RÉSUMÉ DU COURS DE PHP

— Formulaires —

`<form>` : définition du formulaire

- **action :** nom de la page executée par le serveur après envoi
- **method :** POST ou GET, méthode de transmission des données
- **enctype :** encodage utilisé pour la transmission des données

`<input>` : champ du formulaire

- **type :** text, password, hidden, checkbox, radio ...
- **name :** nom de la variable passée en GET ou POST
- **value :** valeur par défaut du champ

`<select>` : liste déroulante de sélection de valeurs (`<option>`)
`<textarea>` : champ de saisie multilignes

`<input type="submit">` : termine la saisie et transmet les informations
`<input type="reset">` : réinitialise le formulaire

Après l'envoi des données du formulaire, la cible traite les données.
Méthode **POST** : `<?php echo $_POST["nom"]; ?>`
Méthode **GET** : `<?php echo $_POST["nom"]; ?>`

— Le langage PHP —

`<?php ... ?>` : structure du code
Le code est interprété par le serveur pour générer la page.
Chaque instruction se termine par un `;` (structure similaire au C).

Commentaires : `/* plusieurs lignes */` ou `//1 ligne`

Variables : pas de type fixe, nom sensible à la casse, préfixé par $
Constantes : `<?php define("NOM", valeur); ?>`
Variables superglobales : `$_GET` , `$_POST` , `$_SERVER` , `$_SESSION`

Les différentes interprétations de chaînes de caractères :

Code PHP	Interprétation
`"Le titre \$titre est : $titre"`	Le titre $titre est : blabla
`'Le titre est : $titre'`	Le titre est : $titre
`"Je goute plusieurs ${boisson}s"`	Je goute plusieurs vins

Types : booléen, entier, flottant, tableaux ...

Particularités : longueur dynamique, type variable.

Indices d'un tableau : entier et/ou chaîne de caractères.

Pour parcourir un tableau en PHP :

```
foreach($tab as $value) { echo "$value \n<br/>"; }
```

```
foreach($tab as $key => $value) { echo "Valeur $key : $value \n<br/>"; }
```

Fonction : `<?php function fun($p1, &$p2) { ... } ?>`

Sans préfixe `&` **:** passage par valeur.

Avec préfixe `&` **:** passage par adresse (modifie la variable).

Syntaxe d'appel : `<?php fun($i, $j); ?>`

— Inclusions de fichiers —

`include(fichier)` : permet d'inclure un fichier (warning si pas possible)

`require(fichier)` : produit une erreur si l'inclusion n'est pas possible

Possible d'ajouter le suffixe `_one` si l'inclusion est répétitive.

Il est préférable d'utiliser des inclusions pour des **définitions de fonctions**.

— Base de données et PHP —

Stockage d'informations propres à l'application / concernant le site web.

Le serveur fait l'interaction avec le **SGBD** (ici, PostgreSQL).

Un exemple d'interaction avec une BDD en PHP :

```php
<?php
$requete = "SELECT * FROM film;";
$connexion = pg_connect("host=localhost dbname=films");
$reponse = pg_query($connexion, $requete);
if($reponse) {
    $nb_tuples = pg_num_rows($reponse);
    echo '<ul>';
    while($tuple = pg_fetch_assoc($requete)) {
        echo '<li>'.$tuple["titre"];
        echo ' sorti en '.$tuple["annee"].'</li>';
    }
    echo '</ul>';
}
else {
    echo 'Problème de la requête.';
}
pg_close($connexion);
?>
```

Champs cachés : transmission de données entre pages.

L'extension PHP PDO (PHP Data Objects) permet la portabilité de l'accès aux BDD.
Plus besoin de se soucier des fonctions spécifiques à un SGBD.

`$db = new PDO("psql:host=$host;dbname=$dbname", $user, $pass, array(PDO::ATTR_PERSISTENT => true));`

Un curseur est un tampon correspondant au résultat d'une requête.

Algorithme de traitement séquentiel des curseurs triés :

```
lecture premier tuple
    boucle client
    traitement début client
        boucle commandes du client courant
        traitement début commande
```

```
            boucle des lignes de la commande courante
            traitement d'un tuple
            lecture du tuple suivant
        traitement fin commande
    traitement fin client
fin
```

— MVC —

***M*odèle :** accès aux données dans la BD.

***V*ue :** affichage et transmission des données au client (HTML ...).

***C*ontrôleur :** traitement et analyse des données.

Le contrôleur accède au modèle et présente la vue.

— Sécurité des pages —

Toujours contrôler les données reçues via `GET` / `POST`.

Contrôler : tout ce qui est attendu est là, les valeurs sont conformes.

Supprimer l'autocomplétion dans un formulaire : `autocomplete="off"`

Supprimer la mise en cache :

`<?php header("Cache-Control : no-cache"); ?>`

`<?php header("Expires: Thu, 01 Jan 1970 00:00:00 GMT"); ?>`

Se protéger des injections SQL :

- filtrer les entrées
- ne pas afficher d'informations en cas d'erreur
- protéger avec la fonction `addcslashes($str, $charlist)`
- échapper les caractères `'`, `"`, `%`, `\0`, `\n`, `\r`, `\x00` et `\x1a`

Se protéger des injections HTML (faille XSS) :

- filtrer les entrées (`utf_decode`, `strip_tags`, `filter_*`)
- ne pas stocker des informations non vérifiées
- préciser le jeu de caractères

- échapper les balises (`htmlspecialchars` , `htmlentities`)