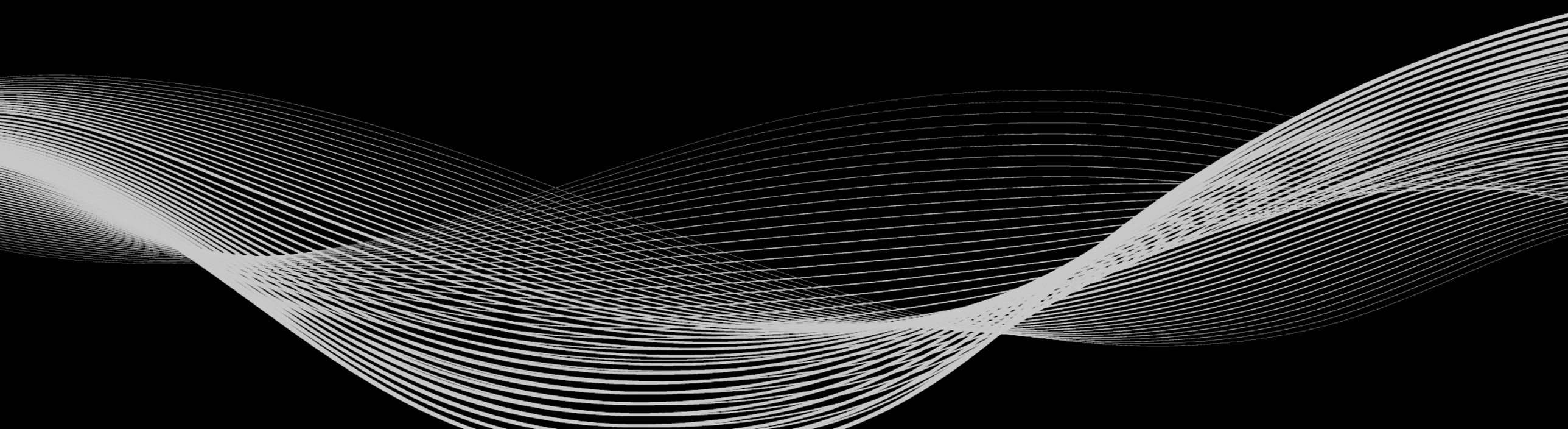
Substrate 快速入门与实战

讲师：Bryan



一块链习

第六节课

模块实现

上节作业

- Dispatchable methods
  - fn ask(origin, kitty_id: KittyIndex, price: Option<Balance>)
  - fn buy(origin, kitty_id: KittyIndex, price: Balance)

上节作业

- Storages
  - KittyPrices: map KittyIndex => Option<Balance>
  - KittyOwners: map KittyIndex => Option<AccountId>

Dependency Injection
依赖注入

依赖注入

- 保证模块的抽象化和重用性
- 去除模块之间不必要的耦合
- 依赖于 Trait 而不是 Module
- 更加方便的测试

依赖注入

```rust
/// Abstraction over a fungible assets system.
pub trait Currency<AccountId> {
    /// The balance of an account.
    type Balance: SimpleArithmetic + As<usize> + As<u64> + Codec + Copy + MaybeSerializeDebug + Default;

    /// The opaque token type for an imbalance. This is returned by unbalanced operations
    /// and must be dealt with. It may be dropped but cannot be cloned.
    type PositiveImbalance: Imbalance<Self::Balance, Opposite=Self::NegativeImbalance>;

    /// The opaque token type for an imbalance. This is returned by unbalanced operations
    /// and must be dealt with. It may be dropped but cannot be cloned.
    type NegativeImbalance: Imbalance<Self::Balance, Opposite=Self::PositiveImbalance>;

    // PUBLIC IMMUTABLES

    /// The combined balance of `who`.
    fn total_balance(who: &AccountId) -> Self::Balance;

    /// Same result as `slash(who, value)` (but without the side-effects) assuming there are no
    /// balance changes in the meantime and only the reserved balance is not taken into account.
    fn can_slash(who: &AccountId, value: Self::Balance) -> bool;

    /// The total amount of issuance in the system.
    fn total_issuance() -> Self::Balance;

    /// The minimum balance any single account may have. This is equivalent to the `Balances` module's
    /// `ExistentialDeposit`.
    fn minimum_balance() -> Self::Balance;
```

依赖注入

```
pub type BalanceOf<T> = <<T as Trait>::Currency as Currency<<T as system::Trait>::AccountId>>::Balance;
pub type NegativeImbalanceOf<T> = <<T as Trait>::Currency as Currency<<T as
system::Trait>::AccountId>>::NegativeImbalance;

pub trait Trait: timestamp::Trait {
    type Currency: Currency<Self::AccountId>;

    /// The outer call dispatch type.
    type Call: Parameter + Dispatchable<Origin=<Self as system::Trait>::Origin>;

    /// The overarching event type.
    type Event: From<Event<Self>> + Into<<Self as system::Trait>::Event>;

    // `As<u32>` is needed for wasm-utils
    type Gas: Parameter + Default + Codec + SimpleArithmetic + Bounded + Copy + As<BalanceOf<Self>> + As
    As<u32>;
```

依赖注入

```
impl contract::Trait for Runtime {
    type Currency = Balances;
    type Call = Call;
    type Event = Event;
    type Gas = u64;
    type DetermineContractAddress = contract::SimpleAddressDeterminator<Runtime>;
    type ComputeDispatchFee = contract::DefaultDispatchFeeComputor<Runtime>;
    type TrieIdGenerator = contract::TrieIdFromParentCounter<Runtime>;
    type GasPayment = ();
}
```

Events 事件

Events 事件

- decl_event
- 区块包含的交易不表示交易是成功的
- 需要监听event事件来确认真正的发生了什么
- 监听storage也能达到类似的效果
- 事件可以提供更多的storage无法提供的消息

Events 事件

- Created 创建小猫
- Transferred 转让小猫
- Ask 要价
- Sold 卖出

重构链表结构

srml-generic-asset
代码分析

作业

作业

- 完成linked_item
- 修复测试

额外作业

- 利用 polkadot.js 开发一个node.js服务器
- 监听并存储所有 kitties 事件到数据库
- 设计并讨论如何使用事件和链下数据库来简化链上存储

一块链习

THANK YOU!

Contact us:
info@yikuailianxi.com