Substrate

River

Kitty Auction-003

Bidding an auction

https://substrate.dev/substrate-collectables-workshop/#/Extras/Auction/bidding-an-auction

Bidding an auction

从代币的角度

1- 竞拍者出价

2- 系统托管竞拍者出价的代币

3- 在结束时，竞拍成功者；从托管中转移代币到Kitty的所有者

4- 在结束时，未竞拍成功者；从托管中恢复自己的出价代币

bid_auction()

验证
1- 这只猫正在竞拍，而且还没有过期
2- 出价高于现有的最高出价
3- 竞拍者有足够的自由余额

在这些验证之后，我们知道新的拍卖的出价是最高的，所以我们将更新拍卖对象的high_bid和high_bidder值与新的值。

Using reserve balance for escrow:托管

我们将把竞拍者的出价添加到出价列表中，并从他们的帐户中**代管**这笔金额，这样如果用户赢得了拍卖，就可以保证用户付款。

为了达到这个目的，我们可以使用balance模块的reserve函数，它设置了一个余额，这个余额仍然是帐户持有人“拥有”的，不能被转移，但是可以被子系统使用，比如我们的拍卖逻辑。

```
<balances::Module<T>>::reserve(sender, amount)?;
```

在下一个步骤中，在完成函数时，将使用balance模块的unreserve函数释放保留的余额。

```
 <balances::Module<T>>::unreserve(sender, amount)?;
```

storage

1- Bids 存储同一只小猫，不同竞拍者的出价

map (T::Hash,T::AccountId) => T::Balance;

2- BidAccounts 存储同一只小猫的所有竞拍者

map T::Hash => Vec<T::AccountId>;

备注：

KittyId和拍卖的关系
KittyAuction get(auction_of): map T::Hash => Option<Auction<T::Hash, T::Balance, T::BlockNumber, T::AccountId>>;

截止时间和所有拍卖的关系
Auctions get(auctions_expire_at): map T::BlockNumber => Vec<(Auction<T::Hash, T::Balance, T::BlockNumber, T::AccountId>

AuctionPeriodLimit get(auction_period_limit) config(): T::BlockNumber = T::BlockNumber::sa(17280);

同一个kitty，不同的AccounId的出价
Bids get(bid_of): map (T::Hash,T::AccountId) => T::Balance;

同一个kitty对应的所有竞拍者
BidAccounts get(bid_accounts): map T::Hash => Vec<T::AccountId>;

Substrate

River

Thanks