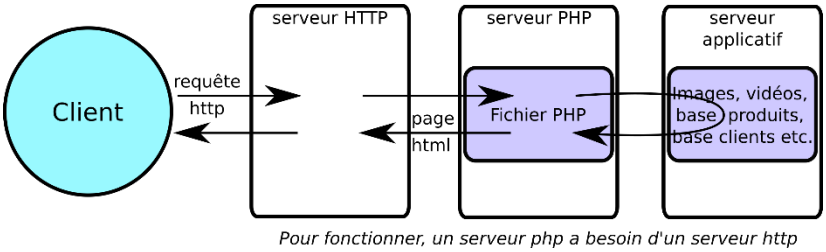
NSI – 1ere	COURS Modalités de l'interaction Homme Machine sur le WEB Formulaire.	LFV

I. Interactions client-serveur et protocole http(s) : généralités



Pour fonctionner, un serveur php a besoin d'un serveur http

Remarque : dans ce cours et dans les TP nous utiliserons php mais le code côté serveur peut être python, Ruby etc.

1. Suite de protocoles de communication utilisés

Comme nous l'avons déjà dit, de nombreux protocoles vont opérer en cascade pour assurer la communication entre un client et un serveur sur le web :

- http et https pour la couche «application» ;
- TCP pour la couche «transport» : découpe en paquets, identification de l'application via les numéros de port (80 et 443 traditionnellement pour http et https), fiabilisation de la transmission (récupération de paquets) ;
- IP pour la couche «internet» : gestion de l'adressage logique (@IP), routage des paquets,
- Ethernet (très majoritairement) pour la couche «liaison» : gestion de l'adressage physique (@MAC), commutation de paquets et évitement des collisions, gestion des caractéristiques physiques de la communication.

2. Deux méthodes d'envoi des données au serveur : POST et GET

Dans le premier cours (introduction aux relations client-serveur), nous avons constaté que le client effectuait des requêtes http avec la méthode GET. Dans le cas de l'utilisation d'un formulaire, nous avons vu que les requêtes avec la méthode GET permettent d'envoyer des paramètres au serveur et que ces paramètres sont visibles dans l'URL :

http://192.168.xxx.yyy/NSI_TABLEAU_A_COMPLETER/reponse.php?nom=jojo+lapin&age=28

Remarquer que :
- l'adresse du fichier traitant la demande est bien indiquée : ici il s'agit de reponse.php,
- les paramètres sont placés après un point d'interrogation et ils sont séparés par le symbole &,
- l'URL ne comporte aucun espace (remplacé par un + dans les valeurs des paramètres).

Il existe en fait deux méthodes (GET et POST) pour envoyer des paramètres au serveur. Voici leur comparatif :

	GET	POST
Données envoyées visibles dans l'URL	☑	
Requête conservée dans le cache	☑	
Requête visible dans l'historique ou enregistrable dans les marque-pages	☑	
Utilisable pour les données sensibles		☑
Données envoyées de longueur illimitée (par exemple envoi d'une image)		☑

Remarque : il existe d'autres méthodes de requête http : HEAD, DELETE etc. mais elles sont très spécifiques et peu utilisées lorsqu'on débute.

3. Transmission chiffrée ou non chiffrée (n'a rien à voir avec méthode d'envoi GET ou POST)

Le fait d'utiliser une méthode de requête GET ou POST est indépendant d'établir une connexion sécurisée.

Par exemple si vous utilisez la méthode POST, vous évitez qu'un autre utilisateur de la machine client puisse voir (dans l'historique par exemple) les paramètres de la requête un jour ou deux jours plus tard. Mais cela n'empêche personne – au moment où vous faites la requête – d'écouter les paquets transmis sur le réseau et donc d'accéder aux paramètres transmis (cela demande juste plus de travail que de regarder l'historique).

Pour éviter que la transmission soit interceptée, il est nécessaire d'utiliser une connexion sécurisée ce qui se fait en utilisant le protocole httpS (et pas http). Dans ce cas tous les paquets d'information sont chiffrés et personne ne peut lire leur contenu mis à part le serveur. L'établissement d'une connexion sécurisée et les méthodes cryptographiques associées seront étudiées au programme de terminale.

Au final, pour transmettre des données sensibles il faut utiliser à la fois la méthode POST et le protocole HTTPS.

4. Code d'état (ou code http) d'une requête renvoyé par le serveur

Lorsqu'un navigateur effectue une requête http (en transmettant ou pas des paramètres), le serveur renvoie un code d'état qui permet d'indiquer le résultat de cette requête ou d'indiquer une erreur. Les plus courants sont :

- 200 : succès de la requête (les codes 2xx coresspondent à des succès)
- 304 : document non modifié depuis la dernière requête (les codes 3xx correspondent à des redirections)
- 404 : ressource non trouvée (les codes 4xx correspondent à des codes d'erreur)

On pourra consulter cette page wikipédia https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP.

II. Les formulaires : côté client en détail

Un formulaire permet à l'utilisateur de saisir des valeurs qui seront traitées sur le serveur. Vous utilisez des formulaires lorsque vous créez un compte, répondez à des questions ou validez un panier d'achat. Le html (interprété côté navigateur) permet de créer le formulaire alors que le php (interprété côté serveur) permet de traiter les informations reçues (voir schéma en début de cours).

Le formulaire est introduit par la balise HTML <form> qui contiendra au moins deux attributs :

- **L'attribut "action" :** il contient le nom du script php qui sera appelé lorsque le formulaire sera validé par l'utilisateur (la page qu'il faut afficher après validation du formulaire).
- **L'attribut "method" :** il indique la méthode d'envoi des informations au serveur (GET ou POST).

Le formulaire comporte ensuite une ou plusieurs balises <input/> pouvant être de différents types :

`<input type="text" name="prenom"/>`	pour saisir du texte
`<input type="radio" name="oui_ou_non" value="oui"/>` `<input type="radio" name="oui_ou_non" value="non"/>`	boutons radio : un seul de tous les boutons ayant le même name peut être sélectionné
`<input type="checkbox" name="chat" />` `<input type="checkbox" name="chien" />`	cases à cocher : chaque case est indépendante des autres : on peut cocher ou pas chacune d'entre elles
`<input type="hidden" name="info_cachee" value=……/>`	information cachée : pour inclure dans le formulaire des informations cachées à l'utilisateur qui seront renvoyées au serveur (par exemple l'adresse IP du client)
`<input type="submit" value="Envoyer !" />`	bouton permettant de soumettre le formulaire au serveur

On voit que toutes les balises <input/> (sauf la dernière qui ne contient pas d'informations) contiennent :

- **un attribut "type" :** il précise quel est le type d'élément graphique à mettre en place,
- **un attribut "name" :** il précise quelle est la «clef» qui permettra de récupérer l'information de la balise correspondante côté serveur. À noter deux exceptions pour les <input/> de type "submit" ou "radio"

Remarque importante : ne pas confondre les attributs id qui sont utiles pour identifier des balises côté client et les attributs name qui sont utiles pour identifier les informations associées aux balises input côté serveur.

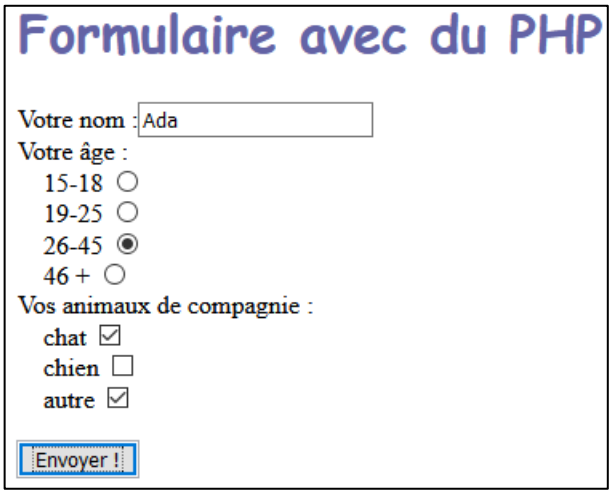
Pour aller plus loin :
Pour coder proprement, on peut rajouter une balise <label><label/> afin de légender les différentes balises <input/> du formulaire. La balise <label> doit alors comporter l'attribut for qui indique l'attribut id de la balise <input> associée (c'est un peu lourd lorsqu'on débute car il faut alors utiliser l'attribut name ET l'attribut id pour les <input/>).

Au final voici un exemple de formulaire:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"/>
        <title>NSI - Formulaire</title>
        <link rel="stylesheet" href="style.css"/>
    </head>
    <body>
        <h1> Formulaire avec du PHP </h1>
    <form method="post" action="traitement.php">
            Votre nom :<input type="text" name="nom"> <br/>
            Votre âge : <br/>
                15-18 <input type="radio" name="age" value="15-18"> <br/>
                19-25 <input type="radio" name="age" value="19-25"> <br/>
                26-45 <input type="radio" name="age" value="26-45"> <br/>
                46 + <input type="radio" name="age" value="46 +"> <br/>
            Vos animaux de compagnie : <br/>
                chat <input type="checkbox" name="chat"> <br/>
                chien <input type="checkbox" name="chien"> <br/>
                autre <input type="checkbox" name="autre"> <br/>
            <input type="hidden" name="ip" value= <?php echo $_SERVER['REMOTE_ADDR'] ; ?> > <br/>
            <input type="submit" value="Envoyer !">
    </form>
    </body>
</html>

```

Qui produira comme affichage ce qui est indiqué ci-contre.



Dans le html du formulaire remarquer que :

- la balise form possède bien les attributs method et action ;

- toutes les balises input ont un attribut type et un attribut name (sauf celle de type "submit") ;

- toutes les balises input de type radio – qui correspondent à la même question – ont le même attribut name ;

- pour les balises input de type radio, c'est un attribut value qui va permettre de les distinguer côté serveur

- la balise input de type "hidden" sert à contenir la valeur de l'adresse IP du client (déterminée par le serveur) qui pourra ainsi être renvoyée au serveur comme paramètre avec le formulaire. Cela permet au serveur d'effectuer la sauvegarde de l'adresse IP du client en même temps que le reste du formulaire. Bien noter ici que l'adresse IP du client fait un aller-retour : serveur --> client --> serveur.

III. Les formulaires : côté serveur en détail

Le fichier PHP indiqué par l'attribut action de la balise form va «recevoir» les données du formulaire sous la forme d'un tableau associatif (un dictionnaire) appelé $_POST si la méthode d'envoi est POST et $_GET si la méthode d'envoi est GET. Dans ce dictionnaire $_POST, on accède aux valeurs des différentes informations saisies en utilisant comme clef la valeur de l'attribut name du formulaire ce qui donne par exemple :

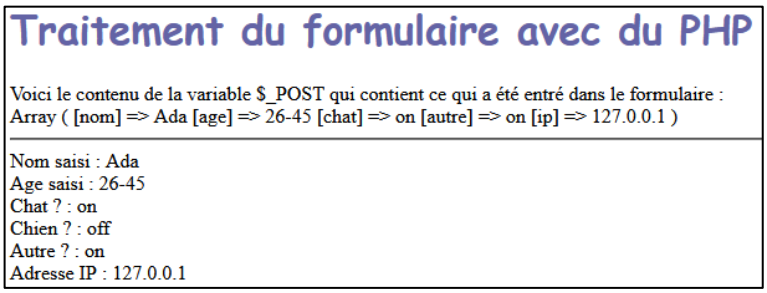
```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"/>
        <title>NSI - Formulaire</title>
        <link rel="stylesheet" href="style.css"/>
    </head>

    <body>
        <h1> Traitement du formulaire avec du PHP </h1>
        <?php
            if($_POST) {
                echo 'Voici le contenu de la variable $_POST qui contient ce qui
                 a été entré dans le formulaire : <br/>';
                print_r($_POST);
                echo "<hr>" ;

                echo "Nom saisi : " . $_POST['nom'] . "<br/>" ;
                echo "Age saisi : " . $_POST['age'] . "<br/>" ;
                if (!isset($_POST['chat'])){
                    $_POST['chat'] = "off" ;
                }
                if (!isset($_POST['chien'])){
                    $_POST['chien'] = "off" ;
                }
                if (!isset($_POST['autre'])){
                    $_POST['autre'] = "off" ;
                }
                echo "Chat ? : " . $_POST['chat'] . "<br/>" ;
                echo "Chien ? : " . $_POST['chien'] . "<br/>" ;
                echo "Autre ? : " . $_POST['autre'] . "<br/>" ;

                echo "Adresse IP : " . $_POST['ip'] . "<br/>" ;
            }
            sauvegarder_csv('donnees_sondages.csv', $_POST) ;
        ?>
    </body>
</html>
```

L'affichage de traitement.php suite à la validation du formulaire de index.html va d'une part renvoyer à l'utilisateur une page de validation comme celle-ci-contre.



Mais va également sauvegarder sur le serveur les données saisies dans un fichier csv (en production, il faudrait sauvegarder sur une base de données, ce sera vu en terminale) grâce à une fonction `sauvegarder_csv()` dont le contenu n'est pas affiché ci-dessus.

Remarque 1 :
Noter que la concaténation de chaînes de caractères s'effectue avec l'opérateur . (et pas + comme en python).

Remarque 2 :
Les `if(!isset( ))` sont là car si une checkbox n'est pas cochée (comme ici pour le chien), le tableau $_POST ne contient pas la clef correspondante. `isset( )` permet de tester si la clef est présente ou pas : lorsque la clef n'est pas présente on l'ajoute au tableau $_POST en lui associant la valeur "off".