API Backend Query Structure

- SMACK PRINT -

WINTER 2016

Compiled By: Kyle Derby MacInnis

Introduction

The SMACK project will incorporate a RESTful API that will be used to access the data from behind the cluster. This will act as a backend and will not be used as the front end for the end user – those who will be using the API will be the developers and grad students who wish to access data for easy modelling and simulation purposes.

The API Control will be implemented using Node.js rather than the shiny-server which will be to handle the frontend graphical presentation of the data – since the graphical presentation. While R does have built-in functionality to support retrieving data, it does a poor job of providing listening callback support for such calls. It can utilize the API driven backend and use it in conjunction with the shiny frontend to provide the graphical 'marketing' display.

The API will have two main components:

1) **API Callback Listener** - Node.js Express Library
2) **Data Retrieval Process** - Node.js Request Library

Web-Based Server Frontend will use R and Shiny's functionality to provide a clean interface quickly and easily that will make simple HTTP GET requests to the API backend for such data as well.

1) **Shiny Graphical Presentation Layer** – Shiny+Dashboard+Leaflet
2) **R Built-in Http Request** – Communicates to backend API

Since both of these services will be running on the same computer (although not mandatory) they should not be affected too poorly by networking delays.

Request Methods:

The following sections outline the various methods that will form the API framework. All the below mentioned methods are RESTful and can be easily accessed through different means.

HEAD Requests:

Returns Meta Information – In our case, we could use it to post a simple message of what that data is. For example, this would be the listing method - we could query what data we were looking at and get a header response that we could compare against and ensure we have queries correctly without transferring data – robust design.

GET Requests:

Returns Raw Data from the backend. This is most likely going to be swift storage data and will most likely be stored and accessed through swift's API. This will be used to access data directly, and this will act as a backend for our frontend Shiny server. Shiny will be used to form the graphical frontend due to its simplicity and ease of data retrieval capabilities. Shiny does not however have as good a foundation for API driven backend, so it would be best to use node.js express.

POST Requests:

Probably will not be used. (If used it may be used to update database information – store into swift)

PUT Requests:

Probably will not be used. (If used it may be used to store database information – store into swift initially)

DELETE Requests:

Probably will not be used. (If used it may be used to delete objects from swift)

OPTIONS Requests:

Returns the used API methods that are available – ie GET and HEAD.

API Query Reference

The following section outlines the specific queries that will be implemented into the API framework.

HEAD Requests:

GET Requests:

OPTIONS Requests: