# Substrate

# River

# Basic-006

# Storing a Structure

https://substrate.dev/substrate-collectables-workshop/#/1/storing-a-structure

自定义结构体

```
use parity_codec::{Encode, Decode};

#[derive(Encode, Decode, Default, Clone, PartialEq)]
#[cfg_attr(feature = "std", derive(Debug))]
pub struct MyStruct<A, B> {
    some_number: u32,
    some_generic: A,
    some_other_generic: B,
}
```

使用

```
MyItem: map T::AccountId => MyStruct<T::Balance, T::Hash>;
```

https://doc.rust-lang.org/rust-by-example/trait/derive.html

# 例子

```
decl_module! {
    pub struct Module<T: Trait> for enum Call where origin: T::Origin {
        fn create_struct(origin, value: u32, balance: T::Balance, hash: T::Hash) -> Result {
            let sender = ensure_signed(origin)?;

            let new_struct = MyStruct {
                some_number: value,
                some_generic: balance,
                some_other_generic: hash,
            };

            <MyItem<T>>::insert(sender, new_struct);
            Ok(())
        }
    }
}
```

# Kitty操作

```
// ACTION: Create a `Kitty` object named `new_kitty` here
//   HINT: You can generate a hash with `<T as system::Trait>::Hashing::hash_of(&0)`
//         and you can generate a `0` balance with `<T::Balance as As<u64>>::sa(0)`
let new_kitty = Kitty {
        id: <T as system::Trait>::Hashing::hash_of(&0),
        dna: <T as system::Trait>::Hashing::hash_of(&0),
        price: <T::Balance as As<u64>>::sa(0),
        gen: 0,
};

// ACTION: Store your `new_kitty` into the runtime storage
<OwnedKitty<T>>::insert(sender, new_kitty);
```

Substrate

River

Thanks