QCM LARAVEL : ZIRI SAMI DD202

1. La commande pour créer un nouveau projet Laravel est laravel new 1 ou composer create-projet Laravel/laravel.

2. Pour passer des données à une vue, nous utilisons la méthode with sur le retour de la vue. Par exemple : return view (‘nom_de_la_vue’) -> with (‘nom_variable’, $variable) ;

3. Laravel utilise le moteur de template Blade pour la création de vues.

4. Pour valider les données de requête dans un contrôleur, nous utilisons la méthode validate disponible dans les contrôleurs de Laravel, qui utilise les règles de validation définies dans les classes de formulaire de requête ou directement dans le contrôleur.

5. La commande pour créer un modèle dans Laravel est php artisan make : model NomDuModele .

6. Pour ajouter une colonne à une table existante via une migration, nous créons une nouvelle migration avec php artisan make : migration add_column_to_table –table=nom_de_la_table et dans la méthode up de la migration, nous utilisons $table->addColumn(‘type’, ‘nom_de_colonne’) ;

7. Pour faire une redirection vers une autre route, nous utilisons la méthode redirect dans un contrôleur, par exemple : return redirect() ->route(‘nom_de_la_route’) ;

8. Pour récupérer toutes les données d’une table avec Eloquent, nous pouvons utiliser la méthode all sur le modèle correspondant, par exemple : User: :all() ;

9. La commande pour créer un middleware est php artisan make : middleware NomDuMiddleware.

10. Pour définir une route qui accepte n’importe quelle méthode http, nous utilisons la méthode Route: :any(‘/chemin’, ‘NomDuController@method’) ; dans le fichier routes/web.php.

11. Pour accéder à l’instance de la requête actuelle dans un contrôleur, nous utilisons la méthode request() ou injectez la classe Illuminate\Http\Request dans la méthode du contrôleur.

12. La méthode qui permet de limiter le nombre de requêtes qu’un utilisateur peut faire est throttle dans le fichier de routes.

13. Pour envoyer un email en utilisant Laravel, vous utili le facade Mail et la méthode send en passant le nom de la classe Mailable.

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

15. Pour accéder aux paramètres d'URL dans une route, nous pouvons utiliser la fonction route dans une vue pour générer des URLs ou utiliser des variables de route dans les méthodes de contrôleur.

16. Pour ajouter une contrainte de clé étrangère dans une migration, nous utilisons la méthode foreign sur l'objet $table dans la méthode up de la migration, par exemple : $table->foreign('nom_de_la_colonne') ->references('nom_de_la_colonne_reference') ->on('nom_de_la_table_reference') ;

17. Pour créer une tâche planifiée (scheduled task) dans Laravel, nous utilisons la commande php artisan schedule :make pour générer une tâche planifiée et ajoutez votre logique dans le fichier app/Console/Kernel.php.

18. La méthode qui permet de valider une requête entrante et rediriger automatiquement en cas d'erreur est validate dans un contrôleur.

19. Pour créer une relation un-à-plusieurs dans Eloquent, nous utilisons la méthode hasMany dans le modèle parent et la méthode belongsTo dans le modèle enfant.

20. Pour charger les relations d'un modèle de manière conditionnelle avec Eloquent, nous utilisons la méthode load sur une instance de modèle, par exemple : $user->load('relation') ;

21. Pour ajouter un attribut muté à un modèle Eloquent, vous ajoutez une propriété $fillable ou $guarded dans le modèle.

22. Le Tinker est un REPL (Read Eval Print Loop) intégré à Laravel qui permet de tester du code Laravel en direct, sans avoir à créer une application complète.

23. Pour publier les assets d'un package, nous utilisons la commande php artisan vendor :publish –tag=public –provider= »NomDuProvider ».

24. Pour générer une clé d'application dans Laravel, nous utilisons la commande php artisan key :generate.

25. La commande pour créer un événement et son listener associé est php artisan event :generate NomDeLEvenement.

26. Pour implémenter une stratégie de cache personnalisée, nous créons une classe qui implémente l'interface Illuminate\Contracts\Cache\Store et la configurez dans le fichier config/cache.php.

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

28. Pour utiliser les slots et les composants dans les vues Blade, nous utilisons la syntaxe @slot('nom_du_slot') pour définir un slot et @component('nom_du_composant') pour inclure un composant .

29. Un trait dans Laravel est une manière de réutiliser du code entre les classes. Nous pouvons l'utiliser en ajoutant use NomDuTrait ; dans votre classe .

30. Pour définir un accesseur dans un modèle Eloquent, nous créons une méthode publique dans le modèle avec le nom de la propriété suivi de getAccessorAttribute .

31. Pour utiliser les transactions de base de données dans Laravel, nous utilisons la méthode transaction sur l'instance de la base de données ou le modèle Eloquent, par exemple : DB ::transaction(function () { /* code transactionnel */ }) ;

32. La méthode pour crypter les données avant de les sauvegarder dans la base de données est encrypt .

33. Pour utiliser Laravel Mix pour compiler les assets, nous utilisons les commandes npm run dev pour le développement, npm run prod pour la production, et npm run watch pour compiler automatiquement en mode développement .

34. Pour créer une réponse JSON dans un contrôleur, nous utilisons la méthode response()->json($data) ;

35. Pour utiliser les événements de modèle dans Laravel, nous utilisons des méthodes spécifiques dans le modèle Eloquent comme creating, created, updating, updated, etc., pour définir des comportements à exécuter avant ou après des actions spécifiques.