
# Formation : BTS
# Cybersécurité, Informatique et réseaux, Électronique
# Informatique et Réseaux

---

## Web dynamique côté serveur
## Partie 1

# Sommaire

# Objectifs

***Objectif de ce TP :***

- Découvrir le langage php

***Activités pratiques :***

- Ecriture de page web en php, interprété par le serveur
- Lecture de données venant du client.
- Enregistrement et lecture dans un fichier des données du client,
- Ecriture et lecture de données dans une base

# 1ère page : écrire du texte avec une fonction php

Ecrire le code suivant, à l'aide d'un simple éditeur de texte :

```php
<?php echo ("Bonjour"); ?>
```

Vous placerez vos fichiers dans votre serveur web
Tester le fichier dans un navigateur web en tapant l'adresse l'adresse de votre site virtuel
Sélectionner le fichier correspondant à votre programme PHP. Que lisez-vous ?

Modifier le code précédent, en ajoutant *$txtBonjour* (ceci est une variable repérée par le signe **$** devant son nom) :

```php
<?php
$varTexte = "Bonjour";
echo $varTexte;
?>
```

Tester le fonctionnement du programme. Que lisez-vous ? Est-ce le même texte que précédemment ?

Modifier le code précédent, en ajoutant *une autre variable et un autre texte* :

```php
<?php
$varTexte = "Bonjour";
$varQuestion = "Comment allez-vous ?";
echo "Une suite de phrase : ".$varTexte." ".$varQuestion;
?>
```

Tester le fonctionnement du programme et constatez l'utilisation du . *(point)* dans l'instruction echo.

Modifier le code précédent, en ajoutant un *define* :

```php
<?php
define("TXT_CONSTANTE","<br/><br/>Voici un premier programmme en php");
$varTexte = "Bonjour";
$varQuestion = "Comment allez-vous ?";
echo "Une suite de phrase : ".$varTexte." ".$varQuestion.TXT_CONSTANTE;
?>
```

Tester le fonctionnement du programme et constatez l'utilisation du *define* dans l'instruction echo.

# 1ère page : écrire du texte de différentes façon en php

Modifier le code précédent et ajouter les lignes manquantes comme ci-dessous :

```php
<?php
define("TXTCONSTANTE","<br/><br/>Vous venez d'écrire votre premier programme en php");

$txtBonjour = "Bonjour à tous <br/>";
$txtQuestion = "Comment allez-vous ?";
$nbrePi = 3.14159;
$age = 19;
$tableauJour = array('Dimanche','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi');
$dateActuelle = getdate();
$numJourActuel = $dateActuelle['wday'];

echo "Une suite de phrases : $txtBonjour $txtQuestion".TXTCONSTANTE;
echo ' Ajout de nombre dans une phrase : Pi='.$nbrePi.' et votre age est probablement '.$age." ans... <br/>";
echo 'Aujourd\'hui nous sommes '.$tableauJour[$numJourActuel].' '.date("d M Y");
?>
```

Tester le programme. Constater l'affichage de l'ensemble des variables utilisées

Vous venez de voir l'imbrication de nombres de différents types (entier, à virgule), de contenu de tableaux ainsi que de date dans une chaîne de caractère.

Vous aurez probablement constatez le changement de guillemets " et ` utilisés pour définir des chaînes de caractères et l'utilisation du \ pour afficher le guillemet dans le mot aujourd'hui.
Avec les " il est possible de mettre les variables dans la chaîne de caractère (comme avec la première phrase), mais en pratique il est plus « propre » de mettre un point pour lier les variables au texte, car celles-ci ne sont pas noyées dans la phrase et sont plus facilement repérables.

Le langage PHP à la faculté de reconnaître les types des variables utilisées sans devoir procéder manuellement à des changements de types avant l'intégration dans une chaine de caractères ou à l'affichage à l'écran, d'où un gain de temps pour le développement d'applications.

# Prise de décision

Comment faire pour que le cadre du jour actuel se trouve encadré en blanc contrairement aux autres dates ?
Pour cela il faut utiliser des instructions conditionnelles permettant de faire un choix de décision.

Recopier le programme ci-dessous afin de tester des instructions de prise de décision.

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...

If($numJourActuel==0)
    echo 'Nous sommes Dimanche, bon week-end !';
else
    echo 'Nous sommes un jour de travail.';

?>
```

Il est possible d'augmenter le nombre de choix pour offrir plus de possibilité de décision. Le programme suivant illustre ce cas. Modifier le programme précédent pour obtenir la même chose que ci-dessous :

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...

If($numJourActuel==0)
    echo 'Nous sommes Dimanche, bon week-end !';
elseif($numJourActuel==1)
    echo 'Nous sommes Lundi, le début de la semaine';
elseif($numJourActuel==2)
    echo 'Nous sommes Mardi';
elseif($numJourActuel==3)
    echo 'Nous sommes Mercredi, le milieu de la semaine, encore 3 jours';
elseif($numJourActuel==4)
    echo 'Nous sommes Jeudi, plus que 2 jours';
elseif($numJourActuel==5)
    echo 'Nous sommes Vendredi, plus que 1 jour';
else
    echo 'Nous sommes Samedi, dernier jours de travail de la semaine !';

?>
```

# Prise de décsion

Heureusement pour ce genre de cas, une instruction plus concise existe : *switch (test) case*.

Recopier le programme ci-dessous afin de tester cette instruction.

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...

switch ($numJourActuel){
    case 0 :
        echo 'Nous sommes Dimanche, bon week-end !';
        break;
    case 1 :
        echo 'Nous sommes Lundi, le début de la semaine';
        break;
    case 2 :
        echo 'Nous sommes Mardi';
        break;
    case 3 :
        echo 'Nous sommes Mercredi, le milieu de la semaine, encore 3 jours';
        break;
    case 4 :
        echo 'Nous sommes Jeudi, plus que 2 jours';
        break;
    case 5 :
        echo 'Nous sommes Vendredi, plus que 1 jour';
        break;
    case 6 :
        echo 'Nous sommes Samedi, dernier jours de travail de la semaine !';
        break;
    default :
        echo 'Tiens, un nouveau jour existe ?';
    }
?>
```

# Prise de décision

Dans ce genre de situation, l'information étant retournée sous forme de tableau, pourquoi ne pas entrer les textes des jours également dans un tableau, il ne restera donc plus qu'un seul test à faire, ce qui réduira considérablement les lignes d'instructions.

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...
$tabNomJour = array('Dimanche, c\'est le week-end !','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi');

if($numJourActuel>=0 || $numJourActuel<=6)
    echo 'Aujourd\'hui, nous sommes : '.$tabNomJour[$numJourActuel];
else
    echo 'Tiens, un nouveau jour existe ?';
?>
```

Nous avons abordés les instructions conditionnelles *if (test) instructions; else instructions;*
ce qui en français donne : *si (test) instructions; sinon instructions;*

Les opérateurs de test sont <, >, <=, >=, ==, != (différent), avec également des tests d'algèbre de boole : ET (AND aussi écrit &&) et OU (OR aussi écrit ||).

Le IF de ci-dessus pourrait donc s'écrire en français :

```
SI ($numJourActuel >= 0 OU $numJourActuel <= 6) ALORS
     affiche 'Aujourd'hui, nous sommes : '+nom_du_jour_actuel
SINON
     affiche 'Tiens, un nouveau jour existe ?'
```

# Répétition et prise de décision

Lorsqu'il est nécessaire de faire des tâches de manière identique, il est plus simple d'utiliser des instructions de répétitions. Par exemple, si l'on souhaite afficher tous les jours de la semaine et de mettre en gras le jour actuel, il sera plus aisé d'utiliser des instructions de répétitions, plutôt que d'écrire les jours les uns à la suite des autres et de tester à chaque fois si c'est le jour actuel.

Les morceaux de programme ci-après vous illustre le fonctionnement des différentes instructions de bouclage utilisables.

Boucle Tant que : *while(test){ instructions }*

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...
$tabNomJour = array('Dimanche, c\'est le week-end !','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi');
$compteur=0;

while($compteur<7){ // boucle tant que
    if($compteur==$numJourActuel)
        echo '<b>'.$tabNomJour[$compteur].' c\'est aujourd\'hui !</b><br/>';
    else
        echo $tabNomJour[$compteur].'<br/>';
    $compteur++;
    }
?>
```

# Répétition et prise de décision

Boucle Faire … Tant que : *do{ instructions } while(test)*

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...
$tabNomJour = array('Dimanche, c\'est le week-end !','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi');
$compteur=0;

do{ // faire ... tant que
    if($compteur==$numJourActuel)
        echo '<b>'.$tabNomJour[$compteur].' c\'est aujourd\'hui !</b><br/>';
    else
        echo $tabNomJour[$compteur].'<br/>';
    $compteur++;
}while($compteur<7);
?>
```

Boucle Pour : *for (valeur_initiale;test;incrément) { instructions }*

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...
$tabNomJour = array('Dimanche, c\'est le week-end !','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi');
$compteur=0;

for($compteur=0;$compteur<7;$compteur++){ // boucle pour
    if($compteur==$numJourActuel)
        echo '<b>'.$tabNomJour[$compteur].' c\'est aujourd\'hui !</b><br/>';
    else
        echo $tabNomJour[$compteur].'<br/>';
    }
?>
```

# Fonction ou comment réutiliser du code déjà écrit

Le principe de la programmation étant de rédiger des instructions qu'une machine doit exécuter, il est nécessaire de développer les lignes d'instructions rapidement et lisiblement. Pourquoi alors devoir à chaque fois écrire des lignes de codes déjà écrites auparavant ? Cela est une perte de temps considérable et un alourdissement du programme source. Pour éviter cela, les fonctions permettent de pouvoir réutiliser des morceaux de codes dans des programmes, tout en allégeant la lecture globale du code source.

Vous avez utilisé déjà une fonction dans les programmes précédent : *getdate().* Cette fonction retourne certaines informations relatives au jour actuel, si aucun paramètre ne lui est donné entre les parenthèses.

Ecrivons une fonction qui affiche les jours de la semaine et met en gras celui donné entre parenthèse :

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...

function fctJourSemaine($valJourActuel){
     $tabNomJour = array('Dim','Lun','Mar','Mer','Jeu','Ven','Sam');
     $ecritTxt='';
     for($compteur=0;$compteur<7;$compteur++){ // boucle pour
          if($compteur==$valJourActuel)
               echo '<b>'.$tabNomJour[$compteur].'</b> ';
          else
               echo $tabNomJour[$compteur].' ';
     }
     echo '<br/>';
}

fctJourSemaine($numJourActuel);
?>
```

# Fonction ou comment réutiliser du code déjà écrit

La fonction précédente n'est pas optimisée, car elle génère un flux de données constant entre le serveur et le pc client de l'internaute à chaque instruction echo.

Pour réduire cette quantité de transfert, nous allons enregistrer les données à afficher dans une variable que nous appellerons *$ecritTxt*, puis nous demanderons à la fin de la fonction de retourner la valeur de cette variable que nous afficherons ensuite.

Modifier la fonction précédente pour arriver à l'écriture de celle-ci-dessous :

```php
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...

function fctJourSemaine($valJourActuel){
     $tabNomJour = array('Dim','Lun','Mar','Mer','Jeu','Ven','Sam');
     $ecritTxt='';
     for($compteur=0;$compteur<7;$compteur++){ // boucle pour
          if($compteur==$valJourActuel)
               $ecritTxt.='<b>'.$tabNomJour[$compteur].'</b> ';
          else
               $ecritTxt.=$tabNomJour[$compteur].' ';
     }
     $ecritTxt.='<br/>';
     return $ecritTxt;
}

fctJourSemaine($numJourActuel);
?>
```

Oui c'est bien, mais où est l'intérêt de la fonction puisque l'on ne s'en sert qu'une fois ici ?
Et bien pour répondre à cette question, il suffit de suivre la diapositive suivante.

# Fonction ou comment réutiliser du code déjà écrit

Modifier le programme comme ci-dessous pour utiliser plusieurs fois la fonction *fctJourSemaine()* :

```
<?php
$dateActuelle = getdate(); // recuperation de la date actuelle sous forme de tableau
$numJourActuel = $dateActuelle['wday']; // recuperation du numéro du jour actuel 0=Dimanche, 1=Lundi,...
$numMoisActuel = $dateActuelle['mon']; // recuperation du numéro du mois actuel 1=Janvier, 2=Février,...
$carSemaine = '<html><head></head><body><table cellspacing="0" cellpadding="5px" border="1px"><tr>';

function fctJourSemaine($valJourActuel){
      $tabNomJour = array('Dim','Lun','Mar','Mer','Jeu','Ven','Sam');
      $ecritTxt='';
      for($compteur=0;$compteur<7;$compteur++){ // boucle pour
            if($compteur==$valJourActuel)
                  $ecritTxt.='<b>'.$tabNomJour[$compteur].'</b> ';
            else
                  $ecritTxt.=$tabNomJour[$compteur].' ';
            }
      $ecritTxt.='<br/>';
      return $ecritTxt;
      }

for($mois=1;$mois<=12;$mois++){
      if($mois==5 || $mois==9)
            $carSemaine .= '</tr><tr>';
      if($mois==$numMoisActuel)
            $carSemaine .= '<td>'.fctJourSemaine($numJourActuel).'</td>';
      else
            $carSemaine .= '<td>'.fctJourSemaine(7).'</td>';
      }

$carSemaine .= '</tr></table></body></html>';
echo $carSemaine;
?>
```

Vous venez de voir l'intérêt de l'écriture des fonctions dans les programmes ainsi que leur simplicité dans leur utilisation. A travers cet exemple vous venez également d'avoir eu un aperçu du principe de l'écriture du calendrier présenté en début de diaporama.

Pour respecter le cahier des charges il est nécessaire de récupérer des informations du client. Nous allons maintenant apprendre à traiter des données venant d'un formulaire (celui-ci sera écrit en html et abordé ultérieurement).

# Transmission de données au serveur : méthode GET ou POST

Pour transmettre des données, 2 méthodes sont possibles : la méthode GET et la méthode POST.

La méthode **GET** récupère des informations passées dans l'adresse de la page à contacter.

Vous allez rédiger 1 page en php afin de tester le passage de paramètres dans le contact de cette page dans la barre d'adresse.

Recopier le programme ci-dessous et enregistrer le sous un nouveau nom  par exemple prog3.php :

```php
<?php
if(isset($_GET['valeur'])) // test de l'existence de la variable $_GET['valeur']
    echo 'Valeur passée : '.$_GET['valeur'];  // si elle existe
else
    echo 'Pas de valeur passée dans l\'adresse de la page';  //si elle n'existe pas
?>
```

Appeler la page *prog3.php* et constatez le résultat.
Appeler la page *prog3.php?valeur=bonjour* et constatez le résultat.
Appeler la page *prog3.php?val=bonjour* et constatez le résultat.

Modifier le programme précédent comme ci-dessous :

```php
<?php
if(isset($_GET['nom']) && isset($_GET['prenom']) && isset($_GET['age']))
    echo 'Vous êtes : '.$_GET['prenom'].' '.$_GET['nom'].' et avez '.$_GET['age']. ' ans';
else
    echo 'Il manque un ou plusieurs paramètres';
?>
```

Recopier le programme ci-dessous et enregistrer le sous prog3.php.

Appeler la page *prog3.php* et constatez le résultat.
Appeler la page *prog3.php?/prenom=votre_prenom&nom=votre_nom&age=votre_age* et constatez le résultat.

# Transmission de données au serveur : méthode GET ou POST

Vous venez de constatez le fonctionnement de la méthode **GET** pour passer 1 ou plusieurs paramètres à la page. L'inconvénient de cette méthode vient du fait que les paramètres sont dans l'adresse de la page et qu'ils sont en clair, d'où un risque de sécurité important pour votre page. L'autre inconvénient est qu'une adresse web ne doit pas dépasser 256 caractères, tous confondus., ce qui peut vite arrivé si l'on ne fait pas attention au nombre de paramètres que l'on passe !

L'autre méthode **POST** permet de passer des paramètres à une page sans écrire ceux-ci dans l'adresse de la page, mais exige pour ce faire un formulaire dans la page web émettrice. Grâce à cette méthode il sera possible d'envoyer autant de paramètres souhaités, mais ce n'est pas pour autant que les données transmisses sont pas sécurisées

Modifier le programme précédent comme ci-dessous en utilisant la méthode **POST** :

```php
<?php
echo '<html><head></head><body>';

if(isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['age']))
    echo 'Vous êtes : '.$_POST['prenom'].' '.$_POST['nom'].' et avez '.$_POST['age']. ' ans';
else
    echo 'Il manque un ou plusieurs paramètres <br/>'."\n";

echo '<form method="post" name="formulaire">
<input type="text" name="nom" value="" placeholder="Votre nom"/>
<input type="text" name="prenom" value="" placeholder="Votre prénom"/>
<input type="number" name="age" value="" placeholder="Votre age"/>
<input type="submit" value="envoyer"/>
</form>
</body></html>';
```

Appeler la page *prog3.php* et constatez le résultat.
Cliquer ensuite sur le bouton *envoyer*
Affichez la source de la page pour voir le contenu de votre page affichée au client.

# Transmission de données au serveur : méthode GET ou POST

Nous avions parlé de faille de sécurité précédemment lors de passage de valeurs dans les adresses ou dans les formulaires. Cette faille appelée faille XSS (pour **cross-site scripting**) est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages pour le faire exécuter à vos visiteurs.

```
<?php
echo '<html><head></head><body>';

if(isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['age']))
    echo 'Vous êtes : '.$_POST['prenom'].' '.$_POST['nom'].' et avez '.$_POST['age']. ' ans';
else
    echo 'Il manque un ou plusieurs paramètres <br/>'."\n";

echo '<form method="post" name="formulaire">
<input type="text" name="nom" value="" placeholder="Votre nom"/>
<input type="text" name="prenom" value="" placeholder="Votre prénom"/>
<input type="number" name="age" value="" placeholder="Votre age"/>
<input type="submit" value="envoyer"/>
</form>
</body></html>';
```

Voici un exemple d'utilisation de la faille. Dans le formulaire entrer :
**jean** pour le prénom,
**<script type="text/javascript">alert("Bonjour");</script>** pour le nom,
**15** pour l'age

Cliquer sur le bouton submit puis constatez le résultat

# Transmission de données au serveur : méthode GET ou POST

Pour éviter cela, il faut échapper les balises html dans les variables passées dans les adresses ou dans les formulaires. Ainsi le code précédent devrait être écrit comme cela :

```
<?php
echo '<html><head></head><body>';

if(isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['age']))
    echo 'Vous êtes : '.htmlspecialchars($_POST['prenom']).' '.htmlspecialchars($_POST['nom']).' et avez
'.htmlspecialchars($_POST['age']). ' ans';
else
    echo 'Il manque un ou plusieurs paramètres <br/>'."\n";

echo '<form method="post" name="formulaire">
<input type="text" name="nom" value="" placeholder="Votre nom"/>
<input type="text" name="prenom" value="" placeholder="Votre prénom"/>
<input type="number" name="age" value="" placeholder="Votre age"/>
<input type="submit" value="envoyer"/>
</form>
</body></html>';
```

Dans le formulaire entrer à nouveau :
**jean** pour le prénom,
**<script type="text/javascript">alert("Bonjour");</script>** pour le nom,
**15** pour l'age

Cliquer sur le bouton submit. Que constatez-vous ?

Affichez la source de votre page et regarder le contenu.

# Stockage locale de données

Récupérer des données d'un client c'est bien, mais les stocker pour pouvoir les traiter ultérieurement c'est mieux. Nous allons aborder ici l'enregistrement ou la lecture de données dans un fichier.
Modifier le fichier précédent avec les lignes manquantes, comme ci-dessous :

```
<?php
$nomFichier = 'fichierReservation.txt';

echo '<html><head></head><body>';

if(isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['age'])) {
    echo 'Vous êtes : '.htmlspecialchars($_POST['prenom']).' '.htmlspecialchars($_POST['nom']).' et avez
'.htmlspecialchars($_POST['age']).' ans';

    if(!$fluxFichier = fopen($nomFichier,"a"))
        echo "Impossible d'ouvrir le fichier ($nomFichier)";
    else{
        fwrite($fluxFichier, htmlspecialchars($_POST['nom']).' '.htmlspecialchars($_POST['prenom']).'
'.htmlspecialchars($_POST['age'])."\n");
        fclose($fluxFichier);
    }
}
else
    echo 'Il manque un ou plusieurs paramètres <br/>'."\n";


echo '<form method="post" name="formulaire">
<input type="text" name="nom" value="" placeholder="Votre nom"/>
<input type="text" name="prenom" value="" placeholder="Votre prénom"/>
<input type="number" name="age" value="" placeholder="Votre age"/>
<input type="submit" value="envoyer"/>
</form>
</body></html>';
```

Testez le résultat. Que constatez-vous ? Rechercher le fichier dans l'arborescence et éditez le. Que contient-il ?
Modifier les droits en écriture sur le répertoire de travail et recommencer le test.

**fopen** (PHP 4, PHP 5) - Ouvre un fichier ou une URL

**Description**

```
resource fopen(string $filename, string $mode [, bool $use_include_path=false [, resource $context ]])
```

**fopen()** crée une ressource nommée, spécifiée par le paramètre filename, sous la forme d'un flux.

**Liste de paramètres**

**filename**

Si filename est de la forme "protocole://", filename est supposé être une URL, et PHP va rechercher un gestionnaire de protocole adapté pour lire ce fichier. Si aucun gestionnaire pour ce protocole n'est disponible, PHP va émettre une alerte qui vous permettra de savoir que vous avez des problèmes dans votre script, et il tentera d'exploiter filename comme un fichier classique.
Si PHP décide que le fichier filename est un fichier local, il va essayer d'ouvrir un flux avec ce fichier. Le fichier doit être accessible à PHP. Il vous faut donc vous assurer que vous avez les droits d'accès à ce fichier. Si vous activez le safe mode, ou la directive open_basedir, d'autres conditions peuvent aussi s'appliquer.

**mode**

Le paramètre mode spécifie le type d'accès désiré au flux. Il peut prendre les valeurs suivantes :

*'r'* Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.

*'r+'* Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.

*'w'* Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

*'w+'* Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.

*'a'* Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

*'a+'* Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

*'x'* Crée et ouvre le fichier en écriture seulement ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, **fopen()** va échouer, en retournant **FALSE** et en générant une erreur de niveau **E_WARNING**. Si le fichier n'existe pas, **fopen()** tente de le créer.

*'x+'* Crée et ouvre le fichier pour lecture et écriture; le comportement est le même que pour *'x'*.

**fwrite** (PHP 4, PHP 5) - Écrit un fichier en mode binaire

**Description**

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

**fwrite()** écrit le contenu de la chaîne string dans le fichier pointé par handle.

**Liste de paramètres**

**handle** : Un pointeur de système de fichiers de type resource qui est habituellement créé en utilisant la fonction fopen().
**string** : La chaîne à écrire.
**length** : Si la longueur length est fournie, l'écriture s'arrêtera après length octets, ou à la fin de la chaîne (le premier des deux).
Notez que si length est fourni, alors l'option de configuration magic_quotes_runtime sera ignorée, et les slash seront conservés.

**Valeurs de retour**

**fwrite()** retourne le nombre d'octets écrits, ou **FALSE** si une erreur survient.

# Stockage locale de données

Maintenant que nous arrivons à enregistrer des données dans un fichier, utilisons une autre fonction : la lecture de fichier.
Compléter le fichier précédent avec les lignes manquantes, comme ci-dessous :

```
<?php
...
...
echo 'Il manque un ou plusieurs paramètres <br/>'."\n";


if(!$fluxFichier = fopen($nomFichier,"r"))
  echo "Impossible d'ouvrir le fichier ($nomFichier)";
else{
  echo "Contenu du fichier :<br/>\n";
  while (($buffer = fgets($fluxFichier, 1024)) !== false) {
    echo $buffer;
    }
   fclose($fluxFichier);
}


echo '<form method="post" name="formulaire">
...
...
?>
```

Testez le résultat. Que constatez-vous ?

**fgets** - Récupère la ligne courante sur laquelle se trouve le pointeur du fichier

**Description**

```
string fgets ( resource $handle [, int $length ] )
```

Récupère la ligne courante sur laquelle se trouve le pointeur du fichier.

**Liste de paramètres**

**handle** : Le pointeur de fichier doit être valide et pointer sur un fichier ouvert avec succès par fopen() ou fsockopen() (et pas encore fermé par fclose()).

**length** : Lit jusqu'à la taille length - 1 octet depuis le pointeur de fichier handle, ou bien la fin du fichier, ou une nouvelle ligne (qui est inclue dans la valeur retournée), ou encore un EOF (celui qui arrive en premier). Si aucune longueur n'est fournie, la fonction lira le flux jusqu'à la fin de la ligne.

Testez maintenant les différents modes possibles de la fonction fwrite et constatez le fonctionnement.