

Développement Web – PHP

Cours 2

Fabrice BOISSIER

Fabrice.Boissier@univ-paris1.fr

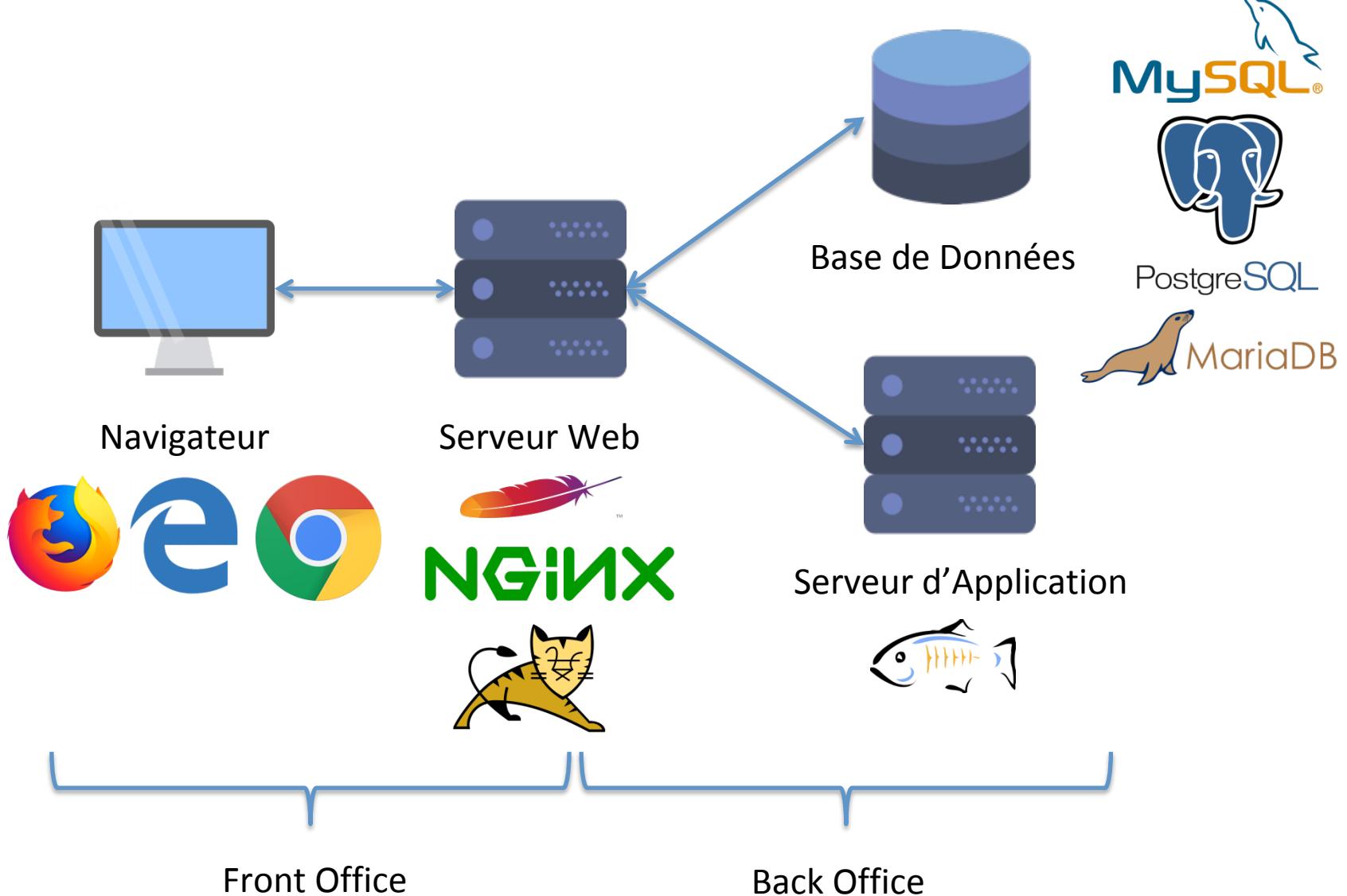
Ali JAFFAL

Ali.Jaffal@univ-paris1.fr

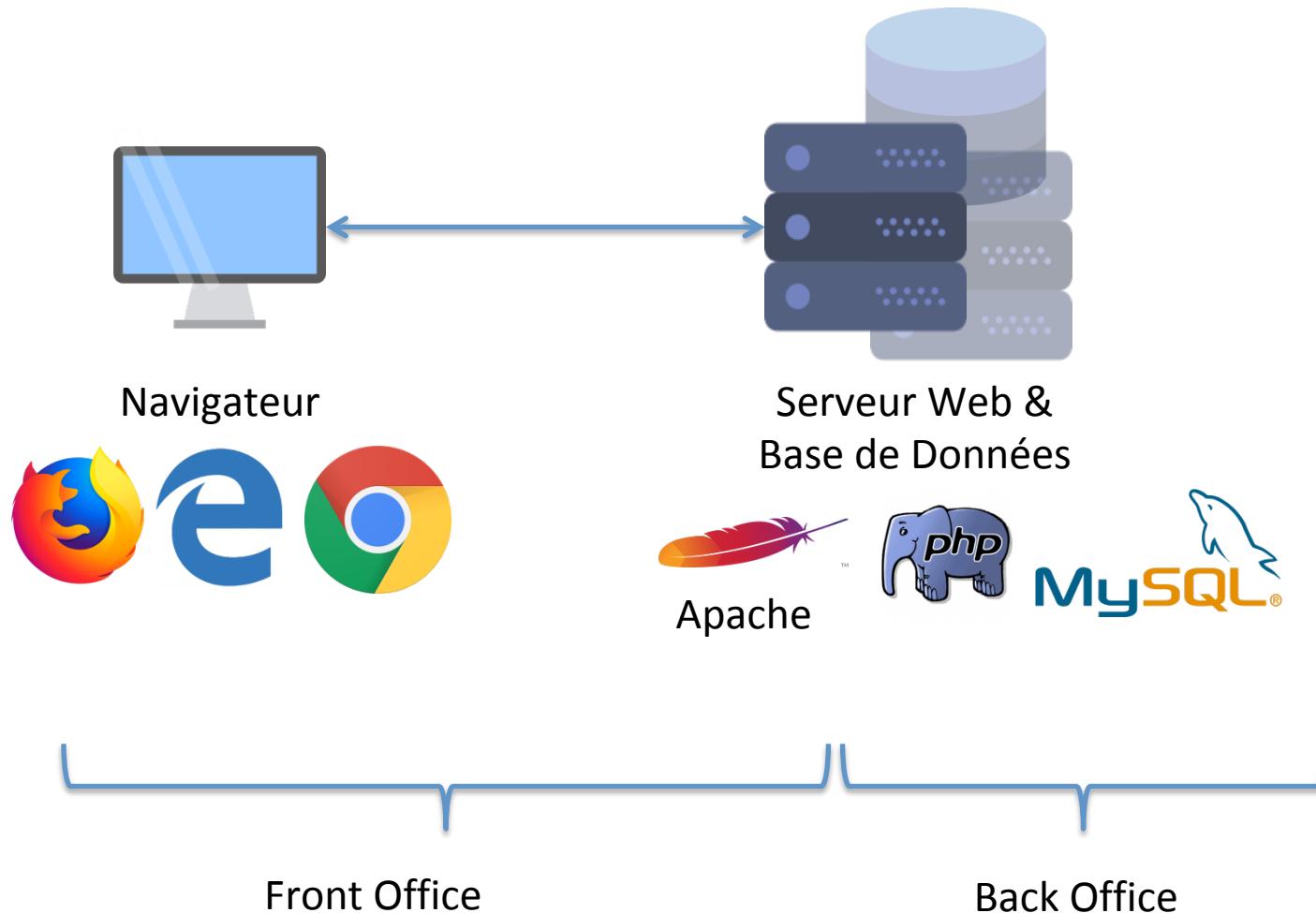
Bureau C.14.05

2018-2019

Rappels



Rappels

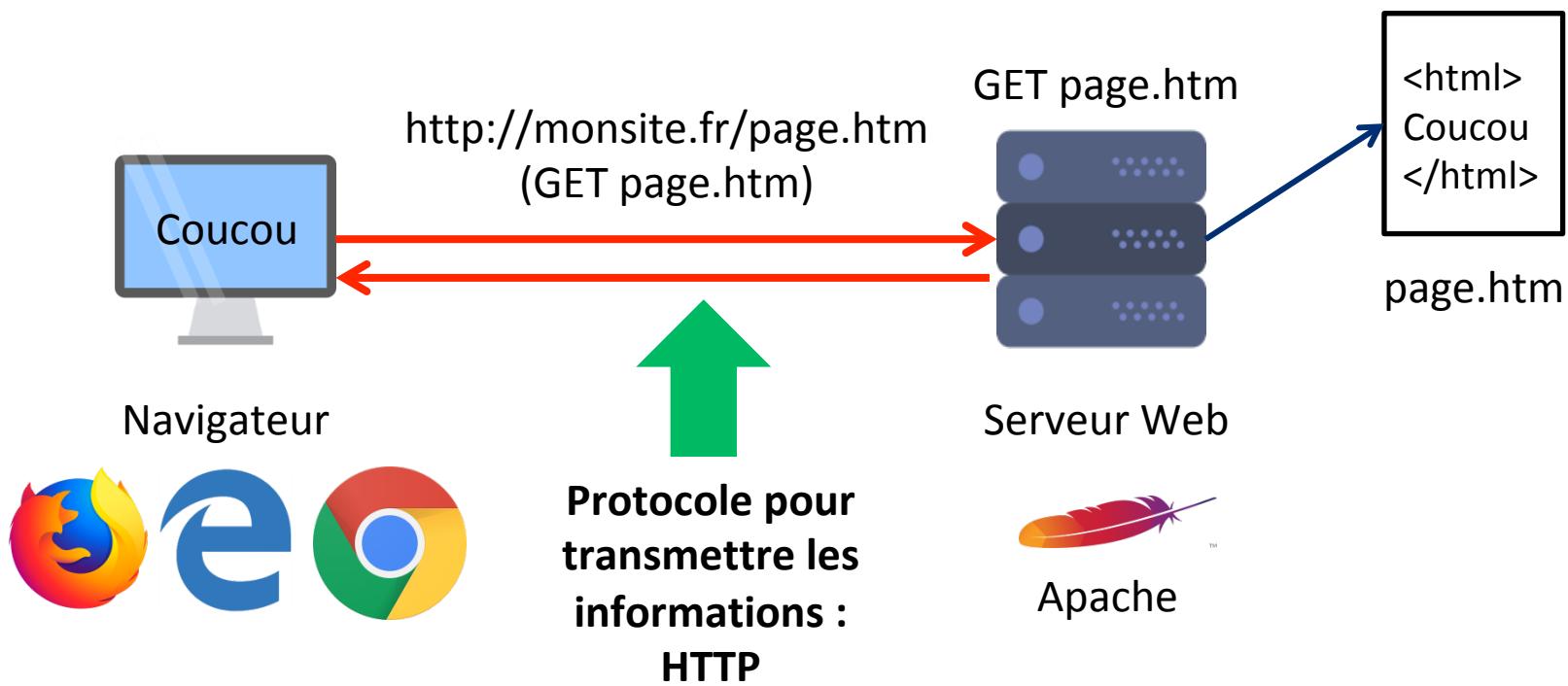


Rappels

- WAMP/MAMP/LAMP/XAMP :
 - Windows/Mac/Linux/Unix
 - Apache = Serveur Web
 - MySQL = Base de Données
 - PHP = Préprocesseur des pages web

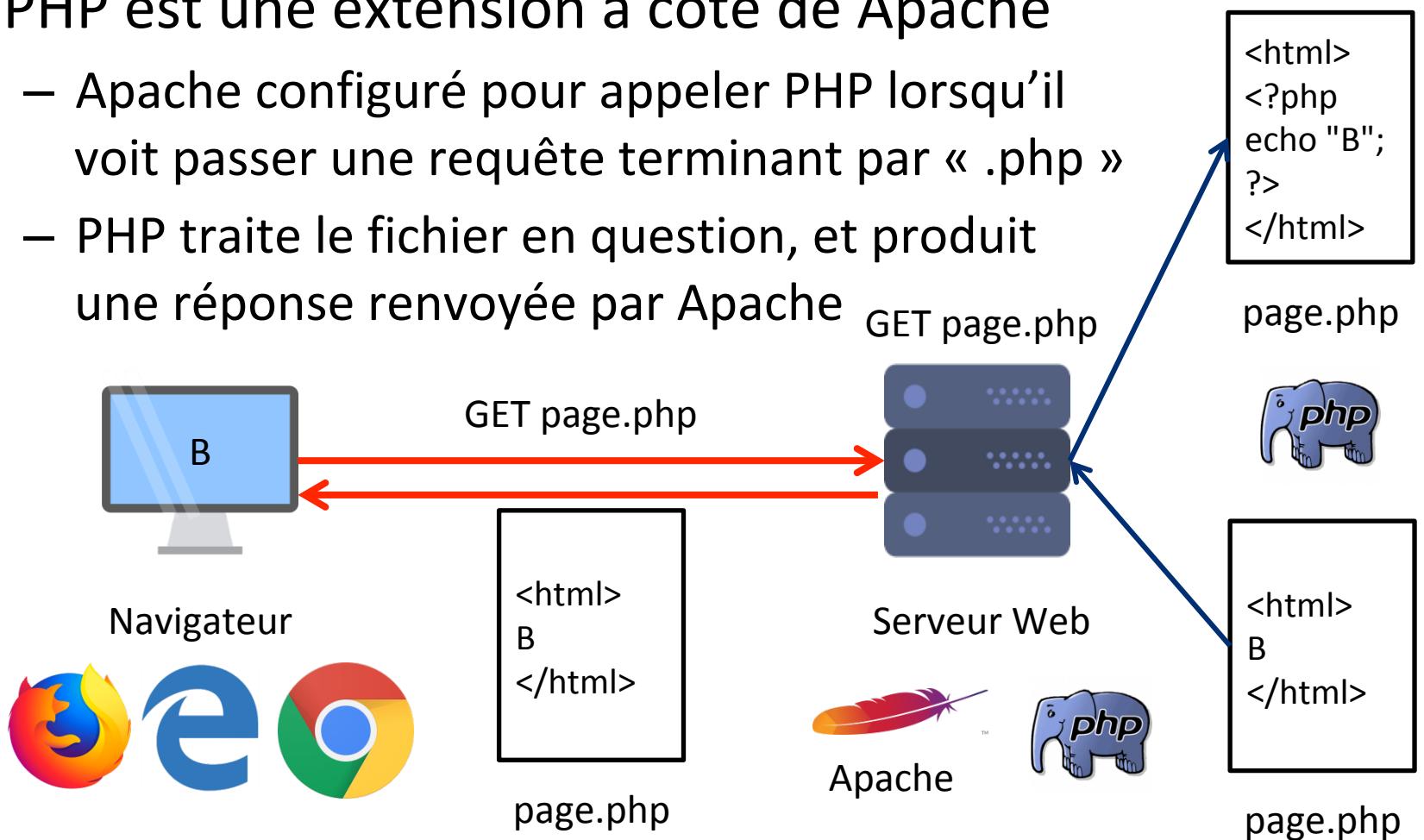
Rappels

- Communications entre « navigateur web » et « serveur web » se font avec protocole HTTP



Rappels

- PHP est une extension à côté de Apache
 - Apache configuré pour appeler PHP lorsqu'il voit passer une requête terminant par « .php »
 - PHP traite le fichier en question, et produit une réponse renvoyée par Apache



Caractéristiques PHP

- Langage faiblement typé
 - Beaucoup plus flexible...
 - ...mais possibilité de faire n'importe quoi
- Langage interprété // *Sur le terminal de Linux*
 - CLI disponible pour scripts // *Command Line Interface*
- Usage dans ce cours : extension apache

Rappel : Formulaires HTML

- **Communication entre le client (navigateur) et le serveur (php)**
 - Les **formulaires en HTML** permettent de recueillir des données auprès de l'utilisateur
 - Les données sont ensuite **communiquées** à un **programme**
 - Le navigateur **envoie les données** récoltées par les formulaires au serveur
 - Le programme (page PHP) récupère les données grâce à des **variables**



Rappel : Formulaires HTML

- Un **formulaire HTML** est défini par la balise
<form ...> ... </form>
 - Tous les éléments sont à l'intérieur de la balise
**<form name="*nomFormulaire*"
action="*page.php*" method="**get | post**" > </form>**

action : à qui on envoie les données

method: comment on envoie les données

- Les **champs du formulaire** sont introduits par différents balises :

- **<input type="..." name="..." value="..." id="..." />**
- **<textarea name="..." id="..." cols="..." rows="..." > ... </textarea>**
- **<select name="..." id="..." size="..." >
<option value="..." > ... </option> </select>**

Rappel : Formulaires HTML

The diagram illustrates the structure of an HTML form based on the screenshot above. Blue arrows point from the browser elements to their corresponding code snippets.

- Nom:** Points to the text input field containing "votre nom".
Code:

```
<input type="text" name="nomClient" value="votre nom" size="40" maxlength="150" />
```
- Produit:** Points to the dropdown menu showing "Super Kdo".
Code:

```
<select name="produit"><option value="SuperKdo">Super Kdo</option>...</select>
```
- Opinion:** Points to the text area containing "Votre opinion sur nos produits".
Code:

```
<textarea name="opinionClient" cols="40" rows="5">Votre opinion sur nos produits</textarea>
```
- Envoyer:** Points to the "Envoyer" button.
Code:

```
<input type="submit" value="Envoyer" />
```
- Nettoyer:** Points to the "Nettoyer" button.
Code:

```
<input type="reset" value="Nettoyer" />
```

input type="submit"
se charge d'envoyer les données du formulaire

Rappel : Formulaires HTML

```
<form name="formClient" action="coursPHP-7.php" method="POST" >
```

```
    <label for="nom">Nom</label>
```

```
    <input type="text" id="nom" name="nomClient"
```

```
        value="votre nom" size="40" maxlength="150" /> <br/>
```

```
    <label>Produit</label>
```

```
    <select name="produit"
```

```
        <option value="SuperKdo"
```

```
        <option value="MegaTruc">Mega Truc</option>
```

```
        <option value="BabyFun">Baby Fun</option>
```

```
    </select> <br/>
```

```
    <label>Opinion</label>
```

```
    <textarea name="opinionClient" cols="40" rows="5" >
```

```
Votre opinion sur nos produits </textarea> <br/>
```

```
    <input type="submit" value="Envoyer" class="bouton" />
```

```
    <input type="reset" value="Nettoyer" class="bouton" />
```

```
</form>
```

À qui les données sont envoyées

input type="text"
Zone de saisie

select ... option
Liste de sélection d'options

textarea
Zone de texte

input type="submit"
Input type="reset"
Boutons d'envoi et de reset du formulaire

Communications Client - Serveur

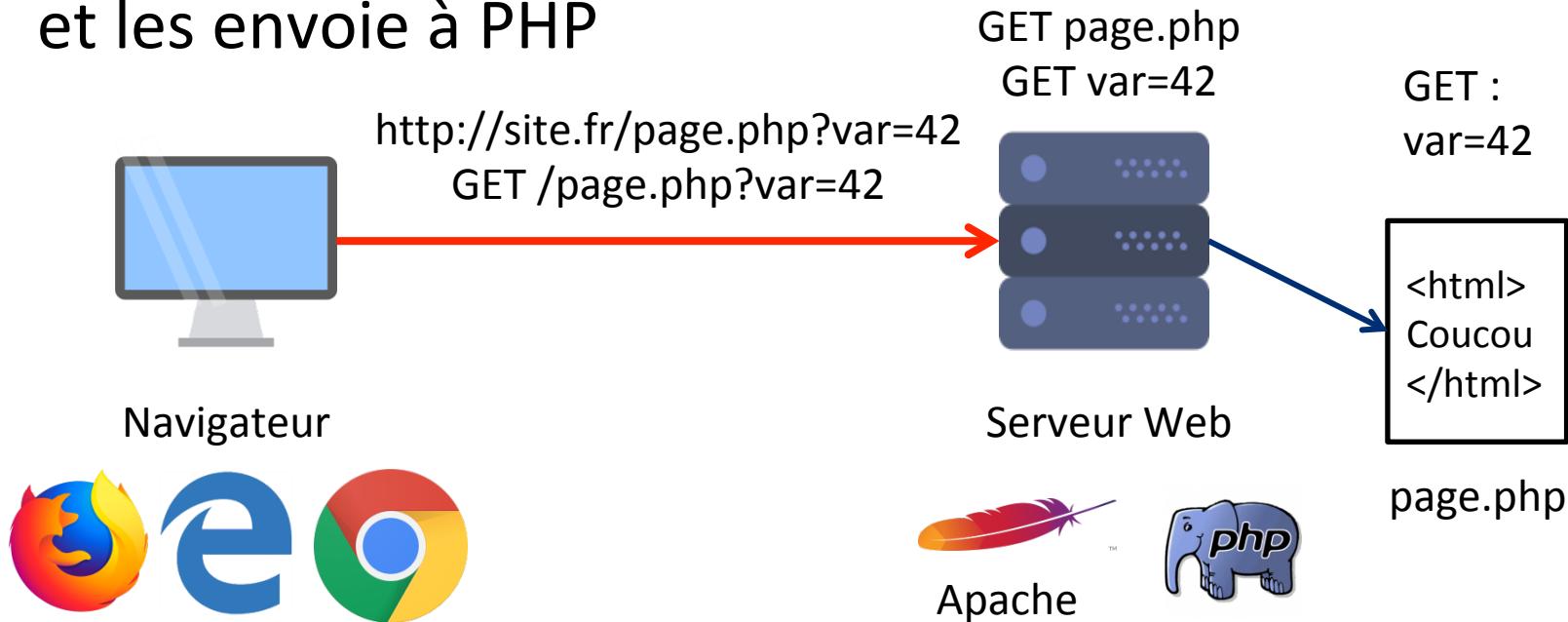
- 2 méthodes génériques d'envoi de données :
 - **GET** : (Query String)
Valeurs passées via l'URL par le navigateur
[valeurs lisibles dans l'en-tête et dans les logs d'accès]
 - <http://monsite.fr/mapage.php?var=truc&nom=moi>
 - **POST** :
On remplit un formulaire, et le navigateur envoie le formulaire dans le « body » de la requête HTTP
[valeurs lisibles si le « body » de la requête est accessible]
 - <http://monsite.fr/mapage.php>

Communications Client - Serveur

- Les données recueillies dans le formulaire sont transmises au programme indiqué dans **action=...**
- Dans **PHP**, on récupère ces données grâce à deux **tableaux associatifs** spéciaux
 - **\$_GET** → *<form action="..." method="get">*
 - **\$_GET["nom"]** *<input ... name="nom" />*
 - **\$_POST** → *<form action="..." method="post">*
 - **\$_POST["nom"]** *<input ... name="nom" />*

Communication C/S : GET

1. Client envoie des données OU clique sur un lien prévu
2. Apache récupère et analyse la requête
3. Apache crée des tableaux contenant les valeurs et les envoie à PHP



Communications C/S : GET

- URL de la requête contient les valeurs
 - Formulaire passé par l'URL
 - Construction de sa propre URL :

Lien

page.php la page qui traitera la requête

var contient « bla »

x contient 42

y contient 0

« ? » sépare l'adresse de ressource des données
« & » sépare chaque donnée (champs/valeur)

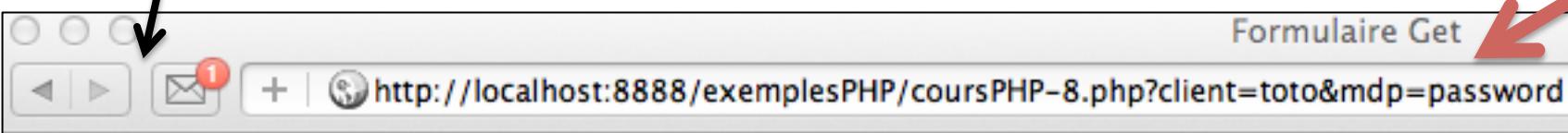
Communications C/S : GET

• Méthode GET

- Les données sont envoyées dans l'**URL** du programme
- Limitée à 256 octets
- **Déconseillé**

Nom toto
Mot de passe *****
Envoyer

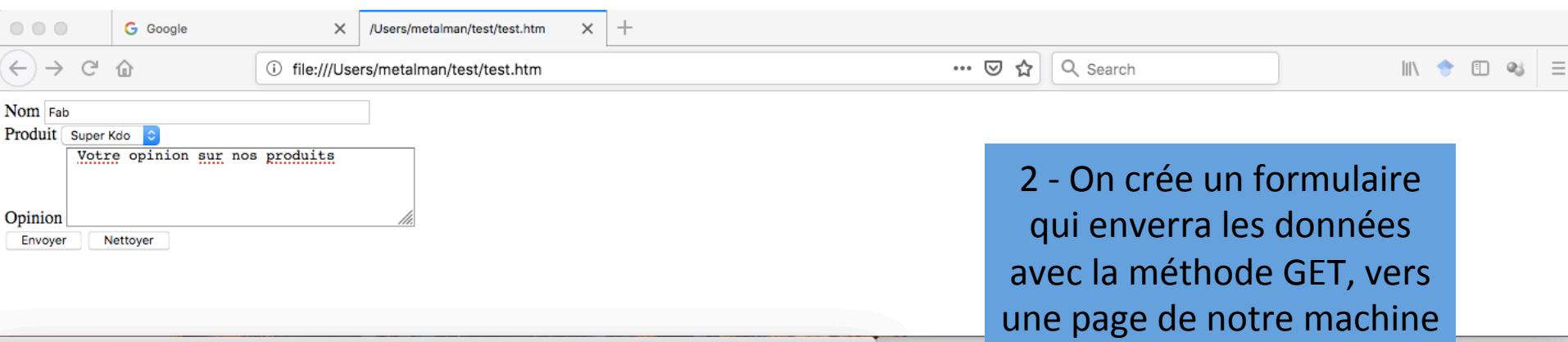
```
<form name="formGet"  
      action="coursPHP-8.php"  
      method="GET">  
  
<label>Nom</label>  
<input type="text" name="client" size="20" /> <br/>  
<label>Mot de passe </label>  
<input type="password" name="mdp" size="10"/>  
<br/>  
<input type="submit" value="Envoyer" />  
</form>
```



Données reçues :
Bienvenue, toto !

```
<?php  
echo "<p>Bienvenue, <i>". $_GET["client"] . "</i> ! </p>";  
?>
```

Communications C/S : GET



2 - On crée un formulaire qui enverra les données avec la méthode GET, vers une page de notre machine sur le port 5000

The terminal window shows the command `nc -l 5000` being run, indicating a listener is active on port 5000. The code editor displays the HTML code for the form, which is highlighted with a red box. The code uses the GET method to send data to the localhost:5000/index.php endpoint.

```
<html>
<body>

<form name="formClient"
      action="http://localhost:5000/index.php"
      method="GET" >
    <label for="nom">Nom</label>
    <input type="text" id="nom" name="nomClient"
           value="votre nom" size="40" maxlength="150" /> <br/>

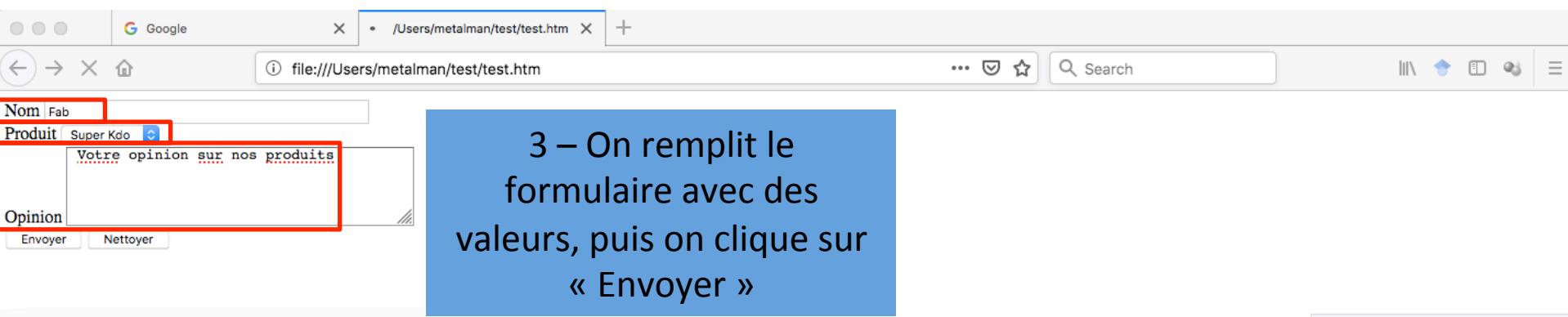
    <label>Produit</label>
    <select name="produit">
      <option value="SuperKdo">Super Kdo</option>
      <option value="MegaTruc">Mega Truc</option>
      <option value="BabyFun">Baby Fun</option>
    </select> <br/>

    <label>Opinion</label>
    <textarea name="opinionClient" cols="40" rows="5"> Votre opinion sur nos produits </textarea> <br/>

    <input type="submit" value="Envoyer" class="bouton" />
    <input type="reset" value="Nettoyer" class="bouton" />
</form>
```

1 - On lance un programme qui écoute les requêtes envoyées sur notre propre machine (localhost), sur le port 5000

Communications C/S : GET



Three terminal windows are shown. The leftmost terminal shows a user session with a red box highlighting the command `GET /index.php?nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+`. The middle terminal shows the source code of the HTML file `test.htm` with a red box highlighting the `<form>` tag. The rightmost terminal shows the response from the server, indicating it is waiting for localhost.

16:00 0 [metalman@Fabrices-MacBook-Air bash > ~
\$ nc -l 5000
GET /index.php?nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+ HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Waiting for localhost...

```
<html>
<body>

<form name="formClient"
      action="http://localhost:5000/index.php"
      method="GET" >
    <label for="nom">Nom</label>
    <input type="text" id="nom" name="nomClient"
           value="votre nom" size="40" maxlength="150" /> <br/>

    <label>Produit</label>
    <select name="produit">
      <option value="SuperKdo">Super Kdo</option>
      <option value="MegaTruc">Mega Truc</option>
      <option value="BabyFun">Baby Fun</option>
    </select> <br/>

    <label>Opinion</label>
    <textarea name="opinionClient" cols="40" rows="5"> Votre opinion sur nos produits </textarea> <br/>

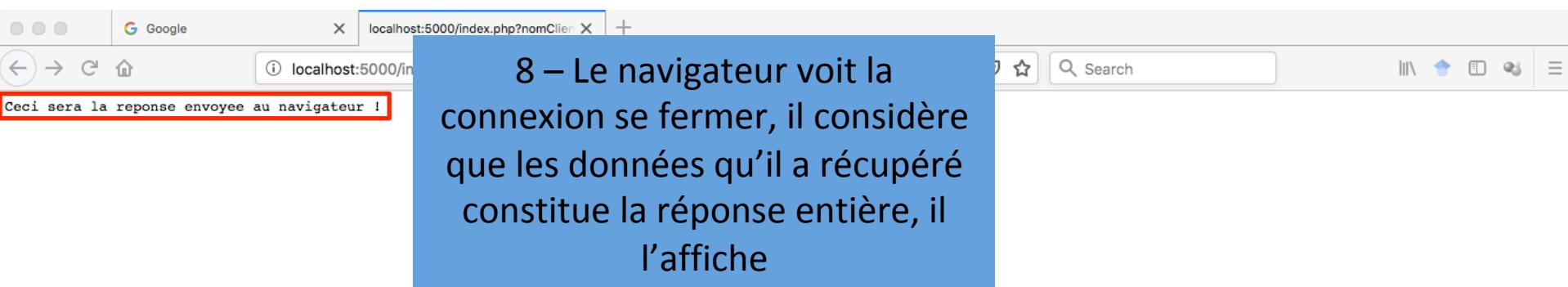
    <input type="submit" value="Envoyer" class="bouton" />
    <input type="reset" value="Nettoyer" class="bouton" />
</form>
```

Communications C/S : GET

The slide illustrates the Client/Server communication process for a GET request, divided into four main stages:

- 5 – Depuis le programme « serveur » on écrit une réponse**: Shows a terminal session where a server program (nc) is listening on port 5000. A client (Firefox) sends a GET request for the URL `localhost:5000/index.php?nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+`. The response message "Ceci sera la reponse envoyee au navigateur !" is highlighted.
- 6 – Le navigateur récupère une réponse, il considère donc qu'il s'agit de la ressource visée par cette URL avec ces variables et valeurs**: Shows a screenshot of a Firefox browser window. The URL bar shows the same GET request. The page content area displays the response message: "6 – Le navigateur commence à récupérer la réponse".
- 6 – Le navigateur commence à récupérer la réponse**: Another screenshot of the Firefox browser showing the same page content.
- Transferring data from localhost...**: Shows an Emacs editor window displaying the HTML source code of the page being loaded. The code includes form fields for "Nom", "Produit" (with options "Super Kdo", "Mega Truc", "Baby Fun"), and "Opinion" (a text area), along with submit and reset buttons.

Communications C/S : GET



Two terminal windows are shown. The left window, titled 'metalman — bash — ttys001 — 79x26', shows a netcat listener on port 5000 receiving a GET request from 'localhost:5000'. The right window, titled 'metalman — emacs test.htm — ttys002 — 63x26', shows the HTML code for a form named 'formClient'.

Left Terminal (Client Side):

```
16:00 0 [ metalman@Fabrices-MacBook-Air bash > ~
$ nc -l 5000
GET /index.php?nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+
nos+produits+ HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 Firefox/64.0
Accept: text/html
Accept-Language: fr
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Ceci sera la reponse envoyee au navigateur !
^D
```

Right Terminal (Server Side):

```
<html>
<body>

<form name="formClient"
      action="http://localhost:5000/index.php"
      method="GET" >
  <label for="nom">Nom</label>
  <input type="text" id="nom" name="nomClient"
         value="votre nom" size="40" maxlength="150" /> <br/>

  <label>Produit</label>
  <select name="produit">
    <option value="SuperKdo">Super Kdo</option>
    <option value="MegaTruc">Mega Truc</option>
    <option value="BabyFun">Baby Fun</option>
  </select> <br/>

  <label>Opinion</label>
  <textarea name="opinionClient" cols="40" rows="5"> Votre opi-
  nion sur nos produits </textarea> <br/>

  <input type="submit" value="Envoyer" class="bouton" />
  <input type="reset" value="Nettoyer" class="bouton" />
</form>
```

Communications C/S : GET (logs)

Nom

Produit

Votre opinion sur nos produits

Opinion

1 – On va regarder les « logs » de apache (les journaux enregistrant toutes les transactions que le serveur web a traité)

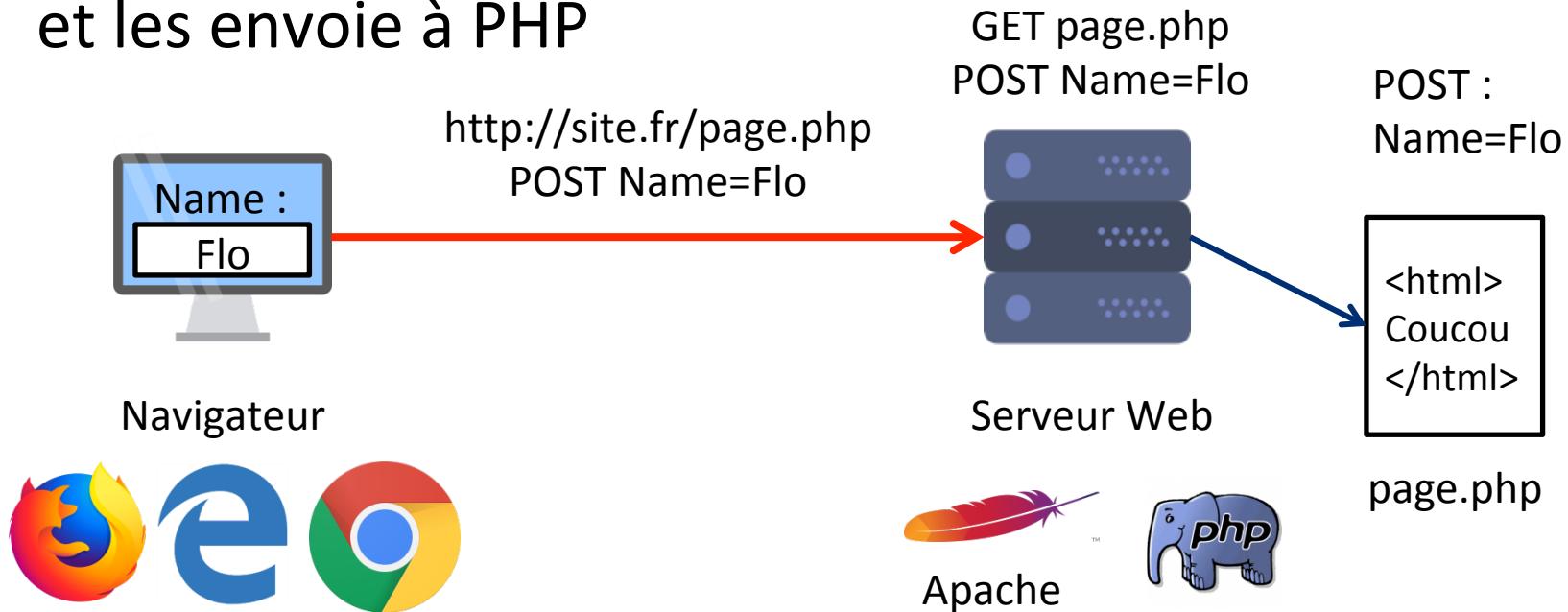
2 – Le navigateur demande la page contenant le formulaire au serveur

```
Tools Help
2019:23:03:07 +0100] "GET /favicon.ico HTTP/1.1" 404 5\
vs NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gec\
19:23:03:37 +0100] "--" 408 0 "--" "--"
86.245.42.53 - - [31/Jan/2019:23:03:58 +0100] "--" 408 0 "--" "--"
86.245.42.53 - - [31/Jan/2019:23:04:33 +0100] "GET /form.php HTTP/1.1" 200 617 \
"--" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
86.245.42.53 - - [31/Jan/2019:23:04:33 +0100] "GET /favicon.ico HTTP/1.1" 404 5\
13 "--" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gec\
kO"
86.245.42.53 - - [31/Jan/2019:23:04:37 +0100] "GET /action.php?nomClient=Fab&pr\
oduit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+ HTTP/1.1" 200 202\
"HTTP://p1web2019.metalman.eu/form.php" "Mozilla/5.0 (Windows NT 6.1; WOW64; T\
rident/7.0; rv:11.0) like Gecko"
```

3 – Le navigateur envoie le formulaire par l'URL... on voit toutes les valeurs dans la requête GET enregistrée

Communication C/S : POST

1. Client rempli formulaire puis déclenche une requête
2. Apache récupère et analyse la requête
3. Apache crée des tableaux contenant les valeurs et les envoie à PHP



Communication C/S : POST

Formulaire

localhost:8888/exemplesPHP/formClient.html

Nom	<input type="text" value="Manuele"/>
Produit	<input type="text" value="Baby Fun"/> <input type="button" value="Nettoyer"/>
Opinion	<input type="text" value="bla bla bla bla bla"/>
<input type="button" value="Envoyer"/>	

```
<form name="formClient" action="coursPHP-7.php"
      method="POST">
<label for="nom">Nom</label>
<input type="text" id="nom"
       name="nomClient" value="votre nom"
       size="40" maxlength="150" /> <br/>
```

Formulaire Client

localhost:8888/exemplesPHP/formClient.php

Données enquête

Merci de votre participation, Manuele !

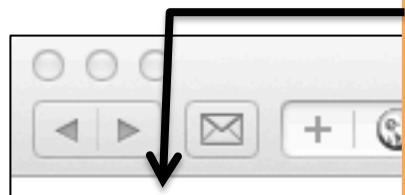
Votre produit est : BabyFun

Votre opinion est : bla bla bla bla bla

```
...
<?php
$nom = $_POST["nomClient"];
$op = $_POST["opinionClient"];
$prod = $_POST["produit"];
echo "<p>Merci de votre participation, $nom ! </p>";
echo "<p>Votre produit est : <i> $prod </i> </p>";
echo "<p> Votre opinion est : <i> $op </i> </p>";
?>
```

Communication C/S : POST - HTML

- Exemple



```
<form name="..." action="coursPHP-9.php" method="POST">
<fieldset>
<legend>Vos données </legend>
<label>...</label> <input type="text" name="nom" ... /><br/>
<label>...</label> <input type="email" name="email" ... /><br/>
<input type="radio" name="sexe" value="Homme" />Homme
<input type="radio" name="sexe" value="Femme" /> Femme<br/>
</fieldset>
```

Nom : Manuele
Email : mkirschpin@univ-paris1.fr
Homme Femme

Vos produits

Produit préféré :

Mega Truc

bla bla bla bla

Suggestions :

Envoyer

Nettoyer

```
<fieldset> <legend>Vos produits </legend>
<label>...</label>
<select name="produit">
<option value="SuperKdo">...</option>
<option value="MegaTruc"> Mega Truc</option>
<option value="BabyFun"> ... </option>
</select> <br/>
<label>...</label>
<textarea name="opinion" ... > ... </textarea>
</fieldset>
```

Communication C/S : POST - PHP

- Exemple

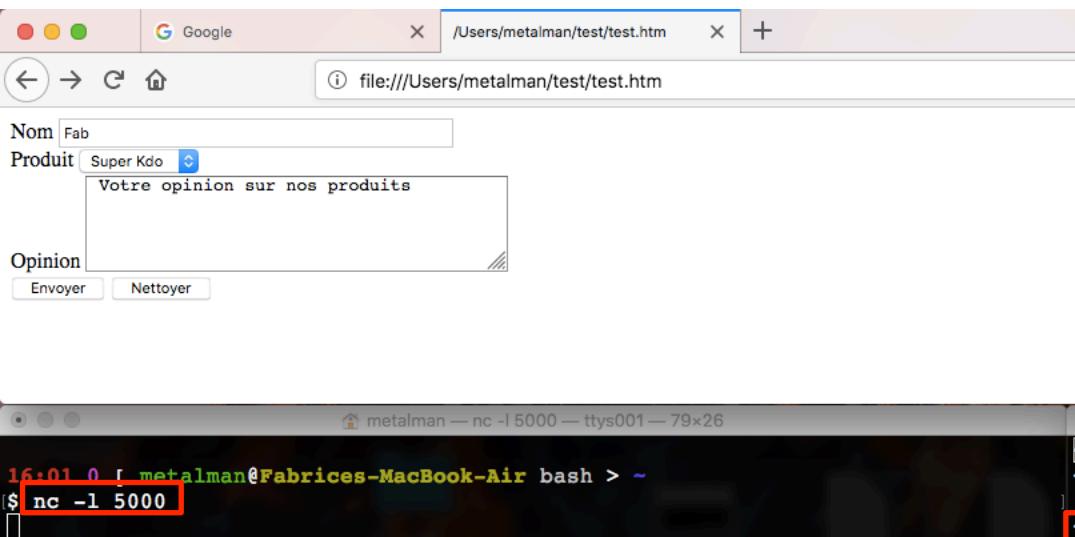


Récapitulatif

- *Nom : Manuele*
- *Email : mkirschpin@univ-paris1.fr*
- *Sexe : Femme*
- *Produit préféré : MegaTruc*
- *Suggestion : bla bla bla bla*

```
<body>
  <h1>Récapitulatif </h1>
  <ul>
    <?php
      echo "<li> Nom : " . $_POST["nom"] . "</li>" ;
      echo "<li> Email : " . $_POST["email"] . "</li>" ;
      echo "<li> Sexe : " . $_POST["sexe"] . "</li>" ;
      echo "<li> Produit préféré : " . $_POST["produit"] . "</li>" ;
      echo "<li> Suggestion : " . $_POST["opinion"] . "</li>" ;
    ?>
  </ul>
</body>
```

Communication C/S : POST



1 - On lance un programme qui écoute les requêtes envoyées sur notre propre machine (localhost), sur le port 5000

2 - On crée un formulaire qui enverra les données avec la méthode POST, vers une page de notre machine sur le port 5000

```
metalman — nc -l 5000 — ttys001 — 79x26
16:01:01 metalman@Fabrices-MacBook-Air bash > -
$ nc -l 5000
```

```
<html>
<body>

<form name="formClient"
      action="http://localhost:5000/index.php"
      method="POST" >

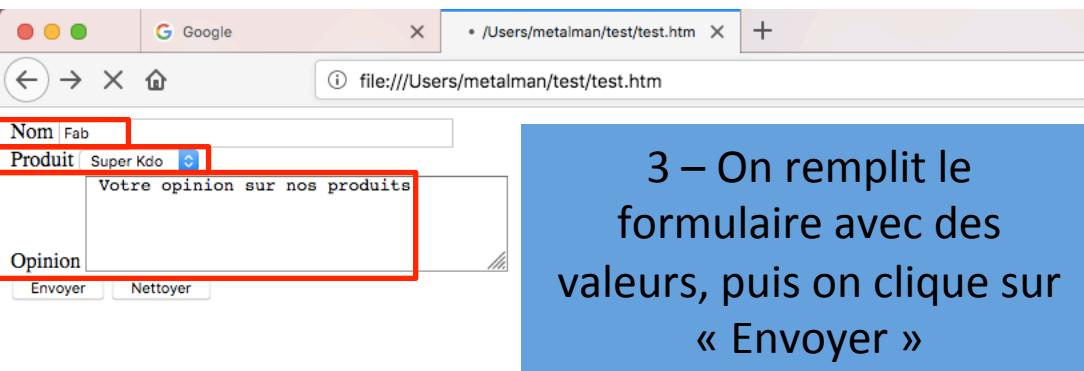
    <label for="nom">Nom</label>
    <input type="text" id="nom" name="nomClient"
           value="votre nom" size="40" maxlength="150" /> <br/>

    <label>Produit</label>
    <select name="produit">
      <option value="SuperKdo">Super Kdo</option>
      <option value="MegaTruc">Mega Truc</option>
      <option value="BabyFun">Baby Fun</option>
    </select> <br/>

    <label>Opinion</label>
    <textarea name="opinionClient" cols="40" rows="5"> Votre opinion sur nos produits </textarea> <br/>

    <input type="submit" value="Envoyer" class="bouton" />
    <input type="reset" value="Nettoyer" class="bouton" />
</form>
```

Communication C/S : POST



3 – On remplit le formulaire avec des valeurs, puis on clique sur « Envoyer »

Waiting for localhost...

```
metalman — nc -l 5000 — ttys001 — 79x26
16:01 0 [ metalman@Fabrices-MacBook-Air bash > ~
$ nc -l 5000
POST /index.php HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 77
Connection: keep-alive
Upgrade-Insecure-Requests: 1
nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+
```

metalman — emacs test.htm — ttys002 — 63x26

```
<html>
<body>
```

4 – Le navigateur envoie une requête POST au serveur (localhost) en visant la ressource /index.php, et en incluant l'ensemble des variables dans le corps (body) de la requête HTTP

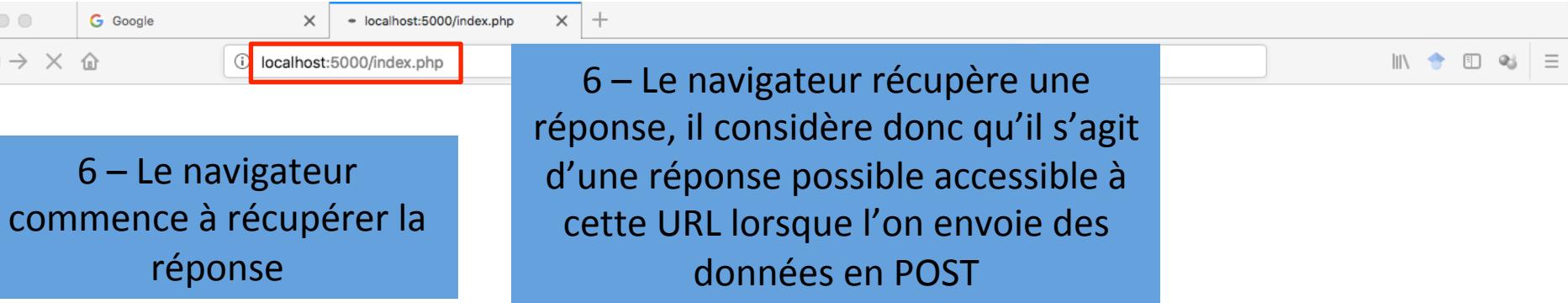
```
"> Votre opi\
```

nion sur nos produits </textarea>


```
<input type="submit" value="Envoyer" class="bouton" />
<input type="reset" value="Nettoyer" class="bouton" />
</form>
```

-uu:---F1 test.htm Top (1,0) (HTML)-----

Communication C/S : POST



Transferring data from localhost...

```
Transferring data from localhost...
metalman — nc -l 5000 — ttys001 — 79x26
16:01 0 [ metalman@Fabrices-MacBook-Air bash > -
$ nc -l 5000
POST /index.php HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 77
Connection: keep-alive
Upgrade-Insecure-Requests: 1
nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+
Une autre reponse envoyee au navigateur...
```

6 – Le navigateur récupère une réponse, il considère donc qu'il s'agit d'une réponse possible accessible à cette URL lorsque l'on envoie des données en POST

```
metalman — emacs test.htm — ttys002 — 63x26
html>
<body>
<form name="formClient"
      action="http://localhost:5000/index.php"
      method="POST" >
  <label for="nom">Nom</label>
  <input type="text" id="nom" name="nomClient"
         value="votre nom" size="40" maxlength="150" /> <br/>
  <label>Produit</label>
  <select name="produit">
    <option value="SuperKdo">Super Kdo</option>
    <option value="MegaTruc">Mega Truc</option>
    <option value="BabyFun">Baby Fun</option>
  </select> <br/>
  Opinion</label>
  <ea name="opinionClient" cols="40" rows="5"> Votre opi<br/>
  nos produits </textarea> <br/>
  <input type="submit" value="Envoyer" class="bouton" />
  <input type="reset" value="Nettoyer" class="bouton" />
uu:---F1 test.htm Top (1,0) (HTML)-----
```

5 – Depuis le programme « serveur » on écrit une réponse

Communication C/S : POST

A screenshot of a web browser window. The address bar shows "localhost:5000/index.php". The main content area displays the text "Une autre reponse envoyee au navigateur...". A blue callout box with white text is overlaid on the right side, stating: "8 – Le navigateur voit la connexion se fermer, il considère que les données qu'il a récupéré constituent la réponse entière, il l'affiche".

Two terminal windows are shown. The left window, titled "metalman — bash — ttys001 — 79x26", shows a netcat listener on port 5000 receiving a POST request for "index.php". The right window, titled "metalman — emacs test.htm — ttys002 — 63x26", shows the HTML code for a form named "formClient" with fields for "nom", "produit", and "opinionClient". The terminal also shows the user entering the form data and pressing Ctrl+D to close the connection.

7 – Depuis le programme « serveur », on termine la réponse (Ctrl + D)/ferme la connexion

```
16:01 0 [ metalman@Fabrices-MacBook-Air bash > ~
$ nc -l 5000
POST /index.php HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html
Accept-Language:
Accept-Encoding:
Content-Type: application/x-www-form-urlencoded
Content-Length: 7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
nomClient=Fab&produit=SuperKdo&opinionClient=+Votre+opinion+sur+nos+produits+
Une autre reponse envoyee au navigateur...
^D
16:02 0 [ metalman@Fabrices-MacBook-Air bash > ~
```

```
<html>
<body>

<form name="formClient"
      action="http://localhost:5000/index.php"
      method="POST" >
  <label for="nom">Nom</label>
  <input type="text" id="nom" name="nomClient"
         value="votre nom" size="40" maxlength="150" /> <br/>

  <label>Produit</label>
  <select name="produit">
    <option value="SuperKdo">Super Kdo</option>
    <option value="MegaTruc">Mega Truc</option>
    <option value="BabyFun">Baby Fun</option>
  </select> <br/>

  <label>Opinion</label>
  <textarea name="opinionClient" cols="40" rows="5"> Votre opinion sur nos produits </textarea> <br/>

  <input type="submit" value="Envoyer" class="bouton" />
  <input type="reset" value="Nettoyer" class="bouton" />
</form>
```

Communications C/S : POST (logs)

Nom

Produit

Votre opinion sur nos produits

Opinion

2 – Le navigateur demande la page contenant le formulaire au serveur

```
Tools Help
GET /form.php HTTP/1.1" 200 202 \
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
86.245.42.53 - - [31/Jan/2019:23:05:19 +0100] "GET /form.php HTTP/1.1" 200 618 \
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
86.245.42.53 - - [31/Jan/2019:23:05:24 +0100] "POST /action.php HTTP/1.1" 200 2 \
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
86.245.42.53 - - [31/Jan/2019:23:05:24 +0100] "-" 408 0 "-" "-"
```

1 – On va regarder les « logs » de apache (les journaux enregistrant toutes les transactions que le serveur web a traité)

3 – Le navigateur envoie le formulaire en POST, on ne voit aucun nom de variable ni valeur dans les logs d'apache !

Statuts HTTP

- Une requête envoyée = une réponse reçue
- Réponses HTTP de 5 types (officiels) :
 - 1xx : informationnelle
 - 2xx : succès
 - 3xx : redirection
 - 4xx : erreur côté client // 404 : url n'existe pas
 - 5xx : erreur côté serveur
 - *[autres erreurs sont liées au serveur web utilisé]*

Statuts HTTP : classiques

200	OK	La requête a bien été reçue, comprise, et exécutée
301 308	Permanent Redirect	La ressource a été définitivement déplacée à une autre URL
302 307	Temporary Redirect	La ressource est temporairement déplacée à une autre URL
403	Forbidden	La requête est correcte, mais le client n'a pas le droit de demander au serveur de faire cette action
404	Not Found	La ressource visée par l'URL n'a pas été trouvée
500	Internal Server Error	Erreur générique/non spécifique... le serveur web a échoué quelque chose à un moment
502	Bad Gateway	Le serveur web visé n'arrive pas à contacter le serveur qui traite la requête envoyée

Statuts HTTP : 200

metalman — bash — ttys001 — 80x25

```
16:55 0 [ metalman@Fabrices-MacBook-Air bash > ~/test
$ printf 'GET / HTTP/1.1\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
HTTP/1.1 200 OK
Date: Thu, 31 Jan 2019 15:55:20 GMT
Server: Apache/2.2.34 (Unix) mod_wsgi/3.5 Python/2.7.13 PHP/7.2.10 mod_ssl/2.2.3
4 OpenSSL/1.0.2o DAV/2 mod_fastcgi/mod_fastcgi-SNAP-0910052141 mod_perl/2.0.9 Pe
rl/v5.24.0
Content-Length: 247
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
    <title>Index of /</title>
  </head>
  <body>
<h1>Index of /</h1>
<ul><li><a href="test.php"> test.php</a></li>
<li><a href="test.php~"> test.php~</a></li>
</ul>
</body></html>

16:55 0 [ metalman@Fabrices-MacBook-Air bash > ~/test
$ ]
```

Statuts HTTP

1 – On envoie une requête au serveur « localhost » sur le port 8888, en demandant la ressource ‘/’

```
16:55 0 [ metalman@Fabrices-MacBook-Air bash > ~/test  
$ printf 'GET / HTTP/1.1\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
```

HTTP/1.1 200 OK

Date: Thu, 31 Jan
Server: Apache/2.
4 OpenSSL/1.0.2o
rl/v5.24.0

2 – Apache répond dans l'en-tête de la réponse HTTP que tout va bien (OK) : code 200

Content-Length: 247
Content-Type: text/html; charset=ISO-8859-1

thon/2.7.13 PHP/7.2.10 mod_ssl/2.2.3
gi-SNAP-0910052141 mod_perl/2.0.9 Pe

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">  
<html>  
  <head>  
    <title>Index of /</title>  
  </head>  
  <body>  
    <h1>Index of /</h1>  
    <ul><li><a href="test.php"> test.php</a></li>  
    <li><a href="test.php~"> test.php~</a></li>  
    </ul>  
  </body></html>
```

2 – Apache envoie la page web demandée dans le corps (body) de la réponse

```
16:55 0 [ metalman@Fabrices-MacBook-Air bash > ~/test  
$ ]
```

Statuts HTTP : 404

```
metalman@Fabrices-MacBook-Air ~ % ./test
$ printf 'GET /inexistant.php HTTP/1.1\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
HTTP/1.1 404 Not Found
Date: Thu, 31 Jan 2019 15:57:10 GMT
Server: Apache/2.2.34 (Unix) mod_wsgi/3.5 Python/2.7.13 PHP/7.2.10 mod_ssl/2.2.34 OpenSSL/1.0.2o DAV/2 mod_fastcgi/mod_fastcgi-SNAP-0910052141 mod_perl/2.0.9 Perl/v5.24.0
Content-Length: 212
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /inexistant.php was not found on this server.</p>
</body></html>

metalman@Fabrices-MacBook-Air ~ %
```

Statuts HTTP

1 – On envoie une requête au serveur « localhost » sur le port 8888, en demandant une ressource qui n'existe pas

```
16:57 0 [ metalman@Fabrices-MacBook-Air bash > ~/test
$ printf 'GET /inexistant.php HTTP/1.1\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
HTTP/1.1 404 Not Found
Date: Thu, 31 Jan 2019 1
Server: Apache/2.2.34 (U
0.2o DAV/2 mod_fastcgi/m
Content-Length: 212
Content-Type: text/html;
```

2 – Apache répond dans l'en-tête de la réponse HTTP que la ressource n'a pas été trouvée : code 404

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /inexistant.php was not found on this server.</p>
</body></html>
```

2 – Apache envoie dans le corps de la réponse HTTP une page web détaillant l'erreur au client (au format HTML pour que le client puisse la lire)

Statuts HTTP : 400

```
metalman@Fabrices-MacBook-Air ~ % ./test
$ printf 'GET / HTTP/1.1\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
HTTP/1.1 400 Bad Request
Date: Thu, 31 Jan 2019 15:57:53 GMT
Server: Apache/2.2.34 (Unix) mod_wsgi/3.5 Python/2.7.13 PHP/7.2.10 mod_ssl/2.2.34 OpenSSL/1.0.2o DAV/2 mod_fastcgi/mod_fastcgi-SNAP-0910052141 mod_perl/2.0.9 Perl/v5.24.0
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>

metalman@Fabrices-MacBook-Air ~ %
```

Statuts HTTP

1 – On envoie une requête incorrecte (protocole inexistant) au serveur « localhost » sur le port 8888

```
16:57 0 [ metalman@Fabrices-MacBook-Air bash > ~/test  
$ printf 'GET / HTTP/42\r\nHost: localhost:8888\r\n\r\n' | nc localhost 8888
```

HTTP/1.1 400 Bad Request

Date: Thu, 31 Jan 2019 15:
Server: Apache/2.2.34 (Ubuntu)
0.20 DAV/2 mod_fastcgi/mod
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

2 – Apache répond dans l'en-tête de la réponse HTTP que la requête est incorrecte (elle ne respecte pas les standards qu'il connaît)

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>400 Bad Request</title>  
</head><body>  
<h1>Bad Request</h1>  
<p>Your browser sent a request that this server could not understand.<br />  
</p>  
</body></html>
```

2 – Apache envoie dans le corps de la réponse HTTP une page web détaillant l'erreur au client (au format HTML pour que le client puisse la lire)

PHP : Contrôle de Flot

- **Instructions de contrôle**
 - Instructions pour gérer le flot d'exécution
 - **Instructions conditionnelles**
 - Elles conditionnent l'exécution
 - Semblables à un **nœud de Décision** (diagramme activités)
 - *if... else ..., switch ... case ...*
 - **Instructions de boucle**
 - Elles permettent la **répétition** d'un bloc d'instructions
 - *for ... , foreach ... , while ... , do... while*

PHP : if ... else ...

- Instructions conditionnelles if ... else...

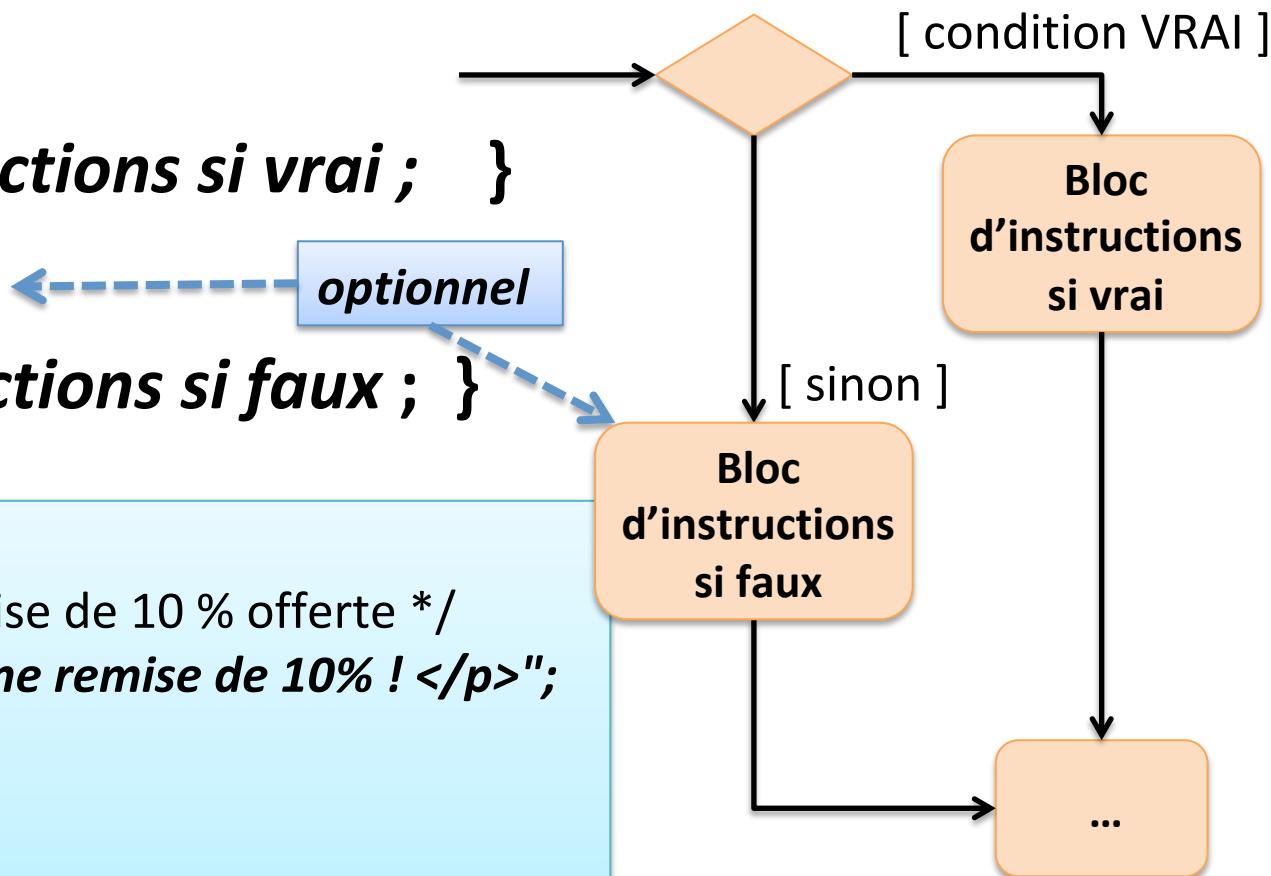
if (*condition*)

{ *bloc d'instructions si vrai ;* }

else

{ *bloc d'instructions si faux ;* }

```
if ( $qte >= 100 )
{
    $remise = 0.10; /* remise de 10 % offerte */
    echo "<p>Vous avez une remise de 10% ! </p>";
}
else {
    $remise = 0.05;
    echo "<p>Vous avez une remise de 5% </p>";
}
```



PHP : if ... else ...

- **Instructions conditionnelles if ... else...**

- Les données pour la condition peuvent venir d'un formulaire

formExemple11.html

```
<form name="..." method="POST"
      action="coursPHP-11.php" >
...
<select name="prix">
    <option value="10">
        Super Kdo - 10€ </option>
    ...
</select>
...
<input type="number" size="10"
       name="qte" />
...
<input type="submit" value="Devis" />
</form>
```

coursPHP-11.php

```
<?php
$qte = $_POST["qte"];
$prixunit = $_POST["prix"];
$remise = 0;

if ( $qte >= 100)
{
    $remise = 0.10; /* remise de 10 % offerte */
    echo "<p>Vous avez une remise de 10% ! </p>";
}

$prix = $prixunit * $qte
      - ($prixunit * $qte * $remise);
echo "<p> Pour un prix de <i> $prixunit </i>
l'unité et <i> $qte </i> unités, vous avez à
régler <i> $prix </i></p>";

?>
```

PHP : if ... else ...

Préparer votre devis

Produit : Super Kdo - 10€

Quantité : 100

Devis

```
<form name="..." method="POST"
      action="coursPHP-11.php" >
<label>Produit : </label>
<select name="prix">
  <option value="10" >Super ... </option>
  ...
</select>  <br/>
<label>Quantité : </label>
<input name="qte" type="number"
       size="10" /> <br/>
<input type="submit" value="Devis" />
</form>
```

Devis

Vous avez droit à une remise de 10% !

Pour un prix de 10 l'unité et 100 unités, vous avez à régler 900

```
<?php
$qte = $_POST["qte"];
$prixunit = $_POST["prix"];
$remise = 0;

...
if ( $qte >= 100)
{
  $remise = 0.10;
  echo "<p>Vous avez .... </p>";
}
... ?>
```

PHP : if ... elseif ... else ...

- **Instructions conditionnelles if ... else ...**
 - Les blocs if ... else ... peuvent contenir n'importe quelle instruction, y compris d'autres blocs if ... else ...

```
if ( condition1 )  
{   bloc d'instructions si condition1 vraie ;   }  
elseif (condition2)  
{   bloc d'instructions si condition2 vraie ;   }  
else  
{   bloc d'instructions si les conditions sont fausses ;   }
```

Préparer votre devis

Produit : Super Kdo - 10€

Quantité : 100

Devis

```
<form name="..." method="POST"
      action="coursPHP-12.php" >
...
<select name="prix"> ... </select>
...
<input type="number" ... name="qte"/>
...
<input type="submit" value="Devis" />
</form>
```

Devis

Prix unitaire : 10 , Quantité : 100 , Remise : 10 %

Total à régler : 900

```
<?php
$qte = $_POST["qte"];
$prixunit = $_POST["prix"];

if ( $qte >= 100 )
    { $remise = 0.10 ; }
elseif ( $qte >= 50 )
    { $remise = 0.05 ; }
else
    { $remise = 0 ; }

$prix = $prixunit * $qte
      - ($prixunit * $qte * $remise);

echo "<p> Prix unitaire : <i> $prixunit </i>,
      Quantité : <i> $qte </i>,
      Remise : <i>" . $remise*100 . "</i> % </p>";

echo "<p><i>Total à régler : </i>
      <b> $prix </b></p>";

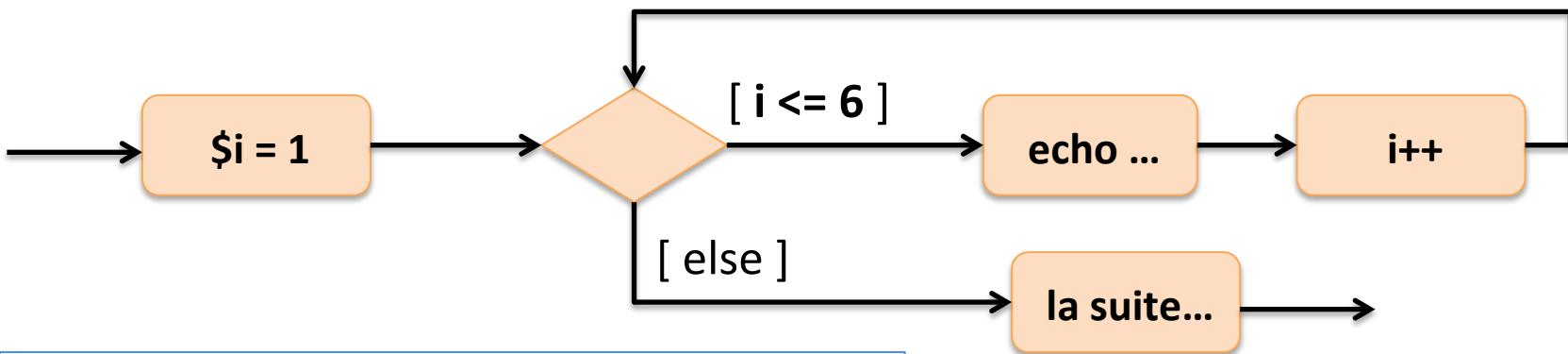
?>
```

PHP : for ...

- **Instructions de boucle : for**

- La boucle **for** permet de **répéter** (un certain nombre de fois) l'exécution d'un bloc d'instructions

for (*initialisation* ; *condition* ; *incrémantation*)
{ *bloc d'instructions à répéter* ; }



```
for ( $i = 1 ; $i <= 6 ; $i++ )
{
    echo "<h{$i}> Titre niveau $i </h{$i}>";
}
```

$\$i++ \rightarrow \$i = \$i + 1$

PHP : for ...

- Instructions de boucle : for

```
<?php  
for ( $i = 1 ; $i <= 6 ; $i++)  
{  
    echo "<h$i> Titre niveau $i </h$i>";  
}  
?>
```

Titre niveau 1

Titre niveau 2

Titre niveau 3

Titre niveau 4

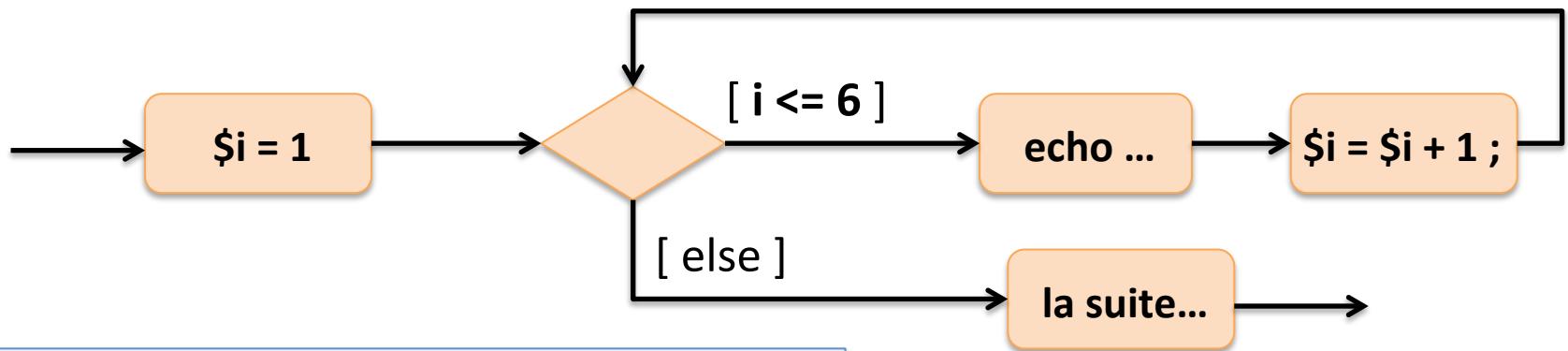
Titre niveau 5

Titre niveau 6

PHP : while ...

- **Instructions de boucle : while**

- La boucle **while** permet de continuer à réaliser un bloc d'opérations **tant qu'une condition soit vraie**



```
$i = 1;  
while ( $i <= 6 ) {  
    echo "<h$i> Titre niveau $i </h$i>";  
    $i = $i + 1;  
}
```

PHP : while ...

- Instructions de boucle : **while**

```
<?php
```

```
    $i = 1;
```

```
    while ( $i <= 6 ) {
```

```
        echo "<h$i> Titre niveau $i </h$i>";
```

```
        $i = $i + 1;
```

```
}
```

```
?>
```

On donne une **valeur initiale** à la variable **\$i**

Tant que **\$i** ne dépasse pas la valeur 6

On met à jour la **valeur de la variable \$i**

Titre niveau 1

Titre niveau 2

Titre niveau 3

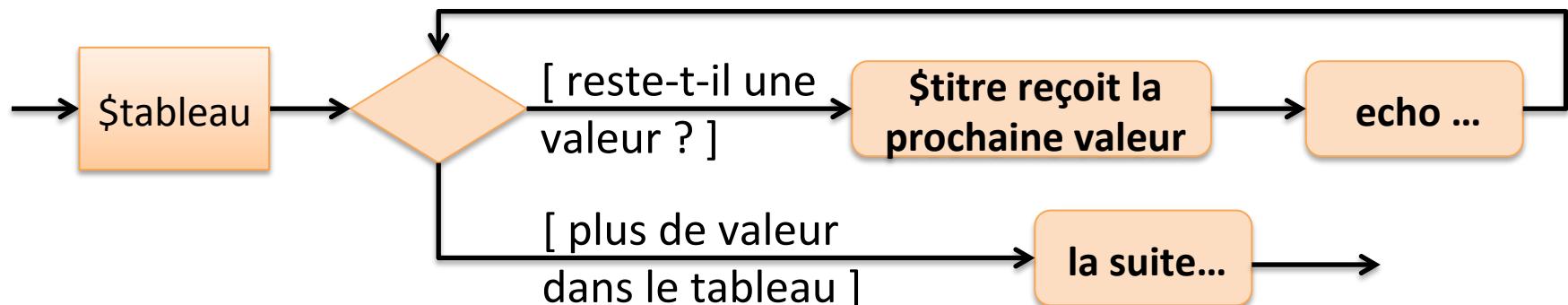
Titre niveau 4

Titre niveau 5

Titre niveau 6

PHP : foreach ...

- **Instructions de boucle : foreach**
 - La boucle **foreach** permet de **répéter** un bloc d'instructions pour **chaque valeur dans un tableau**



```
foreach ($tableau as $titre ) {  
    echo "<$titre> Titre $titre  
          </$titre>";  
}
```

PHP : foreach ...

- Instructions de boucle : foreach

```
<?php  
    On définit un tableau  
    $tableau = array("h1", "h2", "h3",  
                    "h4", "h5", "h6");  
foreach ($tableau as $titre ) {  
    echo "<$titre> Titre $titre  
          </$titre>";  
}  
?>
```

Pour chaque valeur
dans le tableau

Titre h1

Titre h2

Titre h3

Titre h4

Titre h5

Titre h6

PHP : foreach ...

- Instructions de boucle : **foreach**
 - ça fonctionne aussi pour les tableaux associatifs

```
<?php  
$tableau = array ("nom" => "Dupont" ,  
                  "prenom" => "Jean" ,  
                  "adresse" => "qq part à Paris" ) ;  
  
foreach ($tableau as $cle=>$valeur) {  
    echo "<li> $cle : $valeur </li>" ;  
}  
?>
```

On définit un tableau associatif : clé => valeur

Pour chaque pair
\$clé => \$valeur
dans \$tableau

- *nom : Dupont*
- *prenom : Jean*
- *adresse : qq part à Paris*

PHP : boucles

- **Instructions de boucle : boucles imbriquées**
 - Il est possible d'imbriquer des boucles les unes dans les autres

```
<table>
<?php
    for ( $lin = 1 ; $lin <= 9 ; $lin++) {
        echo "<tr> ";
        for ( $col = 1 ; $col <= 9 ; $col++) {
            echo "<td> "
                . ($col * $lin) . "</td> ";
        }
        echo "</tr>";
    }
?>
</table>
```

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Evaluation des Etudiants

Date	Type	Description	Binôme
S05 (19/02)	[DM] Devoir Maison	1 mini projet à rendre par mail (25/02 – 23h42)	OUI
S10 (02/04)	[DST] Devoir sur Table	1 examen écrit sans document ni machine	NON
S12 (16/04)	[PROJ] Projet	Présentation du projet	OUI
mai	Partiel	1 examen écrit sans document ni machine	NON

50%

50%

PHP : Fonctions

- **Fonctions**
 - PHP offre une large panoplie de fonctions
 - Exemple : **isset(\$var)** → TRUE si \$var a été déclarée
 - Exemple : **empty(\$var)** → TRUE si \$var est vide (ou vaut 0)
 - On peut aussi écrire les nôtres
(même en dehors des classes)
 - **function nomFonction (\$paramètre , ...) { instructions }**

```
function salutation ( $nom ) {  
    echo "<h1>Bienvenue, $nom ! </h1>";  
    echo "<p class=droite>Aujourd'hui, nous sommes le " .date('d / m / Y'). "</p>";  
}
```

```
...  
<form name="..." method="POST"  
      action="coursPHP-15.php" >  
<label>Nom : </label>  
<input type="text" name="client"  
      size="25"/>  
...  
<input type="submit" value="OK" />  
</form>
```

```
<?php  
function salutation ( $nom ) {  
    date_default_timezone_set("Europe/Paris");  
    echo "<h1>Bienvenue, $nom ! </h1>";  
    echo "<p class=droite>Aujourd'hui..."  
        . date('d / m / Y'). "</p>" ;  
}  
  
if ( isset ($_POST["client"]) AND  
    ! empty ($_POST["client"]) ) {  
    salutation ( $_POST["client"] ) ;  
}  
else { salutation ("cher client") ; }  
?>
```

Identification

Nom :

Mot de passe :

Nom :

Mot de passe :

Bienvenue, manuele !

Aujourd'hui, nous sommes le 04 / 03 / 2012

Bienvenue, cher client !

Aujourd'hui, nous sommes le 04 / 03 / 2012

Visibilité des variables

En PHP il existe 3 niveaux de visibilité d'une variables selon le contexte :

- **Les variables superglobales** : elles sont disponibles n'importe où dans le programme.
- **Les variables globales** : ce sont toutes les variables, tableaux, objets et constantes que nous créons nous-même dans le programme principal. Elles ne sont généralement visibles que dans le programme principal.
- **Les variables locales** : ce sont toutes les variables d'une fonction (paramètres compris). Leur visibilité n'est que locale, et le programme principal ne peut pas agir sur ces variables.

Portée des variables globales

- En PHP, une variable globale peut être utilisée à l'intérieur d'une fonction sans la passer en paramètre
 - Usage du mot clé « global »
 - La variable doit être déclarée avec « global » dans chaque fonction où cela est nécessaire

Portée des variables globales

```
<?php  
$a = 1;  
$b = 2;  
  
function somme()  
{  
    global $a, $b;  
    $b = $a + $b;  
    echo « ». $b;  
}  
  
somme();  
somme();  
?>
```

On a déclaré \$a et \$b avec le mot clé « global » à l'intérieur de somme pour qu'elles soient manipulées par celle-ci.

Portée des variables globales

- Une autre méthode pour accéder aux variables globales est d'utiliser la superglobale `$GLOBALS` :

```
<?php
$a = 1;
$b = 2;

function somme()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
    echo $GLOBALS['b'];
}
?>
```

Les variables superglobales

- Les variables superglobales sont générées automatiquement par PHP :
 - elles sont disponibles quel que soit le contexte du script (*même à l'intérieur d'une fonction sans être passées en paramètre de celle-ci*)
 - elles sont écrites en majuscules et commencent, par un underscore « `_` »
 - elles sont généralement des tableaux associatifs

Les variables superglobales

- **`$GLOBALS`** : rassemble les variables globales.
- **`$_ENV`** : ce sont des variables d'environnement toujours données par le serveur.
- **`$_SESSION`** : se sont des variables de session. Ces variables restent stockées sur le serveur le temps de la présence d'un visiteur.
- **`$_COOKIE`** : contient les valeurs des cookies enregistrés sur l'ordinateur du visiteur.
- **`$_GET`** : contient les données envoyées en paramètres dans l'URL.
- **`$_POST`** : contient les informations qui viennent d'être envoyées par un formulaire.

Les variables superglobales

```
<html><head>
<title>La variable $_SERVER</title>
</head><body>
<?php
echo 'Nom du fichier en cours d\'execution ' .
      'a partir de la racine : ' . $_SERVER['PHP_SELF'];
echo '<br/>';
echo 'Nom de la racine du script : ' . $_SERVER['DOCUMENT_ROOT'] ;
echo '<br/>';
echo 'Nom du client HTML : ' . $_SERVER['HTTP_USER_AGENT'] . '<br>';
echo 'Nom du serveur qui execute le script : ' . $_SERVER['SERVER_NAME'] ;
echo '<br/><br/>';
echo 'Votre adresse IP est : ' . $_SERVER['REMOTE_ADDR'] ;
?>
</body></html>
```

Fonctions avancées

- Fonctions avancées PHP (serveur)
 - ===, isset, empty, is_null
 - include, require, include_once, require_once
 - ...
- Fonctions avancées HTTP (client)
 - header()

PHP : Fonctions avancées

```
<?php  
$var1 = 42;  
$var2 = "42";  
$var3 = "42 ";  
$var4 = " 42";  
  
echo(( $var1 == $var2 )."#");  
echo(( $var1 == $var3 )."#");  
echo(( $var1 == $var4 )."#");  
echo(( $var1 === $var2 )."#");  
echo(( $var1 === $var3 )."#");  
echo(( $var2 == $var3 )."#");  
?>
```

PHP faiblement typé :

== compare les valeurs

==== compare valeurs ET types

TRUE == 1

FALSE == 0

1# 1# 1# # # #

PHP : Fonctions avancées

- **Importation des fichiers**

- Incorporer le contenu d'un fichier dans une page PHP
- But : réutilisation des fichiers, uniformisation du site

- **include "fichier"** et **include_once "fichier"**

- **include** remplace la ligne par le contenu du fichier
- **include_once** fait ça **une seule fois** (même dans une boucle)

- **require "fichier"** et **require_once "fichier"**

- idem **include**, mais si le fichier n'existe pas, on a une **erreur**

PHP : Fonctions avancées

```
<meta charset="UTF-8" />
...
<title>Mon site</title>
<link rel="stylesheet"
      href="css/blocs.css" />
```

```
<header>
  <h1>Mon site</h1>
</header>
<nav>
  <h2>Exemples </h2>
  <ul>
    <li>...</li>
  ...
  </ul>
</nav>
```

```
<head> <?php
      include_once "head.html";
      require "mesfonctions.php" ; ?>
```

```
</head> <body>
```

```
<?php include_once "headerNav.html"; ?>
```

```
...
```

```
<?php
```

```
  salutation ("cher client") ;
```

```
?>
```

```
<article>
```

```
  <h2> News </h2>
```

```
  <p> ... </p>
```

```
</article>
```

```
...
```

```
<?php
```

```
  function salutation ( $nom ) {
```

```
    echo "<p class=droite><b>Bienvenue,  
          $nom ! </b></p>";
```

```
    echo "<p class=droite>Aujourd'hui,  
          nous sommes le "  
      .date('d / m / Y'). " </p>" ;
```

```
}
```

```
?>
```

include_once "head.html"

```
<head> ...
<title>Mon site</title>
<link rel="stylesheet"
      href="css/blocs.css" />
```

```
</head>
<body>
```

```
<header> <h1>Mon site</h1> </header>
```

```
<nav>
```

```
<h2>Exemples </h2>
```

include_once "headerNav.html";

```
<ul>
```

```
  <li>...</li>
```

```
...
```

```
  </ul>
```

```
</nav>
```

```
<section>
```

```
<p class=droite><b>Bienvenue, cher client ! </b></p><p
class=droite>Aujourd'hui, nous sommes le 22/ 03 / 2014 </p>
```

```
...
```

Mon site

Bienvenue, cher client !

Aujourd'hui, nous sommes le 22 / 03 / 2014

News

bla blabla bla blabla bla blabla bla blablabla bla bl
blablabla bla blabla bla blabla bla blabla bla blabla bla bl
bla blabla bla blabla bla blabla bla blabla bla blabla bla bl

```
require "mesfonctions.php" ;
salutation ("cher client") ;
```

PHP : Fonctions avancées

- **gettype(\$var)** : retourne le type de la variable
- **addslashes()** : ajoute des antislashes devant les caractères spéciaux

```
$res = addslashes("L'a"); // retourne L\'a.
```

- **strstr(texte, chaîneAchercher)** : trouve la première occurrence dans une chaîne

```
$email = 'name@example.com';
$domain = strstr($email, '@');
echo $domain; // Affiche : @example.com
```

PHP : Fonctions avancées

- **htmlspecialchars** (chaine, flags) : Conversion des caractères spéciaux en entités HTML.
Remplace par exemple & (ET commercial) en &

Un exemple de flag est *ENT_QUOTES* qui Convertit les guillemets doubles et les guillemets simples.

```
$str = "This is some <b> bold </b> text.";  
echo htmlspecialchars($str);
```

// affichera : This is some bold text.

PHP : Fonctions avancées

- **strip_tags** (chaine, allowableTags) : Supprime les balises HTML et PHP d'une chaîne. Les commentaires HTML et PHP sont également supprimés

Ce comportement peut être modifié avec le paramètre

```
$text = '  
<p>Paragraph.</p><!-- Comment --><a href="#fragment">Other  
text</a>  
';  
  
// Autorise <p> et <a>  
echo strip_tags($text, '<p><a>');  
  
// affichera :  
// <p>Paragraph.</p> <a href="#fragment">Other text</a>
```

PHP : Fonctions avancées

- **strlen()** : retourne la longueur de la chaîne
- **strtolower()** : passe tous les caractères en minuscules
- **strtoupper()** : passe tous les caractères en majuscules
- **strpos(texte, chaine)** : recherche la position de la première « chaine » trouvée

PHP : Fonctions avancées

- **trim()** : efface les espaces blancs au début et à la fin d'une chaîne
- **ereg(chaine, texte)** : recherche dans *texte* , les séquences des caractères « *chaine* ».

Retourne la longueur de l'occurrence trouvée si une occurrence a été trouvée dans la chaîne *string*, FALSE dans le cas contraire ou si une erreur est survenue.

```
if ( ereg("BCD", "ABCDEF") )
    { echo "oui"; }
else
    { echo "non"; }
```

PHP : Fonctions avancées

- **isset(\$var1, \$var2, ...)** : Détermine si une variable ou plusieurs sont définies et sont différentes de NULL
- **unset(\$var1,\$var2,...)** : Détruit les variables
- **empty(\$var)** : Détermine si une variable est vide

PHP : Fonctions avancées

- Une variable non déclarée et une variable vide ne sont pas pareilles...

```
<?php  
echo(isset($var) . "-" . empty($var) . "<br>");  
$var = 42;  
echo(isset($var) . "-" . empty($var) . "<br>");  
$var = "";  
echo(isset($var) . "-" . empty($var) . "<br>");  
unset($var);  
echo(isset($var) . "-" . empty($var) . "<br>");  
?>
```

FALSE = 0
TRUE = 1

-1
1-
1-1
-1

PHP : Fonctions avancées

\$Var	isset	empty
<i>(\$var non déclarée)</i>	(0) FAUX	(1) VRAI
42	(1) VRAI	(0) FAUX
« »	(1) VRAI	(1) VRAI
(unset \$var)	(0) FAUX	(1) VRAI

PHP : Fonctions avancées

```
<?php  
echo (is_null($var) . "-" . empty($var) . ";;");  
  
$var=0;  
echo (is_null($var) . "-" . empty($var) . ";;");  
  
$var="";  
echo (is_null($var) . "-" . empty($var) . ";;");  
  
$var = null;  
echo (is_null($var) . "-" . empty($var) . ";;");  
?>
```

- Valeur « null » disponible, mais signifie que la variable existe
- Null != Empty

PHP Notice: Undefined variable: var in Standard input code on line 2
1-1 ;; -1 ;; -1 ;; 1-1

PHP : Fonctions avancées

\$Var	is_null	empty
<i>(\$var non déclarée)</i>	(1) VRAI*	(1) VRAI
0	(0) FAUX	(1) VRAI
« »	(0) FAUX	(1) VRAI
null	(1) VRAI	(1) VRAI

* : is_null() déclenche une alerte (PHP Notice) si la variable n'a pas été allouée

PHP : Fonctions avancées

- **array_values(array)** : Retourne toutes les valeurs d'un tableau
 - elles-mêmes dans un tableau
(utile pour vider cases vides)
- **array_keys(array, valeur)** : Retourne toutes les clés associées à « valeur »
 - si valeur n'est pas spécifiée alors retourne toutes les clés du tableau
(utile pour connaitre toutes les clés possibles)

PHP : Fonctions avancées

```
<?php  
$tab[0] = 42;  
$tab[1] = 42;  
$tab[2] = 9;  
$tab[5] = 42;  
$tab[8] = 6;  
  
$var1=array_values($tab);  
foreach($tab as $id => $val)  
{ echo "$id - $val ;; "; }  
echo "<br>";  
foreach($var1 as $id =>  
$val)  
{ echo "$id - $val ;; "; }  
?>
```

- **array_values(array)** : Retourne toutes les valeurs d'un tableau
 - Elles-mêmes dans un tableau

0 - 42 ;; 1 - 42 ;; 2 - 9 ;; 5 - 42 ;; 8 - 6

0 - 42 ;; 1 - 42 ;; 2 - 9 ;; 3 - 42 ;; 4 - 6

PHP : Fonctions avancées

```
<?php  
$tab["Prenom"] = "Jean";  
$tab["Nom"] = "Dupont";  
$tab["Prenom2"] = "Olivier";  
$tab["Prenom3"] = "Jean";  
$tab["Nom"] = "Martin";  
  
$var1=array_values($tab);  
foreach($tab as $id => $val)  
{ echo "$id - $val ;;"; }  
echo "<br>";  
foreach($var1 as $id => $val)  
{ echo "$id - $val ;;"; }  
?>
```

Prenom - Jean ;;
Nom - Martin ;;
Prenom2 - Olivier ;;
Prenom3 - Jean

0 - Jean ;;
1 - Martin ;;
2 - Olivier ;;
3 - Jean

PHP : Fonctions avancées

```
<?php  
$tab["Prenom"] = "Jean";  
$tab["Nom"] = "Dupont";  
$tab["Prenom2"] = "Olivier";  
$tab["Prenom3"] = "Jean";  
$tab["Nom"] = "Martin";  
  
$var1=array_keys($tab, "Jean");  
foreach($tab as $id => $val)  
{ echo "$id - $val ;"; }  
echo "<br>";  
foreach($var1 as $id => $val)  
{ echo "$id - $val ;"; }  
?>
```

- **array_keys(array, valeur)** : Retourne toutes les clés associées à « valeur »
 - si valeur n'est pas spécifiée alors retourne toutes les clés du tableau

0 - Prenom ;;
1 - Prenom3 ;;

PHP : Fonctions avancées

```
<?php  
$tab["Prenom"] = "Jean";  
$tab["Nom"] = "Dupont";  
$tab["Prenom2"] = "Olivier";  
$tab["Prenom3"] = "Jean";  
$tab["Nom"] = "Martin";  
  
$var1=array_keys($tab);  
foreach($tab as $id => $val)  
{ echo "$id - $val ;"; }  
echo "<br>";  
foreach($var1 as $id => $val)  
{ echo "$id - $val ;"; }  
?>
```

- **array_keys(array, valeur)** : Retourne toutes les clés associées à « valeur »
 - si valeur n'est pas spécifiée alors retourne toutes les clés du tableau

0 - Prenom ;;
1 - Nom ;;
2 - Prenom2 ;;
3 - Prenom3 ;;

PHP : Fonctions avancées

- **in_array(valeur, array)** : Vérifie si une valeur appartient à un tableau
- **array_key_exists(key, array)** : Vérifie si une clé existe dans un tableau

PHP : Fonctions avancées

```
<?php  
$tab["Prenom"] = "Jean";  
$tab["Nom"] = "Dupont";  
$tab["Prenom2"] = "Olivier";  
$tab["Prenom3"] = "Jean";  
$tab["Nom"] = "Martin";  
  
$var1=in_array("Jean", $tab);  
$var2=array_key_exists("Nom",  
$tab);  
$var3=in_array("A", $tab);  
$var4=array_key_exists("A", $tab);  
echo "$var1 - $var2 - $var3 - $var4";  
?>
```

TRUE = 1

FALSE = 0

1 - 1 - -

PHP : Fonctions avancées HTTP

- En PHP, la fonction `header()` se charge d'envoyer au client les en-têtes passés en paramètre
- Règle importante : l'appel à ‘`header()`’ doit se faire avant tout envoi au navigateur
 - Attention aux includes qui écriraient quelque chose...
 - Ne PAS écrire de `<html>` vu que l'on est dans les en-têtes HTTP et non HTML !

PHP : Fonctions avancées HTTP

- On peut utiliser la fonction « header() » pour envoyer des codes HTTP :

```
<?php  
    header("HTTP/1.0 404 Not Found");  
?>
```

```
<?php  
header("Status: 301 Moved Permanently", false, 301);  
header('Location: http://www.votresite.com/  
pageprotegee.php');  
?>
```

PHP : Fonctions avancées HTTP

```
$ telnet www.perdu.com 80
Trying 208.97.177.124...
Connected to www.perdu.com.
Escape character is '^]'.
```

Connexion au serveur par telnet

```
GET / http/1.1
Host: www.perdu.com
```

Requête HTTP

```
HTTP/1.1 200 OK
Date: Sat, 17 Aug 2013 11:59:04 GMT
Server: Apache
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.1.23.1-2169
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
Content-Length: 204
Content-Type: text/html
```

Réponse du serveur : headers

Header HTTP renvoyé par Apache...
...donc ce que vous pouvez modifier
avec la fonction « header() »

```
<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Interne
t ?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>      * ----- vous
&ecirc;tes ici</pre></strong></body></html>
```

Réponse du serveur : body

PHP : Fonctions avancées HTTP

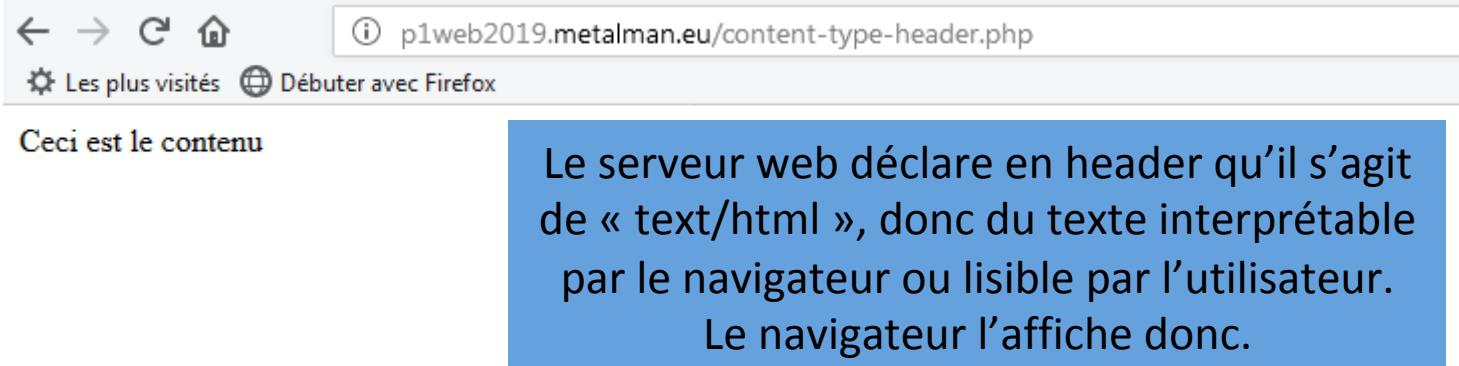
- Pour créer une redirection avec PHP, on utilise `header` pour envoyer des en-têtes de type *Location (adresse)*
- Script de redirection (HTTP 302) :

```
<?php  
header('Location: http://monsite.com/page.php');  
?>
```

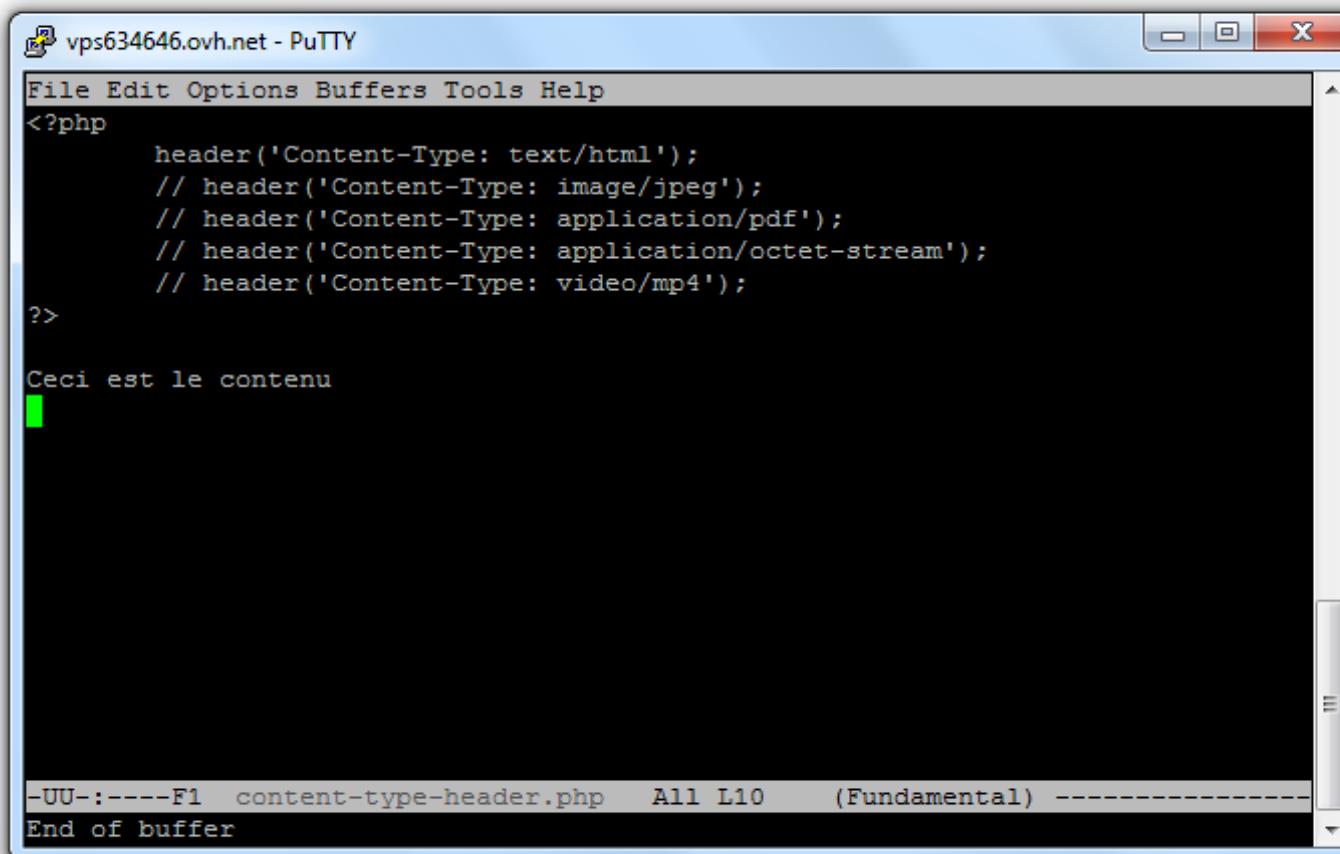
PHP : Fonctions avancées HTTP

- On peut envoyer des fichiers et jouer avec d'autres champs HTTP tels que le « Content-Type »
 - « Content-Type » indique le type de donnée, donc le navigateur essayera d'utiliser l'outil préféré de l'utilisateur pour lire ce fichier
- `readfile()` permet d'envoyer un fichier

PHP : Fonctions avancées HTTP



A screenshot of a Firefox browser window. The address bar shows the URL `p1web2019.metalman.eu/content-type-header.php`. The page content area contains the text "Ceci est le contenu". To the right, a blue callout box contains the explanatory text: "Le serveur web déclare en header qu'il s'agit de « text/html », donc du texte interprétable par le navigateur ou lisible par l'utilisateur. Le navigateur l'affiche donc."



A screenshot of a PuTTY terminal window titled "vps634646.ovh.net - PuTTY". The window displays the following PHP code:

```
File Edit Options Buffers Tools Help
<?php
    header('Content-Type: text/html');
    // header('Content-Type: image/jpeg');
    // header('Content-Type: application/pdf');
    // header('Content-Type: application/octet-stream');
    // header('Content-Type: video/mp4');
?>

Ceci est le contenu
```

The terminal window also shows the file path at the bottom: "-UUU:----F1 content-type-header.php All L10 (Fundamental) -----".

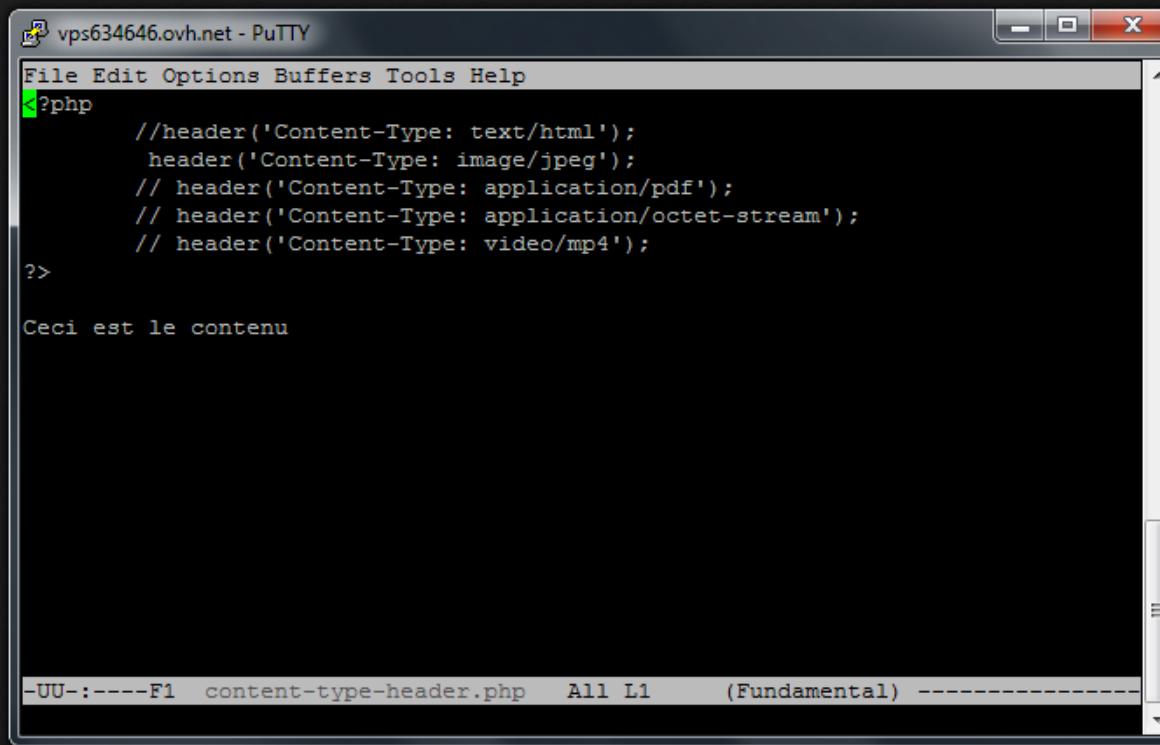
PHP : Fonctions avancées HTTP

Le serveur web déclare en header qu'il s'agit de « image/jpeg », donc d'une image au format JPEG. Le texte n'en étant pas, il est impossible d'afficher une image, mais le navigateur essaye tout de même

p1web2019.metalman.eu/content-type-header.php

Débuter avec Firefox

L'image < http://p1web2019.metalman.eu/content-type-header.php > ne peut être affichée car elle contient des erreurs.



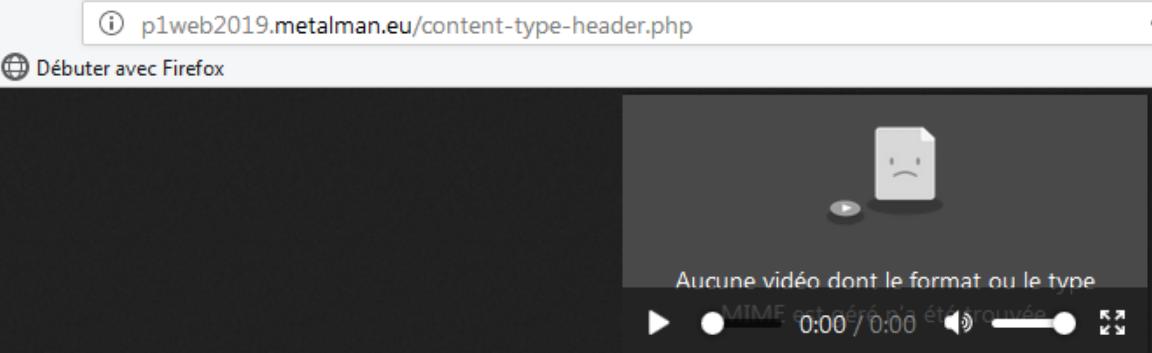
The screenshot shows a PuTTY terminal window titled "vps634646.ovh.net - PuTTY". The window contains the following PHP code:

```
<?php
    //header('Content-Type: text/html');
    header('Content-Type: image/jpeg');
    // header('Content-Type: application/pdf');
    // header('Content-Type: application/octet-stream');
    // header('Content-Type: video/mp4');
?>

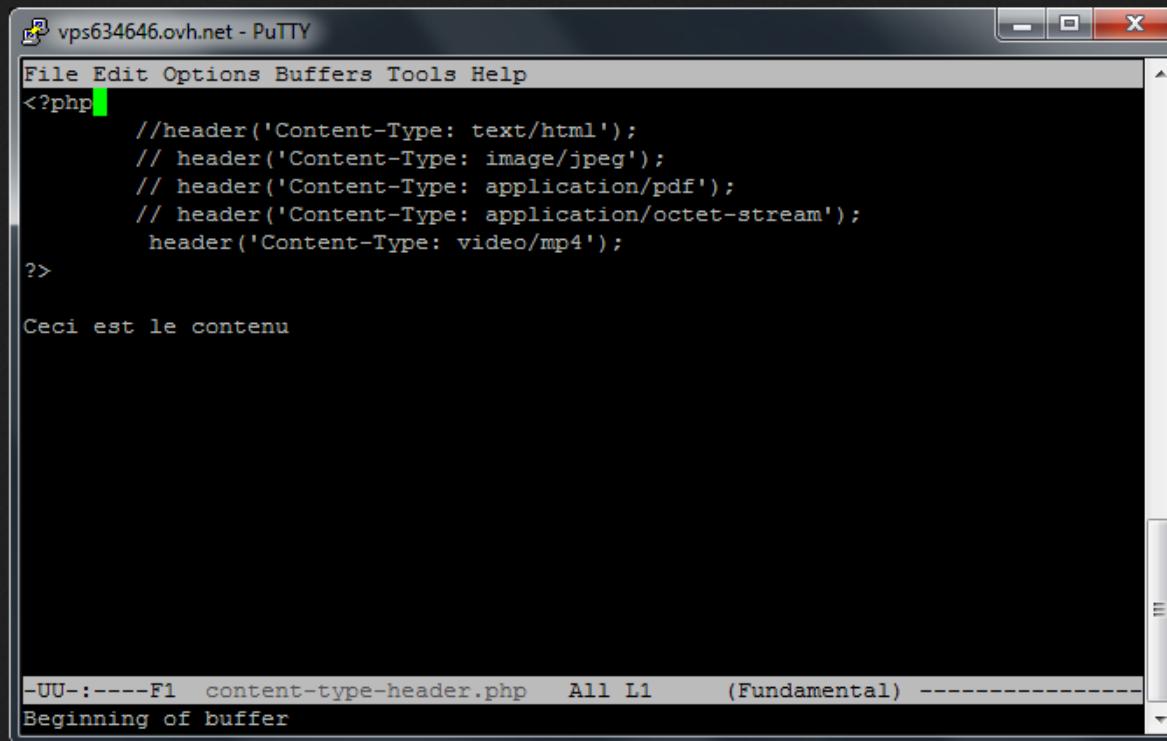
Ceci est le contenu
```

The output of the code is displayed below the code block in the terminal window. At the bottom of the window, the status bar shows the file name "content-type-header.php", the line count "All L1", and the word count "(Fundamental)".

PHP : Fonctions avancées HTTP



Le serveur web déclare en header qu'il s'agit de « video/mp4 », donc d'une vidéo au format MP4. Le texte n'en étant pas, il est impossible d'afficher la vidéo, mais le navigateur essaye tout de même de le lire avec un lecteur de vidéos

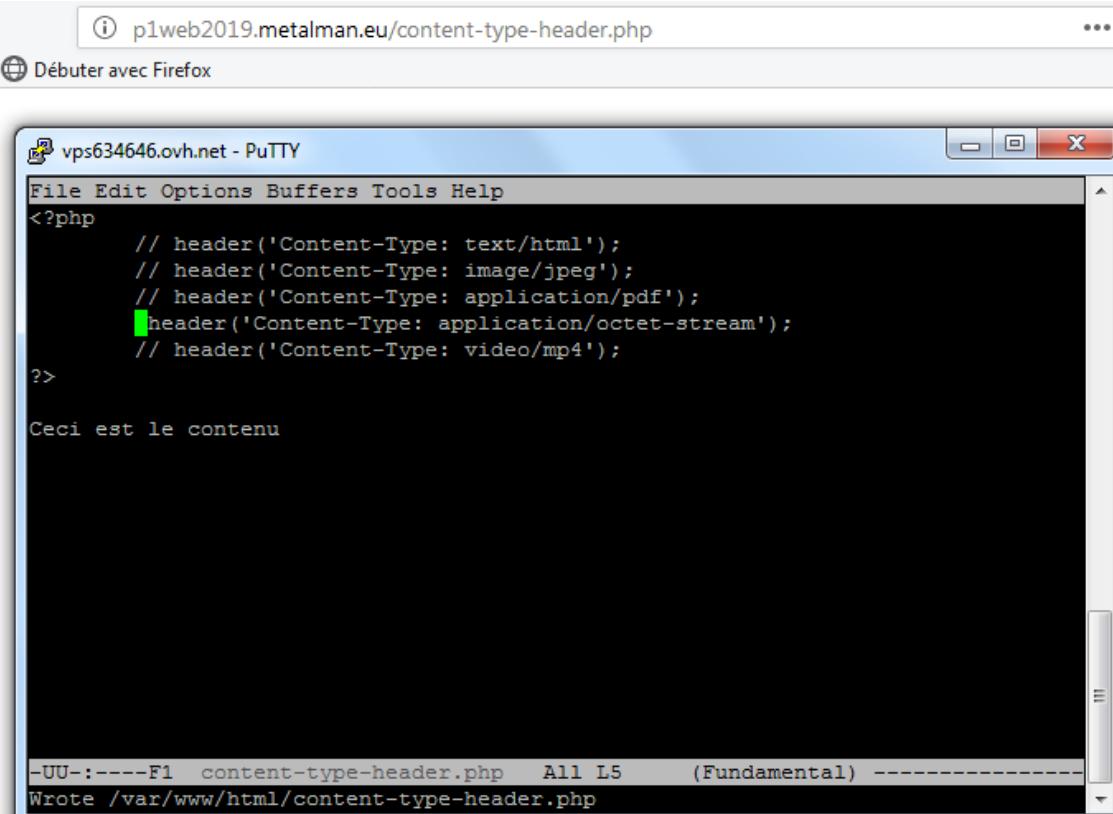


```
vps634646.ovh.net - PuTTY
File Edit Options Buffers Tools Help
<?php
    //header('Content-Type: text/html');
    // header('Content-Type: image/jpeg');
    // header('Content-Type: application/pdf');
    // header('Content-Type: application/octet-stream');
    header('Content-Type: video/mp4');
?>

Ceci est le contenu

-UU-:----F1  content-type-header.php  All L1      (Fundamental) -----
Beginning of buffer
```

PHP : Fonctions avancées HTTP



A screenshot of a Firefox browser window. The address bar shows the URL p1web2019.metalman.eu/content-type-header.php. The page content displays the output of a PHP script. The script contains several `header()` statements, one of which is highlighted with a green rectangle. The output shows the text "Ceci est le contenu".

```
<?php
    // header('Content-Type: text/html');
    // header('Content-Type: image/jpeg');
    // header('Content-Type: application/pdf');
    header('Content-Type: application/octet-stream');
    // header('Content-Type: video/mp4');

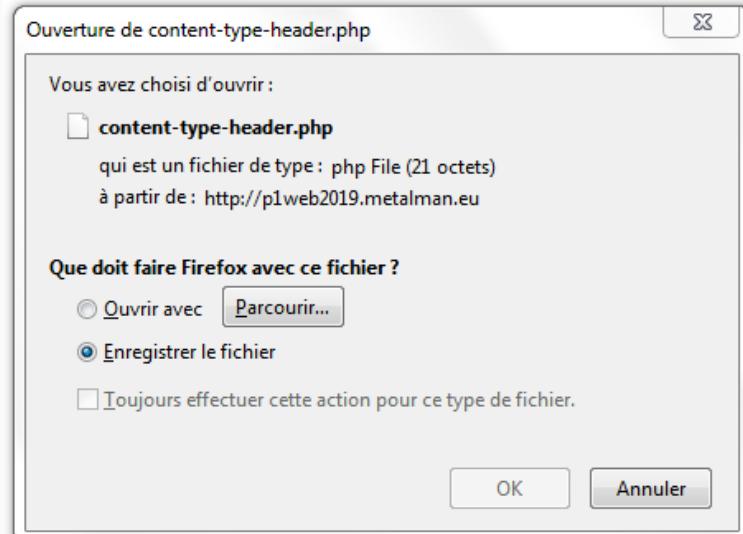
?>

Ceci est le contenu
```

-UU-:----F1 content-type-header.php All L5 (Fundamental) -----
Wrote /var/www/html/content-type-header.php

Le serveur web déclare en header qu'il s'agit de « application/octet-stream », donc d'un flux d'octets.

Par défaut, les navigateurs proposent de télécharger le fichier associé plutôt que de l'afficher



PHP : Fonctions avancées HTTP

```
<?php
// On declare un PDF
header('Content-Type: application/pdf');

// Le client verra un fichier "downloaded.pdf"
header('Content-
Disposition: attachment; filename="downloaded.pdf"');

// Le fichier du serveur web qui sera transmis
readfile('original.pdf');
?>
```

PHP : Fonctions avancées HTTP

```
<?php
$file = 'monkey.gif';

if (file_exists($file)) {
    header('Content-Description: File Transfer');
    header('Content-Type: application/octet-stream');
    header('Content-
Disposition: attachment; filename="'.basename($file).'"');
    header('Expires: 0');
    header('Cache-Control: must-revalidate');
    header('Pragma: public');
    header('Content-Length: ' . filesize($file));
    readfile($file);
    exit;
}
?>
```