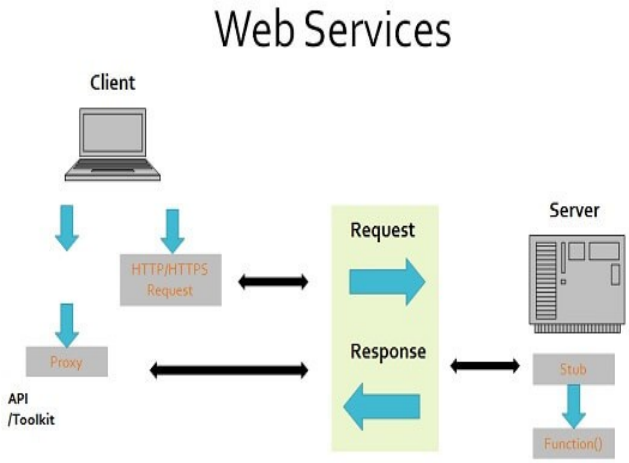
1. Pengenalan Web Service



sumber: dicoding.com

1.1. API vs WebServices

Application Programming Interface (API) dapat dibuat tanpa perlu dideploy dalam sebuah web server tetapi Web Service (W/S) harus dideploy dalam web server.
W/S merupakan kumpulan dari fungsi-fungsi tertentu dengan protocol http/https untuk pertukaran data antar sistem ataupun aplikasi. Kenapa membuat W/S ? Karena dengan adanya W/S maka kita dapat menjalankan fungsi-fungsi dari kode yang ada via http/s request baik fungsi untuk mengeksekusi program tertentu di sebuah server ataupun fungsi untuk mengakses data tertentu tanpa harus terlibat lebih dalam aturan-aturan dalam sistem/server tersebut. Tiap web service merupakan jenis dari API tapi tidak semua API adalah W/S. Semula W/S terdiri atas SOAP dan XML, namun pada perkembangannya terdapat Restful W/S. Restful W/S menggunakan format JSON (Javascript Object Notation) yang menjadi lebih populer karena lebih flexible dan simple. Untuk selanjutnya kita akan berfokus pada Resful W/S.

sumber: nesabamedia.com (contoh penerapan W/S)

1.2. Restful WebServices

Representational State Transfer (Restful) pertama kali dikenalkan oleh Roy Fielding pada tahun 2000, yang merupakan ‘gaya’ dari dalam men-design dan mengembangkan arsitektur web. REST memberikan aturan-aturan tentang bagaimana arsitektur dari sebuah sistem yang *scalable* untuk berperilaku. Arsitektur yang dimaksud mempunyai 6 aturan:

1. Client–Server

   Separation of Concern (SoC): memisahkan concern antara UI dan storage
2. Statelessness

   Tidak adanya session antara client dan server
3. Cacheability

   Response dapat di*cache* jika diperlukan oleh client sesuai dengan aturan dari server
4. Layered system

   Client tidak perlu tahu apakah dia terhubung dengan *end server* secara langsung atau via proxy/load balancer. Hal ini tidak berpengaruh pada komunikasi antara client dan server
5. Code on Demand (optional)

   Server dapat mengembangkan fungsi dengan cara mengirimkan kode lain untuk dieksekusi oleh client, semisal kode javascript jika diperlukan.
6. Uniform Interface
   - Identification of resources (URL is uniq, resource has no relationship with response eg. format)
   - Resource Manipulations through representation (for example a http code 201, 200, 400 so clients know)
   - Self Descriptive Messages (readable request for server)
   - Hypermedia as the engine of application state (response can contain a link to point more detail info)

1.3. Instilah Dasar

Resources: Data yang disajikan menjadi response
Base URI (Uniform Resource Identifier). Contoh: http://api.poltek.com
Standar HTTP method. Contoh: GET, POST, PUT, and DELETE

- GET retrieves resources
- POST submits new data to the server.
- PUT updates existing data.
- DELETE removes data.

Endpoint: Path/url dari sebuah W/S. Contoh: /about
Clean URL: SEO friendly

http://api.poltek.com/about	http://api.poltek.com/about
http://api.poltek.com/user ?id=1	http://api.poltek.com/user/1
http://api.poltek.com/index ?page=name	http://api.poltek.com/name
http://api.poltek.com/kb/index ?cat=1&id=23	http://api.poltek.com/kb/1/23
http://en.wikipedia.org/w/index ?title=Clean_URL	http://en.wikipedia.org/wiki/Clean_URL

2. Perancangan Web Service

Dalam merancang web service, setidaknya ada 8 panduan:

1. JSON format

Requesting: GET curl -i http://127.0.0.1:5000/books

HTTP/1.0 200 OK

Content-Type: application/json

Content-Length: 264

content-type: application/json

Server: Werkzeug/0.16.1 Python/3.8.10

Date: Wed, 09 Mar 2022 15:45:15 GMT


```
[
  {
    "title": "OSX for 99"
  },
  {
    "title": "SQL for kids"
  },
  {
    "title": "Titanic 3"
  }
]
```


Sending Data POST

```
curl -i -X POST http://127.0.0.1:5000/books/create \
    -H 'Content-Type: application/json' \
    -d '{"title":"Buku Baru"}'
```

2. Gunakan kata benda pada Endpoint (Mencerminkan Entity)

Contoh: GET http://127.0.0.1:5000/books, VS http://127.0.0.1:5000/list_books

Alasannya adalah karena dalam http metod sudah menggunakan kata kerja verbs sehingga kata kerja pada endpoint hanya redundancy

Cobalah pada terminal anda untuk request

```
curl https://api.github.com/users/defunk
```

```
curl -i https://api.github.com/repos/twbs/bootstrap
```

3. **Endpoint Bersarang**

Contoh: GET http://127.0.0.1:5000/books/1/reviews VS http://127.0.0.1:5000/books?action=show_reviewss&id=1

4. **Penanganan Error dan kode status http**

400 Bad Request – This means that client-side input fails validation.

401 Unauthorized – This means the user isn't not authorized to access a resource. It usually returns when the user isn't authenticated.

403 Forbidden – This means the user is authenticated, but it's not allowed to access a resource.

404 Not Found – This indicates that a resource is not found.

```
curl -i http://127.0.0.1:5000/books/Titanic%203_XYZ

HTTP/1.0 404 NOT FOUND

Content-Type: application/json

Content-Length: 34

Server: Werkzeug/0.16.1 Python/3.8.10

Date: Wed, 09 Mar 2022 16:08:26 GMT

{

  "msg": "Book cant be found"

}
```

500 Internal server error – This is a generic server error. It probably shouldn't be thrown explicitly.

502 Bad Gateway – This indicates an invalid response from an upstream server.

503 Service Unavailable – This indicates that something unexpected happened on server side (It can be anything like server overload, some parts of the system failed, etc.).

5.**Mendukung fungsi filter, sorting dan paging**

Contoh:

GET http://127.0.0.1:5000/books?tipe=hadist&author=albani

GET http://127.0.0.1:5000/books?page=2

6. **Mengimplementasikan security**

    1. **Protocol https**

    2. **Http auth basic**

    Jalankan: python security_authbasic.py

curl -u john:hello http://127.0.0.1:7002/

3. **Konsep Token**

Jalankan: python security_token.py

curl -H "Authorization: Bearer secret-token-1" http://127.0.0.1:7001/

4. **Konsep Role User**

Dengan mengetahui role dari seorang user maka perilaku W/S dapat diatur sesuai dengan kewenangannya masing-masing. Contoh: user dengan level operator harusnya tidak bisa menghapus data master ketika mencoba mengakses endpoint hapus data.

7. **Cache data**

https://flask-caching.readthedocs.io/en/latest/

Inti dari caching adalah untuk menghindari server melakukan query, ataupun mengakses resource besar yang sama untuk **meresponse** secara berulang-ulang sehingga dikuatirkan mengganggu performance dari server itu sendiri. Caching sangat *tricky* dalam hal: batasan caching baik dari size data ataupun kapan harus expires, dan apakah sebuah resource/data memang "layak" untuk disimpan dalam cache atau tidak yang terutama dipengaruhi oleh seberapa sering data itu berubah.

8. **Versioning our APIs**

http://127.0.0.1:5000/api/v1/books
http://127.0.0.1:5000/api/v2/books ?page=2
Tujuan utama dari versioning API adalah untuk memberikan dukungan "versi lama" vs "versi baru" terutama jika API tersebut sudah dipublish dan digunakan banyak pihak. Jangan sampai perubahan API menjadikan para *consumer* menjadi *crash* ataupun *error*. Contoh diatas menjelaskan bahwa API versi 1 tidak mendukung *paging* sedangkan versi 2 sudah mendukung paging. Dengan adanya 2 versi, maka dapat memberikan pilihan apakah aplikasi yang menggunakan *endpoint* tersebut ingin meng-*upgrade* untuk mengakses daftar buku secara terbagi per-halaman atau tetap pada versi 1 dengan menampilkan seluruh daftar buku.

3. Playing Web Service

Pada dasarnya dalam mengirim data ketika mengakses ataupun membuat webservices tergantung pada spesifikasi yang ditentukan karena dalam request data dapat diletakan dibeberapa tempat sesuai spesifikasi sehingga jika dokumentasi teknis sebuah endpoint hanya menerima request dalam body dengan tipe json maka biasanya data parameter tidak akan diterima. Meskipun tidak ada ketentuan khusus dapat melakukan kombinasi. Misal kombinasi request dengan data embeded pada URL dan sekaligus data request pada body:

https://api.harber.io/api/v3/pull ?type=’student’

{

ids: “[53,3535,64,633]”

}

Dalam prakteknya setidaknya ada 4 cara meletakkan data dalam request ketika membuat sebuah endpoint.

1. Data parameter dalam URL
2. Data parameter dalam request
3. Data dalam body request misal dengan tipe JSON
4. Data dalam header request

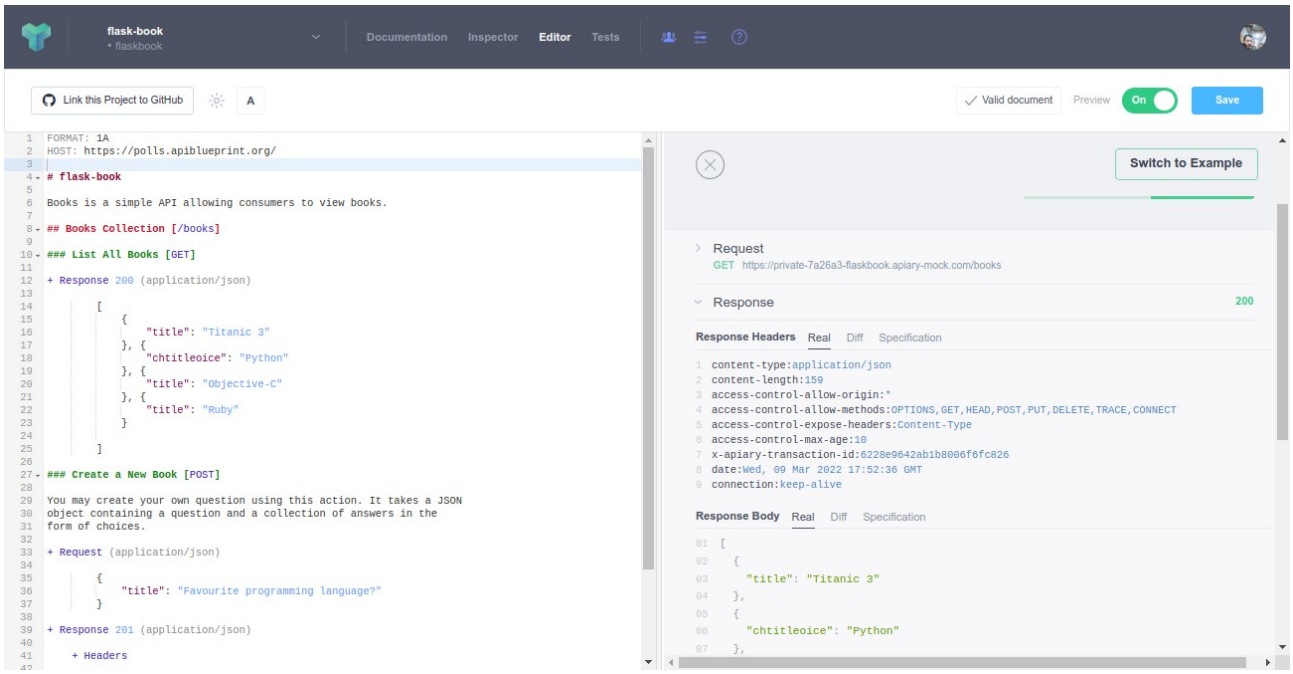
Jalankan dan pelajari file dalam repository referensi: play_ws.py

Tools:

curl, postman

adalah client untuk melakukan request untuk mencoba endpoint

apiary adalah salah satu tools untuk merancang w/s, sehingga request dan response W/S api dapat diketahui tanpa menunggu hingga development selesai.



Referensi

[1] Benslimane, D.; Dustdar, S.; Sheth, A. (2008). "Services Mashups: The New Generation of Web Applications". IEEE Internet Computing. 10 (5): 13–15. doi:10.1109/MIC.2008.110. S2CID 8124905

[2] Erl, Thomas; Carlyle, Benjamin; Pautasso, Cesare; Balasubramanian, Raj (2012). "5.1". SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST. Upper Saddle River, New Jersey: Prentice Hall. ISBN 978-0-13-701251-0

[3] Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.

[4] Source code api dengan flask, https://github.com/arissetyawan-teaching/flask.git

[5] Dan lain-lain