开发环境

- 启动一条dev chain

```
$ ./target/debug/utxo --dev
```

- 使用Polkadot UI查看dev chain

https://polkadot.js.org/apps

- 重置dev chain

```
$ ./target/debug/utxo purge-chain --dev
```

新建runtime module

- 安装substrate-up

```
f=`mktemp -d`

git clone https://github.com/paritytech/substrate-up $f

cp -a $f/substrate-* ~/.cargo/bin

cp -a $f/polkadot-* ~/.cargo/bin
```

- 新建utxo module

```
substrate-module-new utxo
```

```
./runtime/src/lib.rs
@@ -59,6 +59,7 @@ pub type Nonce = u64;

 /// Used for the module template in `./template.rs`
 mod template;
+mod utxo;
```

TransactionOutput

```
./runtime/src/utxo.rs
@@ -1,3 +1,5 @@
+use parity_codec::{Decode, Encode};
+use primitives::H256;
 /// A runtime module template with necessary imports

 /// Feel free to remove or edit this file as needed.
@@ -9,6 +11,12 @@
 use support::{decl_event, decl_module, decl_storage, dispatch::Result, StorageValue};
 use system::ensure_signed;

+#[derive(Encode, Decode)]
+struct TransactionOutput {
+    value: u128,
+    pubkey: H256,
+}
+
/// The module's configuration trait.
 pub trait Trait: system::Trait {
     // TODO: Add other types and constants required configure this module.
@@ -24,6 +32,7 @@ decl_storage! {
         // Here we are declaring a StorageValue, `Something` as a Option<u32>
         // `get(something)` is the default getter which returns either the stored `u32` or `None` if nothing stored
         Something get(something): Option<u32>;
+         UnspentOutputs: map H256 => Option<TransactionOutput>;
     }
 }
```

```
{
  "TransactionOutput": {
    "value": "u128",
    "pubkey": "Hash"
  }
}
```

创建一笔UTXO

```
./runtime/src/utxo.rs
@@ -1,5 +1,6 @@
 use parity_codec::{Decode, Encode};
 use primitives::H256;
+use runtime_primitives::traits::{BlakeTwo256, Hash};
 /// A runtime module template with necessary imports

 /// Feel free to remove or edit this file as needed.
@@ -8,8 +9,8 @@ use primitives::H256;

 /// For more guidance on Substrate modules, see the example module
 /// https://github.com/paritytech/substrate/blob/master/srml/example/src/lib.rs
-use support::{decl_event, decl_module, decl_storage, dispatch::Result, StorageValue};
-use system::ensure_signed;
+use support::{decl_event, decl_module, decl_storage, dispatch::Result, StorageMap, StorageValue};
+use system::{ensure_root, ensure_signed};
```

```
@@ -58,6 +59,21 @@ decl_module! {
             Ok(())
         }
+
+        fn mint(origin, value: u128, pubkey: H256) -> Result {
+            ensure_root(origin)?;
+
+            let utxo = TransactionOutput {
+                value: value,
+                pubkey: pubkey,
+            };
+            let hash = BlakeTwo256::hash_of(&utxo);
+            runtime_io::print("new utxo");
+            runtime_io::print(hash.as_bytes());
+            <UnspentOutputs<T>>::insert(hash, utxo);
+
+            Ok(())
+        }
     }
 }
```

subkey

```
USAGE:
    subkey [FLAGS] [OPTIONS] [SUBCOMMAND]

SUBCOMMANDS:
    inspect         Gets a public key and a SS58 address from the provided Secret URI

$ subkey inspect 0x2a52069fc3ab4eb166c7d0a0525994000d3191527b88f8f56fb40a0821138add
```

执行Transaction

```
./runtime/src/utxo.rs
@@ -1,5 +1,5 @@
 use parity_codec::{Decode, Encode};
-use primitives::H256;
+use primitives::{H256, H512};
 use runtime_primitives::traits::{BlakeTwo256, Hash};
 /// A runtime module template with necessary imports

@@ -9,15 +9,34 @@ use runtime_primitives::traits::{BlakeTwo256, Hash};

 /// For more guidance on Substrate modules, see the example module
 /// https://github.com/paritytech/substrate/blob/master/srml/example/src/lib.rs
-use support::{decl_event, decl_module, decl_storage, dispatch::Result, StorageMap, StorageValue};
+use support::{
+    decl_event, decl_module, decl_storage,
+    dispatch::{Result, Vec},
+    StorageMap, StorageValue,
+};
 use system::{ensure_root, ensure_signed};
```

```diff
-#[derive(Encode, Decode)]
+#[cfg_attr(feature = "std", derive(Debug))]
+#[derive(Clone, PartialEq, Eq, Encode, Decode)]
+struct TransactionInput {
+    parent_output: H256,
+    signature: H512,
+}
+
+#[cfg_attr(feature = "std", derive(Debug))]
+#[derive(Clone, PartialEq, Eq, Encode, Decode)]
 struct TransactionOutput {
     value: u128,
     pubkey: H256,
 }

+#[cfg_attr(feature = "std", derive(Debug))]
+#[derive(Clone, PartialEq, Eq, Encode, Decode)]
+pub struct Transaction {
+    inputs: Vec<TransactionInput>,
+    outputs: Vec<TransactionOutput>,
+}
```

```
   // TODO: Add other types and constants required configure this module.
@@ -74,6 +93,25 @@ decl_module! {

             Ok(())
         }
+
+        fn execute(origin, transaction: Transaction) -> Result {
+            ensure_root(origin)?;
+
+            Self::check_transaction(&transaction)?;
+            Self::update_storage(&transaction)?;
+            Self::deposit_event(RawEvent::TransactionExecuted(transaction));
+            Ok(())
+        }
+    }
+}
+
+impl<T: Trait> Module<T> {
+    fn check_transaction(transaction: &Transaction) -> Result {
+        Ok(())
+    }
+
+    fn update_storage(transaction: &Transaction) -> Result {
+        Ok(())
     }
 }

@@ -86,6 +124,7 @@ decl_event!(
         // Event `Something` is declared with a parameter of the type `u32` and `AccountId`
         // To emit this event, we call the deposit funtion, from our runtime funtions
         SomethingStored(u32, AccountId),
+        TransactionExecuted(Transaction),
     }
 );
```

```
{
        "TransactionInput":
        {
                "parent_output": "Hash",
                "signature": "H512"
        },
        "TransactionOutput":
        {
                "value": "u128",
                "pubkey": "Hash"
        },
        "Transaction":
        {
                "inputs": "Vec<TransactionInput>",
                "outputs": "Vec<TransactionOutput>"
        }
}
```

验证和更新存储

```
./runtime/src/utxo.rs
@@ -1,5 +1,6 @@
 use parity_codec::{Decode, Encode};
 use primitives::{H256, H512};
+use runtime_io::sr25519_verify;
 use runtime_primitives::traits::{BlakeTwo256, Hash};
 /// A runtime module template with necessary imports

@@ -9,28 +10,30 @@ use runtime_primitives::traits::{BlakeTwo256, Hash};

+#[cfg(feature = "std")]
+use serde::{Deserialize, Serialize};
 use support::{
     decl_event, decl_module, decl_storage,
     dispatch::{Result, Vec},
-    StorageMap, StorageValue,
+    ensure, StorageMap, StorageValue,
 };
 use system::{ensure_root, ensure_signed};
```

```diff
-#[cfg_attr(feature = "std", derive(Debug))]
+#[cfg_attr(feature = "std", derive(Serialize, Deserialize, Debug))]
 #[derive(Clone, PartialEq, Eq, Encode, Decode)]
 struct TransactionInput {


-#[cfg_attr(feature = "std", derive(Debug))]
+#[cfg_attr(feature = "std", derive(Serialize, Deserialize, Debug))]
 #[derive(Clone, PartialEq, Eq, Encode, Decode)]
 struct TransactionOutput {


-#[cfg_attr(feature = "std", derive(Debug))]
+#[cfg_attr(feature = "std", derive(Serialize, Deserialize, Debug))]
 #[derive(Clone, PartialEq, Eq, Encode, Decode)]
 pub struct Transaction {
     inputs: Vec<TransactionInput>,
@@ -107,10 +110,35 @@ decl_module! {
 impl<T: Trait> Module<T> {
     fn check_transaction(transaction: &Transaction) -> Result {
+        for input in transaction.inputs.iter() {
+            if let Some(output) = <UnspentOutputs<T>>::get(input.parent_output) {
+                ensure!(
+                    sr25519_verify(
+                        input.signature.as_fixed_bytes(),
+                        input.parent_output.as_fixed_bytes(),
+                        output.pubkey
+                    ),
+                    "signature must be valid"
+                );
+            } else {
+                return Err("parent output not found");
+            }
+        }
         Ok(())
     }
```

```
  fn update_storage(transaction: &Transaction) -> Result {
+     for input in &transaction.inputs {
+         <UnspentOutputs<T>>::remove(input.parent_output);
+     }
+
+     for output in &transaction.outputs {
+         let hash = BlakeTwo256::hash_of(output);
+         runtime_io::print("insert utxo");
+         runtime_io::print(hash.as_bytes());
+         <UnspentOutputs<T>>::insert(hash, output);
+     }
+
      Ok(())
  }
```