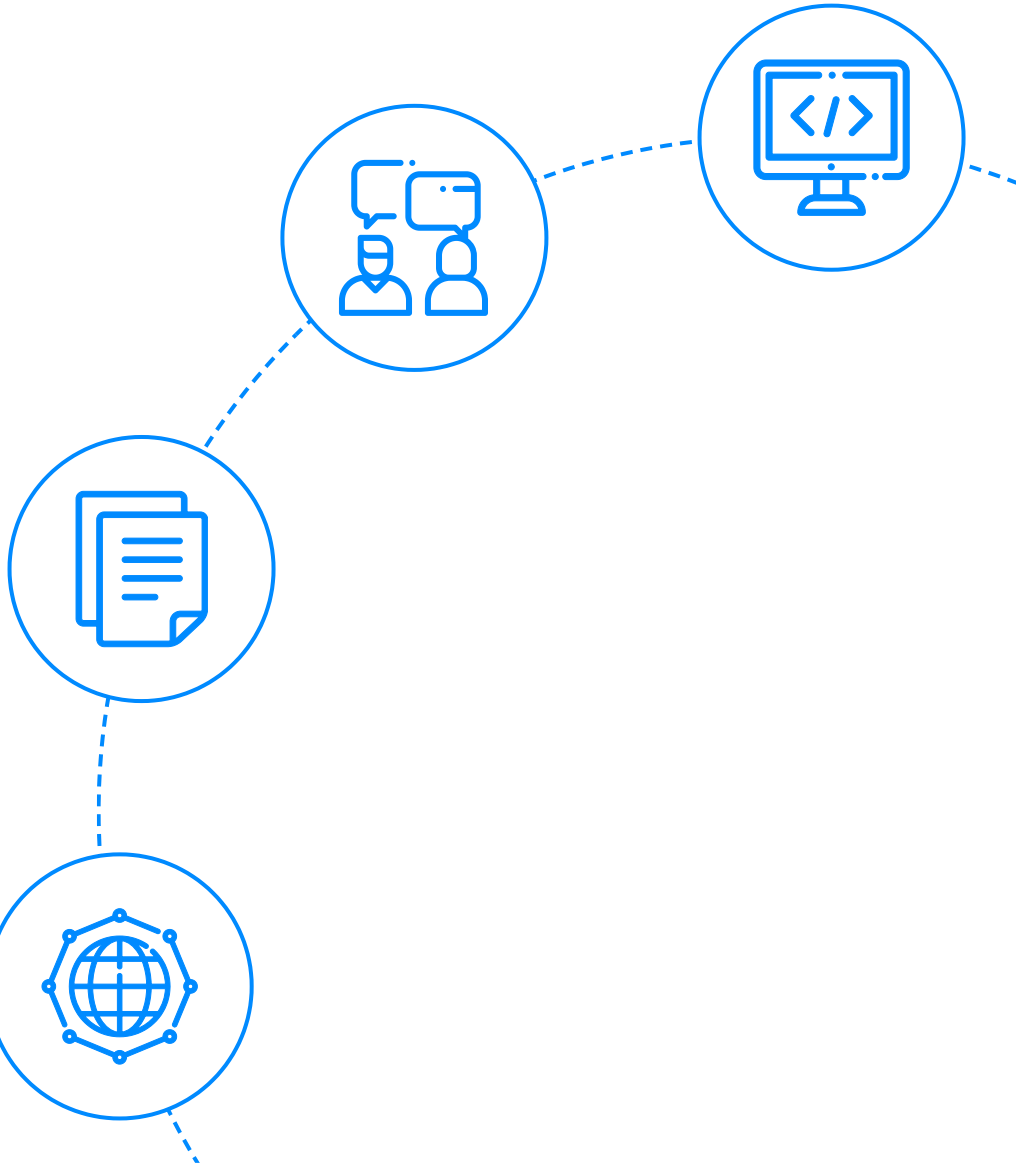
InterviewBit

Laravel Interview Questions



To view the live version of the page, click here.

Contents

Laravel Interview Questions For Freshers

Advanced Laravel Interview Questions

Advanced Laravel Interview Questions

(.....Continued)

19. What are Events in Laravel?
20. Explain logging in Laravel?
21. What is Localization in Laravel?
22. What are Requests in Laravel?
23. How to do request validation in Laravel?
24. What is a Service Container in Laravel?
25. What is a Service Provider?
26. What is the register and boot method in the Service Provider class?
27. How to define routes in Laravel?
28. What are named routes?
29. What are route groups?
30. What is Middleware and how to create one in Laravel?
31. How to create a route for resources in laravel?
32. What is dependency Injection in Laravel?
33. What are collections?
34. What are contracts?
35. What are queues in Laravel?
36. What are accessors and mutators?

Let's get Started

What is Laravel Framework?

Laravel is an open-source PHP web application framework. It is a very well documented, expressive, and easy to learn framework. Laravel is very developer-friendly as the framework can help beginners as well as advanced users. As you grow as a developer you can go more deep into Laravel functionalities and give more robust and enterprise solutions.

Additionally, the framework is very scalable as you can use packages like Vapor to handle hundreds of thousands of requests using AWS serverless technology.

This article will walk you through basic Laravel interview questions to advanced questions.

Laravel Interview Questions For Freshers

1. What is the latest Laravel version?

The latest Laravel version is 8.x.

2. Define Composer.

Composer is the package manager for the framework. It helps in adding new packages from the huge community into your laravel application.

For example, one of the most used packages for authentication will be Passport, for including that into your project, you can run the below command on your terminal:

```
composer requires laravel/passport
```

It generates a file(composer.json) in your project directory to keep track of all your packages. A default composer.json file of your laravel project will look something like below:

```
{
    "name": "laravel/laravel",
    "type": "project",
    "description": "The Laravel Framework.",
    "keywords": [
        "framework",
        "laravel"
    ],
    "license": "MIT",
    "require": {
        "php": "^7.3|^8.0",
        "fideloper/proxy": "^4.4",
        "fruitcake/laravel-cors": "^2.0",
        "guzzlehttp/guzzle": "^7.0.1",
        "laravel/framework": "^8.12",
        "laravel/tinker": "^2.5"
    },
    "require-dev": {
        "facade/ignition": "^2.5",
        "fakerphp/faker": "^1.9.1",
        "laravel/sail": "^1.0.1",
        "mockery/mockery": "^1.4.2",
        "nunomaduro/collision": "^5.0",
        "phpunit/phpunit": "^9.3.3"
    }
}
```

The “require” and “require-dev” keys in composer.json specify production and dev packages and their version constraints respectively.

3. What is the templating engine used in Laravel?

The templating engine used in Laravel is **Blade**. The blade gives the ability to use its mustache-like syntax with the plain PHP and gets compiled into plain PHP and cached until any other change happens in the blade file. The blade file has .blade.php extension.

4. What are available databases supported by Laravel?

The supported databases in laravel are:

- PostgreSQL
- SQL Server
- SQLite
- MySQL

5. What is an artisan?

Artisan is the command-line tool for Laravel to help the developer build the application. You can enter the below command to get all the available commands:

PHP artisan list: Artisan command can help in creating the files using the make command. Some of the useful make commands are listed below:

`php artisan make:controller` - Make Controller file

`php artisan make:model` - Make a Model file

`php artisan make:migration` - Make Migration file

`php artisan make:seeder` - Make Seeder file

`php artisan make:factory` - Make Factory file

`php artisan make:policy` - Make Policy file

`php artisan make:command` - Make a new artisan command

6. How to define environment variables in Laravel?

The environment variables can be defined in the .env file in the project directory. A brand new laravel application comes with a .env.example and while installing we copy this file and rename it to .env and all the environment variables will be defined here.

Some of the examples of environment variables are APP_ENV, DB_HOST, DB_PORT, etc.

7. Can we use Laravel for Full Stack Development (Frontend + Backend)?

Laravel is the best choice to make progressive, scalable full-stack web applications. Full-stack web applications can have a backend in laravel and the frontend can be made using blade files or SPAs using Vue.js as it is provided by default. But it can also be used to just provide rest APIs to a SPA application.

Hence, Laravel can be used to make full-stack applications or just the backend APIs only.

8. How to put Laravel applications in maintenance mode?

Maintenance mode is used to put a maintenance page to customers and under the hood, we can do software updates, bug fixes, etc. Laravel applications can be put into maintenance mode using the below command:

```
php artisan down
```

And can put the application again on live using the below command:

```
php artisan up
```

Also, it is possible to access the website in maintenance mode by whitelisting particular IPs.

9. What are the default route files in Laravel?

Below are the four default route files in the routes folder in Laravel:

- web.php - For registering web routes.
- api.php - For registering API routes.
- console.php - For registering closure-based console commands.
- channel.php - For registering all your event broadcasting channels that your application supports.

10. What are migrations in Laravel?

In simple, Migrations are used to create database schemas in Laravel. In migration files, we store which table to create, update or delete.

Each migration file is stored with its timestamp of creation to keep track of the order in which it was created. As migrations go up with your code in GitHub, GitLab, etc, whenever anyone clones your project they can run `PHP artisan migrate` to run those migrations to create the database in their environment. A normal migration file looks like below:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
          // Create other columns
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

The up() method runs when we run `php artisan migrate` and down() method runs when we run `php artisan migrate:rollback`.

If we rollback, it only rolls back the previously run migration.

If we want to rollback all migrations, we can run 'php artisan migrate:reset`.

If we want to rollback and run migrations, we can run `PHP artisan migrate:refresh`, and we can use `PHP artisan migrate:fresh` to drop the tables first and then run migrations from the start.

11. What are seeders in Laravel?

Seeders in Laravel are used to put data in the database tables automatically. After running migrations to create the tables, we can run `php artisan db:seed` to run the seeder to populate the database tables.

We can create a new Seeder using the below artisan command:

```
php artisan make:seeder [className]
```

It will create a new Seeder like below:

```
<?php

use App\Models\Auth\User;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Seeder;

class UserTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        factory(User::class, 10)->create();
    }
}
```

The run() method in the above code snippet will create 10 new users using the User factory.

Factories will be explained in the next question.

12. What are factories in Laravel?

Factories are a way to put values in fields of a particular model automatically. Like, for testing when we add multiple fake records in the database, we can use factories to generate a class for each model and put data in fields accordingly. Every new laravel application comes with database/factories/UserFactory.php which looks like below:

```php
<?php

namespace Database\Factories;

use App\Models\User;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

class UserFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = User::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'name' => $this->faker->name,
            'email' => $this->faker->unique()->safeEmail,
            'email_verified_at' => now(),
            'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi'
            'remember_token' => Str::random(10),
        ];
    }
}
```

We can create a new factory using `php artisan make:factory UserFactory --class=User` .

The above command will create a new factory class for the User model. It is just a class that extends the base Factory class and makes use of the Faker class to generate fake data for each column. With the combination of factory and seeders, we can easily add fake data into the database for testing purposes.

13. How to implement soft delete in Laravel?

Soft Delete means when any data row is deleted by any means in the database, we are not deleting the data but adding a timestamp of deletion.

We can add soft delete features by adding a trait in the model file like below.

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model {

    use SoftDeletes;

    protected $table = 'posts';

    // ...
}
```

14. What are Models?

With Laravel, each database table can have a model representation using a model file which can be used to interact with that table using Laravel Eloquent ORM.

We can create a model using this artisan command:

```
php artisan make:model Post
```

This will create a file in the models' directory and will look like below:

```
class Post extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [];
}
```

A Model can have properties like table, fillable, hidden, etc which defines properties of the table and model.

Advanced Laravel Interview Questions

15. What are Relationships in Laravel?

Relationships in Laravel are a way to define relations between different models in the applications. It is the same as relations in relational databases.

Different relationships available in Laravel are:

- One to One
- One to Many
- Many to Many
- Has One Through
- Has Many Through
- One to One (Polymorphic)
- One to Many (Polymorphic)
- Many to Many (Polymorphic)

Relationships are defined as a method on the model class. An example of One to One relation is shown below.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the phone associated with the user.
     */
    public function phone()
    {
        return $this->hasOne(Phone::class);
    }
}
```

The above method phone on the User model can be called like : `$user->phone` or `$user->phone()->where(...)->get()`.

We can also define One to Many relationships like below:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the addresses for the User.
     */
    public function addresses()
    {
        return $this->hasMany(Address::class);
    }
}
```

Since a user can have multiple addresses, we can define a One to Many relations between the User and Address model. Now if we call `$user->addresses`, eloquent will make the join between tables and it will return the result.

16. What is Eloquent in Laravel?

Eloquent is the ORM used to interact with the database using Model classes. It gives handy methods on class objects to make a query on the database.

It can directly be used to retrieve data from any table and run any raw query. But in conjunction with Models, we can make use of its various methods and also make use of relationships and attributes defined on the model.

Some examples of using the Eloquent are below:

- `DB::table('users')->get()`
- `User::all()`
- `User::where('name', '=', 'Eloquent')->get()`

17. What is throttling and how to implement it in Laravel?

Throttling is a process to rate-limit requests from a particular IP. This can be used to prevent DDOS attacks as well. For throttling, Laravel provides a middleware that can be applied to routes and it can be added to the global middlewares list as well to execute that middleware for each request.

Here's how you can add it to a particular route:

```
Route::middleware('auth:api', 'throttle:60,1')->group(function () {
    Route::get('/user', function () {
        //
    });
});
```

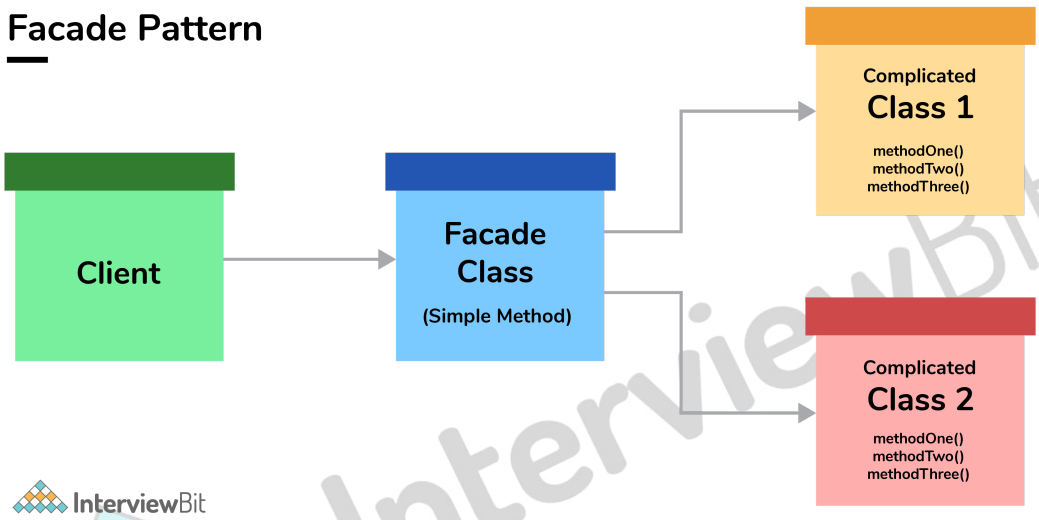
This will enable the /user route to be accessed by a particular user from a particular IP only 60 times in a minute.

18. What are facades?

Facades are a way to register your class and its methods in Laravel Container so they are available in your whole application after getting resolved by Reflection.

The main benefit of using facades is we don't have to remember long class names and also don't need to require those classes in any other class for using them. It also gives more testability to the application.

The below image could help you understand why Facades are used for:



Facades in Laravel

19. What are Events in Laravel?

In Laravel, Events are a way to subscribe to different events that occur in the application. We can make events to represent a particular event like user logged in, user logged out, user-created post, etc. After which we can listen to these events by making Listener classes and do some tasks like, user logged in then make an entry to audit logger of application.

For creating a new Event in laravel, we can call below artisan command:

```
php artisan make:event UserLoggedIn
```

This will create a new event class like below:

```php
<?php

namespace App\Events;

use App\Models\User;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;

class UserLoggedIn
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    /**
     * The user instance.
     *
     * @var \App\Models\User
     */
    public $user;

    /**
     * Create a new event instance.
     *
     * @param  \App\Models\User  $user
     * @return void
     */
    public function __construct(User $user)
    {
        $this->user = $user;
    }
}
```

For this event to work, we need to create a listener as well. We can create a listener like this:

```
php artisan make:listener SetLogInFile --event=UserLoggedIn
```

The below resultant listener class will be responsible to handle when the UserLoggedIn event is triggered.

```
use App\Events\UserLoggedIn;

class SetLogInFile
{
    /**
     * Handle the given event.
     *
     * @param  \App\Events\UserLoggedIn
     * @return void
     */
    public function handle(UserLoggedIn $event)
    {
        //
    }
}
```

20. Explain logging in Laravel?

Laravel Logging is a way to log information that is happening inside an application. Laravel provides different channels for logging like file and slack. Log messages can be written on to multiple channels at once as well.

We can configure the channel to be used for logging in to our environment file or in the config file at config/logging.php.

21. What is Localization in Laravel?

Localization is a way to serve content concerning the client's language preference. We can create different localization files and use a laravel helper method like this: `__('auth.error')` to retrieve translation in the current locale. These localization files are located in the resources/lang/[language] folder.

22. What are Requests in Laravel?

Requests in Laravel are a way to interact with incoming HTTP requests along with sessions, cookies, and even files if submitted with the request.

The class responsible for doing this is Illuminate\Http\Request.

When any request is submitted to a laravel route, it goes through to the controller method, and with the help of dependency Injection, the request object is available within the method. We can do all kinds of things with the request like validating or authorizing the request, etc.

23. How to do request validation in Laravel?

Request validation in laravel can be done with the controller method or we can create a request validation class that holds the rules of validation and the error messages associated with it.

One example of it can be seen below:

```
/**
 * Store a new blog post.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|unique:posts|max:255',
        'body' => 'required',
    ]);

    // The blog post is valid...
}
```

24. What is a Service Container in Laravel?

Service Container or IoC in laravel is responsible for managing class dependencies meaning not every file needs to be injected in class manually but is done by the Service Container automatically. Service Container is mainly used in injecting class in controllers like Request object is injected. We can also inject a Model based on id in route binding.

For example, a route like below:

```
Route::get('/profile/{id}', 'UserController@profile');
```

With the controller like below.

```
public function profile(Request $request, User $id)
{
    //
}
```

In the UserController profile method, the reason we can get the User model as a parameter is because of Service Container as the IoC resolves all the dependencies in all the controllers while booting the server. This process is also called route-model binding.

25. What is a Service Provider?

A Service Provider is a way to bootstrap or register services, events, etc before booting the application. Laravel's own bootstrapping happens using Service Providers as well. Additionally, registers service container bindings, event listeners, middlewares, and even routes using its service providers.

If we are creating our application, we can register our facades in provider classes.

26. What is the register and boot method in the Service Provider class?

The register method in the Service Provider class is used to bind classes or services to the Service Container. It should not be used to access any other functionality or classes from the application as the service you are accessing may not have loaded yet into the container.

The boot method runs after all the dependencies have been included in the container and now we can access any functionality in the boot method. Like you can create routes, create a view composer, etc in the boot method.

27. How to define routes in Laravel?

Laravel Routes are defined in the routes file in routes/web.php for web application routes. Routes can be defined using Illuminate\Support\Facades\Route and calling its static methods such as to get, post, put, delete, etc.

```
use Illuminate\Support\Facades\Route;

Route::get('/home', function () {
    return 'Welcome to Home Sweet Home';
});
```

A typical closure route looks like the above, where we provide the URI and the closure function to execute when that route is accessed.

```
Route::get('/hello', 'HomeController@index');
```

Another way is like above, we can directly give the controller name and the method to call, this can again be resolved using Service Container.

28. What are named routes?

A named route is a route definition with the name assigned to it. We can then use that name to call the route anywhere else in the application.

```
Route::get('/hello', 'HomeController@index')->name('index');
```

This can be accessed in a controller using the following:

```
return redirect()->route('index');
```

29. What are route groups?

Route Groups in laravel is used when we need to group route attributes like middlewares, prefixes, etc. we use route groups. It saves us a headache to put each attribute to each route.

Syntax:

```
Route::middleware(['throttleMiddleware'])->group(function () {
    Route::get('/', function () {
        // Uses throttleMiddleware
    });

    Route::get('/user/profile', function () {
        // Uses throttleMiddleware
    });
});
```

30. What is Middleware and how to create one in Laravel?

Middleware gives developers the ability to inspect and filter incoming HTTP requests of our application. One such middleware that ships with laravel are the authentication middleware which checks if the user is authenticated and if the user is authenticated it will go further in the application otherwise it will throw the user back to the login screen.

We can always create a new middleware for our purposes. For creating a new middleware we can use the below artisan command:

```
php artisan make:middleware CheckFileIsNotTooLarge
```

The above command will create a new middleware file in the app/Http/Middleware folder.

31. How to create a route for resources in laravel?

For creating a resource route we can use the below command:

```
Route::resource('blogs', BlogController::class);
```

This will create routes for six actions index, create, store, show, edit, update and delete.

32. What is dependency Injection in Laravel?

The Laravel Service Container or IoC resolves all of the dependencies in all controllers. So we can type-hint any dependency in controller methods or constructors. The dependency in methods will be resolved and injected in the method, this injection of resolved classes is called dependency Injection.

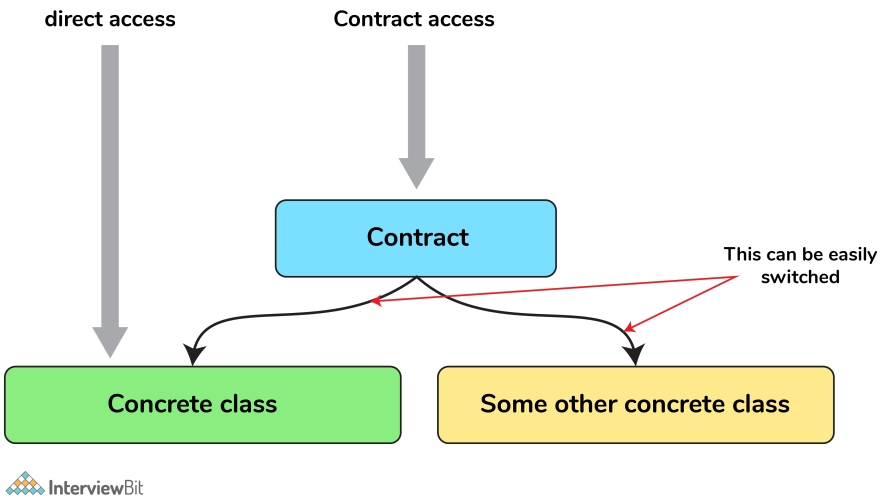
33. What are collections?

Collections in laravel are a wrapper over an array of data in Laravel. All of the responses from Eloquent ORM when we query data from the database are collections (Array of data records).

Collections give us handy methods over them to easily work with the data like looping over data or doing some operation on it.

34. What are contracts?

Laravel Contracts are a set of interfaces with implementation methods to complete the core tasks of Laravel.



Laravel Contracts

Few examples of contracts in Laravel are Queue and Mailer. Queue contract has an implementation of Queuing jobs while Mailer contract has an implementation to send emails.

35. What are queues in Laravel?

While building any application we face a situation where some tasks take time to process and our page gets loading until that task is finished. One task is sending an email when a user registers, we can send the email to the user as a background task, so our main thread is responsive all the time. Queues are a way to run such tasks in the background.

36. What are accessors and mutators?

Accessors are a way to retrieve data from eloquent after doing some operation on the retrieved fields from the database. For example, if we need to combine the first and last names of users but we have two fields in the database, but we want whenever we fetch data from eloquent queries these names need to be combined.

We can do that by creating an accessor like below:

```
public function getFullNameAttribute()
{
    return $this->first_name . " " . $this->last_name;
}
```

What the above code will do is it will give another attribute(full_name) in the collection of the model, so if we need the combined name we can call it like this: `$user->full_name`. Mutators are a way to do some operations on a particular field before saving it to the database.

For example, if we wanted the first name to be capitalized before saving it to the database, we can create something like the below:

```
public function setFirstNameAttribute($value)
{
    $this->attributes['first_name'] = strtoupper($value);
}
```

So, whenever we are setting this field to be anything:

```
$user->first_name = Input::get('first_name');
$user->save();
```

It will change the first_name to be capitalized and it will save to the database.

Conclusion

Laravel is the most used PHP framework for web development. Laravel interviews consist of questions about a deep understanding of PHP MVC architecture and app development basics like routes, controllers, views, and advanced topics such as Service Container, Dependency Injection, Accessors & Mutators, etc.

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions