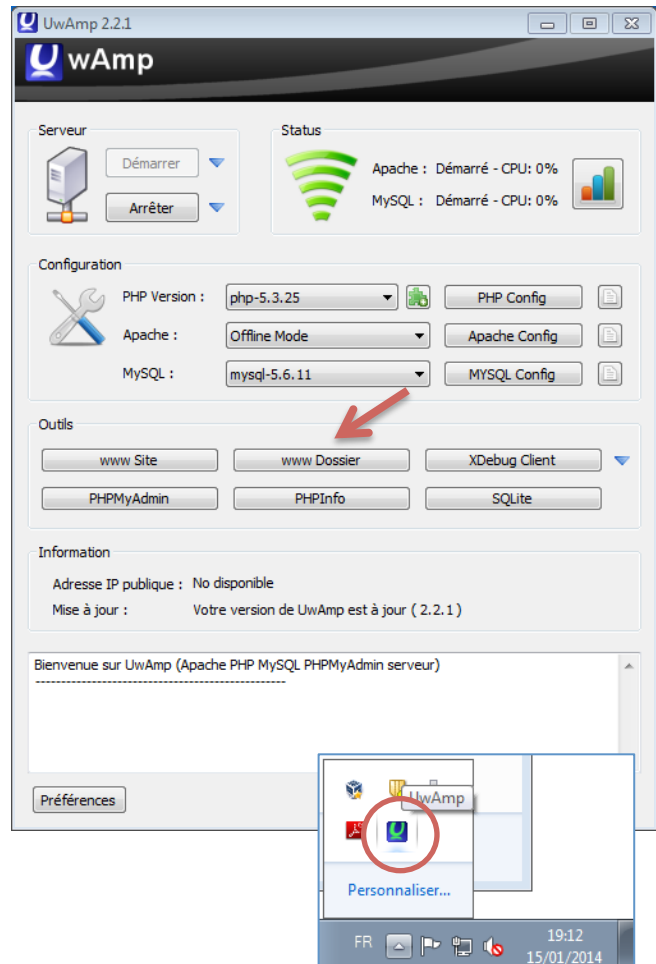
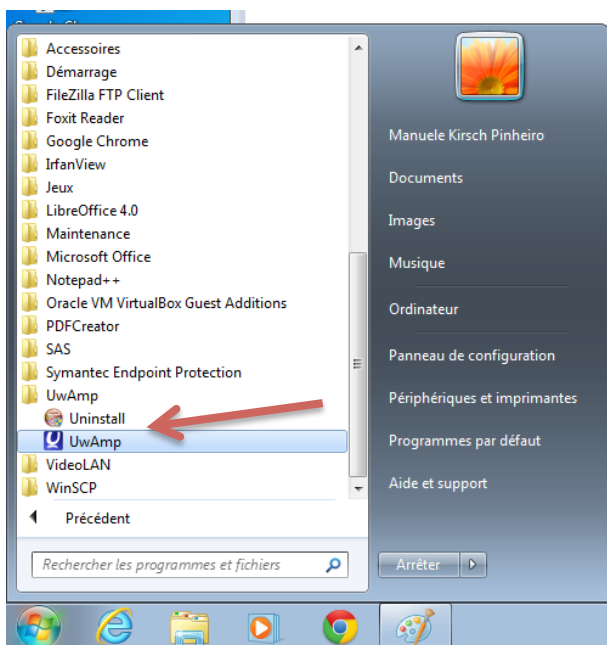


## Fiche de TD – Sites Web Dynamiques avec PHP

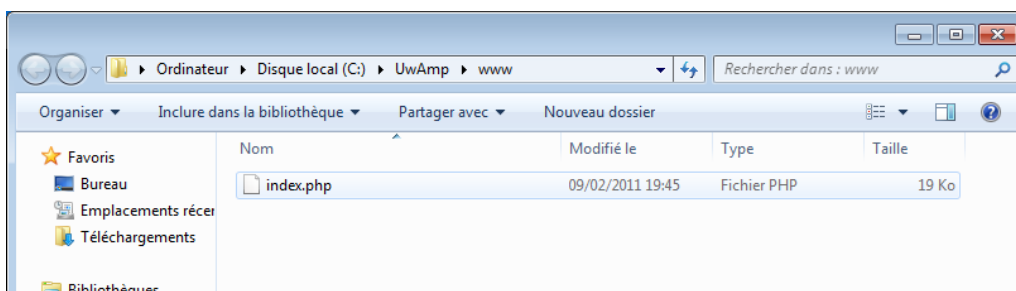
### Démarrage

Afin de pouvoir pratiquer le PHP, il vous faut un **serveur Web**. On peut transformer n'importe quel ordinateur en serveur Web à condition d'avoir installé les bons logiciels. Des multiples alternatives existent, autant pour la plateforme Windows (**UwAmp**, Wamp, Mov'Amp, EasyPHP), que pour la plateforme Mac (Mamp). Ces logiciels installent sur les machines non seulement un serveur Web, mais également PHP et même MySQL. Dans les salles machines, nous avons installé **UwAmp**. Il faut donc le démarrer avant de commencer cette fiche de TD.



Une fois démarré, UwAmp ouvrira une fenêtre de contrôle (image en haut à droite) et se mettra en exécution en arrière plan (indiqué par l'icône à côté de l'horloge).

A partir de la fenêtre de contrôle, nous allons pouvoir accéder au dossier « **www** », où doivent se trouver les pages pour notre site. En cliquant sur le bouton « **www Dossier** », UwAmp nous ouvrira le dossier **c:\UwAmp\www**, qui est le dossier utilisé par défaut dans nos salles de TDs. C'est dans ce dossier qu'on enregistrera les pages qu'on réalisera tout au long de cette fiche.



Pour éditer nos pages Web (HTML, CSS et PHP), nous allons utiliser « **Notepad++** », que nous avons déjà utilisé pour les fiches d'exercices sur HTML et CSS. Les pages PHP doivent ainsi être enregistrées dans le dossier **www** (indiqué ci-dessus) pour que le serveur Web **UwAmp** puisse les interpréter. A titre d'exemple, nous allons donc d'abord créer une première page PHP (voir image ci-dessous) que nous allons enregistrer dans ce dossier « **c:\UwAmp\www** » sous le nom « **premier.php** ». **Attention à bien enregistrer la page au format PHP (et non TXT).**

The image shows a workflow for creating and displaying a PHP page. It consists of three main parts:

- Notepad++ Editor:** The top window shows the code for 'premier.php'. The code includes an HTML header and a PHP block that sets the default timezone to Europe/Paris and displays the current date and time. A red arrow points to the PHP code.
- File Save Dialog:** The middle window, titled 'Enregistrer sous', shows the file being saved in the 'www' directory. The file name is 'premier.php' and the type is 'Fichier PHP'. A red arrow points to the 'www' directory in the 'Enregistrer dans' field.
- Browser View:** The bottom-left window shows a browser displaying the output of the PHP page: 'Mon premier exemple' and 'Paris, le 10 / 03 / 2014'. A red arrow points to the browser tab.
- File Save Dialog (Bottom):** The bottom-right window shows the 'Enregistrer' button being clicked, with a red arrow pointing to it. The file name 'premier.php' and the file type 'PHP Hypertext Preprocessor file (\*.php)' are visible.

Une fois enregistrée sur notre serveur local, notre page est accessible par le Web : <http://localhost/premier.php>

## Exercices :

- 1) Dans une boutique de vêtements, on souhaite appliquer une remise de prix en fonction de la valeur total des achats. Pour plus de 50€ en achat, on souhaite offrir aux clients une remise de 10%. Pour plus de 100€ d'achat, on lui offrira une remise de 20%, alors que pour plus de 150€ d'achats, on offrira 25%. Construire une page Web permettant à un client d'indiquer le montant de ses achats et de savoir ainsi le montant qu'il aura en remise.

Dans cet exercice, nous allons implémenter un des algorithmes que nous avons réalisé lors de la Fiche n° 3. Pour le réaliser, nous avons besoin d'une valeur en entrée correspondant au montant total d'achats. On va donc devoir demander à l'utilisateur de nous fournir cette valeur. Pour interagir avec notre utilisateur, nous allons lui proposer un formulaire très simple, où il pourra renseigner ce montant. Nous allons donc démarrer l'exercice par la **création d'un formulaire en HTML** comme celui-ci.

Pour faire notre formulaire, nous allons donc créer un document **.html** que nous allons enregistrer dans notre serveur Web (sur le fameux dossier « **c:\UwAmp\www** »).

The image shows a Notepad++ editor window on the left and a browser window on the right. The browser window displays a form titled 'Devis Remise' with a text input field labeled 'Montant :', a 'Calculer' button, and a title bar 'Fiche PHP - Exercice 1'. The Notepad++ window shows the following HTML code:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Fiche PHP - Exercice 1</title>
5  </head>
6  <body>
7
8  <h1>Devis Remise</h1>
9
10 <form action="fichePHP-0.php" method="post">
11 <label>Montant :</label>
12 <input type="text" name="montant" />
13 <input type="submit" value="Calculer" />
14 </form>
15 </body>
16 </html>

```

Red arrows point from the browser preview to the corresponding HTML tags in the code: one from the title to the <title> tag, one from the form to the <form> tag, one from the label to the <label> tag, one from the text input to the <input type="text" name="montant" /> tag, and one from the submit button to the <input type="submit" value="Calculer" /> tag.

Cette page Web devra contenir, en plus des balises traditionnelles (**html, head, body...**), une balise spéciale, la balise **form**, qui sert à construire un formulaire.

Deux informations sont essentielles à un formulaire : la méthode et l'action. L'action (**action=...**) indique à qui les données récoltées dans le formulaire seront transmises, alors que la méthode (**method=...**) indique au navigateur comment il doit transférer

ces données. L'action va donc indiquer la page PHP qui devra traiter ces données (donc, celle qui implémentera notre algorithme). Pour la méthode, deux méthodes sont possibles, le GET et le POST. Nous allons utiliser la méthode POST.

Pour l'entrée des données, nous allons proposer à l'utilisateur un champ texte à remplir et un bouton. Tous les deux sont représentés par une balise « **input** » : **<input type="text" ... >** pour le champ de texte et **<input type="submit" ... >** pour le bouton. Très important : notre champs texte devra aussi avoir un nom (**<input type="text" name="montant" />**) qui va nous permettre après de récupérer les données que l'utilisateur aurait rempli à l'intérieur

Une fois terminé notre formulaire, on va pouvoir se concentrer sur notre page PHP. Pour commencer, nous allons créer une nouvelle page et l'enregistrer sur le dossier dossier « **c:\UwAmp\www**. Attention à l'enregistrer avec le même nom que nous avons indiqué dans « **form action="..."** » de notre formulaire.

Comme on peut voir dans l'image ci-contre, une page PHP est une page Web où on introduit des morceaux de code PHP. Ceux-ci sont délimités par la balise spéciale « **< ?php .... >** ».

Dans un premier moment, nous allons simplement récupérer les informations de notre champ de texte « montant ». Puisque, dans notre formulaire, nous avons indiqué la méthode POST (**method="post"**), les données seront disponibles pour nous dans le **tableau associatif** `$_POST`. On récupère les informations et les stocke dans une variable (« `$mont` ») :

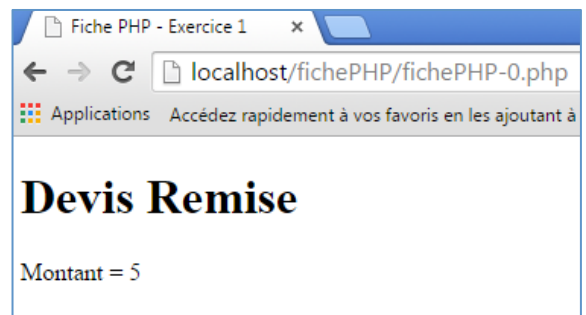
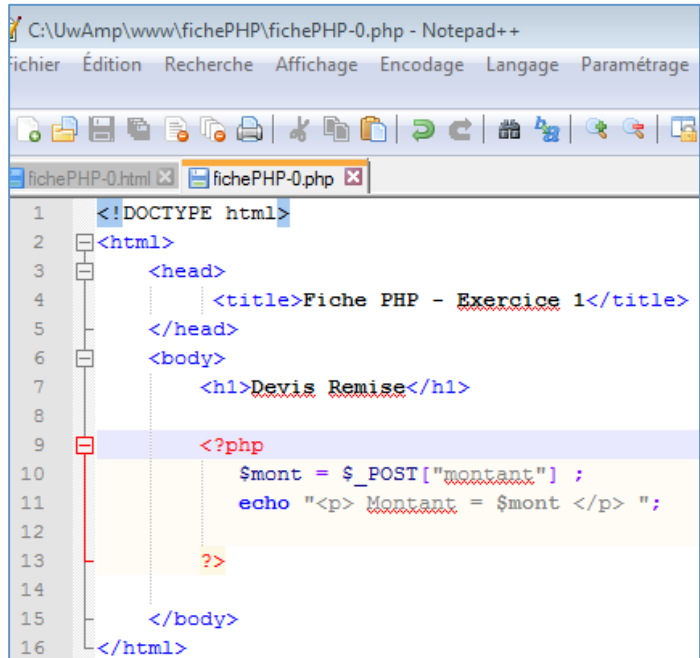
```
$mont = $_POST["montant"] ;
```

Puis on va afficher ce montant à l'aide de l'instruction « echo » :

```
echo "<p> Montant = $mont </p> " ;
```

Ce qui va nous donner une sortie comme celle à côté, si on met la valeur « 5 » dans notre formulaire et qu'on clique sur « calculer ».

```
if ($mont >= 150) {
    $remise = $mont * 0.25 ;
}
elseif ($mont >= 100) {
    $remise = $mont * 0.20 ;
}
elseif ($mont >= 50) {
    $remise = $mont * 0.10 ;
}
else {
    $remise = 0;
}
echo "<p> remise = $remise </p>";
```



Il nous reste qu'à terminer notre code, suivant l'algorithme que nous avons défini dans la fiche n° 3. Celui-ci était composé par une séquence de tests « **Si ... Sinon Si... Sinon** » qu'on va reproduire en PHP à l'aide de l'instruction « **if ... elseif ... else ...** », suivant ce que nous avons vu en cours.

Puis, nous allons conclure en affichant, à l'aide de l'instruction « **echo** », la valeur de remise que nous venons de calculer.



- 2) Rédiger un formulaire HTML qui demande à l'utilisateur son salaire mensuel. Rédiger une page PHP qui, à partir des informations fournies par l'utilisateur dans le formulaire, indiquer à l'utilisateur quelle la tranche d'imposition qui convient à ses revenus. Les barèmes d'imposition sont indiqués dans la table ci-dessous.

Tranches de revenus et taux applicables aux revenus 2011 (impôt 2012)	
jusqu'à 5 963 €	0 %
de 5 963 € à 11 896 €	5,5 %
de 11 896 € à 26 420 €	14 %
de 26 420 € à 70 830 €	30 %
plus de 70 830 €	41 %

Comme pour l'exercice précédent, pour faire cet exercice, il faudra construire à la fois une page HTML contenant un formulaire comme celui illustré par la figure ci-dessus, et une page PHP qui réalisera les « calculs » (en occurrence, analyser la valeur indiquée pour le salaire). On a donc deux fichiers à faire, un **.html** (disons « **fichePHP-1.html** ») et un **.php** (appelons-le « **fichePHP-1.php** »).

Pour réaliser le formulaire dans la page HTML, on utilisera à nouveau la balise « **form** », laquelle contiendra au moins un champ d'entrée (balise « **input** ») pour que l'utilisateur indique son salaire. Cette information sera reprise par la page PHP. Le lien entre la page HTML contenant le formulaire et la page PHP se fait dans la balise « **form** » à travers son attribut « **action** ».

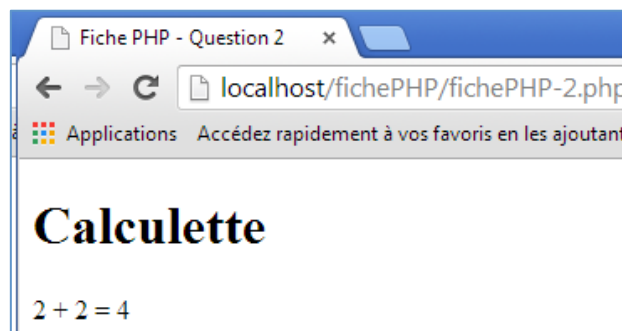
Dans la page PHP, nous allons récupérer les données remplies par l'utilisateur dans le formulaire. Pour cela, on utilisera à nouveau le tableau « **\$\_POST** » (puisque'on a indiqué la méthode « **post** » dans la balise « **form** »). La valeur de chaque champ du formulaire est accessible à travers ce tableau associatif par son nom (par exemple, pour un

```
<form action="fichePHP-1.php" method="post">
...
<label>Votre salaire mensuel : </label>
<input type="number" name="salaire" />
<br/>
...
<input type="submit" value="Calculer" />
<input type="reset" value="Restaurer"/>
</form>
```

```
<?php
...
$salaire = $_POST["salaire"];
...
if ( $salaireAnnuel < 5963 )
{ $tranche = 0; }
elseif ( $salaireAnnuel < 11896 )
{ $tranche = 5.5; }
...
else
{ $tranche = 41; }
... ?>
```

champ nommé « salaire », on fera `$_POST["salaire"]` ). Une fois qu'on aura récupéré le contenu du champ « **salaire** », on pourra alors calculer la valeur du salaire annuel (`$salaire * 12`), puis le tester à l'aide de l'instruction « **if** » afin de connaître dans quelle catégorie de salaire notre salarié se trouve.

### 3) Réaliser une calculatrice simple (avec les quatre opérations de base : +, -, \* et / ) à l'aide des langages HTML/CSS et PHP.



Comme pour l'exercice précédent, nous allons faire deux fichiers, un `.html` (« `fichePHP-2.html` », par exemple) et un `.php` (« `fichePHP-2.php` »). Une calculatrice peut être réalisée à l'aide d'un formulaire très simple, avec les deux champs pour les opérands (balise « `input` » de `type` « `number` ») et des boutons de type « `radio` » (balise « `input` » de `type` « `radio` ») pour chacune des opérations.

Dans notre page PHP (« `fichePHP-2.php` »), nous allons devoir récupérer ces informations à partir du tableau associatif « `$_POST` », comme pour l'exercice précédent. Nous allons faire, par exemple, « `$_POST["op1"]` » pour récupérer le contenu du champ « `op1` », et « `$_POST["opérateur"]` » pour connaître l'opérateur choisi par l'utilisateur.

Une fois en possession de ces informations (les opérands et l'opérateur choisi), il nous suffit de tester, à l'aide des instructions « `if ... elseif ... else` », quel a été l'opérateur choisi et donc réaliser l'opération correspondante.

```
<form action="fichePHP-2.php" method="post">
...
  <input type="number" name="op1" />
  <input type="radio" name="opérateur"
    value="+" checked /> +
  <input type="radio" name="opérateur"
    value="-" /> -
...
  <input type="submit" value="Calculer" />
</form>
```

```
$opérateur = $_POST["opérateur"];
$op1 = $_POST["op1"];
...
if ( $opérateur == '+' )
{ $res = $op1 + $op2; }
elseif ( $opérateur == '-' )
...
echo "<p>$op1 $opérateur $op2 = $res </p>";
```

- 4) Une banque en ligne offre un nouvel investissement de type épargne, le Livret Alpha+, lequel offre pendant les 3 premières années un taux de 2% à l'an et de 1,5% pour les années suivantes. Construire un formulaire et une page PHP qui, à partir d'un montant initial renseigné par le client, montrent au client une prévision de ses gains pour les 8 prochaines années.

Année	Taux	Montant estimé
1 e année	2,00%	102,00
2 e année	2,04%	104,04
3 e année	2,0808%	106,1208
4 e année	1,591812%	107,712612
5 e année	1,61568918%	109,32830118
6 e année	1,6399245177%	110,9682256977
7 e année	1,6645233854635%	112,63274908317
8 e année	1,6894912362475%	114,32224031941

Comme pour les exercices précédents, dans celui-ci, nous allons construire une page HTML (comme celle illustrée ci-dessus à gauche) et une page PHP. La page HTML contiendra le formulaire dans lequel le client renseignera le montant initial. La procédure est la même que pour les exercices 1 et 2 : on créera un formulaire à l'aide de la balise « **form** », dans lequel on ajoutera au moins un champ (balise « **input** ») pour le montant. Ce formulaire fera appel à la page PHP que nous allons créer (« **fichePHP-3.php** ») à travers l'attribut « **action** » de la balise « **form** ».

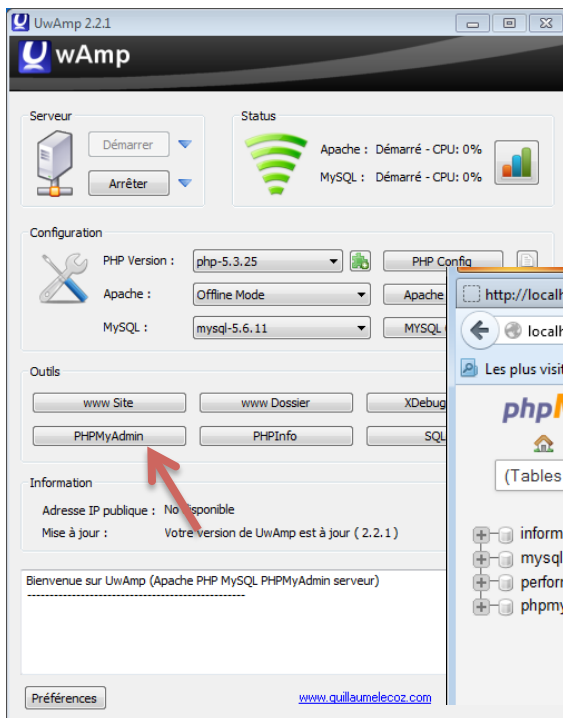
Dans la page PHP, nous allons à nouveau implémenter un algorithme que nous avons proposé dans la fiche n° 3. Plusieurs solutions existent, mais elles auront toutes d'abord besoin de récupérer les informations renseignées par le client, et notamment le montant initial, à l'aide du tableau associatif « **\$\_POST** », exactement comme nous l'avons fait dans l'exercice 1 (« **\$\_POST["montant"]** »).

```
$montant = $_POST["montant"];
...
$taux = 0.02;
for ($i=1; $i <= 3; $i++) {
    $rend = $montant * $taux;
    $montant = $montant + $rend;
    ...
}
```

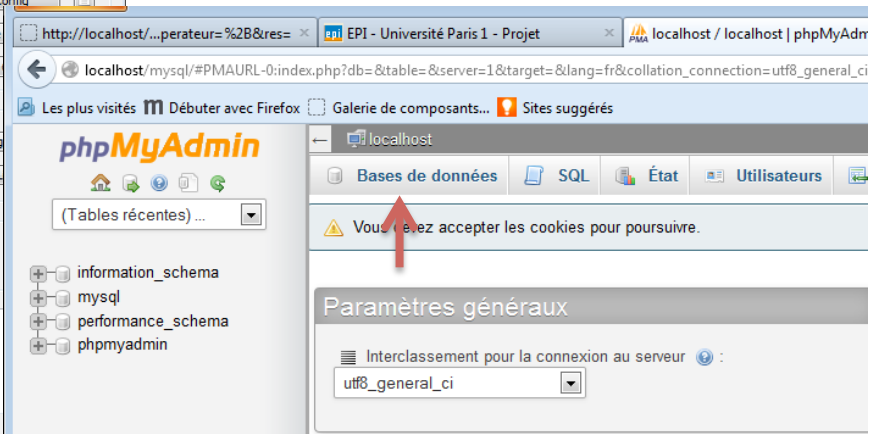
Ensuite, on devrait utiliser au moins une boucle pour pouvoir incrémenter ce montant initial d'un rendement selon un taux fixe 2% pendant 3 ans, puis de 1,5% pendant les 5 années suivantes. Par exemple, on peut faire une première boucle (une boucle « **for** », par exemple), qui va nous permettre de répéter une même opération (le fait d'incrémenter le montant), pour les 3 premières années. Puis, on peut faire exactement la même chose pour les 5 années suivantes, ce qui implique une nouvelle boucle, mais cette fois-ci, de l'année 4 à l'année 8 (« **for (\$i=4; \$i<=8; \$i++)** ») avec une valeur de taux de 1,5% (« **\$taux = 0.015;** »). Il s'agit d'une solution possible. A vous d'expérimenter d'autres solutions... ☺

- 5) A l'aide de l'outil phpMyAdmin (accessible à partir de la fenêtre de contrôle UwAmp), créer une base « Produits » contenant une table « Produit » avec les données indiquées ci-dessous.

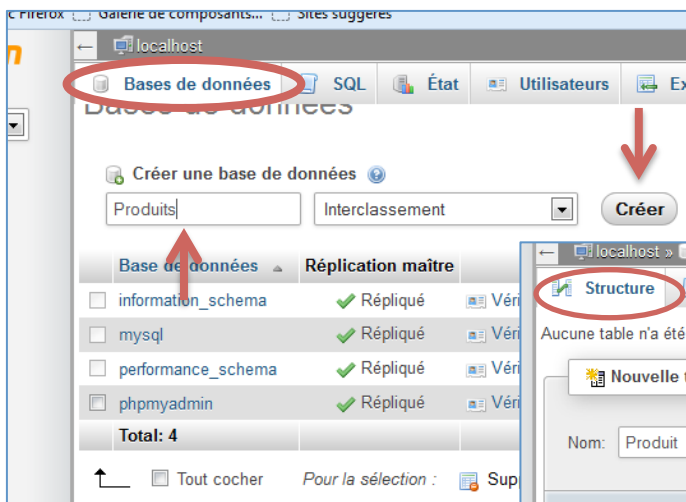
Produit (code , nom , description, prix )			
code (Int PRIMARY KEY Auto-Incrément)	nom (Varchar 25)	description (Varchar 50 NULL)	prix (Float)
1	Autocollant	Autocollant au symbole du club	15,00
2	T-shirt Official	T-shirt officiel du club	35,00
3	T-shirt baby-look	T-shirt au symbole du club, modèle féminin	25,00



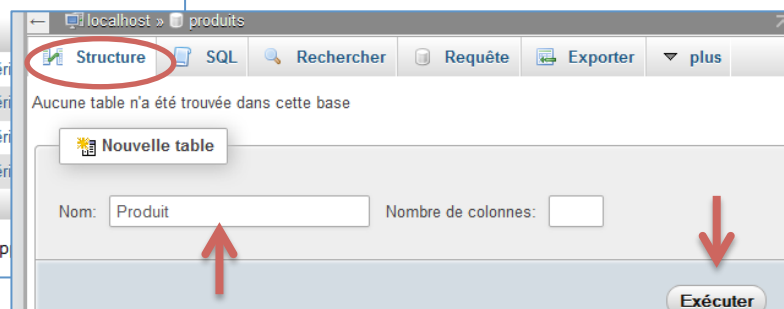
Pour faire cet exercice, nous allons devoir créer une nouvelle base de données dans notre serveur local. Pour cela, nous allons utiliser l'outil « **phpMyAdmin** » qui est disponible à partir de la fenêtre de contrôle **UwAmp** (voir figure ci-contre).



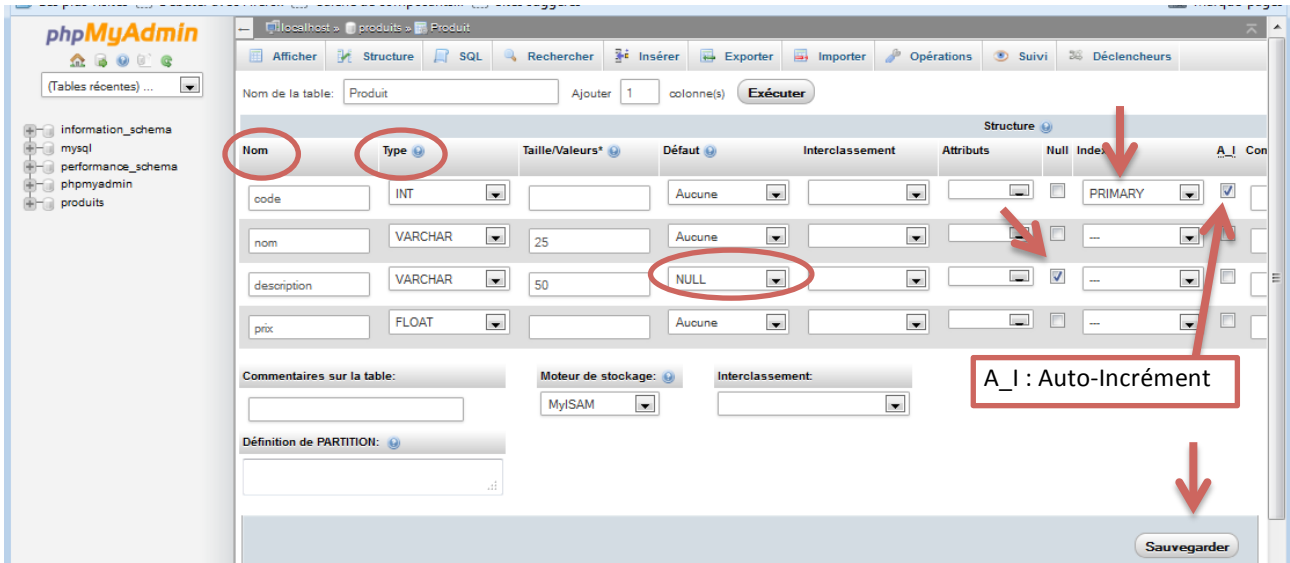
L'outil **phpMyAdmin** nous permet de créer des nouvelles bases de données et d'y ajouter de nouvelles tables et des nouvelles données. Pour créer une nouvelle base, il faut cliquer sur « **Bases de données** », puis renseigner le nom de la nouvelle base (« **Produits** » dans notre cas).



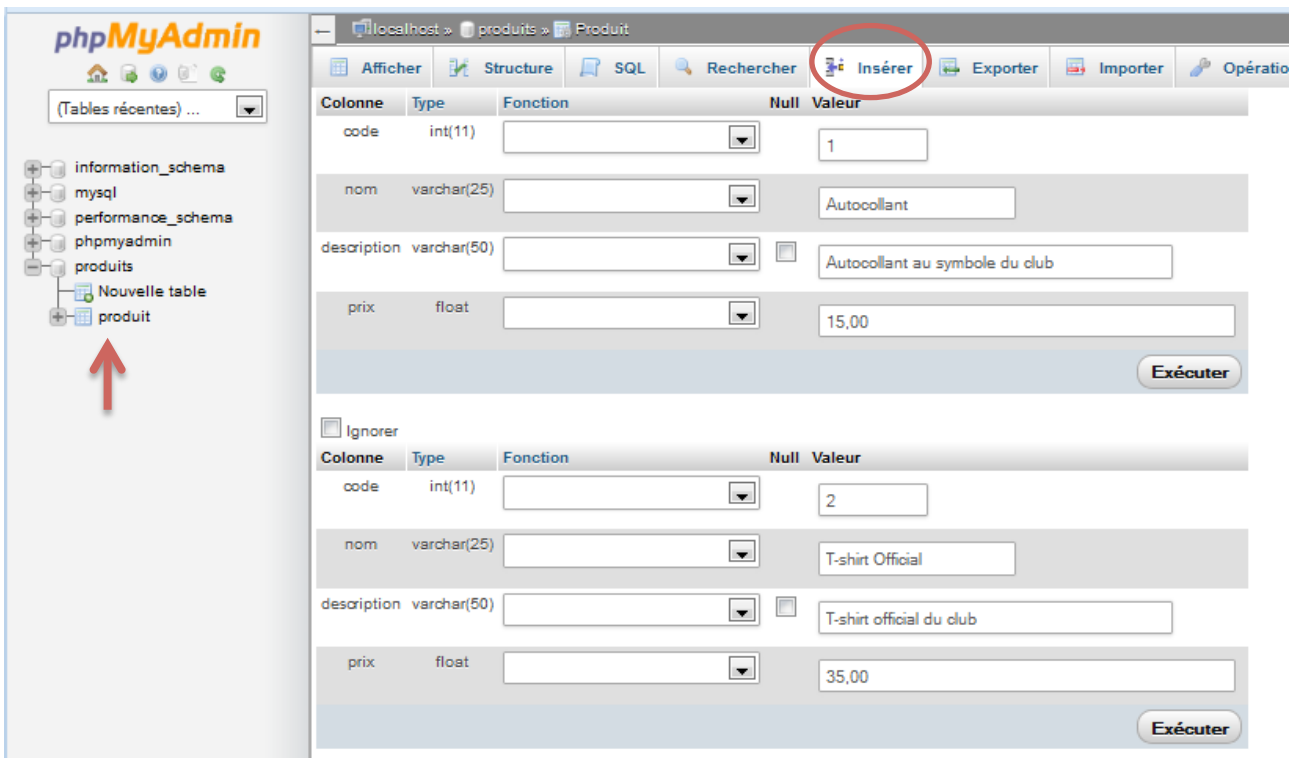
Une fois la base créé, nous allons pouvoir y **ajouter la table « Produit »**, et avec elle, chacun de ses champs.







Puis, en cliquant sur « insérer », nous allons pouvoir y insérer les données demandées. En cliquant sur « afficher », nous allons pouvoir vérifier après si nos données ont bien été insérées.



- 6) Toujours à l'aide de l'outil phpMyAdmin, modifier les privilèges de la base de données « Produits » afin de lui ajouter un nouvel utilisateur nommé « php » avec, pour mot de passe « php ».

La création d'un utilisateur est possible à partir du lien « privilèges » qui apparaît une fois qu'on clique sur la base de données. A partir de là, nous allons pouvoir ajouter un nouvel utilisateur.

Lors de la création de notre utilisateur « php », nous allons pouvoir lui attribuer tous les droits sur l'ensemble de la base de données « **Produit** ». A ne pas oublier d'indiquer que l'accès à cette base est « **local** » (puisque le serveur qui héberge la base de données et le même qui héberge les pages PHP).

### Ajouter un utilisateur

**Information pour la connexion**

Nom d'utilisateur: Entrez une valeur:  ←

Client: Local  ←

Mot de passe: Entrez une valeur:  ←

Entrer à nouveau:

Générer un mot de passe:

**Base de données pour cet utilisateur**

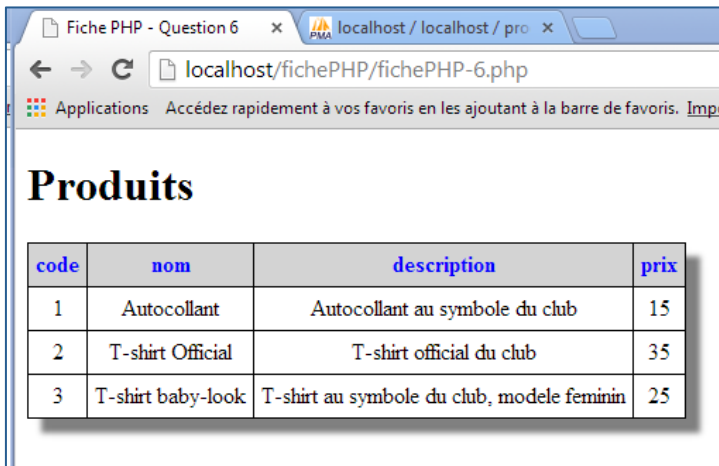
Créer une base portant son nom et donner à cet utilisateur tous les privilèges

Donner les privilèges passpartout (utilisateur\_%)

Donner tous les privilèges sur la base de données "produits"

←

- 7) Une fois les exercices 4 et 5 réalisés, créer une page PHP capable de se connecter à la base de données « Produits », utilisant l'utilisateur et le mot de passe que vous venez de créer, et de lister l'ensemble de produits présents dans la base de données. Vous pouvez utiliser pour cela la requête SQL « **SELECT \* FROM Produit** ».



code	nom	description	prix
1	Autocollant	Autocollant au symbole du club	15
2	T-shirt Official	T-shirt officiel du club	35
3	T-shirt baby-look	T-shirt au symbole du club, modele feminin	25

Dans cet exercice, nous allons créer une page PHP capable de se connecter à la base de données « **Produits** » que nous venons de créer afin d'y récupérer les informations de la table « **Produit** ».

La première étape consiste donc à se **connecter à la base de données** utilisant l'utilisateur que nous venons de créer (user « **php** », mot de passe « **php** »). Pour se connecter, nous avons besoin de connaître non seulement les informations de

l'utilisateur, mais également le nom du serveur (« **localhost** » pour nous) et de la base de données (« **Produits** » dans notre cas). Nous pouvons garder ces informations dans des variables.

Une fois en possession de ces informations, nous allons pouvoir nous connecter à l'aide d'un objet de la classe « **mysqli** ». Puisque toute connexion n'est pas forcément réussie (le serveur peut être éteint, ou on peut simplement oublier de démarrer Uwamp ☺), il est important de tester si la connexion s'est correctement effectuée. Pour cela, nous allons utiliser un test « **if** » : le test « **if (\$mysqli->connect\_errno)** » nous permet de savoir s'il y a eu un problème de connexion. Si c'est le cas, le mieux est d'arrêter l'exécution de la page PHP grâce à la fonction « **die** ».

Maintenant que nous avons la connexion avec la base de données établie, nous allons pouvoir lui soumettre une requête pour récupérer les informations sur les produits. La requête SQL « **select \* from Produits** » nous permet ceci. Pour la soumettre à la base de données, nous allons utiliser l'opération « **query** » de l'objet « **mysqli** » (« **\$mysqli->query** »).

Une fois en possession des résultats de la requête, nous allons pouvoir les afficher dans un tableau (un « **echo " <table> " ;** » nous aidera pour cela). Ces résultats incluent non seulement les données (nos produits), mais aussi les noms des colonnes (attributs) de la table « **Produit** ». Nous allons donc pouvoir récupérer les noms des colonnes pour les afficher dans notre tableau. Ceci est possible à travers l'opération « **fetch\_fields** » de l'objet **\$resultat** (« **\$resultat -> fetch\_fields()** »). Cette opération nous retournera un tableau associatif que nous allons parcourir à l'aide d'une boucle « **foreach** ».

```
//données pour la connexion à la Bdd
$host = "localhost";
$user="php";
$password="php";
$database = "Produits";
//connexion à la Bdd
$mysqli = new mysqli($host, $user,
    $password, $database);
```

```
if ($mysqli->connect_errno) {
    die ("Echec lors de la connexion "
        . $mysqli->connect_error);
}
```

```
$sql = "SELECT * FROM Produit" ;
$resultat = $mysqli->query($sql);
```

```
$titres = $resultat -> fetch_fields() ;
foreach ($titres as $colonne) {
    echo "<th> "
        . $colonne->name . " </th>";
}
```

Après avoir affiché les noms de colonnes, nous pouvons nous concentrer sur les données proprement parlées. Nous allons récupérer ligne à ligne (c'est-à-dire, produit à produit), à l'aide de l'opération « **fetch\_object** » de l'objet **\$resultat** dans une boucle (« **while ( \$ligne = \$resultat->fetch\_object() )** »). Cette opération nous retournera un objet dont les attributs correspondent aux colonnes de la table **Produit**. Nous pouvons ainsi utiliser une boucle « **foreach** » pour récupérer la valeur de chacun de ces attributs (« **foreach (\$ligne as \$colonne=>\$val)** »).

Une fois les données récupérées et affichées, nous pouvons fermer la connexion que nous avons ouvert tout au début (« **\$mysqli->close();** »).

```
while ($ligne = $resultat->fetch_object()) {
    echo "<tr>";
    //on récupère chaque attribut de la ligne
    foreach ($ligne as $colonne=>$val) {
        echo "<td> " . $val . " </td>";
    }
    echo "</tr>";
}
```

## 8) Construire un formulaire HTML et une page PHP permettant à un utilisateur de insérer un nouveau produit dans la base de données.

Dans cet exercice, nous aurons besoin de construire une page HTML contenant un formulaire qui nous permettra de renseigner les données du nouveau produit. Nous devons avoir ainsi un champs pour chaque attribut de la table « **Produit** ».

Ces informations seront récupérées par la page PHP, toujours à l'aide du tableau associatif « **\$\_POST** ». Puisque nous allons utiliser celles-ci

pour créer un nouveau produit, nous devons vérifier si toutes les informations nécessaires (et notamment le nom et le prix) ont bien été remplies. Ceci se fera à l'aide d'un test « **if** » et de la fonction « **empty** ». Nous pouvons aussi tester si la valeur renseignée pour le prix est bel et bien un numéro (et pas un texte) à l'aide de la fonction « **is\_numeric** », ce qui nous donne le test suivant « **if (empty(\$nom) OR empty(\$prix) OR !is\_numeric(\$prix) )** ».

```
<form action="fichePHP-7.php" method="post">
  <label>Code produit</label>
  <input type="number" name="code" /> <br/>
  ...
  <input type="text" name="nom" maxlength="25" />
  ...
  <input type="text" name="prix" maxlength="10" />
  <br/>
  <input type="submit" value="Calculer" />
  <input type="reset" value="Restaurer"/>
</form>
```

Une fois en possession de toutes les informations nécessaires, nous allons pouvoir les insérer dans la base de données. Pour le faire, la procédure est très similaire à celle réalisée dans l'exercice précédent. Comme dans l'exercice précédent, nous allons ouvrir une connexion à l'aide d'un objet « **mysqli** ». Puis, nous allons lui soumettre une requête. Celle-ci sera une requête d'inclusion :

```
$sql = "INSERT INTO Produit (code, nom, description, prix)
VALUES ( $code, '$nom', '$descr', $prix);
```

Pour soumettre cette requête, nous allons utiliser, comme dans l'exercice précédent, l'opération « **query** » de l'objet **\$mysqli** : « **\$resultat = \$mysqli->query(\$sql);** ».

```
$mysqli = new mysqli($host, $user, $password, $database);

if ($mysqli->connect_errno) {
    die ("Echec lors de la connexion : "
        . $mysqli->connect_error);
}

$sql = "INSERT INTO Produit ( code, nom, description, prix)
VALUES ( $code, '$nom', '$descr', $prix );" ;

$resultat = $mysqli->query($sql);
```

```
$code = $_POST["code"];
$nom = $_POST["nom"];
$descr = $_POST["descr"];
$prix = $_POST["prix"];

//on vérifie qu'on a les données
if (empty($nom) OR empty($prix)
    OR !is_numeric($prix) ) {
    echo "<p> Impossible de créer un nouveau
produit </p>";
}
else {
    // si on n'a pas une valeur pour le code,
    //on laisse l'auto-increment en choisir une.
    if ( empty($code) ) { $code = '\N'; }
    ... }
}
```

Pour être sûr que notre requête a bien marché (et donc que le nouveau produit a été introduit dans la base de données), nous allons tester l'objet « **\$resultat** ». Les requêtes « **INSERT** » retournent vrai (ou faux) selon si elles ont réussi pas. Nous pouvons donc tester la valeur de **\$resultat** avec un « **if** » : « **if (! \$resultat )** ».

```
if (! $resultat ) {
    echo "<p>Impossible de créer le produit $nom. </p>";
}
else {
    //on recupere le dernier id utilisé pour le code
    $code = $mysqli->insert_id;
    echo "<p> Nouveau produit : $code $nom
        $descr $prix </p>";
}
}
```