

The background of the slide is a full-page image of marbled paper. It features a complex, organic pattern of swirling, cell-like shapes in various shades of brown, tan, and black. The texture is dense and intricate, resembling traditional stone or shell marbling techniques.

Welcome to Rebase CodingDay

Gear 合约入门开发

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/>



[GitHub](#) [简体中文](#) [🌞](#)

欢迎

- [Gear 介绍](#) >
- [生态](#) >
- [快速入门指南](#)
- [Gear IDEA](#) >
- [Gear 节点](#) >
- [开发智能合约](#) >
- [工具](#)
- [API](#) >
- [通用概念](#) >

[👤](#) > [欢迎](#)

Gear 介绍

欢迎

欢迎来到 Gear 的文档门户。这是一个中央信息源，供每个人寻找关于 Gear 的重要信息、指南和开发人员的文档。

我们的 Wiki 概述了所需的基础信息、一般技术概述、Gear 技术的主要优势、帮助开始设置 Gear 节点、如何上传你的智能合约，同时还提供完整的 API 细节和例子。

所有的代码、库和工具都可以在 Github 上获得，并采用 Apache-2.0 许可证。欢迎使用这些工具和库，记录你发现的问题，也可以为你发现的 bug 或增加的功能创建 PR。

由于我们的项目正在发展和壮大，我们非常欢迎你的贡献。

什么是Gear?

Gear 是一个基于 Substrate 的智能合约平台，使任何人都能在几分钟内部署一个 dApp。Gear 是运行智能合约的最具成本效益的方式，这些智能合约是由许多流行的编程语言（如 Rust、C、C++ 等）编译而成。它确保了非常简约、直观和充分的 API，可以在多个网络上运行新写的和现有的程序，而不必重写。智能合约存储在区块链的状态中，并被调用，根据要求保留其状态。

Gear 正计划成为 Polkadot 和 Kusama 网络中的一个准成员，在这些各自的网络中托管智能合约。这将意味着，通过在 Gear 上的部署，开发者将能够以最小的成本利用 Polkadot 和 Kusama 的网络及[生态系统](#)的所有好处。

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。

欢迎

- [什么是Gear?](#)
- [如何与Gear社区互动](#)
- [Github](#)
- [Discord](#)
- [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/>




Staging Testnet

Explorer

</> IDE

Balance: 162.4808 mUnit

 GEAR FANS

Upload program

My programs

All programs

Messages


Get test balance

Last block


0.3 s

Total issuance

1,152 Munit



Click "Upload program" to browse or drag and drop your .wasm files here



Click "Upload code" to browse or drag and drop your .wasm files here

RECENT BLOCKS: 4934113

4934113

0x68a76702d8a28c36922d8d48aa52c5b43d3b51647ee596192c287c8917eadcaf

4:31:52 PM

4934112

0x846867a0c2e7847c529edd41f5c0183ef2d6f59f9b923a2f6ed185bc822d0ab7

4:31:51 PM

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

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

常用方法

exec

```
1  gstd::exec::block_timestamp() // 获取当前时间
2  gstd::exec::block_height() // 获取当前区块高度
3  gstd::exec::program_id() // 获取当前程序的id
4  gstd::exec::value_available // 获取当前余额
5
6  gstd::exec::exit // 终止程序的执行
7
```

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

msg

```
1  gstd::msg::id // 获取消息 Id
2  gstd::msg::source // 获取消息发送者的钱包地址, 类似 solidity 中的 msg.sender
3
4  gstd::msg::load // 获取发送给合约的消息
5  gstd::msg::load_bytes // 获取发送给合约的消息
6
7  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
8  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
9  gstd::msg::reply_to // 获取当前 handle_reply 函数被调用的初始信息的标识符
10
11 gstd::msg::send_bytes // 向合约或者用户发现新消息
12 gstd::msg::send // 向合约或者用户发现新消息
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

 Staging Testnet Explorer </> IDE

[←](#)  NEW-FUNGIBLETOKEN

Id:

0x1d6ba8dfe602db9b2b8107832f57abb2422e4afeadefcdc24ce5b1d72fa3e907

Name:

new-FungibleToken

Title:

FungibleToken

Metadata:

init_input: InitConfig

handle_input: Action

handle_output: Event

谢谢关注 Gear

Gear 是波卡的计算组件

