



Bachelier en Informatique de Gestion

Projet de Développement Web

Enseignement supérieur économique de type court

Code FWB : 7534 30 U32 D1

Code ISFCE : 4IPDW



= BUROTIX ()

Table des matières

Généralités

- 01. Introduction au web
- 03. Outils
- 05. Frameworks

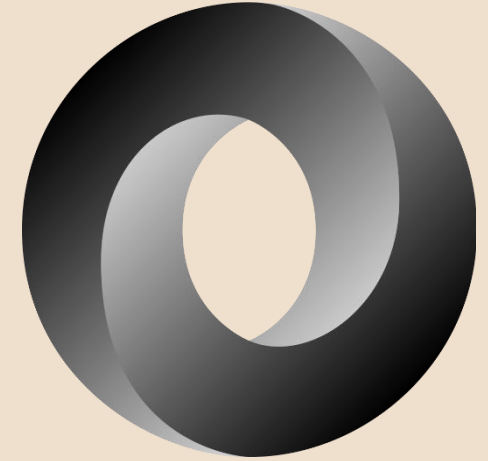
Côté Client

- 12. Structure HTML
- 13. Formulaire HTML
- 14. Mise en forme CSS
- 15. Adaptabilité
- 17. Javascript
- 18. Framework jQuery
- 19. AJAX

Côté Serveur

- 21. Middleware PHP
- 22. Traitement du formulaire
- 23. Architecture MVC
- 24. Base de données SQL
- 25. Données XML
- 26. Données JSON





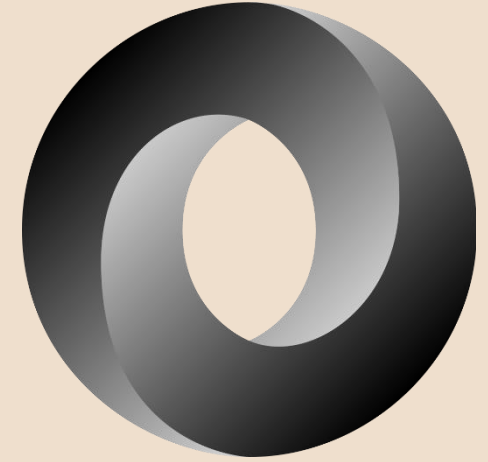
26. Données JSON

Structure

JSON vs XML



= BUROTIX ()



JSON : Structure



= BUROTIX ()

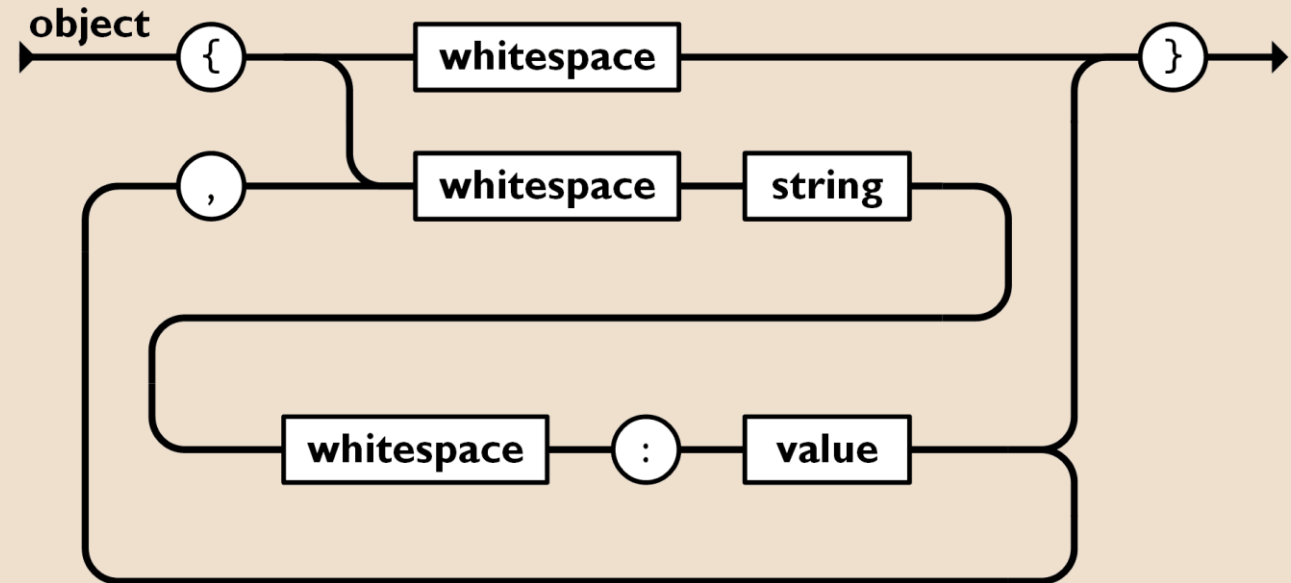
JSON en quelques mots

- JavaScript Object Notation = format d'échange de données
- Facile à lire et à écrire pour les humains
- Facile à analyser et à générer pour les machines
- Basé sur JavaScript.
- Format "texte"
- Indépendant du langage, universel
- Basé sur des conventions familières aux programmeurs de la famille "C" (C, C++, C#, Java, JavaScript, Perl, PHP, ...)
- www.json.org



JSON : deux structures

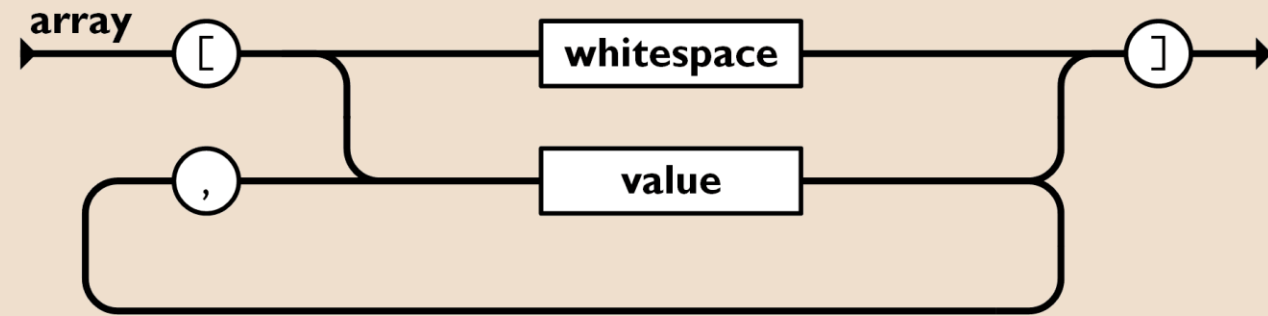
1. Une collection de paires $\langle \text{nom} \rangle / \langle \text{valeur} \rangle$
 - // structure, dictionnaire, table de hachage, liste à clés ou tableau associatif
 - Début : $\{$
 - Fin : $\}$
 - Entre $\langle \text{nom} \rangle$ et $\langle \text{valeur} \rangle$: $:$
 - Séparateur : $,$



JSON : deux structures

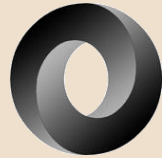
2. Une liste ordonnée de valeurs

- // tableau, vecteur, liste, séquence
- Début : **[**
- Fin : **]**
- Séparateur : **,**



JSON vs. XML

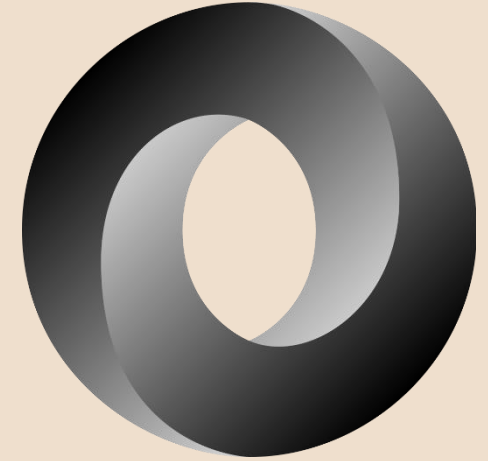
```
{
  Menu : {
    id : "file",
    Value : "File",
    Popup : {
      menuitem: [
        {
          value: "New",
          onclick:"CreateNewDoc()"
        },
        {
          value: "Open",
          onclick: "OpenDoc()"
        },
        {
          value: "Close",
          onclick: "CloseDoc()"
        }
      ]
    }
  }
}
```



```
<menu
  id="file"
  value="File">
  <popup>
    <menuitem
      value="New"
      onclick="CreateNewDoc()"
    />
    <menuitem
      value="Open"
      onclick="OpenDoc()"
    />
    <menuitem
      value="Close"
      onclick="CloseDoc()"
    />
  </popup>
</menu>
```



= BUROTIX ()



JSON : intégration en PHP

Exo 11 : PHP et JSON

- Données JSON fournies sous forme de `string php`.
- Fonction PHP `json_decode()`
- Exemple
 - `exo11.php`

```
$string = <<< JSON
{
    "id": "Margaret",
    "image": ".\\model\\media\\robes.jpg",
    "type": "Robe",
    "ref": "41V2138",
    "couleur": "Virtual Pink",
    "lien": "http:\\\\www.desigual.com\\fr_BE",
    "prix": 79,
    "taille": [
        "XL",
        "L"
    ]
}
JSON;
```



json_decode()

- Pour convertir un string JSON en un objet ou un assoc array PHP

```
$array = json_decode($string, $bool);
```

- Params
 - **\$string** : la chaîne sous format JSON
 - **\$bool** : **false** => return object (default)
true => return assoc array

- Référence

- <https://www.php.net/manual/fr/function.json-decode.php>



Exo 13 : PHP et JSON

- Données JSON : localisées sur un serveur
- Fonctions PHP

`file_get_contents()`
`json_decode()`

- Exemple
`exo13.php`



The screenshot shows a web browser window with the address bar displaying `playground.burotix.be/adv/banner_for_isfce.json`. The browser's security indicator shows a warning icon and the text "Not secure". The main content area displays a JSON object with the following structure:

```
{
  "banner_4IPDW": {
    "link": "https://www.burotix.be/",
    "image": "https://www.burotix.be/images/logo103x90",
    "text": "Entrepreneur, indépendant, artisan, ...",
    "background_image": "https://www.burotix.be/images/516277_480.jpg",
    "color": "#0dd3d1"
  }
}
```

Exo 14 : PHP et JSON

- Données JSON : produites par votre serveur (export)

- Fonctions PHP :

json_encode()

- Example :

exo14.php

[illegible]

json_encode()

- Pour convertir un **objet** ou un **assoc array** PHP en un string JSON

```
$json_str = json_encode( $array, $flags );
```

- Params

- **\$array** : les données sous format object ou assoc array ou ...

- **\$flags** :

JSON_HEX_TAG

JSON_PRETTY_PRINT

JSON_FORCE_OBJECT etc.

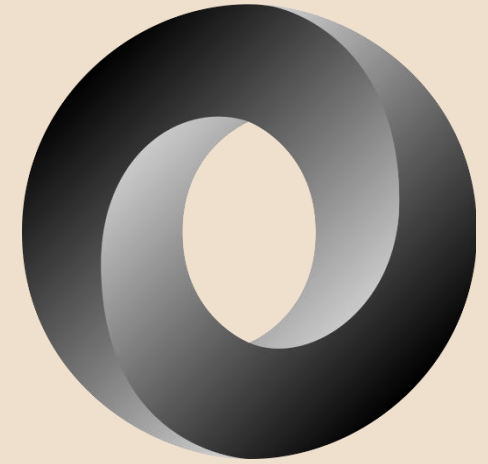
- Référence

- <https://www.php.net/manual/fr/function.json-encode.php>
- <https://www.php.net/manual/fr/json.constants.php>



The screenshot shows a web browser window with the address bar displaying 'localhost/13_json/exo13-14_make_json.php'. The main content area shows a JSON object representing a robe item. The JSON is formatted with indentation and line breaks. The object has the following properties: 'id' (Margaret), 'image' (a relative path to a media file), 'type' (Robe), 'ref' (41V2138), 'couleur' (Virtual Pink), 'lien' (a URL to a website), 'prix' (79), and 'taille' (an array containing 'XL' and 'L').

```
{
  "id": "Margaret",
  "image": ".\\model\\media\\robes.jpg",
  "type": "Robe",
  "ref": "41V2138",
  "couleur": "Virtual Pink",
  "lien": "http:\\\\www.desigual.com\\fr_BE",
  "prix": 79,
  "taille": [
    "XL",
    "L"
  ]
}
```



JSON : API



= BUROTIX ()

API : Principe

- Application Programming Interface
- Ensemble de définitions et de protocoles qui permettent à un logiciel de communiquer avec un autre logiciel
- Architecture client-serveur
- Scénario
 - L'application cliente envoie une demande à l'application serveur via une API.
 - L'application serveur répond ensuite à la demande de l'application cliente en renvoyant les données demandées.
 - format des données : JSON, XML, ou CSV.



API : Applications

- récupération de données
- envoi de données
- automatisation de tâches
- ajouter des fonctionnalités à une application
- améliorer l'expérience utilisateur
 - Par exemple, une application de planification de voyage peut utiliser l'API Google Maps pour afficher une carte et les directions vers une destination.



API : Types

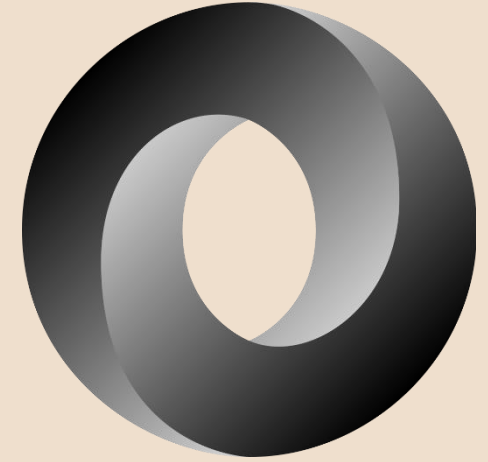
- API Web
 - communication via HTTP ou HTTPS
- API REST
 - type particulier d'API Web
 - méthodes HTTP GET, POST, PUT et DELETE
- API SOAP
 - communication via XML
- API de bibliothèque
 - API internes permettant à un logiciel de communiquer avec les bibliothèques système.



Exo 24 : log-in, log-out avec API et JSON

- Partez du chapitre 22, exo 24.
 - mécanisme de log-in
 - nom et mot de passe
 - identification est permanente : **\$_SESSION**
- Validation sur un serveur extérieur
- URI :
`http://playground.burotix.be/login/ ? login=<login> & passwd=<passwd>`
- Retour : format JSON

```
{  
    "identified": true,  
    "name": "Luke Skywalker",  
    "role": "user"  
}
```
- Développez cet API Web !



JSON : Références



= BUROTIX ()

Références

- Général
 - <https://www.json.org/json-fr.html>
 - https://www.w3schools.com/js/js_json_intro.asp
- API
 - <https://itds.fr/creer-une-api-rest-en-php/>
 - <https://blog.nicolashachet.com/developpement-php/larchitecture-rest-expliquee-en-5-regles/>
 - <https://blog.nicolashachet.com/developpement-php/exemples-api-rest-en-php/>

