CODATA – RDA Data Schools

Cyber Infrastructures

Creating Web Services

Instructor: Raphael Cobe
@raphaelmcobe / raphaelmcobe@gmail.com

CODATA – RDA Data Schools

Materials

Exercises for the Lecture

1. Warmup/Data Exploration using Google Colab;
2. Regression Analysis using Apache Spark on Colab;
3. Setting up a cloud resource using OpenStack;
4. Accessing a Virtual Machine using the SSH protocol;
5. Deploying a Jupyter Notebook Server at an IaaS platform using the conda environment manager;
6. Deploying a Jupyter Notebook Server at an IaaS platform using Docker containers

REST Applications - Properties

CODATA – RDA Data Schools

- **Client-Server**: Separation of responsibilities between user interface (UI) and data storage/manipulation.
- **Uniform Interface**: Well-defined "contract" for communication between client and server components.
- **Stateless**: Independent of state. Server doesn't store any context.

REST Applications - Properties

CODATA – RDA Data Schools

- **Cache**: Improves performance, scalability, and efficiency by reducing average response time.
- **Layers**: Architecture should be built with independently managed layers.
- A service (in the sense of SOA) that follows these principles is called **a RESTful service**.
- Ideally, RESTful URIs name the operations.

REST - Definition

- REpresentational State Transfer
- An architectural style for defining loosely coupled systems,
  - The Web (URL/HTTP/HTML/XML) is an instance of this style.
- The Web, when used appropriately (not as a transport protocol),
  - HTTP was constructed using RESTful principles.
- Most importantly, HTTP is an application protocol (not a transport protocol).

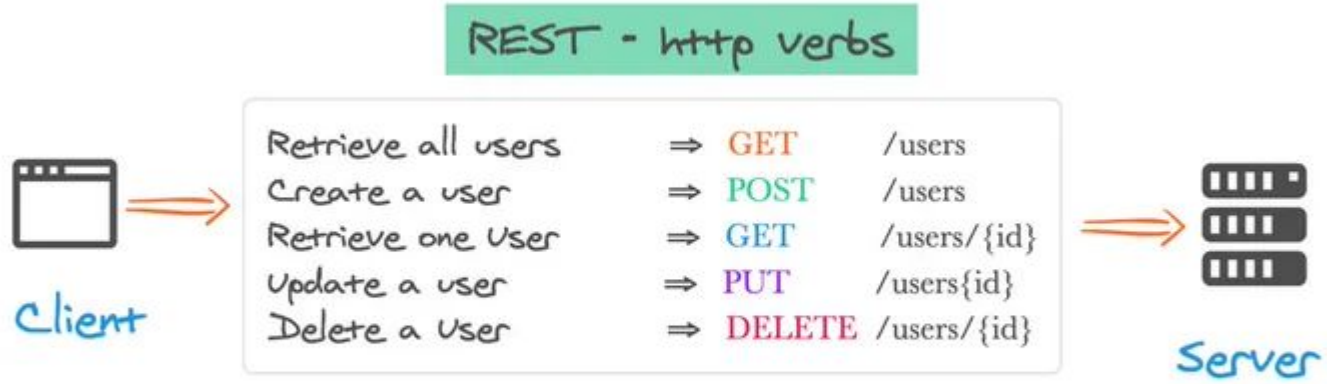
HTTP Protocol - Methods

CODATA – RDA Data Schools

- Defines a set of request methods responsible for indicating the action to be performed on a given resource.
- Commonly referred to as **HTTP Verbs**.
- **GET:** requests the representation of a specific resource.
  - Requests using the GET method should only retrieve data.
- **POST:** used to submit an entity to a specific resource, often causing a change in the state of the resource or side effects on the server.
- Other Verbs: PUT, DELETE, HEAD, OPTIONS

HTTP Protocol - Methods

- Verbs Example:



HTTP Protocol - Methods

- **Best Practices:**
  - In general, URIs should be named with nouns that specify the contents of the resource;
  - Use Clear and Consistent Names;
  - Separate Words with Hyphens;
  - Avoid Special Characters and File Extensions;
- **Bad Design**:
  - Mixcase Style;
  - Non Consistency;

HTTP Protocol - Response

- Response status codes for requests: greatly assist the consuming client in understanding the application's behavior;
  - Informational responses (100 – 199),
  - Successful responses (200 – 299),
  - Redirection messages (300 – 399),
  - Client error responses (400 – 499),
  - Server error responses (500 – 599)

HTTP Protocol - Response

- Response status codes for requests - most common ones:
- **102 Processing**: indicates that the server has received and is processing the request, but no response is available yet.
- **200 OK**: The request succeeded.
- **301 Moved Permanently**: The requested resource has been changed permanently.
- **404 Not Found**: The server cannot find the requested resource.

REST in R

- **Plumber package**;
- Creating functions:
  - Annotating functions: method, URL, and return format:

plumber.R

```
#* Echo back the input
#* @param msg The message to echo
#* @get /echo
function(msg=""){
  list(msg = paste0("The message is: '", msg, "'"))

}
```

REST in R

- Initiate an Application:

From the R prompt:

```
> library(plumber)
> r <- plumb("plumber.R") # Where 'plumber.R' is the location of
the file
> r$run(host='0.0.0.0', port=8888)
```

- Access the Application:
  - http://VM_IP_ADDRESS:8888/echo?msg=Hello World

REST in R

CODATA – RDA Data Schools

- Testing from the Command Line:

```
$ curl "http://localhost:8888/echo"
{"msg":["The message is: ''"]}

$ curl "http://localhost:8888/echo?msg=hello"
{"msg":["The message is: 'hello'"]}
```

REST in R

- Receiving parameters:
  - Using the POST method

plumber.R

```
#* Return the sum of two numbers
#* @param a The first number to add
#* @param b The second number to add
#* @post /sum
function(a, b) {
  as.numeric(a) + as.numeric(b)
}
```

REST in R

CODATA – RDA Data Schools

- Testing from the Command Line:

```
$ curl --data "a=4&b=3" "http://localhost:8000/sum"
 [7]

$ curl -H "Content-Type: application/json" --data '{"a":4, "b":5}'
http://localhost:8000/sum
 [9]
```

Creating an Inference Service

1. Load Required Library
2. Load the Pre-trained model in RDS format;
3. Read **and convert** the request parameters;
4. Perform the Inference;
5. Return the result;

Example

Example on how to save a pre-trained model