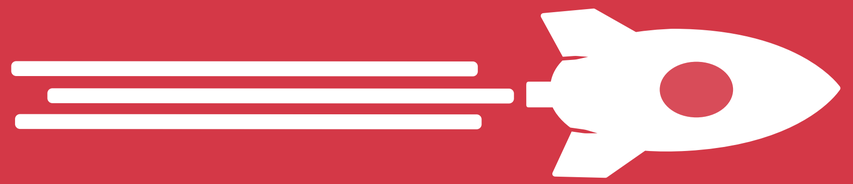
Rocket

Sergio Benitez

sb@sergio.bz



1. Introduction to Rocket

2. Code Generation in Rocket and Rust

3. What’s Next?

1 Introduction to Rocket

- Simple, Fast, Type-Safe Web Framework
- Powered by Rust's Code Generation Facilities
- Enables Secure, Robust Web Applications

2 Code Generation in Rocket and Rust

3 What's Next?

1 Introduction to Rocket

2 Code Generation in Rocket and Rust

- Demystifying the “Magic” of Code Generation
- Present and Future Code Generation APIs

3 What’s Next?

1 Introduction to Rocket

2 Code Generation in Rocket and Rust

3 What's Next?

- What's Coming in Future Versions of Rocket
- Code Generation at Large

1

Introduction to Rocket

Simple, Fast, Type-Safe Web Framework

2

Code Generation in Rocket and Rust

Demystifying the “Magic” of Code Generation

3

What’s Next?

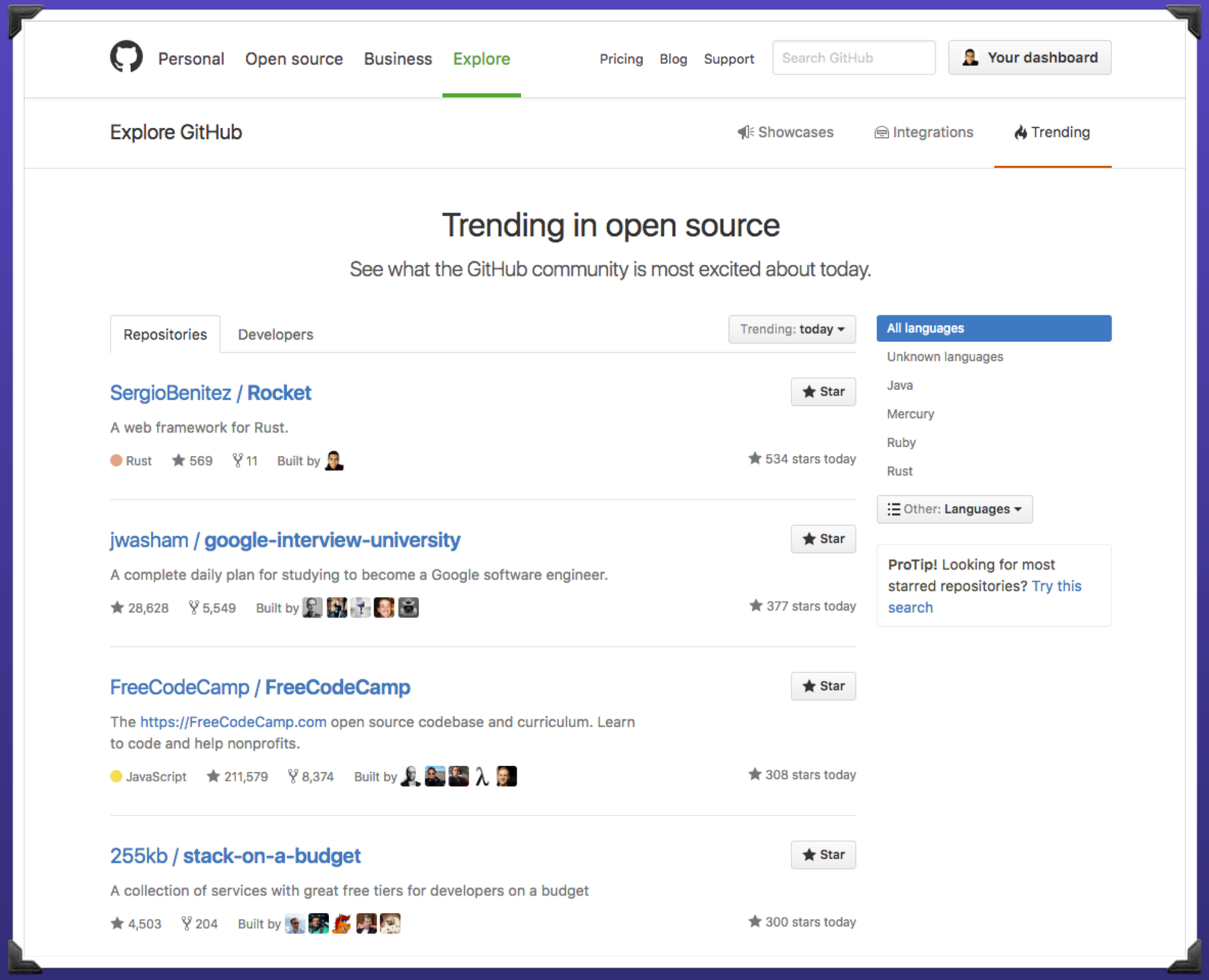
What’s Coming in Future Versions of Rocket

1 Introduction to Rocket

Rocket is a web framework for Rust that makes it simple to write fast web applications without sacrificing flexibility or type safety.


☆2,670
Launch!
Dec. 23, 2016
v0.2
Feb. 06, 2017
v0.3
Jul. 14, 2017


in production
2,670
Launch!
Dec. 23, 2016
v0.2
Feb. 06, 2017
v0.3
Jul. 14, 2017


Personal
Open source
Business
Explore
Pricing
Blog
Support
Search GitHub
Your dashboard
Explore GitHub
Showcases
Integrations
Trending
Trending in open source
See what the GitHub community is most excited about today.
Repositories
Developers
Trending: today
SergioBenitez / Rocket
Star
A web framework for Rust.
Rust
569
11
Built by
534 stars today
jwasham / google-interview-university
Star
A complete daily plan for studying to become a Google software engineer.
28,628
5,549
Built by
377 stars today
FreeCodeCamp / FreeCodeCamp
Star
The https://FreeCodeCamp.com open source codebase and curriculum. Learn to code and help nonprofits.
JavaScript
211,579
8,374
Built by
308 stars today
255kb / stack-on-a-budget
Star
A collection of services with great free tiers for developers on a budget
4,503
204
Built by
300 stars today
All languages
Unknown languages
Java
Mercury
Ruby
Rust
Other: Languages
ProTip! Looking for most starred repositories? Try this search


☆2,670
Launch!
Dec. 23, 2016
v0.2
Feb. 06, 2017
v0.3
Jul. 14, 2017

Early 2016

Launch!
Dec. 23, 2016

v0.2
Feb. 06, 2017

Rocket is a web framework for Rust that makes it simple to write fast web applications without sacrificing flexibility or type safety.

Routes (Hello, world!)

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
```

Routes (Hello, world!)

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
```

1. Route **Attribute:** Description of *matching* condition.

Routes (Hello, world!)

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
```

1 Route **Attribute:** Description of *matching* condition.

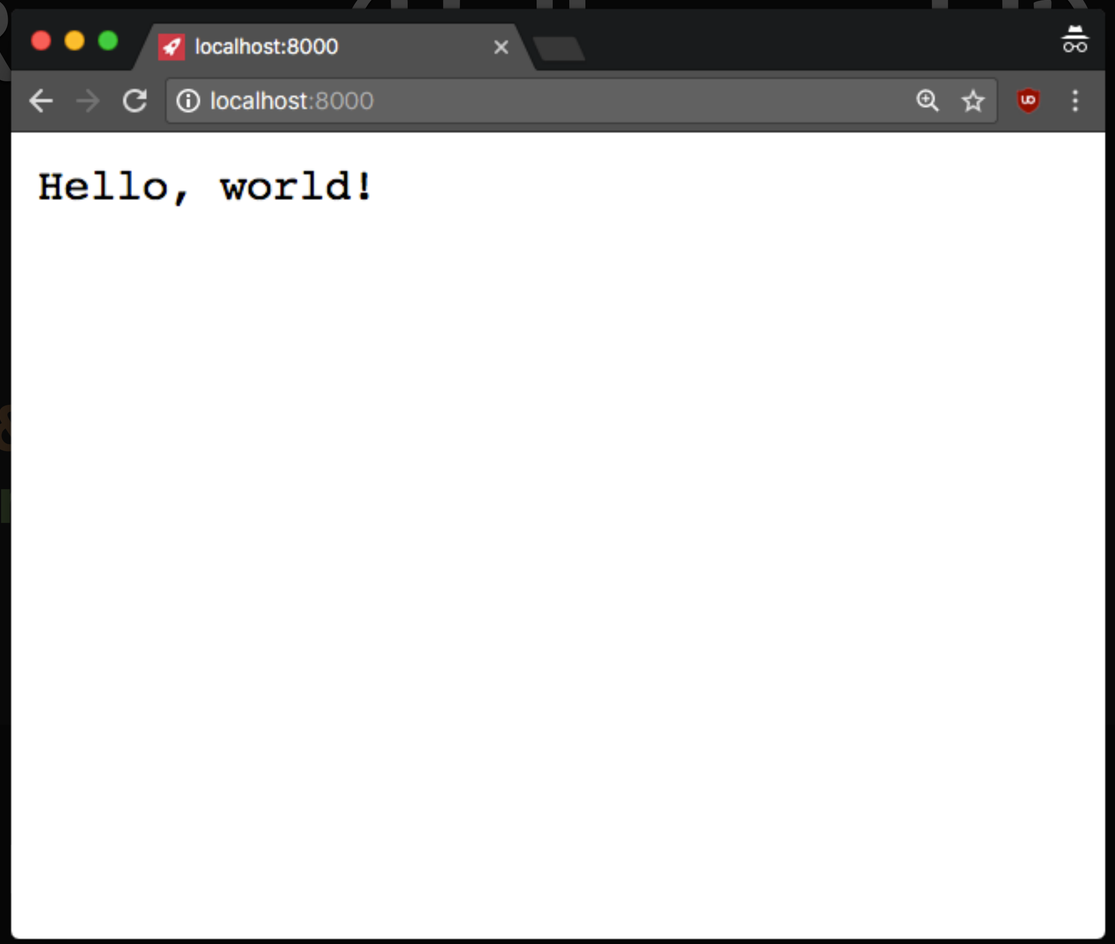
2 Route **Handler:** Request processing, produces response.

Routes (Hello, world!)

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
```

1. Route **Attribute:** Description of *matching* condition.
2. Route **Handler:** Request processing, produces response.

```
1  #[get("/")]
2  fn hello() -> &
3      "Hello, wor
4  }
```



1. Route **Attribute:**
2. Route **Handler:** Request processing, produces response.

Routes (Hello, world!)

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
```

1. Route **Attribute:** Description of *matching* condition.
2. Route **Handler:** Request processing, produces response.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Routes need to be ***mounted*** to make Rocket aware.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Mounting *namespaces* routes according to a root path.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/hello", routes![hello]).launch();
8  }
```

Mounting *namespaces* routes according to a root path.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Mounting *namespaces* routes according to a root path.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Launching starts up the server.

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Launching starts up the server. (also prints emojis 🚀)

Mounting & Launching

```
🔧 Configured for development.
    => address: localhost
    => port: 8000
    => log: normal
    => workers: 16
    => secret key: generated
    => limits: forms = 32KiB
    => tls: disabled
🛰  Mounting '/':
    => GET / (hello)
🚀 Rocket has launched from http://localhost:8000
```

Launching starts up the server. (also prints emojis 🚀)

Mounting & Launching

```
1  #[get("/")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Mounting & Launching

```
1  #[get("/world")]
2  fn hello() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Mounting & Launching

```
1  #[get("/sergio")]
2  fn hello() -> &'static str {
3      "Hello, Sergio!"
4  }
5
6  fn main() {
7      rocket::ignite().mount("/", routes![hello]).launch();
8  }
```

Rocket is a web framework for Rust that makes it simple to write fast web applications without sacrificing flexibility or type safety.

Dynamic Paths

Dynamic Paths

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, age: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> **match *any* text in segment.**

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, age: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Name in <brackets> must have matching function argument.

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, number: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Name in <brackets> must have matching function argument.

Dynamic Paths

```
error: 'age' is declared as an argument...
 --> src/main.rs:9:16
  |
9 | #[get("/<name>/<age>")]
  |                ^^^^^

error: ...but isn't in the function signature.
  --> src/main.rs:10:1
   |
10 | / fn hello(name: String, number: u8) -> String {
11 | |     format!("Hello, {} year old named {}!", age, name)
12 | | }
   | |_^
```

Parameters i

- Name in <brackets> must have matching function argument.

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, number: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Name in <brackets> must have matching function argument.

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, age: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Name in <brackets> must have matching function argument.

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, age: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Type of dynamic parameter is declared in handler signature.
- *Any* type implementing `FromParam` is allowed.

Dynamic Paths

```
1  #[get("/<name>/<age>")]
2  fn hello(name: String, age: u8) -> String {
3      format!("Hello, {} year old named {}!", age, name)
4  }
```

Parameters in <brackets> match *any* text in segment.

- Handler can only be called if `FromParam` conversion succeeds.

Preventing Directory Traversal

```
1  #[get("/<path..>")]
2  fn files(path: PathBuf) -> Option<NamedFile> {
3      NamedFile::open(Path::new("static/").join(path)).ok()
4  }
```

Implication: handlers are only called with validated data!

- `FromParam`* implementation for `PathBuf` verifies path safety.

Request Guards

Request Guards

```
1  #[get("/admin")]
2  fn admin_panel(admin: AdminUser) -> &'static str { ... }
```

Arbitrary number of `FromRequest` parameters allowed.

Request Guards

```
1  #[get("/admin")]
2  fn admin_panel(admin: AdminUser) -> &'static str { ... }
```

Request guards validate incoming request, protect handlers.

- Guards may *fail*, preventing further request processing.
- Guards may *forward*, indicating local failure.

Request Guards & Forwarding

```rust
#[get("/admin")]
fn admin_panel(admin: AdminUser) -> &'static str { ... }

#[get("/admin", rank = 2)]
fn admin_panel_user(user: User) -> &'static str { ... }
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```
1  #[get("/admin")]
2  fn admin_panel(admin: AdminUser) -> &'static str { ... }
3
4  #[get("/admin", rank = 2)]
5  fn admin_panel_user(user: User) -> &'static str { ... }
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```
1  #[get("/admin")]
2  fn admin_panel(admin: AdminUser) -> &'static str { ... }
3
4  #[get("/admin", rank = 2)]
5  fn admin_panel_user(user: User) -> &'static str { ... }
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```
🔧 Configured for development.
    => address: localhost
    => port: 8000
    => log: normal
    => workers: 16
    => secret key: generated
    => limits: forms = 32KiB
    => tls: disabled
🛰  Mounting '/':
    => GET /admin (admin_panel)
    => GET /admin (admin_panel_user)
Error: Rocket failed to launch due to the following routing collisions:
    => GET /admin (admin_panel) collides with GET /admin (admin_panel_user)
    => Note: Collisions can usually be resolved by ranking routes.
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```
🔧 Configured for development.
    => address: localhost
    => port: 8000
    => log: normal
    => workers: 16
    => secret key: generated
    => limits: forms = 32KiB
    => tls: disabled
🛰  Mounting '/':
    => GET /admin (admin_panel)
    => GET /admin (admin_panel_user)
Error: Rocket failed to launch due to the following routing collisions:
    => GET /admin (admin_panel) collides with GET /admin (admin_panel_user)
    => Note: Collisions can usually be resolved by ranking routes.
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```rust
#[get("/admin")]
fn admin_panel(admin: AdminUser) -> &'static str { ... }

#[get("/admin", rank = 2)]
fn admin_panel_user(user: User) -> &'static str { ... }
```

Forwarding: Rocket attempts *colliding* routes in ascending rank order.

Request Guards & Forwarding

```
#[get("/admin")]
fn admin_panel(admin: AdminUser) -> &'static str { ... }

#[get("/admin", rank = 2)]
fn admin_panel_user(user: User) -> &'static str { ... }

#[get("/admin", rank = 3)]
fn admin_panel_redirect() -> Redirect { ... }
```

1 Introduction to Rocket

1

Introduction to Rocket

Simple, Fast, Type-Safe Web Framework

2

Code Generation in Rocket and Rust

Demystifying the “Magic” of Code Generation

3

What’s Next?

What’s Coming in Future Versions of Rocket

2 Code Generation in Rocket and Rust

Hello, World! *Revisited*

```
1  #[get("/")]
2  fn hi() -> &'static str {
3      "Hello, world!"
4  }
5
6  fn main() {
7      rocket::ignite()
8          .mount("/", routes![hi])
9          .launch();
10 }
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
..
.mount("/", vec![Route::from(&hi_info)])
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
...
.mount("/", vec![Route::from(&hi_info)])
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
..
.mount("/", vec![Route::from(&hi_info)])
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
...
.mount("/", vec![Route::from(&hi_info)])
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
..
.mount("/", vec![Route::from(&hi_info)])
```

Hello, World! *Revisited*

```
#[get("/")]
fn hi() -> &'static str {
    "Hello, world!"
}

fn main() {
    rocket::ignite()
        .mount("/", routes![hi])
        .launch();
}
```

```
static hi_info = RouteInfo {
    name: "hi",
    method: Method::Get,
    path: "/",
    handler: hi_route,
    format: None,
    rank: None,
};

fn hi_route(req: &Request) -> Outcome {
    let responder = hi();
    Outcome::from(req, responder)
}
..
.mount("/", vec![Route::from(&hi_info)])
```

Guards & Params

```
#[get("/item/<id>")]
fn item(user: User, id: u32) {
    ...
}
```

Guards & Params

```
#[get("/item/<id>")]
fn item(user: User, id: u32) {
    ...
}
```

```
static item_info = RouteInfo {
    name: "item",
    method: Method::Get,
    path: "/item/<id>",
    handler: item_route,
    format: None,
    rank: None,
};
```

Guards & Params

```
#[get("/item/<id>")]
fn item(user: User, id: u32) {
    ...
}
```

Guards & Params

```
fn item(user: User, id: u32)
```

```
fn item_route(req: &Request) -> Outcome {
    let id_param = match req.param_str(0).map(u32::from_param) {
        Some(Ok(v)) => v,
        _ => return Outcome::Forward
    };

    let user_param: User = match User::from_request(req) {
        Outcome::Success(v) => v,
        Outcome::Forward(_) => return Outcome::Forward,
        Outcome::Failure(code) => return Outcome::Failure(code)
    };

    let responder = item(id_param, user_param);
    Outcome::from(req, responder)
}
```

Guards & Params

```
fn item(user: User, id: u32)
```

```
fn item_route(req: &Request) -> Outcome {
    let id_param = match req.param_str(0).map(u32::from_param) {
        Some(Ok(v)) => v,
        _ => return Outcome::Forward
    };

    let user_param: User = match User::from_request(req) {
        Outcome::Success(v) => v,
        Outcome::Forward(_) => return Outcome::Forward,
        Outcome::Failure(code) => return Outcome::Failure(code)
    };

    let responder = item(id_param, user_param);
    Outcome::from(req, responder)
}
```

Guards & Params

```
fn item(user: User, id: u32)
```

```
fn item_route(req: &Request) -> Outcome {
    let id_param = match req.param_str(0).map(u32::from_param) {
        Some(Ok(v)) => v,
        _ => return Outcome::Forward
    };

    let user_param: User = match User::from_request(req) {
        Outcome::Success(v) => v,
        Outcome::Forward(_) => return Outcome::Forward,
        Outcome::Failure(code) => return Outcome::Failure(code)
    };

    let responder = item(id_param, user_param);
    Outcome::from(req, responder)
}
```

Guards & Params

```
fn item(user: User, id: u32)
```

```
fn item_route(req: &Request) -> Outcome {
    let id_param = match req.param_str(0).map(u32::from_param) {
        Some(Ok(v)) => v,
        _ => return Outcome::Forward
    };

    let user_param: User = match User::from_request(req) {
        Outcome::Success(v) => v,
        Outcome::Forward(_) => return Outcome::Forward,
        Outcome::Failure(code) => return Outcome::Failure(code)
    };

    let responder = item(id_param, user_param);
    Outcome::from(req, responder)
}
```

Code Generation

```
#[get("/item/<id>")]
fn item(user: User, id: u32) {
    ...
}
```

```
static item_info = RouteInfo { .. };
fn item_route(req: &Request) -> Outcome;
vec![Route::from(&item_info)]
```

- Static route information from route attribute.
- Monomorphized route handler from attribute and handler signature.
- Tied together through mounted structure.

Internals

Proc. Macro API

```
fn(Syntax) -> Syntax;
```

Proc. Macro API

```
fn(Syntax) -> Syntax;
```

```
struct Syntax(Vec<TokenTree>);

struct TokenTree {
    node: TokenNode,
    span: Span
}
```

Proc. Macro API

```
fn(Syntax) -> Syntax;
```

```
struct Syntax(Vec<TokenTree>);

struct TokenTree {
    node: TokenNode,
    span: Span
}
```

Proc. Macro API

```
fn(Syntax) -> Syntax;
```

```
struct Syntax(Vec<TokenTree>);

struct TokenTree {
    node: TokenNode,
    span: Span
}
```

2 Code Generation in Rocket and Rust

1

Introduction to Rocket

Simple, Fast, Type-Safe Web Framework

2

Code Generation in Rocket and Rust

Demystifying the “Magic” of Code Generation

3

What’s Next?

What’s Coming in Future Versions of Rocket

3 What’s Next?

Typed URIs

```
#[get("/<id>")]
fn retrieve(id: PasteId) -> Option<Plain<File>>> {
    let filename = format!("upload/{id}", id = id);
    File::open(&filename).map(|f| content::Plain(f)).ok()
}
```

Typed URIs

```
#[get("/<id>")]
fn retrieve(id: PasteId) -> Option<Plain<File>>> {
    let filename = format!("upload/{id}", id = id);
    File::open(&filename).map(|f| content::Plain(f)).ok()
}
```

```
url!(retrieve: id = PasteId(100));
```

```
url!(retrieve: id = 100);
```

Typed URIs

```rust
#[get("/<id>")]
fn retrieve(id: PasteId) -> Option<Plain<File>>> {
    let filename = format!("upload/{id}", id = id);
    File::open(&filename).map(|f| content::Plain(f)).ok()
}
```

```rust
url!(retrieve: id = PasteId(100));
```

```rust
url!(retrieve: id = 100);
```

```
"/100"
```

Database Support

Today

```
type Pool = r2d2::Pool<ConnectionManager<SqliteConnection>>;

static DATABASE_URL: &'static str = env!("DATABASE_URL");

fn init_pool() -> Pool {
    let config = r2d2::Config::default();
    let manager = ConnectionManager::<SqliteConnection>::new(DATABASE_URL);
    r2d2::Pool::new(config, manager).expect("db pool")
}

pub struct DbConn(pub r2d2::PooledConnection<ConnectionManager<SqliteConnection>>);

impl<'a, 'r> FromRequest<'a, 'r> for DbConn {
    type Error = ();

    fn from_request(request: &'a Request<'r>) -> request::Outcome<DbConn, ()> {
        let pool = request.guard::<State<Pool>>()?;
        match pool.get() {
            Ok(conn) => Outcome::Success(DbConn(conn)),
            Err(_) => Outcome::Failure((Status::ServiceUnavailable, ()))
        }
    }
}
```

In Rocket v0.4

Rocket.toml

```toml
[[database]]
name = "my_db"
adapter = "diesel_sqlite"
```

main.rs

```rust
#[derive(DbConn)]
#[database("my_db")]
struct DbConn(SqliteConnection);
```

3 What’s Next?

1

Introduction to Rocket

Simple, Fast, Type-Safe Web Framework

2

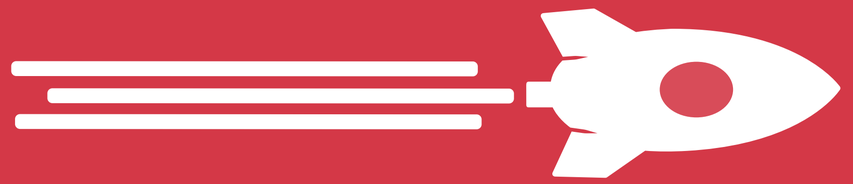
Code Generation in Rocket and Rust

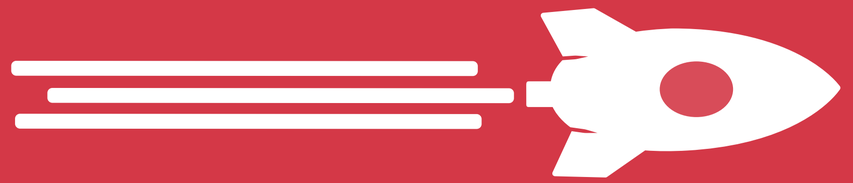
Demystifying the “Magic” of Code Generation

3

What’s Next?

What’s Coming in Future Versions of Rocket





myths

Too Much “Magic”

~~Too Much “Magic”~~

Unstable

~~Unstable~~

Forever on Nightly

~~Forever on Nightly~~

Rocket

https://rocket.rs

guide, tutorial, docs, news, code

Sergio Benitez

sb@sergio.bz