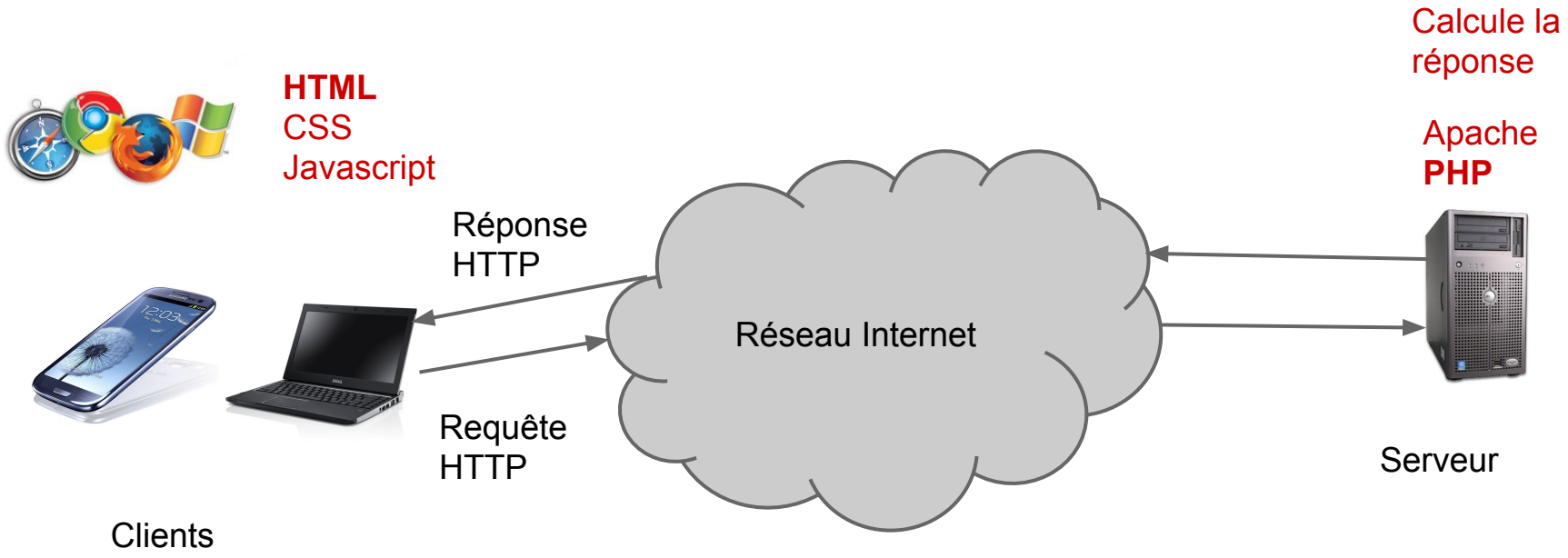
Technologies du web

4 - Formulaires

Architecture client / serveur



Détails d’une requête

1. L’utilisateur fait une requête depuis son navigateur (url, lien…)
2. La requête est envoyé vers le serveur désigné (DNS/IP / Protocol HTTP)
3. Le logiciel serveur HTTP reçoit et analyse la requête
4. Il charge la ressource désignée depuis le répertoire ‘root’ du site
5. Si la ressource est un fichier PHP, le code est executé
6. Le résultat du traitement est renvoyé par le serveur vers le client (HTTP)
7. Le navigateur reçoit le résultat HTML/CSS/Javascript
8. Le navigateur analyse ce résultat et l’affiche

Les formulaires HTML

Pourquoi ?

- Permettre à l'utilisateur de saisir des informations et de les envoyer au serveur (et non pas juste d'en lire)

- Point d'entrée de tout type d'interaction utilisateur sur une application web

Exemple de formulaire

En HTML : La balise form

```
<p>Mon formulaire</p>
<form>

    <!-- Le contenu du formulaire -->

</form>
```

Types d’entrée

- Texte
- Choix multiple
- Boutons

Balise <input ...> ou <input … />

Text

Password

Date mm/dd/yyyy

Number

Range

Du texteâ€¦

Checkboxes
- ketchup
- Mayonnaise

Radio
- ketchup
- Mayonnaise

Select
France

bouton

Formulaire : texte court

```
<form>
    <label>Nom</label>
    <input type="text" name="nom" id="nom" />
</form>
```

Attributs optionnels: *placeholder, maxlength, size, …*

Attention : name != id

Formulaire : texte spéciaux

```
<form>
    <label>Mot de passe</label>
    <input type="password" name="mdp" id="mdp">
    <label>Email</label>
    <input type="email" name="courriel" id="courriel">
    <label>URL</label>
    <input type="url" name="site" id="site">
</form>
```

Formulaire : nombre

```
<form>
    <input type="number" name="age" id="age">
    <input type="range" name="un-nbr" id="un-nbr">
</form>
```

Attributs : min, max, step, value

Formulaire : autre

```
<form>
    <input type="date">
    <input type="search">
</form>
```

Formulaire : texte long

```
<form>
    <label>Nom : </label><br/>
    <textarea name="details" id="details" rows="10">
    Du texte…
    </textarea>
</form>
```

Formulaire : case à cocher

<input type="checkbox" name="choix" checked>

“checked” est optionnel

Formulaire : Cases à cocher

```
<form>
   <p> Cochez votre choix: <br />
      <input type="checkbox" name="frites" id="frites">
      <label for="frites">Frites</label><br />
      <input type="checkbox" name="steak" id="steak">
      <label for="steak">Steak haché</label><br />
   </p>
</form>
```

Formulaire : Liste de choix

```
<form>
   <p> Cochez un seul choix: <br/>
      <input type="radio" name="sauce" id="ketchup">
      <label for="ketchup">ketchup</label><br/>
      <input type="radio" name="sauce" id="mayo">
      <label for="mayo">Mayonnaise</label><br/>
   </p>
</form>
```

Liste deroulante (combo box)

```
<form>
    <select id="pays">
        <option value="france">France</option>
        <option value="espagne">Espagne</option>
        <option value="italie">Italie</option>
  </select>
</form>
```

Formulaire : Bouton

```
<input type="button" value="un bouton">
<button type="button">Mon bouton</button>
```

Champ obligatoire

```
<input type="text" name="prenom" id="prenom" required />
```

Fieldset et legend

```
<form>
    <fieldset>
    <legend>Utilisateur</legend>
    <input …. >
    </fieldset>
</form>
```

Type hidden

```
<input type="hidden" name="variable-secrete" value="toto">
```

Permet d’avoir des valeurs cachées (pour garder par exemple un id dans le cas d’un formulaire multi page etc...)

Formulaire : Envoi du formulaire

```
<input type="submit" value="Envoyer">
```

Action du formulaire

Par défaut, une requête HTTP est envoyée lorsque le **bouton submit** est cliqué.

On spécifie la méthode HTTP et l'URL dans les attributes de la balise <form>

```
method="GET" ou "POST"
action="URL_TO_SEND_DATA"
```

Exemple de formulaire

```
<form method="GET" action="login.php">
    <fieldset>
        <legend>Sign in</legend>
        <label for="login">Login</label>
        <input name="login" id="login" type="text"><br/>
        <label for="pwd">Password</label>
        <input name="pwd" id="pwd" type="password"><br/>
        <input type="submit" value="login">
    </fieldset>
</form>
```

Exemple de formulaire

Envoi de fichiers

```
<form enctype="multipart/form-data" action="./upload.php"
method="post">
    <input type="file">
    <input type="submit">
</form>
```

Présente un bouton permettant de sélectionner un fichier. La méthode GET ne fonctionne pas pour les fichiers.

Requête formulaire

1. L’utilisateur remplit le formulaire
2. Il clique sur **submit**
3. Le navigateur envoie une nouvelle requête à l’adresse spécifiée dans l’attribut “**action**” avec les données du formulaire et suivant la **méthode** choisie (GET ou POST)
4. Le serveur traite la requête et envoie une réponse sous la forme d’une nouvelle page HTML
5. Le navigateur affiche cette nouvelle page

Pour aller plus loin

- http://www.w3schools.com/html/html_forms.asp

Traitement de formulaire coté serveur en PHP

Traitement de Formulaire HTML

PHP a été inventé en partie pour ce type de traitement.

Dans le formulaire HTML, on spécifie comment sont traitées les données

action : l'adresse d'un script PHP qui traitera le formulaire

method : defini le verbe HTTP utilisée

```
<form action="http://monsite/login.php" method="GET">
    <input type="text" id="login" name="login">
    <input type="password" id="pass" name="pass">
    <input type="submit">
</form>
```

Méthode GET

GET : les données du formulaire sont ajoutées à l'url automatiquement par le navigateur.

Le navigateur ajoute un **?** après l'url et chaque élément du formulaire sous la forme ***name=valeur***

Chaque couple nom/valeur est séparé par un **&**

http://monserveur.fr/login.php**?**login=toto**&**pass=titi

Méthode POST

POST : les données sont passées **dans le corps de la requête HTTP**, les valeurs ne sont pas visibles dans la barre d'adresse (dans l'url), mais ne sont pas cryptées

http://monserveur.fr/login.php

GET ou POST

GET : Quand on veut pouvoir utiliser l'url avec les paramètres (ex moteur de recherche). **Volume de données limité**

http://www.google.fr?q=toto

POST : Quand **le volume de données est grand** et/ou que l'on veut cacher les données, ou que l'on envoie un fichier.

Traitement de formulaire en PHP

Le script PHP, sur le serveur, peut utiliser des données envoyées depuis un formulaire.

PHP fournit des **variables globales** contenant les données de formulaire sous forme de **array** :

$_POST : Array contenant les paires **name**/value passées en POST

$_GET : Array contenant les paires **name**/value de l'URL

$_REQUEST : Array contenant les valeurs de $_POST et $_GET

```
echo $_GET["login"], $_GET["pass"];
```

Les Arrays PHP

- Peut contenir une liste

```
$list = array(6,7,5,3);
echo $list[2]; // 5
```

- Peut contenir un tableau associatif
  - association clé / valeur

```
$ages = array("jean"=>20, "vincent"=>22, "julie"=>21);
echo $ages["vincent"]; // 22
```

Tableau simple (Array)

- Un Array contient une liste de valeur
  - indexée par leur position

```
<?php
  // liste de couleurs
  $couleurs = array("rouge", "vert", "bleu", "jaune");
  echo $couleurs[0] . "<br/>";
  echo $couleurs[2] . "<br/>";
  print_r($couleurs); // affiche le tableau
?>
```

Tableau associatif

- Un Array peut contenir aussi un tableau associatif
  - association clé / valeur

```
<?php
$ages = array("jean"=>20, "vincent"=>22, "julie"=>21);
echo $ages["vincent"]; // 22
 ?>
```

Formulaire PHP

On peut utiliser **le même fichier PHP** pour envoyer le formulaire au client et pour traiter les données envoyées depuis le client (lorsque l'utilisateur clique sur submit).

Mais ici, nous **séparons** le formulaire et le code de traitement pour une meilleure lisibilité.

- Le code HTML du formulaire : *form.php*
- Le code PHP permettant de traiter les données du formulaire et renvoyer la reponse : *login.php*

Requête avec formulaire séparé

Clients

1 - demande de form.php

2 - Retour de form.php

3 - Submit form vers login.php

4 - Retour du resultat de login.php

Apache
PHP

Serveur

Le Formulaire : form.php

```
<html>
 <body>
   <form action="login.php" method="post">
           <input type="text" name="login" id="login" value="">
           <input type="password" name="pass" id="pass" value="">
           <input type="submit" name="submit" value="Identification">
   </form>
 </body>
</html>
```

login.php (appelé à l'envoi)

```php
<?php
  $message = '';      // Message à afficher à l'utilisateur
  if(!empty($_POST)) { // le formulaire a été envoyé
      // on verifie login et le mot de passe
       if ($_POST['login'] == 'sam' && $_POST['pwd'] =='toto') {
          $message = 'login ok';
      } else {  $message = 'erreur de login';       }
   } else { // $_POST EMPTY
      $message = 'aucune donnée envoyée'
   }
?>
<html>
 <body>
     <?php echo $message; ?>
 </body>
</html>
```

Redirection

Dans le formulaire:

- le login/mdp est erroné -> on renvoie le formulaire avec un message d’erreur
- le login/mdp est correct -> on redirige vers une nouvelle page

Redirection

```php
<?php
 $message = '';      // Message à afficher à l'utilisateur
  if(!empty($_POST)) { // Si le tableau $_POST (le formulaire a été envoyé)
   if ($_POST['login'] == LOGIN && $_POST['PASS'] == PASS) {

     header("Location: http://www.monsite.fr/autre_page.php");

   } else {
     // erreur, on renvoie le formulaire
     header("Location: http://www.monsite.fr/form.html");
   }
 } ?>
```

Envoi de fichier par formulaire

HTML

```
<input type="file">
```

PHP

```
$_FILES['userfile']['name'] #Le nom original du fichier, tel que sur la machine du client web.
$_FILES['userfile']['type'] #Le type MIME du fichier. ex image/jpg
$_FILES['userfile']['size'] #La taille, en octets, du fichier téléchargé.
$_FILES['userfile']['tmp_name'] #Le nom temporaire du fichier qui sera chargé sur le serveur.
$_FILES['userfile']['error'] #l’erreur s’il y en a une
```

Conserver des valeurs utilisateurs

L’appel de chaque page PHP est **indépendant**.

Or on peut avoir besoin de **conserver des valeurs** entre les différentes requêtes au serveur

Exemples : préférences utilisateur, numéro client en cours, panier d’achat, clé d’authentification (pour conserver une authentification entre les pages), formulaire sur plusieurs pages.

Login

Password

Formulaire 2

Clé d'identification

Test le login et renvoie une clé

Valide la clé
Et le deuxième formulaire

Solution 1 : Préremplir le formulaire

Lorsqu'on envoie le formulaire au client, on peut pré-remplir les champs du formulaire avec des valeurs obtenues lors des appels précédents.

=> conserver une clé d'authentification

=> conserver les valeurs de formulaires précédents

Ces valeurs seront renvoyées au submit suivant.

On peut utiliser un champ de type "hidden"

Exemple de champ prérempli

```
// search.php?q="ma requete"
// on veut que le champ de recherche conserve la valeur saisie par l'utilisateur
// d'un appel sur l'autre

<form action="search.php" method="GET">
     <input name="q" type="search" value="<?php echo $_GET["q"] ?>">
     <input type="submit">
</form>
<p>
     les resultats :
     <ul>
       <?php /*on calcule le resultat ici si  q n'est pas vide * / ?>
     </ul>
</p>
```

Solution 2 : Les Cookies

Un cookie est une information stockée **sur le Poste client** (navigateur). On peut y stocker

- des données relatives aux visites effectuées
- des clés d'accès temporaires
- des préférences qui seront conservées entre plusieurs visites du site

Les cookies sont supportés par HTTP

Manipulation de cookie en PHP

Ecrire un cookie

```
setcookie ($name, $value, $expire);
```

Lire un cookie

```
$_COOKIE[name]
```

Les sessions PHP

Les sessions sont un mécanisme propre à PHP pour conserver des données utilisateur entre les différents appels à des pages.

Ils sont basés sur l'utilisation de cookies.

On préférera utiliser les sessions aux cookies.

Session PHP

`session_start()` : crée ou restaure une session, à placer en debut de fichier dans toute page participant à une session

`$_SESSION` : variable globale; permet l'enregistrement de variables de session
`$_SESSION["nomdelavariable"] = x` ajoute une variable à la session
`$_SESSION["nomdelavariable "]` accède à une variable de la session

`unset($_SESSION["nomdelavariable "])` retire la variable
`session_destroy()` détruit la session en cours

Exemple de Session

On veut faire un formulaire sur 2 pages:

- Page 1 : Nom + Prenom + Adresse : form1.php
- Page 2 : Details de paiement : form2.php

Nous avons besoin de conserver le nom et prenom pour la deuxieme page.

Form1.php

```
<html>
<body>
    <form action="form2.php">
        <input type="text" name="nom">
        <input type="text" name="prenom">
        <input type="text" name="adresse">
        <input type="submit">
    </form>
</body>
</html>
```

form2.php

```
<?php
     session_start();
     // sauvegarde de champ du formulaire
     $_SESSION["nom"] = $_REQUEST["nom"];
     $_SESSION["prenom"] = $_REQUEST["prenom"];
?>
<html>
  <body>
     <form action="payment.php">
          <input type="text" name="credit-card">
          <input type="submit">
     </form>
  </body>
</html>
```

payment.php

```
<?php
     session_start();
     // recupere les valeurs saisies précédemment
     $nom = $_SESSION["nom"];
     $prenom = $_SESSION["prenom"];
     $cb = $_REQUEST["credit-card"];
?>
<html>
     <body>
       <p>Paiement accepté pour Mr <?php echo $nom ?></p>
    </body>
</html>
```

Ce qu’il faut retenir

- PHP permet de générer des formulaires HTML et de traiter les données envoyées par l’utilisateur

- Il permet de conserver des valeurs entre les différentes requêtes.