**Skenario Adding Notes**

Kita mulai dengan skenario 1. Mari ingat kembali skenarionya.

Skenario 1: Adding Notes (Memasukkan catatan baru)

Pastikan response memiliki status code 201.

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

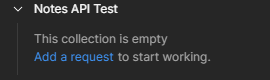
Pastikan body response adalah object.

Pastikan body response memiliki properti dan nilai yang sesuai.

Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.

Yuk kita buat request baru di dalam collection Notes API Test.

Silakan buka tab Collections. Klik Add a request untuk menambahkan request.



Beri nama “Add Notes” pada request kali ini.



Isi request URL dengan nilai localhost:8081/notes dan gunakan method POST.



Lalu, tambahkan data catatan yang akan ditambahkan pada Body. Silakan klik tab Body, pilih opsi raw, kemudian ganti format text menjadi JSON, dan tuliskan struktur JSON berikut:

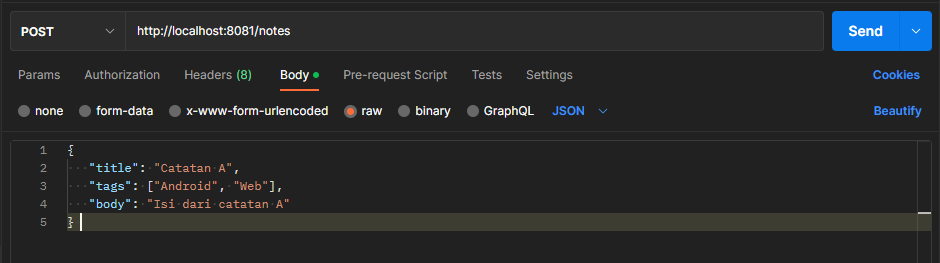
{

"title": "Catatan A",

"tags": ["Android", "Web"],

"body": "Isi dari catatan A"

}



Eits! Jangan kirim request dulu ya. Kita akan tuliskan skenario testing untuk request ini terlebih dahulu. Silakan pilih tab Tests..



Mulai tuliskan skenario testing di sini.

Postman memiliki global object pm. Untuk membuat testing, gunakan method pm.test(). Method tersebut menerima dua parameter, yang pertama nama test dan yang kedua adalah spec function. Method test() mirip seperti method it() pada Jest (Anda sudah mempelajarinya di kelas Belajar Dasar Pemrograman JavaScript).

Buat testing baru dengan nama ‘response status code should have 201 values’.

pm.test('response status code should have 201 value', () => {

});

Pastikan response status bernilai 201 dengan cara:

pm.test('response status code should have 201 value', () => {

pm.response.to.have.status(201);

});

Bila nanti respons status tidak memiliki nilai 201, maka tesnya akan gagal. Satu testing telah dituliskan.

Skenario 1: Adding Notes (Memasukkan catatan baru)

~~Pastikan response memiliki status code 201.~~

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

Pastikan body response adalah object.

Pastikan body response memiliki properti dan nilai yang sesuai.

Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.

Lanjut tuliskan tes selanjutnya.

Buat test dengan nama ‘response Content-Type header should have application/json value’.

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

});

Kemudian uji nilai Content-Type pada response header dengan nilai application/json.

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

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

});

Done!

Skenario 1: Adding Notes (Memasukkan catatan baru)

~~Pastikan response memiliki status code 201.~~

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

Pastikan body response adalah object.

Pastikan body response memiliki properti dan nilai yang sesuai.

Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.

Kita lanjut ke tes selanjutnya.

Buat test dengan nama ‘response body should be an object’.

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

});

Kemudian dapatkan nilai response body dalam bentuk JSON dengan cara:

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

const responseJson = pm.response.json();

});

Selanjutnya uji responseJson bahwa ia harus merupakan sebuah objek. Caranya:

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

const responseJson = pm.response.json();

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

});

Done!

Skenario 1: Adding Notes (Memasukkan catatan baru)

~~Pastikan response memiliki status code 201.~~

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

~~Pastikan body response adalah object.~~

Pastikan body response memiliki properti dan nilai yang sesuai.

Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.

Lanjut. Silakan buat testing baru bernama ‘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 dan uji kesesuaian properti serta nilai atau tipe datanya.

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

const responseJson = pm.response.json();

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

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

pm.expect(responseJson).to.ownProperty('message');

pm.expect(responseJson.message).to.equals('Catatan berhasil ditambahkan');

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

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

});

Done!

Skenario 1: Adding Notes (Memasukkan catatan baru)

~~Pastikan response memiliki status code 201.~~

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

~~Pastikan body response adalah object.~~

~~Pastikan body response memiliki properti dan nilai yang sesuai.~~

Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.

Lanjut ke kriteria terakhir. Kita akan memastikan bahwa nilai data yang dibawa oleh respons memiliki noteId dan nilainya tidaklah kosong.

Buat test baru dengan nama ‘response body data should have noteId property and not equal to empty’.

pm.test('response body data should have noteId property and not equal to empty', () => {

});

Di dalamnya, dapatkan nilai objek data dari response yang sudah diubah dengan bentuk JSON.

pm.test('response body data should have noteId property and not equal to empty', () => {

const responseJson = pm.response.json();

const { data } = responseJson;

});

Kemudian uji data bahwa ia harus memiliki properti noteId dan nilainya tidak kosong.

pm.test('response body data should have noteId property and not equal to empty', () => {

const responseJson = pm.response.json();

const { data } = responseJson;

pm.expect(data).to.ownProperty('noteId');

pm.expect(data.noteId).to.not.equals('');

});

Seperti yang kita ketahui, ketika permintaan menambahkan catatan baru, respons akan mengembalikan noteId dari catatan baru tersebut. Simpanlah nilainya pada environment variabel noteId agar nilainya dapat digunakan pada skenario pengujian selanjutnya.

Untuk menyimpan nilai pada variabel environment, kita bisa gunakan method pm.environment.set() Method tersebut menerima dua parameter, yakni nama variabel dan nilai yang akan ditetapkan padanya.

pm.test('response body data should have noteId property and not equal to empty', () => {

const responseJson = pm.response.json();

const { data } = responseJson;

pm.expect(data).to.ownProperty('noteId');

pm.expect(data.noteId).to.not.equals('');

pm.environment.set('noteId', data.noteId);

});

Done! Pengujian pada skenario pertama selesai.

Skenario 1: Adding Notes (Memasukkan catatan baru)

~~Pastikan response memiliki status code 201.~~

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

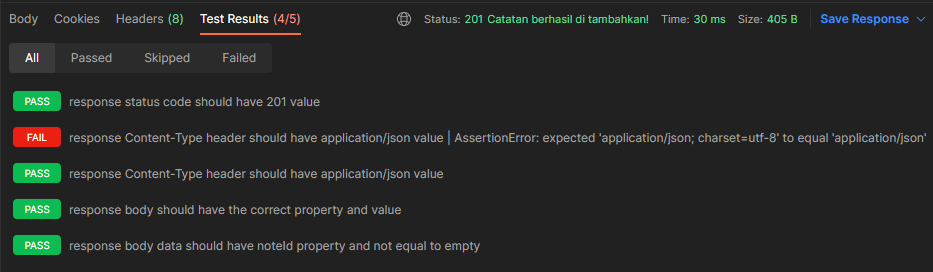
~~Pastikan body response adalah object.~~

~~Pastikan body response memiliki properti dan nilai yang sesuai.~~

~~Pastikan data pada response body memiliki noteId dan nilainya tidak kosong.~~

Sekarang coba kirim permintaan dengan klik tombol Send.

Lihat hasil uji pada kolom Test Result.



Wah! Ada satu pengujian yang gagal. Pesan eror mengatakan bahwa nilai Content-Type bukanlah “application/json” (expectation) namun “application/json; charset=utf-8” (actual).

Nilai actual sebenarnya adalah nilai yang kita ekspektasikan, namun karena kita salah menetapkan nilai ekspektasi, maka pengujian gagal. Dengan begitu, Anda jadi tahu kan bahwa pengujian otomatis ini sangat ketat, ia tidak akan mentoleransi bila terjadi perbedaan sedikit pun antara nilai ekspektasi dan actual.

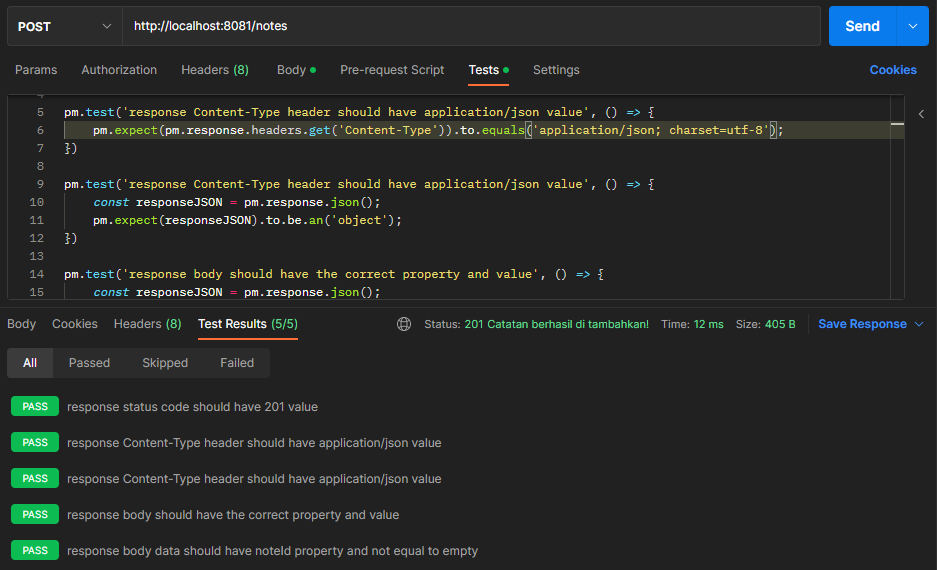
Untuk memperbaiki pengujian tersebut, kita perlu mengubah nilai ekspektasi menjadi “application/json; charset=utf-8”.

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');

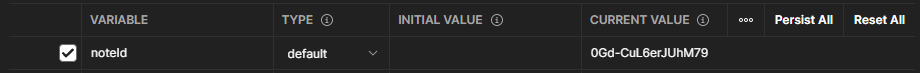
});

Silakan kirim kembali permintaan dengan klik tombol Send.



Woa! Sekarang seluruh hasil pengujian berhasil. Dengan begitu respons server dapat dipastikan sesuai dengan yang Anda inginkan atau ekspektasikan.

Pastikan juga variabel noteId pada environment sudah terisi nilainya yah. Karena variabel ini akan kita gunakan pada skenario testing selanjutnya.



Sebelum melanjutkan ke skenario berikutnya, jangan lupa untuk simpan request ini dengan cara klik tombol Save.