**Skenario Getting Specified Note**

Kita lanjutkan ke skenario pengujian ke-3 yaitu mendapatkan catatan secara spesifik berdasarkan noteId yang berada di variable environment. Sebelum itu, kita ingat-ingat kembali kriteria dari skenario ini.

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

Pastikan response memiliki status code 200.

Pastikan header response Content-Type memiliki nilai application/json.

Pastikan body response merupakan object.

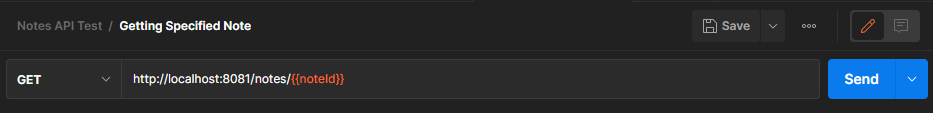
Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.

Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

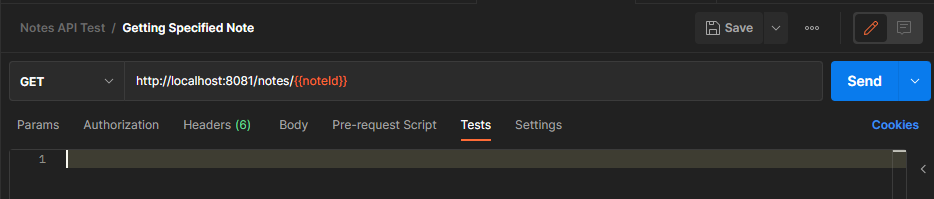
Sudah siap? Yuk kita buat request baru di dalam Notes API Test. Beri nama request tersebut dengan “Getting Specified Note”.

Pada request URL isikan nilai localhost:5000/notes/{{noteId}} dan biarkan method yang digunakan tetap GET.



Seperti yang sudah Anda ketahui, {{noteId}} merupakan nilai yang diambil dari environment variabel. Di mana variabel tersebut terisi nilainya ketika kita melakukan permintaan memasukkan catatan baru.

Sekarang yuk kita mulai tuliskan saja pengujian pada tab Tests.



Seperti biasa, pertama kita buat test untuk menguji status code dari respons dengan nama “response status code should have 200 value”.

pm.test('response status code should have 200 value', () => {

});

Uji status code dengan nilai 200.

pm.test('response status code should have 200 value', () => {

pm.response.to.have.status(200);

});

Done!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

Pastikan header response Content-Type memiliki nilai application/json.

Pastikan body response merupakan object.

Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.

Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

Lanjut, buat testing baru dengan nama ‘response Content-Type header should have application/json value’.

pm.test('response Content-Type header should have application/json value', () => {

});

Seperti biasa, di dalamnya kita uji response header Content-Type dengan nilai “application/json; charset=utf-8”.

pm.test('response Content-Type header should have application/json value', () => {

pm.expect(pm.response.headers.get('Content-Type')).to.equals('application/json; charset=utf-8');

});

Done!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

~~Pastikan header response Content-Type memiliki nilai application/json.~~

Pastikan body response merupakan object.

Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.

Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

Lanjut buat tes baru dengan nama “response body should be an object”.

pm.test('response body should be an object', () => {

});

Dapatkan nilai response body dalam bentuk JSON.

pm.test('response body should be an object', () => {

const responseJson = pm.response.json();

});

Kemudian uji responseJson, ia adalah sebuah objek.

pm.test('response body should be an object', () => {

const responseJson = pm.response.json();

pm.expect(responseJson).to.be.an('object');

});

Done!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

~~Pastikan header response Content-Type memiliki nilai application/json.~~

~~Pastikan body response merupakan object.~~

Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.

Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

Lanjut kita uji objek response body-nya, ia harus memiliki properti yang sesuai. Buat test baru dengan nama “response body should have the correct property and value”.

pm.test('response body should have the correct property and value', () => {

});

Dapatkan response body dalam bentuk JSON.

pm.test('response body should have the correct property and value', () => {

const responseJson = pm.response.json();

});

Lalu uji responseJson, ia harus memiliki properti status dengan nilai success dan data yang merupakan object.

pm.test('response body should have the correct property and value', () => {

const responseJson = pm.response.json();

pm.expect(responseJson).to.have.ownProperty('status');

pm.expect(responseJson.status).to.equals('success');

pm.expect(responseJson).to.have.ownProperty('data');

pm.expect(responseJson.data).to.be.an('object');

});

Done!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

~~Pastikan header response Content-Type memiliki nilai application/json.~~

~~Pastikan body response merupakan object.~~

~~Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.~~

Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

Lanjut uji object data dari response body bahwa ia harus memiliki properti note yang merupakan sebuah objek.

Buat test baru dengan nama “'response body data should contain note object”.

pm.test('response body data should contain note object', () => {

});

Dapatkan objek data dari response body dalam bentuk JSON.

pm.test('response body data should contain note object', () => {

const responseJson = pm.response.json();

const { data } = responseJson;

});

Uji objek data bahwa ia harus memiliki properti note yang merupakan sebuah objek.

pm.test('response body data should contain note object', () => {

const responseJson = pm.response.json();

const { data } = responseJson;

pm.expect(data).to.have.ownProperty('note');

pm.expect(data.note).to.be.an('object');

});

Done!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

~~Pastikan header response Content-Type memiliki nilai application/json.~~

~~Pastikan body response merupakan object.~~

~~Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.~~

~~Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.~~

Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.

Lanjut ke pengujian terakhir. Kita akan uji objek note, ia harus memiliki properti id, title, body, dan tags. Properti-properti tersebut harus memiliki nilai yang sesuai dengan nilai yang dimasukan pada skenario pertama.

Buat test baru dengan nama “note object should contain correct value for id, title, body, and tags property”.

pm.test('note object should contain correct value for id, title, body, and tags property', () => {

});

Di dalamnya, dapatkan objek note dari data di dalam response body.

pm.test('note object should contain correct value for id, title, body, and tags property', () => {

const responseJson = pm.response.json();

const { data: { note } } = responseJson;

});

Sebelum menguji properti dan nilai di dalam objek note, tentukan dulu nilai ekspektasi dari id, title, body, dan tags. Nilai id bisa Anda dapatkan dari variabel environment melalui method pm.environment.get(‘noteId’), sedangkan yang lain Anda bisa lihat nilainya pada request Adding Notes yang dikirimkan melalui body.

Yuk kita buat dulu nilai ekspektasinya.

pm.test('note object should contain correct value for id, title, body, and tags property', () => {

const responseJson = pm.response.json();

const { data: { note } } = responseJson;

const expectedId = pm.environment.get('noteId');

const expectedTitle = 'Catatan A';

const expectedTags = ['Android', 'Web'];

const expectedBody = 'Isi dari catatan A';

});

Setelah menentukan nilai ekspektasi, barulah kita uji objek note bahwa ia harus memiliki properti dan nilai yang sesuai dengan nilai ekspektasi.

pm.test('note object should contain correct value for id, title, body, and tags property', () => {

const responseJson = pm.response.json();

const { data: { note } } = responseJson;

const expectedId = pm.environment.get('noteId');

const expectedTitle = 'Catatan A';

const expectedTags = ['Android', 'Web'];

const expectedBody = 'Isi dari catatan A';

pm.expect(note).to.have.ownProperty('id');

pm.expect(note.id).to.equals(expectedId);

pm.expect(note).to.have.ownProperty('title');

pm.expect(note.title).to.equals(expectedTitle);

pm.expect(note).to.have.ownProperty('tags');

pm.expect(note.tags).to.equals(expectedTags);

pm.expect(note).to.have.ownProperty('body');

pm.expect(note.body).to.equals(expectedBody);

});

Kriteria pengujian dari skenario 3 selesai!

Skenario 3: Getting Specified Note (Mendapatkan catatan secara spesifik)

~~Pastikan response memiliki status code 200.~~

~~Pastikan header response Content-Type memiliki nilai application/json.~~

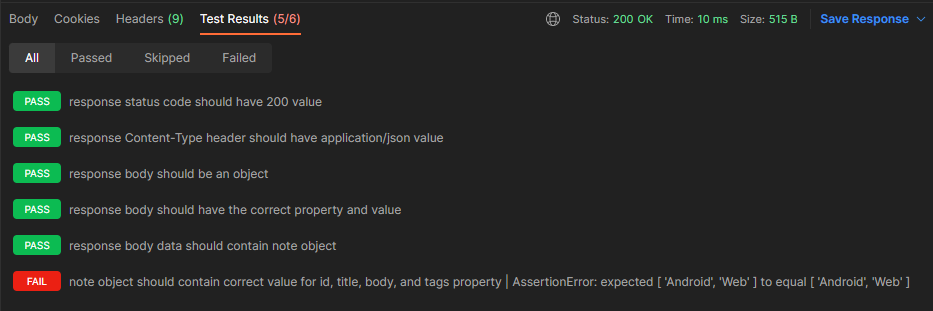
~~Pastikan body response merupakan object.~~

~~Pastikan body response memiliki properti dan nilai atau tipe data yang sesuai.~~

~~Pastikan data pada response body memiliki properti note yang merupakan sebuah objek.~~

~~Pastikan objek note di dalam data memiliki properti id, title, body, dan tags dengan nilai yang sesuai.~~

Yuk kita kirim permintaan. Silakan klik tombol Send, kemudian lihat hasil pengujiannya.



Oh tidak! Ada satu pengujian yang gagal, pesannya:

AssertionError: expected [ 'Android', 'Web' ] to equal [ 'Android', 'Web' ]

Apa yang salah ya? Bila dilihat sih nilai expected dan actual sudah sama persis. Lalu apa yang menyebabkan ia eror?

Ketahuilah bahwa untuk menguji nilai array dan objek, kita tidak bisa menggunakan method equals() seperti yang biasa kita gunakan. Itu karena array dan objek tidak bisa disamakan secara identik walaupun ia memiliki item atau properti dan nilai yang sama persis. Anda bisa membuktikan ini dengan menjalankan kode berikut pada node REPL:

["Harry", "Potter"] === ["Harry", "Potter"]

// -> false

{ firstName: "Harry", lastName: "Potter" } === { firstName: "Harry", lastName: "Potter" }

// -> false

Lalu bagaimana solusinya? Bagaimana cara kita menguji nilai array?

Caranya adalah dengan melakukan deep equals. Deep equals paling sederhana dapat dilakukan menggunakan bantuan JSON.stringify. Lebih jelasnya, kedua objek atau array yang akan diuji diubah menjadi JSON string, kemudian kedua JSON string tersebutlah yang akan diuji nilainya.

JSON.stringify(["Harry", "Potter"]) === JSON.stringify(["Harry", "Potter"]);

// -> true

JSON.stringify({ firstName: "Harry", lastName: "Potter" }) === JSON.stringify({ firstName: "Harry", lastName: "Potter" });

// -> true

Cukup mudah ‘kan? Bahkan, akan lebih mudah lagi untuk melakukan deep equal pada pengujian di Postman karena kita tidak perlu repot mengubah nilai ekspektasi dan actual ke JSON string. Cukup ganti penggunaan method equals() ke eql().

pm.test('note object should contain correct value for id, title, body, and tags property', () => {

const responseJson = pm.response.json();

const { data: { note } } = responseJson;

const expectedId = pm.environment.get('noteId');

const expectedTitle = 'Catatan A';

const expectedTags = ['Android', 'Web'];

const expectedBody = 'Isi dari catatan A';

pm.expect(note).to.have.ownProperty('id');

pm.expect(note.id).to.equals(expectedId);

pm.expect(note).to.have.ownProperty('title');

pm.expect(note.title).to.equals(expectedTitle);

pm.expect(note).to.have.ownProperty('tags');

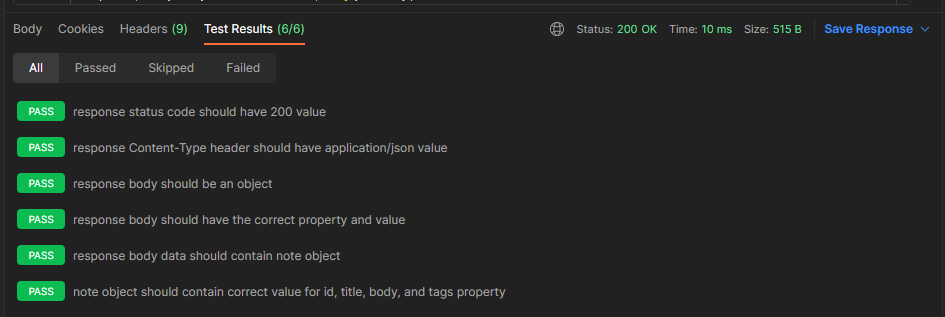
pm.expect(note.tags).to.eql(expectedTags);

pm.expect(note).to.have.ownProperty('body');

pm.expect(note.body).to.equals(expectedBody);

});

Kirim kembali permintaan dengan menekan Send.



Voila! Sekarang seluruh pengujian berhasil dijalankan. Jangan lupa simpan request kali ini dengan menekan tombol Save sebelum melanjutkan ke skenario selanjutnya.