

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，常用於追蹤代幣 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。

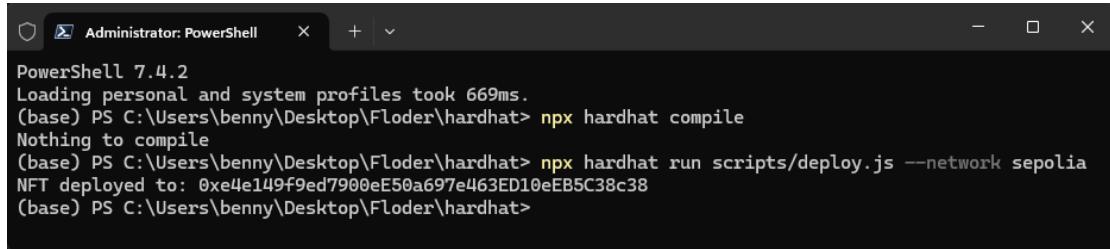
合約:

```
hw4 > ⚒ MyNFT.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
6 import "@openzeppelin/contracts/utils/Counters.sol";
7
8 contract MyNFT is ERC721URIStorage {
9     using Counters for Counters.Counter;
10    Counters.Counter private _tokenIds;
11
12    constructor() ERC721("MyNFT", "MNFT") {}
13
14    function mintNFT(address recipient, string memory tokenURI)
15        public
16        returns (uint256)
17    {
18        _tokenIds.increment();
19
20        uint256 newItemId = _tokenIds.current();
21        _mint(recipient, newItemId);
22        _setTokenURI(newItemId, tokenURI);
23
24        return newItemId;
25    }
26}
27
```

Deploy:

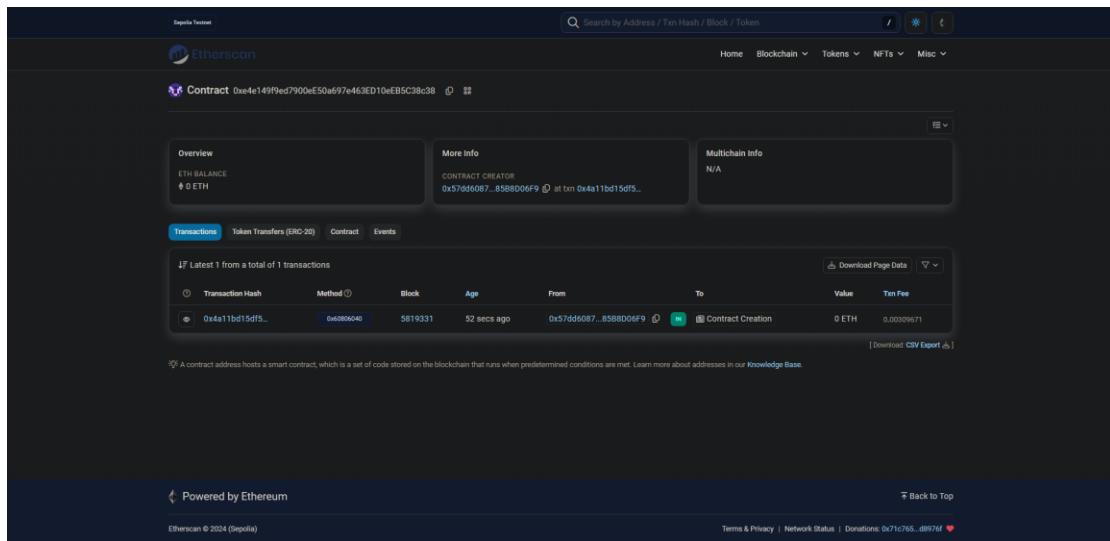
```
hw4 > JS deploy.js > ...
1 async function main() {
2     const NFT = await hre.ethers.getContractFactory("MyNFT");
3     const nft = await NFT.deploy();
4
5     console.log("NFT deployed to:", nft.address);
6 }
7
8 main().catch((error) => {
9     console.error(error);
10    process.exitCode = 1;
11 });
12 |
```

執行：



```
PowerShell 7.4.2
Loading personal and system profiles took 669ms.
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat compile
Nothing to compile
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat run scripts/deploy.js --network sepolia
NFT deployed to: 0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38
(base) PS C:\Users\benny\Desktop\Floder\hardhat>
```

0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38



The screenshot shows the Etherscan interface for the deployed NFT contract. The address is 0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38. The 'Transactions' tab is selected, showing one transaction from block 5819331 at timestamp 52 secs ago. The transaction hash is 0x57dd6087...8588D06F9, and it is labeled as 'Contract Creation'. The value transferred was 0 ETH with a gas fee of 0.000399671. The 'More Info' section shows the contract creator as 0x57dd6087...8588D06F9 at tx 0x4a11bd15df5...

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 > ...
1  async function main() {
2    const contractAddress = "0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38";
3    const [owner] = await ethers.getSigners();
4    const NFTContract = await ethers.getContractFactory("MyNFT");
5    const contract = await NFTContract.attach(contractAddress); // Attach to deployed contract
6
7    const recipient = owner.address; // The recipient of the NFT
8    const metadataURI = "https://metadata.location.com/token-id.json"; // The metadata URI
9
10   // Mint NFT and get the transaction response
11   const response = await contract.mintNFT(recipient, metadataURI);
12   // Wait for the transaction to be mined to get the token ID from event
13   const receipt = await response.wait();
14
15   // Find the transfer event from the transaction receipt to extract the token ID
16   const transferEvent = receipt.events?.filter((x) => { return x.event == 'Transfer' })[0];
17   const tokenId = transferEvent.args.tokenId;
18
19   console.log(`Minted NFT with token ID: ${tokenId}`);
20 }
21
22 main().catch((error) => {
23   console.error(error);
24   process.exitCode = 1;
25 });
26 |
```

## 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:

The screenshot shows the Etherscan interface for the MyNFT contract. At the top, it displays the contract address: 0xe4e149f9ed7900eE50a697e463ED10eEB5C38c38. Below this, there are three tabs: Overview, More Info, and Multichain Info. The Overview tab shows ETH BALANCE: \$0 ETH. The More Info tab shows CONTRACT CREATOR: 0x57666087...858B006F9 at txn 0xa11bd15df5... and TOKEN TRACKER: MyNFT (MNFT). The Multichain Info tab shows N/A. At the bottom, the Transactions tab is selected, showing the latest 3 transactions from a total of 3. The table includes columns for Transaction Hash, Method, Block, Age, From, To, Value, and Gas Fee.

Transaction Hash	Method	Block	Age	From	To	Value	Gas Fee
0x1c5ab5e90c...	Mint NFT	5819376	1 min ago	0x57666087...858B006F9	0xe4e149f9...EB5C38c38	0 ETH	0.00001108
0x392100766c...	Mint NFT	5819374	1 min ago	0x57666087...858B006F9	0xe4e149f9...EB5C38c38	0 ETH	0.00002425
0x4a11bd15df5...	9d009640	5819331	10 mins ago	0x57666087...858B006F9	Contract Creation	0 ETH	0.00306671