Substrate 区块链应用开发

Runtime 宏介绍

孙凯超
kaichao@parity.io

获取帮助: https://substrate.dev

内容

- Rust 宏
- Runtime 常用的宏
- cargo expand
- 其它宏

Rust 宏

宏（Macro）是一种元编程的方式，常见的还有Java里的反射，Rust提供了两种宏：

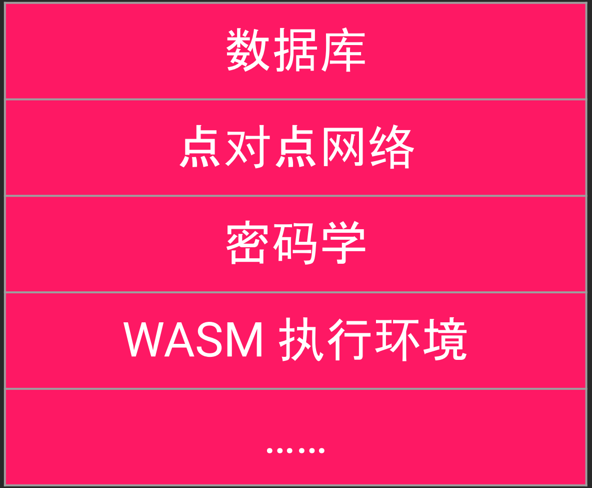
- 声明宏
- 过程宏

https://doc.rust-lang.org/book/ch19-06-macros.html

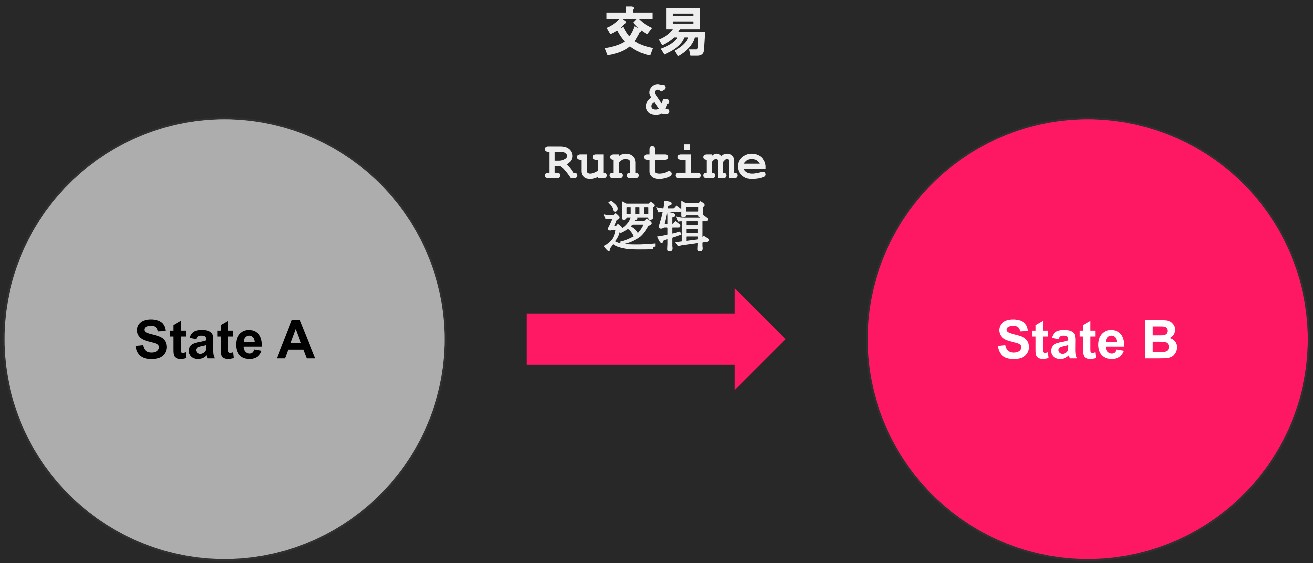
Substrate 为什么使用宏

为了简化 Runtime 的开发，Substrate 使用宏建立了一套 DSL (Domain Specific Language)，设计合理的DSL可以：

- 很好的被用户理解
- 代码更加简洁，提升效率
- 解放应用开发者，只需实现业务组件



Substrate Runtime 定义



Substrate Runtime 定义

内置的模块也称为 Pallet (调色板)

Substrate Module			
assets	babe	balances	collective
contract	democracy	elections	grandpa
indices	grandpa	indices	membership
offences	session	staking	sudo
system	timestamp	treasury	and more...

Runtime 模块的组成

使用Substratet进行 Runtime 模块开发的过程中，常用到的宏有：

- decl_storage 定义存储单元
- decl_module 包含可调用函数
- decl_event 事件
- decl_error 错误信息
- construct_runtime 添加模块到 Runtime

decl_storage

不管是 web2.0 传统的互联网应用，还是采用区块链技术的 web3.0 应用，关键数据都需要存起来。

decl_storage宏，就是用来定义 runtime 模块的存储单元。

decl_storage 例子

```
/// The pallet's configuration trait.
pub trait Trait:  system::Trait {
        /// The overarching event type.
        type Event:  From<Event<Self>>  +  Into<<Self as system::Trait>::Event>;
}

// This pallet's storage items.
decl_storage! {
        trait Store for Module<T:  Trait> as TemplateModule {
                Something get(fn something):  Option<u32>;
        }
}
```

decl_storage 例子

```
/// The pallet's configuration trait.
pub trait Trait:  system::Trait {
        /// The overarching event type.
        type Event:  From<Event<Self>>  +  Into<<Self as system::Trait>::Event>;
}

// This pallet's storage items.
decl_storage! {
        trait Store for Module<T:  Trait> as TemplateModule {
                Something get(fn something):  Option<u32>;
        }
}
```

decl_storage 例子

```
/// The pallet's configuration trait.
pub trait Trait:  system::Trait {
        /// The overarching event type.
        type Event:  From<Event<Self>>  +  Into<<Self as system::Trait>::Event>;
}

// This pallet's storage items.
decl_storage! {
        trait Store for Module<T:  Trait> as TemplateModule {
                Something get(fn something):  Option<u32>;
        }
}
```

数据类型：

- 单值
- 映射
- 双键映射

decl_module

区块链的链上状态变化由交易触发，Substrate 不仅支持自定义的存储数据结构，还支持自定义的交易，例如转账、注册身份、投票等等，也叫做 extrinsic *外部交易*。

decl_module 用来定义模块里可调用函数，每一个外部交易都会触发一个可调用函数，并根据交易体信息也就是函数参数，更新链上状态。

decl_module 例子

```
decl_module! {
        pub struct Module<T: Trait> for enum Call where origin:  T::Origin {
                type Error = Error<T>;
                fn deposit_event() = default;

                #[weight = 10_000]
                pub fn do_something(origin, something: u32) -> dispatch::DispatchResult {
                        // -- snip --
                        Something::put(something);
                        Self::deposit_event(RawEvent::SomethingStored(something, who));
                        Ok(())
                }
        }
}
```

decl_module 例子

```
decl_module! {
        pub struct Module<T: Trait> for enum Call where origin:  T::Origin {
                type Error = Error<T>;
                fn deposit_event() = default;

                #[weight = 10_000]
                pub fn do_something(origin, something: u32) -> dispatch::DispatchResult {
                        // -- snip --
                        Something::put(something);
                        Self::deposit_event(RawEvent::SomethingStored(something, who));
                        Ok(())
                }
        }
}
```

decl_module 例子

```
// -- snip --
#[weight = 10_000]
pub fn cause_error(origin) -> dispatch::DispatchResult {
        // -- snip --
        match Something::get() {
                None => Err(Error::<T>::NoneValue)?,
                Some(old) => {
                        let new = old.checked_add(1).ok_or(Error::<T>::StorageOverflow)?;
                        Something::put(new);
                        Ok(())
                },
        }
}
```

parity 一块+ OneBlock+

decl_module

Runtime 模块里存在保留函数，除了 deposit_event 之外，还有：

- on_initialize，在每个区块的开头执行；
- on_finalize，在每个区块结束时执行；
- offchain_worker：开头且是链外执行，不占用链上的资源；
- on_runtime_upgrade：当有 runtime 升级时才会执行，用来迁移数据。

decl_event

区块链是一个异步系统，runtime 通过触发事件通知交易执行结果。

```
decl_event!(
	pub enum Event<T> where AccountId = <T as system::Trait>::AccountId {
		SomethingStored(u32, AccountId),
	}
);
```

```
// -- snip --
Self::deposit_event(RawEvent::SomethingStored(something, who));
```

decl_error

```
decl_error! {
	pub enum Error for Module<T: Trait> {
		/// Value was None
		NoneValue,
		/// Value reached maximum and cannot be incremented further
		StorageOverflow,
	}
}
```

decl_error

可调用函数里的错误类型，

- 不能给它们添加数据；
- 通过 metadata 暴露给客户端；
- 错误发生时触发system.ExtrinsicFailed 事件，包含了对应错误的信息。

construct_runtime 加载模块

```
impl template::Trait for Runtime {
    type Event = Event;
}

construct_runtime!(
        pub enum Runtime where
                Block = Block,
                NodeBlock = opaque::Block,
                UncheckedExtrinsic = UncheckedExtrinsic
        {
                // -- snip --
                TemplateModule:  template::{Module, Call, Storage, Event<T>},
        }
);
```

cargo expand

将宏里的代码展开，得到 Rust 的标准语法。

https://github.com/dtolnay/cargo-expand

https://github.com/kaichaosun/play-substrate/blob/master/pallets/template/expanded.rs

其它宏

decl_runtime_apis & imp_runtime_apis，定义runtime api：
https://substrate.dev/recipes/3-entrees/runtime-api.html

https://substrate.dev/rustdocs/master/sp_api/macro.decl_runtime_apis.html

https://substrate.dev/rustdocs/master/sp_api/macro.impl_runtime_apis.html

runtime_interface，定义在 runtime 里可以调用的 Host 提供的函数：
https://substrate.dev/rustdocs/v2.0.0-alpha.8/sp_runtime_interface/attr.runtime_interface.html

多实例模块

Substrate 的模块在 runtime 里可以有多个实例，
例如，可以添加多个内置的 collective 模块实例，
分别用来表示理事会和技术委员会，来实现复杂的
治理模型。

多实例模块 - 例子

```
pub trait Trait<I: Instance = DefaultInstance>:  frame_system::Trait {
	type Event: From<Event<Self, I>> + Into<<Self as
frame_system::Trait>::Event>;
}

decl_storage! {
	trait Store for Module<T: Trait<I>, I: Instance=DefaultInstance> as Collective {
		// -- snip --
	}
}
```

多实例模块 - 例子

```
decl_module! {
   pub struct Module<T: Trait<I>, I: Instance = DefaultInstance> for enum Call
where origin: T::Origin {
      // -- snip --
}
```

parity 一块+ OneBlock+

多实例模块 - 例子

```
decl_event!(
	pub enum Event<T, I: Instance = DefaultInstance> where
		<T as frame_system::Trait>::AccountId,
	{
		// -- snip --
	}
);

decl_error! {
	pub enum Error for Module<T: Trait<I>, I: Instance> {
		// -- snip --
	}
}
```

多实例模块 - 例子

```
type CouncilCollective = pallet_collective::Instance1;
impl pallet_collective::Trait<CouncilCollective> for Runtime {
        // -- snip --
}

construct_runtime!(
        pub enum Runtime where
                Block = Block,
                NodeBlock = node_primitives::Block,
                UncheckedExtrinsic = UncheckedExtrinsic
        {
                Council:  pallet_collective::<Instance1>::{Module, Call, Storage, Origin<T>, Event<T>,
Config<T>},
    }
);
```

Questions?

官网文档：substrate.io
知乎专栏：parity.link/zhihu

kaichao@parity.io

https://parity.link/asia-support