CREATING VOTE QUESTION & ANSWER ENDPOINTS

In this lesson we're going to create api endpoints for voting question and answers. We're going to copy our existing controllers then make a bit changes on them and then make some tests in postman.

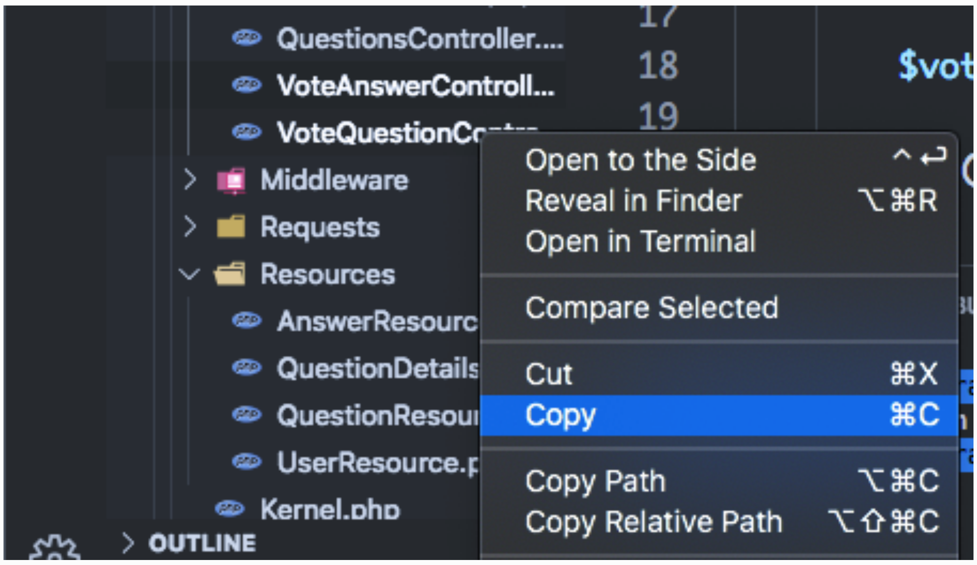
Alrigh, before we get started let's go ahead and open up our terminal. Then create a new branch to isolate our work today:

```
git checkout -b lesson-52
```

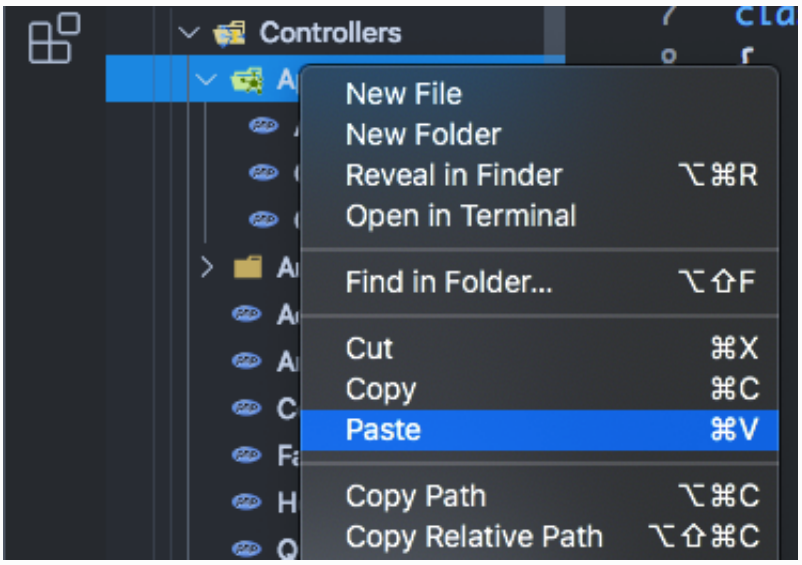
CREATE VOTE QUESTION & ANSWER ENDPOINTS

1. Copy & paste the `VoteAnswerController` and `VoteQuestionController``

Alright, let's go to `app/Http/Controllers` directory. Then copy `VoteAnswerController` as well as `VoteQuestionController`.



Go to `Api` folder and paste them inside.



Firstly let's open the `VoteAnswerController`. Then at the top let's adjust the controller namespace to `Api` like so.

```
namespace App\Http\Controllers\Api;
```

Also we also need to import the base controller since we're in different namespace.

```
use App\Http\Controllers\Controller;
```

Remove the constructor since we're going to use `auth:api` middleware in the route level. Let's also get rid of the if statement and `return back()` from the `__invoke` method. So out `Api/VoteAnswerController` will look like this:

```
# Api/VoteAnswerController.php
<?php

namespace App\Http\Controllers\Api;
use App\Answer;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class VoteAnswerController extends Controller
{
    public function __invoke(Answer $answer)
    {
        $vote = (int) request()->vote;
        $votesCount = auth()->user()->voteAnswer($answer, $vote);

        return response()->json([
            'message'    => 'Thanks for the feedback',
            'votesCount' => $votesCount
        ]);
    }
}
```

Second, let's open the `VoteQuestionController` and do the same thing as before:

```
# VoteQuestionController
<?php

namespace App\Http\Controllers\Api;
use App\Question;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class VoteQuestionController extends Controller
{
    public function __invoke(Question $question)
    {
        $vote = (int) request()->vote;

        $votesCount = auth()->user()->voteQuestion($question, $vote);

        return response()->json([
            'message'    => 'Thanks for the feedback',
            'votesCount' => $votesCount
        ]);
    }
}
```

2. Define api routes

Alright, let's open up our `web.php` file. Then copy both these routes:

```
Route::post('/questions/{question}/vote', 'VoteQuestionController');
Route::post('/answers/{answer}/vote', 'VoteAnswerController');
```

Let's open the `api.php`, then pate inside the previous routes in `auth:api` middleware group. Don't forget to adjust the controller's path to `Api`.

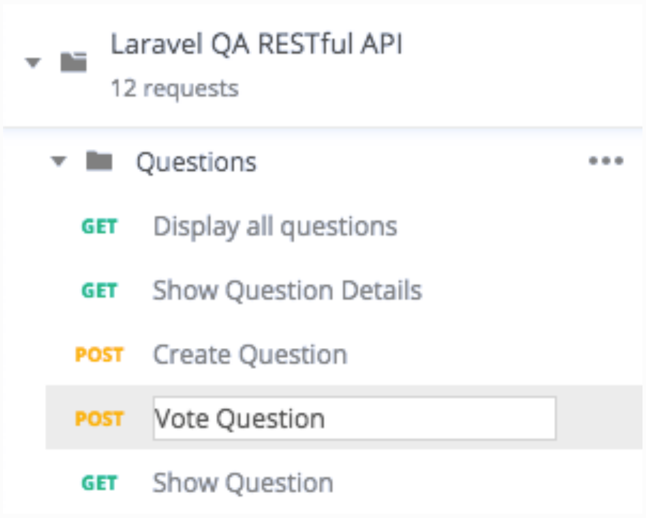
```
Route::middleware(['auth:api'])->group(function() {
    // ...
    Route::post('/questions/{question}/vote',
'Api\VoteQuestionController');
    Route::post('/answers/{answer}/vote', 'Api\VoteAnswerController');
});
```

Alright, I think that's all we need. Now we can test these endpoint out in the Postman.

TEST ENDPOINTS IN POSTMAN

1. Testing Vote Question Endpoint

In Postman we can go to `Questions` folder and duplicate the `Create Question` request. Let's rename it to `Vote Question`.



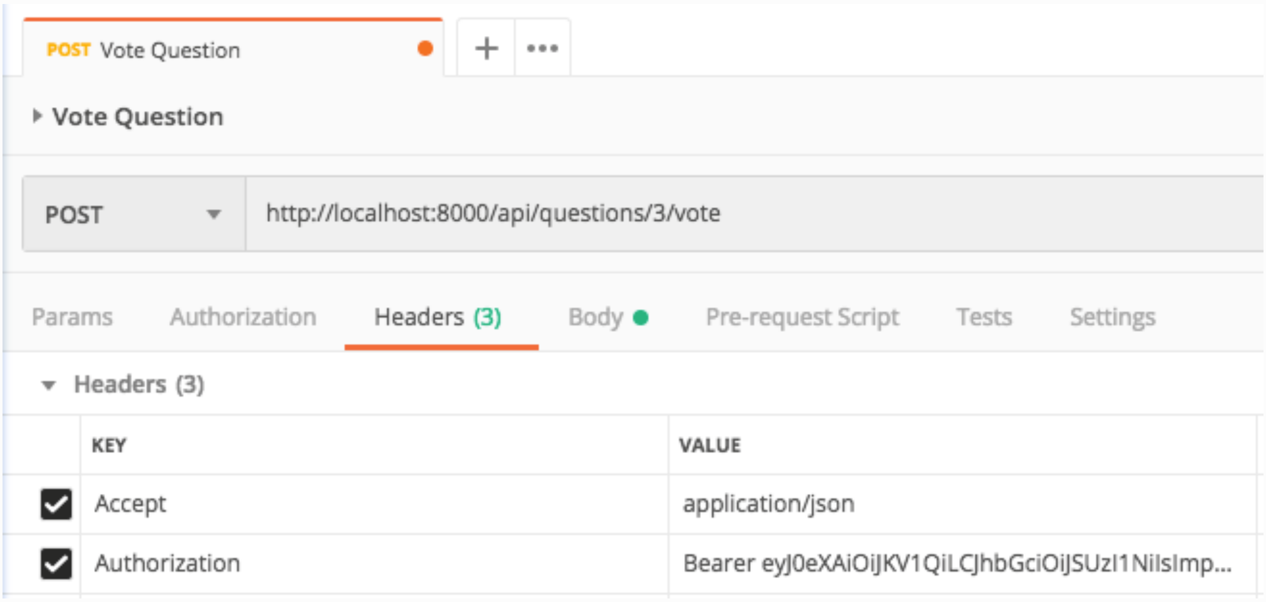
Laravel QA RESTful API
12 requests

Questions

GET Display all questions
GET Show Question Details
POST Create Question
POST Vote Question
GET Show Question

The HTTP method is going to be `POST` while the url is going to be `~/api/questions/3/vote`. You can change the question id to any existing number you have.

Go to **Headers** and make sure you add `Accept` to `application/json`, `Authorization` to `Bearer` and add the access token to it.



POST Vote Question

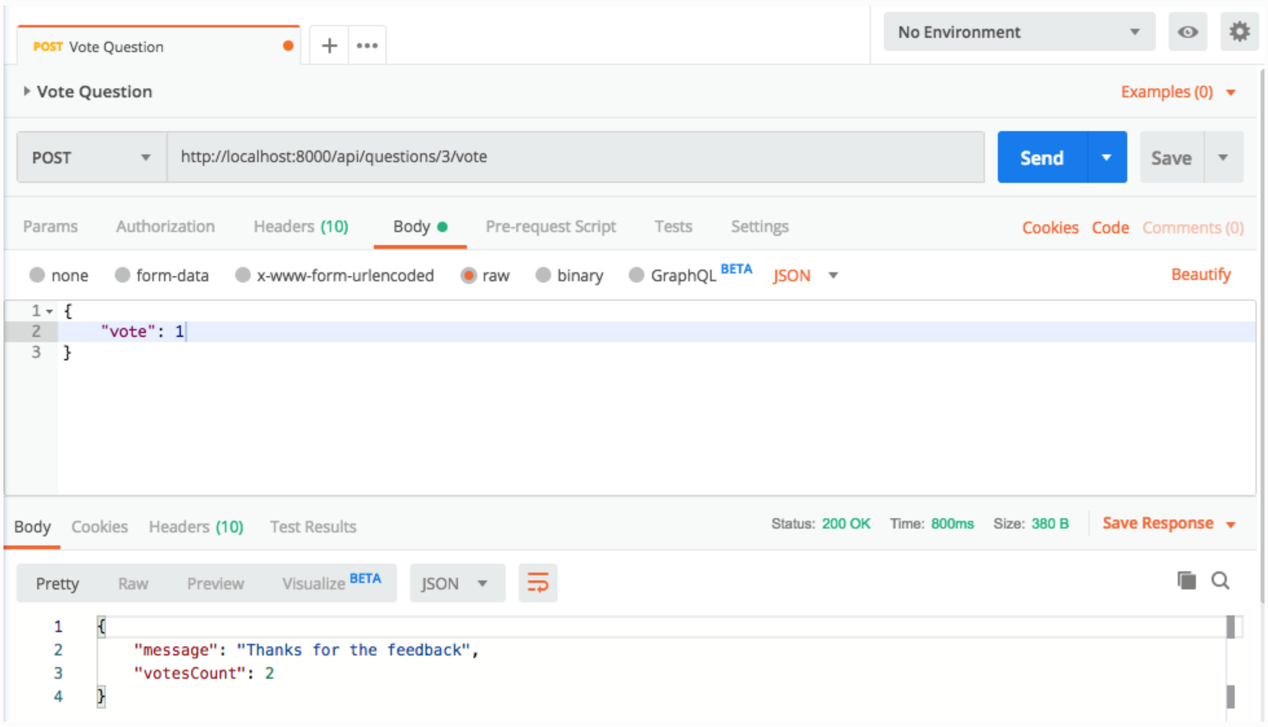
Vote Question

POST http://localhost:8000/api/questions/3/vote

Headers (3)

	KEY	VALUE
✓	Accept	application/json
✓	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp...

Then in `body` we can choose `raw` option with `application/json` selected. Then specify `vote: 1` or `vote: -1` in the input.


POST Vote Question
No Environment
Vote Question
Examples (0)
POST
Send
Save
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON
Beautify
Body Cookies Headers (10) Test Results
Save Response
Pretty Raw Preview Visualize BETA JSON

http://localhost:8000/api/questions/3/vote

```
{
    "vote": 1
}
```

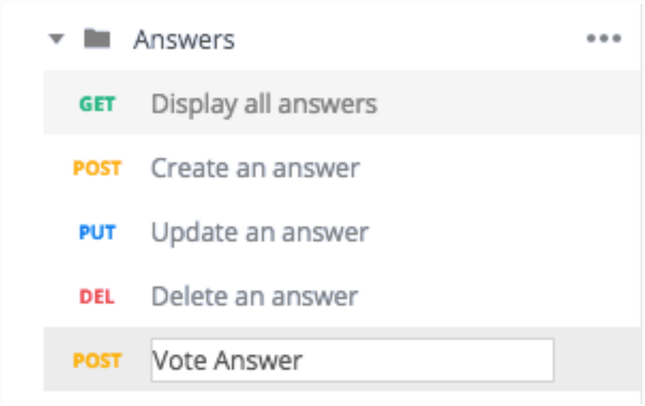
Status: 200 OK Time: 800ms Size: 380 B

```
{
    "message": "Thanks for the feedback",
    "votesCount": 2
}
```

Now if you hit the **Send** button. Make sure you're get back the success message along with votes count.

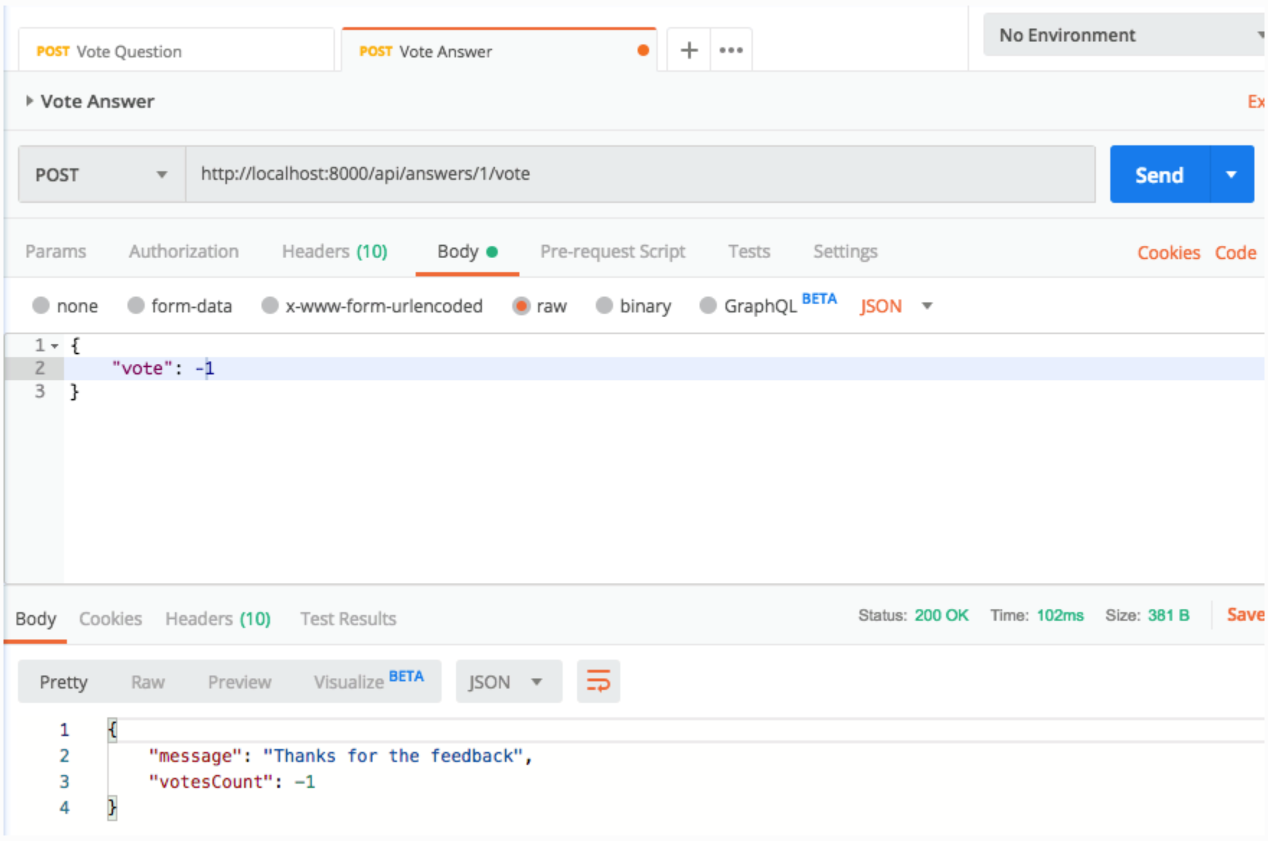
2. Testing Vote Answer Endpoint

To test Vote Answer endpoint we can duplicate `Vote Question` request in `Questions` folder. Then move it to `Answers` folder. And then rename it to `Vote Answer`.



All request attributes excpet the url are going to be the same. So all you need to do is change the url to `~/api/answers/1/vote`. You can adjust the answer id in the url to any existing number that you have in `answers` table. You can also adjust the vote whether `1` or `-1`.

If you hit the **Send** button. Make sure you're getting back the proper response like this figure:



POST http://localhost:8000/api/answers/1/vote

```
{
    "vote": -1
}
```

Status: 200 OK Time: 102ms Size: 381 B

Body: JSON

```
{
    "message": "Thanks for the feedback",
    "votesCount": -1
}
```

SUMMARY

In this lesson, we looked at how to create api endpoints for voting question and answer by simply copy our existing controllers. In the next lesson we're gonna make other endpoints that are used for favoriting question and marking an answer as best answer.

Alright, let's go ahead and commit our changes that we made today into our git repo.

```
git add .
git commit -m "Create api endpoints for vote question and answer"
git push origin lesson-52
```