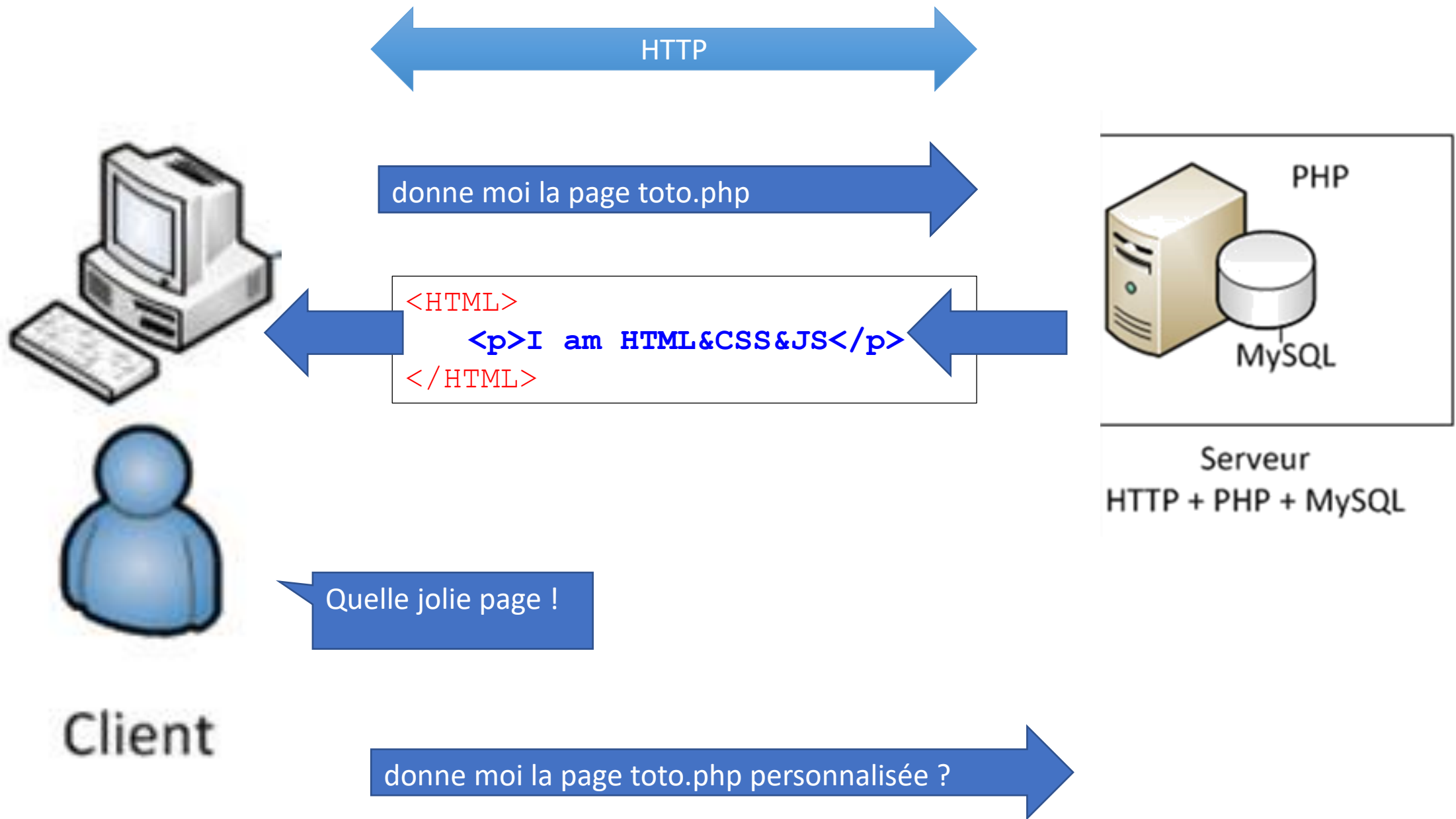




# PHP & cie

Alexandre Ahmad





## URL absolue [ modifier | modifier le code ]

Une URL absolue permet d'indiquer comment accéder à une ressource indépendamment de tout contexte où elle peut être précisée ou transmise. Elle commence par l'indication d'un schéma de représentation (spécifique au protocole de communication utilisé pour accéder à cette ressource), suivi de l'ensemble des paramètres permettant de localiser sur le réseau le service hébergeant la ressource, puis permet de préciser à ce service le nom d'une ressource à traiter, transmettre des données de traitement, acheminer et récupérer les résultats, puis de préciser éventuellement quelle partie de ce résultat sera utilisée.

Exemple : `http://Jojo:lApIn@www.exemple.com:8888/chemin/d/acc%C3%A8s.php?q=req&q2=req2#signet`

- Protocole, normalement obligatoire (mais certains clients web peuvent tenter de déterminer le protocole à partir de la forme du nom du service codé ci-dessous) :
  - `http` : **protocole de communication**, en l'occurrence ici **HTTP**, pour accéder à un **serveur web**,
  - `:` : caractère de séparation obligatoire si le protocole est précisé.
- Localisation complète de la ressource, représentée selon le protocole de représentation ci-dessus :
  - emplacement du service hébergeant la ressource sur l'espace du réseau global :
    - `//` : chaîne de caractères pour les **protocoles** dont la requête comprend un chemin d'accès, permettant de préciser et localiser le service avant ce chemin,
    - données d'authentification (optionnelles, le service peut les demander séparément de façon plus sécurisée que via l'URL) :
      - `Jojo` : nom d'utilisateur, notamment utile pour accéder à des parties non publiques d'un **site web**,
      - `:` : caractère de séparation si un mot de passe est indiqué,
      - `lApIn` : mot de passe de l'utilisateur, indiqué ici « **en clair** »,
      - `@` : caractère terminant les données d'identification présentes avant le nom du service.
    - `www.exemple.com` : **nom de domaine** du service ; on peut parfois utiliser plutôt son **adresse IP**. Si le nom de service ou l'adresse peuvent contenir des caractères réservés comme `:`, le nom de service ou l'adresse sera encadré de crochets doubles `[ ]`. Lui-même se décompose en :
      - `www` : sous-domaine (par défaut **www**),
      - `exemple` : nom de domaine de **deuxième niveau**,
      - `com` : nom de **domaine de premier niveau**,
    - indication optionnelle d'un numéro de port (au cas où le même serveur possède des services n'utilisant pas le port par défaut pour le protocole de communication) :
      - `:` : caractère indiquant qu'un numéro de port est précisé en suffixe,
      - `8888` : numéro de **port TCP/IP** du **serveur HTTP**, doit être précisé lorsqu'il ne s'agit pas du **port standard** pour le protocole utilisé (qui est 80 pour **HTTP**, 21 pour **FTP**...),
      - `[1234:abcd::1234]:8888` : Dans le cas d'une adresse IPv6, si on veut spécifier le port, il est obligatoire de mettre l'adresse entre crochets pour ne pas confondre le port et l'adresse.
  - Nom complet de la ressource à demander sur le service une fois connecté :
    - `/chemin/d/` : chemin absolu (commençant par un `/`) sur le service contenant la **page web**, obligatoire pour les services à chemin d'accès (par défaut ce chemin sera `/`),
    - `acc%C3%A8s.php` : nom de la page web, optionnel (de nombreux services web déterminent un nom de ressource par défaut pour chaque chemin indiqué). On remarque qu'un caractère non **ASCII** comme « è » est codé en « `%C3%A8` »<sup>7</sup>. L'extension n'a aucune signification directe pour le client, mais en revêt parfois pour le serveur qui l'utilise localement pour savoir comment traiter la ressource demandée et la présenter au client.
  - Données supplémentaires optionnelles, transmises au service lors de la demande à la ressource :
    - `?` : caractère de séparation obligatoire pour indiquer que des données complémentaires suivent.
    - `q=req&q2=req2` - chaîne de requête, traitée par la page web sur le **serveur**.
  - Données supplémentaires optionnelles, pour l'exploitation de la ressource après son obtention par le logiciel client (non transmises dans la requête au service) :
    - `#` : caractère de séparation obligatoire pour indiquer un **signet** ou une **balise**,
    - `signet` : identificateur du signet ou de la balise. Il s'agit d'un emplacement à l'intérieur de la page web retournée par le service, cette donnée sera traitée par le **navigateur web**.

Quelques exemples pratiques :

- URL de **Wikipédia** :

# Transmettre des données: GET

<http://localhost/index.php?page=xxx>

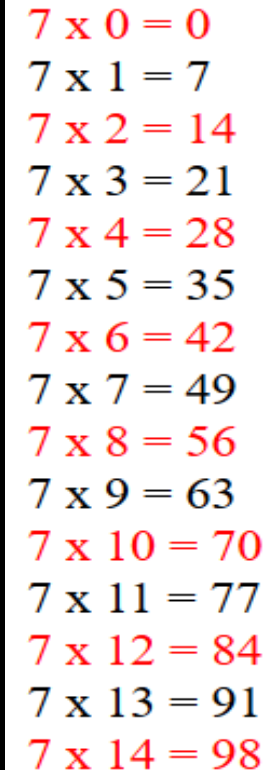
<?php

`$ma_var = $_GET['page']; // $ma_var = xxx`

?>

Ich bin ein **superglobal**

- Afficher la table de multiplication d'un nombre
- Si il est pair, **en rouge**
- <http://.../indexTP0.php?table=7&max=14>



7 x 0 = 0  
7 x 1 = 7  
7 x 2 = 14  
7 x 3 = 21  
7 x 4 = 28  
7 x 5 = 35  
7 x 6 = 42  
7 x 7 = 49  
7 x 8 = 56  
7 x 9 = 63  
7 x 10 = 70  
7 x 11 = 77  
7 x 12 = 84  
7 x 13 = 91  
7 x 14 = 98

- **Contrainte**
  - Transmettre un chiffre correspondant à la table de multiplication
  - Et un chiffre correspondant au chiffre maximal via GET

# Transmettre des données: GET

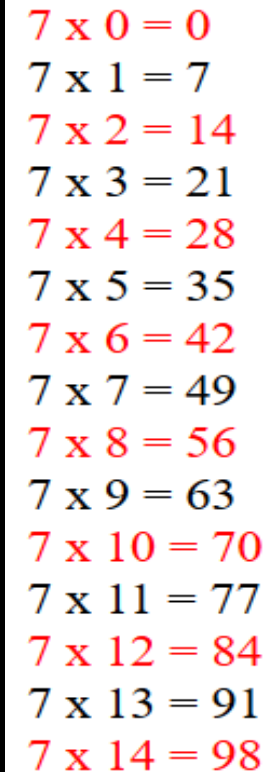
<http://localhost/indexTP0.php?page=xxx>

```
<form action="indexTP0.php" method="get">
  <p><input type="text" name="page" /></p>
  <p><input type="submit" value="OK"/></p>
</form>
```

Dans fichier `indexTP0.php` (qui peut être ce même fichier HTML qui s'appelle)

```
<?php
    $ma_var = $_GET['page']; //
?>    $ma_var = xxx
```

- Afficher la table de multiplication d'un nombre
- Si il est pair, **en rouge**
- <http://.../indexTP0.php?table=7&max=14>



7 x 0 = 0  
7 x 1 = 7  
7 x 2 = 14  
7 x 3 = 21  
7 x 4 = 28  
7 x 5 = 35  
7 x 6 = 42  
7 x 7 = 49  
7 x 8 = 56  
7 x 9 = 63  
7 x 10 = 70  
7 x 11 = 77  
7 x 12 = 84  
7 x 13 = 91  
7 x 14 = 98

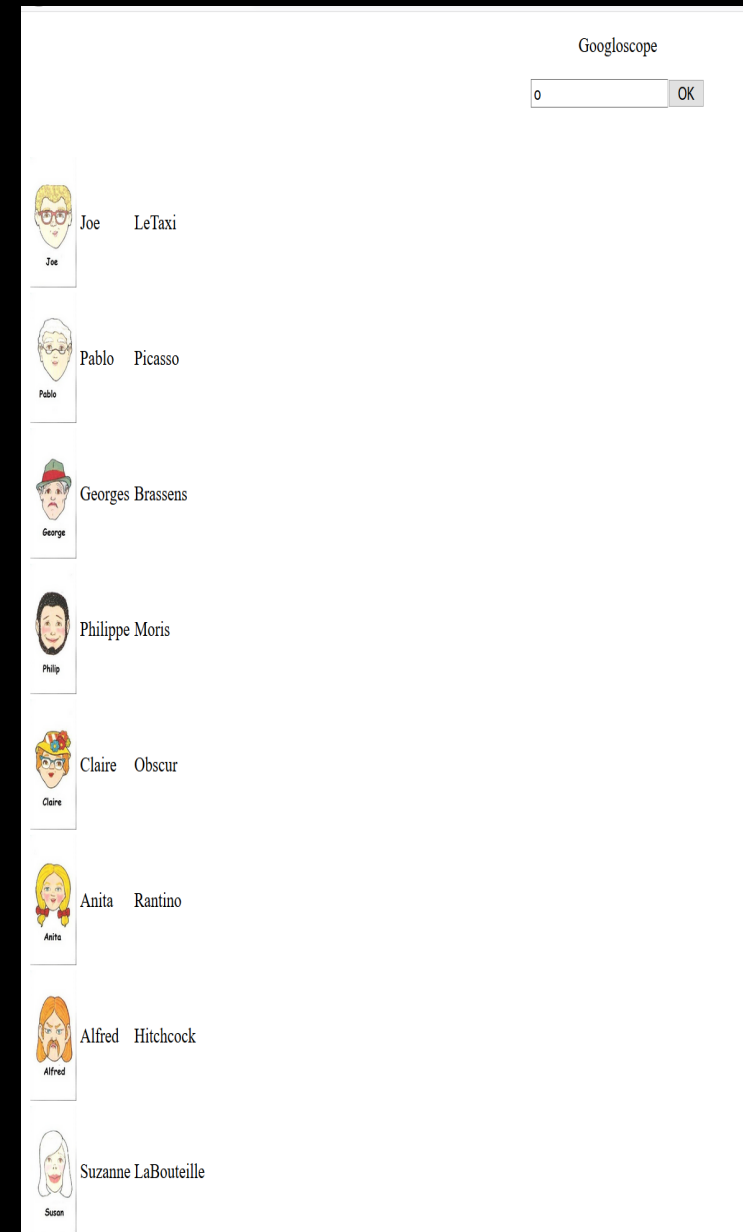
- **Contrainte**
  - Transmettre un chiffre correspondant à la table de multiplication
  - Et un chiffre correspondant à au chiffre maximal via GET
  - Avec un formulaire

- \$login = « toto »
- \$mdp = « toto » => FAIL
- Ecrire une fonction qui vérifie que le mot de passe respecte les règles suivantes
  - Au moins 8 caractères
  - Au moins un lettre majuscule
  - Au moins un lettre minuscule
  - Au moins un chiffre
- **Contraintes**
  - Signature: paramètre1 string
  - Sortie: un booléen (valide ou non)
  - Regardez `str_split`, `strlen`, `ctype_upper`, `ctype_lower`, `ctype_digit`
  - Pas de regex pour le moment, sauf si vous avez fini !

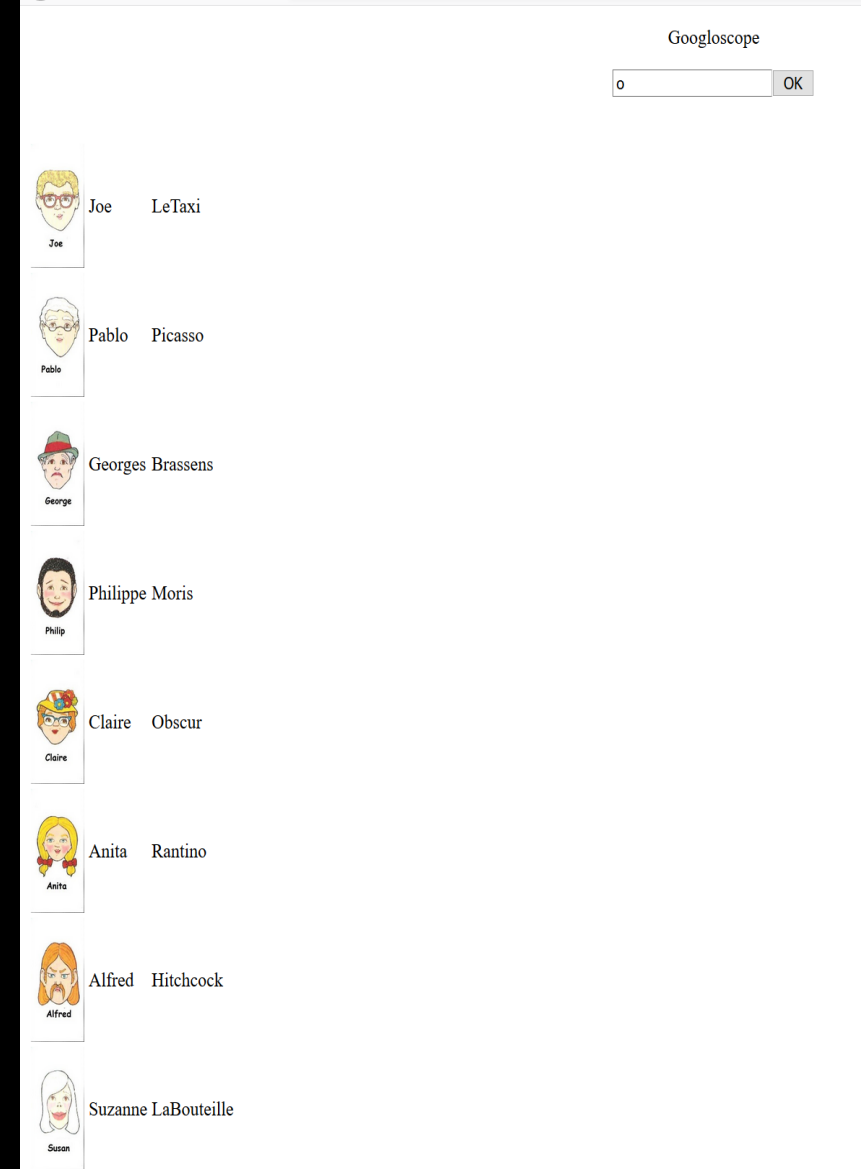


# Exercice

- Trombinoscope de la classe
- Afficher Photo + Prénom + Nom + Age
- Contraintes
  - Utiliser un tableau uniquement pour les données
  - Pas de limite de dimensions au tableau
  - Filtrer par prénom ou nom
  - Si pas de recherche → afficher tout le monde
  - Si pas de résultats, afficher « pas de résultat »
  - Aérer votre code à l'aide de fonctions
  - Architecture 2-tiers
  - De jolies pages et « têtes » ! (Schtroumpfs, chats, ...)
  - Images stockées sur votre serveur
  - Sur GitHub, par groupe de 2-3 (invitez-moi)
  - Typé, commenté, indenté, bien nommé
- Aide
  - `stripes($pTextToSearchIn, $pTextToFind)`
  - // allez chercher la définition
  - `empty($var)`
  - `array_push($lFoundEntries, $lIndex);`
- Par où commencer ?











- Quand on clique sur une personne
  - → on affiche uniquement cette personne
  - → et sa description
- Contraintes
  - Rajouter des ID et description dans les données
  - Utiliser une nouvelle « page » pour cet affichage
  - Filtrer par prénom ET nom



## Groupe

- Elisa - Alexandre
- Rémi - Jérôme
- Florian - Alexis
- Patricia - Khang - Romain
- Corinne - Hugo

Googloscope

 Joe	Joe	LeTaxi
 Pablo	Pablo	Picasso
 George	Georges	Brassens
 Philip	Philippe	Moris
 Claire	Claire	Obscur
 Anita	Anita	Rantino
 Alfred	Alfred	Hitchcock
 Susan	Suzanne	LaBouteille

May the force be with you





*That's all Folks!*