Pertemuan 12
REST API Server dengan PHP

Pendahuluan

Nah kemarin kita menggunakan server orang lain untuk mempelajari RESTFUL API, untuk materi kali ini kita akan buat RESTFUL Server dan RESTFUL Client, yang kita buat hampir sama dengan yang kemarin jadi bisa menggunakan yang kemarin tetapi bedanya kita akan menggunakan server kita sendiri untuk menerima request dari client dan data tersebut akan disimpan menuju ke database ataupun di-ambil dari database, agar nanti-nya di Client walaupun sudah di close data masih ada seperti yang kemarin.

Kita nanti menggunakan function header di php untuk memberi tau si request kalau response-nya merupakan JSON, karena kita memberitahu kalau response-nya JSON kita harus melakukan Serialization dulu, apa sih Serialization, Serialization merupakan istilah yang bermakna translasi/pengubahan suatu data ke data lain.

Desain

Sebelum kita mulai ngoding kita harus buat tabel-nya dulu, kalau ada nama tabel yang sama di hapus agar tidak menggangu contoh atau kalau tidak membuat database baru saja.

```
CREATE TABLE IF NOT EXISTS todo(
  id INT AUTO_INCREMENT PRIMARY KEY,
  judul VARCHAR(255) NOT NULL,
  selesai BOOLEAN DEFAULT FALSE
);
```

Hal yang baru di schema ini adalah tipe data *BOOLEAN* tipe data ini seperti di dalam bahasa pemrograman, dan *DEFAULT* ini bermaksud kalau tidak di-isi, isi default-nya apa

URI yang kita akan buat nanti-nya(Kalu disimpan di xampp/htdocs)

NO	HTTP	URI	Effect
1	GET	localhost/nama-folder-di-htdocs/todo	Untuk mengambil ke-seluruhan data todo
2	POST	localhost/nama-folder-di-htdocs/todo	Untuk menambah todo
3	DELETE	localhost/nama-folder-di-htdocs/todo/delete.php?id=xxx	Untuk menghapus todo
4	PATCH	localhost/nama-folder-di-htdocs/todo/complete.php?id=xxx	Untuk menandai selesai todo

Kenapa kok di masukan di folder sendiri, karena secara konvensi RESTFUL API itu URI endpoint-nya harus gitu, di nomor 1 dan 2 itu kita akan pakai file yang bernama index.php karena di php index.php yang dicari kalau kitadi URI-nya tidak men-spesifikan file-nya, agar rapi kita pecah file tersebut menjadi.

1. /nama-folder-di-htdocs/todo/index.php
2. /nama-folder-di-htdocs/todo/get.php
3. /nama-folder-di-htdocs/todo/create.php
4. /nama-folder-di-htdocs/todo/complete.php
5. /nama-folder-di-htdocs/todo/db.php (File ini digunakan untuk pengurangan duplikasi kode)

6. /nama-folder-di-htdocs/todo/response.php (File ini digunakan untuk pengurangan duplikasi kode)
7. /nama-folder-di-htdocs/index.html (File client / pengguna RESTFUL API)

Perlu di-ingat setiap URI harus memberikan HTTP Response Code yang benar

HTTP Code	Arti Standar	Arti
200	OK	Aman
201	Created	Data berhasil di buat
202	Accepted	Data diterima
203	Non-Authoritative Information	Si pengirim tidak memiliki akses di URI tersebut
400	Bad Request	Data bentuk-nya salah
404	Not Found	Halaman/Endpoint tidak ada
500	Server Error	Server-nya error

Di php ada function http_response_code($code), function ini bisa nge-set Status Code yang kita inginkan. **Perlu di-ingat jangan main-main dengan HTTP Code karena bisa merubah behavior si browser dan nanti-nya(kemungkinan juga) bisa si browser malah tidak bisa meng-akses website tersebut karena diberikan HTTP Code yang *Site Moved*.**

Schema API yang kita buat nanti ialah

1. GET localhost/nama-folder-di-htdocs/todo
   Request :
   ```
   {

   }
   ```
   Response :
   ```
   {
       “code” : 200,
       “data” : [
           {
               “id”: xxx,
               “judul”: “xxx”,
               “selesai”: 0
           },
           {
               “id”: xxx,
               “judul”: “judul”,
               “selesai”: 1
           }
           ...
       ]
   }
   ```
2. POST : localhost/nama-folder-di-htdocs/todo
   Request :
   ```
   {
       “judul”: “Todo post pertama ku!”
   }
   ```

Response :

```
{
    “code”: 201,
    “message”: “Data berhasil ditambah”,
    “todo”: {
        “id”: 1,
        “judul”: “Todo post pertama ku!”,
        “selesai”: 0
    }
}
```

3. DELETE : localhost/nama-folder-di-htdocs/todo/delete.php?id={angka}

Request :

```
{

}
```

Response :

```
{
    “code”: 200,
    “message”: “Data berhasil di hapus!”
}
```

4. PATCH : localhost/nama-folder-di-htdocs/todo/complete.php?id={angka}

Request :

```
{

}
```

Response :

```
{
    “code”: 200,
    “message”: “Data berhasil dihapus!”
}
```

Schema dan URI Endpoint yang kita buat ini sedikit mengikuti standar yang sudah di tentukan oleh konvensi RESTFUL API, jadi jangan khawatir.

Note :URI yang ada kurung kurawal *{}* ini dinamis jadi misal
localhost/nama-folder-di-htdocs/todo/complete.php?id=1
localhost/nama-folder-di-htdocs/todo/complete.php?id=2

Praktek
Setelah kita buat schema, desain jawaban, dan database, kita dapat langsung ngoding, pertama-tama kita ngoding di bagian file pembantu (db.php dan response.php)

```php
<?php

// db.php

$pdo = new PDO('mysql:host=localhost;dbname=ngoding', 'root', '');

// response.php

/**
 * Function yang mengirimkan data ke client dengan format json
 * dan sekalian nge-set http response code
 * @param  int   $code
 * @param  array $data
 */
function response(int $code, array $data)
{
    // Kasih response code
    http_response_code($code);

    // Nge-set Content-Type agar si client tau kalau response-nya json juga
    header("Content-Type: application/json");

    // Kita melakukan serialization data dan kita echo kan
    echo json_encode($data);

    // Lalu kita stop eksekusi php-nya
    die;
}

```

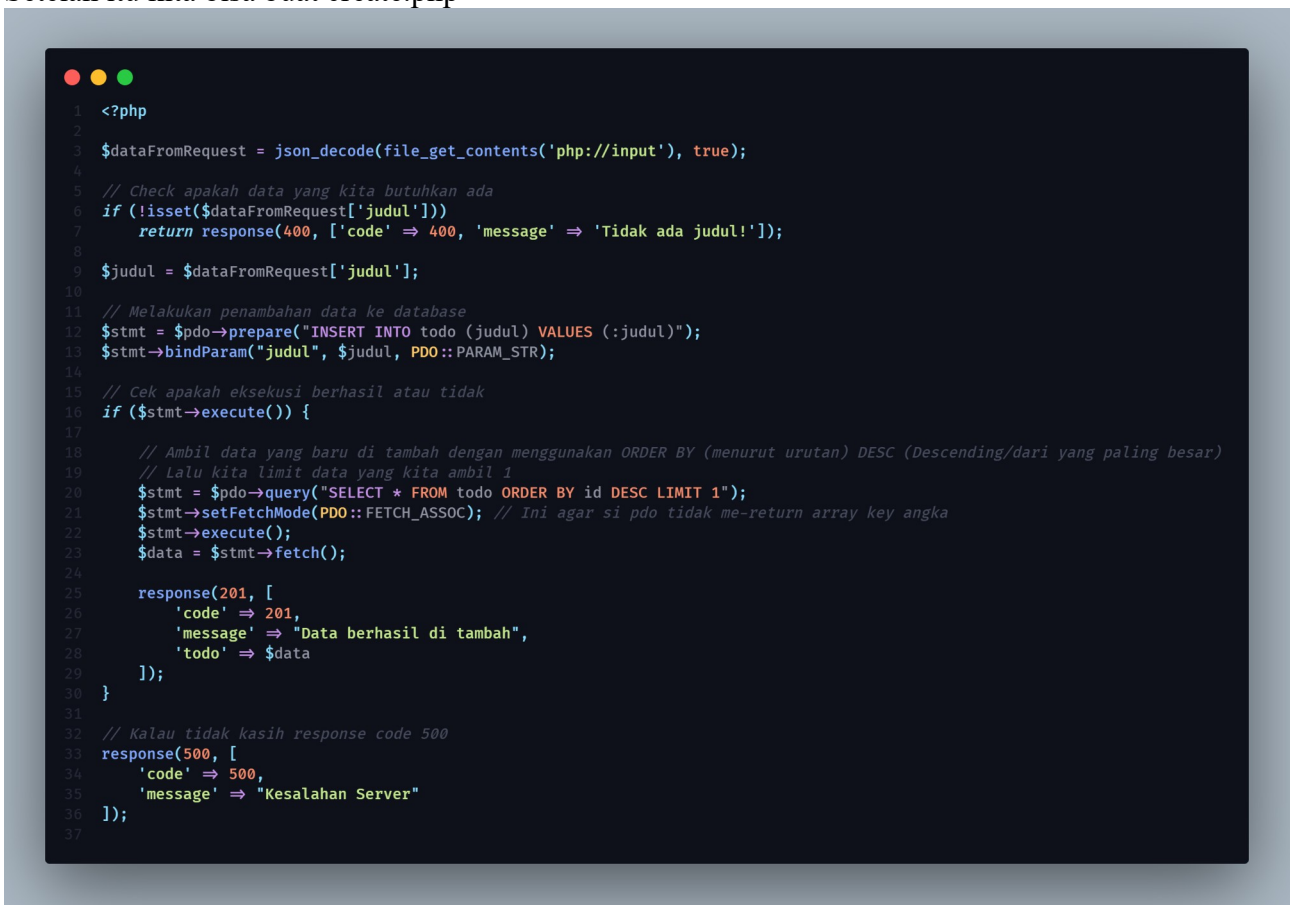
di db.php seperti biasa kita membuat koneksi ke database, dan di response.php kita men-definisikan function response yang memiliki parameter $code dan $data, $code ini digunakan untuk http_response_code, dan karena function ini digunakan untuk memberikan respond ke user/client dengan bentuk JSON kita harus melakukan Serialization di $data, setelah di echo hasil dari Serialization, kita langsung stop eksekusi.

Setelah itu kita mulai ngoding di bagian index.php

```php
<?php

// Kita include db.php karena kedua file yang nanti kita include membutuhkan database juga
// Kita juga include response.php karena kita tidak ingin duplikasi kode
require "db.php";
require "response.php";

// Ambil Request HTTP Method
$method = $_SERVER['REQUEST_METHOD'];


// Check apakah method-nya tadi POST kalau ya include create.php
// Karena implementasi create todo ada di create.php
if ($method == "POST") {
    require "create.php";


    // Jika method-nya GET include get.php karena implementasi pengambilan data ada di get.php
} else if ($method == "GET") {
    require "get.php";

    // Kalau semua tidak cocok beri 404
} else {
    return response(404, [
        'code' => 404,
        'message' => "Halaman tidak ada"
    ]);
}

```

Kita akan menggunakan $_SERVER untuk mengambil HTTP Request Method yang mengirimkan Request, lalu kita menggunakan data tersebut untuk melakukan perbandingan, karena file ini menerima 2 jenis HTTP Method.

Setelah itu kita bisa buat create.php



```php
<?php

$dataFromRequest = json_decode(file_get_contents('php://input'), true);

// Check apakah data yang kita butuhkan ada
if (!isset($dataFromRequest['judul']))
    return response(400, ['code' => 400, 'message' => 'Tidak ada judul!']);

$judul = $dataFromRequest['judul'];

// Melakukan penambahan data ke database
$stmt = $pdo->prepare("INSERT INTO todo (judul) VALUES (:judul)");
$stmt->bindParam("judul", $judul, PDO::PARAM_STR);

// Cek apakah eksekusi berhasil atau tidak
if ($stmt->execute()) {

    // Ambil data yang baru di tambah dengan menggunakan ORDER BY (menurut urutan) DESC (Descending/dari yang paling besar)
    // Lalu kita limit data yang kita ambil 1
    $stmt = $pdo->query("SELECT * FROM todo ORDER BY id DESC LIMIT 1");
    $stmt->setFetchMode(PDO::FETCH_ASSOC); // Ini agar si pdo tidak me-return array key angka
    $stmt->execute();
    $data = $stmt->fetch();

    response(201, [
        'code' => 201,
        'message' => "Data berhasil di tambah",
        'todo' => $data
    ]);
}

// Kalau tidak kasih response code 500
response(500, [
    'code' => 500,
    'message' => "Kesalahan Server"
]);
```

Karena request berbentuk json dan cara pengiriman-nya beda dengan <form> jadi kita tidak bisa menggunakan $_POST untuk mengambil data yang dikirim lewat HTTP POST (DELETE, PATCH, dan HTTP Method yang lain-nya), sebab itu kita mengambil data yang dikirimkan menggunakan function file_get_content(“php://input”) sebenar-nya function ini digunakan untuk mengambil konten file tetapi bisa diggunakan untuk mengambil data request tadi, setelah di-ambil kita harus Deserialization menggunakan function json_decode() parameter yang pertama menerima tipe data String, kita langsung selip-kan function file_get_content() setelah itu parameter kedua menerima tipe data boolean, yang ini merupakan flag di function ini kita menginginkan associative array atau tidak, kalau ya kita beri true.

Setelah itu kita cek apakah dari data input tadi key yang bernama judul yang nanti kita pakai, kalau tidak ada kita kirim-kan response yang berisi “tidak ada judul”, setelah itu hal yang sama seperti yang prakek yang dulu, cuman bedanya kita menggunakan response untuk mengirimkan data kembali ke user/client yang mengirimkan request tadi.

Dan di file get.php

```php
<?php

// Mengambil semua data di tabel todo
$stmt = $pdo->prepare("SELECT * FROM todo");
$stmt->setFetchMode(PDO::FETCH_ASSOC); // Ini agar si pdo tidak me-return array key angka
$stmt->execute();
$data = $stmt->fetchAll();

// Lalu kita berikan hasil-nya ke user lewat function response yang kita buat
response(200, [
    'code' => 200,
    'data' => $data
]);

```

Pasti dah tau ya ini maksud-nya apa

Setelah itu kita buat file complete.php

```php
<?php

require 'db.php';
require 'response.php';

// Check apakah yang me-request penghapusan meng-ikut sertakan id
// dan apakah http method-nya delete
if (!isset($_GET['id']) && $_SERVER['REQUEST_METHOD'] != 'PATCH')
    response(400, [
        'code' => 400,
        'message' => 'Data id tidak ada atau http method-nya salah'
    ]);

$true = true;

$stmt = $pdo->prepare("UPDATE todo SET selesai=:selesai WHERE id=:id");
$stmt->bindParam('selesai', $true, PDO::PARAM_BOOL);
$stmt->bindParam('id', $_GET['id'], PDO::PARAM_INT);

if ($stmt->execute()) {
    response(200, [
        'code' => 200,
        'message' => 'berhasil!'
    ]);
}

response(500, [
    'code' => 500,
    'message' => 'gagal!'
]);
```

Sama seperti di create.php hanya beda-nya kita melakukan query update data, dan entah kenapa di bindParam meminta referensi data jadi itu ada variabel $true, kalau temen-temen tidak tau referensi di konteks pemrograman tidak apa-apa karena ini hanya ada di C, C++, Rust, dll.

Untuk delete.php

```php
<?php

require 'db.php';
require 'response.php';

// Check apakah yang me-request penghapusan meng-ikut sertakan id
// dan apakah http method-nya delete
if (!isset($_GET['id']) && $_SERVER['REQUEST_METHOD'] != 'DELETE')
    response(400, [
        'code' => 400,
        'message' => 'Data id tidak ada atau http method-nya salah'
    ]);


$stmt = $pdo->prepare("DELETE FROM todo WHERE id=:id");
$stmt->bindParam('id', $_GET['id'], PDO::PARAM_INT);

if ($stmt->execute()) {
    response(200, [
        'code' => 200,
        'message' => 'Data berhasil dihapus!'
    ]);
}

response(500, [
    'code' => 500,
    'message' => 'Penghapusan data gagal!'
]);

```

Sama dengan create.php dan complete.php beda-nya hanya query yang kita kirimkan ke database adalah penghapusan suatu data.

Nah REST Server kita sudah selesai tinggal client kita, kita bisa menggunakan file html yang kemarin kita buat dengan sedikit pengubahan.

Dan di file index.html

```html
<body>
    <h2>App Todo List</h2>
    <form id='form'>
        <input type="text" name="Todo" id="input">
        <input type="submit" value="Tambah">
    </form>
    <div id="app">
        <!--  -->
    </div>
</body>
```

Dan untuk Javascript-nya kita tambah

```javascript
// siapkan html element id form dan input-nya
const form = document.getElementById('form');
const input = document.getElementById('input');

// Siapkan base URI yang akan kita pakai (Kalau beda bisa diganti)
// Mungkin temen temen pengen localhost/nama-folder
const BASE_URI = '127.0.0.1:8000'

// Ini request header yang nanti kita include
// ini guna-nya untuk nge-set tipe data json yang kita kirim dan kita menerima json juga
// ini digunakan din function
const REQUEST_HEADER = {
    'Content-Type': 'application/json',
    'Accept': 'application/json'
};
```

Lalu kita bisa ambil form dan input itu tadi, dan kita juga definisikan variabel BASE_URI karena setiap orang pasti beda folder di htdocs (dan kalau punya-ku ini 127.0.0.1:8000 merupakan server yang aku nyalain sendiri biar cepat praktek), lalu variabel REQUEST_HEADER kita definisikan tipe yang kita kirimkan (karena kita akan melakukan request ke server dengan data tidak seperti yang kemarin), lalu kita juga menambah Accept untuk memberitahu server kalau kita menerima juga json

dan di function appendTodoToApp(todoObject) kita ganti seperti ini

```
function appendTodoToApp(todoObject) {
    const id = todoObject.id;
    const judul = todoObject.judul;
    const selesai = checkMark(todoObject.selesai);
    const html = `<div id="todo-${id}">
    <h2>${judul}</h2>
    <span>Status : ${selesai}</span>
    <button onclick='selesaiTodo(this)'>selesai</button>
    <button onclick='hapusTodo(this)'>hapus</button>
</div>`

    app.innerHTML += html;
}
```

Lalu kita men-definisikan function baru yang digunakan untuk mengambil id didalam String yang kita berikan di setiap todo

```
function getId(string) {
    return string.split('-', 2)[1]
}
```

Nah function ini hanya memanggil split dengan patokan ‘-’ dan kita hanya menggambil 2 saja, itu function jadi me-return array dan kita hanya mengambil di index 1 (bukan index pertama tetapi kedua karena setiap index di pemrograman mulai dari angka 0 kecuali bahasa pemrograman lua)

Lalu kita bisa ganti fetch() yang digunakan untuk mengambil data di server

```
// Buat request ke REST Server
fetch(`http://${BASE_URI}/todo`)
    // Karena promise kita bisa tambahkan method then
    // then ini akan melakukan konversi terhadap response-nya
    .then(res => res.json())
    // Karena res.json() return promise juga kita dapat menambahkan then lagi
    // then kali ini akan membuat html elemen todo-nya
    .then(data => {
        data.data.forEach(todoElement => appendTodoToApp(todoElement));
    })
    // catch ini ketika ada error di promise
    .catch(err => console.error(err));
```

Nah ada hal yang menarik disini kita menggunakan `` tidak “” atau ‘’ karena kita ingin menggunakan string format yang di sediakan oleh Javascript agar kita bisa menaruh variabel di dalam String itu tadi.

```
form.onsubmit = (event) => {
    event.preventDefault();
    const payload = {
        "judul": input.value
    }

    // Kita kirim request ke server dengan membawa data yang sudah kita dapatkan
    fetch(`http://${BASE_URI}/todo`, {
        // kita set method-nya ke post
        method: 'POST',
        headers: REQUEST_HEADER,
        // karena di headers kita kirim json, kita harus serialize dulu
        body: JSON.stringify(payload)
    }).then(res => res.json())
        .then(data => {
            appendTodoToApp(data.todo);
        })
        .catch(err => console.error(err));
}
```

Kita menambahkan event listener di <form> tadi dengan event onsubmit, tugas-nya ialah mengirimkan data yang di-input di client ke server dengan HTTP Method POST dan menggunakan headers yang kita definisikan tadi, dan karena kita mengirimkan JSON kita harus Serialize data dulu dengan menggunakan JSON.stringify(payload), variabel payload merupakan data yang akan kita kirim, setelah itu ketika di server mengirimkan response ke kita kita bisa langgsung men-selipkan todo yang baru di browser.

Lalu kita definisikan function hapusTodo yang menerima parameter event (karena kita panggil langsung dengan event listener onclick yang tadi kita definisikan lewat function appendTodoToApp)

```
function hapusTodo(event) {
    const parentElement = event.parentElement;
    // kita split id todo tadi dengan mengandalkan - yang ada di id attribute
    // method ini return array yang berisi 2 karena kita menspesifikan 2
    // lalu kita ambil yang terakhir
    const id = getId(parentElement.id);
    fetch(`http://${BASE_URI}/todo/delete.php?id=${id}`, {
        method: 'DELETE',
        headers: REQUEST_HEADER,
        // kalau sukses kita langsung delete element-nya
    }).then(res => app.removeChild(parentElement))
        // kalau gagal kita kirimkan ke console
        .catch(err => console.error(err));
}
```

Hal pertama yang akan kita lakukan ialah mengambil parent Element, karena event ini memiliki elemen yang kita klik tadi jadi parent-nya adalah dan itu bisa kita ambil id-nya lalu kita masukan ke function getId() untuk mengambil id-nya

Lalu kita bisa melakukan fetch() ke server untuk penghapusan data, dan kita menggunakan HTML Elemen yang ber-id app untuk menghapus variabel parentElement

```
function selesaiTodo(event) {
    const parentElement = event.parentElement;
    // kita split id todo tadi dengan mengandalkan - yang ada di id attribute
    // method ini return array yang berisi 2 karena kita menspesifikan 2
    // lalu kita ambil yang terakhir
    const id = getId(parentElement.id);
    fetch(`http://${BASE_URI}/todo/complete.php?id=${id}`, {
        method: 'PATCH',
        headers: REQUEST_HEADER,
    })
        .then(res => {
            const childNodes = parentElement.childNodes;
            for (let i = 0; i < childNodes.length; i++) {
                if (childNodes[i] instanceof HTMLSpanElement) {
                    childNodes[i].textContent = checkMark(1);
                    break;
                }
            }
        })
        // kalau gagal kita kirimkan ke console
        .catch(err => console.error(err));
}
```

Hampir sama dengan yang hapusTodo() beda-nya setelah request-nya di-terima kita melakukan for loop untuk mencari dan diganti isi-nya dengan menggunakan checkMark(1)

Nah selesai sudah ini, sangat banyak yang harus di ketik tetapi ini tidak seberapa dengan project yang di-dunia nyata karena bisa mencapai 3000 baris, kalau ini 97 Kode PHP, 74 kode Javascript, 20 kode HTML dengan total 191 baris (tidak meng-hitung baris yang berisi komentar atau kosong).

Note : untuk variabel BASE_URI value-nya bisa diganti ke *.localhost/nama-folder-kalian*