Homework 3: Programming an ERC20 Token contract (Using Hardhat/OpenZeppelin)

110321006 蔡秉翰、110321007 劉竑毅

摘要：

MyToken 是基於 OpenZeppelin 庫實現的 ERC20 代幣智能合約。 合約包含了標準的 ERC20 功能，並擴展了鑄造（mint）和燃燒（burn）功能。 這份報告總結了合約的主要組件和部署過程。

合約功能概述：

受到 OpenZeppelin 標準 ERC20 和 Ownable 合約的支持，確保了合約的基本功能和所有權管理。

建構函式接收初始供應量作為參數，並將代幣初始發行給合約的部署者。

提供了鑄造功能，允許合約的擁有者為任何地址增加新的代幣。

提供了一個燃燒功能，允許任何代幣持有者銷毀他們持有的代幣。

關鍵函數或元件：

constructor：在合約部署時初始化代幣的名稱、代號並鑄造初始供應量傳給部署者。

mint：被合約的擁有者召喚來鑄造代幣。

burn：允許代幣持有者燃燒自己的代幣，從而減少流通中的代幣總量。

部署概覽：

部署腳本使用 Hardhat 開發框架的指令 npx hardhat run scripts/deploy.js network sepolia 在 Sepolia 測試網路上部署合約。

使用 Ethers 函式庫取得簽署者和合約工廠，建立 MyToken 實例。

部署過程中傳遞了必要的構造參數 —— 初始代幣供應量設為 1000 代幣。

部署完成後，腳本輸出合約的部署位址。

合約所有權的控制使得僅授權用戶能夠鑄造代幣，降低了惡意行為的風險。

燃燒函數呼叫時會銷毀呼叫者帳戶中指定數量的代幣，而無需接收其他使用者額外的權限。

合約:

contracts > MyToken.sol

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

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyToken is ERC20, Ownable {
    constructor(
        uint256 initialSupply
    )
        ERC20("MyToken", "MTK")
        Ownable(msg.sender)
    {
        _mint(msg.sender, initialSupply);
    }

    // 鑄造函數允許合約擁有者鑄造代幣
    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    // 燃燒函數允許代幣持有者燒毀自己的代幣
    function burn(uint256 amount) public {
        _burn(msg.sender, amount);
    }
}

```

部屬:

scripts > JS deploy.js > catch() callback

```js
async function main() {
    const [deployer] = await ethers.getSigners();
    const MyToken = await ethers.getContractFactory("MyToken");

    const myTokenDeployed = await MyToken.deploy(
        ethers.utils.parseUnits("1000", 18)
    );

    await myTokenDeployed.deployed();
    console.log("合約部署在地址:", myTokenDeployed.address);
}

main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error("部署脚本发生错误:", error);
        process.exit(1);
    });
```

```
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat run scripts/deploy.js --network sepolia
Compiled 1 Solidity file successfully (evm target: paris).
合約部署在地址: 0xaAA34F6F3cEAe487e12510244E6cC287a757700F
```

交互合約:

scripts > JS interact.js > ...

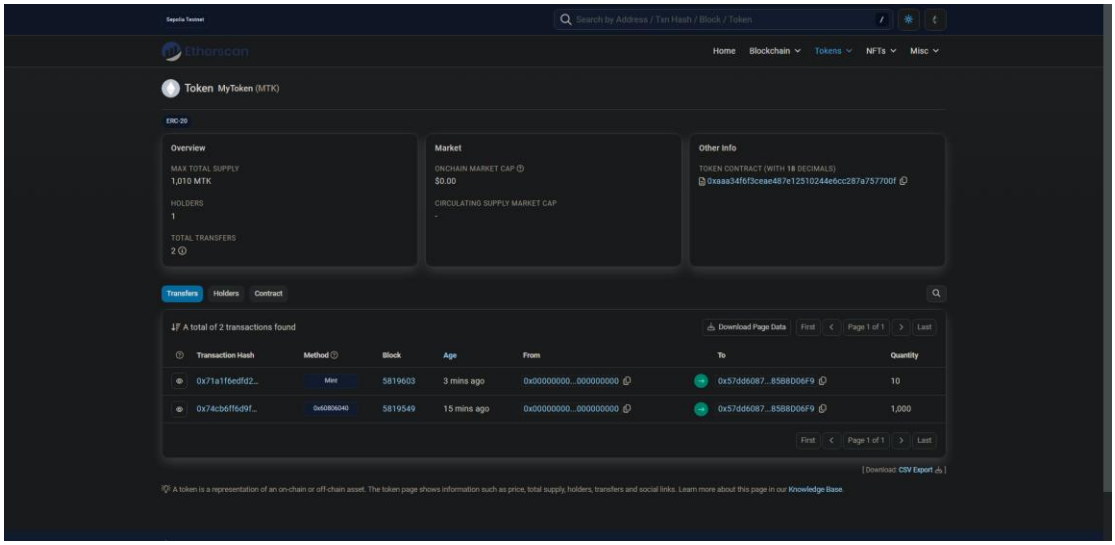
```js
async function main() {
    const contractAddress = "0xaAA34F6F3cEAe487e12510244E6cC287a757700F";
    const myToken = await ethers.getContractAt("MyToken", contractAddress);

    // 查詢部署者的代幣餘額
    const [deployer] = await ethers.getSigners();
    const balance = await myToken.balanceOf(deployer.address);
    console.log(`部署者的代幣餘額是: ${balance.toString()}`);

    // 如果您是合約的擁有者，可以進行鑄造操作
    const mintTx = await myToken.mint(deployer.address, ethers.utils.parseUnits("10", 18))
    await mintTx.wait();
    console.log("鑄造了10個代幣");
}

main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error("交互脚本發生錯誤:", error);
        process.exit(1);
    });
```

可以看到呼叫了函式:




```
(base) PS C:\Users\benny\Desktop\Floder\hardhat> npx hardhat run scripts/interact.js --network sepolia
部署者的代幣餘額是： 10000000000000000000
鑄造了10個代幣
```

添加到錢包: