Serve It With Rust

Writing web servers with Rust and Gotham

Turki Jamaan

Why would you use Rust to write a server?

🤔

Why would you use Rust to write a server?

- Rust is a systems language
  - Code is compiled to binaries that can be run directly
  - No runtime overhead

Why would you use Rust to write a server?

- Rust does not require a garbage collector
  - No GC memory overhead
  - No GC pauses
  - Predictable performance

Why would you use Rust to write a server?

- Strong parallelism and concurrency support
  - Able to use resources more efficiently
  - Essential for servers

Why would you use Rust to write a server?

- Rust lifetime system takes care of memory management without resorting to garbage collection
- Rust’s borrowing rules prevents classes of bugs from compiling
  - Can’t alias and mutate at the same time => No more data races
    - Other kinds of races and deadlocks can happen though!
  - Can’t refer to data longer than it lives => No more use-after-free

Why would you use Rust to write a server?

- Rust’s type system is powerful
  - Pattern matching
  - Traits
  - Rust Enums (a.k.a. Algebraic Data Types)
  - Etc…

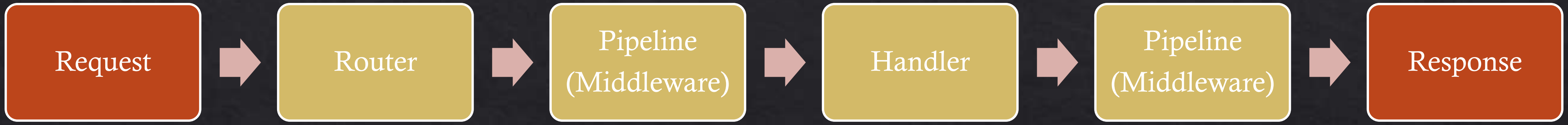
What is Gotham

- Gotham is a Rust web framework that provides a safe and fast way to write web servers.
- Gotham is written to be as concurrent as possible
  - Gotham makes use of Tokio and Hyper to achieve concurrency
  - Tokio provides the platform for asynchronous code
    - Provides asynchronous types like Futures, Streams, and Sinks
    - Provides the event loop mechanism to drive the asynchronous types
  - Hyper provides an asynchronous HTTP framework built on top of Tokio
    - Handles the HTTP protocol
    - Is really really fast!

What is Gotham

- Gotham makes use of Tokio and Hyper to achieve concurrency
- Tokio provides the platform for asynchronous code
  - Provides asynchronous types like Futures, Streams, and Sinks
  - Provides the event loop mechanism to drive the asynchronous types
- Hyper provides an asynchronous HTTP framework built on top of Tokio
  - Handles the HTTP protocol
  - Is really really fast!

How does Gotham work?



Handler

- A plain old function. It’s that simple really.
- Takes a State and a Request.
- Returns a State and a Response.

Router

- Looks at the request and decides which pipeline and handler to apply
- Has a tree of routes.

Pipeline

- Each pipeline is a list of Middlewares
- Middlewares interact with Requests and can add extra state information
- The biggest example is the default session middleware provided by Gotham

Let’s create a website!

- We will create a small website to demonstrate Gotham
- Our route tree looks like this:
  - /
  - /capitalize?text={string}
  - /cube/{number}
  - /session_visit_counter