

## Gear

技术白皮书

https://github.com/GearFans/gear-technical/blob/cn/cn.md

### Gear 的关键技术

■ Gear 使用 Actor 通信模型

简而言之,这意味着程序(智能合约)从不共享任何 "状态",只在彼此之间交换消息。 这些消息的结果可能导致 发送其他消息、创建新的 actor 或为 actor 收到的下一个消息设置指定的行为。

■ 支持并行处理

## 开发资料 Wiki

https://wiki.gear-tech.io/zh-cn/



Gear 将通过使创新的 dApps、微服务、中间件和开放的 API 的运行,协助过渡到大规模用 Web3 技术。

GitHub □ XA 简体中文 ▼ -O-

欢迎

什么是Gear?

如何与Gear社区互动

Github

Twitter

Telegram

Medium

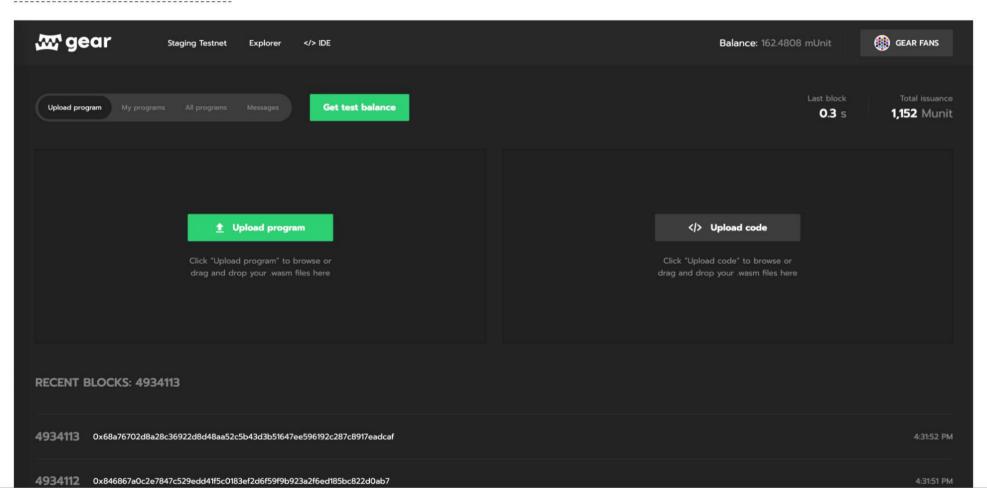
### gear-js SDK

gear-js 是 Gear 的 js SDK,通过这个工具我们可以连接节点,上传合约,发送交易,还有解析 Gear 合约等。

相关介绍:如何使用 gear-js SDK

### 在线 IDE

### https://idea.gear-tech.io/



# 合约 合约资料

Gear 合约大揭秘

Rust 初学者如何编写 Gear 智能合约(1)

### 合约基础结构

```
#![no_std]
     gstd::metadata! {
     title: "ERC20",
       init:
         input: InitConfig,
 5
       handle:
 6
        input: Action,
         output: Event,
 8
9
       state:
         input: State,
10
11
         output: StateReply,
12
13
     #[no_mangle]
14
     pub unsafe extern "C" fn handle() {}
15
16
17
     #[no_mangle]
     pub unsafe extern "C" fn init() {}
18
19
     #[no_mangle]
20
     pub unsafe extern "C" fn meta_state() -> *mut [i32; 2] {}
21
22
```

### 常用方法

#### exec

```
gstd::exec::block_timestamp() // 获取当前时间
gstd::exec::block_height() // 获取当前区块高度
gstd::exec::program_id() // 获取当前程序的id
gstd::exec::value_available // 获取当前余额

gstd::exec::exit // 终止程序的执行
```

详细资料: https://docs.gear.rs/gstd/exec/index.html

#### msg

```
gstd::msg::id // 获取消息 Id
1
    gstd::msg::source //获取消息发送者的钱包地址, 类似 solidity 中的msg.sender
    gstd::msg::load // 获取发送给合约的消息
    gstd::msg::load bytes // 获取发送给合约的消息
 6
    gstd::msg::reply // 发送一条新信息作为对当前正在处理的信息的回复
    gstd::msg::reply_bytes // 发送一条新信息作为对当前正在处理的信息的回复
 8
    gstd::msg::reply_to // 获取当前handle_reply函数被调用的初始信息的标识符
9
10
    gstd::msg::send_bytes // 向合约或者用户发现新消息
11
    gstd::msg::send // 向合约或者用户发现新消息
12
13
```

详细资料: https://docs.gear.rs/gstd/msg/index.html

### 2月相关更新

### Gear 2月更新记录

### 重要更新:

- msg::reply() 破坏性更新,参数不再传递 gas,合约会自动计算gas消耗
- gear-wasm-builder 通过 `build.rs`, 简化编译配置

# 例子-ERC20

### 安装 Rust

```
1 curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

### 添加 wasm32-unknown-unknown toolchains

- 1 rustup toolchain add nightly
- 2 rustup target add wasm32-unknown-unknown --toolchain nightly

### 安装 Polkadot.js 插件

从 https://polkadot.js.org/extension/ 下载插件

### 获取测试coin

用 `Polkadot.js 插件 创建地址,点击 Get test account 按钮,获取测试 coin

### 编译合约

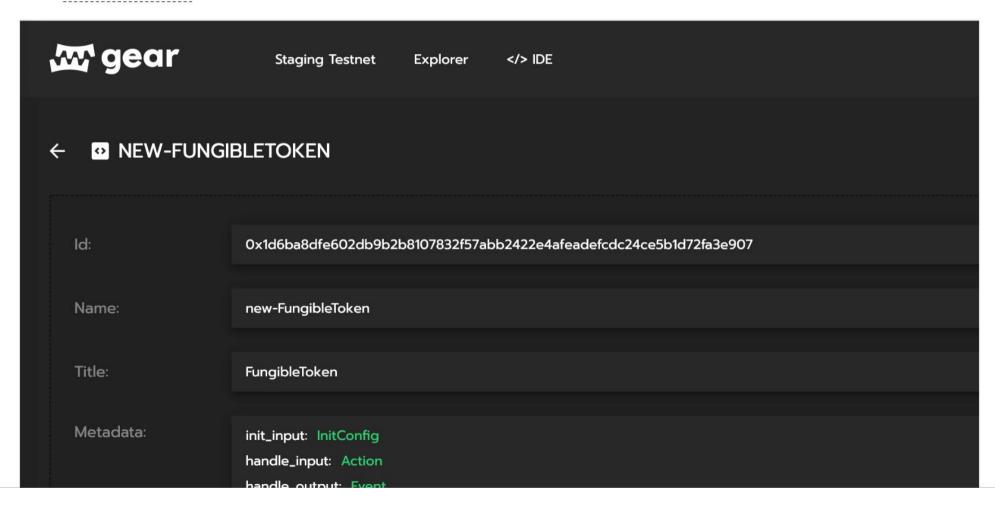
- clone 代码: `git clone https://github.com/gear-tech/apps`
- 切换路径: `cd fungible-token`
- 编译合约: `cargo build --release`
- 发现wasm合约文件,请注意文件路径: `ls ../target/wasm32-unknown-unknown/release/\*.wasm`
  - 1 -rw-r--r-- fungible\_token.meta.wasm # meta文件, 类似abi文件, 后缀 meta.wasm 结尾
  - 2 -rw-r--r-- fungible\_token.opt.wasm # 主合约文件, 后缀 opt.wasm 结尾
  - 3 -rwxr-xr-x fungible\_token.wasm # meta + opt 的"合体"文件

### 上传合约

- https://idea.gear-tech.io/,请确保 rpc 为 `wss://rpc-node.gear-tech.io:443`
- 上传 `.opt.wasm` 合约文件,和 `.meta.wasm` 文件,设置好 gas limit,可以使用默认gas limit

### 调用合约方法

调用 FungibleToken 合约



# 谢谢关注 Gear

