Homework4: Programming an ERC721 NFT contract (Using Hardhat/OpenZeppelin)

110321006 蔡秉翰、110321007 劉竑毅

摘要：

本報告對一個使用 Solidity 編程語言和 OpenZeppelin 庫實現的 ERC721 非同質化代幣 NFT 智能合約 MyNFT 的結構與功能進行了詳細介紹。

合約功能概述：

1. 引入外部合約（Imports）：

ERC721：OpenZeppelin 提供的標準 ERC721 代幣實現。

ERC721URIStorage：一個 OpenZeppelin 合約，為每個 NFT 提供 URI 存儲功能。

Counters：幫助合約安全計數，常用於追踪代幣 ID。

2. 合約構造（Constructor）：

constructor()：定義了 NFT 的名稱（"MyNFT"）和符號（"MNFT"）。這是部署合約時自動執行的初始化函數。

3. 代幣鑄造功能（Minting Function）：

mintNFT(address recipient, string memory tokenURI)：允許用戶為指定的接收者地址鑄造一個新的 NFT。該函數使用 _tokenIds 來給新 NFT 賦予唯一 ID 並設定其 metadata URI。

腳本運行與部署：

透過 Hardhat 的部署腳本，合約被部署到 Sepolia 網絡。腳本首先通過 hre.ethers.getContractFactory 加載合約，然後使用 deploy()方法將其部署到網絡。

Hardhat 設定：

hardhat.config.js 中定義了合約的 Solidity 編譯器版本為 0.8.20，並配置了 Sepolia 測試網絡的連接參數，包括 Node URL 和部署賬戶的私鑰。

合約:

hw4 > MyNFT.sol

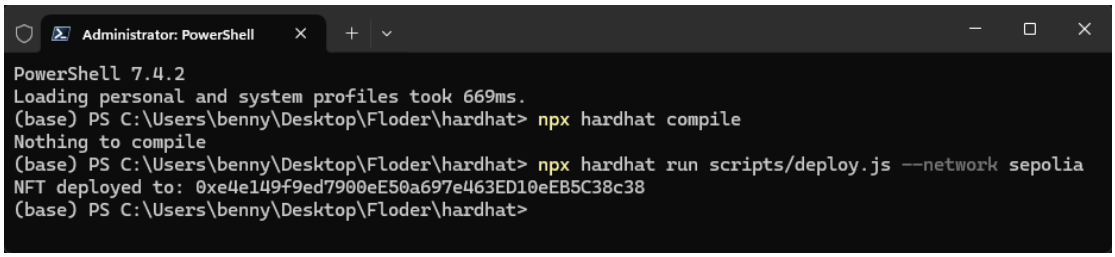
```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract MyNFT is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("MyNFT", "MNFT") {}

    function mintNFT(address recipient, string memory tokenURI)
        public
        returns (uint256)
    {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);

        return newItemId;
    }
}

```

Deploy:

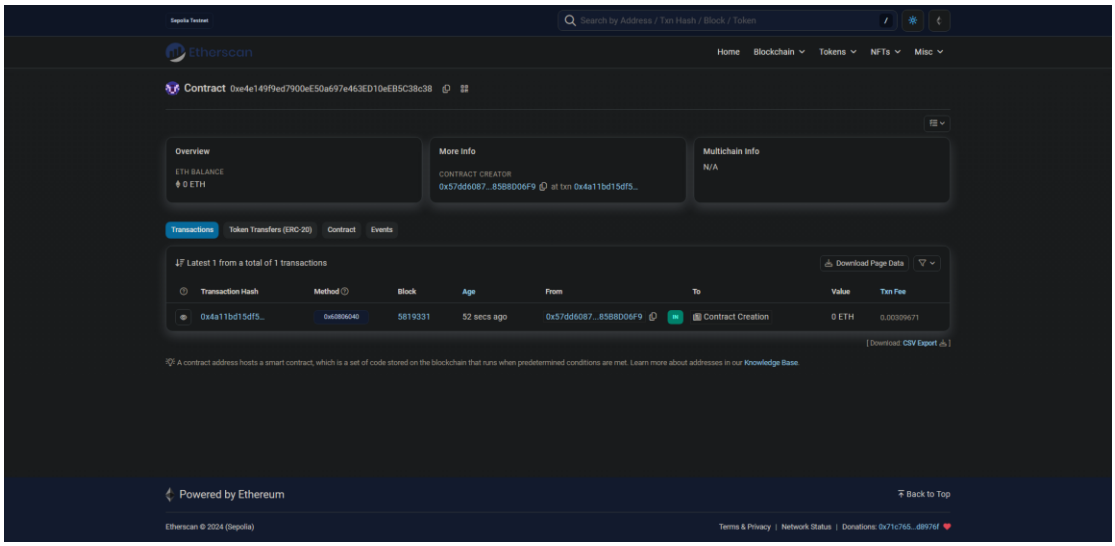
hw4 > JS deploy.js > ...

```js
async function main() {
    const NFT = await hre.ethers.getContractFactory("MyNFT");
    const nft = await NFT.deploy();

    console.log("NFT deployed to:", nft.address);
}

main().catch((error) => {
    console.error(error);
    process.exitCode = 1;
});

```

執行:



0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38



1. 從 Hardhat 環境獲取已部署合約的一些參數。
2. 使用合約地址將合約代碼依附到已部署的合約實例。
3. 定義鑄造 NFT 的接收者地址和元數據 URI。
4. 調用合約的 mintNFT 方法來創造一個新的 NFT。
5. 輸出新創建的 NFT 的 Token ID 到控制台。

交互:

```
scripts > JS mintNFT.js > ...
async function main() {
    const contractAddress = "0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38";
    const [owner] = await ethers.getSigners();
    const NFTContract = await ethers.getContractFactory("MyNFT");
    const contract = await NFTContract.attach(contractAddress); // Attach to deployed contract

    const recipient = owner.address; // The recipient of the NFT
    const metadataURI = "https://metadata.location.com/token-id.json"; // The metadata URI

    // Mint NFT and get the transaction response
    const response = await contract.mintNFT(recipient, metadataURI);
    // Wait for the transaction to be mined to get the token ID from event
    const receipt = await response.wait();

    // Find the transfer event from the transaction receipt to extract the token ID
    const transferEvent = receipt.events?.filter((x) => { return x.event == 'Transfer' })[0];
    const tokenId = transferEvent.args.tokenId;

    console.log(`Minted NFT with token ID: ${tokenId}`);
}

main().catch((error) => {
    console.error(error);
    process.exitCode = 1;
});
```

執行:

```
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat run scripts/mintNFT.js --network sepolia
Minted NFT with token ID: 1
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat run scripts/mintNFT.js --network sepolia
Minted NFT with token ID: 2
(base) PS C:\Users\benny\Desktop\Floder\hardhat>
```

區塊鍊上有: