Laravel Interview Questions & Answers

Basic Level Questions

1. What is Laravel?

Answer: Laravel is a free, open-source PHP web framework created by Taylor Otwell. It follows the Model-View-Controller (MVC) architectural pattern and provides elegant syntax, built-in features like routing, authentication, sessions, and caching. Laravel aims to make development enjoyable and creative by easing common tasks used in web projects.

2. What are the key features of Laravel?

Answer:

- Eloquent ORM (Object-Relational Mapping)
- Blade Templating Engine
- Artisan Command Line Interface
- Database Migration and Schema Builder
- Middleware
- Authentication and Authorization
- Routing
- Session Management
- Validation
- Caching
- Queue Management

3. What is Artisan in Laravel?

Answer: Artisan is Laravel's built-in command-line interface that provides helpful commands for developing applications. It can generate boilerplate code, manage database migrations, run tests, and perform various maintenance tasks.

Common Artisan commands:

```bash
```

```
php artisan make:controller UserController
php artisan make:model User
php artisan migrate
php artisan serve
php artisan route:list
```

4. Explain Laravel's directory structure

Answer:

- **app/**: Contains core application code (Models, Controllers, Middleware)
- **bootstrap/**: Contains app bootstrapping files
- **config/**: Configuration files
- **database/**: Database migrations, factories, and seeds
- **public/**: Entry point and public assets
- **resources/**: Views, raw assets, and language files
- **routes/**: Route definitions
- **storage/**: Logs, compiled templates, sessions, and cache
- **tests/**: Test files
- **vendor/**: Composer dependencies

5. What is MVC architecture in Laravel?

Answer:

- **Model**: Represents data and business logic (database interactions)
- **View**: Handles the presentation layer (user interface)
- **Controller**: Acts as intermediary between Model and View, handles user requests

Example:

```php
```

```php
// Model (User.php)
class User extends Model {
    protected $fillable = ['name', 'email'];
}

// Controller (UserController.php)
class UserController extends Controller {
    public function index() {
        $users = User::all();
        return view('users.index', compact('users'));
    }
}

// View (users/index.blade.php)
@foreach($users as $user)
    <p>{{ $user->name }}</p>
@endforeach
```

Intermediate Level Questions

6. What is Eloquent ORM?

Answer: Eloquent is Laravel's built-in Object-Relational Mapping (ORM) that provides an elegant ActiveRecord implementation for working with databases. Each database table has a corresponding "Model" that interacts with that table.

```php
// Creating a record
$user = new User();
$user->name = 'John Doe';
$user->email = 'john@example.com';
$user->save();

// Retrieving records
$users = User::all();
$user = User::find(1);
$users = User::where('active', 1)->get();
```

7. Explain Laravel Middleware

Answer: Middleware provides a convenient mechanism for filtering HTTP requests entering your application. It sits between the request and response, allowing you to examine and modify requests.

```php
// Creating middleware
php artisan make:middleware CheckAge

// In CheckAge middleware
public function handle($request, Closure $next)
{
    if ($request->age <= 18) {
        return redirect('home');
    }
    return $next($request);
}

// Registering in routes
Route::get('admin/profile', function () {
    //
})->middleware('auth', 'checkage');
```

8. What are Laravel Migrations?

Answer: Migrations are version control for your database schema. They allow you to define database structure in PHP code and share it with your team.

```php
```

```
// Creating migration
php artisan make:migration create_users_table

// In migration file
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamps();
    });
}

public function down()
{
    Schema::dropIfExists('users');
}
```

9. What is the difference between get() and first() methods?

Answer:

- **get()**: Returns a collection of all matching records
- **first()**: Returns only the first matching record as a model instance

```
php

$users = User::where('active', 1)->get(); // Collection
$user = User::where('active', 1)->first(); // Single model or null
```

10. Explain Laravel Relationships

Answer: Laravel supports various database relationships:

```
php
```

```php
// One-to-One
class User extends Model {
    public function phone() {
        return $this->hasOne(Phone::class);
    }
}

// One-to-Many
class Post extends Model {
    public function comments() {
        return $this->hasMany(Comment::class);
    }
}

// Many-to-Many
class User extends Model {
    public function roles() {
        return $this->belongsToMany(Role::class);
    }
}
```

Advanced Level Questions

11. What are Service Providers in Laravel?

Answer: Service Providers are the central place where Laravel applications are bootstrapped. They register services, bind classes into the service container, and configure various components.

```php
```

```
// Creating a service provider
php artisan make:provider PaymentServiceProvider

// In the provider
public function register()
{
    $this->app->bind('PaymentGateway', function () {
        return new StripePaymentGateway();
    });
}

public function boot()
{
    // Boot services after all providers registered
}
```

12. Explain Laravel's Service Container

Answer: The Service Container is a powerful tool for managing class dependencies and performing dependency injection. It automatically resolves dependencies and injects them into classes.

```
php

// Binding in service provider
$this->app->bind(UserRepositoryInterface::class, DatabaseUserRepository::class);

// Using in controller constructor
class UserController extends Controller {
    public function __construct(UserRepositoryInterface $userRepo) {
        $this->userRepo = $userRepo;
    }
}
```

13. What are Laravel Queues?

Answer: Queues allow you to defer time-consuming tasks (like sending emails) to be processed later, improving application response time.

```
php
```

```
// Creating a job
php artisan make:job SendEmailJob

// In the job class
class SendEmailJob implements ShouldQueue {
    public function handle() {
        // Send email logic
    }
}

// Dispatching the job
SendEmailJob::dispatch($user);
```

14. What is Laravel Caching?

Answer: Laravel provides a unified API for various caching systems (Redis, Memcached, database).

```
php

// Storing in cache
Cache::put('key', 'value', 60); // 60 minutes

// Retrieving from cache
$value = Cache::get('key');

// Cache with default value
$value = Cache::get('key', 'default');

// Remember method
$users = Cache::remember('users', 60, function () {
    return User::all();
});
```

15. Explain Laravel Events and Listeners

Answer: Events provide a simple observer implementation, allowing you to subscribe and listen for events in your application.

```
php
```

```php
// Creating event
php artisan make:event UserRegistered

// Event class
class UserRegistered {
    public $user;

    public function __construct(User $user) {
        $this->user = $user;
    }
}

// Creating listener
php artisan make:listener SendWelcomeEmail

// Firing event
event(new UserRegistered($user));
```

16. What are Laravel Policies?

Answer: Policies are classes that organize authorization logic around a particular model or resource.

```php
// Creating policy
php artisan make:policy PostPolicy

// In policy
public function update(User $user, Post $post) {
    return $user->id === $post->user_id;
}

// Using in controller
public function update(Post $post) {
    $this->authorize('update', $post);
    // Update logic
}
```

17. What is Laravel Sanctum?

Answer: Laravel Sanctum provides authentication for SPAs, mobile applications, and simple, token-based APIs. It allows users to generate multiple API tokens with specific scopes.

```php
// Creating token
$token = $user->createToken('API Token')->plainTextToken;

// Using middleware
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
```

18. Explain Database Seeding in Laravel

Answer: Seeding allows you to populate your database with test data using seed classes.

```php
// Creating seeder
php artisan make:seeder UserSeeder

// In seeder
public function run() {
    User::factory(50)->create();

    // Or manual seeding
    User::create([
        'name' => 'Admin User',
        'email' => 'admin@example.com'
    ]);
}

// Running seeders
php artisan db:seed
php artisan db:seed --class=UserSeeder
```

19. What are Laravel Factories?

Answer: Factories define how to generate fake data for models, useful for testing and seeding.

```php
```

```php
// Creating factory
php artisan make:factory UserFactory

// In factory
public function definition() {
    return [
        'name' => $this->faker->name(),
        'email' => $this->faker->unique()->safeEmail(),
        'password' => bcrypt('password')
    ];
}

// Using factory
$user = User::factory()->create();
$users = User::factory(10)->create();
```

20. How do you optimize Laravel application performance?

Answer:

- **Database optimization**: Use indexes, optimize queries, use eager loading
- **Caching**: Implement route, view, and data caching
- **Queue jobs**: Move time-consuming tasks to queues
- **Optimize Composer autoload**: Use `composer dump-autoload -o`
- **Use CDN**: For static assets
- **Enable OPcache**: For PHP optimization
- **Database query optimization**: Use `select()` to limit columns
- **Use Laravel Telescope**: For debugging and monitoring

```php
```

```php
// Eager loading to avoid N+1 problem
$posts = Post::with('comments', 'author')->get();

// Query optimization
$users = User::select(['id', 'name', 'email'])->where('active', 1)->get();

// Route caching
php artisan route:cache

// View caching
php artisan view:cache
```

Bonus Questions

21. What's new in Laravel 10?

Answer: Laravel 10 introduced:

- PHP 8.1 minimum requirement
- Laravel Pennant (Feature Flags)
- Process Interaction improvements
- Test Profiling
- Native type declarations in framework code
- Invokable validation rules

22. Explain Laravel Horizon

Answer: Horizon provides a dashboard and configuration system for Laravel's Redis-powered queues, offering real-time monitoring, job retry management, and metrics.

23. What is Laravel Vapor?

Answer: Laravel Vapor is a serverless deployment platform for Laravel applications powered by AWS Lambda, providing auto-scaling and pay-per-use pricing.