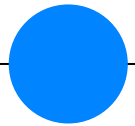
API REST



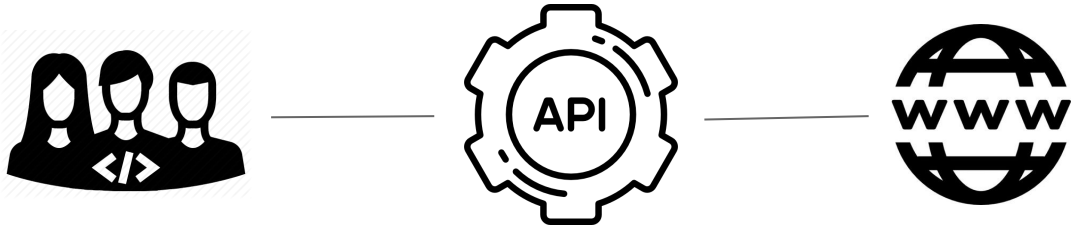
An API is a user interface for a developer.
So put some effort into making it pleasant

Sommaire

- Qu'est-ce qu'une API ?
- API REST
- Standard et principes d'utilisation
- Utilisation

Qu’est-ce qu’une API ?

Application Programming Interface



YouTube

Google

PayPal

ebay

API REST

- Architecture d'URL
- Verbes HTTP
- Choix du format de résultats
- Relation sur les liens
- Jeton d'authentification

URI comme identifiant

URL logique et lisible

- ✅ http://site.com/persons/83
- ❌ http://site.com/83/persons
- ✅ http://site.com/persons
- ❌ http://site.com/{1424-sjru-jd23}

Verbes HTTP comme identifiant

POST / GET / PUT / DELETE

- ✅ GET http://site.com/persons/83
- ❌ http://site.com/persons/83/get
- ✅ DELETE
- ❌ http://site.com/persons/83/comments/32

  http://site.com/persons/83/comments/32/delete

Réponses HTTP comme représentation des ressources

API, C’est une représentation de ressources

- XML
- JSON
- CSV
- HTML

Lien comme relation

Lien possède une relation

```
<link rel="self" title="self"
href="http://mywebsite.com/books/5?&page=1"/>
<link rel="next" title="next"
href="http://mywebsite.com/books/5?&page=2"/>
<link rel="last" title="last"
href="http://mywebsite.com/books/5?&page=42"/>
```

Paramètre comme jeton d'authentification

Jeton d'authentification


http://site.com/persons/key=hc732uigf8dsb


http://site.com/persons/user=Doe&pass=$ecr€t

RESTful Request

Verbes HTTP

- GET
- POST
- PUT
- DELETE

URI

Représentatif de la ressource

`https://api.github.com/users/:username`

`https://api.github.com/users/:username/repos`

`https://api.github.com/repos/:owner/:repo/pulls`

More info : *https://bastiennicoud.gitbooks.io/api-xml-json/content/principles/introduction.html*

RESTful Content

```
[
  {
    "id": 1296269,
    "owner": {
      "login": "octocat",
      "id": 1,
      "type": "User"
    },
    "name": "Hello-World",
    "full_name": "octocat/Hello-World",
    "description": "This your first repo!",
    "private": false,
    "fork": false
  }
]
```

Nommage des clefs

- “clef”
- id / slug
- snake _case
- [] ou {}

Contenu

- Bool
- Int
- Strings
- Object

More info : *https://bastiennicoud.gitbooks.io/api-xml-json/content/principles/introduction.html*

Utilisation - XML

Parse

```
var xml = "xml goes here",
  xmlDoc = $.parseXML( xml ),
  $xml = $( xmlDoc );
```

```
$xml.find( "book[category='COOKING']>author" ).text();
```

Génération

```
var xmlDocument = $.parseXML("<note/>");
var elTo = xmlDocument.createElement('to');
```

Utilisation - JSON

Parse

```
var json = JSON.parse(text);
```

```
var name = json.name;
```

Génération

```
var names = { "name":"John", "age":30, "city":"New York"};
var json = JSON.stringify(names);
```

A retenir

URL *représentatif de la ressource*

XML/JSON *comme corps de la requête*

FORMAT *standardisé*

Exercices !

Exercices - API GET

- bit.ly/APIXML
- https://jsonplaceholder.typicode.com
- Reprendre des données depuis API
- Afficher dans un tableau

Exercices - API POST

- Générer du JSON
- Envoyer le JSON dans l'API
- Retourner les valeurs du JSON

Thanks!

Any questions ?