

PROJECT

เว็ขายหนังสือ ออนไลน์

เว็ขายหนังสือ

1. เพื่อศึกษาวิธีการออกแบบ **WEB SERVER** ด้วย **APIARY**
2. เพื่ออำนวยความสะดวกในการเลือกซื้อหนังสือ
3. เพื่อศึกษาวิธีการเขียนตัว **WEB SERVER** ด้วย ภาษา **RUST**

เว็ขายหนังสือ

- 1.เพื่อความสะดวกในการ เลือกซื้อหนังสือ
- 2.ฝึกฝนการใช้ **HTTP METHODS** และเรียนรู้การ ออกแบบ **API**
- 3.เรียนรู้วิธีการใช้ภาษา **RUST**

เป้าหมาย

RUST FILE STRUCTURE

Main

Main.rs

```
src >  main.rs > ...  
1 use actix_web::{App, HttpServer,};  
2 use env_logger::Env;  
3 pub mod routes;  
4 pub mod handlers;  
5 pub mod models;  
6 use crate::routes::*;  
7  
8 #[actix_web::main]  
  ▶ Run | Debug  
9 async fn main() -> std::io::Result<()> {  
10     env_logger::Builder::from_env(Env::default().default_filter_or("debug")).init();  
11     HttpServer::new(|| {  
12         App::new()  
13             .configure(profile_route::config)  
14             .configure(list_route::config)  
15             .configure(add_route::config)  
16             .configure(update_route::config)  
17             .configure(delete_route::config)  
18     })  
19     .bind("127.0.0.1:8080")?  
20     .run()  
21     .await  
22 }
```

Handlers

handlers/add_handlers.rs

```
src > handlers > add_handlers.rs > create_book
1 use actix_web::{post, web, HttpResponse};
2 use crate::models::list::*;
3 use serde_json::json;
4
5 #[post("/books/{id}")]
6 async fn create_book(category: web::Json<RequestBook>, list_id: web::Path<i32>) -> HttpResponse {
7     let req = category.into_inner();
8     let id: i32 = list_id.to_string().parse().unwrap();
9     let messageerror = "Not Found".to_string();
10    let date = "2023-03-19T08:40:51.620Z";
11    let book = RequestBook{
12        category:req.category,
13        list:req.list
14    };
15    if id != 20 {
16        let respondbook = DeleteMessage{
17            message: messageerror
18        };
19        let response_body = json!(respondbook);
20        HttpResponse::NotFound().json(response_body)
21    } else {
22        let respondbook = BookCategory{
23            category: book.category,
24            published_at: date.to_string(),
25            list: book.list
26        };
27        let response_body = json!(respondbook);
28        HttpResponse::Created().json(response_body)
29    }
30 }
31 }
32
33
```

Handlers

handlers/delete_handlers.rs

src > handlers > delete_handlers.rs > delete_book

```
1 use actix_web::{delete, web, HttpResponse};
2 use crate::models::list::*;
3 use serde_json::json;
4
5 #[delete("/books/{id}")]
6 async fn delete_book(category: web::Json<DeleteBook>, list_id: web::Path<i32>) -> HttpResponse {
7     let req: DeleteBook = category.into_inner();
8     let id: i32 = list_id.to_string().parse().unwrap();
9     let message: String = "Delete Complete".to_string();
10    let messageerror: String = "Id Not Found".to_string();
11    let mut _book: DeleteBook = DeleteBook{
12        category: req.category,
13        list: req.list
14    };
15
16    if id != 20 {
17        let responsebook: DeleteMessage = DeleteMessage{
18            message: messageerror
19        };
20        let response_body: Value = json!(responsebook);
21        HttpResponse::NotFound().json(response_body)
22    }
23    else {
24        let responsebook: DeleteMessage = DeleteMessage{
25            message: message
26        };
27        let response_body: Value = json!(responsebook);
28        HttpResponse::Ok().json(response_body)
29    }
30 } fn delete_book
```

Handlers

handlers/list_handlers.rs

```
src > handlers > @ list_handlers.rs > @ get_books
1 use actix_web::{get, HttpResponse};
2 use crate::models::list::Book;
3 use crate::models::list::BookList;
4 use crate::models::list::BookCategory;
5 //use serde_json::json;
6
7 #[get("/books")]
8 async fn get_books() -> HttpResponse {
9     let book_list: BookList = BookList {
10         books: vec![
11             BookCategory {
12                 category: String::from("kid"),
13                 published_at: String::from("2023-03-19T08:40:51.620Z"),
14                 list: vec![
15                     Book {
16                         book_id: 1,
17                         book_name: String::from("Little Red Riding Hood"),
18                         price: 100,
19                         stock: 6,
20                     },
21                     Book {
22                         book_id: 2,
23                         book_name: String::from("Peterpan"),
24                         price: 120,
25                         stock: 8,
26                     },
27                     Book {
28                         book_id: 3,
29                         book_name: String::from("Alice In Wonderland"),
30                         price: 100,
31                         stock: 12,
32                     },
33                     Book {
34                         book_id: 4,
35                         book_name: String::from("Pinocchio"),
36                         price: 120,
37                         stock: 15,
```


Handlers

handlers/list_handlers.rs

```
src > handlers > list_handlers.rs > get_books
38         },
39     ],
40 },
41 BookCategory {
42     category: String::from("education"),
43     published_at: String::from("2023-03-19T08:50:51.620Z"),
44     list: vec![
45         Book {
46             book_id: 5,
47             book_name: String::from("คิดไม่ต่องคิด"),
48             price: 100,
49             stock: 2,
50         },
51         Book {
52             book_id: 6,
53             book_name: String::from("ไทยวิบัติ"),
54             price: 120,
55             stock: 6,
56         },
57         Book {
58             book_id: 7,
59             book_name: String::from("เคมีแห่งนรก"),
60             price: 100,
61             stock: 3,
62         },
63         Book {
64             book_id: 8,
65             book_name: String::from("แพทกกฎหมาย"),
66             price: 120,
67             stock: 1,
68         },
69     ],
70 },
71 ],
72 };
73 HttpResponse::Ok().json(book_list)
74 } fn get_books
```

Handlers

handlers/profile_handlers.rs

```
src > handlers > 📄 profile_handlers.rs > 📦 get_id
1  use actix_web::{get, Responder, HttpResponse};
2  use crate::models::profile::Profile;
3  use serde_json::json;
4
5  #[get("/profile")]
6  async fn get_id() -> impl Responder{
7      let id: i32 = 20;
8      let profile: Vec<Profile> = vec![
9          Profile {
10             id: id,
11             username: "Pond".to_string()
12         }
13     ];
14     let response_body: Value = json!(profile);
15
16     HttpResponse::Ok().json(response_body)
17 }
```



Handlers

handlers/update_handlers.rs

```
src > handlers > update_handlers.rs > update_book
4
5  #[put("/books/{id}")]
6  async fn update_book(category: web::Json<RequestBook>, list_id: web::Path<i32>) -> HttpResponse {
7      let req: RequestBook = category.into_inner();
8      let id: i32 = list_id.to_string().parse().unwrap();
9      let messageerror: String = "Not Found".to_string();
10     let date: &str = "2023-03-19T08:40:51.620Z";
11     let book: RequestBook = RequestBook{
12         category: req.category,
13         list: req.list
14     };
15     if id != 20 {
16         let respondbook: DeleteMessage = DeleteMessage{
17             message: messageerror
18         };
19         let response_body: Value = json!(respondbook);
20         HttpResponse::NotFound().json(response_body)
21     }else {
22         let respondbook: BookCategory = BookCategory{
23             category: book.category,
24             published_at: date.to_string(),
25             list: book.list
26         };
27         let response_body: Value = json!(respondbook);
28         HttpResponse::Ok().json(response_body)
29     }
30
31 } fn update_book
32
33
```

Models

models/list.rs

```
src > models >  list.rs >  DeleteBook
1  use serde::{Serialize, Deserialize};
2
3  #[derive(Serialize, Deserialize)]
4  2 implementations
5  pub struct Book {
6      pub book_id: u32,
7      pub book_name: String,
8      pub price: u32,
9      pub stock: u32,
10 }
11
12 #[derive(Serialize, Deserialize)]
13 2 implementations
14 pub struct BookCategory {
15     pub category: String,
16     pub published_at: String,
17     pub list: Vec<Book>,
18 }
19
20 #[derive(Serialize, Deserialize)]
21 2 implementations
22 pub struct BookList {
23     pub books: Vec<BookCategory>,
24 }
25
26 #[derive(Serialize, Deserialize)]
27 2 implementations
28 pub struct RequestBook {
29     pub category: String,
30     pub list: Vec<Book>,
31 }
```

Models

models/list.rs


src > models > list.rs > DeleteBook

```
27
28  #[derive(Serialize, Deserialize)]
    2 implementations
29  pub struct BookId {
30      pub book_id: u32
31  }
32
33  #[derive(Serialize, Deserialize)]
    2 implementations
34  pub struct DeleteBook {
35      pub category: String,
36      pub list: Vec<BookId>,
37  }
38
39  #[derive(Serialize, Deserialize)]
    2 implementations
40  pub struct DeleteMessage {
41      pub message: String
42  }
```

Model



models/profile.rs

src > models >  profile.rs > ...

```
1 use serde::{Serialize, Deserialize};  
2   
3 #[derive(Serialize, Deserialize)]  
  2 implementations  
4 pub struct Profile {  
5     pub id: i32,  
6     pub username: String,  
7  
8 }
```

Routes


routes/add_route.rs

```
src > routes >  add_route.rs >  config
```

```
1 use actix_web::web;  
2 use crate::handlers::add_handlers::create_book;  
3 pub fn config(cfg: &mut web::ServiceConfig){  
4     cfg.service(factory: create_book);  
5 }
```

Routes

routes/delete_route.rs

```
src > routes >  delete_route.rs > ...
```

```
1 use actix_web::web;  
2 use crate::handlers::delete_handlers::delete_book;  
3 pub fn config(cfg: &mut web::ServiceConfig){  
4     cfg.service(factory: delete_book);  
5 }
```





Routes

routes/profile_route.rs

```
src > routes > 📄 profile_route.rs > 📦 config
1   use actix_web::web;
2   use crate::handlers::profile_handlers::get_id;
3
4   pub fn config(cfg: &mut web::ServiceConfig){
5       |   cfg.service(factory: get_id);
6   }
```



Routes

routes/update_route.rs

```
src > routes >  update_route.rs >  config  
1 use actix_web::web;  
2 use crate::handlers::update_handlers::update_book;  
3  pub fn config(cfg: &mut web::ServiceConfig){  
4     | cfg.service(factory: update_book);  
5 }
```

Routes

routes/list_route.rs

src > routes >  list_route.rs >  config

```
1 use actix_web::web;  
2 use crate::handlers::list_handlers::get_books;  
3 pub fn config(cfg: &mut web::ServiceConfig){  
4     cfg.service(factory: get_books);  
5 }
```

Mod

routes/mod.rs

```
src > routes >  mod.rs > {} delete_route
```

```
1   pub mod profile_route;
```

```
2   pub mod list_route;
```


```
3   pub mod add_route;
```

```
4   pub mod update_route;
```

```
5   pub mod delete_route;
```

Mod

handlers/mod.rs

```
src > handlers >  mod.rs > {} delete_handlers
```

```
1  pub mod profile_handlers;  
2  pub mod list_handlers;  
3  pub mod add_handlers;  
4  pub mod update_handlers;  
5  pub mod delete_handlers;
```

Mod

models/mod.rs

```
src > models >  mod.rs > {} list
```

```
1   pub mod profile;
```

```
2   pub mod list;
```

คณะผู้จัดทำ

ยชญ์อนันต์ พิชญานนทพัฒน์
ศรุต มาลามูลศรี
ฐานวัฒน์ บรรดาศักดิ์
ปริญ จันทรสิน



**THANK
YOU**