



M107 Développer des sites web dynamiques

D. Réaliser un site web en MVC

1

Filière: Développement Digital

Formatrice: Asmae YOUNALA

PLAN DU MODULE

- A. Introduction
- B. Programmer en PHP
- C. Manipuler les données
- D. Réaliser un site web avec l'architecture MVC**

PLAN :

- D.1 Développer des sites dynamiques avec MVC en mode natif
- D.2 Découvrir les services web et les API REST

Présentation de l'MVC

M



Model : Les modèles seront les éléments qui se chargeront des échanges avec la base de données (CRUD). On ne mettra pas de traitement dans ces fichiers, uniquement des requêtes.

V



View : Les vues contiendront uniquement le code HTML destiné à structurer les pages.

C



Controller : Les contrôleurs contiendront toute l'intelligence de l'application, le traitement des données en vue de leur affichage, par exemple.

5

Présentation de l'MVC

MVC (Modèle-Vue-Contrôleur) est un modèle de conception, il est donc indépendant du langage de programmation. Il met l'accent sur la séparation entre la logique métier et l'affichage du logiciel.

- Les modèles et les contrôleurs sont généralement des classes.
- Les vues sont généralement des Templates HTML ou PDF

6

Présentation de l'MVC

► Quelques avantages

- Meilleure organisation du code;
- Diminution de la complexité lors de la conception;
- Conception claire et efficace grâce à la séparation des données de la vue et du contrôleur;
- Possibilité de réutilisation de code dans d'autres applications;
- Un gain de temps de maintenance et d'évolution du site;
- Une plus grande souplesse pour organiser le développement du site entre différents développeurs;
- Plus de facilité pour les tests unitaires.

► Quelques inconvénients

- Augmentation de la complexité lors de l'implantation;
- Éventuel cloisonnement des développeurs;
- Architecture complexe pour des petits projets;
- Le nombre important de fichiers représente une charge non négligeable dans un projet.

Présentation de l'MVC

Frameworks MVC pour PHP

L'utilisation de Framework peut s'avérer d'un grand avantage au développeur. Il permet de faciliter le travail et réduire le temps de réalisation tout en profitant des avantages qu'offre l'architecture.

Exemples de frameworks utilisant l'architecture MVC

- ▶ Lavarel : <https://laravel.com/>
- ▶ Lumen : <https://lumen.laravel.com/>
- ▶ Symfony : <https://symfony.com/>
- ▶ Cake PHP : <https://cakephp.org/>
- ▶ Zend Framework : <https://framework.zend.com/>
- ▶ Yii Framework : <https://www.yiiframework.com>
- ▶

8

Présentation de l'MVC

Frameworks MVC pour PHP

L'utilisation de Framework peut s'avérer d'un grand avantage au développeur. Il permet de faciliter le travail et réduire le temps de réalisation tout en profitant des avantages qu'offre l'architecture.

Exemples de frameworks utilisant l'architecture MVC

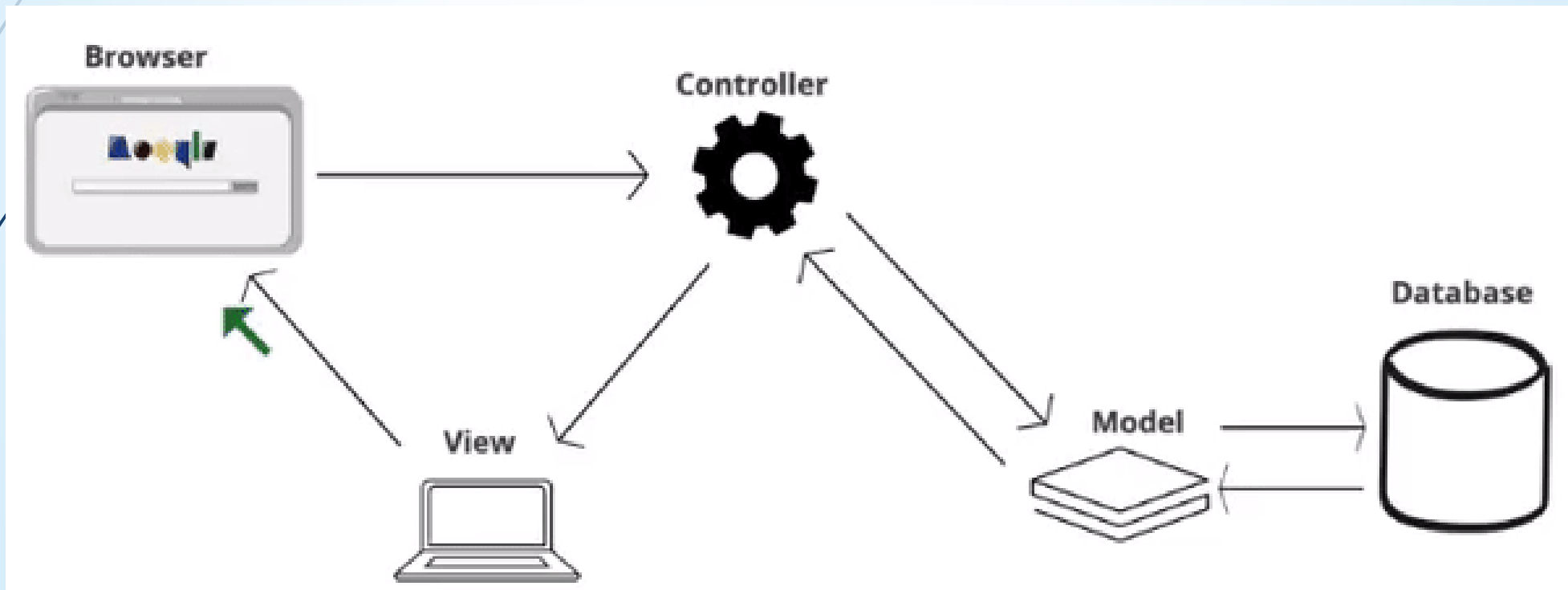
- ▶ Lavarel : <https://laravel.com/>
- ▶ Lumen : <https://lumen.laravel.com/>
- ▶ Symfony : <https://symfony.com/>
- ▶ Cake PHP : <https://cakephp.org/>
- ▶ Zend Framework : <https://framework.zend.com/>
- ▶ Yii Framework : <https://www.yiiframework.com>
- ▶

9

Présentation de l'MVC

Fonctionnement:

Le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue.



Présentation de l'MVC : Rôle de **Model**

- Le modèle contient les **données** manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.
- Le modèle offre des méthodes pour mettre à jour ces données (**insertion suppression, changement de valeur**).
- Il offre aussi des méthodes pour **recupérer** ses données.
- Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données.
- Si par exemple le programme manipule une base de données pour les emplois du temps, le modèle peut avoir des méthodes pour avoir, tous les cours d'une salle, tous les cours d'une personne ou tous les cours d'un groupe précis.

Présentation de l'MVC : Rôle de **View**

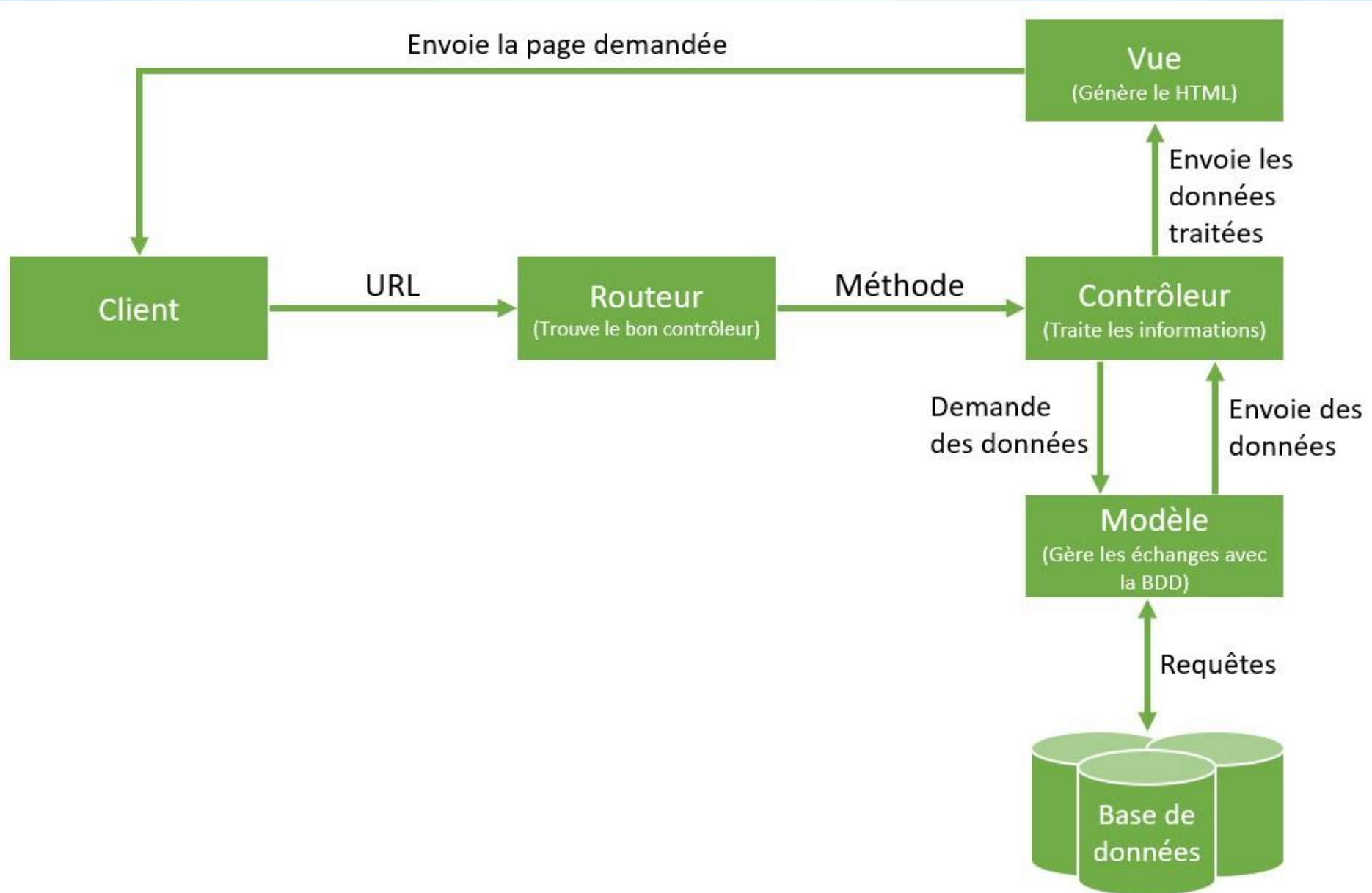
- La vue fait **l'interface** avec l'utilisateur.
- Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle.
- Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, ...).
- Ses différents événements sont envoyés au contrôleur.
- La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

Présentation de l'MVC : Rôle de **Controller**

- Le contrôleur est chargé de la **synchronisation** du modèle et de la vue.
- Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.
- Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour.
- Certains événements de l'utilisateur ne concerne pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.
- Le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants.
- En effet si un même objet reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement.
- Ce tri des événements peut s'avérer fastidieuse et peut conduire à un code pas très élégant (un énorme switch). C'est pour éviter ce problème que le contrôleur est réparti en **plusieurs objets**.

Présentation de l'MVC

13



Présentation de l'MVC : Rôle de **router**

Le routeur

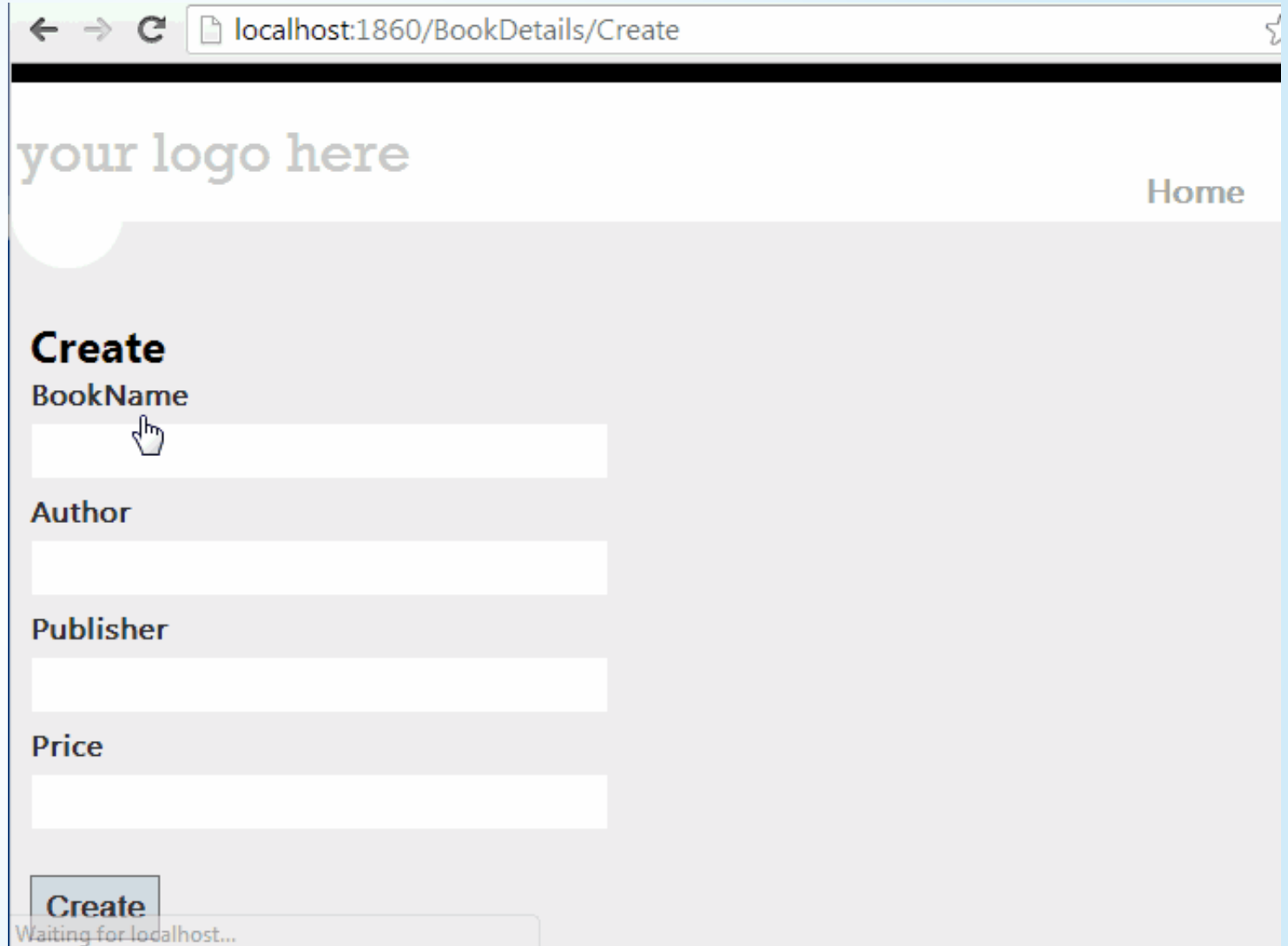
- Dans la structure MVC, un seul et unique fichier est le point d'entrée de l'application, quelle que soit la page affichée.
- Il est systématiquement appelé, et envoie la demande au bon contrôleur.
- Il est chargé de trouver le bon chemin pour que l'utilisateur récupère la bonne page, d'où le nom de routeur.

MVC: Amélioration d'écriture d'URL

L'URL rewriting est la possibilité de réécrire les adresses afin de les rendre plus significatives.

Un des intérêts est de cacher l'appel des variables à l'intérieur de l'URL, ce qui la rend plus lisible pour l'utilisateur ainsi que pour les moteurs de recherche, améliorant ainsi votre référencement.

Cette partie ne sera pas traitée dans ce cours



The screenshot shows a web browser window with the address bar displaying `localhost:1860/BookDetails/Create`. The page has a header with the text "your logo here" and a "Home" link. The main content area is titled "Create" and contains a form with the following fields:

- BookName**: A text input field with a mouse cursor hovering over it.
- Author**: A text input field.
- Publisher**: A text input field.
- Price**: A text input field.

At the bottom of the form is a "Create" button. A status bar at the very bottom of the browser window indicates "Waiting for localhost..."

D.2 Découvrir les services web et les API REST

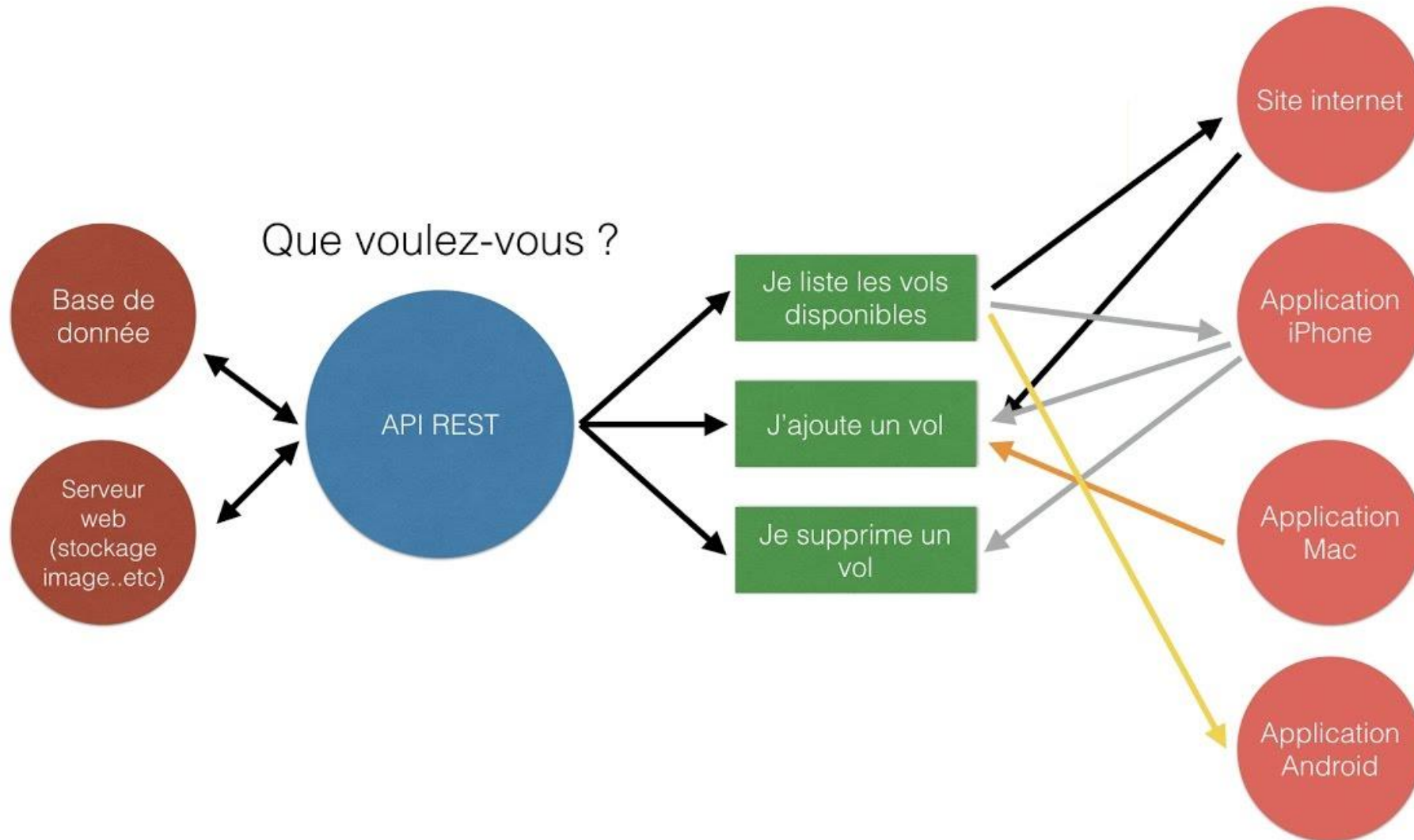
Service web : Définition

- Les services web permettent à différentes applications écrites dans des langages de programmation différents de communiquer entre elles.
- Un service web est un médium standardisé permettant la **communication entre les applications clients et serveur sur le World Wide Web**. Il s'agit d'un module logiciel conçu pour effectuer certaines tâches.

API : Définition

- Le terme **API** est l'acronyme de “**Application Programming Interface**” qui signifie “**Interface de programmation applicative**”.
- Les APIs sont des “passerelles” qui permettent à **deux ou plusieurs applications** (ex. application mobile ou site Internet) de communiquer entre eux et de faciliter **les échanges de données** entre un “client” et un “Serveur” par exemple.
- Les API permettent à un produit ou service de communiquer avec d'autres produits et services sans connaître les détails de leur mise en œuvre.
- Les API sont parfois considérées comme des **contrats**, avec une documentation qui constitue un accord entre les parties : si la partie 1 envoie une requête à distance selon une structure particulière, le logiciel de la partie 2 devra répondre selon les conditions définies.
- Exemple de site fournissant une liste des API: <https://apist.fun/>

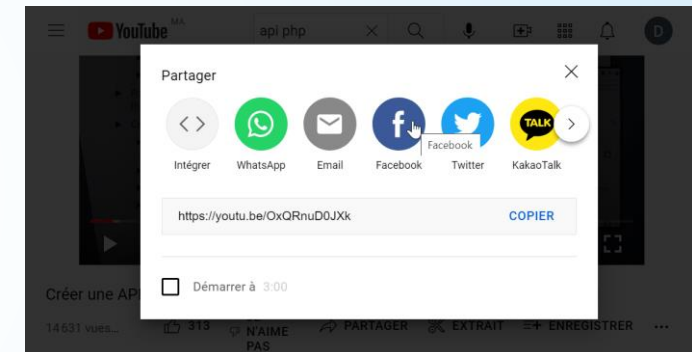
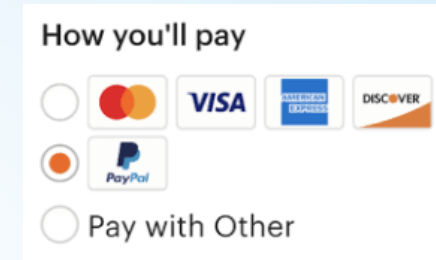
API : Exemple



20

API : Exemples

- Paiement d'une commande dans un site e-commerce :
- Publications dans un réseau social depuis un site web
- Service de météo, d'heure de prière et de bourse dans le site des informations lematin.ma



SERVICES

Météo
Casablanca 04 juillet 2022
65.3°F
18.5°C 

< 17.5°C 15.4°C 16.6°C 17.7°C 17.6°C     

Météo d'autres villes >

Horaires de Prière
Casablanca 04 juillet 2022

| | |
|---------------|-------|
| As-sobh : | 04:36 |
| Al-chourouq : | 06:22 |
| Ad-dohr : | 13:40 |
| Al-asr : | 17:19 |
| Al-maghrib : | 20:49 |
| Al-ichae : | 22:20 |

Horaires d'autres villes >

Cinéma

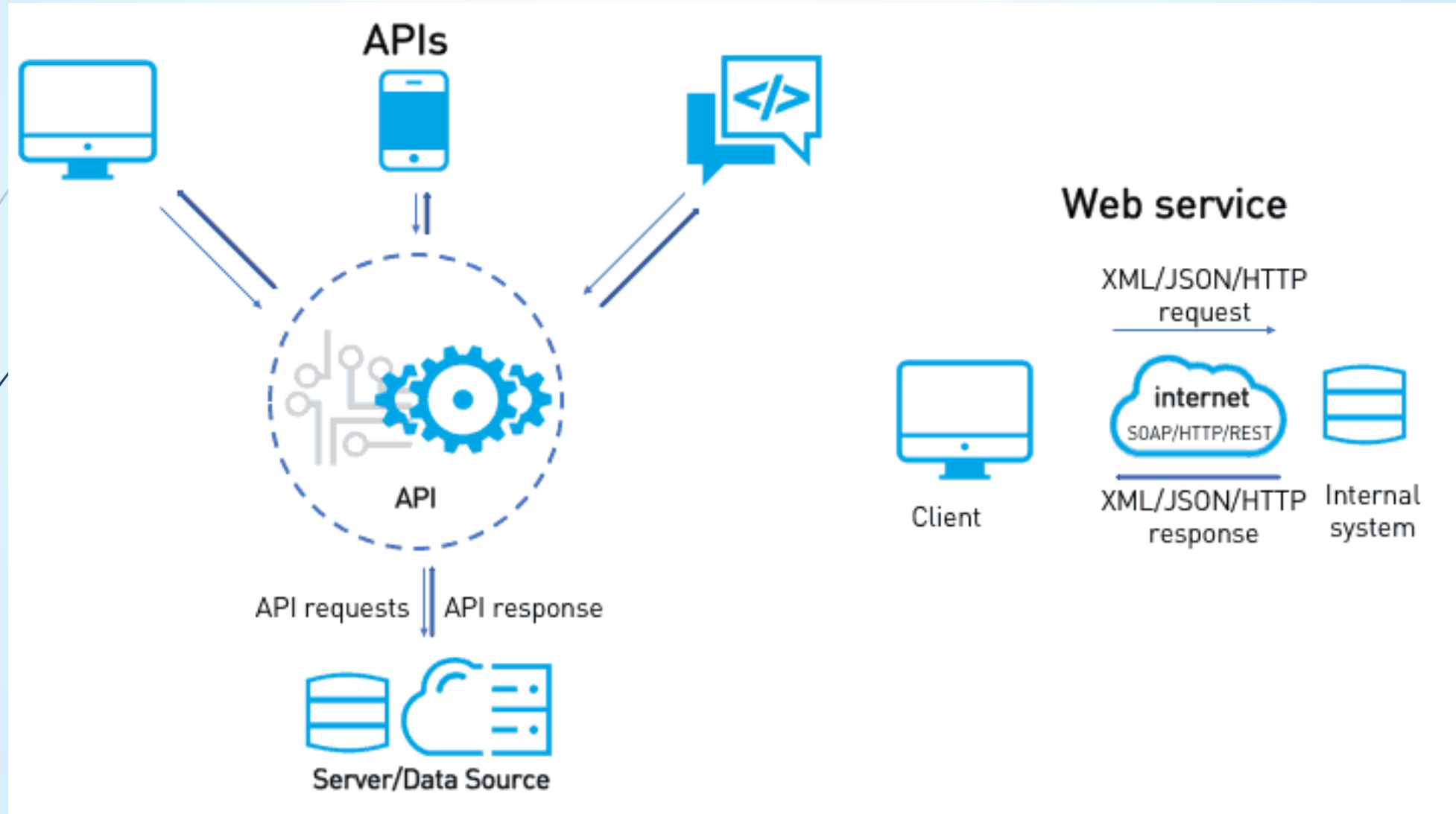


Films dans les salles >

Différence entre service web et API

- APIs et Services Web servent de “moyen de communication” entre plusieurs sites ou applications. La seule différence est qu’un **service Web** facilite l’interaction entre **deux machines sur un réseau** alors qu’une **API** sert d’interface entre **deux applications différentes** afin qu’elles puissent communiquer entre elles. Le protocole HTTP est le protocole le plus couramment utilisé pour la communication.
- **Un service Web est simplement une API enveloppée dans le protocole HTTP.** Une API n’a pas toujours besoin d’être basée sur le Web.
- Une API consiste en un ensemble complet de règles et de spécifications qu’un programme logiciel doit suivre afin de faciliter l’interaction.
- **Un service Web peut ne pas** contenir un **ensemble complet de spécifications** et parfois ne pas être en mesure d’exécuter toutes les tâches qui peuvent être possibles à partir d’une API complète.

Différence entre service web et API



Types de données transférés:

- Le principal composant d'un service web ou d'une api sont les **données** transférées entre le client et le serveur.
- Ces données transférées sont en **XML (Extensible Markup Language)**, en **JSON (JavaScript Object Notation)** ou autre.



Exemples : XML et JSON

XML

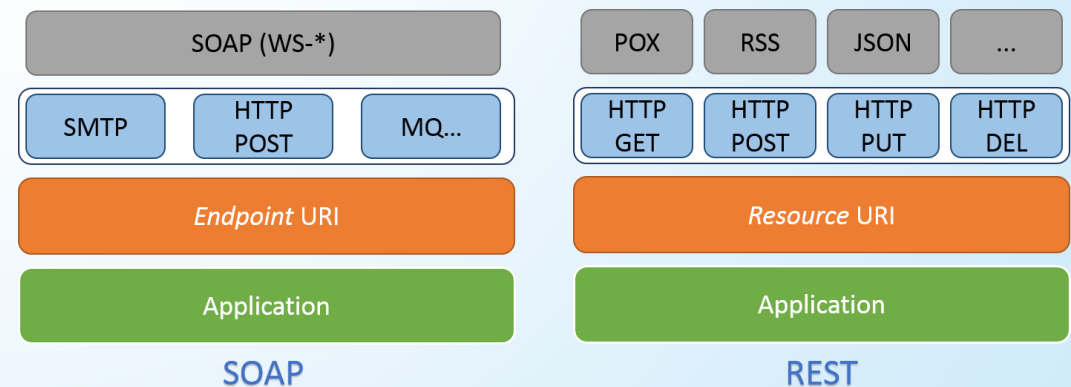
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<repertoire>
  <contact id="1">
    <nom>codeur</nom>
    <prenom>java</prenom>
    <mobile>098745126</mobile>
    <email>codeurjava8@gmail.com</email>
  </contact>
</repertoire>
```

JSON

```
{
  "name"      : "Amit Pathak",
  "profile"   : "Software Engineer",
  "age"       : 24,
  "location"  : "London, UK"
}
```


API : Types

- Il existe 2 types d'API :
 - Les API SOAP (Simple Object Access Protocol)
 - Les API REST (Representational State Transfer)
- **SOAP** est un protocole de communication basé sur XML qui permet aux applications de s'échanger des informations via HTTP.
- **REST est un style d'architecture de services web** qui opère comme une chaîne de communication entre différents systèmes et Internet. On utilise principalement le REST pour mettre en œuvre des API modernes.



API REST

- **Les API REST** utilisent généralement des méthodes HTTP (**H**yper **T**ext **T**ransfer **P**rotocol) pour récupérer ou envoyer des informations entre les applications.
- Ce sont les mêmes **méthodes HTTP**, que nous utilisons avec un **navigateur Web** pour **visiter un site Web**, sauf qu'ici, c'est plutôt utilisé pour interagir avec une application.
- **Voici les 4 méthodes les plus populaires de HTTP :**
 - « **GET** »: est une méthode, en **lecture seule**, qui permet de récupérer une ressource spécifiée.
 - « **POST** »: permet de **soumettre** les données à la **ressource** pour les **traiter**. Il est aussi **capable** de **créer** de nouvelles ressources.
 - « **PUT** »: est la méthode capable de **mettre à jour** la ressource, en remplaçant les données existantes.
 - « **DELETE** »: permet de **supprimer** une ressource.

HTTP Methods

GET

Receive information
about an API resource

POST

Create an
API resource

PUT

Update an
API resource

DELETE

Delete an
API resource

API REST

Fonctionnement :

- Le client référence une ressource Web à l'aide d'une URL
- Surcouche du protocole HTTP
- Une représentation de la ressource est retournée
- Le client se trouve dans un nouvel état grâce à la ressource
- Le nouvel état peut conduire à la demande d'une nouvelle ressource



Consommation d'une API REST en php : CURL

- **cURL (client URL request library)** est une bibliothèque qui permet d'effectuer des requêtes HTTP en PHP.
- **cURL** supporte les méthodes de requêtes en HEAD, GET et POST ainsi qu'en PUT.
- cURL est activée par défaut en PHP, vous pouvez vérifier si elle est activée en appelant la fonction **phpinfo()**. Il vous est demandé de la vérifier avant d'écrire votre premier programme simple en PHP.

```
<?php  
phpinfo();  
?>
```

| curl | |
|------------------|---------|
| cURL support | enabled |
| cURL Information | 7.55.0 |
| Age | 3 |
| Features | |
| AsynchDNS | Yes |
| CharConv | No |
| Debug | No |

Consommation d'une API REST en php : CURL

► Pour utiliser la bibliothèque cURL, il suffit de:

1. Initialiser la session avec : [curl_init\(\)](#)
2. Définir les options de transfert avec la fonction [curl_setopt\(\)](#)
3. Exécuter la requête avec [curl_exec\(\)](#)
4. Mettre fin à la session avec la fonction [curl_close\(\)](#)

Exemple de consommation d'une API REST

- On va prendre, comme exemple, une API listant la plupart des universités du monde :

<https://github.com/Hipo/university-domains-list>

- Cet API permet de retourner un fichier JSON qui contient les domaines, les noms et les pays de la plupart des universités du monde.

- Exemples de cas d'utilisation :

- Vous pouvez créer un script de validation qui vérifie le domaine de messagerie.
- Vous pouvez générer automatiquement le pays et l'université d'un utilisateur en consultant ses e-mails.

- Dans notre exemple, on va créer un formulaire de recherche des universités par pays (country). Pour ce, on utilisera ce lien:

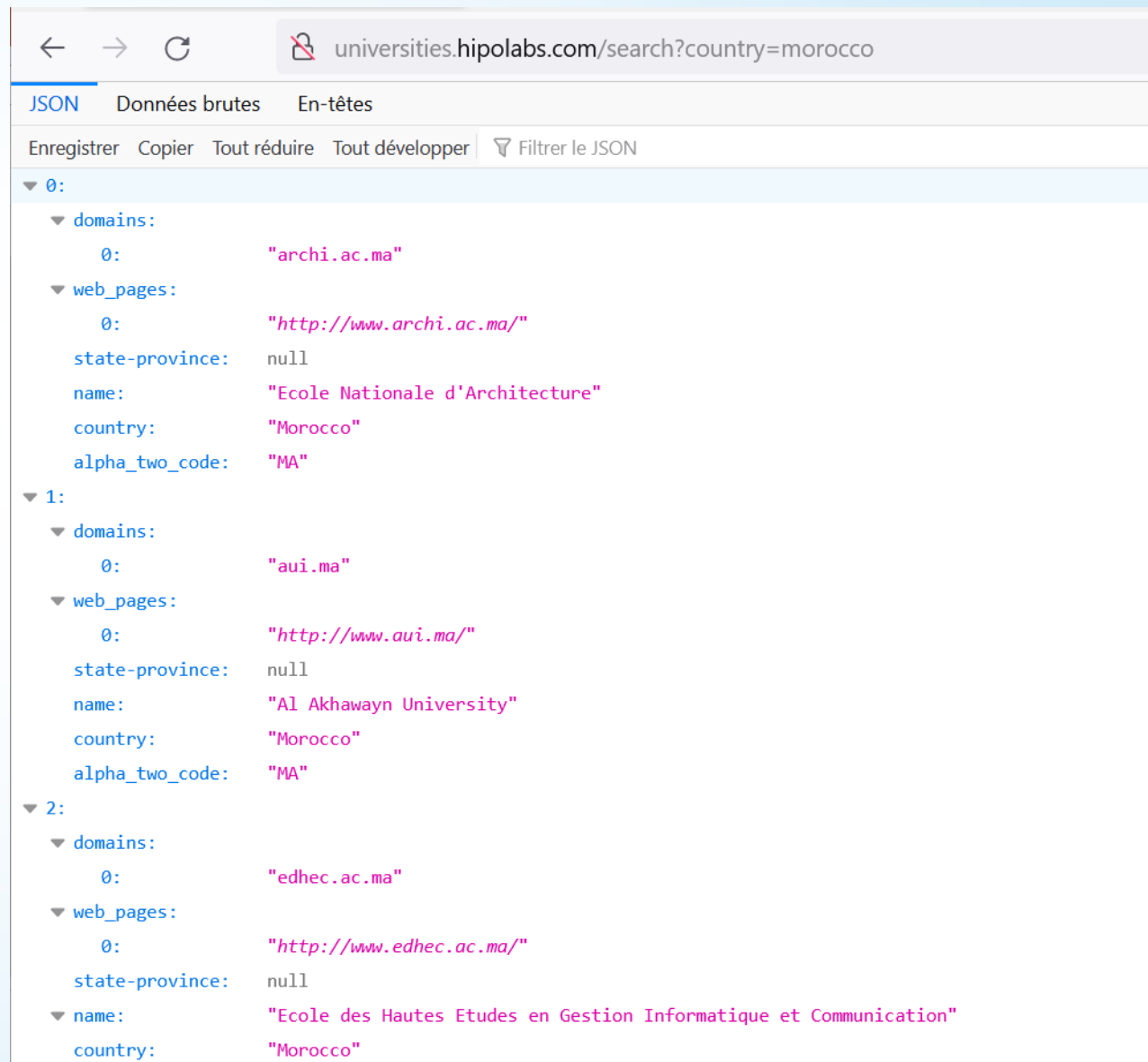
<http://universities.hipolabs.com/search?country=.....>

Exemple de consommation d'une API REST

Voici un extrait du résultat retourné par cet url:
<http://universities.hipolabs.com/search?country=morocco>

Il s'agit d'un fichier JSON, où on stocke sous forme d'un tableau des objets « université ».
Chaque objet a les attributs suivants:

- **domains** : array
- **web_pages**: array
- **state-province**: string
- **name**: string
- **country** : string
- **alpha_two_code** : string



The screenshot shows a web browser with the URL `universities.hipolabs.com/search?country=morocco`. The browser's developer tools are open, displaying the JSON response. The JSON is a list of three university objects, each containing details like domain, web pages, state-province, name, country, and alpha_two_code.

```
{
  "0": {
    "domains": [
      "archi.ac.ma"
    ],
    "web_pages": [
      "http://www.archi.ac.ma/"
    ],
    "state-province": null,
    "name": "Ecole Nationale d'Architecture",
    "country": "Morocco",
    "alpha_two_code": "MA"
  },
  "1": {
    "domains": [
      "aui.ma"
    ],
    "web_pages": [
      "http://www.aui.ma/"
    ],
    "state-province": null,
    "name": "Al Akhawayn University",
    "country": "Morocco",
    "alpha_two_code": "MA"
  },
  "2": {
    "domains": [
      "edhec.ac.ma"
    ],
    "web_pages": [
      "http://www.edhec.ac.ma/"
    ],
    "state-province": null,
    "name": "Ecole des Hautes Etudes en Gestion Informatique et Communication",
    "country": "Morocco"
  }
}
```


Exemple de consommation d'une API REST

Le résultat qu'on souhaite avoir aura la forme suivante:

Entrez un nom d'un pays (*en anglais*)

Liste des universités du pays: Morocco

| Université | Site web |
|--|---|
| Ecole Nationale d'Architecture | http://www.archi.ac.ma/ |
| Al Akhawayn University | http://www.aui.ma/ |
| Ecole des Hautes Etudes en Gestion Informatique et Communication | http://www.edhec.ac.ma/ |
| Ecole Mohammadia d'Ingénieurs | http://www.emi.ac.ma/ |
| Ecole National d'Agriculture de Meknes | http://www.enameknes.ac.ma/ |
| Ecole Supérieure de Commerce et des Affaires | http://www.esca.ac.ma/ |
| Université Ibn Zohr Agadir | http://www.esta.ac.ma/ |
| Institut Supérieur de Commerce et d'Administration des Entreprises | http://www.groupeiscae.ma/ |
| Ecole des Hautes Etudes Commerciales MAROC | http://www.hec.ac.ma/ |
| Institut des Hautes Etudes de Management | http://www.hem.ac.ma/ |
| Ecole Supérieure d'Informatique et de Management | http://www.hightech.edu/ |

Exemple de consommation d'une API REST

Code HTML de création du formulaire :

```
<form action="" method="post">  
  Entrez un nom d'un pays <i>(en anglais)</i> <input type="text" name="country" />  
  <input type="submit" value="Rechercher" name="search" />  
</form>
```

Entrez un nom d'un pays *(en anglais)*

Rechercher

Exemple de consommation d'une API REST : Appel de la ressource

```
<?php
if (isset($_POST["search"])) {
    //Encoder les caractères spéciaux et espaces récupéré dans le champ
    country
    $pays = filter_var($_POST["country"], FILTER_SANITIZE_ENCODED);
    if (!empty($pays)) {
        //L'url de l'api à utiliser
        $url = "http://universities.hipolabs.com/search?country=$pays";
        //Initialisez une session CURL en passant l'url à utiliser
        $client = curl_init($url);
        // Activer l'option RETURNTRANSFER pour retourner le
        transfert en tant que chaîne de caractères
        curl_setopt($client, CURLOPT_RETURNTRANSFER, true);
        //Executer la requête
        $response = curl_exec($client);
        //Convertir le résultat json en un objet PHP
        $response = json_decode($response);
        //Fermer la session CURL
        curl_close($client);
    }
}
?>
```

Exemple de consommation d'une API REST : Exploiter les résultats

```

<?php
    if (isset($response)) :
        if (count($response) != 0) :
            ?>
                <h2>Liste des universités du pays: <?= $response[0]->country
            ?></h2>
                <table class="tab">
                    <tr>
                        <th>Université</th>
                        <th>Site web</th>
                    </tr>
                    <?php foreach ($response as $universite) : ?>
                        <tr>
                            <td><?= $universite->name ?></td>
                            <td><?= $universite->web_pages[0] ?></td>
                        </tr>
                    <?php endforeach; ?>
                </table>
            <?php
                endif;
            endif;
        ?>

```

Exemple de consommation d'une API REST :

Vous trouverez des exemples bien expliqués des autres méthodes HTTP comme POST, PUT et DELETE sur la page web suivante :

➡ <https://waytolearnx.com/2020/01/tutoriel-curl-en-php.html>

Exemple de création d'une API REST

Dans l'exemple qui suit, on souhaite créer une API REST permettant de retourner une liste des produits par catégorie sous format JSON. (On utilisera la base de données gestionStock)

On définira la fonction suivante:

```
function DelivreJSONResponse($status, $statusMessage, $data)
{
    header("charset=UTF-8");
    header("HTTP/1.1 $status $statusMessage");
    $response["status"] = $status;
    $response["status_message"] = $statusMessage;
    $response["data"] = $data;
    $jsonResponse = json_encode($response); //Transformation du tableau php en
    json
    echo $jsonResponse; //Affichage chez le client
}
```

Exemple de création d'une API REST

```
//Cet entête permettra à n'importe quelle client d'accéder à ces ressources
header('Access-Control-Allow-Origin: *');
//Indiquer que le résultat est sous forme de json et avec un codage utf-8
header("Content-type: application/json; charset=utf-8");
//Chercher si la requête indique la catégorie à chercher ou pas
if (isset($_REQUEST["categorie"])) {
    $categorie = $_REQUEST["categorie"];
    //Création de l'objet PDO
    $db = new PDO("mysql:host=localhost;dbname=gestionstock", "root");
    $stm = $db->prepare("select * from produit where catégorie=? ");
    $stm->execute([$categorie]);
    $tab = $stm->fetchAll(PDO::FETCH_OBJ);
}
```


Exemple de création d'une API REST

```
if ($tab) {  
    DelivreJSONResponse(200, "Liste des produits", $tab); //Cas de recherche réussie  
} else {  
    DelivreJSONResponse(200, "Aucun produit n'est trouvé!", null); //Cas de aucune  
    ressource n'est trouvé  
}  
} else {  
    DelivreJSONResponse(400, "Aucune catégorie n'est indiquée!", null); //Cas de non  
    indication de la catégorie  
}
```

Code des réponses HTTP

Les codes de statut de réponse HTTP indiquent si une requête [HTTP](#) a été exécutée avec succès ou non. Les réponses sont regroupées en cinq classes :

- Les réponses informatives (100 - 199),
- Les réponses de succès (200 - 299),
- Les messages de redirection (300 - 399),
- Les erreurs du client (400 - 499),
- Les erreurs du serveur (500 - 599).

41

Exemple d'appel de l'API REST créée:

