Best Practices on Building RESTful API

Nikola Vasilev

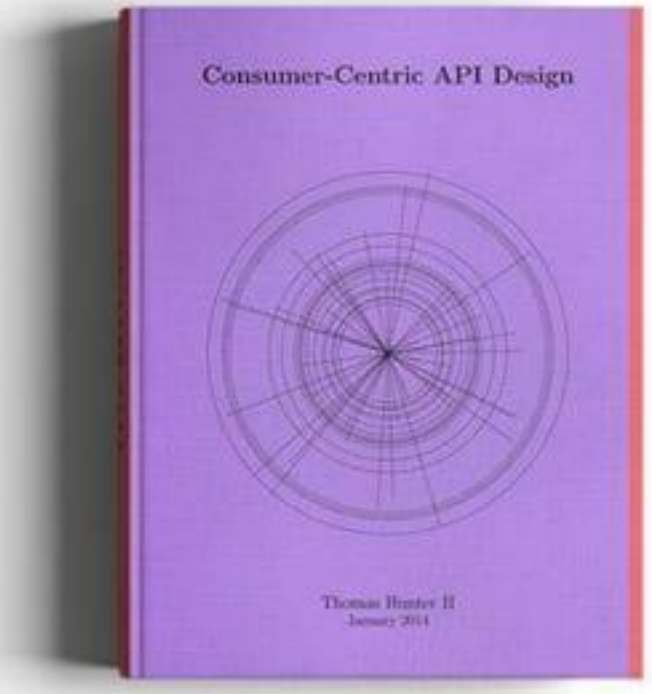
Thursday, 1st of December 2016

Tricode BV
De Schutterij 12 -18
3905 PL Veenendaal
The Netherlands

tel: 0318 - 559210
fax: 0318 - 650909
www.tricode.nl
info@tricode.nl

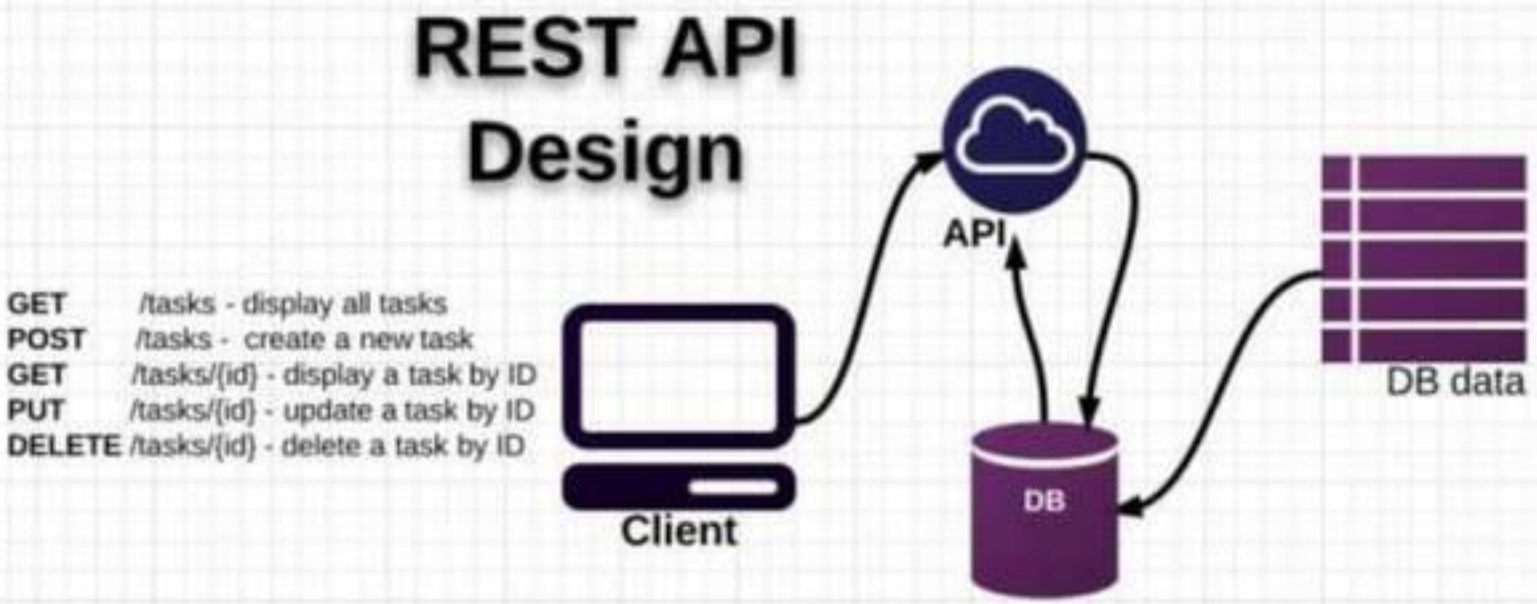
Agenda

- Introduction
- Data Design and Abstraction
- Verbs
- Endpoints
- Request
- Response
- Demo
- Security
- Documentation
- References


Consumer-Centric API Design
Thomas Hunter II
January 2014

Introduction

- What is REST?
- What is a RESTful API?



Introduction

- Good RESTful API design is hard!
- Language Agnostic Approach

Data Design and Abstraction

- API First Development
- Attaching an API to an existing project
- Don't expose the whole functionality via API

{A}1ST

Verbs

- **GET (SELECT):** Retrieve a specific Resource from the Server, or a listing of Resources.
- **POST (CREATE):** Create a new Resource on the Server.
- **PUT (UPDATE):** Update a Resource on the Server, providing the entire Resource.
- **PATCH (UPDATE):** Update a Resource on the Server, providing only changed attributes.
- **DELETE (DELETE):** Remove a Resource from the Server.

API Root URL

- The root location of your API is important.
- The API Root URL needs to be as simple as possible:
- Provide a list of all endpoints on the root url.
- Simple endpoints:
  - https://api.github.com/
  - https://graph.facebook.com
  - https://api.example.com/v1
  - https://yourproduct.com/api/v2

Endpoints

- **Use plural nouns:**
  - https://api.example.com/v1/**employees**
  - https://api.example.com/v1/**departments**
  - https://api.example.com/v1/**employees**
- **Use uniform endpoint for each functionality**
- **Don't use verbs:**
  - https://api.example.com/v1/**add_employee**
  - https://api.example.com/v1/**edit_employee**
  - https://api.example.com/v1/**delete_employee**

Endpoints (2)

- **GET /employees**: List all Employees (ID and Name, not too much detail)
- **POST /employees**: Create a new Employee
- **GET /employees/EID**: Retrieve an entire Employee object
- **PUT /employees/EID**: Update an Employee (entire object)
- **PATCH /employees/EID**: Update an Employee (partial object)
- **DELETE /employees/EID**: Delete an Employee

Response

- **GET /employees**: Return a listing (array) of Employees
- **GET /employees/EID**: Return an individual Employee
- **POST /employees**: Return the newly created Employee
- **PUT /employees/EID**: Return the complete Employee
- **PATCH /employees/EID**: Return the complete Employee
- **DELETE /employees/EID**: Return an empty document

Status Codes

- **200 OK** – [GET/PUT/PATCH] The Consumer requested data from the Server, and the Server found it for them (Idempotent)
- **201 CREATED** – [POST] The Consumer gave the Server data, and the Server created a resource
- **204 NO CONTENT** – [DELETE] The Consumer asked the Server to delete a Resource, and the Server deleted it
- **400 BAD REQUEST** – [POST/PUT/PATCH] The Consumer gave bad data to the Server, and the Server did nothing with it (Idempotent)
- **404 NOT FOUND** – [GET/PUT/PATCH/DELETE] The Consumer referenced a nonexistent Resource or Collection, and the Server did nothing (Idempotent)
- **500 INTERNAL SERVER ERROR** – [*] The Server encountered an error, and the Consumer has no knowledge if the request was successful

Content Type

- JSON

```
{
  "id": 12,
  "firstName": "John",
  "lastName": "Doe",
  "dateOfBirth": "1987-12-26",
  }
```

- XML

```
<?xml version="1.0 encoding="UTF-8"?>
<employee>
          <id>12</id>
          <firstName>John</fristName>
          <lastName>Doe</lastName>
          <dateOfBirth>1987-12-28</dateOfBirth>
</employee>
```

It’s Time For

Demo

Versioning

- No matter how the API has been built. It will be change by time.
- A good mechanism for versioning the API should be introduced.
- The old version for the existing customers needs to be kept.
- The new customers will implement the new version.
- Introduce deprecation notice of your api
  - https://api.yourdomain.com/v1
  - https://api.yourdomaincom/v2

Authentication

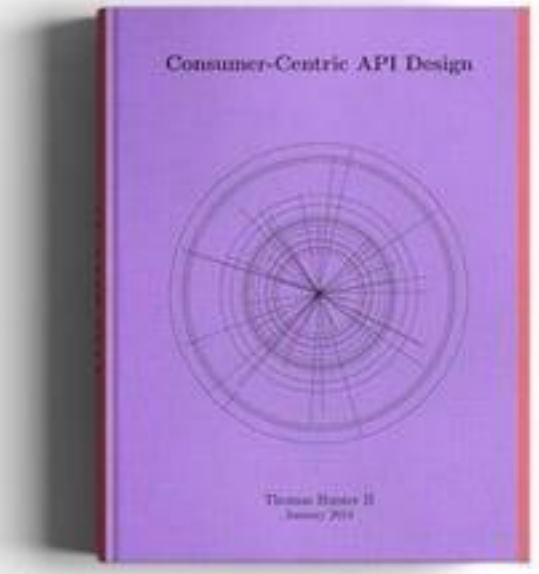
- Secure your API
- Build a customer token and use Basic Authorization over SSL
- OAuth2

Documentation

- No Documentation? - No one will know how to use your API.
- Make the documentation available publicly (Google needs to know about it)
- Document each endpoint, with each action, every response possible.
- Build developer API console if possible.

References

- Blog: https://codeplanet.io/principles-good-restful-api-design/
- Ebook: https://github.com/tlhunter/consumer-centric-api-design
- Hardcopy: https://www.amazon.com/Consumer-Centric-API-Design-Thomas-Hunter/dp/136498900X/



Follow us:

tricode.nl
facebook.com/tricode
linkedin.com/company/tricode
slideshare.net/tricode
twitter.com/tricode