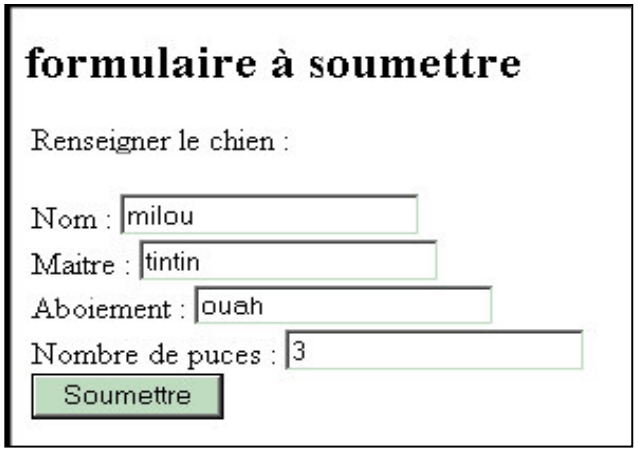
PHP et les formulaires HTML

Intérêt d'un formulaire

- Les formulaires sont l'élément essentiel qui permet l'interactivité entre un site et ses visiteurs. Ils constituent pour cette raison la base de la création de sites dynamiques.
- Tout échange entre visiteur et serveur passe par un formulaire, dans lequel l'utilisateur peut saisir textes ou mots de passe, opérer des choix grâce à des boutons radio, des cases à cocher ou des listes de selection, voire envoyer ses propres fichiers depuis le poste client.
- Il est donc important d'en maîtriser la création à la fois avec HTML, pour obtenir des formulaires présentables, et avec PHP, pour gérer les informations fournies par le formulaire au script côté serveur.

```
<html>
<head><title>formulaire à soumettre</title>/head>
<body>
<h2 align=center> formulaire à soumettre</h2>
<p>
Renseigner le chien :<br>
<form method="post" action="recup_valeurs.php">
Nom : <input type="text" name="nom"><br>
Maitre : <input type="text" name="maitre"><br>
Aboiement : <input type="text" name="aboiement"><br>
Nombre de puces : <input type="text" name="nombrePuces"><br>
<input type="submit" name="envoi" value="Soumettre">
</form>
</p>
</body>
</html>
```



Ecrire un formulaire HTML

<form method="post" action="submit1.php">

- **<FORM METHOD=… ACTION=…> ... </FORM>**, une balise qui permet de regrouper plusieurs éléments de formulaire (boutons,champs de saisie,...) et dont les attributs les plus courants sont :

  - **METHOD** indique la méthode de transmission HTTP des données saisies dans le formulaire :
    - « **POST** » est la valeur qui correspond à un envoi de données stockées dans le corps de la requête (les données sont cachées)
    - « **GET** » correspond à un envoi des données **visibles dans l'URL** (séparées de l'adresse du script par un point d'interrogation) : **à éviter pour des raisons évidentes de sécurité.**
    - **si cet attribut est omis, il vaut "get" par défaut.**

  - **ACTION** permet d'indiquer l'URL qui va recevoir les informations entrées dans le formulaire, lorsque l'on cliquera sur le bouton de validation. Plus précisément, l'URL est l'adresse d'un programme (un script) qui va récupérer les données et les traiter. Si le champ ACTION est absent, l'URL sera celle du document courant.

Ecrire un formulaire HTML

- La balise form peut avoir d'autres attributs optionnels. On peut citer par exemple :
  - enctype : détermine comment les données envoyées par le formulaire (du texte, des fichiers joints) vont être encodées
    - *"application/x-www-form-urlencoded*" c'est la valeur par défaut. C'est celle qu'il faut utiliser dans la plupart des cas.
    - "*multipart/form-data*" C'est la valeur à utiliser pour envoyer des fichiers.

Ecrire un formulaire HTML

<input type="submit" name="envoi" value="Soumettre">

- La balise **input** définit des champs d'information. L'attribut **name** désigne l'élément, l'attribut **value** désigne éventuellement une valeur initiale, et l'attribut **type** définit des objets:
  - **text** : petit champ pour que l'utilisateur saisisse un nombre ou une phrase
  - **button** : pour cliquer dessus
  - **checkbox** : cases à cocher (information on/off)
  - **radio** des "boutons radio" (information on/off mais un seul peut être coché à la fois)
    - les checkbox et radios n'envoient rien (même pas l'attribut name) s'ils ne sont pas cochés
  - **hidden** : champ caché pour transmettre des informations vers le serveur, sans que l'utilisateur ne le sache : très utile pour gérer un caddie ...
  - **password** : champ texte mais masquant le texte tapé
  - **reset** pour remettre le formulaire dans son état initial
  - **submit** : action indispensable lorsque le formulaire est **utilisé pour envoyer de l'information depuis l'utilisateur vers le serveur** : "envoie" les données du formulaire vers le serveur
  - **image** : même principe que submit mais c'est une image au lieu d'un bouton.
  - **file** : Affiche un bouton permettant de sélectionner un fichier de l'ordinateur ; la plupart des navigateurs l'accompagnent d'une case de texte contenant le chemin d'accès au fichier

Ecrire un formulaire HTML

- Deux principales exceptions à la balise <input/> pour insérer des éléments dans un formulaire
  - **textarea** pour une zone de texte multi-lignes : <textarea> </textarea>
  - **select** pour une liste de sélection :
    - <select name="toto">
    - <option value="truc">choix1</option>
    - <option value="bidule">choix2</option>
    - </select>

Récupérer les valeurs d'un formulaire avec PHP

- 1er cas : les valeurs uniques.
  - Les valeurs uniques proviennent des champs de formulaire dans lesquels l'utilisateur peut entrer qu'une valeur, un texte par exemple, ou ne peut faire qu'un seul choix (bouton radio, liste de sélection à choix unique).
  - Depuis PHP 4.1, ces valeurs sont contenues sur le serveur dans des tableaux associatifs superglobaux appelés $_POST ou $_GET selon la méthode choisie.
  - Les clés de ces tableaux sont les noms associés aux champs par l'attribut *name*

```
<body>
<form action= "<?php echo $_SERVER['PHP_SELF'] ?>" method="post" enctype="application/x-www-form-urlencoded">
 <fieldset>
  <legend><b>Infos</b></legend>
<div>

Nom : <input type="text" name="nom" size="40" />
<br />
Débutant : <input type="radio" name="niveau" value="débutant" />
Initié :   <input type="radio" name="niveau" value="initié" /><br />
<input type="reset" value="Effacer" />
<input type="submit" value="Envoyer" />
</div>
</fieldset>
</form>
<?php
if(isset($_POST["nom"]) && isset($_POST["niveau"]))
{
  echo "<h2> Bonjour ". $_POST["nom"]. " vous êtes ".$_POST["niveau"]." en PHP</h2>";
}
?>
</body>
```

Récupérer les valeurs d'un formulaire avec PHP

- 2ième cas : les valeurs multiples.
  - Certains champs de formulaire peuvent permettre aux visiteurs de saisir plusieurs valeurs sous un même nom de composant.
  - Cela peut concerner un groupe de cases à cocher ayant le même attribut name, par exemple, dont il est possible de cocher une ou plusieurs cases simultanément.
  - Ce peut également être le cas d'une liste de sélection ayant toujours un nom unique mais dans laquelle l'attribut multiple= "multiple" est défini.
  - Dans tous les cas, ce n'est pas une valeur scalaire mais un tableau qui est récupéré côté serveur. Il faut pour cela faire suivre le nom du composant de crochets, comme pour créer une variable array.

Récupérer les valeurs d'un formulaire avec PHP

- Dans l'exemple suivant :

```
Bleu:<input type=”checkbox” name=”choix[ ]” value=”bleu” />
Blanc:<input type=”checkbox” name=”choix[ ]” value=”blanc” />
```

L'utilisateur peut cocher les deux cases simultanément.
Le programmeur récupère ces valeurs dans les variables suivantes :

```
$_POST["choix"][0] qui contient la valeur "bleu"
$_POST["choix"][1] qui contient la valeur "blanc"
```

Récupérer les valeurs d'un formulaire avec PHP

- 3ième cas : les fichiers.
  - L'inclusion d'un élément XHTML <*input type="file"* /> dans un formulaire crée une situation particulière. Il ne s'agit plus de transmettre une ou plusieurs valeurs scalaires au serveur mais l'intégralité d'un fichier, lequel peut avoir une taille importante et un type quelconque. Ce fichier doit évidemment être présent sur l'ordinateur du visiteur.
  - Par rapport au transfert de données, le transfert de fichiers présente un problème de sécurité pour votre site puisque des fichiers vont être écrits et éventuellement exécutés sur le serveur.
  - Supposez qu'un utilisateur mal intentionné transfère un fichier nommé *index.php* ou *index.html* sur le serveur. Ce fichier est exécuté automatiquement à l'appel de l'URL de votre nom de domaine. Pour éviter tout problème, il est prudent de prévoir une vérification de l'extension du fichier préalablement à la définition des extensions autorisées lors de la création du formulaire. Pour cela, il faudra utiliser l'attribut *accept* de l'élément <input /> avec pour valeur associé le type de fichier accepté (e.g., image/gif, text/html, etc...).

Récupérer les valeurs d'un formulaire avec PHP

- 3ième cas : les fichiers (suite).
  - L'élément <*form*> doit nécessairement avoir l'attribut *method* à la valeur *post* et l'attribut enctype à la valeur *multipart/form-data*.
  - Depuis la version PHP 4.1, on dispose du tableau associatif multidimensionnel *$_FILES* qui contient les informations nécessaires au traitement du fichier transféré.
  - En supposant que le nom de l'élément <*input type=”file” … /*> est *name="MonFichier"*, on peut notamment lire :
    - *$_FILES["MonFichier"]["name"]* : contient le nom qu'avait le fichier sur le poste client
    - *$_FILES["MonFichier"]["type"]* : contient le type MIME initial du fichier et permet un contrôle et une censure éventuelle du fichier transféré
    - *$_FILES["MonFichier"]["size"]* : donne la taille réelle en octet du fichier transféré
    - *$_FILES["MonFichier"]["tmp_name"]* : donne le nom temporaire que le serveur attribue automatiquement au fichier en l'enregistrant dans le répertoire tampon
  - Pour procéder à l'enregistrement et au renommage du fichier sur le serveur :
    - *Boolean move_uploaded_file(string "fichtmp", string "fichfinal")*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Transfert de fichier</title>
</head>
<body>
<form action="form.php" method="post" enctype="multipart/form-data" >
<fieldset>
<legend><b>Transfert de fichier</b></legend>
<table>
<tbody>
<tr>
<th>Fichier</th>
<td> <input type="file" name="MonFichier" accept="image/gif" size="50"/></td>
</tr>
<tr>
<th>Clic!</th>
<td> <input type="submit" value="Envoi" /></td>
</tr>
</tbody>
</table>
</fieldset>
</form>
```

```
<!-- suite du fichier form.php : Code PHP   -->
<?php
if(isset($_FILES['MonFichier']))
{
//Enregistrement et renommage du fichier
$result=move_uploaded_file($_FILES["MonFichier"]
["tmp_name"],"imagephp.gif");
if($result==TRUE)
{echo "<hr /><big>Le  transfert est réalisé !</big>";}
else
{echo "<hr /> Erreur de transfert"}
}
?>
</body>
</html>
```

Variables superglobales

Variables superglobales prédéfinies

- PHP dispose d'un grand nombre de variables prédéfinies, qui contiennent des informations à la fois sur le serveur et sur toutes les données qui peuvent transiter entre le poste client et le serveur, comme les valeurs saisies dans un formulaires, les cookies ou les sessions.
- Depuis PHP 4.1, ces variables se présentent sous la forme de tableaux, accessibles en tout point de n'importe quel script. On appelle ces tableaux *superglobaux*

Variables superglobales prédéfinies

- $GLOBALS
  - Contient le nom et la valeur de toutes les variables globales du script. Les noms des variables sont les clés du tableau.
  - $GLOBAL["mavar"] récupère la valeur de la variable $mavar en dehors de sa zone de visibilité (dans les fonctions par exemple)
- $_SERVER
  - Contient les informations liées au serveur web, tel que l'adresse IP du serveur ou le nom du script en cours d'exécution.

Variables d'environnement

- Variables superglobales prédéfinies accessibles dans tous les scripts via $_SERVER

$_SERVER['HTTP_ACCEPT_LANGUAGE'] Langage accepté par le navigateur

$_SERVER['HTTP_HOST'] Nom de domaine du serveur

$_SERVER['HTTP_USER_AGENT'] Type de navigateur

$_SERVER['REMOTE_ADDR'] Adresse IP du client

$_SERVER['SERVER_ADDR'] Adresse IP du serveur

$_SERVER['PHP_SELF'] Nom du script en cours d'exécution

...

Variables superglobales prédéfinies

- $_GET
  - Contient le nom et la valeur des données issues d'un formulaire envoyé par la méthode GET. Les noms des champs du formulaire sont les clés de ce tableau.
- $_POST
  - Contient le nom et la valeur des données issues d'un formulaire envoyé par la méthode POST. Les noms des champs du formulaire sont les clés de ce tableau
- $_COOKIE
  - Contient le nom et la valeur des cookies enregistrés sur le poste client. Les noms des cookies sont les clés de ce tableau.
- $_FILES
  - Contient le noms des fichiers téléchargés à partir du poste client.

Exemple

```
<?php
$bouton = $_POST['send'];
if(!empty($bouton))
{
  $nom = trim($_POST['nom']);
  $prenom = trim($_POST['prenom']);
  if(!empty($nom) && !empty($prenom))
  {
     echo 'Bonjour, '.$prenom.' '.$nom;
   }
else
   {
     echo 'vous n\'avez pas rempli tous les champs';
    }
}
?>
```

dans le tableau associatif $_POST, une entrée à été créée avec le nom de chaque variable du formulaire, ici deux champs nom et prénom.

Le formulaire contient :

```
<input type="submit"
value="envoyer"
name="send">
```

Remarque : !empty permet de vérifier que :

- la variable n'est pas vide
- la variable ne contient ni 0, NULL, “0” ou FALSE

Autre exemple

```
<?php
if(isset($_POST)) {
  foreach($_POST as $key=>$val) {
     echo $key.'=>'.$val.'<br/>';
   }
}
else {
   echo 'le formulaire n\'a pas été envoyé';
}
?>
```

Exemple : récupération d'un fichier

$_FILES['variable']['name'] Le nom original du fichier provenant de la machine de l'utilisateur

$_FILES['variable']['type'] Le type mime du fichier

$_FILES['variable']['size'] Le taille du fichier en bytes

$_FILES['variable']['tmp_name'] Le nom temporaire du fichier stocké sur le serveur

$_FILES['variable']['error'] Le code erreur associé à l'upload (attention cette option à été ajoutée en PHP 4.2.0)

Formulaire :

```
<form enctype="multipart/form-data" action="exemple2.php" method="post">
<input name="fichier" type="file">
<input type="submit" value="send" name="bouton">
</form>
```

Exemple : récupération d'un fichier

```
<?php
$bouton = $_POST['bouton'];
if(!empty($bouton)) {
  $fichier = $_FILES['fichier']['name'];       $size = $_FILES['fichier']['size'];
  $tmp = $_FILES['fichier']['tmp_name'];   $type = $_FILES['fichier']['type'];
  $error = $_FILES['fichier']['error'];          $max = 10000;
}
if(isset($fichier)) {
  if($size <= $max) {
    echo 'Nom d\'origine =>'.$fichier.'<br />';  echo 'Taille =>'.$size.'<br />';
    echo 'Nom sur le serveur =>'.$tmp.'<br />'; echo 'Type de fichier =>'.$type.'<br />';
    echo 'Code erreur =>'.$error.'<br />';
  }
  else {
     echo 'Le fichier est trop volumineux';
  }
}
else {
   echo 'aucun fichier envoyé';
}
}
?>
```

Variables superglobales prédéfinies

- $_ENV
  - Contient le nom et la valeur des variables d'environnement qui sont changeantes selon les serveurs.
- $_REQUEST
  - Contient l'ensemble des variables superglobales $_GET, $_POST, $_COOKIE et $_FILES.
- $_SESSION
  - Contient l'ensemble des noms des variables de sessions et leurs valeurs (voir chapitre ultérieur sur les sessions).

Fichiers

Fichiers

- Comme nous l'avons vu précédemment, les formulaires sont un outil privilégié pour recueillir les informations en provenance des visiteurs du site et que ces informations étaient récupérées dans des variables créées côté serveur.
- Le problème avec ces données est qu'elles sont volatiles. Sitôt le script terminé, elles sont perdues. Quand elles présentent un intérêt qui va au-delà de la simple connexion en cours, il faut envisager les moyens de les réutiliser plus tard.
- La solution la plus simple à ce problème consiste à enregistrer les données sur le serveur dans un fichier, généralement de type texte. Ce type de stockage est principalement utilisé pour des quantités de données de taille modeste et quand il n'est pas nécessaire d'effectuer par la suite des recherches complexes parmi elles.
- Dans le cas contraire, il faut avoir recours à une base de données spécialisée, comme MySQL, qui permet d'effectuer des recherches “pointues”.

Création d'un fichier

- **Pour créer un fichier vide, utilisez la fonction touch**
  - **Boolean touch (string "NomFichier");**

```
If (!file_exists("toto.txt"))
{
touch("toto.txt") ;
}
```

Ouverture et fermeture d'un fichier

- Ouvrir un fichier :
  - Ressource fopen (string $NomFichier, string mode)
- Fermeture d'un fichier
  - Boolean fclose ($idfile)

Paramètres de la fonction fopen()

fopen (nom_fichier, mode)
ouvre un fichier selon le mode spécifié
retourne un objet de type ressource sinon False
le paramètre mode (de type string) peut prendre les valeurs suivantes :

- 'r' lecture seule, la lecture commence au début du fichier
- 'r+' en lecture et écriture, ces opérations commencent en début de fichier.
- 'w' en écriture seul : si le fichier existe, son contenu est écrasé sinon il est créé. L'écriture commence au début du fichier
- 'w+' en lecture et écriture : si le fichier existe, son contenu est écrasé sinon il est créé. Ces opérations commencent en début de fichier
- 'a' en écriture seul; l'écriture commence à la fin du fichier
- 'a+' en lecture et écriture; ces opérations commencent à la fin du fichier

Attention : Pour les modes "w" et "w+", le contenu antérieur est effacé. Il faut donc prendre des précautions avant d'utiliser ce mode d'accès.

Identifiant de fichier

- Depuis PHP 4, la fonction fopen() retourne un identifiant de fichier de type ressource (il était de type integer dans PHP3), qui doit être utilisé comme premier paramètre de la plupart des fonctions de manipulation des fichiers.
- Il est donc important de récupérer cette valeur dans une variable pour pouvoir l'utiliser dans les autres opérations d'accès au même fichier.

```
$id_file = fopen(''MonFichier.txt'', ''a'');
if(!$id_file) echo ''Problème accès fichier'';
```

Verrouillage des fichiers

- Quand un script utilise une base de données telle que MySQL pour y stocker des informations, l'ampleur du trafic sur le site n'a pas une importance primordiale. Plusieurs utilisateurs peuvent accéder en même temps à la même table et y effectuer des opérations de lecture et d'écriture.
- En effet MySQL se charge de gérer les priorités des opérations entre les différentes connexions, évitant ainsi toute confusion.
- Il n'en va pas de même avec les fichiers, dont la structure est moins évoluée qu'une base de données.

Verrouillage des fichiers

- Si plusieurs utilisateurs accèdent au même fichier à partir du même script ou de deux scripts différents et y effectuent simultanément des opérations de lecture ou d'écriture, le fichier risque de devenir inutilisable pour chacun d'eux.
- Afin d'éviter la corruption des fichiers qui pourrait en résulter, il est essentiel, avant d'écrire ou de lire un fichier, que les scripts qui y accèdent définissent une priorité d'accès au premier script effectuant une opération sur le fichier : cela empêche les autres d'y accéder et d'y faire des modifications tant que le fichier n'est pas fermé.
- Vous disposez de la fonction flock() pour faire le verrouillage en lecture et/ou écriture.
  - Flock ($id_file,LOCK_SH) : bloque l'écriture dans le fichier mais laisse le libre accès en lecture à tous les utilisateurs.
  - Flock ($id_file,LOCK_EX) : bloque en lecture et en écriture
  - Flock ($id_file,LOCK_UN) : libère le verrou installé précedemment.
- LOCK_SH, LOCK_EX, LOCK_UN peuvent être resp. remplacés par 1,2, 3

Verrouillage des fichiers

- Compte tenu des fonctions que l'on vient de voir, le shéma général d'utilisation d'un fichier conduit à écrire systématiquement le code suivant :

```
$id_file = fopen("MonFichier.txt", "mode");
flock($id_file, 1 ou 2) ;
// opérations de lecture et/ou d'écriture
flock($id_file, 3) ;
fclose($id_file);
```

Ecriture dans un fichier

- Les fonctions fwrite() et fputs, alias l'une de l'autre, ont la syntaxe suivante :
  - Integer fwrite($id_file, "texte à insérer") ;
  - Integer fputs($id_file,"texte à insérer");

```
$id_file = fopen("MonFichier.txt", "w");
flock($id_file, 2) ;
fwrite($id_file, "toto");
flock($id_file, 3) ;
fclose($id_file);
```

Lecture dans un fichier

- Pour effectuer la lecture elle même, vous avez le choix entre plusieurs méthodes, chacune étant réalisable grâce à une fonction PHP spécialisée.
  - String fgets (resource $id_file)
    - Lit le fichier ligne par ligne
  - String fgetc (ressource $id_file)
    - Lit le fichier caractère par caractère
  - Il y'en a d'autres : fpassthru($id_file), readfile("nom_fichier"), etc...

Lire un fichier ligne par ligne

```
   if($ressource_fichier = fopen('test.txt', 'r')) //Si $ressource_fichier ne vaut pas
//FALSE on continue
    {
        flock($ressource_fichier,1);
        $contenu_fichier   = "";

        while($ligne=fgets($ressource_fichier)) //Tant que l'on est pas à la fin du
//fichier
        {
            $contenu_fichier .= $ligne;
  //Récupère la ligne en cours et l'ajoute au contenu de la variable $contenu_fichier
        }

       flock($ressource_fichier,3);
       fclose($ressource_fichier);

        echo $contenu_fichier; //affiche le contenu du fichier
    }
?>
```

Modifications de fichiers

- Copier un fichier :
  - Boolean copy (string "nom_fichier", string "nom_copie")
- Renommer un fichier :
  - Boolean rename (string "nom_actuel", string "nom_futur")
- Effacer un fichier :
  - Boolean unlink ("fichier_à_effacer")

Expressions régulières

Expressions régulières

- Une expression régulière permet de définir un modèle général, appelé motif de l'expression régulière, au moyen de caractères particuliers, représentatifs d'un ensemble de chaînes de caractères très variées.
- Un modèle peut simplement être une chaîne de caractère "toto", "tracteur" etc... Mais il peut également avoir une forme plus complexe : [a-t]{1|2}, [^abc], etc...
- Les expressions régulières ont de nombreuses applications. Elles permettent :
  - De vérifier la validité d'une chaîne de caractères
    - Adresse mail, url, etc
  - Extraire des parties bien précises d'une chaîne, d'un texte ou d'une page
  - Gérer une mise en forme spécifique
    - Rendre cliquable des adresses mails, url
  - Insérer, supprimer ou modifier des chaînes de caractères dans un texte
- Les modèles se combine donc généralement avec une fonction prédéfinie de PHP. La plus emblématique est la fonction preg_match : mais il y en a d'autres.

preg_match(motif,chaîne)

preg_match(motif, chaîne) : renvoie 1 si la chaîne contient le motif, 0 sinon

le motif est délimité par deux /

le motif /U.*x/ veut dire

un mot commençant par 'U', suivi d'une chaîne de caractère possiblement vide (symbolisé par '.*'), suivi du caractère 'x'.

Exemple :

echo preg_match("**/U.*x$/**", "Vive Unix") affiche 1;

echo preg_match("**/U.*x$/**", "Vive unix") affiche 0;

Exemple

```
<?php
$texte = "Les expressions régulières viennent du monde Unix.";
echo "texte = $texte <br>";
echo 'occurrence de "monte" :';
echo preg_match("/monte/", $texte);
echo '<br>occurrence de "mond" :';
echo preg_match("/mond/", $texte);
echo '<br>occurrence de "U.*x" :';
echo preg_match("/U.*x/", $texte);
?>
```

Exécution

texte = Les expressions régulières viennent du monde Unix.
occurrence de "monte" :0
occurrence de "mond" :1
occurrence de "U.*x" :1

Recherche de 1 ou plusieurs caractères

- Pour rechercher la présence d'un caractère particulier, il suffit de l'inclure entre des guillemets simples ou doubles. Pour rechercher le caractère @, il suffit donc d'écrire
  - $modèle="/@/" ; preg_match ($modele, $chaine)
- Pour vérifier si une chaîne contient au moins un des caractères d'une liste, il suffit d'énumérer tous ces caractères entre crochets
  - $modèle="/[xyz]/" ; preg_match ($modele, $chaine)
- Avec la même syntaxe, on peut définir comme motif, un intervalle de lettres ou de chiffres. Par exemple :
  - $modele=”/[a-z]/” ; preg_match ($modele, $chaine) recherche un caractère en lettre minuscule
  - $modele="/[A-Z]/"
  - $modele="/[0-9]/"

Recherche de 1 ou plusieurs caractères

- Pour rechercher dans une chaîne des caractères spéciaux, vous devez les faire précéder d'un antislash \
  - ., $, ^, ?, \, [,]
- Il existe des classes de caractères prédéfinies qui évitent d'avoir à créer les ensembles de caractères recherchés.
  - [[:alpha:]] = n'importe quelle lettre
  - [[:digit:]] ou \d = n'importe quel chiffre
  - \D = tout caractère sauf numérique
  - \w = une lettre [a-z] [A-Z] ou lettre accentuée, un chiffre ou '_'
  - \W = le contraire de \w
  - [[:alnum:]] n'importe quelle lettre ou chiffre
  - [[:space:]] ou \s n'importe quel espace blanc
  - \S = tout caractère sauf ceux définis par \s
  - [[:punct:]] n'importe quel signe de ponctuation
  - [[:lower:]] n'importe quelle lettre en minuscule

Recherche de 1 ou plusieurs caractères

- [[:upper:]] n'importe quelle lettre capitale
- [[:blank:]] espace ou tabulation
- \b = « word boundary character » : espace, ponctuation, début d'un texte, fin d'un texte
- \B = le contraire de \b

- Exemple : $modele="/[[:digit:]]/" permet de rechercher la présence d'un chiffre

Recherche d'une chaîne précise

- Pour rechercher la présence d'un mot précis dans une chaîne, il suffit de créer le motif en écrivant le mot entre guillemets.
  - $modèle="/toto/" ; preg_match ($modele, $chaine) renvoie 1 si la chaîne “toto” est dans la chaîne $chaine.

Restriction de caractères

- Pour interdire la présence d'un groupe de caractères, faites-le précéder par le caractère ^
- Pour exclure une plage entière de caractères, incluez le caractère ^ devant les caractères entre crochets qui définissent l'ensemble des caractères ou la plage des caractères.
  - $modele="/[^abc]/"
    - caba ne répond pas à cette définition
    - cabas est conforme au modèle
  - $modele="/[^A-Z]/" exclut les chaînes composées uniquement de majuscules
  - $modele="/[^0-9]/"

Restriction de caractères

- Attention : ce même caractère ^ peut avoir une autre signification s'il est placé devant un caractère ou une classe de caractères mais pas entre crochets. Dans ce cas, la chaîne sur laquelle est effectué la recherche doit commencer par les caractères qui suivent.
  - $modèle="/^bon/" ; preg_match ($modele, "bonjour") = 1.
  - $modèle="/^bon/" ; preg_match ($modele, "Gabon") = 0.
- Pour imposer qu'une chaîne se termine par un ou plusieurs caractères déterminés, faites suivre ces derniers du caractères $.
  - $modèle="/\.com/$" ; preg_match ($modele, "google.com") = 1.
  - $modele="/\.com/$" ; preg_match ($modele, "google.fr")= 0.
  - $modele="/\.com/$" ; preg_match ($modele, "communication")= 0.
  - $modele="/^\.com/$" ; preg_match ($modele, ".com")= 1.

Choix multiple

- Pour construire un modèle dans lequel plusieurs choix peuvent se faire sur un (voire plusieurs) sous-facteur, il faut utiliser la barre verticale | qui se comporte en tant qu'opérateur OU
- "/(un|le) chien/": chaîne qui contient "un chien" ou "le chien"
- "/(a|b)*/": chaîne qui contient une suite de "a" ou de "b"

Création de modèles généraux

- L'utilisation de caractères spéciaux permet de créer des modèles, dans lesquels vous pouvez précisez si un caractère doit être présent 0, 1 ou plusieurs fois de suite.
- Le caractère (.) permet de rechercher n'importe quel caractère
  - $modele="/mat./"
  - preg_match($modele, "les maths") = 1 ; preg_match($modele, "echec et mat") = 0
- Le caractère (?) indique que le caractère qui précède doit être présent 0 ou 1 fois. Le caractère recherché peut être présent plusieurs fois sans pour autant invalider la recherche.
  - $modele="/math?/" ; preg_match($modele, "mathematiques") = 1
  - preg_match($modele, "mat") = 1 ; preg_match($modele, "mathh") = 1
- Le caractère + indique que le caractère qui précède doit être présent au moins une fois dans la chaîne analysée
- Le caractère * indique que le caractère qui précède doit être présent 0, 1 ou plusieurs fois
  - $modele="pour*" ; preg_match($modele, "pou") = preg_match($modele, "poule")=1

Création de modèles généraux

- En combinant les caractères . et * vous pouvez rechercher une suite quelconque de caractères.
  - $modele="/mat.*/" ; permet de rechercher tous les mots d'un nombre de caractères quelconque contenant les lettres mat.
- L'usage des parenthèses associées aux caractères précédents effectue des regroupements permettant des recherches plus précises.
  - Le motif "/(ma)+/" recherche la présence du groupe de caractère "ma" au moins une fois.

Recherche d'un nombre donné de caractères

- L'usage des accolades permet de préciser le nombre de fois que vous souhaitez voir apparaître un caractère ou un groupe de caractères, selon les définitions suivantes :
  - {n} cherche exactement n fois le caractère ou le groupe qui précède
  - {n,} cherche au minimum n fois le caractère ou le groupe qui précède
  - {n,m} cherche entre n et m fois le caractère ou le groupe qui précède
- Par exemple le modèle "/cor{2}/" permet de rechercher des mots contenant la sous-chaîne "corr"

Définition d'un motif complexe : exemple de la validation d'un nom

- <?php

```
$modele="/^[a-zA-Z]+((-| )[a-zA-Z]+)*$/";
$nom="Jean-Paul";
if(!preg_match($modele,$nom))
{echo "Le nom \"$nom\" n'est pas conforme. Veuillez le ressaisir!";}
?>
```

- Le nom Jean-Paul Deux est accepté mais pas Jean-Paul 2.
- Ce genre de script peut être utilisé pour vérifier les saisies effectuées dans un formulaire avant l'enregistrement des données.

preg_match_all(motif,chaîne, tableau)

preg_match_all(motif, chaîne, tableau)

analyse la chaîne selon le motif et stocke dans le tableau les

occurrences incluses dans la chaîne qui satisfont l'expression régulière du motif **: tableau[0][n] contient la (n+1)-ième occurrence trouvée**

renvoie 1 si la chaîne contient au moins un motif de l'expression régulière sinon renvoie 0

d'autres paramètres sont possibles ainsi que des résultats dans tableau[1].

Exemple

```
<?php
$texte = "Les expressions régulières viennent du monde Unix.";
$date = "4/05/2003";
if (preg_match_all("/es/", $texte, $tab_occurrences))
    echo "$texte contient ".count($tab_occurrences[0])
    ." fois le motif \"es\"<br>" ;
if (preg_match_all("/[0-9]+/", $date, $tab_occurrences))
    echo "$date se décompose en : <br>"
    ."jour = ".$tab_occurrences[0][0]
    ."<br> mois = ".$tab_occurrences[0][1]
    ."<br> annee = ".$tab_occurrences[0][2]." <br>" ;
?>
```

Exécution : Les expressions régulières viennent du monde Unix.
contient 3 fois le motif "es"
4/05/2003 se décompose en :
jour = 4
mois = 05
annee = 2003

preg_split, preg_replace

- **tableau = preg_split(motif, chaîne)**
  - découpe(éclate) la chaîne selon le motif séparateur
  - stocke dans le tableau les sous-chaînes séparées par le motif. **tableau[n] contient la (n+1)-ième sous-chaîne.**
- **chaine=preg_replace(motif, substitut, chaîne)**
  - fonction chercher-remplacer classique
  - remplace, dans la chaîne, les sous-chaînes correspondant à l'expression régulière du motif par la chaîne substitut et retourne la chaîne obtenue.
  - la fonction fonctionne aussi avec des tableaux de recherche et remplacement et aussi sur des tableaux de chaînes à soumettre...
- **chaine =preg_replace (motif, substitut, chaîne, limite) ;**
  - Si l'entier limite est renseigné, le nombre de remplacements sera limité.

Exemple pour preg_split

```
<?php
$texte = "Les expressions régulières viennent du monde Unix.";
$tab_recup = preg_split("/[,\.\s]+/", $texte);
echo $texte."<br> découpé aux espaces, virgules et point : <br>" ;
for ($i = 0; $i < count($tab_recup); $i++)
   echo "une découpe : ".$tab_recup[$i]."<br>";
?>
```

\. despécialise le métacaractère '.', il devient le caractère '.'
\s signifie espace, tabulation ou passage à la ligne

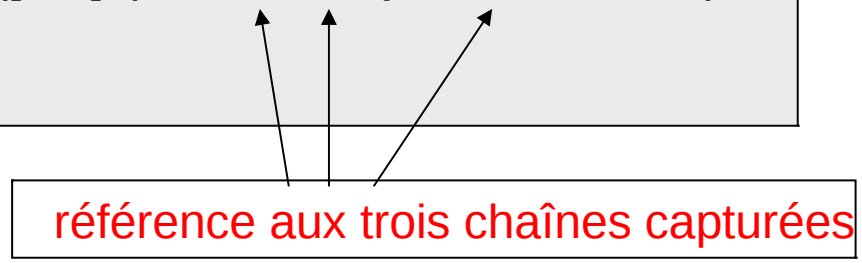
Exécution :

Les expressions régulières viennent du monde Unix.
découpé aux espaces, virgules et point :
une découpe : Les
une découpe : expressions
une découpe : régulières
une découpe : viennent
une découpe : du
une découpe : monde
une découpe : Unix
une découpe :

ce blanc est dû au point de fin de ligne

Exemple pour preg_replace

```
<?php
$date = "4:05:2003";
echo $date."<br>";
$date = preg_replace("/^(([0-9]+):([0-9]+):([0-9]+)$/", "$2 $1 year $3", $date);
echo $date."<br>";
?>
```



Exécution :
4:05:2003
05 4 year 2003

Les parenthèses () définissent des sous-expressions à capturer, de forme ([0-9]+)
Il y a 3 sous-expressions capturées : "4" "05" "2003"
La chaîne substitut comporte des $1, $2, $3 qui font référence, dans l'ordre, aux chaînes capturées.
Le remplacement se fait en remplaçant les $i par la sous-expression capturée correspondante.

preg_replace_callback(motif, fonction, chaîne)

Le comportement de preg_replace_callback est presque identique à celui de preg_replace, hormis le fait qu'à la place du paramètre de remplacement, il faut spécifier une fonction de callback qui sera appelée, avec les éléments trouvés en arguments. Cette fonction retourne alors la chaîne de remplacement.

```
<?php
$chaine='Mise en majuscules des initiales du prénom composé anne-marie.';
function mot_compose($masque){
return ucfirst($masque[1]).'-'.ucfirst($masque[2]);
}
echo preg_replace_callback("/([[:alpha:]]+)-([[:alpha:]]+)/","mot_compose",$chaine);
?>
```

les deux expressions capturées

Exécution : Mise en majuscules des initiales du prénom composé Anne-Marie.

La fonction ucfirst(chaîne) met la première lettre de la chaîne en majuscule.

Parenthèses non capturantes

Il est possible d'utiliser des parenthèses sans que ces dernières n'impliquent une capture.

Plus précisément, pour qu'une paire de parenthèses ne soit pas capturante, il faut remplacer dans l'expression régulière la parenthèse ouvrante (par (?:

tableau=preg_grep (motif, tableau, option)

preg_grep retourne dans un tableau les éléments extraits d'un autre tableau correspondant au motif, ou ne correspondant pas au motif avec l'option PREG_GREP_INVERT.

```
<?php
$tablo=array('1','2.25','3','4.75','6','7.33','9','11','14.45','19');
$trouve=array_values(preg_grep('`^\d+\.\d+$`',$tablo));
$nb=count($trouve);
for($i=0;$i<=$nb;$i++){

   echo $trouve[$i].'-';
}
?>
```

Exécution : 2.25 - 4.75 - 7.33 - 14.45 -