# 记录hardhat踩坑

1.合约实例.address已不能获取到合约地址，得改用合约实例.target

2.在测试代码中，若部署的合约涉及到引用库合约，则先部署库合约，然后导入到部署主合约上 例子如下：

const CSHARE = await hre.ethers.getContractFactory("Contract",{

            libraries:{

                Libname:lib实例.target,

            },

        }

3.另外，在没有配置hardhat运行环境，即hre时，调用如ethers.getContractFactory

时则要加上hre. 若不加上，则会导致找不到

1. loadFixture的使用：

第一次调用 loadFixture 时，将执行fixture。但第二次时，loadFixture 不会再次执行fixture，而是将网络状态重置为执行fixture后的状态。这更快，并且它会撤消先前测试所做的任何状态更改。

可代替掉beforeEach，在每一次测试时不用再次部署合约。 另外如果想重新要一个deployment，则可以不用loadFixture，可以手动部署，例子如下：

    it("Should fail if the unlockTime is not in the future", async function() {

      // We don't use the fixture here because we want a different deployment

      const latestTime = await time.latest();

      const Lock = await ethers.getContractFactory("Lock");

      await expect(Lock.deploy(latestTime, { value:1 })).to.be.revertedWith(

        "Unlock time should be in the future"

      );

});

1. 在hardhat test脚本文件里，若调用合约的函数，函数有返回值且是view修饰符，则可以直接定义一个变量来接收返回值：

合约：function owner() public view returns (address) {

return \_owner;

}

脚本：const owner = await myNFT.owner();

如果不是view修饰，则不能直接定义一个变量接收返回值，如：

函数： function mintNFT(address recipient, string memory tokenURI)

public returns (uint256)

{

require(msg.sender == owner(), "Only owner is allowed to mint");

uint newItemId = ++\_counter;

ERC721.\_mint(recipient, newItemId);

ERC721URIStorage.\_setTokenURI(newItemId, tokenURI);

return newItemId;

}

这样的函数则按上述方法调用会报错，网上解释：You can not directly receive a return value coming from a function that you are sending a transaction to off-chain. You can only do so on-chain - i.e. when one SC function invokes another SC function.

解决办法：再写个合约，自定义变量来接收该函数返回值，即在本合约里导如函数所在合约，创建实例并且调用函数

在写测试脚本的时候，若合约有两个函数同名，但通过传入参数类型不同or数量不同来区分函数的，在hardhat里行不通。此问题仍未解决..........之后补充

时隔三天，已解决：

**解决办法，注册abi：**

如果遇到两个safeTransferFrom函数，但参数类型或个数不同

错误方式contract.safeTransferFrom(addr1, addr2, 1);

正确方式：contract["safeTransferFrom(address,address,uint256)"](addr1, addr2, 1);

Git指令：指定克隆某分支git clone --branch <branchname> <remote-repo-url>或者git clone -b <branchname> <remote-repo-url>

Git rebase 命令：合并分支或者合并之前提交的记录