Wiam Laziz dd202

Ex :

Question 1: La commande pour créer un nouveau projet Laravel est **laravel new [nom-du-projet]**.

Question 2: Pour passer des données à une vue dans Laravel, on utilise la méthode **with()** ou en passant un tableau associatif à la méthode **view()**.

Question 3: Le moteur de template utilisé par Laravel est Blade.

Question 4: On peut valider les données de requête dans un contrôleur en utilisant la méthode **validate()** ou en définissant des règles de validation dans la méthode **rules()** du contrôleur.

Question 5: La commande pour créer un modèle est **php artisan make:model [NomDuModele]**.

Question 6: Pour ajouter une colonne à une table existante via une migration, on utilise la commande **php artisan make:migration [nom-de-la-migration] --table=[nom-de-la-table]**.

Question 8: Pour récupérer toutes les données d'une table avec Eloquent, on utilise la méthode **all()** sur le modèle correspondant.

Question 9: La commande pour créer un middleware est **php artisan make:middleware [NomDuMiddleware]**.

Question 10: Pour définir une route qui accepte n'importe quelle méthode HTTP, on utilise la méthode **any()** dans le fichier de définition des routes.

Question 11: Pour accéder à l'instance de la requête actuelle dans un contrôleur, on peut injecter la classe **Request** dans la méthode du contrôleur.

Question 12: Pour limiter le nombre de requêtes qu'un utilisateur peut faire, on peut utiliser la méthode **throttle()** dans les routes ou les contrôleurs.

Question 13: Pour envoyer un email en utilisant Laravel, on utilise la classe **Mail** et ses méthodes pour composer et envoyer des emails.

Question 14: La commande pour lancer le serveur de développement de Laravel est **php artisan serve**.

Question 15: Pour accéder aux paramètres d'URL dans une route, on utilise la méthode **request()** ou on injecte la classe **Request** dans la méthode de la route. Question 16: Pour ajouter une contrainte de clé étrangère dans une migration, on utilise la méthode **foreign()** dans la définition de la colonne.

Question 17: Pour créer une tâche planifiée (scheduled task) dans Laravel, on utilise la commande **php artisan make:command [NomDeLaCommande]** pour créer une nouvelle commande artisan, puis on définit la logique de la tâche planifiée dans la méthode **handle()** de la commande.

Question 18: La méthode qui permet de valider une requête entrante et de rediriger automatiquement en cas d'erreur est **validate()** dans les contrôleurs.

Question 19: Pour créer une relation un-à-plusieurs dans Eloquent, on définit une méthode sur le modèle parent qui retourne le résultat de **hasMany()** ou **hasOne()** pour définir la relation avec le modèle enfant.

Question 20: La méthode Eloquent utilisée pour charger les relations d'un modèle de manière conditionnelle est **load()**.

Question 21: Pour ajouter un attribut muté à un modèle Eloquent, on définit une méthode dans le modèle avec le préfixe **get** suivi du nom de l'attribut et suffixé par **Attribute**.

Question 24: Pour générer une clé d'application dans Laravel, on utilise la commande **php artisan key:generate**.

Question 25: La commande pour créer un événement et son listener associé est **php artisan make:event [NomDeLEvenement]** et **php artisan make:listener [NomDuListener] --event=[NomDeLEvenement]**.

Question 27: Le fichier de configuration qui détermine les services externes que l'application utilise est **config/services.php**.

Question 28: Pour utiliser les slots et les composants dans les vues Blade, on utilise la syntaxe **@slot** pour définir un slot et **@component** pour définir un composant.

Question 29: Un trait dans Laravel est une collection de méthodes qui peuvent être réutilisées dans plusieurs classes. On peut l'utiliser en l'incluant dans une classe à l'aide du mot-clé **use**.

Question 30: Pour définir un accesseur dans un modèle Eloquent, on crée une méthode dans le modèle avec le préfixe **get** suivi du nom de l'attribut et suffixé par **Attribute**.

Question 31: Pour utiliser les transactions de base de données dans Laravel, on utilise les méthodes **beginTransaction()**, **commit()** et **rollback()** sur l'objet de connexion à la base de données.

Question 32: Pour crypter les données avant de les sauvegarder dans la base de données, on utilise la méthode **encrypt()** dans les mutateurs des modèles Eloquent ou la méthode **Crypt::encrypt()** manuellement.

Question 37: Pour configurer Laravel pour utiliser un système de files d'attente différent, on modifie le fichier de configuration **queue.php** pour utiliser le pilote de file d'attente désiré.

Question 40: Pour modulariser une application Laravel en utilisant des packages, vous pouvez suivre ces étapes :

1. Créez un nouveau package en utilisant la commande Artisan : **php artisan make:package NomDuPackage**.
2. Organisez votre code dans le package en suivant les conventions de Laravel.
3. Publiez votre package en utilisant la commande Artisan : **php artisan vendor:publish --provider="NomDuFournisseur" --tag="public"**.
4. Installez votre package dans votre application Laravel en utilisant Composer : **composer require nom-du-package**.
5. Utilisez les fonctionnalités fournies par votre package dans votre application en les appelant comme vous le feriez avec n'importe quelle autre classe ou méthode.