
php

Php - SESSION

Pour toute utilisation d'une session en php il faudra initialiser les sessions à l'aide de « session_start() ».

Cette fonction devra être appelé au début de la page AVANT TOUT AUTRE CODE (y compris le doctype) et sur toutes les pages ou les sessions sont utilisées.

Les sessions ont une durée de vie qui dépendra de la configuration de votre serveur.

Pour créer une session, exemple : `$_SESSION['test']="test validé";`

Aussi simple que ça.

Il est possible de réécrire par-dessus une session en choisissant un index existant (c'est un tableau).

Il est également possible d'enlever des variables précises des sessions avec unset(), toutes les variables avec session_unset(), et de détruire les sessions avec session_destroy().

Php – SESSION tp-6

A l'aide d'une session, je vous invite à créer une page de connexion, avec un formulaire :

- Le form va demander un pseudo, un mdp
- Si le pseudo et le mot de passe sont bon, la page n'affiche plus le formulaire mais un bonjour + pseudo ainsi qu'un bouton se deconnecter.
- Tant que l'utilisateur ne clique pas sur le bouton se déconnecter il restera connecté.
- Si il clique sur se déconnecter, le formulaire de connexion est de nouveau affiché.

Php – BDD - PDO

Pour ce connecter à la base de données nous allons utiliser PDO, vous connaissez peut être mysqli mais c’est dépassé et nous allons donc rester sur PDO.

C’est une extension PHP orientée objet. Elle permet de se connecter à une base de données et d’effectuer des requêtes.

Php – BDD – PDO – se connecter à une bdd

```php
try {
    $db = new PDO("mysql:host=localhost;dbname=tp_zoo_debutant;charset=utf8", "root", "");
} catch (Exception $e) {
    die($e->getMessage());
}
```

Php – BDD – PDO – créer sa requête

```php
// JE stock ma requête
$sql = requête sql ici ex : SELECT * from table

// J'envoi la requête et stock la reponse
$req = $db->query($sql);

// Je traite la reponse et stock les données
// fetchAll crée un tableau avec toutes les réponses, si vous utilisez fetch, cela renvoi le premier résultat et il faut boucler pour avoir la suite
$data = $req->fetchAll(PDO::FETCH_OBJ);

// Fermeture de la requête (inutile avec mysql)
$req->closeCursor();
```

Php – BDD – PDO – Rollback

Un des avantages de l'arrivé de PDO est la possibilité de rollback, ce système est mis en place en général en cas d'erreur, il est possible d'annuler toutes les opérations sql qui ont eu lieu.

```
$db->beginTransaction();

if(!$result){
        $db->rollBack();
    }
else{
      $db->commit();
  }
```

Php – BDD – PDO – info requête

Quand vous créez une requête il peut être utile d'avoir des informations détaillées, nottament quand c'est des requêtes avec des paramètres.

```
$req->debugDumpParams();
```

Php – BDD – PDO – requête préparée

Les requêtes préparées permettent de sécuriser nos requêtes afin de se protéger des injections sql.

C’est un moyen de traiter notre script avant de l’executer et il est indispensable d’utiliser ce genre de requêtes surtout quand vous faites passer des variables dans votre requête.

Cela va se passer en deux temps :

```
$req =  $db->prepare($sql); préparation

$result = $req->execute() execution
```

Php – BDD – PDO – passer des paramètres

Comme expliqué précédemment, nous pouvons avoir besoin de faire passer des variables dans nos requêtes sql, souvent dans la clause where.

Exemple v1 :

```
$sql = "DELETE FROM animal WHERE animal_nom = :animal";
  $req =  $db->prepare($sql);

  $result = $req->execute([
      ":animal"=>$_GET['animal-choice']
  ]);
```

Php – BDD – PDO – passer des paramètres

Exemple v2 :

```
$sql = "DELETE FROM animal WHERE animal_nom = :animal";
$req =  $db->prepare($sql);


$req->bindValue(“:animal", $_GET['animal-choice'],PDO::PARAM_STR);
  $result = $req->execute();
```

Php – BDD – PDO – passer des paramètres

Exemple v3 :

```
$sql = "DELETE FROM animal WHERE animal_nom = ?";
$req =  $db->prepare($sql);


$req->bindValue(“1", $_GET['animal-choice'],PDO::PARAM_STR);
  $result = $req->execute();
```