Hi everyone this is Tracy Salander. By making this video, I would like to share something about Substrate I learned.

What is Substrate Node Template? Substrate Node Template is the smallest Substrate available demo, which can help us to quickly start to study.

Now, let’s see the structure of the Substrate Node Template. We can use ‘brew install tree’ to get ‘tree’ command which will help us to see the structure better. Then we use ‘tree -L 1’, here ‘L’ is layer and ‘L 1’ means layer 1 so we can see the first layer.

[Cargo图]

Now let’s see `Cargo.toml`. It can be used with catch\_unwind to catch exceptions thrown by panic in a thread. Common scenarios are: 1. Embed Rust in other languages. 2. Custom thread processing logic. 3. Test framework, because test cases can be panic, but the running of the test cannot be interrupted. There have three parts in the members right now. ‘node’ is a executable program. It has an executable main function entry in the corresponding ‘src/main.rs’ file. ‘pallets/template’ or ‘pallets/\*’ and ‘runtime’ are a component libraries. They define functions and data structures that can be used externally in ‘src/lib.rs’.

[Node图]

Now let’s see `node/build.rs`. The custom build script is placed in the build.rs file of the project. You can let Cargo compile and execute the script before compiling and building the project. The usage scenarios are: 1. Compile and link third-party non-Rust code. 2. Code generation function before build.

[Scripts图]

Now let’see ‘scripts/init.sh’. It can initialize the compilation environment, including upgrading the version of Rust, including nightly and stable two release channels, and adding support tools for building WebAssembly, which is a one kind of code that can let us run our code online as fast as local. ‘scripts/init’ can execute regular script to solve some common compilation problems such as a dependency installation failure.

[pallets图]

Now let’s see pallets. It contains a custom runtime module, and there is only one template module by default.Take this module as an example: cargo.toml contains:

[pallets\_basic\_info图]

1. The basic infomation of the package, such as name, version, authors.

[pallets\_frame\_support图]

1. The third-party library the package depends on. Take frame-support as an example, set default-features to false, that is, do not use the default features for editing.

[pallets\_features图]

1. Conditional compilation is performed through features. When Cargo is used for building, the following configuration indicates that the std feature is also used by default when compiling dependent libraries such as frame-support. This configuration ensures that the runtime module can either be compiled as a Native execution version by using std feature, or as a Wasm execution version by using no\_std feature and compiled by WasmBuilder.

[pallets\_lib图]

Now let’s see ‘template/src/lib.rs’. In order to ensure the security and stability of the application, Substrate deliberately adds a constraint to the runtime, that is, only the core library and some auxiliary libraries of Rust can be used in the runtime code, but the standard library cannot be used. Using the standard library will cause Wasm execution version compilation to fail. ‘src/lib.rs’ is the specific function realization of runtime module. The first line means that if the feature is not std when compiling, it must be no\_std. Line 9 and line 11 mean that the mock and test modules are only compiled when running tests. Line 24 to line 27 define the interface of the module, inherit from the interface of the frame\_system module, and add an associated type Event. This Event type can be converted into an Event under the frame\_system module, or it can be converted into an Event defined by the current template module.

[pallets\_define\_overall图]

‘#[pallet::storage]’ part defines the storage unit of the module. ‘#[pallet::event]’ part defines the events that the module can trigger. ‘#[pallet::error]’ part defines the types of errors that the module can return. The ‘#[pallet::call]’ part defines callable functions.

[pallets\_mock图]

Now, let’s see ‘src/mock.rs’. It is for setting the tests. Line 10 and 11 are to rename the right side. The ‘parameter\_types’ generates some data types that meet the Get interface required by the following functional modules. From line 27 to line 51, it implements various functional module interfaces for runtime, and here uses a large number of ‘()’ to mock data types that are not concerned. From line 56 to the end, it initializes GenesisConfig and returns a TestExternalities based on HashMap for storage read and write and other extended operations.

[pallets\_tests图]

`src/test.rs` contains all test cases. Line 1 import the data and assertations from mock. A simulated test runtime environment is defined in mock. Sp\_io::TestExternalities defines that this test is stored in memory instead of in the database, which is highly efficient. The’#[test]’ identifies the test function. The tests call functions of the template module and return the execution results. ‘assert\_ok’ is to check the answer is ok or not. ‘assert\_eq’ is to check the answer is equal with to expectation or not. ‘assert\_noop’ is to check the result is Error or not and it won’t change the storage condition on chains.

Thank you for watching.