REST Planning and Implementation

- **Due** Feb 4 by 11:59pm
- **Points** 10
- **Submitting** a file upload

In this assignment you will plan out a REST URL structure and implement a working example of it which you will prove works by writing tests for it in much the same way as you did for GitHub. It will be up to you to decide on the appropriate URLs to use to access information. The grading will be divided evenly between using good RESTful URLs along with appropriate HTTP verbs to complete actions and the implementation of the API.

You will need to have a live, functioning server on a publicly accessible GAE instance. You will also need to turn in a PDF describing your URLs and a set of tests we can use to make sure they work as you say they do.

The information you will be modeling is a simple marina. There will be boats and slips (the `parking spots` for boats). Boats can either be 'at sea' or they can be currently in a slip. Slips are not assigned to only one boat. That is, `Boat A` can be in `Slip 1` then later, it can depart and be at sea and `Boat B` can then be in `Slip 1`.

The data is as follows:

Boat

```
{ "id":"abc123", #this should be automatically generated by your API and shoul
d probably be a string

"name": "Sea Witch", #The name of the boat, a string

"type":"Catamaran", #The type of boat, power boat, sailboat, catamaran etc. a
string

"length":28, #The length of the boat

"at_sea":false #A boolean indicating if the boat is at sea or not
}
```

Slip

```
{ "id":"123abc", #A string generated by your API

"number": 5, #The the slip number, essentially the human understandable identi
fier
```

```
"current_boat":"abc555",  #The id of the current boat, null if empty

"arrival_date":"1/1/2015", #A string indicating the date the boat arrived in t
he slip
"departure_history":[{"departure_date":"11/4/2014","departed_boat":"123aaa"}..
.] #Optional for 5% extra credit a list of the dates that previous boats depar
ted the slip
}
```

Supported functions

Your API should support the following

1. All operations on Boats and Slips
    1. Add
        1. All newly created boats should start "At sea"
        2. All newly created slips should be empty
    2. Delete
        1. Deleting a ship should empty the slip the boat was previously in
            1. The behavior of the history of deleted boats (in the extra credit option) is undefined
        2. Deleting a pier a boat is currently in should set that boat to be "At sea"
    3. Modify
    4. View
        1. You should be able to either view a single entity or the entire collections of entities, for example, I should be able to view the details of a single boat as well as get a list of all boats
        2. It should be possible, via a url, to view the specific boat currently occupying any slip.
2. Setting a boat to be "At sea"
    1. This should cause the previously occupied slip to become empty
    2. If you are doing the extra credit portion, this should cause the ship departure to be added to the slip history
    3. Setting the ship to be "At sea" and updating the slip status should happen via a single API call
3. Managing a boat arrival
    1. A ship should be able to arrive and be assigned a slip number
    2. If the slip is occupied the server should return an Error 403 Forbidden message
    3. This will require knowing the slip, date of arrival and boat

Deliverables

You need to submit a PDF which lists all URLs and what verbs work on those URLs. If an id is in a URL it should be noted in curly braces. For example GET /client/{client_id}/orders might be a URL to return a list of all orders made by the client with the ID that is substituted for {client_id}.

The format of the URLs should be:

HTTP_VERB /url/without/the/root

So if your full url is http://foobar.com/a/b You need to only include /a/b in the URL on the PDF. In addition you should mention what happens when calls are made to that URL and what data needs to be provided. A single URL may have several entries if different HTTP verbs can be used on it.

You need to supply tests that demonstrate that the above functionality works. In addition you should prove that requests which provide invalid data are ignored.

Finally you should include the source code for your project.

So you should submit a PDF and a zip that includes your test results, the tests themselves, the test environment if needed and finally your source code.

There is a fair amount of flexibility in the assignment, this means that you will almost certainly need to ask clarifying questions to make sure you are properly meeting the expectations. This is expected and you should ask these questions on Piazza so that other students may benefit from them.

This is a two week assignment.

Here is a video that demonstrates the basics of getting connected to the GAE data-store and handling a basic
request: https://media.oregonstate.edu/media/t/0_nph1lmls