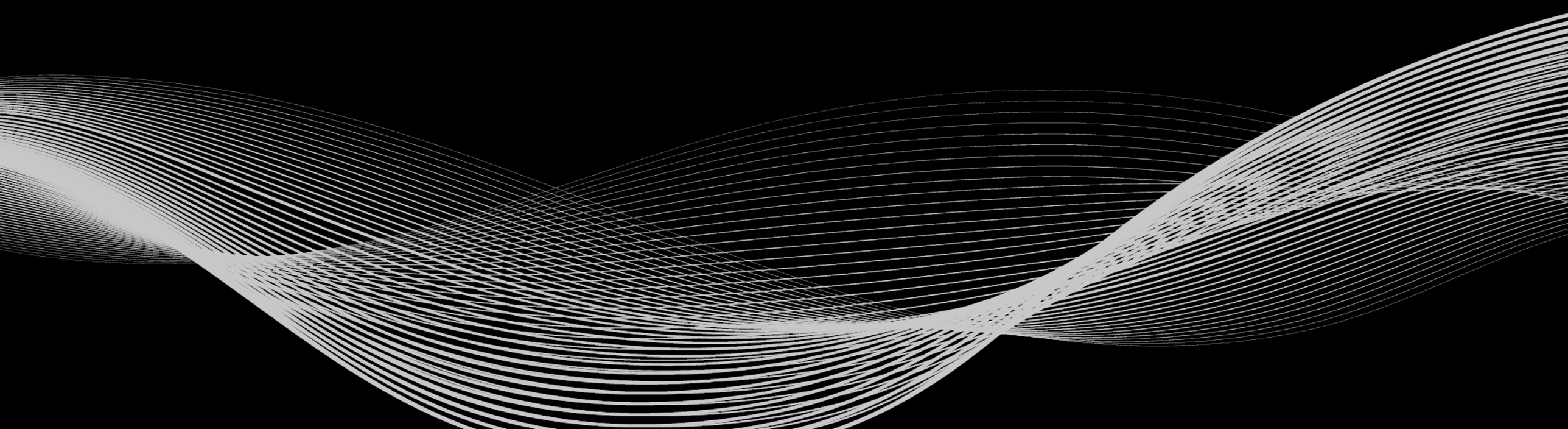
Substrate 快速入门与实战

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 + FullCodec + Copy + MaybeSerializeDeserialize + Debug + 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>;
```

```rust
pub trait Time {
    type Moment: SimpleArithmetic + Parameter + Default + Copy;

    fn now() -> Self::Moment;
}
```

```rust
// A trait that is able to provide randomness.
pub trait Randomness<Output> {
    /// Get a "random" value
    ///
    /// Being a deterministic blockchain, real randomness is difficult to come by. This gives you
    /// something that approximates it. `subject` is a context identifier and allows you to get a
    /// different result to other callers of this function; use it like
    /// `random(&b"my context"[..])`.
    fn random(subject: &[u8]) -> Output;
```

依赖注入

```rust
pub trait Trait: frame_system::Trait {
    type Currency: Currency<Self::AccountId>;
    type Time: Time;
    type Randomness: Randomness<Self::Hash>;
```

```rust
impl pallet_contracts::Trait for Runtime {
    type Currency = Balances;
    type Time = Timestamp;
    type Randomness = RandomnessCollectiveFlip;
```

依赖注入

```
fn random(&self, subject: &[u8]) -> SeedOf<T> {
    T::Randomness::random(subject)
}
```

```
T::Currency::ensure_can_withdraw(
    transactor,
    value,
    WithdrawReason::Transfer.into(),
    new_from_balance,
)?;
```

```
pub fn top_level(origin: T::AccountId, cfg: &'a Config<T>, vm: &'a V, loader: &'a L) -> Self {
    ExecutionContext {
        parent: None,
        self_trie_id: None,
        self_account: origin,
        overlay: OverlayAccountDb::<T>::new(&DirectAccountDb),
        depth: 0,
        deferred: Vec::new(),
        config: &cfg,
        vm: &vm,
        loader: &loader,
        timestamp: T::Time::now(),
        block_number: <frame_system::Module<T>>::block_number(),
    }
}
```

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

Errors 错误信息

Errors 错误信息

- decl_error
- 提供有限的错误信息查询
- 在更多的信息和更小的链上空间直接的平衡
- 更多的错误信息可以通过replay block来得到
- 未来会有类似Etherscan的服务提供更多的信息索引

Errors 错误信息

- 相关的设计讨论
  - https://github.com/paritytech/substrate/issues/2246
  - https://github.com/paritytech/substrate/pull/2880
  - https://github.com/paritytech/substrate/pull/3433
  - https://github.com/paritytech/substrate/issues/4384
  - https://github.com/paritytech/substrate/issues/4040
  - https://github.com/paritytech/substrate/pull/4449

重构链表结构

一块链习

Open Runtime Module Library

ORML

- 由社区维护的通用Substrate模块
- 提供各种常用功能的实现
  - tokens
  - oracle
  - auction
- 保持与最新版本的Substrate兼容

ORML

- Github: https://github.com/laminar-protocol/open-runtime-module-library
- 使用的案例：
  - https://github.com/AcalaNetwork/Acala
  - https://github.com/laminar-protocol/flowchain

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