Day-3: Building News Platform with Laravel API - Filtering, Sorting, and Pagination

Prepared by: Rana M. Fakeeh

Application overview:

Our goal is to create a News Platform's CRUD APIs with the following entities:

- ***Category***: To categorize news articles.
- ***News***: To store news articles.
- ***Image***: To associate image album with news articles.
- ***User***: To manage user accounts.

Training program:

Day-1	Setup. CRUD for Category entity.
Day-2	CRUD for NewsArticle, Image entities. Upload image files to server. One-to-Many and Many-to-Many associations.
Day-3	Recap. Filtering, Sorting, and Pagination of list results.
Day-4	Laravel Passport authentication (OAuth2 token-based) Register, Login, Logout, and Profile APIs for User entity. Public vs. Protected endpoints.
Day-5	Integration with a ReactJS Frontend. Localization. Deployment (Docker).

Learning objectives:

- Apply filtering to resource collection.
- Apply sorting to resource collection.
- Apply pagination to resource collection.
- Hands-on.

Steps:

1- **Environment**
   - Run XAMP control panel.
   - Start Apache server
   - Start MySQL server
   - Open VS Code on previous project.
2- **News Module (Resource -> Controller): Filter by attribute using where() and whereBetween()**
   - To filter by ***visible*** attribute:
   - **Resource**
   - Open ***app/Http/Resources/NewsResource.php*** file.
   - Add the attribute to display it

```
-     'visible' => $this->resource->visible,
```

   - **Controller**
   - Open ***app/Http/Controllers/NewsController.php*** file.
   - The Eloquent ***all()*** method will return all of the results in the model's table.
   - To build queries instead, the ***query()*** method will return a query builder that allows you to add constraints on the query and then call ***get()*** method to retrieve the results.
   - To read attribute ***visible*** value from URL parameter ***visible***:

```
-         public function index(Request $request)
-         {
-             $articles = News::query();
-             // Filter by attribute
-             $visible = $request->input('visible');
-             if($visible) {
-                 $articles = $articles->where('visible' ,'=', $visible);
-             }.
-            $articles = $articles->get();
```

   - To filter by ***updated_at*** date attribute:
   - **Resource**
   - Open ***app/Http/Resources/NewsResource.php*** file.
   - Add the attribute to display it

```
-     ' updated_at' => $this->resource-> updated_at,
```

   - **Controller**
   - Open ***app/Http/Controllers/NewsController.php*** file.
   - To read start and end dates from URL parameters ***start_date*** and ***end_date***:

```
-             $start_date = $request->input('start_date');
-             $end_date = $request->input('end_date');
-             if ($start_date && $end_date) {
-                 $articles->whereBetween('updated_at', [$start_date, $end_date]);
-             }
```

3- News Module (Controller): Filter by relation using whereHas()

- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- To read category id from URL parameter ***category***:

```
-           // Filter by category
-           $category_id = $request->input('category');
-           if ($category_id) {
-               $articles->whereHas('categories', function($query) use($category_id) {
-                   $query->where('categories.id', $category_id);
-               });
-           }
```

- To read category id **array** from URL parameter ***category[]***:

```
-                   //$query->where('categories.id', $category_id);
-                   $query->whereIn('categories.id', $category_id);
```

4- . News Module (Resource -> Controller): Filter by a query string

- To filter by query string in either ***title*** or ***body*** date attribute:
- **Resource**
- Open ***app/Http/Resources/NewsResource.php*** file.
- Add the attribute to display it

```
-    ' body' => $this->resource-> body,
```

- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- To read query string from URL parameter ***q***:

```
-           // Filter by query string in title or body
-           $q = $request->input('q');
-           if ($q) {
-               $articles->where(function ($query) use ($q) {
-                   $query->where('title', 'LIKE', '%' . $q . '%')->orWhere('body', 'LIKE', '%' . $q . '%');
-               });
-           }
```

5- . News Module (Controller): Sort by field and direction

- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- To read query string from URL parameters ***sort_field*** and ***sort_direction***:

```
-           // Sorting
-           $sortField = $request->input('sort_field', 'id');
-           $sortDirection = $request->input('sort_direction', 'DESC');
-           $articles->orderBy($sortField, $sortDirection);
```

6- News Module (Controller): Pagination

- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- To read page size from URL parameter ***per_page*** or from default static variable ***$perPage***:

```
-           // Pagination
-           $perPage = $request->input('per_page', Controller::$perPage);
-           $articles = $articles->paginate($perPage);
-           $articles->appends($request->query());
-
-           //$articles = $articles->get();
-
-           return response()->json([
-              "status" => "success",
-              "error" => false,
-              //"data" => new NewsCollection($articles),
-              "data" => $articles,
-           ],200);
```

- The static variable ***$perPage*** above is defined in base controller class:
- Open ***app/Http/Controllers/Controller.php*** file.

```
-    public static $perPage = 5;
```

- the ***paginate()*** method is not provided for resource collection like eloquent collection, therefore, a specific ***macro*** can be defined for resource collection. A ***macro*** allows to add methods or behaviors to existing classes without directly modifying their source code.
- Open ***app/Providers/AppServiceProvider.php*** file.
- Add the code in Appendix-I.
- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- Update the ***NewsController*** as the following:

```
-           // Paginatable collection
-           $articles = new NewsCollection($articles->get());
-
-           // Pagination
-           $perPage = $request->input('per_page', Controller::$perPage);
-           $articles = $articles->paginate($perPage);
-           $articles->appends($request->query());
-
-           //$articles = $articles->get();
-
-           return response()->json([
-              "status" => "success",
-              "error" => false,
-              //"data" => new NewsCollection($articles),
-              "data" => $articles,
-           ],200);
```

- Another approach without using ***macro***:
- **Controller**
- Open ***app/Http/Controllers/NewsController.php*** file.
- Update the ***NewsController*** as the following:

```
-           // Paginatable collection
-           //$articles = new NewsCollection($articles->get());
-
-           // Pagination
-           $perPage = $request->input('per_page', Controller::$perPage);
-           $articles = $articles->paginate($perPage);
-           $articles->appends($request->query());
-
-           //$articles = $articles->get();
-
-           return response()->json([
-               "status" => "success",
-               "error" => false,
-               "data" => new NewsCollection($articles),
-               //"data" => $articles,
-               'next_page_url' => $articles->nextPageUrl(),
-               'prev_page_url' => $articles->previousPageUrl(),
-               'current_page' => $articles->currentPage(),
-               'last_page' => $articles->lastPage(),
-               'per_page' => $articles->perPage(),
-               'total' => $articles->total(),
-           ],200);
-
```

7- Testing API with Postman

- Apache server is started.
- MySQL server started.
- Migrate changes to database schema.

```
-   php artisan migrate
```

- Test the server running

```
-   php artisan serve
```

- Open Postman
- ***http://localhost:8000/api/category/****
- ***http://localhost:8000/api/news/****
- ***http://localhost:8000/api/album/****
- For create, use ***Body*** -> ***form-data***
- For update, use ***Body*** -> ***x-www-form-urlencoded***

Appendix-I

AppServiceProvider.php

```php
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Collection;
use Illuminate\Pagination\LengthAwarePaginator;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        /**
         * Paginate a standard Laravel Collection.
         *
         * @param int $perPage
         * @param int $total
         * @param int $page
         * @param string $pageName
         * @return array
         */
        Collection::macro('paginate', function($perPage, $total = null, $page =
null, $pageName = 'page'): LengthAwarePaginator {
            $page = $page ?: LengthAwarePaginator::resolveCurrentPage($pageName);

            return new LengthAwarePaginator(
                $this->forPage($page, $perPage)->values(),
                $total ?: $this->count(),
                $perPage,
```

```
                $page,
                [
                    'path' => LengthAwarePaginator::resolveCurrentPath(),
                    'pageName' => $pageName,
                ]
            );
        });
    }
}
```