NFT_Market contract

```
    struct SellOrder {
        address seller; // 卖家地址
        address contractID; // NFT 合约地址
        uint256 tokenID; // NFT 的 tokenId
        uint256 price; // 卖单的价格
        bytes signature; // 签名
        uint256 expirationTime; // 时间戳
        uint256 nonce; // nonce每次提交订单nonce需要不重复
    }
```

CreateProxyWallet

```
function createProxyWallet() public
```

- 使用`msg.sender`作为owner为用户创建`ProxyWallet`合约作为该用户的代理钱包
- 每一个用户只有`ProxyWallet`，不重复创建

getProxyWallet

```
function getProxyWallet(address user) public returns (address)
```

params	explanation
user	需要查询的用户地址

- 获得对应的用户的代理钱包地址
- 如果没有创建代理钱包就直接创建一个

MatchOrder

```
function matchOrder(
        address seller, // 卖家地址
        address contractID, // NFT 合约地址
        uint256 tokenID, // NFT 的 tokenId
        uint256 price, // 卖单的价格
        bytes memory signature, // 签名
        uint256 expirationTime, // 时间戳
        uint256 nonce // nonce
    ) public payable {
```

params	explanation
sellOrder	卖方订单参数

- `_value=msg.value`
- 校验卖家合法
  - token持有者
  - 存在代理钱包
- 检查订单是否超时
  - 超时：进行撤销，并且触发`OrderExpired`
- 检查订单没有被使用
- 检查`_value - fee >= sellOrder.price`，调用`Atomictx`转移NFT
  - 转移成功
    - 支付钱给卖家，支付费率
    - 触发匹配成功事件
  - 转移失败
    - 退钱给买家
    - 触发匹配失败事件

CancelSellOrder

```
function CancelSellOrder(
        address seller, // 卖家地址
        address contractID, // NFT 合约地址
        uint256 tokenID, // NFT 的 tokenId
        uint256 price, // 卖单的价格
        bytes memory signature, // 签名
        uint256 expirationTime, // 时间戳
        uint256 nonce // nonce
    ) public onlySeller(seller)
```

params	explanation
sellerorder	需要撤销的订单

- 判定调用者是不是卖家（仅有卖家可以主动删除）
- `call`调用`ProxyWallet`的`canceSellOrder`修改订单hash的map，确保不会被重复使用
- 触发订单删除事件`event SellOrderCancelled`

setMall

```
function setMall(address _mall) public{
```

params	explanation
_mall	设定将手续费支付给该地址

- 修改后在交易中会将手续费支付给该地址
- 仅拥有者

getFeeRate

```
function getFeeRate() public view returns (uint256) {
```

- 返回当前市场的费率

updateFeeRate

```
function updateFeeRate(uint256 newFeeRate) public  {
```

params	explanation
newFeeRate	设定新的费率

- 修改当前市场新的费率
- 只允许合约`owner`操作
- 触发费率修改事件`emit UpdateFeeRate(oldFeeRate, newFeeRate);`

ProxyWallet contract

AtomicTx

```
function AtomicTx(
        SellOrder memory sellOrder,
        address buyer
    ) public payable {
```

params	explanation
sellOrder	卖家订单
buyer	买家地址

- 校验`sellOrder`的签名
- 检查sellOrder是否有效
- 匹配成功转移`nft`给买家，并标记订单已被使用（添加map）

CancelOrder

```
function cancelOrder(SellOrder memory sellOrder) public {
```

params	explanation
sellOrder	撤销的订单

- 校验签名
- 添加map，记为true

isOrderInvalid

```
function isOrderInvalid(bytes32 orderHash) public view returns (bool) {
```

params	explanation
orderhash	orderid

- `bool`为`true`表示已被使用或撤销

events

```
event TradeSuccess(
        address indexed seller,
        address indexed buyer,
        address indexed contractID,
        bytes32 orderID,
        uint256 tokenID,
        uint256 price
    );
```

```
 event TradeFail(
        address indexed seller,
        address indexed buyer,
        address indexed contractID,
        bytes32 orderID,
        uint256 tokenID,
        uint256 price
    );
```

```
event SellOrderCancelled(
        address indexed seller,
        address indexed contractID,
        bytes32 orderID,
        uint256 tokenID
    );
```

```
event MatchSuccess(
        address indexed seller,
        address indexed buyer,
        address indexed contractID,
        uint256 tokenID,
        bytes32 orderID,
        uint256 price
    );
```

```
 event MatchFail(
        address indexed seller,
        address indexed buyer,
        address indexed contractID,
        uint256 tokenID,
        bytes32 orderID,
        uint256 price
    );
```

```
//事件：更新费率
    event UpdateFeeRate(uint256 oldFeeRate, uint256 newFeeRate);
```

```
//订单超时
    event OrderExpired(
        address indexed seller,
        address indexed contractID,
        bytes32 orderID,
        uint256 tokenID
    );
```

OrderID Sig

```
orderID = keccak256(
            abi.encodePacked(
                sellOrder.seller,
                sellOrder.contractID,
                sellOrder.tokenID,
                sellOrder.price,
                sellOrder.expirationTime,
                sellOrder.nonce
            )
        );
```

- `signature`:对`orderid`进行签名

```
//即对此进行签名
bytes32 hash = keccak256(
            abi.encodePacked(
                seller,
                contractID,
                tokenID,
                price,
                expirationTime,
                nonce
            )
        );
```