

RESTful API dengan Framework go-chi :

Fitur Meliputi :

1. GET/books (Menampilkan Data Seluruh Buku)

routes.go

```
r.Route("/books", func(r chi.Router) {  
    r.Get("/", handlers.GetBooks)  
})
```

Kode pada **routes.go** ini mendefinisikan rute dengan awalan **/books** menggunakan router **chi**. Baris **r.Get("/", handlers.GetBooks)** mengarahkan permintaan **GET /books** ke fungsi **GetBooks** di package **handlers**. Ini merupakan implementasi prinsip RESTful untuk mengambil daftar buku melalui endpoint **/books**.

book_handler.go

```
func GetBooks(w http.ResponseWriter, r *http.Request) {  
    books := store.GetAllBooks()  
    utils.RespondJSON(w, http.StatusOK, books)  
}
```

Fungsi *GetBooks* pada **book_handler.go** bertugas menangani permintaan **GET /books**. Fungsi ini mengambil seluruh data buku dari penyimpanan dengan memanggil *store.GetAllBooks()*, lalu mengirimkannya sebagai respons JSON menggunakan *utils.RespondJSON* dengan status HTTP 200.

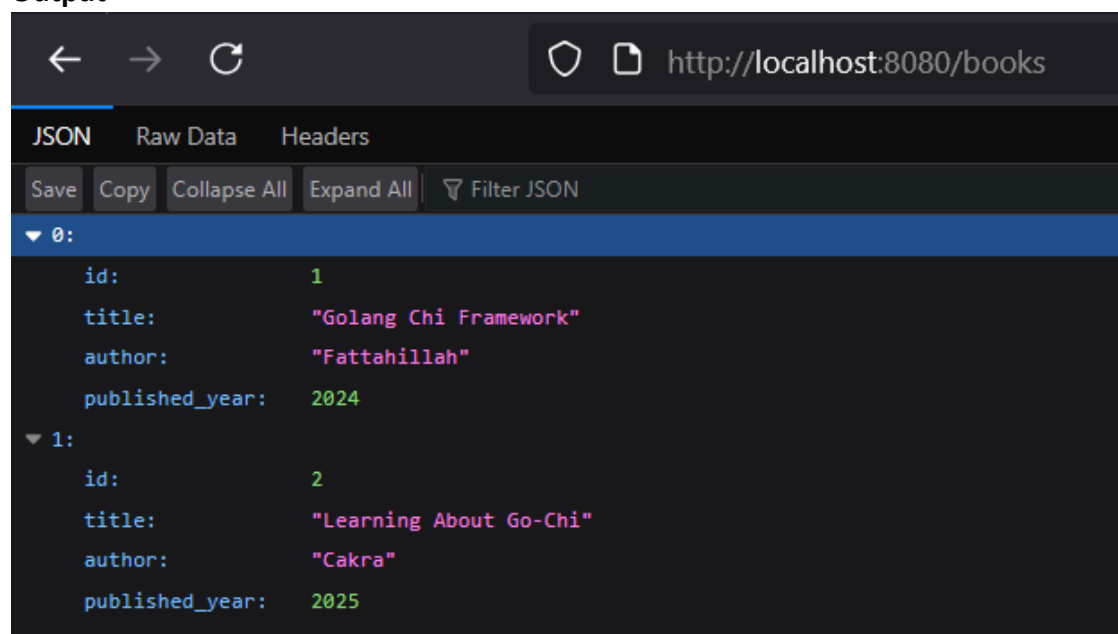
storage.go

```
func GetBookStore() *BookStore {  
    once.Do(func() {  
        instance = &BookStore{  
            Books: []models.Book{  
                {ID: 1, Title: "Golang Chi Framework", Author: "Fattahillah", PublishedYear: 2024},  
                {ID: 2, Title: "Learning About Go-Chi", Author: "Cakra", PublishedYear: 2025},  
            },  
            lastID: 2,  
        }  
    })  
    return instance  
}
```

```
func (s *BookStore) GetAllBooks() []models.Book {  
    return s.Books  
}
```

Pada `storage.go`, data buku disimpan di dalam `struct BookStore` menggunakan slice `Books`. Fungsi `GetBookStore()` menginisialisasi singleton `BookStore` dengan dua data statis awal, dan fungsi `GetAllBooks()` mengembalikan seluruh isi slice tersebut. Dengan demikian, saat `GetBooks` dipanggil, data yang ditampilkan berasal langsung dari memori, sesuai syarat penyimpanan data in-memory pada case study.

Output



2. GET/books{id} (Menampilkan Data Buku Berdasarkan ID)

routes.go

```
r.Get("/{id}", handlers.GetBookByID)
```

Baris di atas mendefinisikan rute untuk endpoint `GET /books/{id}`. Artinya, saat permintaan HTTP GET dikirim ke `/books` dengan ID tertentu (contoh: `/books/2`), permintaan tersebut akan diteruskan ke fungsi `GetBookByID` yang berada di package `handlers`. Ini merupakan implementasi RESTful untuk mengambil satu *resource* (buku) berdasarkan IDnya.

book_handler.go

```
func GetBookByID(w http.ResponseWriter, r *http.Request) {
    id, ok := utils.ParseIDParam(w, r)
    if !ok {
        return
    }

    book, found := store.GetBookByID(id)
    if !found {
        utils.RespondNotFound(w, "Book")
        return
    }

    utils.RespondJSON(w, http.StatusOK, book)
}
```

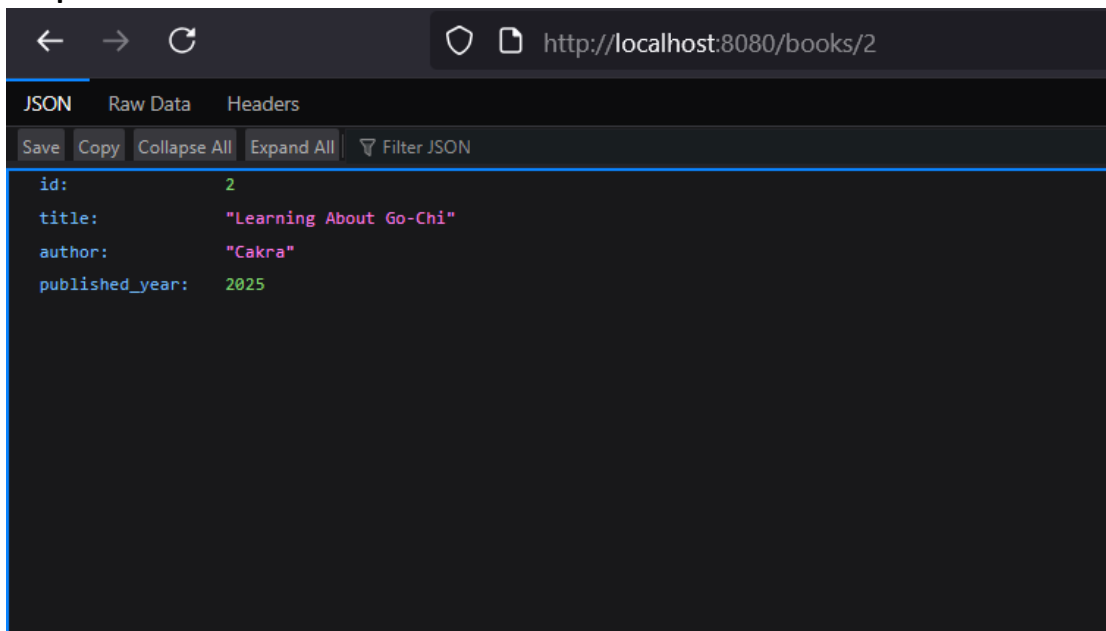
Fungsi ini menerima permintaan *GET /books/{id}*. Pertama, ID diambil dari parameter URL menggunakan helper *ParseIDParam*. Jika ID tidak valid, fungsi berhenti dan merespons *error 400*. Jika valid, data buku dicari di penyimpanan melalui *store.GetBookByID(id)*. Bila ditemukan, data buku dikirim sebagai respons JSON dengan status 200. Jika tidak ditemukan, respons *error 404* dikirim menggunakan *RespondNotFound*.

storage.go

```
func (s *BookStore) GetBookByID(id int) (models.Book, bool) {
    for _, b := range s.Books {
        if b.ID == id {
            return b, true
        }
    }
    return models.Book{}, false
}
```

Fungsi ini mencari buku berdasarkan ID dari slice *Books*. Jika ditemukan, data dikembalikan bersama nilai *true*. Jika tidak, mengembalikan data kosong dan *false*. Ini memastikan handler bisa membedakan antara data yang ada dan tidak ada.

Output



3. POST/books (Menambahkan Data Buku)

routes.go

```
r.Post("/", handlers.CreateBook)
```

Baris ini mendaftarkan rute *POST /books* dan mengarahkannya ke fungsi *CreateBook* di *package handlers*. Endpoint ini digunakan untuk menambahkan data buku baru ke sistem, sesuai prinsip REST untuk membuat *resource / buku* baru.

book_handler.go

```
func CreateBook(w http.ResponseWriter, r *http.Request) {
    var book models.Book
    if !utils.DecodeJSONBody(w, r, &book) {
        return
    }

    book = store.AddBook(book)
    utils.RespondJSON(w, http.StatusCreated, book)
}
```

Fungsi ini membaca data buku dari body permintaan dalam format JSON, lalu menyimpannya ke dalam penyimpanan menggunakan *store.AddBook*. Jika data JSON tidak valid, akan dikembalikan error 400. Jika berhasil, buku baru dikembalikan dalam respons JSON dengan status 201 (*Created*), sesuai standar REST.

Storage.go

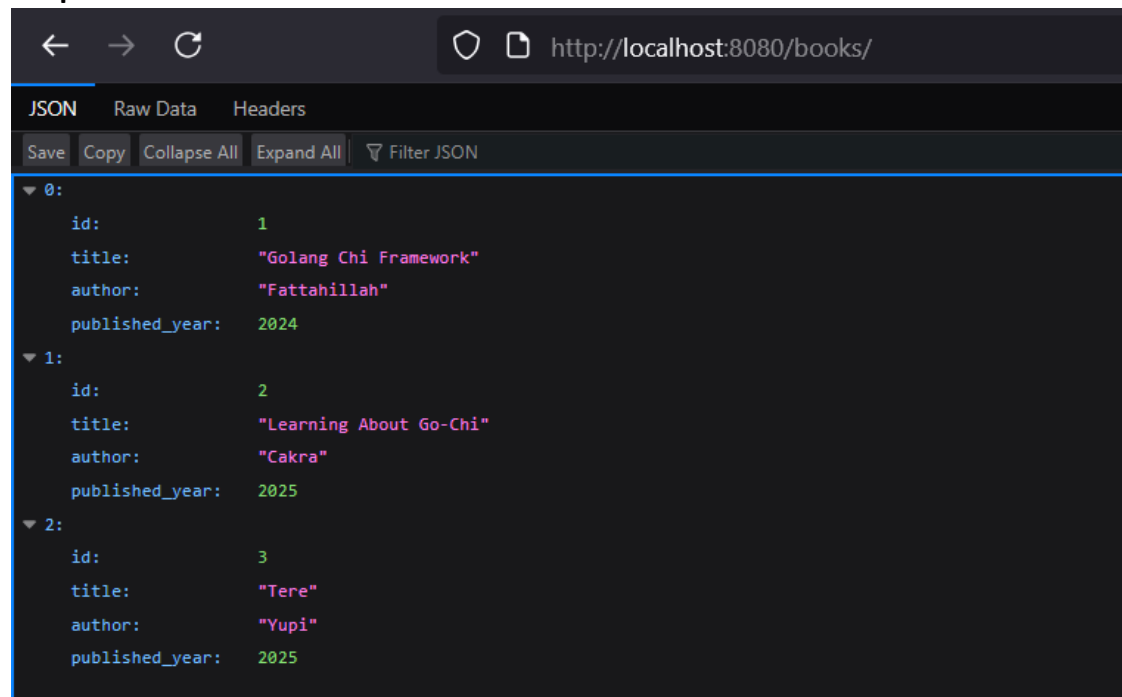
```
func (s *BookStore) AddBook(book models.Book) models.Book {  
    s.mu.Lock()  
    defer s.mu.Unlock()  
  
    s.lastID++  
    book.ID = s.lastID  
    s.Books = append(s.Books, book)  
    return book  
}
```

Fungsi ini menambahkan buku baru ke slice Books, secara otomatis menetapkan ID baru berdasarkan *lastID*. Fungsi ini juga aman digunakan secara paralel karena menggunakan *sync.Mutex*.

Input (Menggunakan Powershell)

```
$body = @{  
    title = "Tere"  
    author = "Yupi"  
    published_year=2025  
} | ConvertTo-Json -Depth 2  
  
Invoke-RestMethod -Uri http://localhost:8080/books `   
    -Method POST `   
    -Body $body `   
    -ContentType "application/json"
```

Output



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/books/`. Below the address bar, there are tabs for "JSON", "Raw Data", and "Headers", with "JSON" selected. There are also buttons for "Save", "Copy", "Collapse All", "Expand All", and a "Filter JSON" input field. The main content area displays a JSON array of three book objects, each with an `id`, `title`, `author`, and `published_year` property.

```
{  
  "0": {  
    "id": 1,  
    "title": "Golang Chi Framework",  
    "author": "Fattahillah",  
    "published_year": 2024  
  },  
  "1": {  
    "id": 2,  
    "title": "Learning About Go-Chi",  
    "author": "Cakra",  
    "published_year": 2025  
  },  
  "2": {  
    "id": 3,  
    "title": "Tere",  
    "author": "Yupi",  
    "published_year": 2025  
  }  
}
```

4. PUT/books{id} (Edit Data Buku)

routes.go

```
r.Put("/{id}", handlers.UpdateBook)
```

Baris ini mendefinisikan rute *PUT /books/{id}* dan mengarahkannya ke fungsi *UpdateBook*. Endpoint ini digunakan untuk memperbarui data buku berdasarkan ID, sesuai dengan konsep REST untuk update resource.

book_handler.go

```
func UpdateBook(w http.ResponseWriter, r *http.Request) {
    id, ok := utils.ParseIDParam(w, r)
    if !ok {
        return
    }

    var updatedBook models.Book
    if !utils.DecodeJSONBody(w, r, &updatedBook) {
        return
    }

    book, ok := store.UpdateBook(id, updatedBook)
    if !ok {
        utils.RespondNotFound(w, "Book")
        return
    }

    utils.RespondJSON(w, http.StatusOK, book)
}
```

Fungsi ini mengambil ID dari parameter URL dan membaca data buku baru dari body JSON. Jika ID tidak valid atau format JSON salah, fungsi akan merespons error. Jika buku ditemukan, data diperbarui dan dikembalikan sebagai JSON dengan status 200. Jika tidak ditemukan, dikembalikan respons 404.

Storage.go

```
func (s *BookStore) UpdateBook(id int, updated models.Book)
(models.Book, bool) {
    for i, b := range s.Books {
        if b.ID == id {
```

```
        updated.ID = id
        s.Books[i] = updated
        return updated, true
    }
}
return models.Book{}, false
}
```

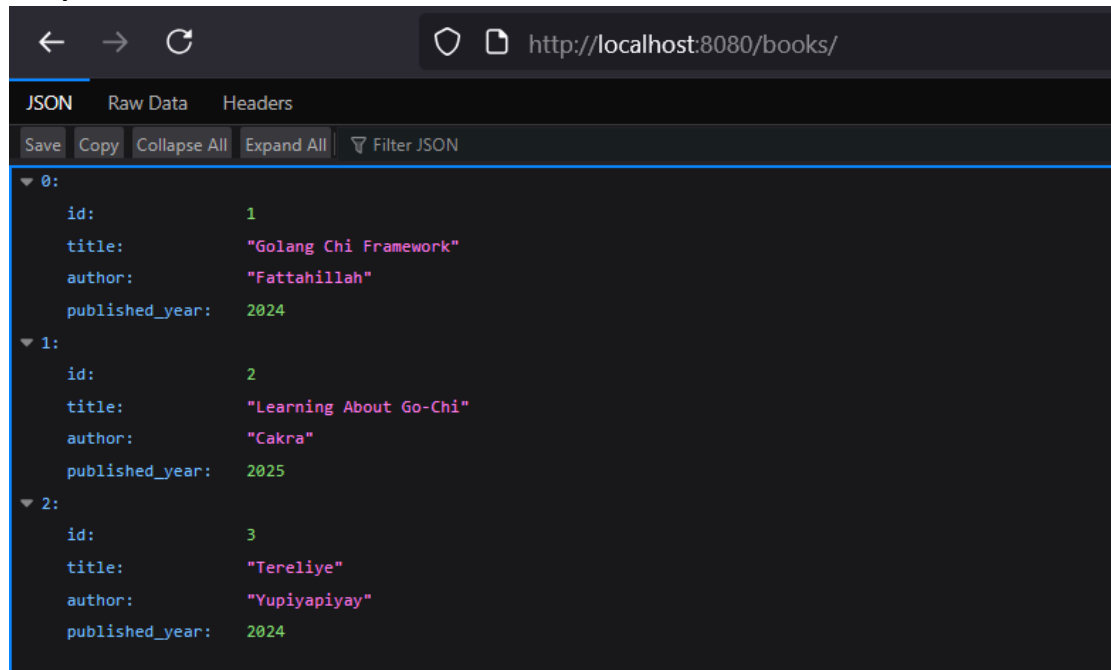
Fungsi ini mencari buku berdasarkan ID. Jika ditemukan, data lama diganti dengan data baru sambil mempertahankan ID-nya. Jika tidak ditemukan, fungsi mengembalikan *false*.

Input (Menggunakan Powershell)

```
$body = @{
    title = "Tereliye"
    author = "Yupiyapiyay"
    published_year=2024
} | ConvertTo-Json
```

```
Invoke-RestMethod -Uri http://localhost:8080/books/3 `
    -Method PUT `
    -Body $body `
    -ContentType "application/json"
```

Output



```
{
  "0": {
    "id": 1,
    "title": "Golang Chi Framework",
    "author": "Fattahillah",
    "published_year": 2024
  },
  "1": {
    "id": 2,
    "title": "Learning About Go-Chi",
    "author": "Cakra",
    "published_year": 2025
  },
  "2": {
    "id": 3,
    "title": "Tereliye",
    "author": "Yupiyapiyay",
    "published_year": 2024
  }
}
```

5. DELETE/books{id} (Hapus Data Buku)

routes.go

```
r.Delete("/{id}", handlers.DeleteBook)
```

Rute ini menetapkan bahwa permintaan HTTP *DELETE* ke */books/{id}* akan diarahkan ke fungsi *DeleteBook* di *package handlers*. Endpoint ini berfungsi untuk menghapus data buku berdasarkan ID tertentu, sesuai prinsip REST untuk operasi penghapusan resource.

book_handler.go

```
func DeleteBook(w http.ResponseWriter, r *http.Request) {
    id, ok := utils.ParseIDParam(w, r)
    if !ok {
        return
    }

    if ok := store.DeleteBook(id); !ok {
        utils.RespondNotFound(w, "Book")
        return
    }

    utils.RespondJSON(w, http.StatusOK, map[string]string{"message":
"Book deleted successfully"})
}
```

Fungsi ini memvalidasi ID dari URL, lalu memanggil *store.DeleteBook(id)* untuk menghapus buku. Jika buku tidak ditemukan, akan dikembalikan respons 404. Jika berhasil dihapus, akan dikirimkan pesan sukses dalam format JSON dengan status 200 (OK).

Storage.go

```
func (s *BookStore) DeleteBook(id int) bool {
    for i, b := range s.Books {
        if b.ID == id {
            s.Books = append(s.Books[:i], s.Books[i+1:]...)
            return true
        }
    }
    return false
}
```

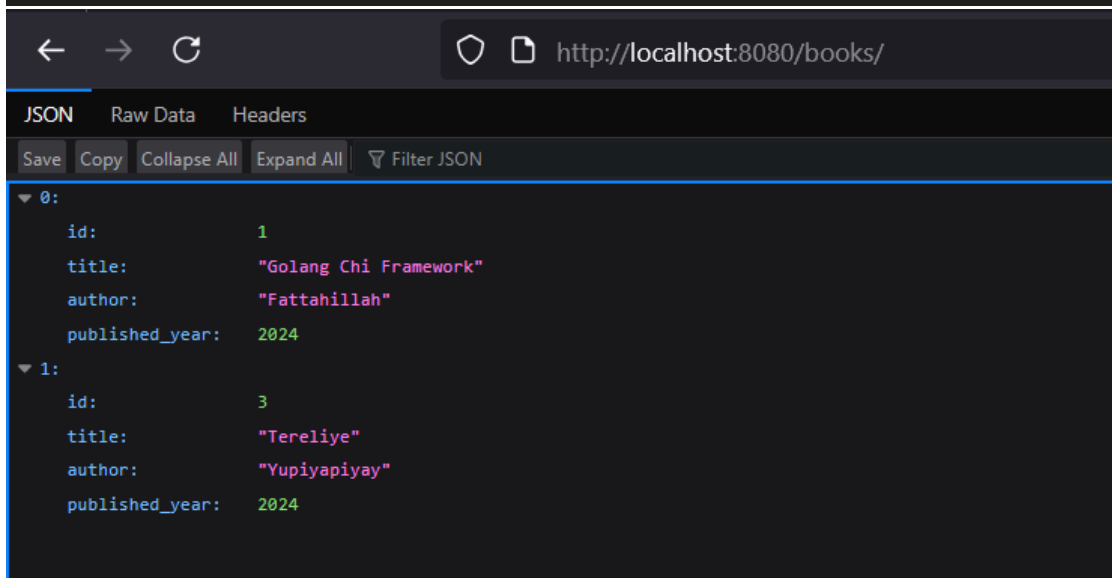

Fungsi ini mencari buku berdasarkan ID, dan jika ditemukan, menghapusnya dari slice *Books* menggunakan slicing. Jika buku berhasil dihapus, mengembalikan *true*; jika tidak ditemukan, mengembalikan *false*.

Input (Menggunakan Powershell)

```
Invoke-RestMethod -Uri http://localhost:8080/books/2 -Method DELETE
```

Output

```
PS F:\gochi-book-api> Invoke-RestMethod -Uri http://localhost:8080/books/2 -Method DELETE
message
-----
Book deleted successfully
```



← → ↻ http://localhost:8080/books/

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
▼ 0:
  id: 1
  title: "Golang Chi Framework"
  author: "Fattahillah"
  published_year: 2024
▼ 1:
  id: 3
  title: "Tereliye"
  author: "Yupiyapiyay"
  published_year: 2024
```