OZ ERC20 Constructor

태그	ERC	ERC-20

- 태그: 작성하려는 글에 맞거나 연관되어 있는 태그를 선택합니다.
- 상태: 해당 글의 상태를 의미합니다.
  - 글을 쓸 예정으로 등록했다면 `작성 예정`, 글을 쓰는 중이거나 아직 완료한 것 같지 않다면 `작성 중`, 완료 후에는 `완료`로 표기합니다.

⚠가독성을 위해 개요와 결론을 작성하시는 것을 추천하지만, 느낌에 따라 자유롭게 작성하시면 됩니다!

개요

OpenZeppelin의 ERC-20 프로토콜에 맞춰서 코드 구현하면서 공식문서에서 제공하는 예시 토큰에 _mint라는 내부 메소드가 존재해서 그걸 찾다보니 _mint에 대해서 알아봐야겠다는 생각이 들었다.

본론

버전변화에 따른 공급 방식 변경

OpenZeppelin v1 에는 아래와 같은 방식으로 사용했다고 한다.

```
contract ERC20FixedSupply is ERC20 {
    constructor() public {
        totalSupply += 1000;
        balances[msg.sender] += 1000;
    }
}
```

하지만 OpenZeppelin v2에서는 totalSupply와 balances가 private화 되어 직접 사용할 수 없고 이 작업을 _mint라는 내부 함수를 통해서 하는 방법으로 변경되었다고 한다.

```
contract ERC20FixedSupply is ERC20 {
    constructor() public {
        _mint(msg.sender, 1000);
    }
}
```

또한 OpenZeppelin에서 제공하는 확장자인 ERC20Mintable을 이용하는 방법도 존재한다고 한다.

```
contract ERC20WithAutoMinerReward is ERC20 {
    function _mintMinerReward() internal {
        _mint(block.coinbase, 1000);
    }
```

```
    function _transfer(address from, address to, uint256 value) internal {
        _mintMinerReward();
        super._transfer(from, to, value);
    }
}
```

결론

_mint와 _burn(반대로 토큰을 파괴하여 공급량을 줄이는 메소드)이 ERC20 표준 프로토콜 속에 있는 함수는 아니지만, OpenZepplin 에서 제공하는 ERC20 인터페이스를 안전하기 사용하기 위함이라는 것을 알았다.

참고

How to implement ERC20 supply mechanisms

In this guide you will learn how to create an ERC20 token with a custom supply mechanism. We will showcase two idiomatic ways to use OpenZeppelin for this purpose that you will be able to apply to your smart contract development

https://forum.openzeppelin.com/t/how-to-implement-erc20-supply-mechanisms/226

ERC 20 - OpenZeppelin Docs

This set of interfaces, contracts, and utilities are all related to the ERC20 Token Standard.

https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#ERC20-_transfer-address-address-uint256-