Desarrollo Blockchain Ethereum con Solidity

Módulo 3 - Transferencia de Ether

Transferencia de Ether

Enviar Ether desde un contrato

Los contratos, al igual que las billeteras, son capaces de almacenar Ether. Por ende, son capaces también de enviar y recibir Ether.

Para enviar Ether a una dirección, primero debe ser **payable.**

Luego, existen distintas formas de transferir:

- **send**: realiza la transacción y devuelve false si la transacción no fue exitosa.
- **transfer**: realiza la transacción y lanza excepción/revert si no es posible realizarla.
- **call/delegatecall**: se utiliza para llamar funciones de otros contratos que reciben parámetros además de Ether.

Enviar Ether desde un contrato

```
address payable x = payable(0xdCad3a6d3569DF655070DEd06cb7A1b2Ccd1D3AF);
address myAddress = address(this);

// transfer
if (x.balance < 10 && myAddress.balance >= 10) x.transfer(10);
// send
x.send(10);
// call
x.call{value: 1 ether}(abi.encodeWithSignature("saludar(string)", "Juan"));
```

Recibiendo pagos desde un contrato

- La palabra **payable** indica que una dirección o una función son capaces de recibir pagos.
- No se puede enviar valor alguno (en Ether) si la función no es **payable.**
- Cuando un contrato recibe un pago, podrá ejecutar funciones especiales si estas están definidas:
  - Si se envían datos buscará una función de tipo **fallback external payable(data).**
  - Si no se envían datos, primero buscará una función **receive external payable()** y en caso de no estar definida, entonces un **fallback** sin parámetros.

```
contract ContratoPagable {
    uint x;
    uint y;

    fallback() external payable { x = 1; y = msg.value; }

    receive() external payable { x = 2; y = msg.value; }
}
```

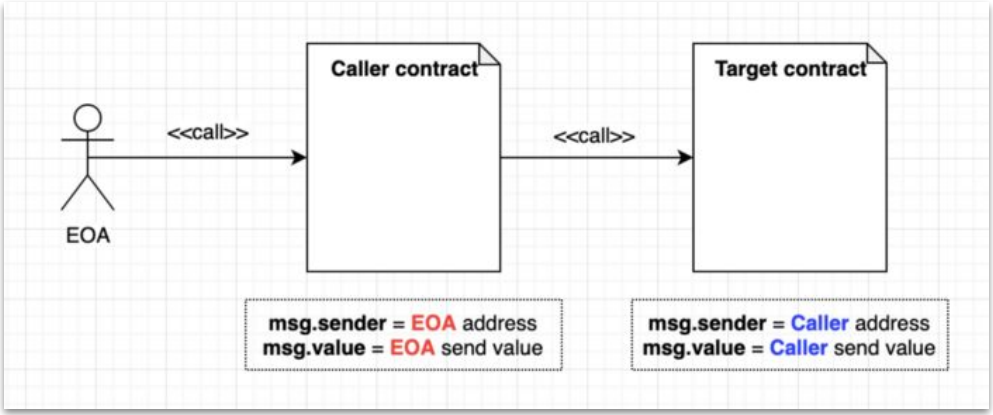
Llamando a contratos desde contratos

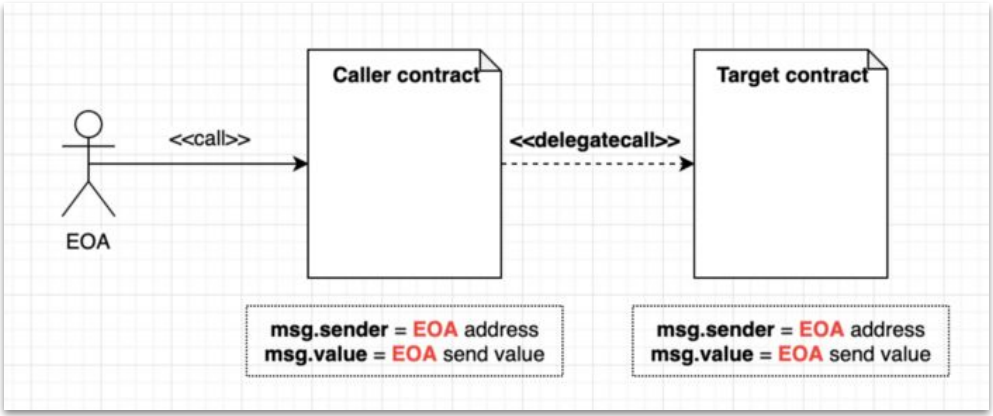
- Es posible llamar a contratos no sólo desde nuestras cuentas personales, si no también desde nuestros contratos.
- Los contratos pueden llamar a funciones y/o pueden enviar Ethers.
- Con la función **call** se puede enviar información pero el contexto es el del contrato receptor (Es decir, el estado que se modifica es del receptor y no del emisor).
- También existe la función **delegatecall** que es similar a call pero el contexto que se utiliza es el del emisor. Por lo tanto hay que evitar el uso de esta función o usarla con mucho cuidado ya que al alterar el estado del contrato original puede delegar la responsabilidad a un contrato malicioso.
- Además, en **delegatecall** no se modifica el valor de msg.sender y msg.value

```
calculator.call(abi.encodeWithSignature("add(uint256,uint256)", a, b));
```

```
calculator.delegatecall(abi.encodeWithSignature("add(uint256,uint256)", a, b));
```

Contextos


Caller contract
Target contract
<<call>>
<<call>>
EOA
msg.sender = EOA address
msg.value = EOA send value
msg.sender = Caller address
msg.value = Caller send value


Caller contract
Target contract
<<call>>
<<delegatecall>>
EOA
msg.sender = EOA address
msg.value = EOA send value
msg.sender = EOA address
msg.value = EOA send value

¡Muchas gracias!

¡Sigamos trabajando!