

LAPORAN 7
Integratif Programming
Validation



Disusun Oleh :

Nama : Ichsanul Dwi Prayitno

Nim : 4.33.20.0.14

Kelas : TI-4A

PROGRAM STUDI S.TR TEKNOLOGI REKAYASA KOMPUTER
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI SEMARANG
TAHUN 2024

PRAKTEK

I. Lembar Kerja

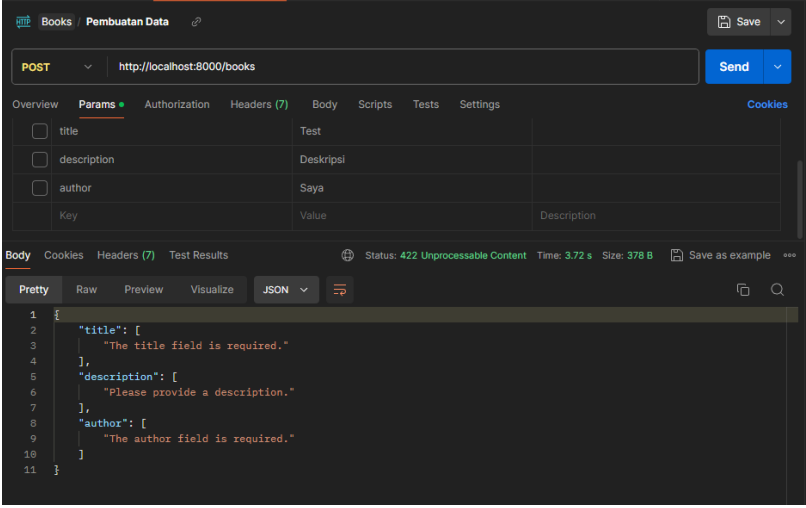
| No | Praktik | Hasil Program |
|----|---|---------------|
| 1 | Membuat File BooksControll erValidationTe st.php | Dengan Isi |

```

1 <?php
2
3 namespace Tests\App\Http\Controllers;
4
5 use Tests\TestCase;
6 use Illuminate\Http\Response;
7 use Laravel\Lumen\Testing\DatabaseMigrations;
8 use App\Models\Book;
9
10 class BooksControllerValidationTest extends TestCase
11 {
12     use DatabaseMigrations;
13
14     /** @test */
15     public function it_validates_required_fields_when_creating_a_new_book()
16     {
17         $this->post('/books', [], ['Accept' => 'application/json']);
18
19         $this->assertEquals(Response::HTTP_UNPROCESSABLE_ENTITY, $this->response->getStatusCode());
20
21         $body = json_decode($this->response->getContent(), true);
22
23         $this->assertArrayHasKey('title', $body);
24         $this->assertArrayHasKey('description', $body);
25         $this->assertArrayHasKey('author', $body);
26
27         $this->assertEquals(['The title field is required.'], $body['title']);
28         $this->assertEquals(
29             ['Please provide a description.'],
30             $body['description']
31         );
32         $this->assertEquals(['The author field is required.'], $body['author']);
33     }
34
35     /** @test */
36     public function it_validates_validated_fields_when_updating_a_book()
37     {
38         // $book = factory(App\Book::class)->create();
39         $book = Book::factory(1)->create();
40
41         $this->put("/books/{$book->id}", [], ['Accept' => 'application/json']);
42
43         $this->assertEquals(Response::HTTP_UNPROCESSABLE_ENTITY, $this->response->getStatusCode());
44
45         $body = json_decode($this->response->getContent(), true);
46
47         $this->assertArrayHasKey('title', $body);
48         $this->assertArrayHasKey('description', $body);
49         $this->assertArrayHasKey('author', $body);
50
51         $this->assertEquals(['The title field is required.'], $body['title']);
52         $this->assertEquals(
53             ['Please provide a description.'],
54             $body['description']
55         );
56         $this->assertEquals(['The author field is required.'], $body['author']);
57     }
58
59     /** @test */
60     public function title_fails_create_validation_when_just_too_long()
61     {
62         // Creating a book
63         // $book = factory(App\Book::class)->make();
64         $book = Book::factory(1)->create();
65         $book->title = str_repeat('a', 256);
66
67         $this->post("/books", [
68             'title' => $book->title,
69             'description' => $book->description,
70             'author' => $book->author,
71             ], ['Accept' => 'application/json']);
72
73         $this
74             ->seeStatusCode(Response::HTTP_UNPROCESSABLE_ENTITY)
75             ->seeJson([
76                 'title' => ['The title may not be greater than 255 characters.']
77             ])
78             ->notSeeInDatabase('books', ['title' => $book->title]);
79     }
80
81     /** @test */
82     public function title_fails_update_validation_when_just_too_long()
83     {
84         // Updating a book
85         // $book = factory(App\Book::class)->create();
86         $book = Book::factory(1)->create();
87         $book->title = str_repeat('a', 256);
88
89         $this->put("/books/{$book->id}", [
90             'title' => $book->title,
91             'description' => $book->description,
92             'author' => $book->author,
93             ], ['Accept' => 'application/json']);
94
95         $this
96             ->seeStatusCode(Response::HTTP_UNPROCESSABLE_ENTITY)
97             ->seeJson([
98                 'title' => ['The title may not be greater than 255 characters.']
99             ])
100             ->notSeeInDatabase('books', ['title' => $book->title]);
101     }
102
103     /** @test */
104     public function title_passes_create_validation_when_exactly_max()
105     {
106         // Creating a new book
107         // $book = factory(App\Book::class)->make();
108         $book = Book::factory(1)->create();
109         $book->title = str_repeat('a', 255);
110
111         $this->post("/books", [
112             'title' => $book->title,
113             'description' => $book->description,
114             'author' => $book->author,
115             ], ['Accept' => 'application/json']);
116
117         $this
118             ->seeStatusCode(Response::HTTP_CREATED)
119             ->seeInDatabase('books', ['title' => $book->title]);
120     }
121
122     /** @test */
123     public function title_passes_update_validation_when_exactly_max()
124     {
125         // Updating a book
126         // $book = factory(App\Book::class)->create();
127         $book = Book::factory(1)->create();
128         $book->title = str_repeat('a', 255);
129
130         $this->put("/books/{$book->id}", [
131             'title' => $book->title,
132             'description' => $book->description,
133             'author' => $book->author,
134             ], ['Accept' => 'application/json']);
135
136         $this
137             ->seeStatusCode(Response::HTTP_OK)
138             ->seeInDatabase('books', ['title' => $book->title]);
139     }
140 }
141

```

| | | |
|---|---|---|
| 2 | <p>Menambahkan Validasi pada metode penyimpanan dan pengenditan di file BooksController.php</p> | <p>Function Store</p> <pre> 1 public function store(Request \$request) 2 { 3 \$this->validate(\$request, [4 'title' => 'required max:255', 5 'description' => 'required', 6 'author' => 'required' 7], [8 'description.required' => 'Please provide a :attribute.' 9]); 10 \$book = Book::create(\$request->all()); 11 \$data = \$this->item(\$book, new BookTransformer()); 12 13 return response()->json(\$data, 201, [14 'Location' => route('books.show', ['id' => \$book->id]) 15]); 16 } 17 </pre> <p>Function Update</p> <pre> 1 public function update(Request \$request, \$id) 2 { 3 try { 4 \$book = Book::findOrFail(\$id); 5 } catch (ModelNotFoundException \$e) { 6 return response()->json([7 'error' => [8 'message' => 'Book not found' 9] 10], 404); 11 } 12 13 \$this->validate(\$request, [14 'title' => 'required max:255', 15 'description' => 'required', 16 'author' => 'required' 17], [18 'description.required' => 'Please provide a :attribute.' 19]); 20 21 \$book->fill(\$request->all()); 22 \$book->save(); 23 24 return \$this->item(\$book, new BookTransformer()); 25 } </pre> |
| 3 | Testing Phpunit | <pre> PS D:\laragon\www\Intergrative Programming\Project> .\vendor\bin\phpunit PHPUnit 10.5.11 by Sebastian Bergmann and contributors. Runtime: PHP 8.1.10 Configuration: D:\laragon\www\Intergrative Programming\Project\phpunit.xml 21 / 21 (100%) Time: 00:34.170, Memory: 34.00 MB OK (21 tests, 91 assertions) PS D:\laragon\www\Intergrative Programming\Project> </pre> |

| | | |
|---|-----------------|--|
| 4 | Testing postman |  |
|---|-----------------|--|

TEORI

I. Lembar Soal

N/A

II. Lembar Kerja

N/A