Traits in Substrate

yuanchao (cdot network)

Introduction to Traits

testing

extensibility

cleaner design

Substrate: as generic as possible

Defining a Trait

```
trait Hash {
        fn hash(s: &[u8]) -> [u8; 256];
}

// associated type
trait Hash {
        type Output;
        fn hash(s: &[u8]) -> Self::Output;
}

// supertrait
trait Hash: Clone {}
```

Implementing a Trait

```
impl Hash for Foo {...}

impl Hash for String {...}

impl Hash for Vec<u8> {...}

impl<T: Hash> Hash for Vec<T> {...}

impl<T: From<U>, U> TryFrom<U> for T {...}
```

Using Traits

```
fn foo<H: Hash>(h: H) {...}

fn foo(h: impl Hash) {...}
```

derive attribute

```
#[derive(Default)]
pub struct Foo {
        bar: Bar,
}

trait Default {
        fn default() -> Self;
}

impl Default for Foo {

        fn default() -> Self {...}
}
```

```
use codec::{Encode, Decode};
/// Reward points of an era. Used to split era total payout between validators.
#[derive(Encode, Decode, Default)]
pub struct EraRewards {
	/// Total number of points. Equals the sum of reward points for each validator.
	total: u32,
	/// Reward at one index correspond to reward for validator in current_elected of this index.
	/// Thus this reward vec is only valid for one elected set.
	rewards: Vec<u32>,
}

pub trait Encode {
	/// Convert self to an owned vector.
	fn encode(&self) -> Vec<u8>;
}
```

https://github.com/paritytech/substrate/blob/dcbe1be4ebb88c6931c79ca93e36c4374bdde23f/srml/staking/src/lib.rs#L320

Block structure

Block header

Extrinsics

Block

Trait: Hash

```
/// Abstraction around hashing
pub trait Hash: Clone + Eq + PartialEq {
	/// The hash type produced.
	type Output: Member + rstd::hash::Hash + AsRef<[u8]> + AsMut<[u8]> + Copy
		+ Default + Encode + Decode;

	/// Produce the hash of some byte-slice.
	fn hash(s: &[u8]) -> Self::Output;
}
```

https://github.com/paritytech/substrate/blob/dcbe1be4ebb88c6931c79ca93e36c4374bdde23f/core/sr-primitives/src/traits.rs#L411

Trait: Header

```
pub trait Header: Clone + Send + Sync + Codec + Eq {
	/// Header number.
	type Number: Member + ::rstd::hash::Hash + Copy + SimpleArithmetic + Codec;
	/// Header hash type
	type Hash: Member + ::rstd::hash::Hash + Copy + Default + SimpleBitOps + Codec + AsRef<[u8]> + AsMut<[u8]>;

	/// Creates new header.
	fn new(number: Self::Number, extrinsics_root: Self::Hash, parent_hash: Self::Hash) -> Self;

	/// Returns a reference to the header number.
	fn number(&self) -> &Self::Number;

	/// Returns a reference to the extrinsics root.
	fn extrinsics_root(&self) -> &Self::Hash;

	/// Returns a reference to the parent hash.
	fn parent_hash(&self) -> &Self::Hash;
}
```

Trait: Extrinsic

```
/// Something that acts like an `Extrinsic`.
pub trait Extrinsic: Sized {
	/// The function call.
	type Call;

	/// Is this `Extrinsic` signed?
	/// If no information are available about signed/unsigned, `None` should be returned.
	fn is_signed(&self) -> Option<bool> { None }

	/// New instance of an unsigned extrinsic aka "inherent". `None` if this is an opaque
	/// extrinsic type.
	fn new_unsigned(_call: Self::Call) -> Option<Self> { None }
}
```

Trait: Block

```
pub trait Block: Clone + Send + Sync + Codec + Eq {
        /// Type of extrinsics.
        type Extrinsic: Member + Codec + Extrinsic;
        /// Header type.
        type Header: Header<Hash=Self::Hash>;
        /// Block hash type.
        type Hash: Member + ::rstd::hash::Hash + Copy + Default + SimpleBitOps + Codec + AsRef<[u8]> + AsMut<[u8]>;


        /// Returns a reference to the header.
        fn header(&self) -> &Self::Header;
        /// Returns a reference to the list of extrinsics.
        fn extrinsics(&self) -> &[Self::Extrinsic];
        /// Creates new block from header and extrinsics.
        fn new(header: Self::Header, extrinsics: Vec<Self::Extrinsic>) -> Self;
        /// Returns the hash of the block.
        fn hash(&self) -> Self::Hash;
}
```

Best practices

use more traits

small

independent

Thanks

Cdot & Substrate 技术社区

该二维码7天内(8月18日前)有效，重新进入将更新