1 示例一

代码 `message(STATUS "[LIBRARY] Adding modules from ${LIBRARY_NAME}")` 展示

2 示例二

```cpp
//
// Created by mstarch on 12/7/20.
//
#include <gtest/gtest.h>
#include <Drv/Ip/TcpClientSocket.hpp>
#include <Drv/Ip/TcpServerSocket.hpp>
#include <Drv/Ip/IpSocket.hpp>
#include <Os/Log.hpp>
#include <Fw/Logger/Logger.hpp>
#include <Drv/Ip/test/ut/PortSelector.hpp>
#include <Drv/Ip/test/ut/SocketTestHelper.hpp>

Os::Log logger;


void test_with_loop(U32 iterations) {
    Drv::SocketIpStatus status1 = Drv::SOCK_SUCCESS;
    Drv::SocketIpStatus status2 = Drv::SOCK_SUCCESS;

    U16 port =  Drv::Test::get_free_port();
    ASSERT_NE(0, port);
    Drv::TcpServerSocket server;
    server.configure("127.0.0.1", port, 0, 100);
    EXPECT_EQ(server.startup(), Drv::SOCK_SUCCESS);
    Drv::Test::force_recv_timeout(server);

    // Loop through a bunch of client disconnects
    for (U32 i = 0; i < iterations; i++) {
        Drv::TcpClientSocket client;
        ASSERT_NE(port, 0);
        client.configure("127.0.0.1",port,0,100);
        status1 = client.open();
        EXPECT_EQ(status1, Drv::SOCK_SUCCESS);

        status2 = server.open();
        EXPECT_EQ(status2, Drv::SOCK_SUCCESS);


        // If all the opens worked, then run this
        if (Drv::SOCK_SUCCESS == status1 && Drv::SOCK_SUCCESS == status2) {
            // Force the sockets not to hang, if at all possible
            Drv::Test::force_recv_timeout(client);
            Drv::Test::force_recv_timeout(server);
            Drv::Test::send_recv(server, client);
            Drv::Test::send_recv(client, server);
        }
        client.close();
        server.close();
    }
    server.shutdown();
}
```

```
52
53
54   TEST(Nominal, TestNominalTcp) {
55       test_with_loop(1);
56   }
57
58   TEST(Nominal, TestMultipleTcp) {
59       test_with_loop(100);
60   }
61
62   int main(int argc, char** argv) {
63       ::testing::InitGoogleTest(&argc, argv);
64       return RUN_ALL_TESTS();
65   }
```

3   示例三

从外部文件导入代码:

```
1    """
2    Class that contains the post process steps used in the various PGEs
3    that are part of the OPERA PCM pipeline.
4
5    """
6    from typing import Dict
7
8    from chimera.commons.constants import ChimeraConstants as chimera_consts
9    from chimera.postprocess_functions import PostProcessFunctions
10
11   from commons.es_connection import get_grq_es, get_mozart_es
12   from commons.logger import logger
13   from opera_chimera.accountability import OperaAccountability
14
15   grq_es = get_grq_es(logger)
16   mozart_es = get_mozart_es(logger)
17
18
19   class OperaPostProcessFunctions(PostProcessFunctions):
20       def __init__(self, context: Dict, pge_config: Dict, settings, job_result):
21           PostProcessFunctions.__init__(
22               self, context, pge_config, settings, job_result, mozart_es=mozart_es, grq_es=grq_es
23           )
24           logger.info("job_result: {}".format(job_result))
25           self.accountability = OperaAccountability(self._context, job_result.get(chimera_consts.WORK_DIR))
26
27       def update_product_accountability(self):
28           self.accountability.set_products(self._job_result)
29           return {}
```