Q1. What is Laravel pattern?

Laravel follows the Model-View-Controller (MVC) architectural pattern. This design pattern separates the application logic into three main components: Models, Views, and Controllers.

Q2. What is M, C, V?

* **M (Model):** Represents the data layer and business logic of the application.
* **C (Controller):** Handles the user's input and interaction, processes it, and returns the appropriate output.
* **V (View):** Represents the presentation layer, displaying the data to the user.

Q3. What is query builder?

The query builder in Laravel provides a convenient, fluent interface for creating and running database queries. It can be used to perform most database operations in your application and works on all supported database systems.

Q4. What is Laravel Eloquent?

Eloquent is Laravel's ORM (Object-Relational Mapper) that allows you to interact with your database using an object-oriented syntax. Each database table has a corresponding "Model" which is used to interact with that table.

Q5. What is .env file?

The .env file in a Laravel application is used to store environment-specific configuration variables. This allows for easy configuration of the application without hardcoding sensitive information or settings into the codebase.

Q6. What is migrations?

Migrations in Laravel are like version control for your database, allowing your team to define and share the application's database schema definition. Migrations are typically paired with Laravel's schema builder to easily build your application's database schema.

Q7. What is Seeder?

Seeders in Laravel are classes that allow you to populate your database with dummy data. They are useful for testing and developing the application in different environments with pre-defined data sets.

Q8. What is file.blade.php?

A file with the .blade.php extension is a Blade template file in Laravel. Blade is Laravel's simple, yet powerful templating engine which provides convenient syntax for writing HTML and PHP code in a clean and readable way.

Q9. What is Factory?

Factories in Laravel are used to define a model's default set of attributes. They are typically used in testing and database seeding to generate large amounts of data quickly.

Q10. What is API?

API stands for Application Programming Interface. It allows different software systems to communicate with each other. In the context of Laravel, an API is typically a set of routes and controllers that expose application functionality to other systems over HTTP.

Q11. What is endpoint, method, request-body, handle error validation and respond?

* **Endpoint:** The URL at which an API can be accessed.
* **Method:** The HTTP method used for the request (GET, POST, PUT, DELETE, etc.).
* **Request Body:** The data sent with the request, typically used in POST and PUT requests.
* **Handle Error Validation:** The process of checking the request data for errors and ensuring it meets the expected format and constraints.
* **Respond:** The data sent back to the client, often in JSON format, after processing the request.

Q12. Which method to use when I want to create new Post?

To create a new Post, you should use the HTTP POST method.

Q13. Why you need to use Resource and Request and getAttribute?

* **Resource:** Transforms data into JSON and structures it in a way suitable for API responses.
* **Request:** Validates and sanitizes input data before it reaches the controller.
* **getAttribute:** Retrieves attributes from the model, useful for accessing data within models.

Q14. What is compact()?

The compact() function in PHP creates an array from variables and their values. It's often used to pass multiple variables to a view.

Q15. List down at least 10 query builder and Laravel Eloquent methods.

1. select()
2. where()
3. join()
4. orderBy()
5. groupBy()
6. having()
7. insert()
8. update()
9. delete()
10. get()

Q16. Why we need to use whereHas?

The whereHas method is used to filter results based on the existence of a relationship. It's useful for querying records that have a related record meeting certain criteria.

Q17. Please give example of Column Types at 10 differences, don't forget to description.

1. integer: A standard integer column.
2. string: A VARCHAR column with a specified length.
3. text: A TEXT column for large amounts of text.
4. boolean: A BOOLEAN column storing true/false values.
5. date: A DATE column storing date values.
6. datetime: A DATETIME column storing date and time values.
7. time: A TIME column storing time values.
8. timestamp: A TIMESTAMP column storing date and time values with time zone.
9. float: A FLOAT column storing decimal values.
10. json: A JSON column storing JSON encoded data.

Q18. Can you describe this: $table->integer('age')->nullable()->default(0);

This command adds an integer column named age to the table. The column is nullable, meaning it can hold a null value, and its default value is set to 0.

Q19. Can you tell me why we use file api.php and web.php? Explain.

* **api.php:** Contains routes that are stateless and typically used for API endpoints.
* **web.php:** Contains routes that use session state, CSRF protection, and other web features.

Q20. What is relationship in Laravel? Please give example:

* **One to One:**
  + Models: User and Profile
  + Relationship: user()->hasOne(Profile::class)
* **One to Many:**
  + Models: Post and Comment
  + Relationship: post()->hasMany(Comment::class)
* **Many to Many:**
  + Models: User and Role
  + Relationship: user()->belongsToMany(Role::class)

Q21. Can you explain why we need to use Route::apiResource in route api and Route::resource in route web?

* **Route::apiResource:** Creates routes for a RESTful resource controller without routes for create and edit, as these are typically handled by the frontend.
* **Route::resource:** Creates routes for a resource controller including create and edit, suitable for web interfaces.

Q22. Can you explain this command "php artisan route

**" and "php artisan install**

**"?**

* **php artisan route:list:** Lists all registered routes in the application, including their methods, URIs, and associated actions.
* **php artisan install:api:** (Note: This command is not a standard Laravel command and may be specific to certain packages or custom commands.)

Q23. Can you explain me about redirect and route name?

* **Redirect:** Sends the user to a different URL.
* **Route Name:** Assigns a name to a route, allowing you to generate URLs or redirects using the route name instead of the URI.

Q24. Please clean code using Route Prefixes.

php

Copy code

Route::prefix('product')->name('product.')->group(function () {

Route::get('list', [ProductController::class, 'index'])->name('list');

Route::post('create', [ProductController::class, 'store'])->name('create');

Route::get('show/{id}', [ProductController::class, 'show'])->name('show');

Route::put('update/{id}', [ProductController::class, 'update'])->name('update');

Route::delete('delete/{id}', [ProductController::class, 'destroy'])->name('destroy');

});

Q25. What is the difference between "php artisan migrate" and "php artisan migrate

**"?**

* **php artisan migrate:** Runs all pending migrations.
* **php artisan migrate:fresh:** Drops all tables and re-runs all migrations from scratch.

Q26. Why do we need to use this?

php

Copy code

protected $fillable = [

'name',

'email',

'password',

];

The $fillable property specifies which attributes should be mass-assignable. This helps protect against mass assignment vulnerabilities.

Q27. Why do we need to use this?

php

Copy code

protected $hidden = [

'password',

'remember\_token',

];

The $hidden property hides attributes when the model is converted to an array or JSON, protecting sensitive information from being exposed.

Q28. Why do we need to use this?

php

Copy code

protected $casts = [

'email\_verified\_at' => 'datetime',

];

The $casts property casts attributes to a specific data type. In this case, it ensures the email\_verified\_at attribute is always treated as a datetime object.

Q29. Why do we need to use SoftDeletes?

SoftDeletes allows for soft deleting records, meaning they are not actually removed from the database but marked as deleted. This allows for easy recovery of deleted records.

Q30. What is file composer.json?

The composer.json file is used to manage project dependencies in PHP. It specifies the packages required for the project and can include metadata like the project's name, description, authors, and more.

Q31. What is README.md?

The README.md file provides an introduction and documentation for the project. It is written in Markdown and typically includes instructions for setup, usage, and contribution guidelines.