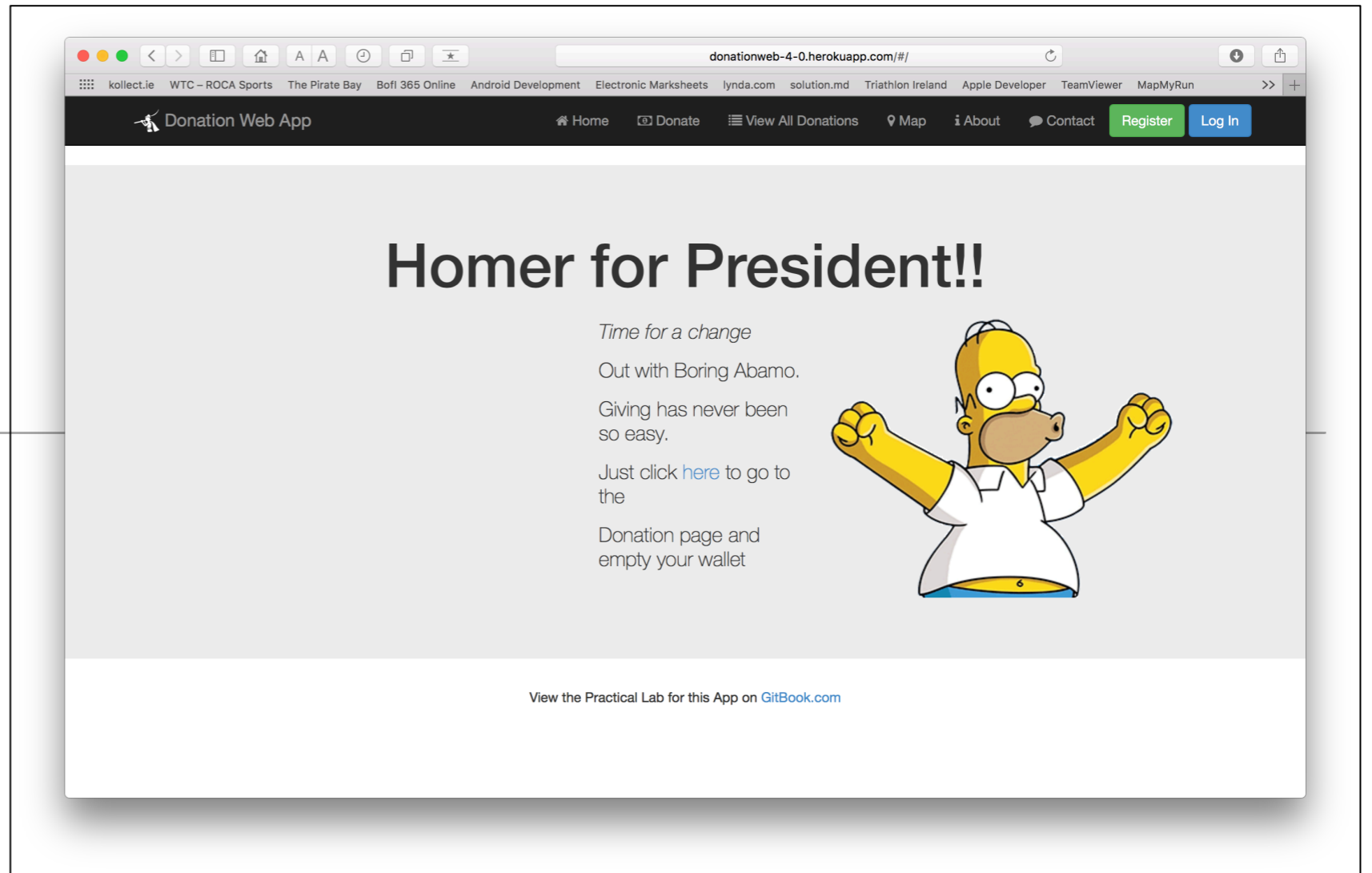


# Assignment 1

30% of Overall Grade



# Options

---

Work on your own app, exhibiting similar level of complexity/feature density as covered in the 1<sup>st</sup> half 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

# POST – Request & Response

The image displays two screenshots of a REST Client interface, illustrating a POST request and its response.

**Top Screenshot (Request):**

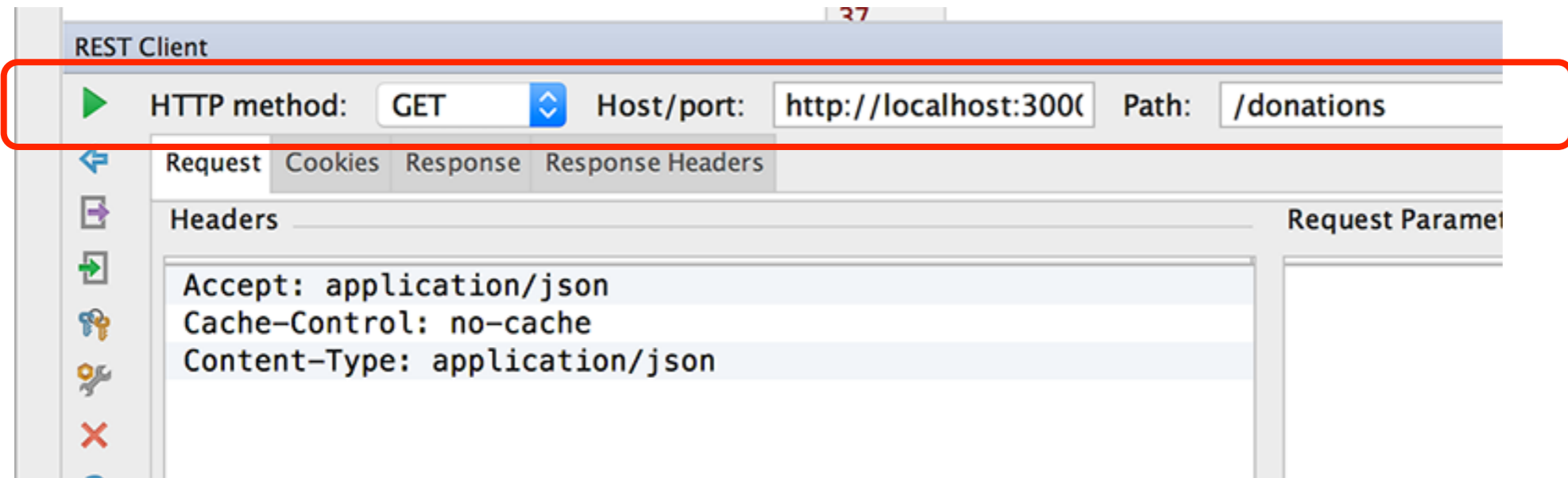
- HTTP method:** POST
- Host/port:** http://localhost:3000
- Path:** /donations
- Request Headers:**
  - Accept: application/json
  - Cache-Control: no-cache
  - Content-Type: application/json
- Request Parameters:** (Empty)
- Request Body:** {"id":0,"paymenttype":"Direct","amount":500,"upvotes":0}

**Bottom Screenshot (Response):**

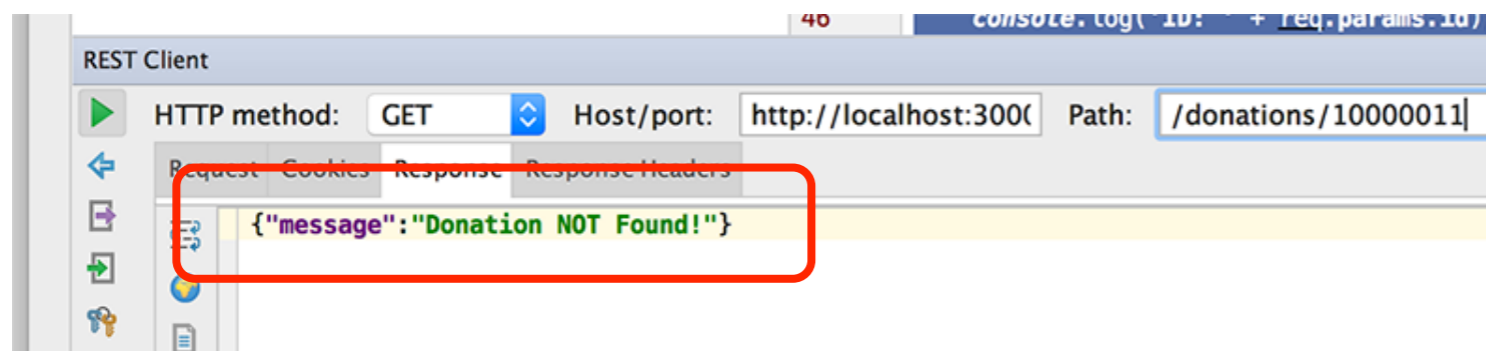
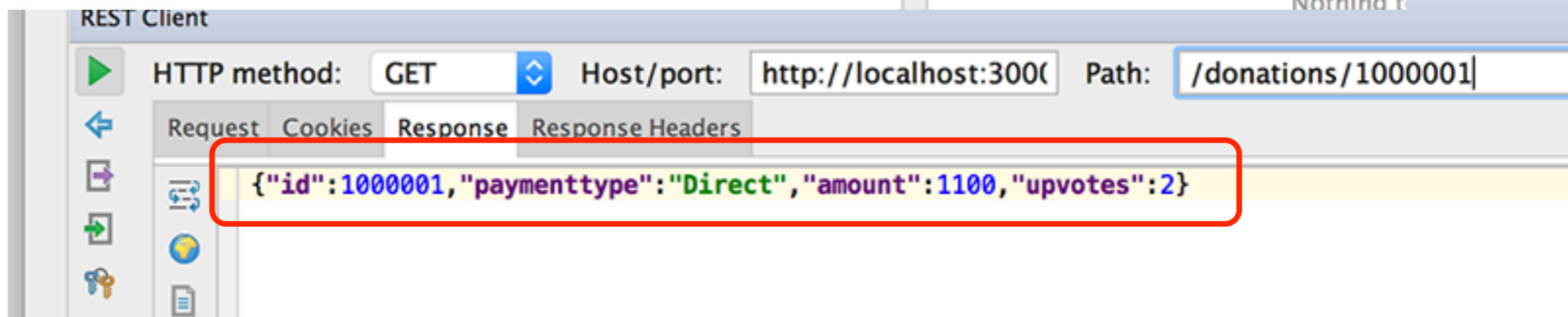
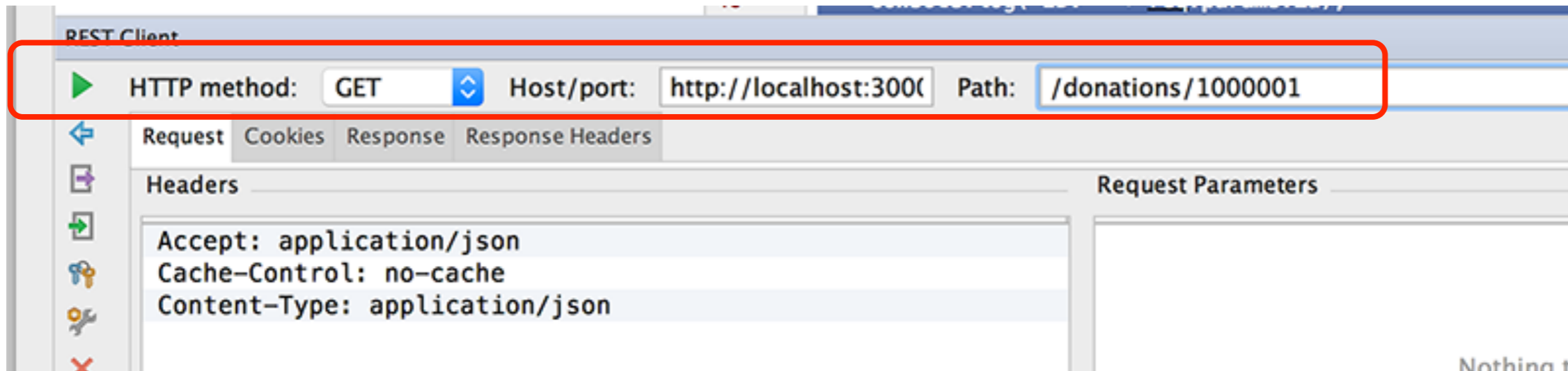
- HTTP method:** POST
- Host/port:** http://localhost:3000
- Path:** /donations
- Response Body:** {"message":"Donation Added!"}

Both screenshots show a console log entry: `console.log('ID: ' + req.params.id)`. The top screenshot shows a log entry with the value 46, and the bottom screenshot shows a log entry with the value 40.

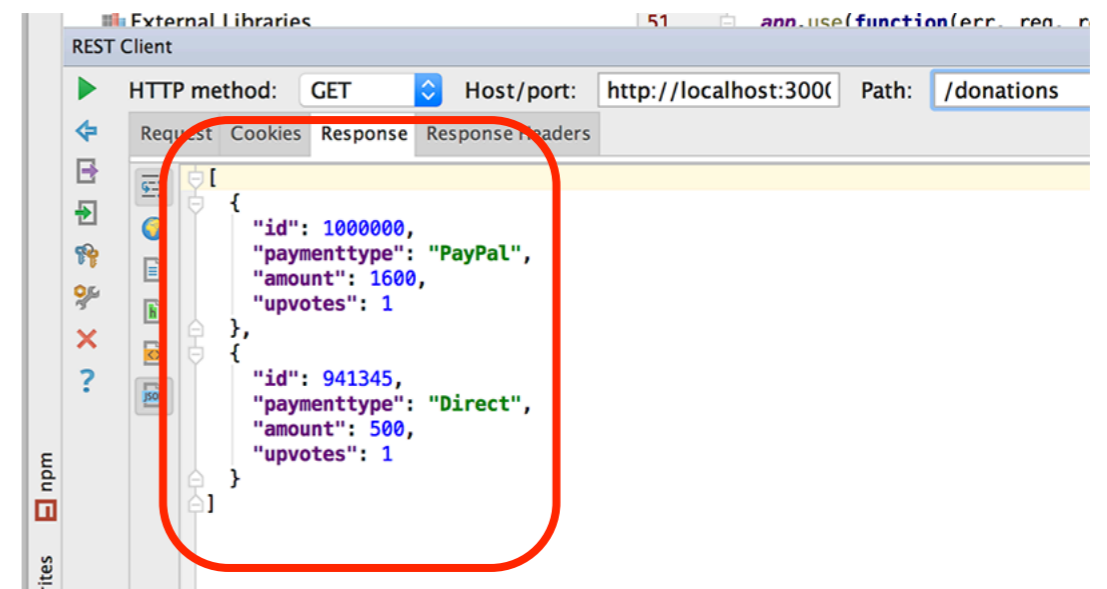
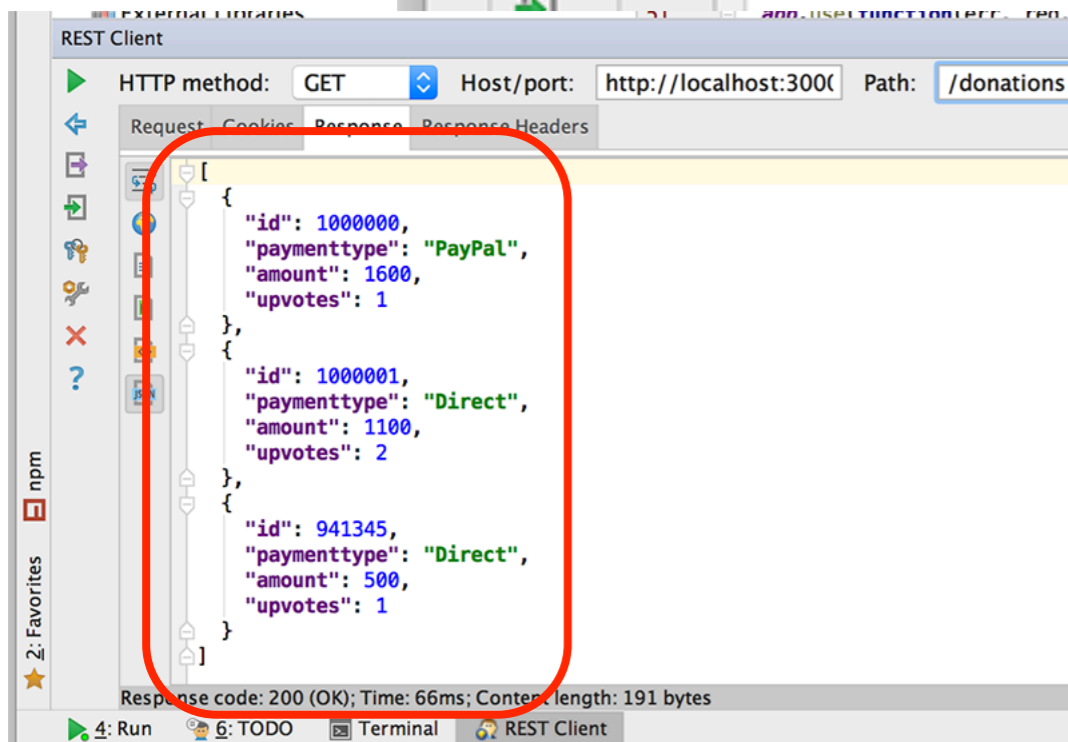
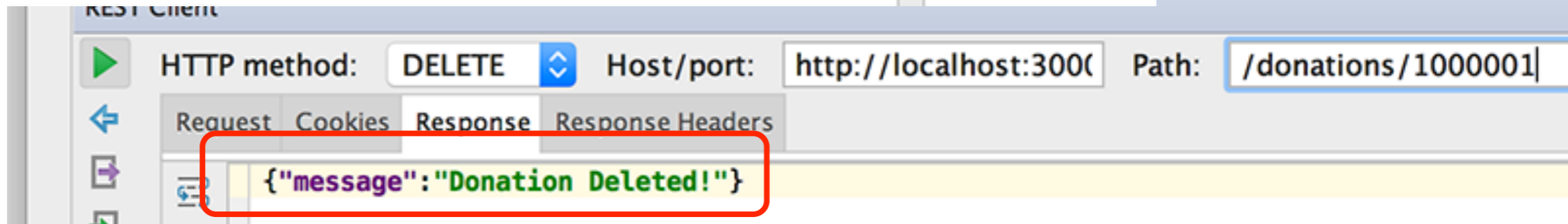
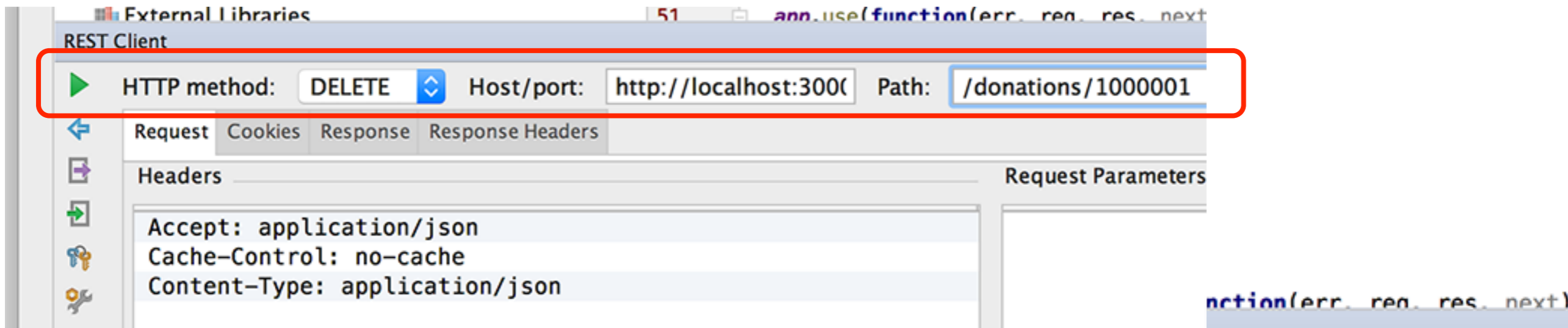
# GET (1) – Request & Response



# GET (2) – Request & Response

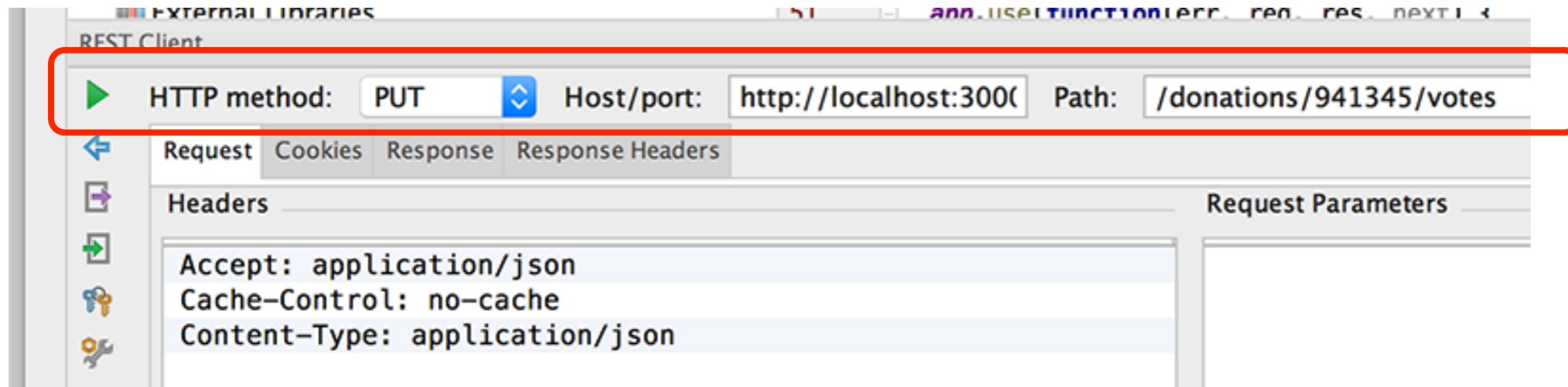


# DELETE – Request & Response

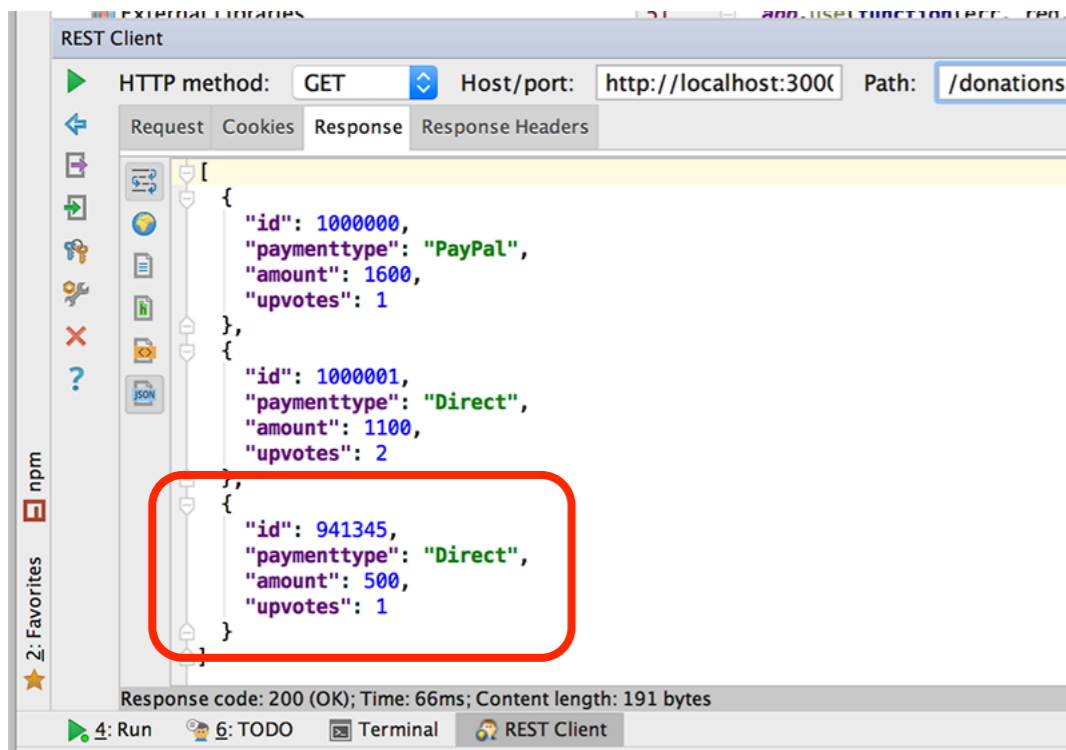




# PUT – Request & Response



- 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 <b>Pass line</b>	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	Advanced Features eg. deployed, authentication	Repo Usage, git etc.

# README file

---

Include a VERY brief README file (max two 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 Code

---

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

- the README file and
- full source of your web project

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

- **ddrohan.**

# Questions?

---



mongoose



mongoDB



Except where otherwise noted, this content is licensed under a [Creative Commons Attribution-NonCommercial 3.0 License](http://creativecommons.org/licenses/by-nc/3.0/).

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

