Solidity智能合约开发

第10.1课：NFT市场

进阶项目 | 实战开发

pragma solidity ^0.8.20;

什么是NFT市场?

What is an NFT Marketplace?

定义

NFT市场是一个去中心化的数字资产交易平台，用户可以在这里铸造、展示、买卖和拍卖NFT。

核心特点

去中心化交易

买卖双方直接通过智能合约交易

透明性

所有交易记录都在区块链上公开可查

安全性

资产所有权由智能合约保障

可编程性

可以实现复杂的交易逻辑

知名平台

- OpenSea
- Rarible
- LooksRare

交易流程

本节课学习内容

Today's Learning Objectives

ERC721标准

理解NFT的技术标准

- 核心接口函数
- 元数据扩展

NFT铸造

实现NFT的创建功能

- 供应量限制
- 价格设置

上架/下架

管理NFT的展示状态

- 挂单数据结构
- 改价功能

买卖功能

实现固定价格交易

- 完整购买流程
- 资金分配

版税系统

为创作者提供持续收益

- ERC2981标准
- 自动版税分配

拍卖功能

实现竞价交易机制

- 英式拍卖机制
- 出价管理

ERC721标准介绍

ERC721 Standard Introduction

ERC20 vs ERC721 标准对比

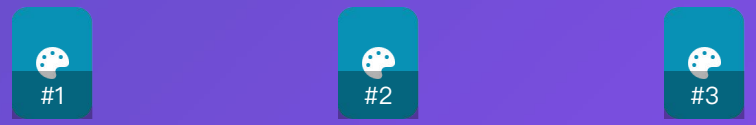
特性	ERC20 (同质化)	ERC721 (非同质化)
唯一性	完全相同	每个都独一无二
Token ID	完全相同	每个有唯一ID
可互换	是	否

ERC20 – 同质化代币

你的100元 = 我的100元

VS

ERC721 – 非同质化代币



CryptoPunk #1 ≠ CryptoPunk #2

简单理解

ERC20像人民币：你的100元 = 我的100元

ERC721像艺术品：每个都是独一无二的收藏品

ERC721核心接口与元数据扩展

ERC721 Core Interface & Metadata Extension

ERC721核心接口

```
// 查询某个地址拥有的NFT数量
function balanceOf(address owner)
    external view returns (uint256)

// 查询某个NFT的所有者
function ownerOf(uint256 tokenId)
    external view returns (address)

// 安全转移NFT
function safeTransferFrom(
    address from,
    address to,
    uint256 tokenId

// 批准其他地址操作你的NFT
function approve(address to, uint256 tokenId)
    external

// 批准操作者操作你的所有NFT
function setApprovalForAll(
    address operator,
    bool approved
```

ERC721元数据扩展

```
// NFT集合名称
function name()
    external view returns (string memory)

// NFT集合符号
function symbol()
    external view returns (string memory)

// 返回tokenId的元数据URI
function tokenURI(uint256 tokenId)
    external view returns (string memory)
```

tokenURI工作流程

tokenURI() 函数
返回元数据URI字符串

元数据URL
指向知识库文件 "https://api.example.com/token/1"

NFT合约设计与铸造功能

NFT Contract Design & Minting Functions

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol"; // 导入Counters库

contract MyNFT is ERC721, ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;

    // tokenIds 变量应为 private
    Counters.Counter private _tokenIds; // 修正变量名并改为 private

    // 构造函数：初始化ERC721合约的名称和符号，并设置合约所有者
    constructor() ERC721("MyNFT", "MNFT") Ownable(msg.sender) {}

    // mint函数：铸造一个新的NFT
    // to: 接收NFT的地址
    // uri: NFT的元数据URI
    function mint(address to, string memory uri)
        public
        returns (uint256)
    {
        _tokenIds.increment(); // 递增token计数器
        uint256 newItemId = _tokenIds.current(); // 获取当前新的token ID

        _safeMint(to, newItemId); // 安全地铸造NFT
        _setTokenURI(newItemId, uri); // 设置NFT的元数据URI

        return newItemId; // 返回新的token ID
    }

    // tokenURI函数：获取指定token ID的元数据URI
    function tokenURI(uint256 tokenId)
        public
```

铸造流程

1. 增加计数器 ↓
2. 创建NFT ↓
3. 设置元数据并返回

⭐ 优化功能

供应量限制

设置最大供应量，控制NFT总量

铸造价格

设置每个NFT的铸造价格

市场合约结构与上架功能

Marketplace Contract Structure & Listing Functions

Listing 数据结构

```
struct Listing {
    address seller;
    address nftContract;
    uint256 tokenId;
    uint256 price;
    bool active;
}
```

数据映射

```
mapping(uint256 => Listing) public listings;
uint256 public listingCounter;
```

上架功能实现

```
event NFTListed( // 事件定义
    uint256 indexed listingId, address indexed seller, address indexed nftContract, uint256
    tokenId, uint256 price );
function listNFT( // 函数签名
    address nftContract, uint256 tokenId, uint256 price ) externalreturns (uint256) {


    // 参数验证
    require(price > 0, "Price > 0");
    // 检查NFT所有权和授权
    IERC721 nft = IERC721(nftContract);
    require(nft.ownerOf(tokenId) == msg.sender, "Not owner");
    require(
    nft.getApproved(tokenId) == address(this) ||
    nft.isApprovedForAll(msg.sender, address(this)),
    "Not approved"
    );
    // 创建新挂单
    listingCounter++;
    listings[listingCounter] = Listing({
    seller: msg.sender,
    nftContract: nftContract,
```

购买功能设计与实现

Buy NFT Function Design & Implementation

核心代码实现

```
function buyNFT(uint256 listingId)
external payable nonReentrant returns (uint256) {

// ① 验证检查
require(listings[listingId].active, "Listing not active");
require(msg.value >= listings[listingId].price, "Insufficient payment");
require(msg.sender != listings[listingId].seller, "Cannot buy your own NFT");

// ② 标记为已售出
listings[listingId].active = false;

// ③ 计算手续费
uint256 fee = (listings[listingId].price * platformFee) / 10000;
uint256 sellerAmount = listings[listingId].price - fee;

// ④ 转移NFT
IERC721(listings[listingId].nftContract).safeTransferFrom(
listings[listingId].seller,
msg.sender,
listings[listingId].tokenId
);

// ⑤ 转账给卖家
(bool successSeller, ) = listings[listingId].seller.call{value: sellerAmount}("");
require(successSeller, "Transfer to seller failed");

// ⑥ 转账手续费
(bool successFee, ) = feeRecipient.call{value: fee}("");
require(successFee, "Transfer fee failed");
```

购买流程图解

1. **验证检查**
   检查挂单状态、支付金额、买家≠卖家
2. **状态更新**
   active = false，防止重复购买
3. **手续费计算**
   手续费 = price × 2.5%
4. **NFT转移**
   从卖家 → 买家，使用safeTransferFrom
5. **资金分配**
   卖家收款、平台手续费、退还多余ETH

版税系统设计与实现

Royalty System Design and Implementation

ERC2981标准

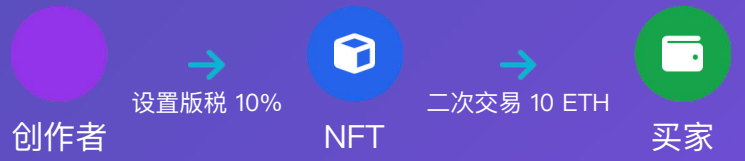
ERC2981是NFT版税标准，允许创作者在NFT转售时获得持续收益。

```
function royaltyInfo(uint256 tokenId, uint256 salePrice)
external view returns (address receiver, uint256 royaltyAmount);
```

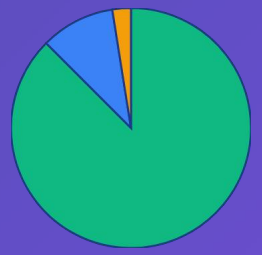
合约实现

```
function _checkRoyalties(address nftContract) internal viewreturns (bool) {
(bool success) = IERC165(nftContract).supportsInterface(type(IERC2981).interfaceId);
return success;
}
function _getRoyaltyInfo(
address nftContract,
uint256 tokenId,
uint256 salePrice
) internal view returns (address receiver, uint256 royaltyAmount) {
if (_checkRoyalties(nftContract)) {
return IERC2981(nftContract).royaltyInfo(tokenId, salePrice);
}
return (address(0), 0);
}
```

版税工作流程



资金分配



- **卖家收益:** 87.5 ETH
- **版税:** 1 ETH
- **平台手续费:** 0.25 ETH

拍卖系统设计与实现

Auction System Design & Implementation

拍卖数据结构

```
struct Auction {
    address seller; // 卖家地址
    address nftContract; // NFT合约地址
    uint256 tokenId; // NFT的tokenId
    uint256 startPrice; // 起拍价
    uint256 highestBid; // 当前最高出价
    address highestBidder; // 当前最高出价者

    uint256 endTime; // 结束时间
    bool active; // 是否激活
}
```

出价规则

最低加价：5%　　minBid = highestBid × 1.05

创建拍卖

```
function createAuction(
address nftContract,
uint256 tokenId,
uint256 startPrice,
uint256 duration
) external returns (uint256) {
require(startPrice > 0, "起拍价必须大于0");
require(duration >= 1 hours, "时间太短");
auctionCounter++;
auctions[auctionCounter] = Auction({
seller: msg.sender,
nftContract: nftContract,
tokenId: tokenId,
startPrice: startPrice,
highestBid: 0,
highestBidder: address(0),
endTime: block.timestamp + duration,
active: true
});
emit AuctionCreated(
auctionCounter, msg.sender, nftContract,
tokenId, startPrice, block.timestamp + duration
);
return auctionCounter;
}
```

结束拍卖

```
function endAuction(uint256 auctionId)
external nonReentrant {
Auction storage auction = auctions[auctionId];
require(auction.active, "拍卖未激活");
require(block.timestamp >= auction.endTime,
"拍卖未结束");
auction.active = false;
if (auction.highestBidder != address(0)) {
uint256 platformFeeAmount =
(auction.highestBid * platformFee) / 10000;
(address royaltyReceiver, uint256 royaltyAmount) =
_getRoyaltyInfo(
auction.nftContract,
auction.tokenId,
auction.highestBid
);
uint256 sellerAmount = auction.highestBid
- platformFeeAmount - royaltyAmount;
IERC721(auction.nftContract).safeTransferFrom(
auction.seller,
auction.highestBidder,
auction.tokenId
);
if (royaltyAmount > 0) {
(bool successRoyalty, ) = royaltyReceiver.call{value:
royaltyAmount}("");
require(successRoyalty, "版税转账失败");
}
```

功能回顾与扩展建议

Course Summary & Extensions

课程核心内容

ERC721标准
NFT技术标准与接口实现

NFT铸造
实现NFT的创建与发行

上架/下架
管理NFT的展示状态

买卖功能
实现固定价格交易

版税系统
为创作者提供持续收益

拍卖功能
实现竞价交易机制

下节课预告

Hardhat开发框架

我们将进入第三阶段，学习专业的开发工具，包括本地测试网络、自动化测试、合约部署与验证等。

功能扩展建议

要约系统 (Offer)
买家可以对任意NFT出价，卖家可以接受或拒绝要约

批量操作
批量上架多个NFT，批量购买节省Gas

私密销售
设置白名单买家，指定特定地址购买

实战建议

- 在测试网部署 (Goerli/Sepolia)
- 完整交互测试 (铸造→上架→购买)
- 观察OpenSea展示效果
- Gas优化实践与分析

学习资源

- OpenZeppelin Docs
- OpenSea Docs
- OpenSource项目 (OpenSea Seaport等)
- 进阶学习: 合约升级、跨链NFT