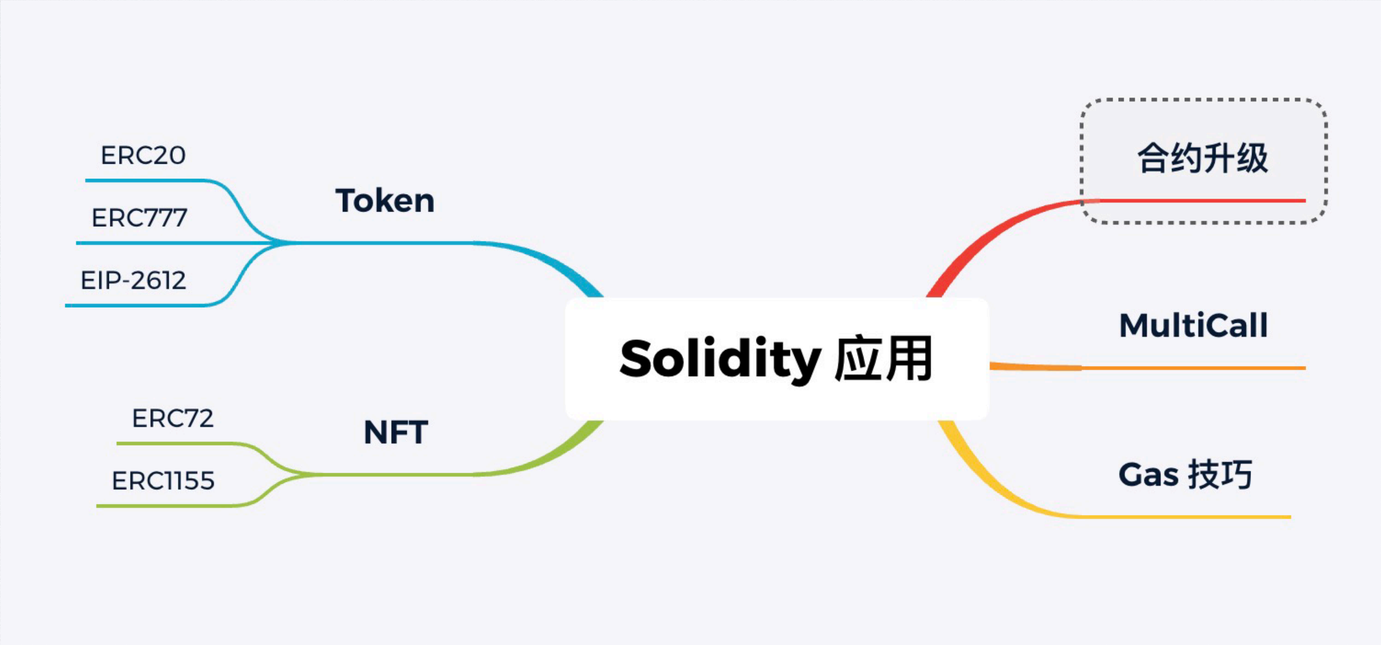
# 区块链集训营
# 二期

登链社区 - Tiny熊

# W3-2




https://github.com/xilibi2003/training_camp_2

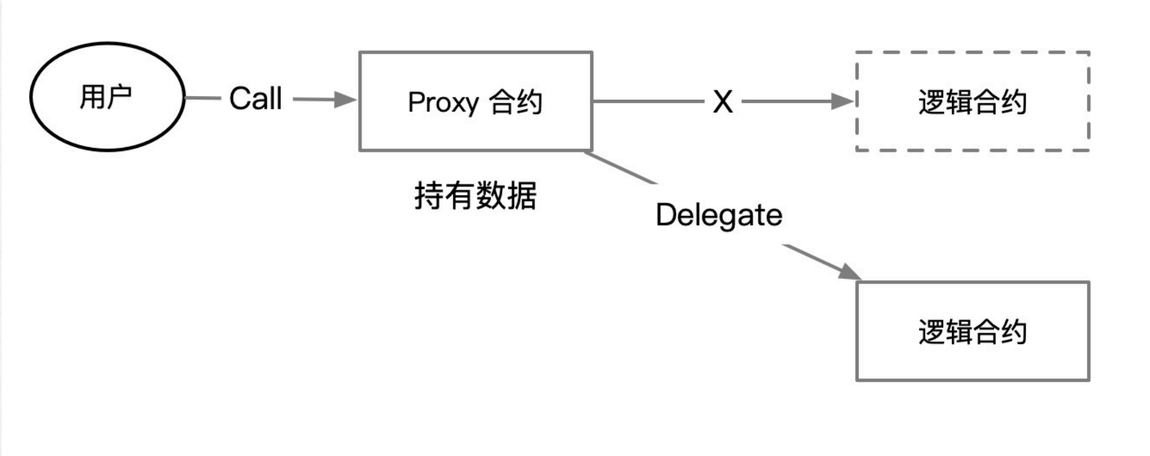
讲课代码是 main 分支.　文件夹：w3_2_code

# 合约升级

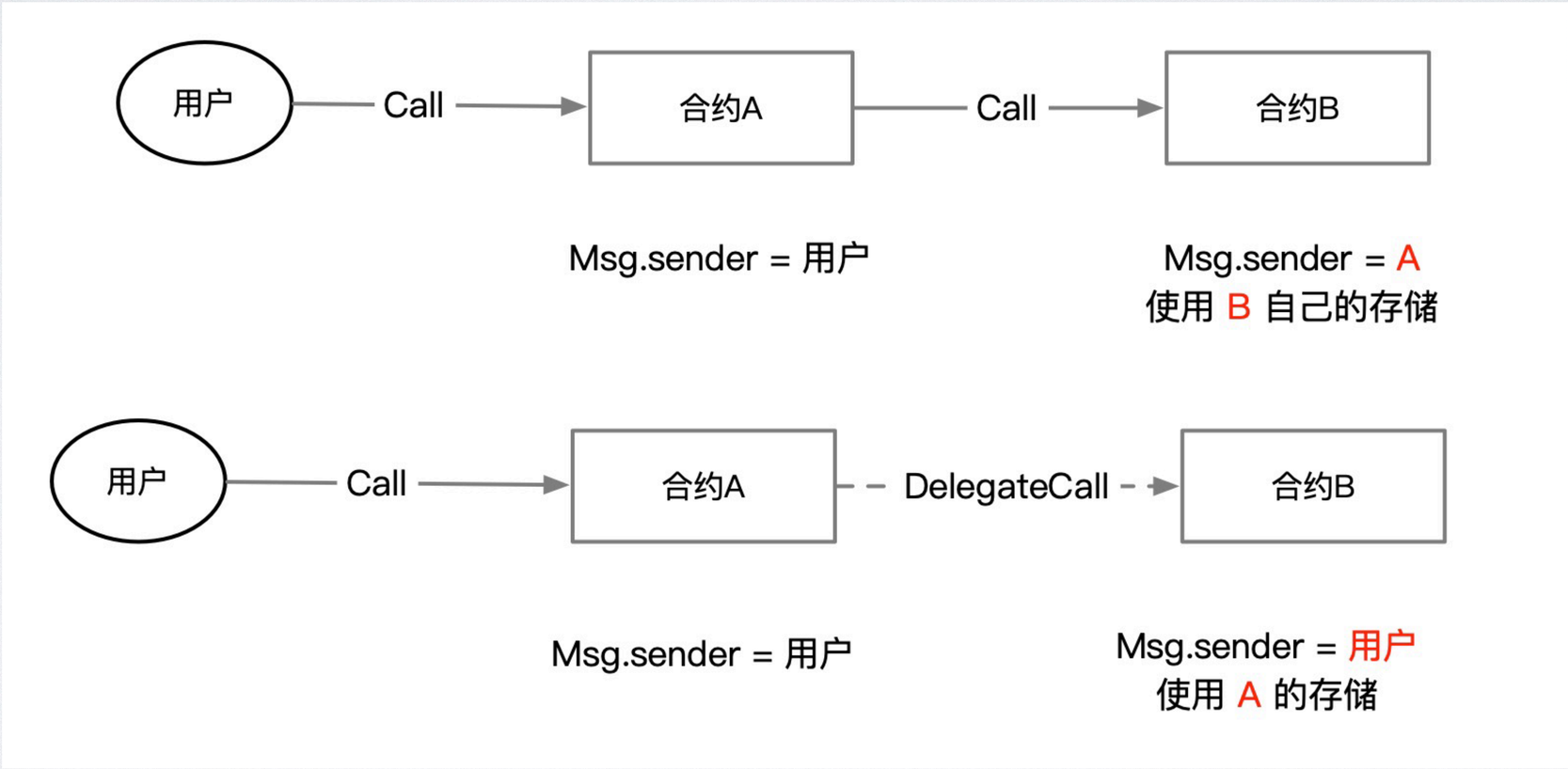

合约升级
重温 Call/DelegateCall
解决1. 代理和逻辑合约存储布局问题
解决2： 函数调用及返回值泛化
解决 3： 函数冲撞

# 合约升级

- 合约一旦部署，便不可更改
- 合约中有错误如何修复?
- 要添加额外功能，要怎么办?



# Call 与 delegateCall 回顾




W2_code : testCall_Delegate.sol

# 使用 DelegateCall 要注意的点

- 代理和逻辑合约的存储布局需要一致。
- delegateCall 返回值
  - (bool success, bytes memory returnData) = address.delegatecall(payload);
  - Bytes 需转化为具体的类型
- 不能有函数冲撞

# 尝试给Counter升级

```
pragma solidity ^0.8.0;

contract Counter {
    uint private counter;

    function add(uint256 i) public {
        counter += 1;
    }

    function get() public view returns(uint) {
        return counter;
    }
}
```

# 升级尝试 1

- 代理和逻辑合约的存储布局不一致发生无法预期的错误

CounterProxy.sol

CounterProxy

| Storage | |
|---|---|
| slot | 变量 |
| 0 | impl |
| 1 | counter |

Counter.sol

| Storage | |
|---|---|
| slot | 变量 |
| 0 | counter |
| | |

**升级添加的变量，必须在末尾添加**

bytes32(uint(keccak256("eip1967.proxy.implementation")) - 1)

# 升级尝试 2

- 如何委托未来添加的函数及获取返回值?
  - fallback 统一委托
  - 用汇编魔法获取返回值

CounterFallback.sol

```
        assembly {
            //获得自由空闲指针
            let ptr := mload(0x40)
            //将返回值从返回缓冲去copy到指针所指位置
            returndatacopy(ptr, 0, returndatasize())

            //根据是否调用成功决定是返回数据还是直接revert整个函数
            switch success
            case 0 { revert(ptr, returndatasize()) }
            default { return(ptr, returndatasize()) }
        }
```

# 升级尝试 – 通用升级

- 函数冲撞问题，若某个函数与upgradeTo() 函数选择器一样会出现什么问题?
- 初始化问题?
- 透明代理（Transparent Proxy）- ERC1967Proxy
- UUPS（universal upgradeable proxy standard）- ERC-1822

# 开发中使用升级

- hardhat-upgrades
  - npm install --save-dev @openzeppelin/hardhat-upgrades
- contracts-upgradeable
  - npm install —save-dev @openzeppelin/contracts-upgradeable

https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable

# MultiCall

- 如何同时调用多个（次）函数?

# Gas 技巧

- 区分 交易 Gas 和 部署 Gas
- 修改变量顺序 -> 合并槽（但尽量使用 uint256 的变量）
- 常量或immutable代替变量
- Struct 合并存储
- 在合约验证数据，而不是存储数据（如使用 Merkel 树）
- 如无依赖，使用事件
- 减少链上数据：IPFS 链下存储

https://learnblockchain.cn/column/1

# 练习题

- 部署一个可升级的 ERC20 Token
  - 第一版本
  - 第二版本，加入方法：function transferWithCallback(address recipient, uint256 amount) external returns (bool)

谢谢大家~