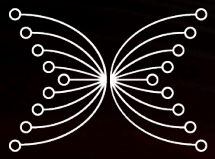

INPUT | OUTPUT

Validadores parametrizados

01

Definición

Validadores parametrizados : Qué son?

Validadores parametrizados : Qué son?

- Son validadores de Plutus que reciben **información adicional** usada en la lógica del validador, lo que los hace más flexibles y reutilizables.

Validadores parametrizados : Qué son?

- Son validadores de Plutus que reciben **información adicional** usada en la lógica del validador, lo que los hace más flexibles y reutilizables.

- Estas funciones **aceptan parámetros para crear validadores** únicos según sus datos de entrada.

Validadores parametrizados : Qué son?

- Son validadores de Plutus que reciben **información adicional** usada en la lógica del validador, lo que los hace más flexibles y reutilizables.

- Estas funciones **aceptan parámetros para crear validadores** únicos según sus datos de entrada.

- Al **aplicar diferentes** parámetros, se generan **validadores distintos**, cada uno con su propio hash y dirección específicos.

Validadores parametrizados : Ejemplo

Digamos que tenemos un script que recibe una **Public Key Hash** como parámetro:

Validadores parametrizados : Ejemplo

Le proveemos un **Public Key Hash** como parámetro y obtenemos el **script final** a partir del cual se calcula la **dirección del script**:

Input
78489d…73ae09

—

→

Output
addr_test1qabc…

Validadores parametrizados : Ejemplo

Ahora, tenemos el mismo **script** previo a la aplicación del **PKH**:

Input
78489d…73ae09

—

→

Output
addr_test1qabc…

Validadores parametrizados : Ejemplo

Al proveerle un **parámetro (PKH) diferente**, la dirección cambia completamente:

Input
78489d…73ae09

Output
addr_test1qabc…

Input
d3fde9…2f2dd0

Output
addr_test1qxyz…

Validadores parametrizados : Ejemplo sin parametrizar

```
pub type VestingDatum {
  beneficiary: VerificationKeyHash,
  deadline: Int
}

validator vesting {
  spend(
    datum: Option<VestingDatum>,
    _redeemer: Data,
    _utxo: OutputReference,
    tx: Transaction
  ) {

    expect Some(vd) = datum
    and {
      must_be_signed_by(tx.extra_signatories, vd.beneficiary)?,
      must_be_after_deadline(tx.validity_range, vd.deadline)?,
    }
  }

  else(_) {
    fail
  }
}
```

Validadores parametrizados : Ejemplo parametrizado

```
pub type VestingParameters {
  VestingParameters {
    beneficiary: VerificationKeyHash,
    deadline: Int
  }
}

validator vesting(p: VestingParameters) {
  spend(
    _datum: Option<Data>,
    _redeemer: Data,
    _utxo: OutputReference,
    tx: Transaction
  ) {

    and {
      must_be_signed_by(tx.extra_signatories, p.beneficiary)?,
      must_be_after_deadline(tx.validity_range, p.deadline)?,
    }
  }

  else(_) {
    fail
  }
}
```

Validadores parametrizados : Código MeshJS

Para poder aplicar parámetros tenemos la función `applyParamsToScript` de MeshJS:

```
applyParamsToScript(codigoDeScript, [param1,param2, ...])
```

Al usar `applyParamsToScript` , hay que proveer los parámetros en el mismo orden.

Validadores parametrizados : ¿Para qué?

Validadores parametrizados : ¿Para qué?

- **Si se pone en el datum**: Puede cambiar en una transacción que actualice el datum.

Validadores parametrizados : ¿Para qué?

- **Si se pone en el datum**: Puede cambiar en una transacción que actualice el datum.

- **Si se hardcodea**: Hay que cambiar el validador. Lo que significa que hay que recompilar, re-testear, y re-auditar el código.

INPUT | OUTPUT

Preguntas?