速查知识库

获取应用程序启动参数

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

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

打印日志

```
# Cargo.toml
# 增加依赖



[dependencies]
log = "0.4.20"
simple_logger = "4.3.0"

# 特别注意,~ust将日志接口与日志实现分开了,所以需要两个组件
```

```
# src/main.rs

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

fn main(){
    // 初始化日志组件
    SimpleLogger::new().init().unwrap();
    info!("application start");
}
```

JSON操作

```
# Cargo.toml
# 增加依赖
[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");

    // 多行文本与引号可以用 r## 处理
    // 构建一段JSON字符串
    let json_str = r#"{"name": "John Doe", "age": 30}"#;

    // 将JSON字符反序列化为对象
    let json_obj: Person = serde_json::from_str(json_str).unwrap();

    // 将对象序列化为JSON字符串
    let json_str = serde_json::to_string(&json_obj).unwrap();
    log::info!("{}", json_str);

    // 创建一个匿名的JSON对象
    let json_obj = json!({"name": "John Doe", "age": 30});

    // 将匿名JSON对象序列号为字符串
```

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

    // 将匿名JSON对象转换为对象
    let json_obj: Person = serde_json::from_value(json_obj).unwrap();
    log::info!("{}", json_obj.name);

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

版本号 #5
由 董列涛 创建于 26 十二月 2023 04:59:38
由 董列涛 更新于 27 十二月 2023 06:31:12