

Index.php

Le fichier index.php agit comme un routeur. En tant que routeur, il est responsable de recevoir les requêtes HTTP, d'analyser les paramètres de l'URL et de diriger les requêtes vers le contrôleur approprié pour gérer la logique métier et retourner la vue correcte.

```
define("URL", str_replace("index.php", "", (isset($_SERVER["HTTPS"]) ? "https" : "http") . "://" . $_SERVER["HTTP_HOST"] . $_SERVER["PHP_SELF"]));
```

Cette ligne définit la constante URL, qui est utilisée pour générer des liens absolus dans votre application.

Rôle de chaque composant

- **Contrôleur (UserController, LoginController, LogoutController):** Gère la logique métier. Chaque contrôleur contient des méthodes spécifiques pour gérer les différentes actions (ajout, suppression, mise à jour, etc.).
- **Vue (accueil.view.php, create.view.php, update.view.php, delete.view.php):** Affiche les pages HTML aux utilisateurs. Chaque vue correspond à une partie spécifique de l'interface utilisateur.
- **Manager (UserManager):** Interagit avec la base de données pour effectuer des opérations CRUD (Create, Read, Update, Delete).

Le fichier agit comme un routeur dans l'application en dirigeant les requêtes entrantes vers les contrôleurs appropriés en fonction de l'URL. C'est une pratique courante dans les architectures MVC (Modèle-Vue-Contrôleur) pour séparer la logique de routage de la logique métier et de la présentation.

```
define("URL", str_replace("index.php", "", (isset($_SERVER["HTTPS"]) ? "https" : "http") . "://" . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF']));
```

1. `define("URL", ...);`

La fonction `define` est utilisée pour définir une constante en PHP. Dans ce cas, elle définit une constante nommée `URL`.

2. `str_replace("index.php", "", ...);`

La fonction `str_replace` remplace toutes les occurrences de la chaîne `"index.php"` par une chaîne vide. Cela est fait pour supprimer `"index.php"` de l'URL finale.

3. `(isset($_SERVER["HTTPS"]) ? "https" : "http")`

Cette partie utilise un opérateur ternaire pour déterminer si la connexion est sécurisée (HTTPS) ou non (HTTP).

- `isset($_SERVER["HTTPS"])`: Vérifie si la variable `$_SERVER["HTTPS"]` est définie, ce qui signifie généralement que la connexion est sécurisée (HTTPS).
- `? "https" : "http"`: Si `$_SERVER["HTTPS"]` est définie, elle retourne `"https"`. Sinon, elle retourne `"http"`.

4. `"http" . "://"`

Cette partie de la chaîne ajoute `://` après le schéma (soit `"http"` soit `"https"`).

5. `$_SERVER['HTTP_HOST']`

`$_SERVER['HTTP_HOST']` contient le nom de l'hôte de la requête actuelle, c'est-à-dire le domaine ou l'adresse IP où le script est exécuté.

6. `$_SERVER['PHP_SELF']`

`$_SERVER['PHP_SELF']` contient le chemin relatif de l'URL vers le script en cours d'exécution. Cela inclut le nom du script lui-même (`"index.php"` dans ce cas).

Assemblage complet

Voici comment les parties s'assemblent :

`(isset($_SERVER["HTTPS"]) ? "https" : "http")`: Détermine si le protocole est `"http"` ou `"https"`.

`://"`: Ajoute `://` après le protocole.

`$_SERVER['HTTP_HOST']`: Ajoute le nom de l'hôte (domaine ou IP).

`$_SERVER['PHP_SELF']`: Ajoute le chemin du script en cours.

Ensemble, cela forme une URL complète du format `http://exemple.com/index.php` ou <https://exemple.com/index.php>.

Remplacement de "index.php"

La fonction `str_replace` remplace "index.php" par une chaîne vide, ce qui donne l'URL de base du site sans "index.php" à la fin :

`http://exemple.com/`

<https://exemple.com/>

Exemple

Supposons que l'application s'exécute sur `https://exemple.com/index.php?page=accueil`, la ligne se décompose comme suit :

`(isset($_SERVER["HTTPS"]) ? "https" : "http")` donne "https".

`$_SERVER["HTTP_HOST"]` donne "exemple.com".

`$_SERVER['PHP_SELF']` donne `"/index.php"`.

`$_SERVER['PHP_SELF']` donne `"/index.php"`.

Ainsi, la chaîne complète serait `https://exemple.com/index.php`.

Enfin, `str_replace("index.php", "", "https://exemple.com/index.php")` remplace "index.php" par une chaîne vide, donnant `https://exemple.com/`.

Résultat final

La constante URL est définie comme `https://exemple.com/`.

Cette ligne de code crée une URL de base qui peut être utilisée pour générer des liens relatifs dans l'application, en s'assurant qu'ils commencent toujours par le domaine de base du site sans le nom du script (`"index.php"`).

```

test.php > ...
1  <?php
2  try {
3      if (empty($_GET["page"])) {
4          header("Location: " . URL . "accueil");
5          exit();
6      } else {
7          // Autre logique de routage...
8      }
9  } catch (Exception $e) {
10     echo $e->getMessage();
11 }
12

```

1. try { ... } catch (Exception \$e) { ... }

Le bloc try-catch est utilisé pour gérer les exceptions. Si une exception est levée dans le bloc try, elle sera capturée par le bloc catch, permettant ainsi de gérer les erreurs de manière contrôlée.

- try { ... } : Contient le code qui pourrait potentiellement lever une exception.
- catch (Exception \$e) { ... } : Capture l'exception et exécute le code pour gérer l'erreur. Ici, il affiche simplement le message d'erreur.

2. if (empty(\$_GET["page"])) { ... }

Ce bloc vérifie si le paramètre page est présent dans l'URL.

`empty($_GET["page"])` : Utilise la fonction `empty` pour vérifier si la variable `$_GET["page"]` est vide ou non définie. La variable `$_GET` est une superglobale en PHP qui contient les paramètres de la requête HTTP. Par exemple, dans l'URL `http://exemple.com/index.php?page=accueil`, le paramètre `page` aurait la valeur `accueil`.

3. header("Location: " . URL . "accueil");

Si le paramètre `page` est vide ou non défini, ce bloc de code redirige l'utilisateur vers la page d'accueil.

`header("Location: " . URL . "accueil")` : Utilise la fonction `header` pour envoyer un en-tête HTTP `Location`, ce qui provoque une redirection du navigateur vers une autre URL.

"Location: " : Indique que l'en-tête de redirection est utilisé.

URL : La constante `URL` définie précédemment, représentant l'URL de base de l'application (par exemple, `http://exemple.com/`).

"accueil" : La chaîne `accueil` est concaténée à `URL`, formant l'URL complète de redirection. Ainsi, l'utilisateur est redirigé vers `http://exemple.com/accueil`.

4. exit();

exit(); : Arrête l'exécution du script immédiatement après l'envoi de l'en-tête de redirection. Cela garantit que le reste du code ne s'exécute pas après la redirection.

L'ensemble de ce bloc de code vérifie si le paramètre page est défini dans l'URL :

Vérification du paramètre page : Si page est vide ou non défini (empty(\$_GET["page"])), l'utilisateur est redirigé vers la page d'accueil.

Redirection vers l'accueil : Utilisation de la fonction header pour envoyer une redirection HTTP à <http://example.com/accueil>.

Arrêt du script : exit() arrête l'exécution du script pour s'assurer qu'aucun autre code n'est exécuté après la redirection.

Ce mécanisme permet de gérer la page par défaut de votre application en redirigeant les utilisateurs vers la page d'accueil lorsqu'aucun paramètre page n'est spécifié dans l'URL.

« PAGE »

Le paramètre page provient de l'URL de la requête HTTP. Lorsque vous visitez une page web, les paramètres peuvent être passés dans l'URL après le symbole ?. Par exemple, dans l'URL suivante :

```
http://example.com/index.php?page=accueil
```

Le paramètre page a la valeur accueil.

En PHP, les paramètres de l'URL sont accessibles via la superglobale \$_GET. Cette superglobale est un tableau associatif qui contient tous les paramètres passés dans l'URL de la requête HTTP. Chaque paramètre est accessible par son nom comme clé du tableau.

Par exemple, pour l'URL suivante :

```
http://example.com/index.php?page=accueil
```

La variable \$_GET aurait la structure suivante :

```
$_GET = [  
    "page" => "accueil"  
];
```

Utilisation de `$_GET["page"]`

```
if (empty($_GET["page"])) {  
    header("Location: " . URL . "accueil");  
    exit();  
} else {  
  
}
```

`$_GET["page"]` vérifie si le paramètre page est défini et non vide dans l'URL de la requête actuelle. Voici comment cela fonctionne :

1. Vérification de l'existence et de la valeur : La fonction `empty()` vérifie si `$_GET["page"]` est vide ou non défini. Si c'est le cas, cela signifie que l'utilisateur n'a pas fourni de paramètre page dans l'URL.
2. Redirection si page est vide : Si `$_GET["page"]` est vide, l'utilisateur est redirigé vers `http://exemple.com/accueil` en utilisant l'en-tête Location.

En résumé, le paramètre page est extrait de l'URL de la requête en utilisant la superglobale `$_GET` de PHP. Cela permet au script de savoir quelle page afficher ou quelle action effectuer en fonction de la valeur du paramètre page.

.htaccess :

Le fichier .htaccess est utilisé pour configurer le serveur web Apache afin de rediriger les requêtes vers le fichier index.php, tout en passant les parties de l'URL en tant que paramètres.

1. RewriteEngine On

Cette ligne active le module de réécriture d'URL d'Apache (`mod_rewrite`). Cela permet de définir des règles de réécriture d'URL dans le fichier .htaccess.

2. RewriteCond %{REQUEST_FILENAME} !-f

Cette ligne définit une condition de réécriture. La condition vérifie si le fichier demandé dans l'URL (`%{REQUEST_FILENAME}`) n'existe pas (`!-f`). En d'autres termes, si le fichier demandé n'est pas un fichier physique sur le serveur, alors la règle de réécriture suivante sera appliquée.

3. RewriteCond %{REQUEST_FILENAME} !-d

Cette ligne ajoute une autre condition de réécriture. Elle vérifie si le fichier demandé n'est pas un répertoire (`!-d`). Si le fichier demandé n'est ni un fichier existant ni un répertoire existant, alors la règle de réécriture suivante sera appliquée.

4. RewriteRule ^(.*)\$ index.php?page=\$1 [L]

Cette ligne définit la règle de réécriture. Elle spécifie que toutes les requêtes qui ne correspondent pas à un fichier ou un répertoire existant doivent être redirigées vers `index.php` avec une variable page qui contient le reste de l'URL.

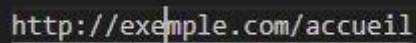
^(.*)\$: Cette expression régulière correspond à toute l'URL après le nom de domaine. Le ^ signifie le début de la chaîne, (.*) capture tout le reste de l'URL, et \$ signifie la fin de la chaîne. La partie capturée est stockée dans \$1.

index.php?page=\$1 : Redirige la requête vers index.php, en passant la partie capturée de l'URL (tout après le nom de domaine) comme valeur du paramètre page. Par exemple, si l'URL demandée est <http://example.com/mon-profil>, la requête sera redirigée vers <http://example.com/index.php?page=mon-profil>.

[L] : Cette drapeau signifie "Last" (dernière règle). Il indique à Apache d'arrêter de traiter d'autres règles de réécriture après cette règle.

Exemple de fonctionnement

URL demandée :



```
http://exemple.com/accueil
```


Vérification des conditions :

accueil n'est ni un fichier existant (!-f) ni un répertoire existant (!-d).

Application de la règle de réécriture :

Redirigé vers : <http://example.com/index.php?page=accueil>

URL demandée :



```
http://exemple.com/public/css/style.css
```

Vérification des conditions :

public/css/style.css est un fichier existant (-f).

Résultat :

La règle de réécriture n'est pas appliquée. Le fichier style.css est servi directement.

Le fichier .htaccess permet de rediriger toutes les requêtes qui ne correspondent pas à des fichiers ou répertoires existants vers index.php. Cela permet de créer des URLs propres et conviviales, tout en centralisant la logique de routage dans un seul fichier PHP (index.php). Ce mécanisme est couramment utilisé dans les frameworks MVC pour gérer les routes de l'application.