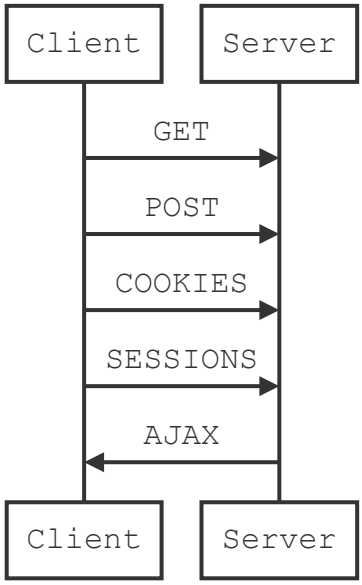
Accès Aux Données

La plus part des informations échangées entre un serveur et son client ne sont pas toujours sous la forme de documents php. Pour rappel, le php est *stateless*[1]. Chaque requête doit donc être traitée de manière indépendante.

Dans la majorité des cas, c'est le client qui enverra ses informations à direction du serveur qui traitera la requête en fonction.



- **GET** : informations dans l'**adresse** de la requête
- **POST** : informations dans l'**entête** de la requête
- **Cookies** : informations **stockées chez le client**
- **Sessions** : informations **stockées sur le server**

Get et Post

GET

Get récupère les informations placées dans l'entête de l'url.

```
1 www.monsite.com/page?clef1=val1&clef2=val2
```

> Attention, pas plus de 2048 caractères

Inconvénients

- Méthode la moins sur car les données sont visibles dans l'url
- Données facilement interceptables
- Données retrouvables dans un historique

Avantages

- Données retrouvables dans un historique
- Possibilité de partager un lien

Récupérer les données

Reprenons l'exemple :

```
www.monsite.com/page? clef1 = val1 & petit = gregory
```

en php :

```
$_Get['clef1'] //donne val1;
$_Get['petit'] //donne gregory;
```

> Attention aux valeurs alphanumériques (espaces,=,&,/,:,...)
>
> Les espaces peuvent êtres remplacés par %20
>
> JS : cfr encodeURIComponent(string)
>
> PHP : urlencode(e), $urldecode$(e)
>
> $_Get sont déjà urldécodés

```
$_GET['clef'];
    if(isset($_GET['nom']))
        $nom = $_GET['nom'];
```

POST

Les informations rentrées dans un formulaire html sont envoyées séparément et leur format n'est pas visible.

L'utilisateur remplit totalement / ou partiellement (aide de js) un formulaire HTML

-

- moins facile à mettre en place

+

- Pas de limite de Taille
- Laisse moins de traces(historique)

Exemple :

```
<form method="post" action="destination.php">
```

```
<form method="post" action="destination.php">

    <p>Nom : <input type="text" name="nom" /></p>
    <p>Sexe : <select name="sexe">
        <option value="m" selected>homme</option>
```

```
        <option value="f">femme</option>
    </select></p>

    <p>Catégorie d'âge :<br/>
        <input type="radio" name="age" value="1-20">Moins de 20</input>
<br/>
        <input type="radio" name="age" value="21-40" checked>Moins de
40</input> <br/>
        <input type="radio" name="age" value="41+">41 ou plus</input>
    </p>

    <input type="submit" value="Submit">
</form>

```

Réception du post

code du formulaire :

```
<input name="myNom" type="text" value="gregory"/>
```

code dans votre fichier de destination php

```
$_POST["myNom"] // donnera gregory
```

Javascript et Formulaires

```
var myFormulaire = document.forms[index];
var myElem = myFormulaire.elements;

myElem[2].checked = true; //coche un radio button
myElem[5].disabled= true // empêche l'activation d'un élément (ex:bouton pour
soumettre le formulaire)


```

On peut cacher certains champs et les révéler avec du JS pour les rendre interactifs

Le fait de cacher des champs permets aussi au serveur et au client de communiquer sans faire intervenir l'utilisateur

```
<input type="hidden" ...>
```

on peut aussi valider un formulaire via du javascript mais il est important de revérifier les données sur le serveur

```
function nomValide() {
    var nom = document.getElementById("nom").value;
    var msgErreur = document.getElementById("erreur");
    if (nom == null || nom == "") {
        msgErreur.innerHTML = "Nom obligatoire !";
        return false;
    }
    return true;
}

```

```
btnSubmid.onclick = nomValide; // bloque l'action si elle renvoie false
```

Types de formulaires

```
<p>Nom : <input type="text" name="nom"></input> </p>
<!-- $_POST['nom'] = valeur entrée dans le champ-->

<p>Sexe : <select name="sexe">
<option value="m" selected>homme</option>
<option value="f">femme</option>
</select></p>
<!--$_POST['sexe'] sera soit "m" soit "f"-->

<p>Catégorie d'âge :<br/>
<input type="radio" name="age" value="1-20">Moins de 20</input> <br/>
<input type="radio" name="age" value="21-40">Moins de 40</input> <br/>
<input type="radio" name="age" value="41+">41 ou plus</input></p>
<!--$_POST['age'] sera soit "1-20", soit "21-40", soit "41+"-->

<input type="checkbox" name="matin" value="matin">Matin</input><br/>
<input type="checkbox" name="soir" value="soir">Soir</input><br/>

<input type="checkbox" name="qd" value="matin">Matin</input><br/>
<input type="checkbox" name="qd" value="soir">Soir</input><br/>
<!--si les deux valeurs sont cochées, on ne garde que la dernière :
$_POST['qd'] = "soir"-->

<input type="checkbox" name="quand[]" value="matin">Matin</input><br/>
<input type="checkbox" name="quand[]" value="soir ">Soir</input><br/>
<!--
$_POST['quand'] = pas défini si aucune valeur cochée ;
                = ["matin"] ou ["soir"] si une seule valeur cochée ;
                = ["matin", "soir"] si les deux valeurs sont cochée  -->
```

Si rien n'a été coché, la variable n'existe pas dans le tableau associatif php.

Si 2 valeurs de la checkbox ont été cochés, on ne sélectionne que le dernier

Purification des données

Même si le Javascript à effectué une vérification des données, il se peut que quelqu'un de mal intentionné les ait modifiés / ou encore que elles ne soient pas correctes.

> voir les sections XSS (chap 11) pour les tentatives XSS (comme injection sql mais en PHP)

Solution 1 : éliminer les balises

```php
strip_tags($s); // supprime toutes les balises html
strip_tags($s,$sauf); // strip tous les tags html sauf ceux listés dans $sauf
$sauf = '<strong> <li>';
```

Solution 2 : Convertir et Les charactères spéciaux pour les rendre inoffensifs

```php
htmlspecialchars($s,mode); // annule les balises
htmlentities($s);
```

Autres fonctions utiles

```php
nl2br($s);                      //  remplace les retours à la ligne d'un string par des <br/>
mysqli_real_escape_string($s);  //  échape tous les caractères spéciaux en sql
trim($s);                       //  retire tous les blancs initiaux et finaux
```

Cookies

Les cookies sont des variables stockées sur le disque dur du client, elles peuvent être écrites/lues par le navigateur et lié à un site. Elles seront envoyés avec chaque requête au serveur du site.

Max 20/site

ATTENTION : La déclaration de cookies doit se trouver avant TOUT élément HTML d'une Page (oui, même un espace ou un retour à la ligne peut foutre la merde)

Côté js

Via :

```javascript
document.cookie; // l'objet

document.cookie = "nom=Grégory";
document.cookie =
"login=drhouse;expires=Fri, 31 Oct 2014 23:00:00 GMT";

document.cookie; // retourne maintenant"nom=Grégory;login=drhouse"

```

```javascript
function getCookie(nom) { // récupérer la valeur d'un cookie
    var cookies = document.cookie;
    var deb = cookies.indexOf("" + nom + "=");
    if (deb == -1)
        return null;
    deb = cookies.indexOf("=", deb) + 1;
    var fin = cookies.indexOf(";", deb);
    if (fin == -1)
        fin = cookies.length;
    return decodeURIComponent(cookies.substring(deb, fin));
}
```

Côté PHP

Accéder à un Cookie

```php
if (isset($_COOKIE['login'])) $userlogin = $_COOKIE['login'];
```

Définir / modifier un cookie

```php
setcookie('login', 'Grégory', time() + 60 * 60 * 24 * 7);
```

Détruire un cookie

Pour détruire un cookie, on mets sa date d'expiration dans le passé

```php
setcookie('login', '', time() - 2529200);
```

Sessions

Les sessions sont comme les cookies mais sauvegardés du côté serveur. Elles peuvent être utiles si le client interdit les cookies. Mais surtout car elles ne sont pas directement stockés chez le client ce qui les rends plus sécurisé.

Comment savoir à quel client correspond quel session ?

1. Les cookies (avec par exemple une variable PHPSESSID)
2. GET
3. Champ Caché

Créer une session

Il suffit de placer :

```php
session_start();
```

en ***début du document HTML*** envoyé pour créer la session.

Si aucune session n'est associée

- construction d'un id de session
- envoie de l'id au client

Si la session est en cours

On rends les informations de session lisibles

Créer / modifier une variable de session

```php
$_session['login'] = $loginUtilisateur;
```

Lire une variable de session

```php
$nomUtilisateur=$_session['nom'];
```

On peut aussi tester si la variable existe avec :

```php
isset($_session['login']);
```

Supprimer une variable de session

```php
unset($_session['nomVariable']);
```

Terminer une session

Pour terminer une session, on peut utiliser :

```php
session_destroy();
```

- détruit le fichier ou les données sont sauvegardées mais pas la variable de session.
- pour détruire tout, on peut faire :

  ```php
  $_session  = array();
  ```

exemple d'utilisation :

```php
<? php
    session_start();
    if (!isset($_SESSION['login']) ||
        $_SESSION['login'] != 'ok')
        {
        header('location: login.php'); //redirection
        exit();
    }
?>
```

En-tête

L'entête d'un fichier HTML est le début du fichier html (pas d'espace de départ ni de retour à la ligne)

```php
<?php
    //setcookie(Name,Value)
    //Ouverture de session session_start()
    //redirection avec redirection("location:newpage.php");
    ?>
```

AJAX

Ajax signifie Asynchronous javascript and XML

Ajax permets de charger de l'information supplémentaire sans devoir recharger une page complète.

Client

Js envoie une requête (qui peut contenir des informations). cette requête peut bloquer la page (syncho) ou ne pas attendre (Asynchrone).

Si il s'agit d'une requête asynchrone on doit préciser comment traiter la réponse une fois que celle-ci arrivera.

Serveur

Requête est traitée comme requête d'une page web

Comment faire ?

1. Créer objet XML HTTP REQUEST

```
req = new XMLHttpRequest()  //  pour créer une requête
req.readyState              //  indique l'état de la requête
req.responseText            //  la réponse reçue (en version texte) ou NULL
req.responseXML             //  la réponse reçue (arbre XML) ou NULL
req.status                  //  le statut de la réponse (code http : 404, 200
= ok)
```

États de la requête

- 0 (unsent) : pas encore envoyée
- 1 (opened) : connexion établie avec le serveur mais requête pas encore envoyée
- 2 (headers received) : le serveur a reçu la requête
- 3 (loading) : réponse en cours de réception
- 4 (done) : réponse entièrement reçue

Initialisation d'une requête

req open

```
req.open(mode,url);
```

> initialise une requête avec un mode (= "GET" ou "POST") et une url (qui peut contenir des données format GET)

```
req.open(mode, url, false)
```

> initialise une requête synchrone (le script Javascript est mis en pause en attendant la réponse)

```
req.open(mode, url, true)
```

> initialise une requête asynchrone (le script Javascript se poursuit en attendant la réponse).

événement associé

```
req.onreadystatechange = MaFct();
```

> action (fonction) à exécuter lors d'un changement d'état

> Pour associer une tâche à la réception de la réponse : tester que readyState == 4 et status == 200

Lancer une requête

Requête en GET

```
req.send() // requête en GET
```

Requête en Post

```
req.setRequestHeader("Content-type","application/x-www-form-urlencoded");
req.send("param=val&param=val…");
```

Obtenir un fichier

synchrone

```
var req = new XMLHttpRequest ();
req.open("GET", "http://monsite.be/liste.txt", false);
req.send(); // l'exécution s'interrompt
var cible = document.getElementById("divtexte");
cible.innerHTML = req.responseText;
```

async

```
var req = new XMLHttpRequest ();
req.open("GET", "http://monsite.be/liste.txt");
req.onreadystatechange = function () {
    if (req.readyState == 4 // terminé
        && req.status == 200) { // http OK
    var cible = document.getElementById("divtexte");
    cible.innerHTML = req.responseText;
}
req.send();
```

Ajax côté serveur

```
var req = new XMLHttpRequest ();
var url = "http://monsite.be/doc.php?";
url += "nom=" + nom + "&sexe=" + sexe;
req.open("GET", url);
req.onreadystatechange = function () {
    if (req.readyState == 4 && req.statue == 200) { // terminé && http OK
        var cible = document.getElementById("divtexte");
        cible.innerHTML = req.responseText;
    }
req.send();
```

requête multibrowser

```
var req;
if (window.XMLHttpRequest)//firefox
    req = new XMLHttpRequest();
else if (window.ArchiveXObject) //pour internet explorer
    try {
        req = new ArchiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            req = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {
            req = null;
        }
    }
else
    alert("Navigateur pas compatible XMLHttpRequest !");

```

1. Ne possède pas de mémoire ↩