使用 OpenZeppelin 创建代币

> *内容出自**C2N数字游民部落***
>
> 加入社区获得更多材料：https://discord.gg/R2qW7yqg

了解代币的重要性以及它们在区块链中的使用方式。

学习目标

学完本模块后，你将能够：

- 说明什么是代币
- 识别不同类型的代币
- 使用 OpenZeppelin 中的合约库
- 创建代币智能合约

先决条件

- 之前使用过 C、Python 或 JavaScript 等编程语言
- 基本了解编程概念
- 熟悉用于创建新指令和安装程序的命令行
- 安装了 Node.js
- 已安装 Hardhat
- 已安装 Visual Studio Code

简介

无论是比特币、Ether 还是其他加密货币，数字资产都表示价值金额。 另一个表示价值的数字资产的例子是代币。 但是，对于代币，价值可能不是货币。 代币可表示能够交易的时间、服务、财产或商品。

在本模块中，你将学习代币的重要性以及它们在区块链中的使用方式。

什么是代币？

虽然经常被误解，但区块链代币是一种强大的工具。 区块链代币几乎可以表示我们想要赋予其价值的任何东西。 这种表示形式包括事物是虚拟的还是属于现实世界的有形物体。

代币可以向用户授予特殊平台权限、授予对服务的独占访问权或表示所有权。 代币也可表示具有货币价值的加密货币。 接下来深入了解代币的历史，探索各种用例。 我们将准确地演示代币是什么以及你能用代币做什么。

代币的历史

代币并不新鲜。 代币早在区块链和加密货币出现之前就已经存在。

在其漫长的历史中，代币表示任何形式的经济价值。 贝壳和珍珠可能是最早的代币形式，当时它们用于交易。 今天的代币形式包括赌场筹码、代金券、航空公司积分、证券、音乐会入场券、晚餐预订券、身份识别卡和俱乐部会员资格。 实际上，表示和使用代币的可能方法的列表不受限制。

甚至包括现金和硬币在内的典型法定货币也是一种代币。 代币的使用思路是，你可以积累代币，这些代币具有一定的分配价值，然后用它们来交易，换取一些有价值的商品或服务。

在计算中，代币用于授予用户完成操作或管理访问的权限。 例如，当我们上网时，Web 浏览器会向网站发送代币。 这些代币包含我们的计算机的相关信息，例如浏览器和 IP 地址。 另一个计算机代币的例子是 QR 码。 扫描 QR 码后，QR 码中包含的信息会将我们重定向到网页或服务。

Blockchain Tokens

在区块链中，代币更类似于其一般用途：它们表示一件商品的所有权、对服务的访问权或货币价值。 财产或服务可以是公共的，就像任何人都可加入和参与的 Ethereum 网络一样。 财产或服务也可以是私有的，比如所购房屋或者你的帐户上用来购买商品或服务的抵现积分。

区块链代币表示智能合约中编程的一组规则。 每个代币都属于唯一标识它的区块链地址。 可使用加密钱包访问代币。 只有拥有该地址私钥的人才能访问这些代币。

Ethereum 区块链平台基于代币的使用，代币可以买卖和交易。 区块链代币一般类似于 Ethereum 平台上使用的加密货币 Ether (ETH)。 但是，它们可能反映出不同的设计决策和目的。 代币有助于运行去中心化应用 (Dapp)，并简化不同区块链生态系统的加密经济学。 Augur 就是一个使用代币的区块链生态系统。

Augur 是一个网络，起着预测市场的作用。 你可以把它看作是一个赌博平台。 Augur 有自己的代币形式 REP（表示信誉），报告者用它来澄清关于预测市场结果的争议。 如果预测正确，会获得 REP。 否则，你就会失去 REP。

在 Augur 中使用代币和平时使用代币（比如去乘坐公共交通工具）没有什么不同。 若要使用公共交通工具，一方面，你可以使用法定货币或当地货币，比如加拿大元 (CAD)。 此外，你也可使用多伦多交通委员会 (TTC) 代币等等。 使用每个 TTC 代币可以乘坐一次地铁。 你可以使用 CAD 购买 TTC 代币（就像你可使用 ETH 购买 REP 一样），但它们不是一回事。 TTC 代币是多伦多交通生态系统特有的代币，而 REP 代币仅在 Augur 生态系统中使用。

两类区块链代币

区块链中有不同类型的代币，但通常分为两大类：

- 可互换代币。 可互换代币的特征是：
  - *等效*
  - 可交换
  - 按你拥有的代币数量来确定价值
- 不可互换代币。 不可互换代币的特征是：
  - *唯一*
  - *Distinct*
  - 按你拥有的代币种类来确定价值

细分来看，区块链代币本质上就是利用 Ethereum 区块链的智能合约。 Ethereum 中的所有内容都可表示为智能合约，而且没有任何规则来限制智能合约的用途。 因此，社区制定了标准来记录代币合约如何与其他合约一起运作。 这些标准还给出了每种代币类型的实现细节。

了解合约标准

Ethereum 改进建议 (EIP) 介绍了 Ethereum 平台的标准。 这些建议包括核心协议规范、客户端 API 和合约标准。 社区成员可通过 EIP 为 Ethereum 平台的各个方面建议新的标准。

Ethereum 征求意见 (ERC) 中定义了代币标准。 尽管有新的标准在不断被提出和接受，但已广泛采用了下面 4 种主要 ERC 类型的标准：

- ERC20
- ERC721
- ERC777
- ERC1155

让我们来探索每种类型的代币。 我们将花一点时间来了解是什么让每一种代币变得重要且独特。

ERC20

ERC20 代币最广为人知、使用范围最广。 ERC20 是 Ethereum 区块链上的智能合约用来实现代币的技术标准。 ERC20 为基本代币提供了一个简单接口。

可以使用 ERC20 代币合约跟踪可互换代币。 任何一个 ERC20 代币都与任何其他代币完全相同。 此外，ERC20 代币没有任何关联的特殊权利或行为，这一特性使得这种代币在押注、货币交换和投票权等方面非常有用。

ERC721

ERC721 是不可互换代币 (NFT) 的顶级解决方案。 与其他所有代币一样，NFT 既表示虚拟资产的所有权，也表示实际资产的所有权。 这些资产可能包括：

- 收藏品，如古董、卡片或艺术品

- 实物资产，如房屋或汽车
- 负值资产，如贷款

每个代币都是唯一的，而且都有必须跟踪的所有权和状态。

尽管 ERC721 代币和 ERC20 代币类似，但在复杂性上有所不同。 ERC721 代币要复杂一些。 此外，对于 ERC721 代币，每个函数还有一个参数，用于指定唯一标识智能合约中使用的代币的 ID。

ERC777

ERC777 是一种更丰富的标准，用于可互换代币。 此标准可用于新的用例，构建在先前代币标准的学习成果的基础之上。 此标准与 ERC20 后向兼容，这意味着你可以像 ERC20 代币一样与 ERC777 代币进行交互。 你可使用 ERC777 代币进行更复杂交易的交互。

ERC1155

ERC1155 是一种用于管理多种代币类型的标准。 一个合约可表示多个可互换代币和不可互换代币。

ERC1155 借鉴了 ERC20、ERC721 和 ERC777 的理念。

出于几项原因，ERC1155 代币类型的设计考虑到了大幅节省 gas。（在 Ethereum 中，gas 是指为执行交易而收取的费用或定价值。）首先，你可将此代币合约用于多个代币，这意味着部署更少且复杂度更低。 它还具有批处理操作，因此单个函数调用可以更简单、更少消耗 gas。

了解 OpenZeppelin

OpenZeppelin 是一个平台，它具有可用来编写、部署和管理去中心化应用程序的工具。
OpenZeppelin 是一款开源工具，它通过提供的产品来提供可靠性和安全性。

OpenZeppelin 提供两个产品：合约库和 SDK。

协定

OpenZeppelin 合约库为 Ethereum 网络提供了一组可靠的模块化和可重用的智能合约。 智能合约是在 Solidity 中编写的。 使用 OpenZeppelin 合约的主要好处是它们已经过彻底的测试、审核和社区审查。

OpenZeppelin 是应用智能合约的行业中最受欢迎的库源，而且是开源的。 使用 OpenZeppelin 合约时，你将了解开发智能合约的最佳做法。 合约类型多种多样，包括：

- 访问控制：在你要决定谁可执行操作时使用。
- 代币：用于创建可交易资产。
- 加油站网络：在你想让你的用户不必付 gas（费用）就能使用合约时使用。
- 实用工具：在你需要通用且有用的工具时使用。

尽管在此模块中，我们将仅使用代币合约，但最好了解其他可用的合约资源。

SDK

你还可使用的一种 OpenZeppelin 产品是 OpenZeppelin SDK。 SDK 提供命令行接口 (CLI)，因此更容易管理智能合约开发。 通过使用 CLI 编译、升级和部署智能合约，可节省数小时的开发时间。 CLI 支持 Ethereum 和其他 Ethereum 虚拟机驱动的区块链。 这些命令是直观、交互式的，有助于指导你完成开发过程。

我们不会在本模块中使用 SDK，但这款工具值得你自行探索并将它用于未来的区块链开发。

练习 - 设置一个新项目并集成 OpenZeppelin

让我们复用上一章节创建Hardhat项目，然后加入 OpenZeppelin 合约库。如果还没创建Hardhat项目,请参照上一章节

设置 OpenZeppelin

接下来，我们希望与 OpenZeppelin 合约库集成。

为此，请在终端中运行 `npm install @openzeppelin/contracts` 。

等待包成功安装到项目中。 你应会在终端中看到类似以下内容的输出：

- 当前安装的版本是 5.0.2，在package.json 中可以看到

请注意，发生了几件事：

1. 包作为依赖项被添加到了 package.json 文件。
2. 一个 node_modules 文件夹从 OpenZeppelin 导入了所有可用的合约，这些合约位于子文件夹 @openzeppelin/contracts 中。

检查该文件夹的内容后将返回以下输出：

```
$  ls node_modules/@openzeppelin/contracts
```

```
README.md  build/    governance/  metatx/      proxy/     token/
access/    finance/  interfaces/  package.json  security/  utils/
```

花些时间查看现可用于你的项目的可用合约源文件。 请特别注意代币合约。 更好地了解每个合约的实现以及通常提供的具体函数。

练习 - 编写一个 ERC20 代币合约

现在我们设置了一个基本项目，接下来使用 OpenZeppelin 的 ERC20 代币标准创建一个新的代币合约。

创建新的代币合约

我们将创建一个代币合约，奖励在区块链中创建新区块的矿工。

首先，在 Visual Studio Code 中打开 我们的hardhat 项目。 打开项目后，在contracts文件夹中新建 ERC20MinerReward.sol。 将以下代码复制到该合约中：

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
contract ERC20MinerReward is ERC20 {
    event LogNewAlert(string description, address indexed _from, uint256 _n);
    constructor() ERC20("MinerReward", "MRW") {}
    function _reward() public {
        _mint(block.coinbase, 20);
        emit LogNewAlert('_rewarded', block.coinbase, block.number);
    }
}
```

了解代码

现在，让我们来看看合约的各个部分。

首先，我们导入要在 pragma 指令之后使用的 OpenZeppelin 合约。 字符串 `import "@openzeppelin/contracts/token/ERC20/ERC20.sol";` 允许合约找到我们将在自己的合约中使用的 ERC20 合约定义。

然后，我们定义一个名为 `LogNewAlert` 的事件，我们稍后将在合约中发出或调用它。

构造函数使用符号 `MRW` 定义一个名为 `MinerReward` 的新 ERC20 代币。 创建合约时，会生成这个新代币。

当调用 `reward` 函数时，当前区块的矿工 `block.coinbase` 会因挖掘该区块获得 20 个 MRW 代币，并且系统会发出一个事件。

生成合约

保存合约文件后，就可开始生成合约了。 我们将使用 Hardhat 来运行该生成项。

1. 打开 hardhat.config.js文件,配置compiler的版本。
2. 在 部分中，确保compiler的版本值为 `0.8.20` 或更高。 此版本号是必需的，因为 OpenZeppelin 合约会将 pragma 指令指定为 `pragma solidity ^0.8.20;` 。我的 hardhat.config.js 如下用的0.8.24 版本：

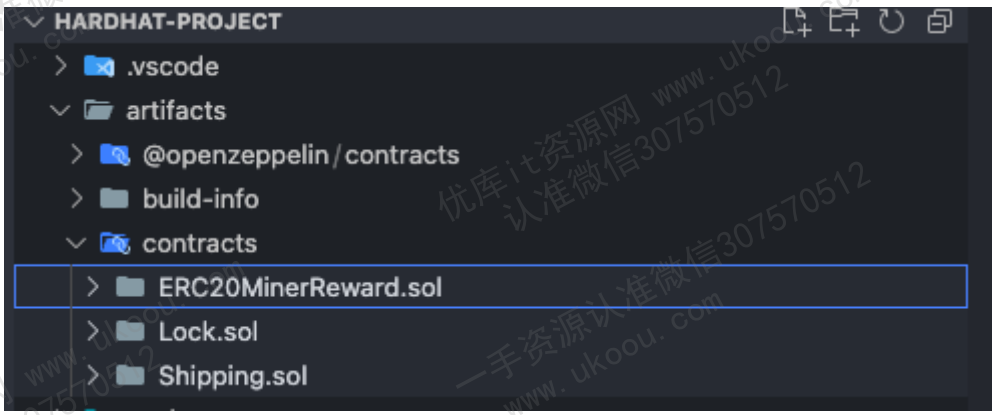
```
require("@nomicfoundation/hardhat-toolbox");

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: {
    compilers: [
      {
        version: "0.8.24",
      },
    ],
  },
};

```

3. 保存文件。
4. 执行 `npx hardhat compile`

编译成功后会在artifacts 文件夹生成相应的文件



请注意，除了合约文件夹中定义的合约外，@openzeppelin/contracts 中的合约也进行了编译。 在继续之前，请确保已成功完成生成。

总结

这个例子是 ERC20 代币的一个基本而直接的实现。 你可以看到编写自己的代币合约是多么轻松，这些合约会继承已定义的 ERC 代币标准中的函数和事件。

请记住，“代币”这个词只是一个隐喻。 它是指由计算机网络或区块链网络共同管理的资产或访问权限。 代币是并入区块链网络的一个重要项目。

若要更熟悉代币，请探索 OpenZeppelin 提供的其他代币合约。 尝试创建你自己的代币合约！