



```
let args: Vec<String> = std::env::args().collect();

for arg in args {
    println!("{}", arg);
}
```



```
# Cargo.toml  
# [ ]  
  
[dependencies]  
log = "0.4.20"  
simple_logger = "4.3.0"  
  
# [ ]rust[ ]
```

```
# src/main.rs

use log::{debug, info, trace, warn};
use simple_logger::SimpleLogger;

fn main(){
    // 000000
    SimpleLogger::new().init().unwrap();
    info!("application start");
}
```

# JSON

```
# Cargo.toml

# [dependencies]

serde = { version = "1.0", features = ["derive"] }
serde_json = "1.0.68"
```

```
# src/main.rs

use log::{debug, info, trace, warn};
use serde::{Deserialize, Serialize};
use serde_json::json;
use simple_logger::SimpleLogger;

#[derive(Serialize, Deserialize)]
struct Person {
    name: String,
    age: u32,
}

async fn root() -> impl IntoResponse {
    info!("entry root");

    // 接收JSON数据
    // 解析JSON数据
    let json_str = r#"{"name": "John Doe", "age": 30}"#;

    // 解析JSON数据
    let json_obj: Person = serde_json::from_str(json_str).unwrap();

    // 生成JSON数据
    let json_str = serde_json::to_string(&json_obj).unwrap();
    log::info!("{}", json_str);

    // 生成JSON数据
    let json_obj = json!({"name": "John Doe", "age": 30});

    // 生成JSON数据
```

```
let json_str = json_obj.to_string();
log::info!("{}", json_str);

// 解析JSON字符串
let json_obj: Person = serde_json::from_value(json_obj).unwrap();
log::info!("{}", json_obj.name);

return (StatusCode::OK, Json(json_obj));
}
```

---

📄 #5

📄 26 📄 2023 04:59:38

📄 27 📄 2023 06:31:12