

# Assignment 1

30% of Overall Grade

The screenshot shows a REST Client window with the following details:

- HTTP method: GET
- Host/port: http://localhost:3000
- Path: /donations

The response body displays a JSON array of four donation documents:

```
[{"upvotes": 0, "_id": "5b1da70d40a09be225f767cb", "paymenttype": "PayPal", "amount": 1101, "__v": 0}, {"upvotes": 0, "_id": "5b1da72540a09be225f767cc", "paymenttype": "Visa", "amount": 1100, "__v": 0}, {"upvotes": 0, "_id": "5b1da73440a09be225f767cd", "paymenttype": "Direct", "amount": 100, "__v": 0}, {"upvotes": 0, "_id": "5b1da74340a09be225f767ce", "paymenttype": "Direct", "amount": 500, "__v": 0}]
```

Below the JSON, the status bar shows:

- Response code: 200 (OK); Time: 27ms; Content length: 654 bytes
- REST Client
- Run
- TODO
- Version Control
- Terminal

A message at the bottom states: "Plugin Error: Problems found loading plugins: // The following plugins are incompatible with the current IDE build: Markdown (today 14:22)".

# Options

---

Work on your own app, exhibiting similar level of complexity/feature density as covered in the 1<sup>st</sup> part of the Semester Case Study - Donation.

# Case Study - Donation

---

- A Node Web Server to manage donations made to ‘Homers Presidential Campaign ’.
- App Features (all via RESTful API)
  - POST a payment type and donation amount in JSON format
  - GET a list of donation amounts and types
  - GET an individual donation using an ID
  - DELETE an individual donation using and ID
  - Upvote a donation via PUT request
- Persistence via MongoDB deployed to Heroku

# POST – Request & Response

46 | `console.log('ID: ' + req.params.id)`

REST Client

HTTP method: POST Host/port: http://localhost:3001 Path: /donations

Request Cookies Response Response Headers

Headers

Accept: application/json  
Cache-Control: no-cache  
Content-Type: application/json

Request Param

Specify the text to send:

1 {"id":0,"paymenttype":"Direct","amount":500,"upvotes":0}

4b | `console.log('ID: ' + req.params.id)`

REST Client

HTTP method: POST Host/port: http://localhost:3001 Path: /donations

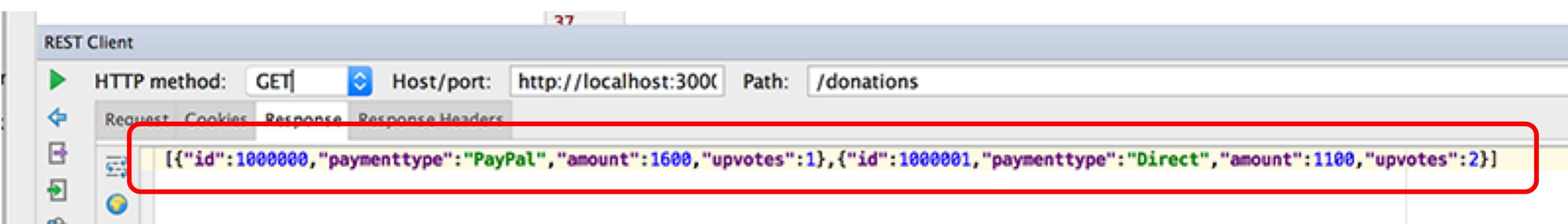
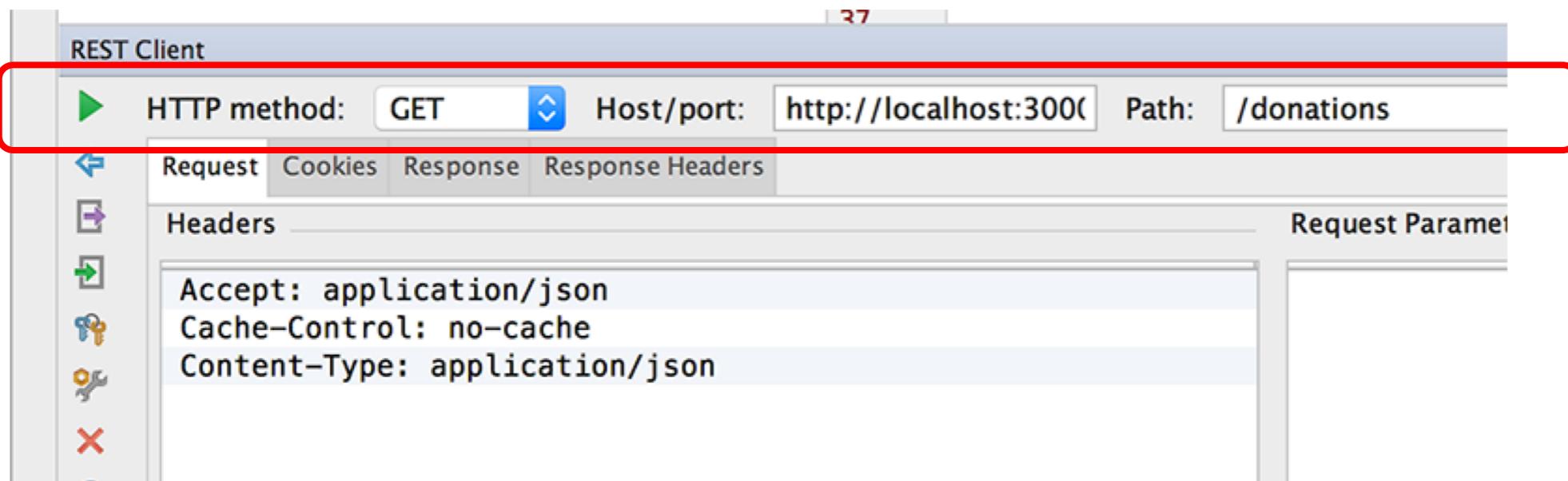
Request Cookies Response Response Headers

{"message": "Donation Added!"}

Cancel OK

The image shows a REST Client interface with two main sections. The top section is labeled '46' and contains a code snippet: `console.log('ID: ' + req.params.id)`. Below it, the 'Request' tab is selected, showing the configuration: HTTP method: POST, Host/port: http://localhost:3001, and Path: /donations. The 'Headers' section includes Accept: application/json, Cache-Control: no-cache, and Content-Type: application/json. A modal window titled 'Specify the text to send:' is open, containing the JSON data: `{"id":0,"paymenttype":"Direct","amount":500,"upvotes":0}`. The bottom section is labeled '4b' and contains a similar code snippet: `console.log('ID: ' + req.params.id)`. The 'Response' tab is selected here, showing the returned message: {"message": "Donation Added!"}. There are 'Cancel' and 'OK' buttons at the bottom right of the modal.

# GET (1) – Request & Response



# GET (2) – Request & Response

The image displays three separate instances of a REST Client tool, each showing a GET request and its corresponding response.

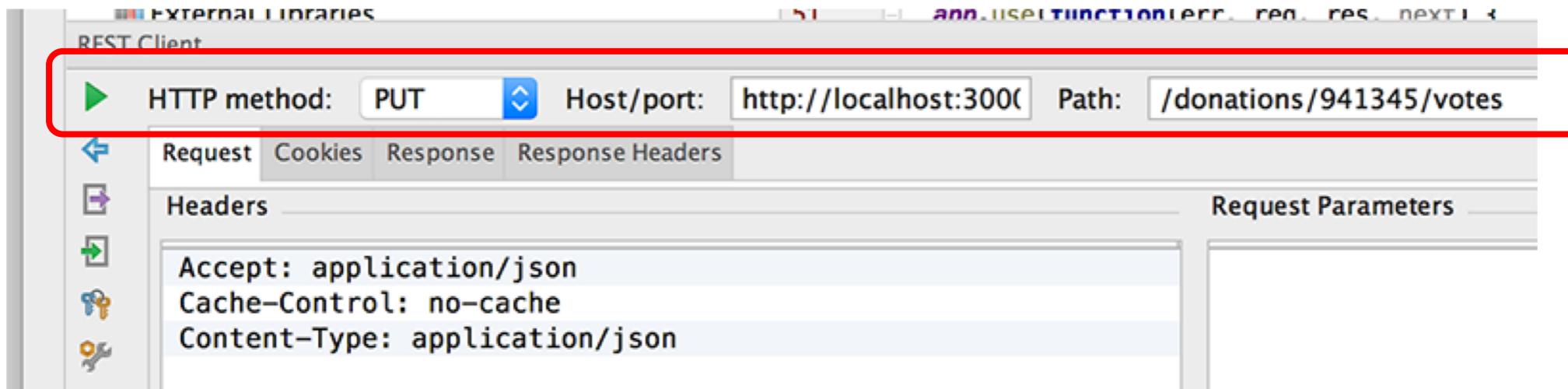
- Top Screenshot:** Shows a GET request to `http://localhost:3000/donations/1000001`. The request headers include `Accept: application/json`, `Cache-Control: no-cache`, and `Content-Type: application/json`. The response is a JSON object: `{"id":1000001,"paymenttype":"Direct","amount":1100,"upvotes":2}`.
- Middle Screenshot:** Shows a GET request to `http://localhost:3000/donations/1000001`. The response is a JSON object: `{"message":"Donation NOT Found!"}`.
- Bottom Screenshot:** Shows a GET request to `http://localhost:3000/donations/1000001`. The response is a JSON object: `{"id":1000001,"paymenttype":"Direct","amount":1100,"upvotes":2}`.

# DELETE – Request & Response

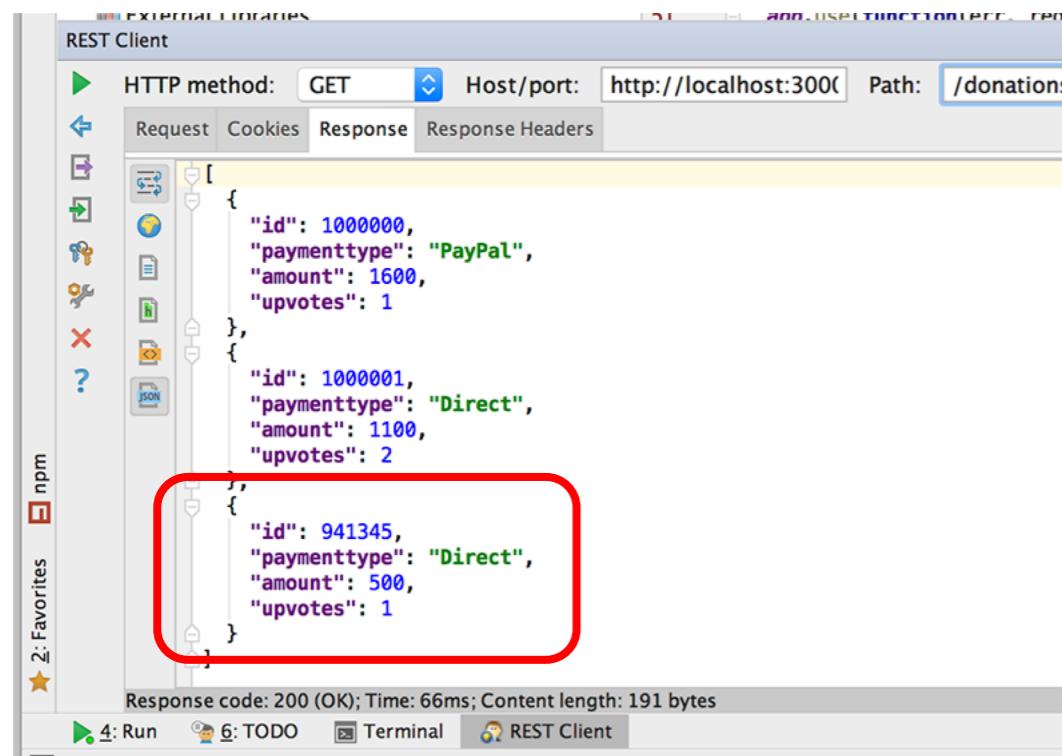
The image shows four panels from a REST Client interface, each demonstrating a different aspect of a DELETE operation.

- Top Left Panel:** Shows a configuration for a DELETE request. The "HTTP method" is set to "DELETE", the "Host/port" is "http://localhost:3000", and the "Path" is "/donations/1000001". This panel is highlighted with a red box.
- Top Right Panel:** Shows the response headers for the DELETE request. The "Content-Type" header is listed as "application/json".
- Middle Left Panel:** Shows the response body of the DELETE request, which is a JSON object: {"message": "Donation Deleted!"}. This panel is highlighted with a red box.
- Middle Right Panel:** Shows the state of the donations collection before the delete operation. It lists three documents with IDs 1000000, 1000001, and 941345.
- Bottom Left Panel:** Shows the state of the donations collection after the delete operation. The document with ID 1000001 is missing, leaving only two documents with IDs 1000000 and 941345.
- Bottom Right Panel:** Shows the response code and content length of the DELETE request. The message "Response code: 200 (OK); Time: 66ms; Content length: 191 bytes" is displayed.

# PUT – Request & Response



- Adds 1 to 'upvotes'



# Assignment Rubric for Assignment 1

Standard	CRUD Node Server [70%]	Model [10%]	Persistence [10%]	DX (Developer eXperience) [10%]
Baseline	> 2 GET routes	1 Basic Model	Basic JS Persistence	Data Validation
Good Pass line	2 GET routes 1 POST route  1 PUT route 1 DELETE route	1 Complex Model of different types	MongoDB Persistence	Adherence to JS Best Practices eg SoC, Design
Very Good	> 3 GET routes > 2 POST route > 2 PUT route > 2 DELETE route	2 Complex Models with Schema	MongoDB Persistence with Schema	Automated Testing (models)
Excellent/ Outstanding (70%+)	Additional Features included, eg fuzzy searches, authentication etc.	> 3 Models with Schema & related to each other	Advanced Features eg. deployed, authentication	Repo Usage, git etc.

# README file

---

Include a brief README file (max 2 - 3 pages):

- Name and Student ID.
- Brief description of functionality.
- Persistence approach adopted i.e. what's persisted and where.
- Git approach adopted and link to git project / access.
- DX approach adopted.
- References

# Submitting Project Deliverables

---

Submit zip of project via Moodle dropbox. This zip should also include:

- the README file and
- full source code of your web project
- Youtube link to video (5 – 10 mins MAX) of Server Testing

Give read access to your lecturer to your GitHub / BitBucket repos. GitHub and BitBucket ids are:

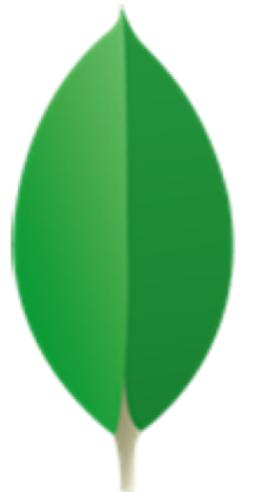
- ddrohan.

# Questions?

---



**mongoose**



**mongoDB**