)

class EventsDetail(APIView):
        def get_object(self, pk):
                try:
                        return Events.object.get(pk=pk)
                except Events.DoesNotExist:
                        raise http404

        def get(self, request, pk, format=None):
                events = self.get_object(pk)
                serialized_events = EventsSerializer(events)
                return Response(serialized_events.data)


class CountryList(APIView):
        def get(self, request, format=None):
                countries = Country.object.all()
                serialized_countries = CountrySerializer(countries, many=True)
                return Response(serialized_countries.data)

```

api.py for Django REST framework.

For Django Rest Framework, we had to create another file called serializers.py that allows the data to be easily rendered to JSON, XML format.

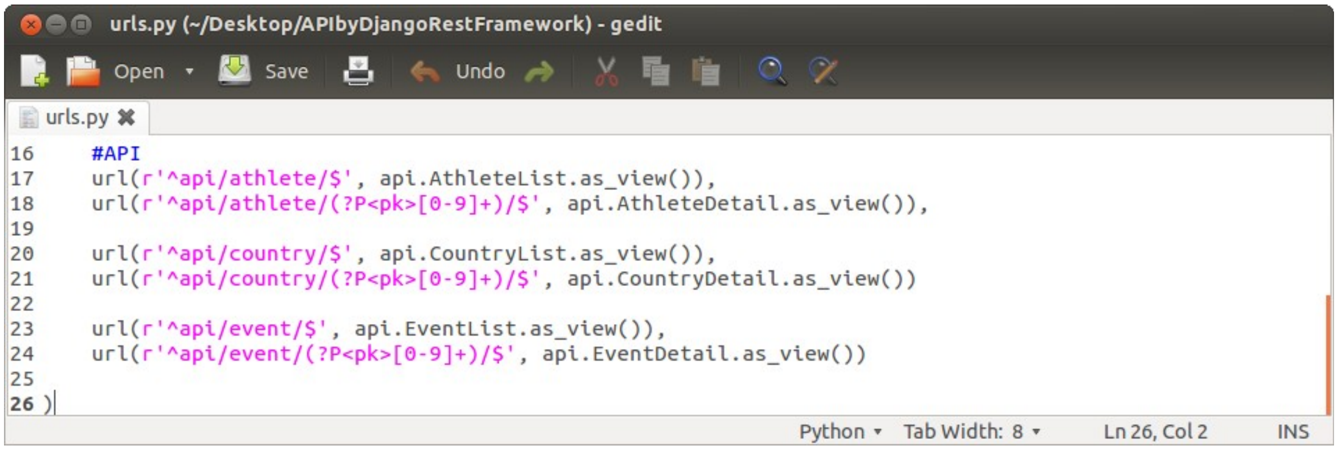

seralizers.py (~/Desktop/APIbyDjangoRestFramework) - gedit

```
from models import Athlete, Country, Events

from rest_framework import serializers

#or use HyperModelSerializer
class AthleteSerializer(serializers.ModelSerializer):
        class Meta:
                model = Athlete
                fields = (
                        'id',
                        'first_name',
                        'last_name',
                        'country',
                        'gender',
                        'birthdate',
                        'gold_medals',
                        'silver_medals',
                        'bronze_medals'
                )

class EventsSerializer(serializers.ModelSerializer):
        class Meta:
                model = Athlete
                fields = (
                        'id',
                        'name',
                        'sport',
                        'country',
                        'desc',
                        'gold_medalists',
                        'silver_medalists',
                        'bronze_medalists'
                )

class CountrySerializer(serializers.ModelSerializer):
        class Meta:
                model = Athlete
                fields = (
                        'id',
                        'name',
                        'description',
                        'country',
                        'total_gold_medals',
                        'total_silver_medals',
                        'total_bronze_medals',
                        'athletes'
                )
```

Python Tab Width: 8 Ln 47, Col 18 INS

Django Rest Framework Serializers.py.

Finally, to make sure that the API pages are running, we update the urls.py to include the new API pages.

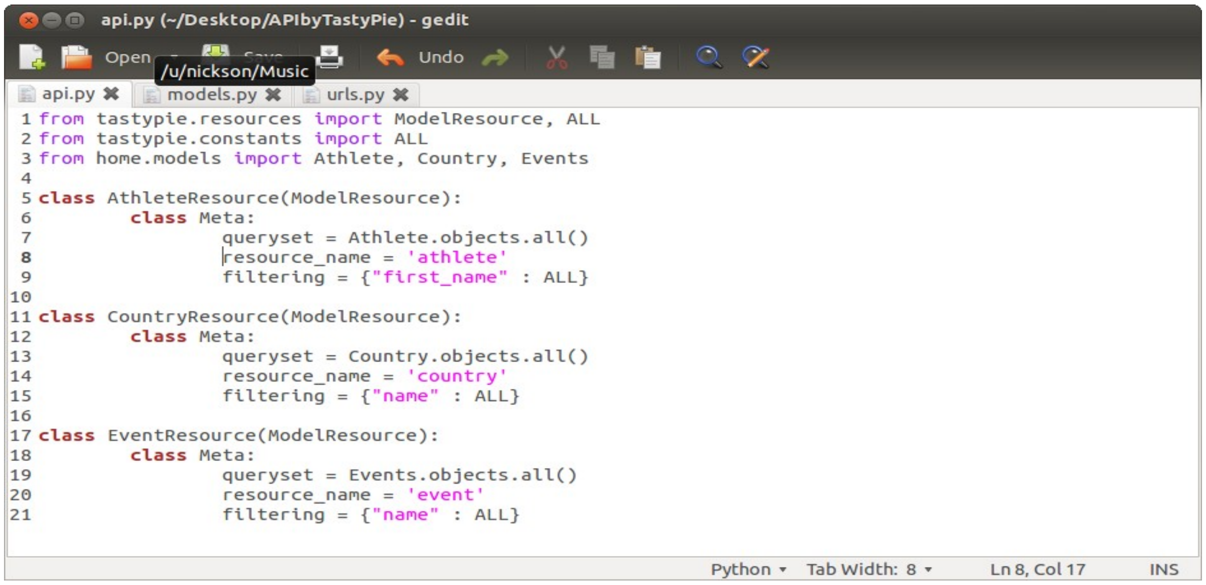


```
    #API
    url(r'^api/athlete/$', api.AthleteList.as_view()),
    url(r'^api/athlete/(?P<pk>[0-9]+)/$', api.AthleteDetail.as_view()),

    url(r'^api/country/$', api.CountryList.as_view()),
    url(r'^api/country/(?P<pk>[0-9]+)/$', api.CountryDetail.as_view())

    url(r'^api/event/$', api.EventList.as_view()),
    url(r'^api/event/(?P<pk>[0-9]+)/$', api.EventDetail.as_view())

)
```

Added URLs for API.

B. <u>Tastypie</u>

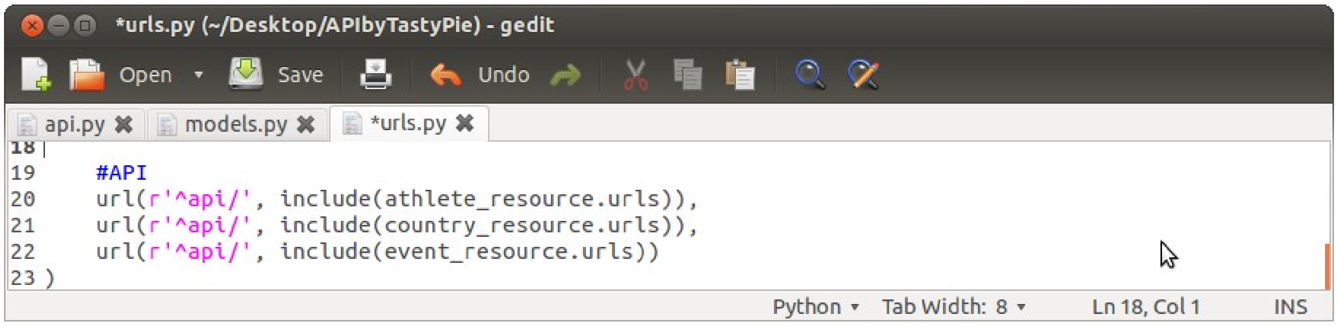
Here, we also included the core code for API in a file called api.py.



```
from tastypie.resources import ModelResource, ALL
from tastypie.constants import ALL
from home.models import Athlete, Country, Events

class AthleteResource(ModelResource):
        class Meta:
                queryset = Athlete.objects.all()
                resource_name = 'athlete'
                filtering = {"first_name" : ALL}

class CountryResource(ModelResource):
        class Meta:
                queryset = Country.objects.all()
                resource_name = 'country'
                filtering = {"name" : ALL}

class EventResource(ModelResource):
        class Meta:
                queryset = Events.objects.all()
                resource_name = 'event'
                filtering = {"name" : ALL}
```

api.py for Tastypie.

We also had to add the API. However, the ModelResource amazingly handled all the urls and we do not have to create any regular expression.

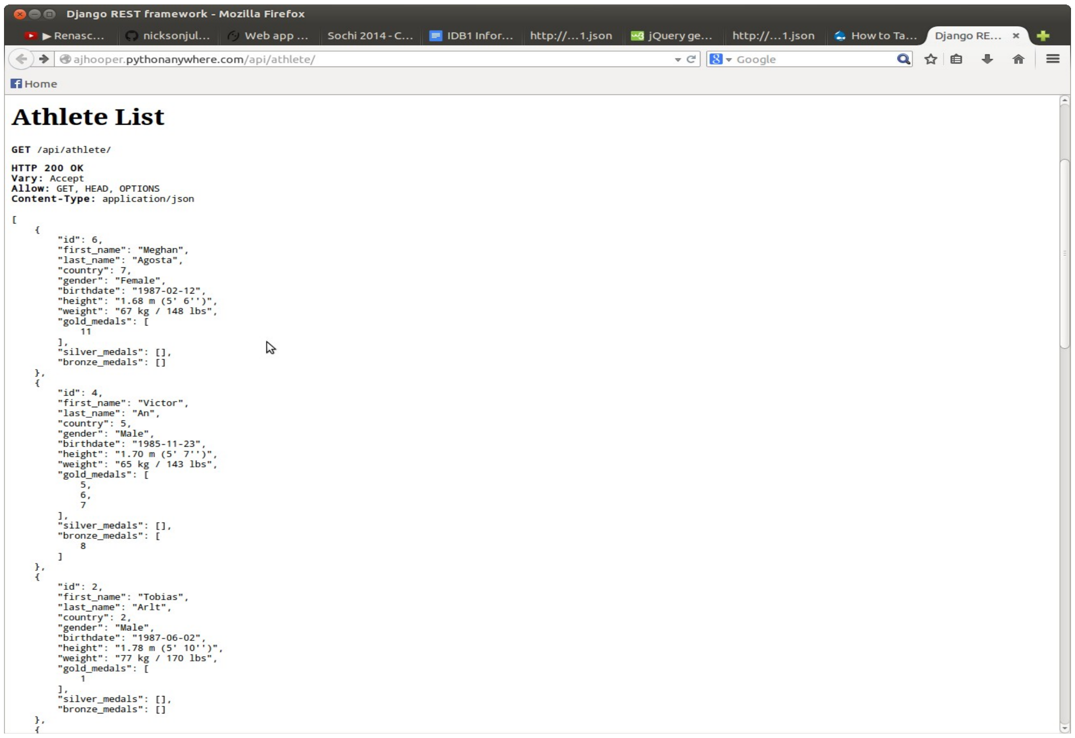


```
    #API
    url(r'^api/', include(athlete_resource.urls)),
    url(r'^api/', include(country_resource.urls)),
    url(r'^api/', include(event_resource.urls))
)
```

urls.py for Tastypie.

C. Conclusion

Eventually, we decided to use Django Rest Framework to build our API. To access our API, you have to add api in the url path. As an example, for getting the information of all countries, the url will be ajhooper.pythonanywhere.com/api/country/

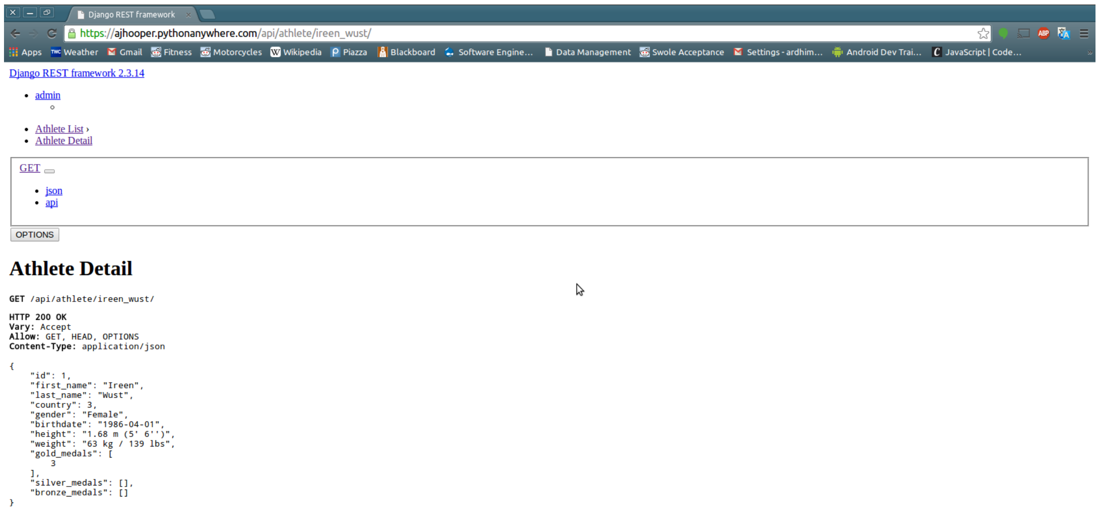

Django REST framework - Mozilla Firefox
ajhooper.pythonanywhere.com/api/athlete/
Home

Athlete List

```
GET /api/athlete/

HTTP 200 OK
Vary: Accept
Allow: GET, HEAD, OPTIONS
Content-Type: application/json

[
    {
        "id": 6,
        "first_name": "Meghan",
        "last_name": "Agosta",
        "country": 7,
        "gender": "Female",
        "birthdate": "1987-02-12",
        "height": "1.68 m (5' 6'')",
        "weight": "67 kg / 148 lbs",
        "gold_medals": [
            11
        ],
        "silver_medals": [],
        "bronze_medals": []
    },
    {
        "id": 4,
        "first_name": "Victor",
        "last_name": "An",
        "country": 5,
        "gender": "Male",
        "birthdate": "1985-11-23",
        "height": "1.70 m (5' 7'')",
        "weight": "65 kg / 143 lbs",
        "gold_medals": [
            5,
            6,
            7
        ],
        "silver_medals": [],
        "bronze_medals": [
            8
        ]
    },
    {
        "id": 2,
        "first_name": "Tobias",
        "last_name": "Arlt",
        "country": 2,
        "gender": "Male",
        "birthdate": "1987-06-02",
        "height": "1.78 m (5' 10'')",
        "weight": "77 kg / 170 lbs",
        "gold_medals": [
            1
        ],
        "silver_medals": [],
        "bronze_medals": []
    },
    {
```

https://ajhooper.pythonanywhere.com/api/country/

For getting the detailed information of a single athlete/country/event, use the id or the name of the full name athlete/country/event (separated with "_") as the last path of the url.



Django REST framework 2.3.14

- admin

- Athlete List ›
- Athlete Detail

GET

- json
- api

OPTIONS

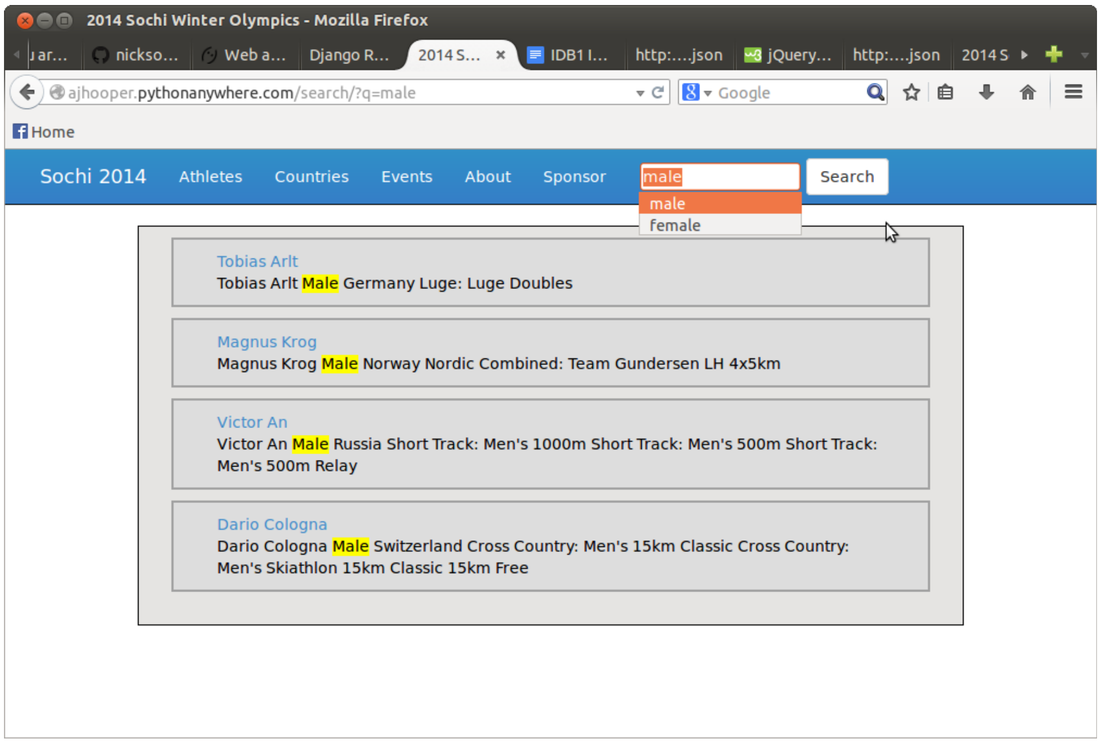
Athlete Detail

```
GET /api/athlete/ireen_wust/

HTTP 200 OK
Vary: Accept
Allow: GET, HEAD, OPTIONS
Content-Type: application/json

{
    "id": 1,
    "first_name": "Ireen",
    "last_name": "Wust",
    "country": 3,
    "gender": "Female",
    "birthdate": "1986-04-01",
    "height": "1.68 m (5' 6'')",
    "weight": "63 kg / 139 lbs",
    "gold_medals": [
        3
    ],
    "silver_medals": [],
    "bronze_medals": []
}
```

https://ajhooper.pythonanywhere.com/api/athlete/ireen_wust/

SEARCH CAPABILITY

In the third phase of the project, we successfully implemented the search bar by using Haystack and Whoosh. User can type any word to cause result boxes to appear. The result box consists of the title of the relevant pages (which is also a link to that page) and some text in the page that contains the word searched for. The word that is searched will be highlighted in the result boxes.


2014 Sochi Winter Olympics - Mozilla Firefox
ajhooper.pythonanywhere.com/search/?q=male
Home
Sochi 2014
Athletes
Countries
Events
About
Sponsor
male
Search
male
female

Tobias Arlt
Tobias Arlt Male Germany Luge: Luge Doubles

Magnus Krog
Magnus Krog Male Norway Nordic Combined: Team Gundersen LH 4x5km

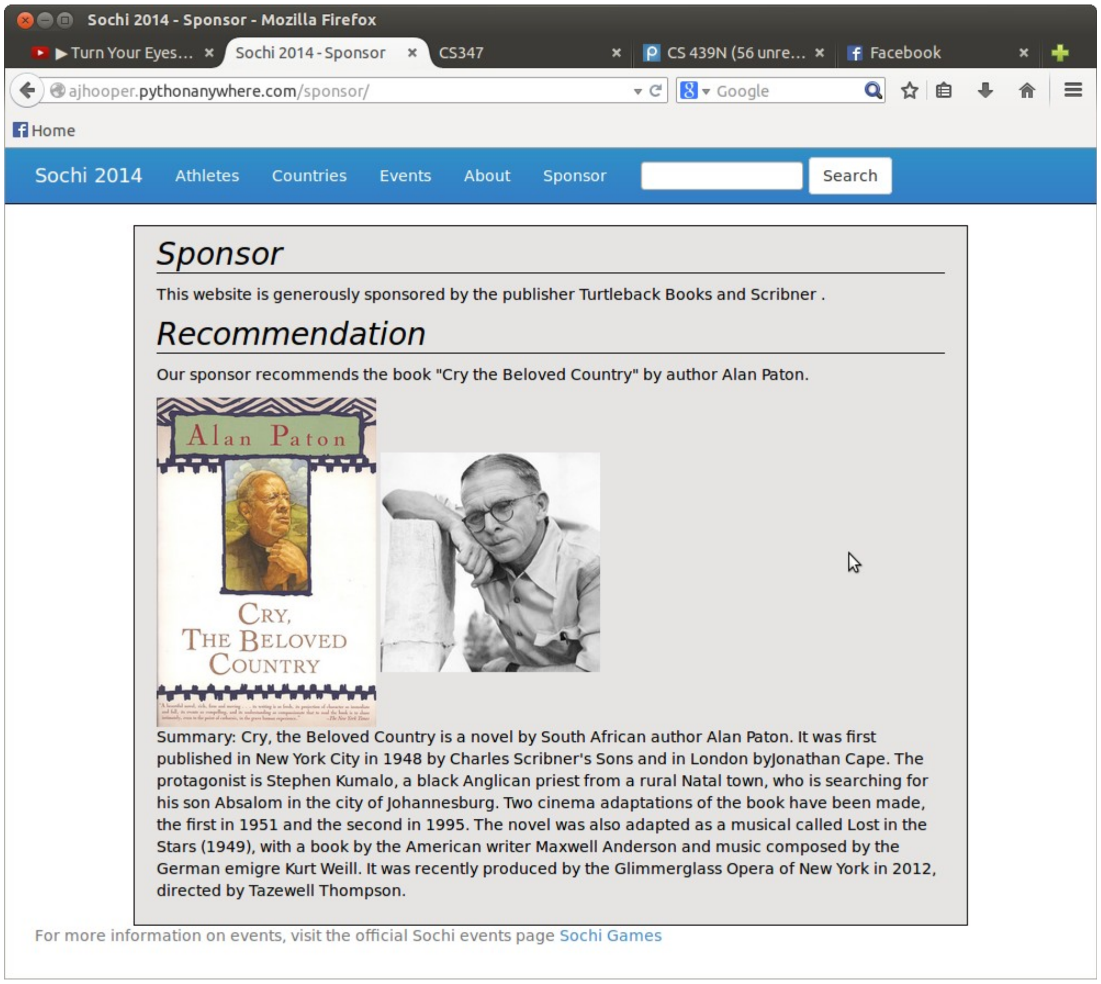
Victor An
Victor An Male Russia Short Track: Men's 1000m Short Track: Men's 500m Short Track: Men's 500m Relay

Dario Cologna
Dario Cologna Male Switzerland Cross Country: Men's 15km Classic Cross Country: Men's Skiathlon 15km Classic 15km Free

Searching for male will bring up all the male athletes' pages.

EXERCISING ANOTHER GROUP'S API

Our group has to exercise A'la Bibilotheque's API and integrate the content to our website. We decided to be creative, and turn this into a sponsor page. Every time a user clicks the sponsor page, one of the publishers from A'la Bibilotheque's site is randomly generated and acts as a sponsor for our site. The sponsor page will also display the book, author, and a brief summary of the book recommended by the publisher.



Sponsor

This website is generously sponsored by the publisher Turtleback Books and Scribner .

Recommendation

Our sponsor recommends the book "Cry the Beloved Country" by author Alan Paton.

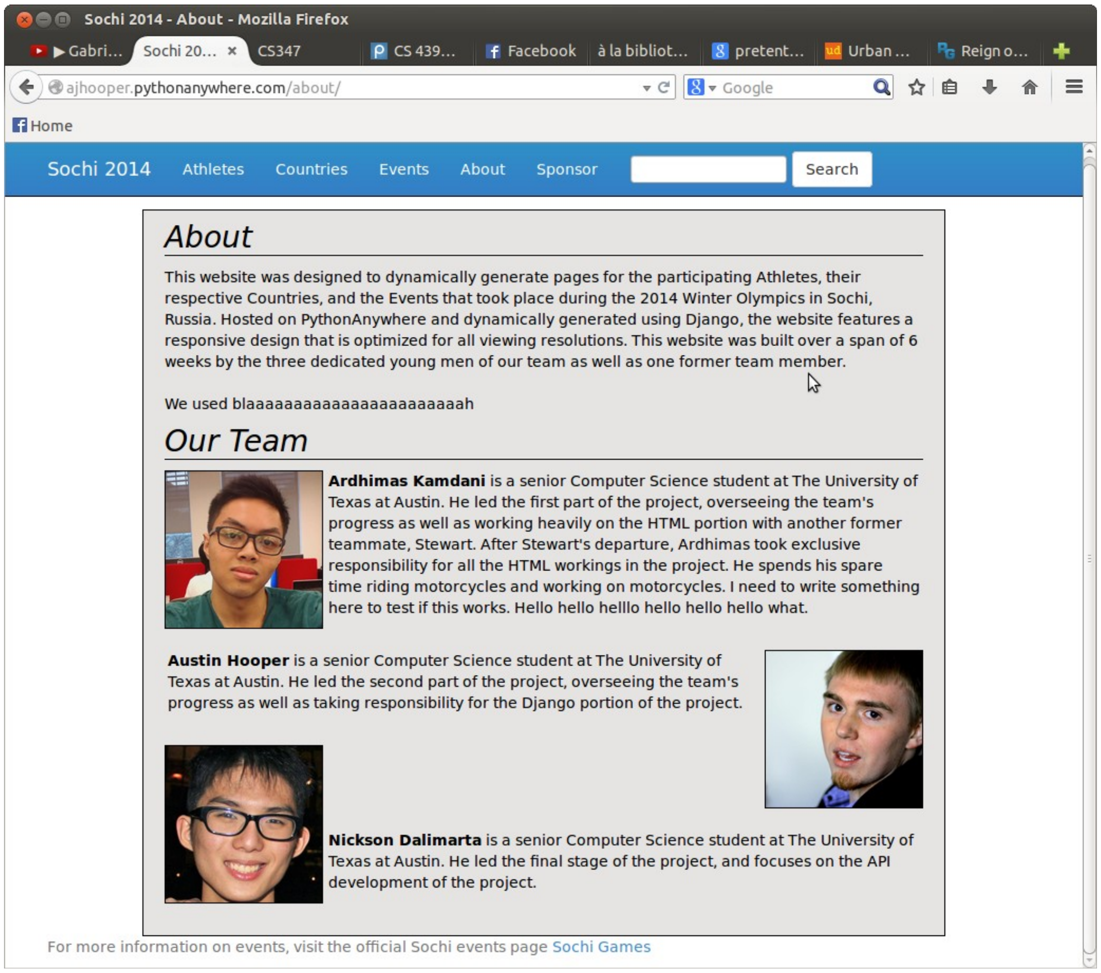
Summary: Cry, the Beloved Country is a novel by South African author Alan Paton. It was first published in New York City in 1948 by Charles Scribner's Sons and in London byJonathan Cape. The protagonist is Stephen Kumalo, a black Anglican priest from a rural Natal town, who is searching for his son Absalom in the city of Johannesburg. Two cinema adaptations of the book have been made, the first in 1951 and the second in 1995. The novel was also adapted as a musical called Lost in the Stars (1949), with a book by the American writer Maxwell Anderson and music composed by the German emigre Kurt Weill. It was recently produced by the Glimmerglass Opera of New York in 2012, directed by Tazewell Thompson.

For more information on events, visit the official Sochi events page Sochi Games

The look of our sponsor page, content is randomly generated.

ABOUT PAGE

Added for idb3 is this about page which tells user about us and each of our contributions to the site.


Sochi 2014 - About - Mozilla Firefox
ajhooper.pythonanywhere.com/about/
Sochi 2014
Athletes
Countries
Events
About
Sponsor
Search

About

This website was designed to dynamically generate pages for the participating Athletes, their respective Countries, and the Events that took place during the 2014 Winter Olympics in Sochi, Russia. Hosted on PythonAnywhere and dynamically generated using Django, the website features a responsive design that is optimized for all viewing resolutions. This website was built over a span of 6 weeks by the three dedicated young men of our team as well as one former team member.

We used blaaaaaaaaaaaaaaaaaaaaaah

Our Team

Ardhimas Kamdani is a senior Computer Science student at The University of Texas at Austin. He led the first part of the project, overseeing the team's progress as well as working heavily on the HTML portion with another former teammate, Stewart. After Stewart's departure, Ardhimas took exclusive responsibility for all the HTML workings in the project. He spends his spare time riding motorcycles and working on motorcycles. I need to write something here to test if this works. Hello hello helllo hello hello hello what.

Austin Hooper is a senior Computer Science student at The University of Texas at Austin. He led the second part of the project, overseeing the team's progress as well as taking responsibility for the Django portion of the project.

Nickson Dalimarta is a senior Computer Science student at The University of Texas at Austin. He led the final stage of the project, and focuses on the API development of the project.

For more information on events, visit the official Sochi events page Sochi Games

The look of our About page.