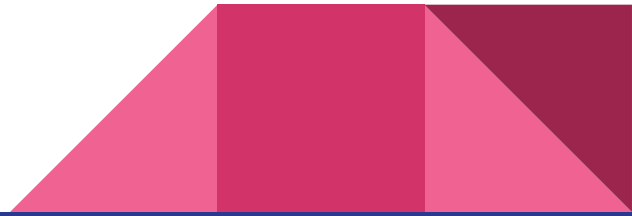
Rust untuk Membangun Aplikasi Web: Rocket, Diesel, Handlebars

Bambang Purnomosidi D. P.
@bpdp

Rust Indonesia Meetup #2 - 24 September 2017
Padepokan Asa - Yogyakarta

Agenda

- Komponen Aplikasi Web dan Rust
- Setup dan Konfigurasi
- Integrasi Komponen
- Workflow



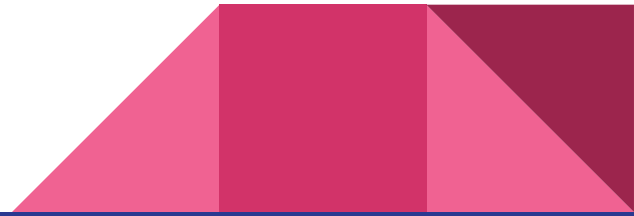
Komponen Aplikasi Web dan Rust

- Pada umumnya, aplikasi Web mengikuti design pattern MVC (Model - View - Controller)
- Request diterima Controller, diproses dengan melakukan query ke database (Model) maupun tanpa query ke db. Hasilnya akan diberikan ke View dan dikirimkan ke client yang me-request dalam bentuk response.
- Model biasanya ditangani oleh ORM (*Object-Relational Mapping*, istilah yang tidak terlalu tepat di Rust, tapi sudah dianggap defacto di dunia pengembangan software. View biasanya ditangani oleh *template engine*.
- Secara minimum, Rust sudah mempunyai komponen-komponen tersebut:
    - Rocket: Web framework yang menyediakan controller untuk menerima request, mencocokkan dengan routing, memproses ke handler sesuai routing
    - ORM: Diesel
    - View: Handlebars masuk sebagai Rocket-contrib)

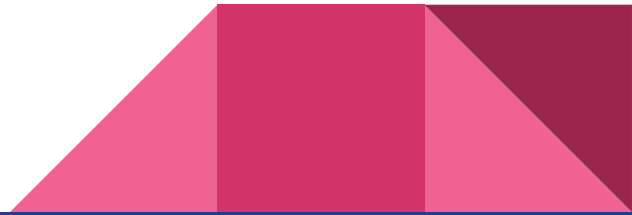
Setup dan Konfigurasi

- Rocket memerlukan Rust Nightly
- Diesel memerlukan Rust Stable
- Keduanya bisa menggunakan Rust Nightly
    - cargo new nama-proyek --bin

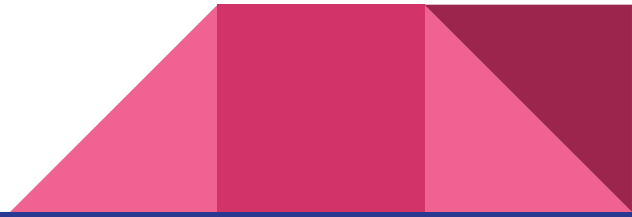
```
[package]

name = "rocket-diesel-handlebars"

version = "0.0.1"
```



```
[dependencies]

rocket = "0.3.*"

rocket_codegen = "0.3.*"

serde = "1.0.*"

serde_derive = "1.0.*"

serde_json = "1.0.*"

diesel = { version = "0.16.*", features = ["postgres"] }

diesel_codegen = { version = "0.16.*", features = ["postgres"] }

dotenv = "0.10.*"
```



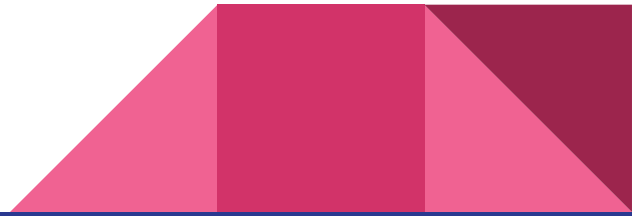
```
[dependencies.rocket_contrib]

version = "0.3.*"

default-features = false

features = ["handlebars_templates"]⏎
```



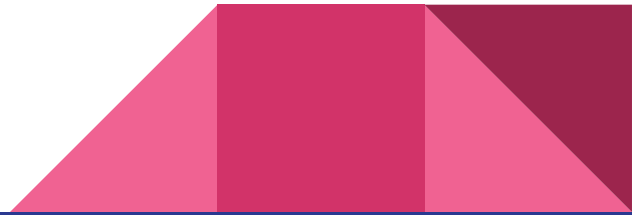
Integrasi Komponen

- Request, Routing, dan Handler:

```
#[get("/")]
    fn index() -> Redirect {
        Redirect::to("/books")
}
```



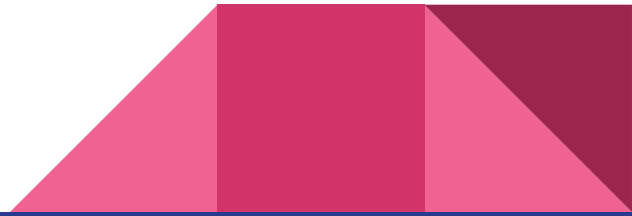
```
#[get("/hello/<name>/<age>/<cool>")]
  fn hello(name: String, age: u8, cool: bool) -> String {
     if cool {
          format!("You're a cool {} year old, {}!", age, name)
       } else {
          format!("{}, we need to talk about your coolness.", name)
       }
     }
```



Integrasi Komponen: ORM

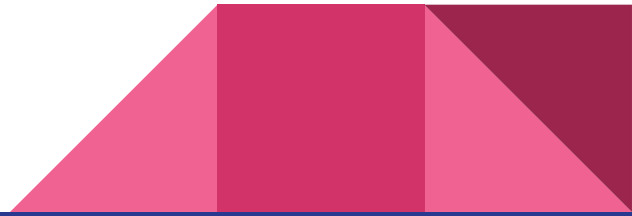
- ORM: Object-Relational Mapping
- Satu ORM, bisa untuk lebih dari satu jenis DB.
- ORM di Rust: Diesel
- Perlu install diesel_cli untuk keperluan migration tool
- Minimum, pada source code terdapat:
  - schema.rs: untuk skema basis data
  - model.rs: konversi ke struct Rust
  - Modul untuk koneksi ke db, query, insert, dst (bisa diletakkan di src/lib.rs sebagai contoh)
- Pada handler dari routing, query ke db setelah sebelumnya koneksi ke db
- Hasil query dilemparkan ke handlebars template

```
let connection = establish_connection();

let items = books.filter()

    .limit(5)

    .load::<Book>(&connection)

    .expect("Error loading books");
```

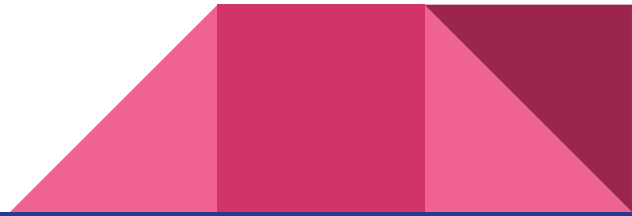


Handlebars

- Berada pada dir **templates/*.hbs**

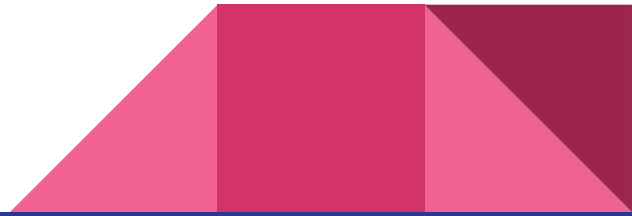


```
<h1>Daftar Buku</h1>

<h3>Berikut adalah daftar buku yang ada di perpustakaan:</h3>

<ul>

  {{#each bookslists}}

  <li>{{this}}</li>

  {{/each}}

</ul>
```



Workflow

- Request => Routing => Validation => Processing Logic (to ORM or not to ORM) => Response



The End

Terima kasih!