DOCUMENTATION API

MODEL

Ce code est écrit en PHP et il semble être un exemple d'un mini-système de gestion d'étudiants utilisant une base de données. Je vais vous expliquer les différentes parties de ce code :

1. **Require_once("./Config.php") :** Cela inclut le fichier Config.php dans le script. Ce fichier pourrait contenir des configurations telles que les informations de connexion à la base de données.

2.**La classe Etudiant :** Cette classe contient plusieurs méthodes statiques pour effectuer des opérations sur les étudiants dans la base de données.

- **enregistrer_Etudiant()**: Cette méthode prend en paramètres les informations d'un étudiant (noms, genre, lieu de naissance, date de naissance, adresse), génère un matricule unique pour l'étudiant en fonction de son genre et de l'ID auto-incrémenté dans la table `compteur_id`, puis insère les informations de l'étudiant dans la table `etudiant`. Elle retourne un tableau avec un message de succès ou d'échec.

- **Supprimer_Etudiant():** Cette méthode prend en paramètre l'ID d'un étudiant et supprime cet étudiant de la base de données. Elle retourne un tableau avec un message de succès ou d'échec.

- **Modifier_Etudiant()** : Cette méthode permet de modifier les informations d'un étudiant dans la base de données en fonction de son ID. Elle retourne un tableau avec un message de succès ou d'échec.

- **get_all_Etudiant():** Cette méthode récupère tous les étudiants de la base de données et les retourne sous forme de tableau. Si aucun étudiant n'est trouvé, elle retourne un message indiquant l'absence de données.

- **compter_etudiant():** Cette méthode compte le nombre total d'étudiants dans la base de données et retourne ce nombre. Si aucun étudiant n'est trouvé, elle retourne un message indiquant l'absence de données.

- **get_one_etudiant():** Cette méthode récupère les informations d'un seul étudiant en fonction de son ID et les retourne sous forme de tableau. Si aucun étudiant n'est trouvé, elle retourne un message indiquant l'absence de données.

- **get_last()**: Cette méthode récupère le dernier enregistrement d'étudiant dans la base de données et le retourne sous forme de tableau.

CONTROLLERS

Ce code PHP est une série de fonctions qui agissent comme des points d'entrée pour interagir avec la classe `Etudiant` définie dans le fichier `Etudiant.php`.

Voici ce que chaque fonction fait :

1. **Enregistrer_Etudiant()** : Cette fonction récupère les données du formulaire POST (noms, genre, lieu de naissance, date de naissance, adresse), nettoie les entrées potentiellement dangereuses avec `htmlspecialchars()`, puis appelle la méthode statique `enregistrer_Etudiant()` de la classe `Etudiant` pour enregistrer l'étudiant dans la base de données.

2. **Supprimer_Etudiant()** : Cette fonction récupère l'ID de l'étudiant à supprimer à partir du formulaire POST, puis appelle la méthode statique `Supprimer_Etudiant()` de la classe `Etudiant` pour supprimer cet étudiant de la base de données.

3. **Modifier_Etudiant()** : Cette fonction récupère les données du formulaire POST (ID, noms, genre, lieu de naissance, date de naissance, adresse), nettoie les entrées potentiellement dangereuses avec `htmlspecialchars()`, puis appelle la méthode statique `Modifier_Etudiant()` de la classe `Etudiant` pour modifier les informations de l'étudiant dans la base de données.

4. **get_all_etudiant() :** Cette fonction appelle la méthode statique `get_all_Etudiant()` de la classe `Etudiant` pour récupérer tous les étudiants enregistrés dans la base de données.

5. **compter_etudiant()** : Cette fonction appelle la méthode statique `compter_etudiant()` de la classe `Etudiant` pour compter le nombre total d'étudiants enregistrés dans la base de données.

6**. get_one_etudiant() :** Cette fonction récupère l'ID de l'étudiant à afficher à partir de la requête GET, puis appelle la méthode statique `get_one_etudiant()` de la classe `Etudiant` pour récupérer les informations de cet étudiant dans la base de données.

Ces fonctions servent de points d'entrée pour interagir avec les fonctionnalités de gestion des étudiants définies dans la classe `Etudiant`. Elles permettent d'isoler la logique métier de la manipulation des données de la logique de présentation, ce qui rend le code plus modulaire et plus facile à maintenir.

ROUTES

Ce code PHP est un script de contrôleur qui gère les requêtes entrantes pour l'API des étudiants. Il suit une logique de routage simple basée sur l'URL pour déterminer quelle action doit être effectuée.

Voici ce que fait chaque partie du code :

1. `header('Content-Type: Application/json');` : Cela définit l'en-tête de la réponse HTTP pour indiquer que le contenu de la réponse sera au format JSON.

2. **`require("./Controllers/Etudiant.php");`** : Cela inclut le fichier `Etudiant.php` où sont définies les fonctions de manipulation des données des étudiants.

3. **Récupération de l'URL**: Le code analyse l'URL de la requête pour extraire les parties pertinentes. Il extrait la troisième et la quatrième partie de l'URL pour déterminer le chemin et l'action à effectuer.

4. **Routage des requêtes** : Le code vérifie d'abord si la première partie de l'URL est "etudiant". Ensuite, selon la méthode de requête HTTP (GET ou POST) et la deuxième partie de l'URL, il appelle différentes fonctions de manipulation des étudiants :

- Pour les requêtes GET :
  - Si la deuxième partie de l'URL est "get_etudiant", il récupère tous les étudiants.
  - Si la deuxième partie de l'URL est "compter", il compte le nombre total d'étudiants.

- Pour les requêtes POST :
  - Si la deuxième partie de l'URL est "get_one", il récupère un seul étudiant en fonction de l'ID fourni.
  - Si la deuxième partie de l'URL est "enregistrer", il enregistre un nouvel étudiant.

- Si la deuxième partie de l'URL est "modifier", il modifie les informations d'un étudiant existant.

- Si la deuxième partie de l'URL est "supprimer", il supprime un étudiant.

5. **Réponse JSON :** Après avoir effectué l'action appropriée, le code encode les données de réponse dans un format JSON et les renvoie en tant que réponse HTTP.

Ce script de contrôleur agit comme un point d'entrée pour l'API des étudiants, acheminant les requêtes HTTP vers les fonctions appropriées de manipulation des données définies dans le fichier `Etudiant.php`, et renvoyant les résultats au format JSON.

INDEX

Ce code PHP gère les requêtes entrantes pour une API en vérifiant d'abord les informations d'identification fournies dans l'URL. Voici ce que fait chaque partie du code :

1. **Headers:** Les trois premières lignes définissent des en-têtes HTTP pour permettre le contrôle d'accès depuis n'importe quelle origine (`Access-Control-Allow-Origin: *`), ainsi que pour accepter n'importe quelle méthode et n'importe quel en-tête (`Access-Control-Allow-Method: *` et `Access-Control-Allow-Headers: *`). Le type de contenu est également défini comme JSON avec UTF-8 (`Content-Type: Application/json;charset=utf-8`).

2. **Vérification des informations d'identification:** Le code vérifie si les paramètres `user` et `mdp` sont présents dans l'URL et correspondent à des valeurs spécifiques. Dans ce cas, il vérifie si `user` est égal à "tplabo" et `mdp` est égal à "12345". Si les informations d'identification sont valides, le script inclut plusieurs fichiers de routage correspondant à différentes parties de l'API.

3. **Inclusion des fichiers de routage:** Si les informations d'identification sont valides, le script inclut plusieurs fichiers de routage (`etudiant.php`, `promotion.php`, `options.php`, `inscription.php`, `annee.php`, `all.php`). Ces fichiers de routage sont responsables de la définition des routes et des actions à prendre pour différentes parties de l'API.

4. **Réponse d'accès refusé:** Si les informations d'identification ne sont pas valides, le script renvoie un message JSON indiquant un accès refusé et termine le script en utilisant `exit`.

En résumé, ce script PHP agit comme un point d'entrée pour une API en vérifiant les informations d'identification fournies dans l'URL et en incluant les fichiers de routage correspondants pour gérer les différentes parties de l'API.

AXIOS & JSON, QUID ?

Axios est une bibliothèque JavaScript populaire utilisée pour effectuer des requêtes HTTP à partir du navigateur ou de Node.js. Il fournit une interface simple et facile à utiliser pour interagir avec des API distantes et récupérer des données.

Voici quelques-unes de ses caractéristiques principales :

1. **Facilité d'utilisation** : Axios offre une syntaxe simple et intuitive pour effectuer des requêtes HTTP, ce qui le rend facile à apprendre et à utiliser.

2**. Prise en charge de différentes méthodes HTTP : Axios** prend en charge toutes les méthodes HTTP courantes telles que GET, POST, PUT, DELETE, etc.

3**. Promesses*:*** Axios utilise des promesses JavaScript pour gérer les requêtes asynchrones, ce qui facilite la gestion des réponses et des erreurs.

4. **Interceptions de requêtes et de réponses:** Il permet de définir des intercepteurs pour les requêtes et les réponses, ce qui permet de modifier les requêtes sortantes ou de traiter les réponses entrantes de manière centralisée.

5. **Annulation de requête:** Axios prend en charge l'annulation de requête, ce qui permet d'annuler une requête en cours si nécessaire, par exemple lorsque l'utilisateur quitte une page ou effectue une nouvelle action.

6. **Support du navigateur et de Node.js :** Axios peut être utilisé à la fois dans un navigateur web et dans un environnement Node.js, ce qui le rend polyvalent et largement utilisé dans différents types de projets.

En résumé, Axios est une bibliothèque JavaScript puissante et polyvalente pour effectuer des requêtes HTTP, offrant une interface simple et des fonctionnalités avancées pour interagir avec des API distantes.

Quant à "**JSON**", c'est l'acronyme de "JavaScript Object Notation". JSON est un format de données léger et facile à lire utilisé pour l'échange de données entre les applications. Il est basé sur une syntaxe JavaScript, mais il est indépendant du langage, ce qui signifie qu'il peut être utilisé avec de nombreux langages de programmation différents. Les données JSON sont souvent utilisées pour communiquer avec des API Web, stocker des configurations et échanger des données entre le client et le serveur dans les applications web.

Voici un exemple de données JSON :

```json
{
  "nom": "John Doe",
  "age": 30,
  "ville": "New York"
}
```

Ces données représentent un objet avec trois propriétés : "nom", "age" et "ville" :

En ce qui concerne les en-têtes HTTP spécifiés dans le code PHP ("Content-Type", "Access-Control-Allow-Origin", etc.), ce sont des instructions fournies dans la réponse HTTP pour indiquer comment le navigateur ou le client doit interpréter la réponse. Par exemple, "Content-Type" spécifie le type de contenu de la réponse (dans ce cas, "Application/json" indique que la réponse est au format JSON), tandis que les en-têtes "Access-Control-Allow-Origin", "Access-Control-Allow-Method", et "Access-Control-Allow-Headers" sont utilisés pour configurer les politiques de partage de ressources inter-origines (CORS) pour permettre ou restreindre l'accès à la ressource depuis différents domaines.