

Web Development

In Rust

Christoph Beberweil 2025-05-28



@privat Christine Weidlich

What is web development?

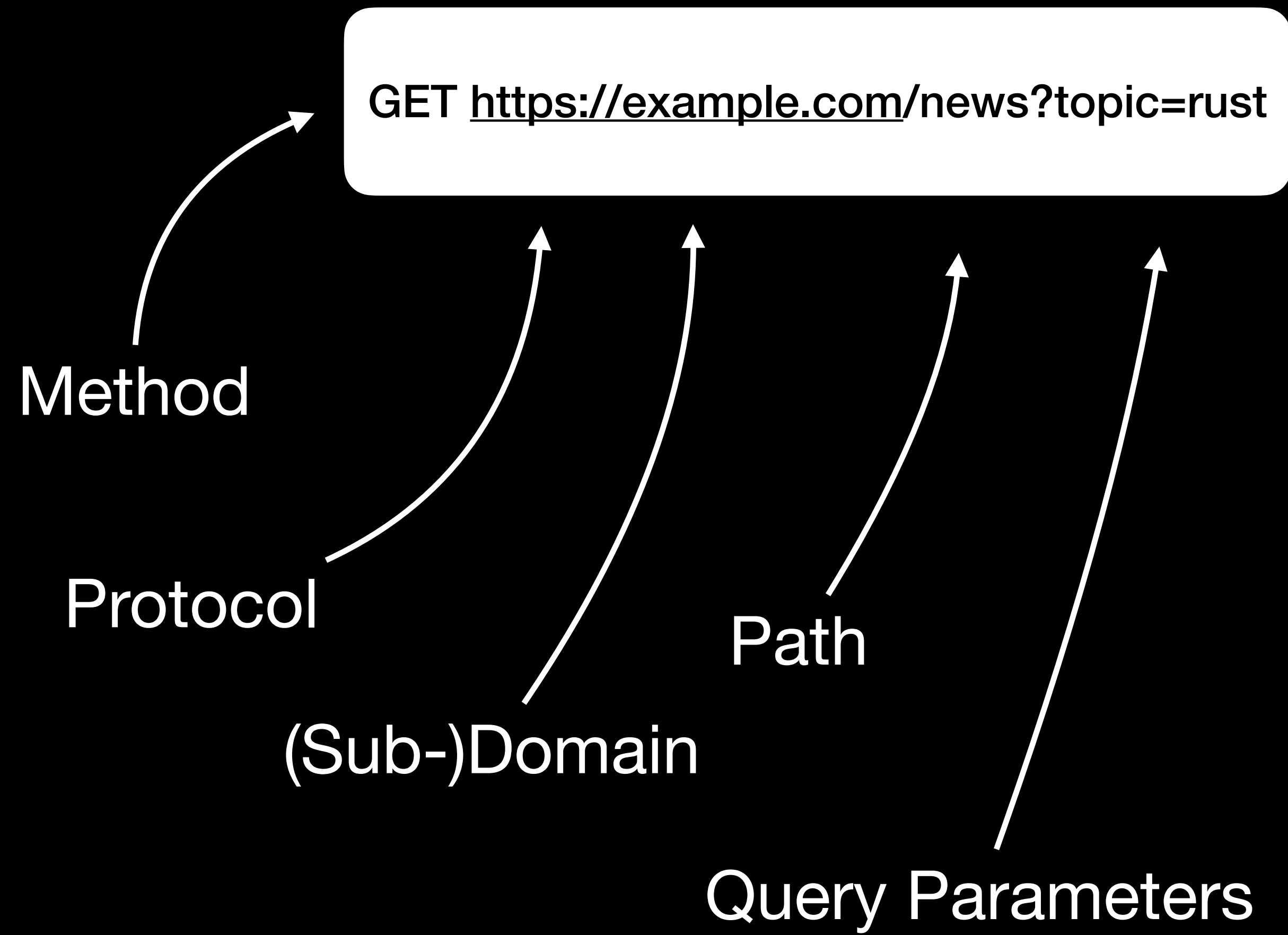


- HTTP Protocol, Headers, Body, Status Codes, Request Methods, Query Params
- Authentication & Authorisation, Session Cookies, JWT, ...
- Workloads in the background
- HTML, CSS, JS
- JSON
- Databases: Postgres, MySQL, MongoDB
- ORM
- “Cloud”
- Security: Cross Site Scripting(CSS), Cross Site Request Forgery (CSRF), ...

Why would I want a Web Application?

- We control the code and the server at all times
- Deployment (installation and updates) is easy
- Monitoring is easy
- One system could scale to many users, especially with Rust
- Delegate heavy workload to the server
- Single source of truth
- Collaboration between many users

A HTTP Request



Headers

Request

```
GET / HTTP/2
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:138.0)
Gecko/20100101 Firefox/138.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
DNT: 1
Upgrade-Insecure-Requests: 1
Connection: keep-alive
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache
```

Response

```
HTTP/2 200
accept-ranges: bytes
content-type: text/html
etag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
last-modified: Mon, 13 Jan 2025 20:11:20 GMT
vary: Accept-Encoding
content-encoding: gzip
cache-control: max-age=1387
date: Tue, 27 May 2025 13:23:02 GMT
alt-svc: h3=":443"; ma=93600,h3-29=":443"; ma=93600,quic=":443";
ma=93600; v="43"
content-length: 648
X-Firefox-Spdy: h2
```

Cookies

Only visible for domain



Not visible to JS



Not send to everyone



Use HTTPS



```
Sessionid: "gsegskaslda3et3qsc"  
Created: "Wed, 28. May 2025 18:00:00 GMT"  
Domain: "test.example.com"  
Expires/Max-Age: "Thu, 29. May 2025 18:00:00  
GMT"  
HostOnly: true  
HttpOnly: true  
Last Accessed: "Wed, 28. May 2025 18:14:12  
GMT"  
Path: "/"  
SameSite: "Strict"  
Secure: true  
Size: 41
```

JSON Web Tokens

Header

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Claims

```
{  
  "id": "924a7165-653e-4eef-a8a6-3ff5489ff931",  
  "name": "John Doe",  
  "admin": true,  
  "tier": "platin"  
}
```

Microservices can validate the signed tokens and do not need to access the user database.

HTML

```
<form method="get" id="searchForm">
  <div>
    <label for="name" class="customLabel">Name</label>
    <input type="text" name="name" id="name" class="formInput"
      required>
  </div>
  <div>
    <label for="url" class="customLabel">Url</label>
    <input type="url" name="url" id="url" class="formInput" required>
  </div>
  <button type="submit">Search</button>
</form>
```

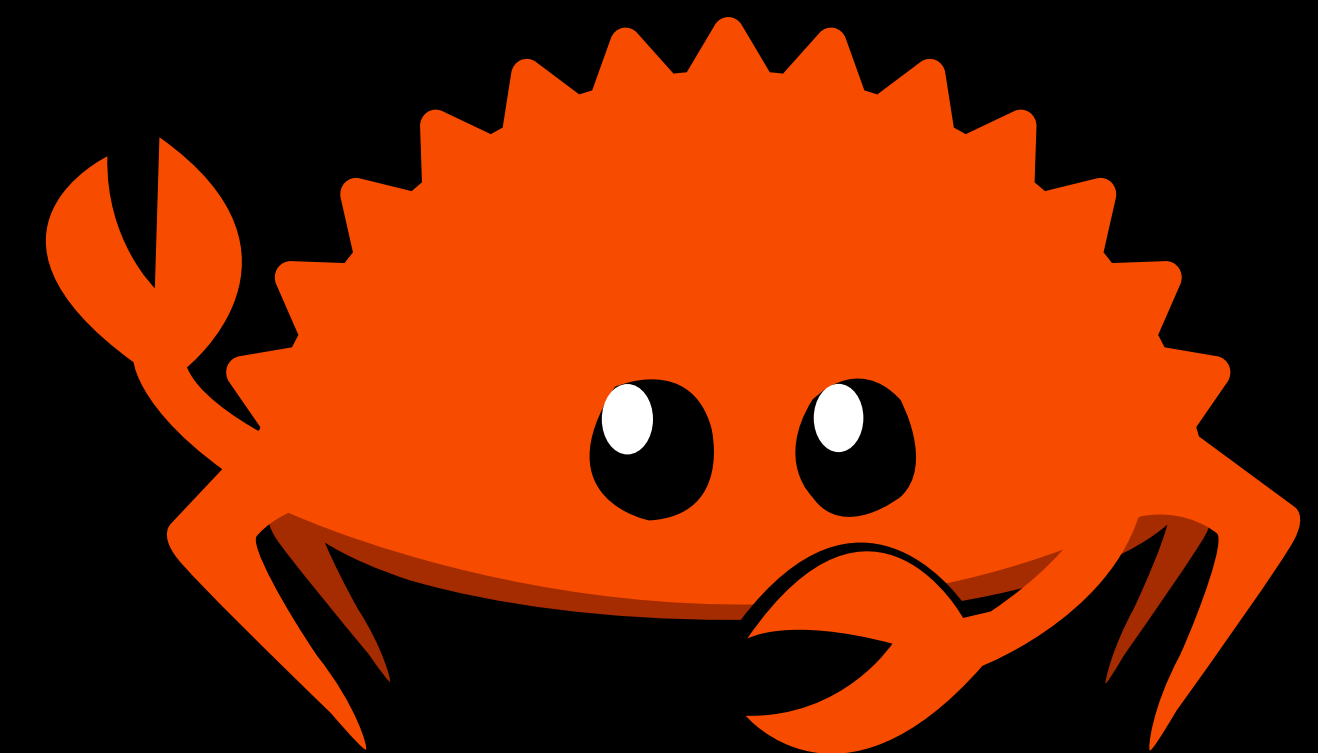

CSS

```
.customLabel {  
  color: red;  
  font-weight: bold;  
}
```

```
.formInput {  
  border: 2px solid green;  
  border-radius: 10px;  
}
```

```
button {  
  background-color: green;  
}
```

```
button:hover {  
  background-color: yellow;  
}
```



Are we web yet?

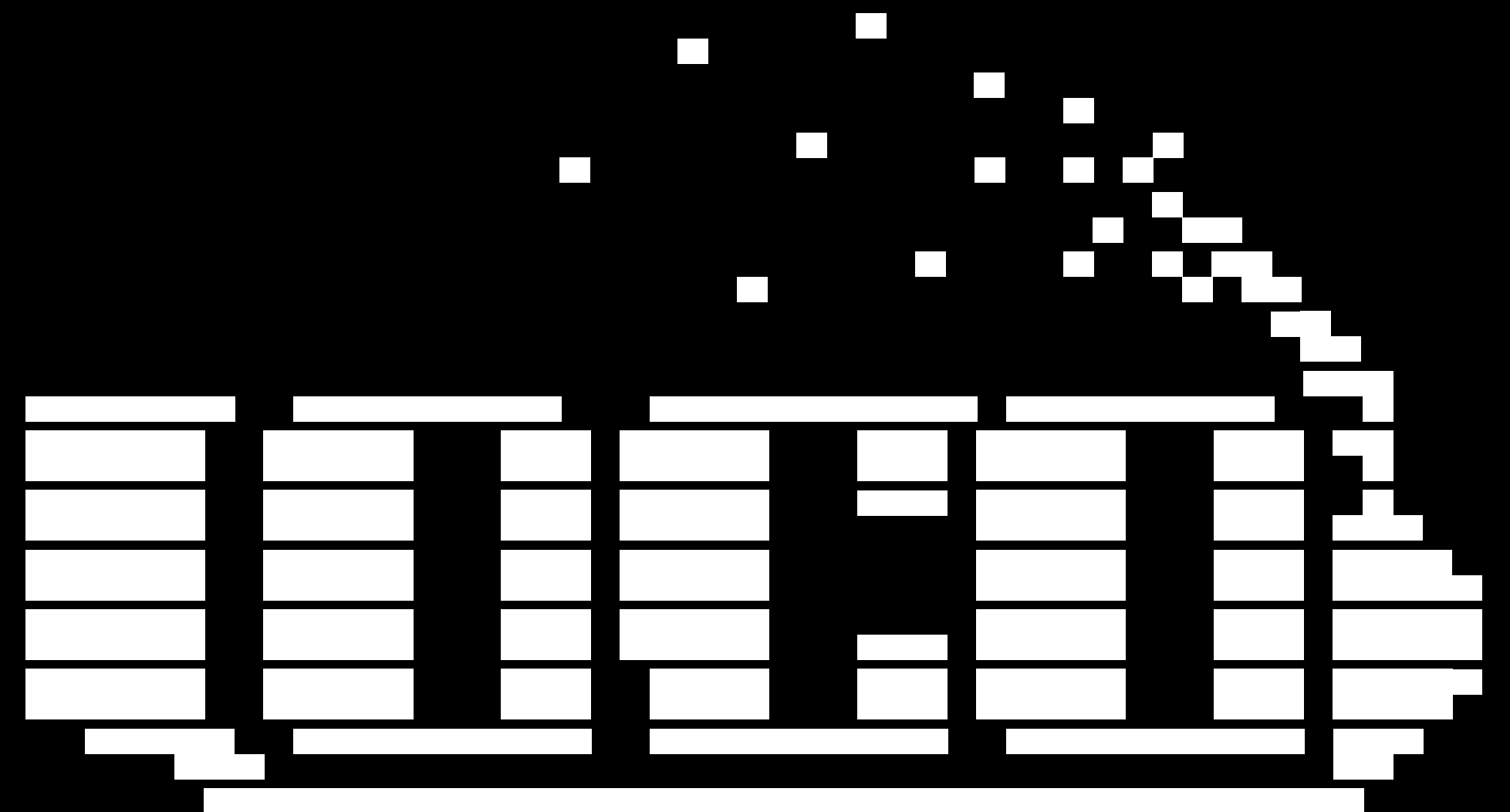
<https://www.arewewebyet.org/>

Yes! And it's freaking fast!

But there is no full fledged web framework like Rails, Django or Laravel just yet.

Except: Loco.

But Axum is more popular...



Axum

A web application framework in Rust

- No Macros needed
- Extract data from requests as needed
- Good error handling model
- Little boilerplate
- Integrates well with `tokio`, `tower` and `tower-http` crates.

Axum Hello World

```
use axum::{
    routing::get,
    Router,
};

#[tokio::main]
async fn main() {
    let app = Router::new().route("/", get(|| async { "Hello, World!" }));

    let listener = tokio::net::TcpListener::bind("0.0.0.0:3000").await.unwrap();
    axum::serve(listener, app).await.unwrap();
}
```

Axum Route Handler

HTML with Askama

```
#[derive(Template)]
#[template(path = "index.html")]
struct IndexTemplate {
    greeting: String
}

async fn index() -> impl IntoResponse {
    let template = OrganizationSelectTemplate {
        greeting: "Hello World".to_owned()
    };
    HtmlTemplate(template).into_response()
}
```

Axum Route Handler

JSON with Serde

```
pub async fn display_entries_api(
    Extension(user): Extension<User>,
    State(state): State<AppState>,
    params: Query<EntriesFilterParams>,
) -> impl IntoResponse {
    let Ok(entries) = state
        .store
        .get_entries(
            &params,
            user,
        )
        .await
    else {
        panic!("some error for entries");
    };
    let mut headers = HeaderMap::new();
    headers.insert(
        "Content-Type",
        "application/json; charset=utf-8".parse().unwrap(),
    );

    (headers, Json(entries))
}
```

Databases

SQLX

Static queries

```
pub async fn list_for_user_id(
    transaction: &mut PgConnection,
    user_id: &Uuid,
) -> Result<Vec<Organization>, sqlx::Error> {
    sqlx::query_file_as!(
        Organization,
        "sql/list/organization/organizations_for_user.sql",
        user_id
    )
    .fetch_all(transaction)
    .await
}
```

No Derives on the
Organization struct necessary

Databases

Diesel

```
async fn list_info_data_inner(
    &self,
    req: &ListInfoDataRequest,
) -> anyhow::Result<Vec<Info>> {
    let connection = self
        .pool
        .get()
        .await
        .map_err(|e| ReadInfoDataError::Unknown(e.into()))?;

    let inner_req = req.clone();
    let res = connection
        .interact(move |conn| {
            let mut query = info::table
                .select(Info::as_select())
                .into_boxed();
            if let Some(limit) = &inner_req.limit() {
                query = query.limit(*limit);
            }
            if let Some(offset) = &inner_req.offset() {
                query = query.offset(*offset);
            }
            query.get_results(conn)
        })
        .await
        .map_err(|e| ReadInfoDataError::Unknown(anyhow!(e.to_string())))?
        .map_err(|e| ReadInfoDataError::Unknown(e.into()))?;

    Ok(res)
}
```

```
#[derive(Queryable, Selectable, Debug)]
#[diesel(table_name = crate::schema::info)]
#[diesel(check_for_backend(diesel::pg::Pg))]
pub struct Info {
    pub id: Uuid,
    pub created_at: NaiveDateTime,
    pub name: String,
}
```

Building the query is nice

Databases

Sea ORM

Organization::Entity is generated code

```
async fn list_organizations(&self) -> Result<Vec<Organization>, CoreRepositoryError> {  
    let entities: Vec<Organization> = organization::Entity::find()  
        .all(&self.db)  
        .await  
        .map_err(|e| CoreRepositoryError::Database {  
            message: e.to_string(),  
        })?  
        .into_iter()  
        .map(Organization::from)  
        .collect();  
    Ok(entities)  
}
```

Cloud Docker

```
FROM rust:1.74-buster AS build
```

```
RUN USER=root cargo new myproject  
WORKDIR /usr/src/myproject
```

```
COPY Cargo.toml Cargo.lock ./  
COPY src ./src
```

```
RUN cargo install --path .
```

```
FROM debian
```

```
RUN mkdir /app  
WORKDIR /app
```

```
COPY --from=build /usr/local/cargo/bin/myproject /app/
```

```
CMD [ "./myproject" ]
```

Thanks!

Have Fun

Building web application in Rust

