

SiPESC 插件开发向导

Xiaodong Xu

大连理工大学

2015/4/1

内容目录

1、使用插件接口设计器设计插件.....	3
2、实现插件代码.....	10
3、编写测试程序.....	15

1、使用插件接口设计器设计插件

打开启动面板，搜索关键字“sip”，如图：



图 1

打开“SiPESC 插件设计器”,并新建一个插件，输入插件信息，如图 2 所示。

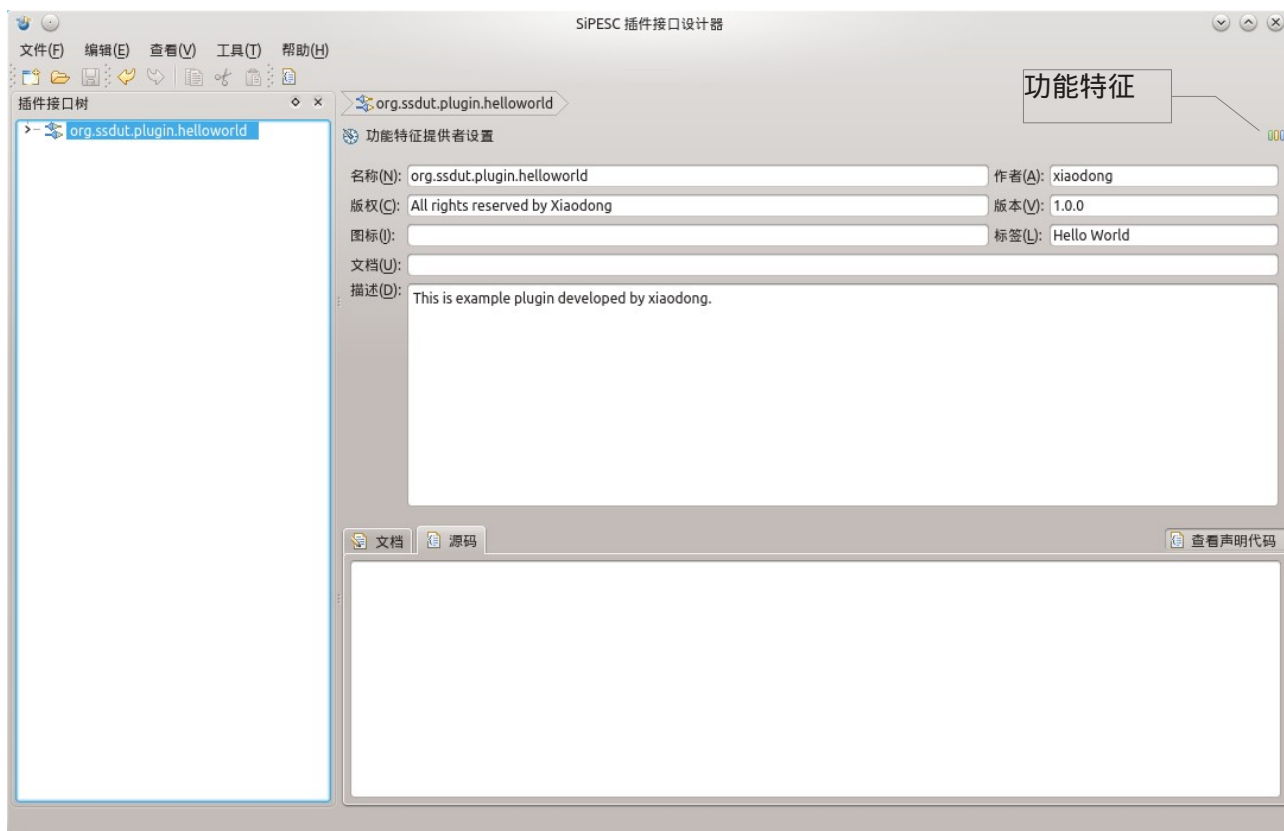


图 2

点击图 2 中的“功能特征”按钮，即可进入图 3,点击“添加功能特征”按钮即可添加一个特征，如图 4 所示。

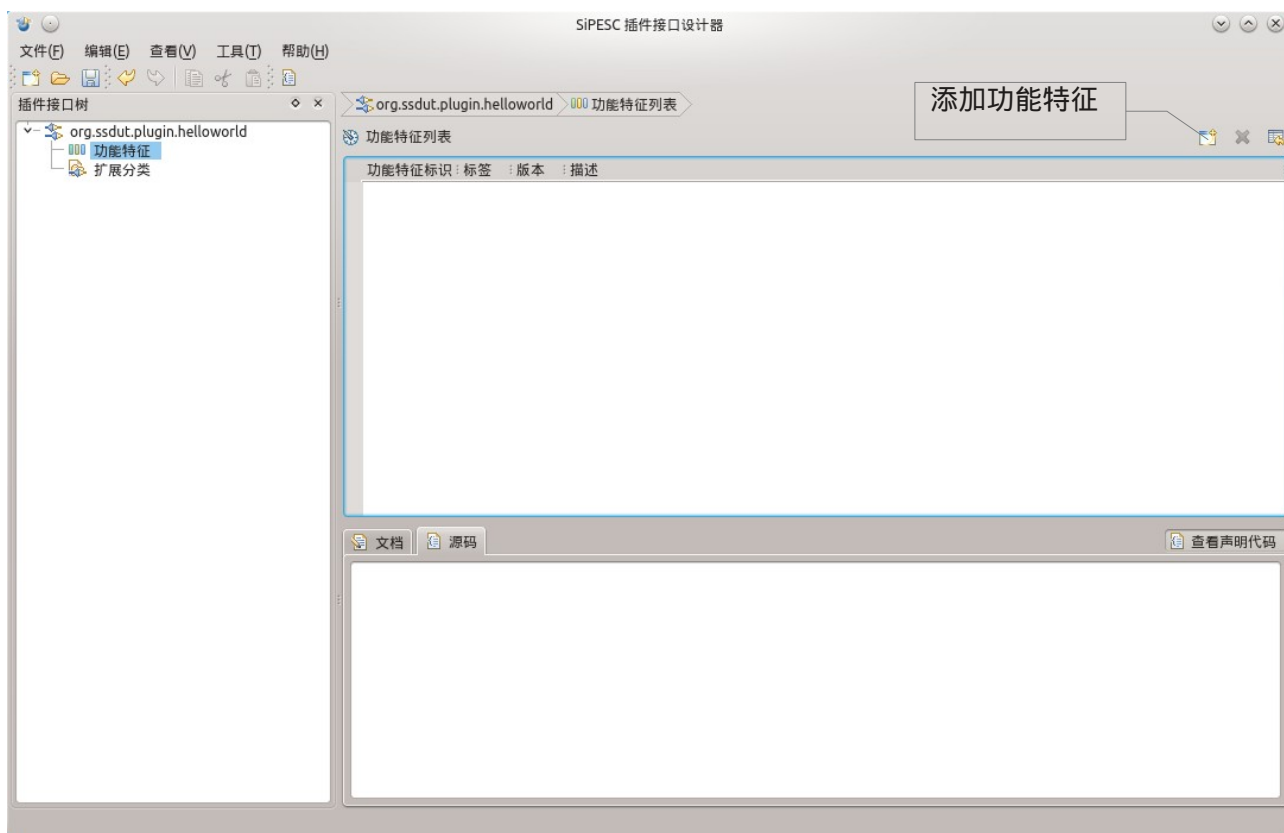


图 3

填写功能特征信息，填好后，点击扩展列表,。

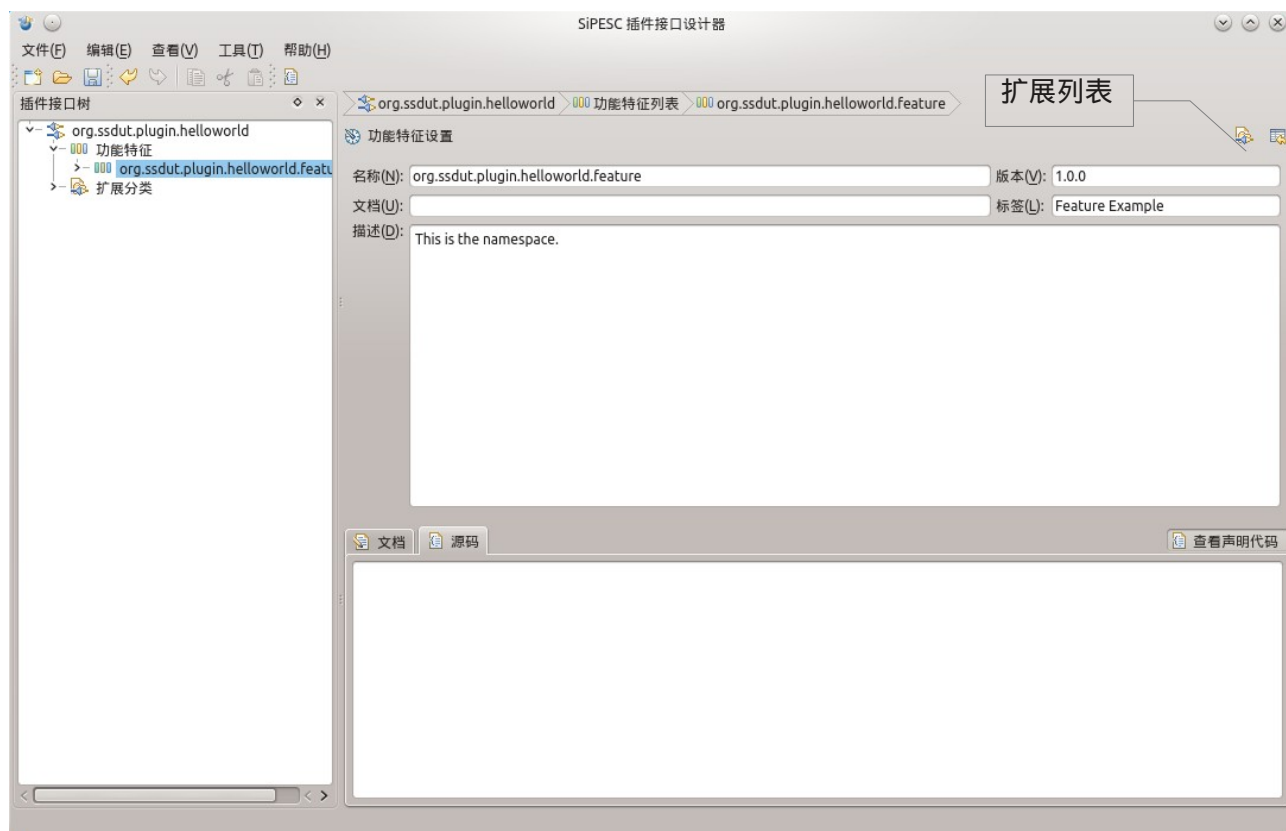


图 4

在图 5 中，可以点击添加一个扩展。

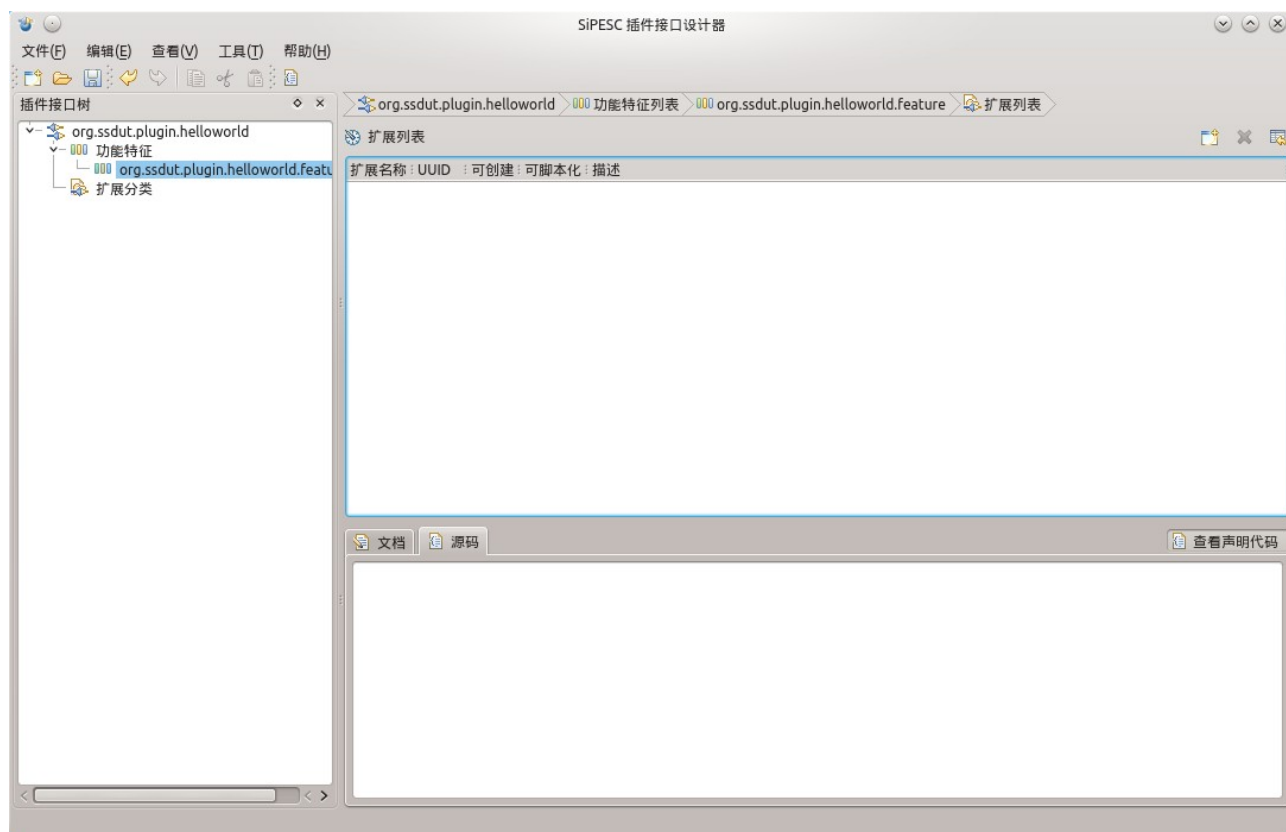


图 5

在图 6 中，填写好扩展的配置信息。

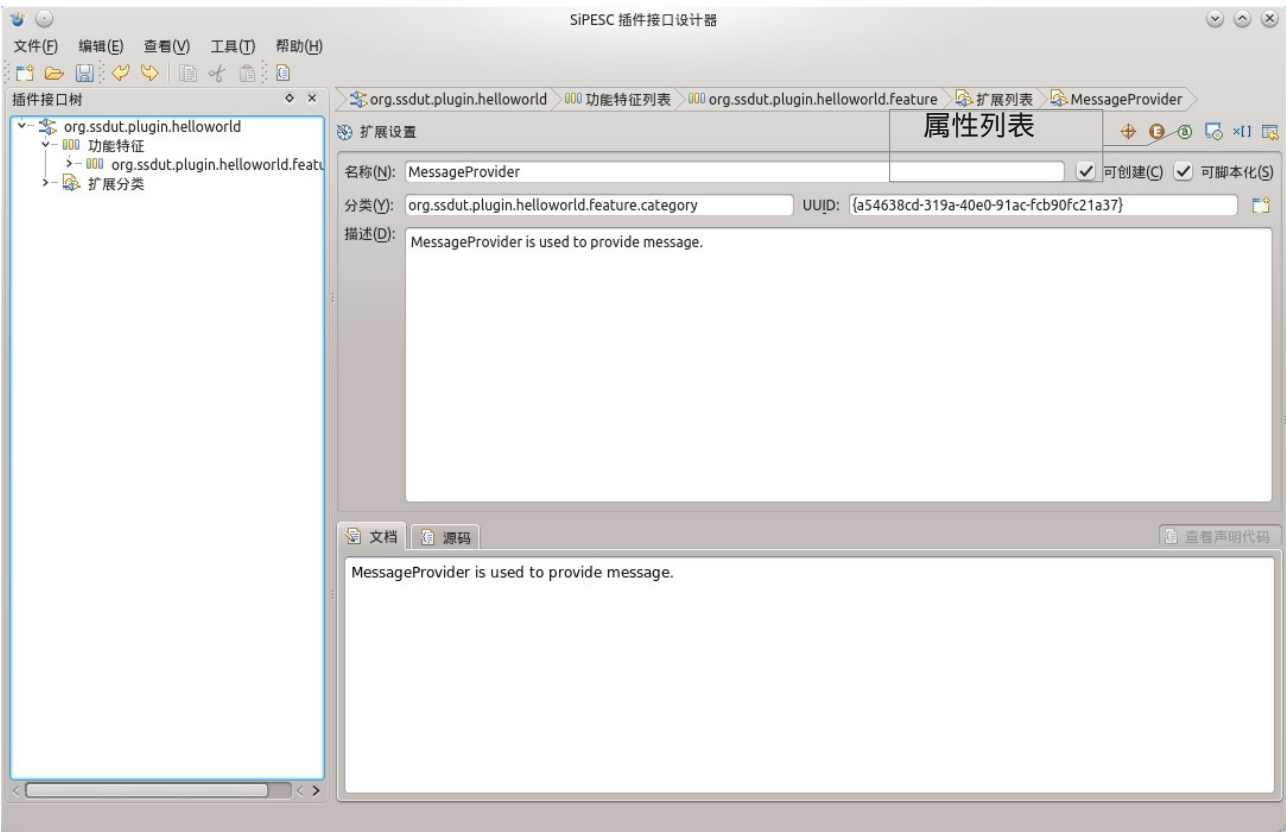


图 6

在图 6 中点击属性列表，添加一个属性“message”，如图 7 所示。

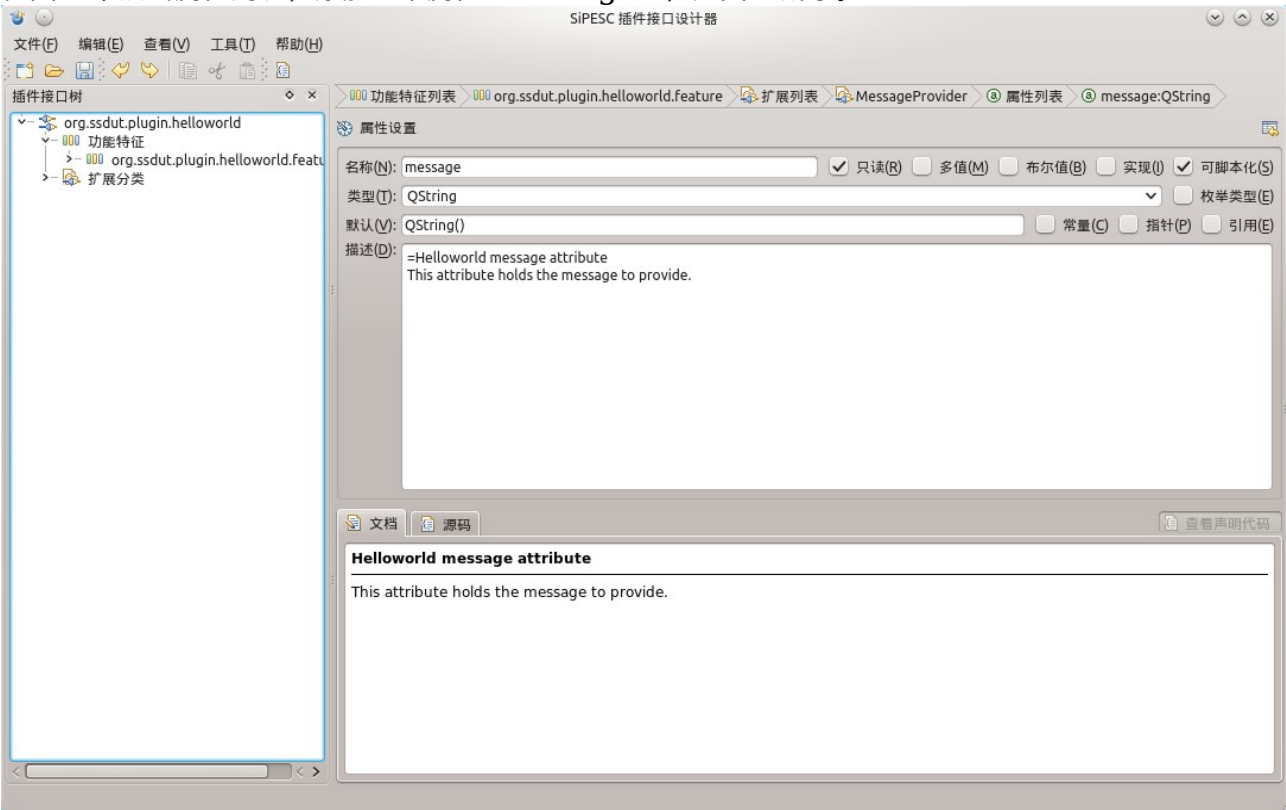


图 7

按照上面的方法，再创建一个扩展“MessageReceiver”。如图 8 所示。

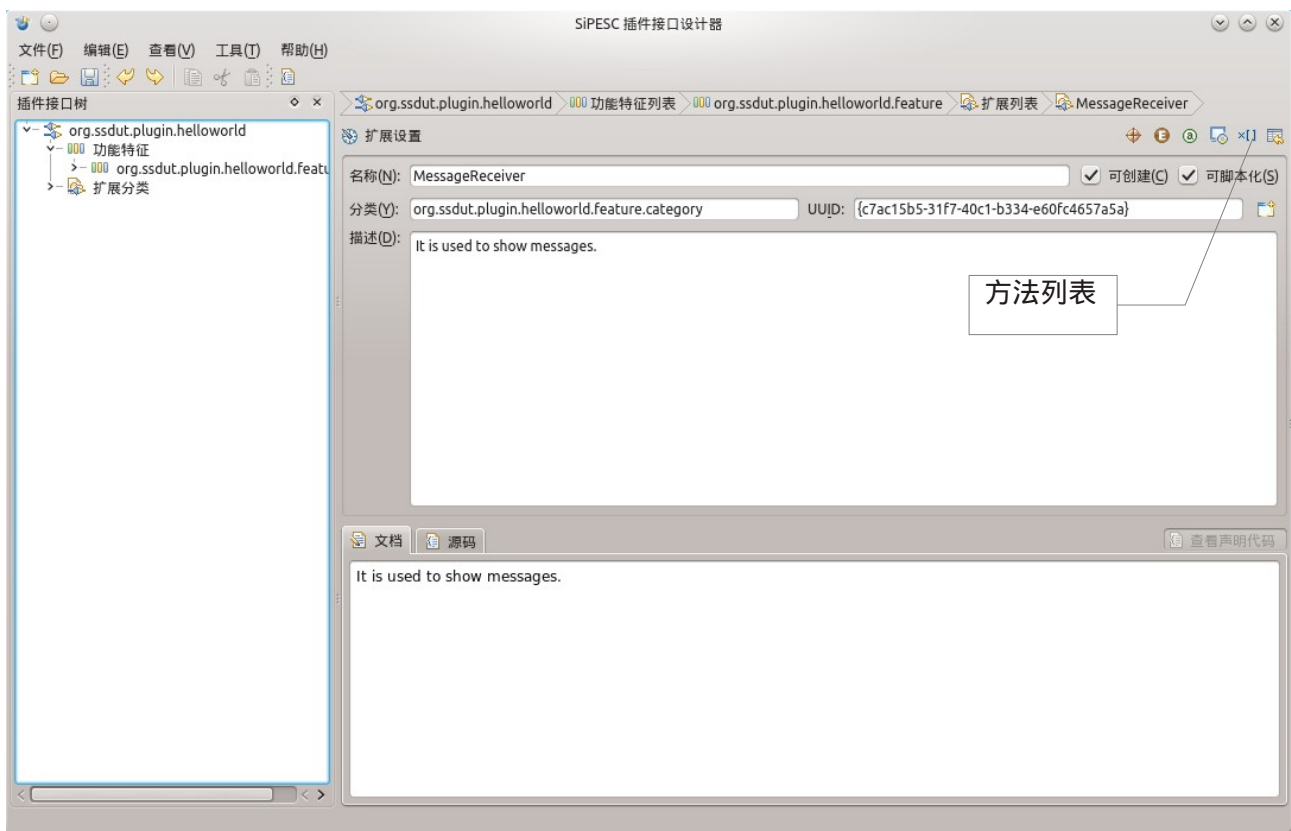


图 8

为“MessageReceiver”扩展添加一个方法，如图 9 所示：

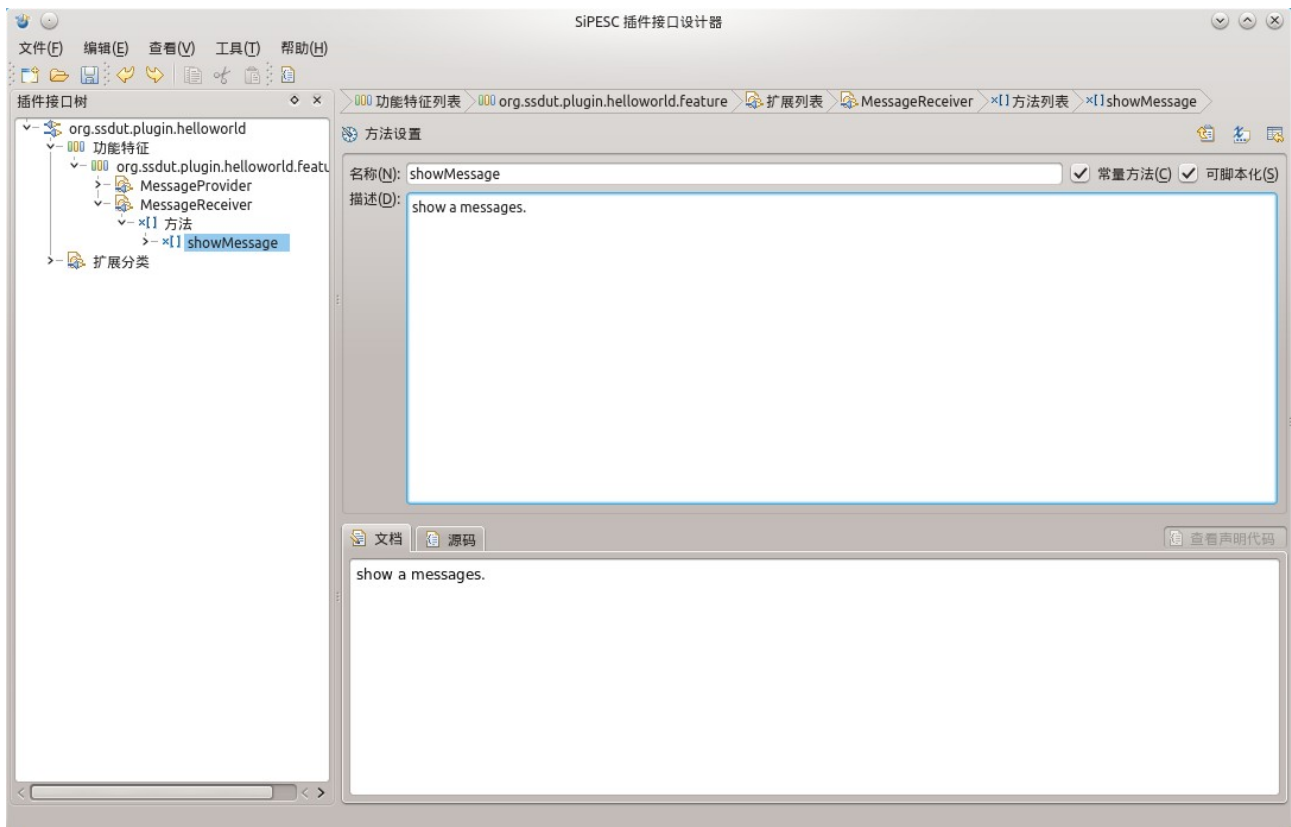


图 9

添加方法时，可以为此方法添加返回值，参数等，此处添加一个参数，如图 10 所示。

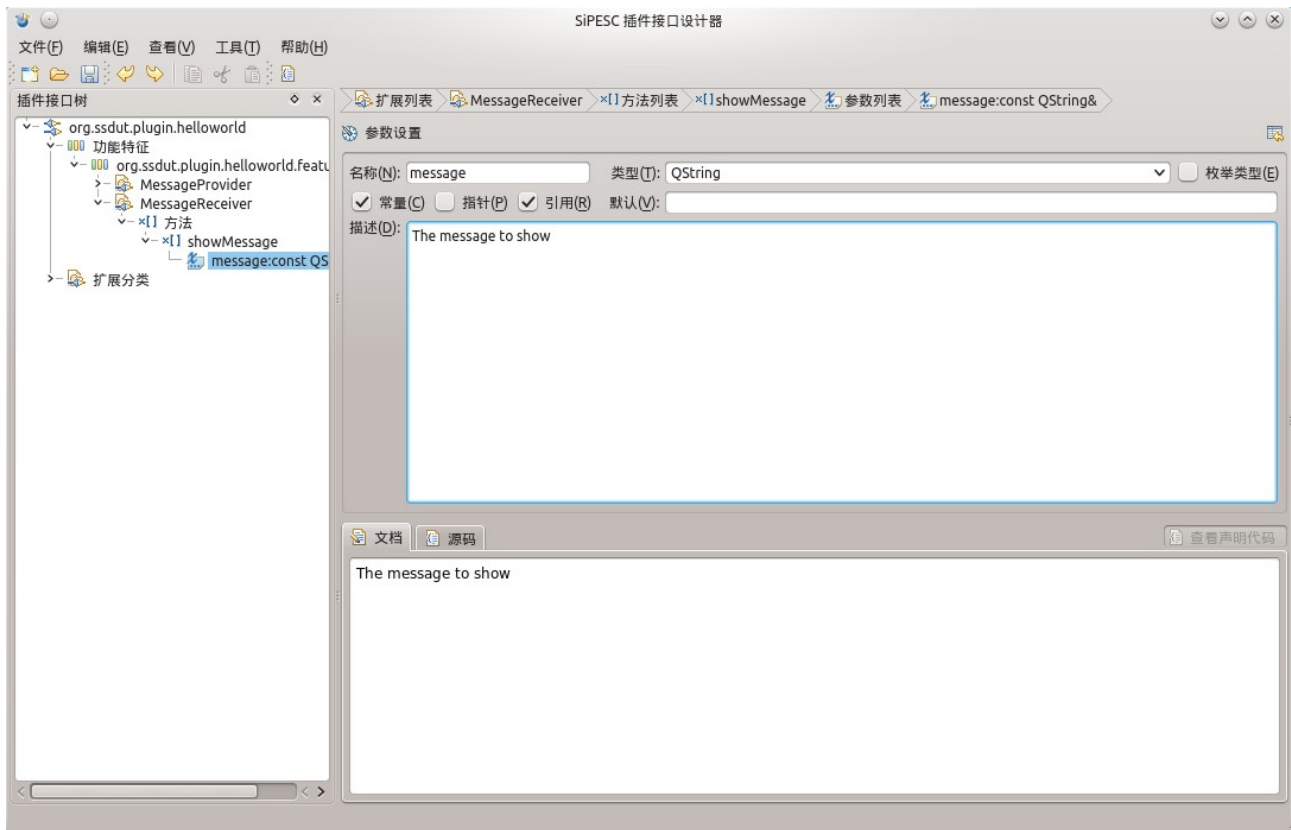


图 10

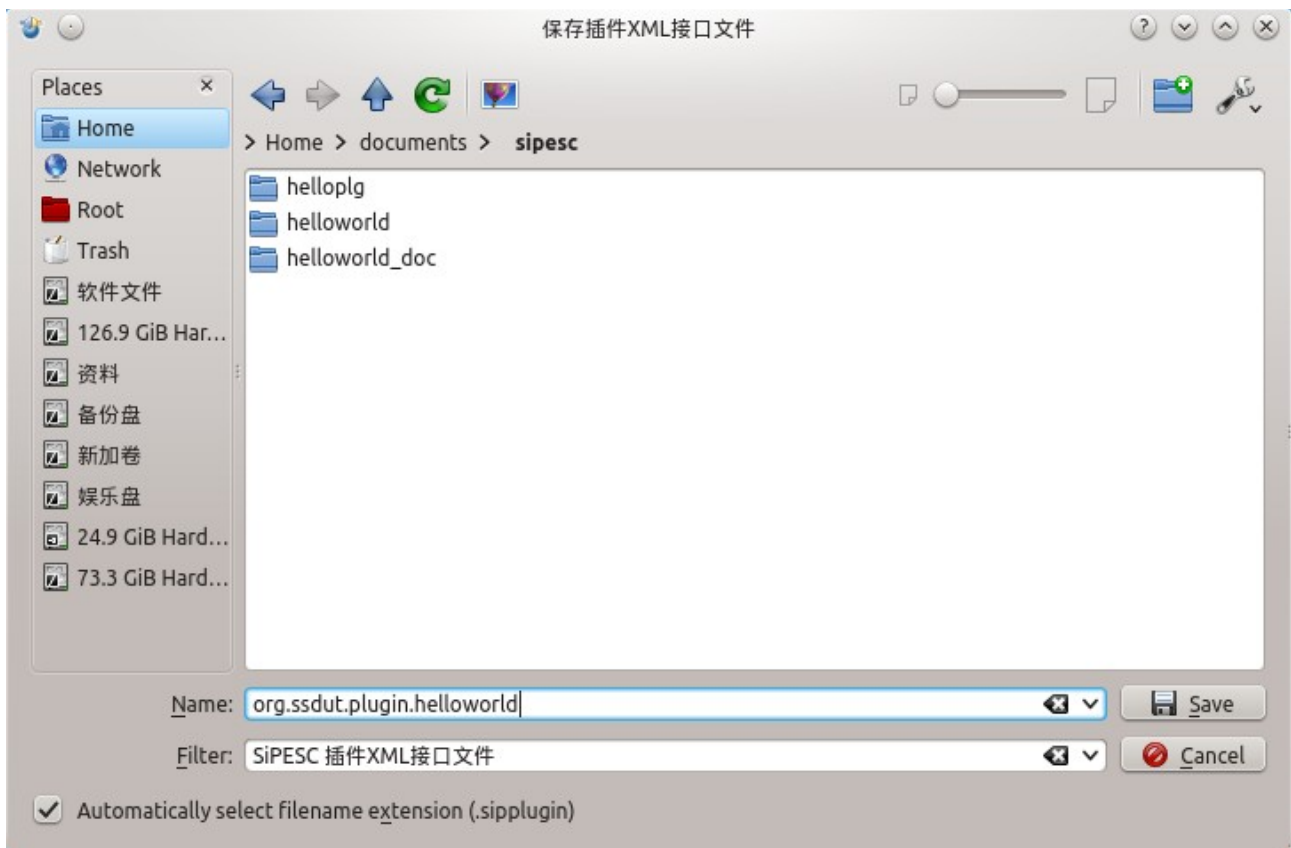


图 11

插件设计好之后，点击保存，选择好保存位置，注意，保存的文件名称需要和以上设计的插件名一致。如图 11。

保存完毕之后，点击工具栏的“生成源代码”按钮，此时弹出对话框如图 12 所示。

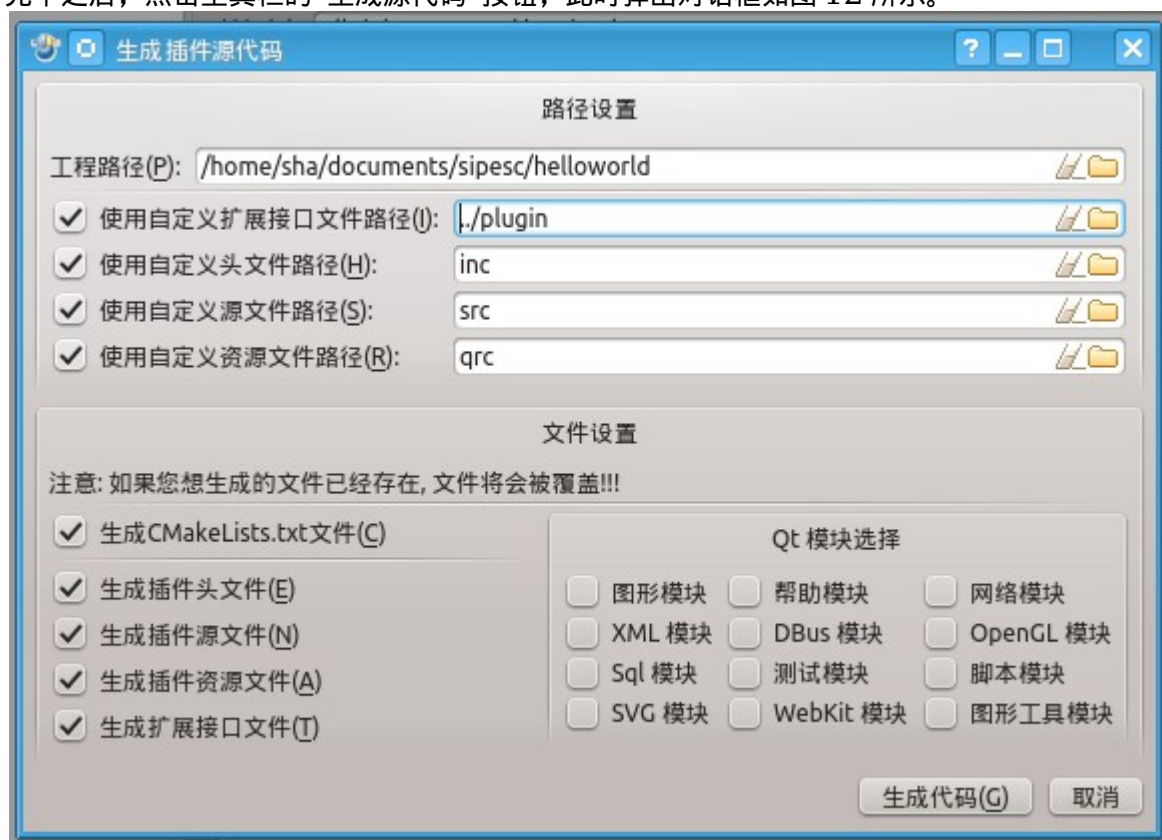


图 12

按图 12 勾选好之后点击“生成代码”，生成成功如图 13 所示。



图 13

2、实现插件代码

在 inc 目录下新建头文件 messageproviderimpl.h, 内容如下:

```
#ifndef MESSAGEPROVIDERIMPL_H
#define MESSAGEPROVIDERIMPL_H
#include <org.ssdut.plugin.helloworld.feature.messageprovider.h>
using namespace org::ssdut::plugin::helloworld::feature;

class MessageProviderImpl: public MessageProviderInterface
{
public:
    MessageProviderImpl();
    ~MessageProviderImpl();
public:
    QString getMessage() const;

private:
    QString _message;
};

#endif
```

在 inc 目录下继续新建头文件 messagereceiverimpl.h, 内容如下:

```
#ifndef MESSAGERECEIVERIMPL_H
#define MESSAGERECEIVERIMPL_H
#include <org.ssdut.plugin.helloworld.feature.messagereceiver.h>
using namespace org::ssdut::plugin::helloworld::feature;

class MessageRecieverImpl: public MessageRecieverInterface
{
public:
    MessageRecieverImpl();
    ~MessageRecieverImpl();
public:
    void showMessage(const QString& message) const;
};

#endif
```

在 src 目录下新建源文件 messageproviderimpl.cxx,内容如下:

```
#include <messageproviderimpl.h>

MessageProviderImpl::MessageProviderImpl()
{
```

```

    _message = "Hello world Plugin from Xiaodong.";
}

MessageProviderImpl::~MessageProviderImpl(){
}

QString MessageProviderImpl:: getMessage() const
{
    return _message;
}

```

在 src 目录下新建另一个源文件 messagereceiverimpl.cxx, 内容如下:

```

#include <qtextstream.h>

#include <messagereceiverimpl.h>

MessageReceiverImpl::MessageReceiverImpl(){
}

MessageReceiverImpl::~MessageReceiverImpl(){
}

void MessageReceiverImpl::showMessage(const QString& message) const{

    QTextStream out(stdout);
    out<<message<<endl;
}

```

修改源文件 org.ssdut.plugin.helloworld.cxx, 红色标记为修改处, 文件内容如下:

```

#include <org.ssdut.plugin.helloworld.h>

#include <messageproviderimpl.h>
#include <messagereceiverimpl.h>
#include <qtranslator.h>

OrgSsdutPluginHelloworldPlugin::OrgSsdutPluginHelloworldPlugin()
{
    _running=false;
}

OrgSsdutPluginHelloworldPlugin::~OrgSsdutPluginHelloworldPlugin()
{
}

bool OrgSsdutPluginHelloworldPlugin::initialize()
{

```

```

return true;
}

bool OrgSsdutPluginHelloworldPlugin::cleanup()
{
return true;
}

void OrgSsdutPluginHelloworldPlugin::start()
{
if(!_running) return;
_running=true;
}

void OrgSsdutPluginHelloworldPlugin::stop()
{
if(!_running) return;
_running=false;
}

QStringList OrgSsdutPluginHelloworldPlugin::
getRequiredFeatures() const
{
return QStringList();
}

MExtensionObject* OrgSsdutPluginHelloworldPlugin
::createOrgSsdutPluginHelloworldFeatureMessageProvider()
{
return new MessageProviderImpl;
}

MExtensionObject* OrgSsdutPluginHelloworldPlugin
::createOrgSsdutPluginHelloworldFeatureMessageReceiver()
{
return new MessageReceiverImpl;
}

```

修改在 helloworld 目录下的 CMakeLists.txt 文件，红色为修改处，内容如下：

```

PROJECT(org.ssdut.plugin.helloworld)

CMAKE_MINIMUM_REQUIRED(VERSION 2.4.0)
IF(COMMAND CMAKE_POLICY)
    CMAKE_POLICY(SET CMP0003 NEW)
ENDIF(COMMAND CMAKE_POLICY)

FILE(TO_CMAKE_PATH $ENV{HOME} HOME_PATH)
SET(CMAKE_MODULE_PATH "${HOME_PATH}/.sipesc/share/cmake")

FIND_PACKAGE(Qt4 REQUIRED)

```

```

FIND_PACKAGE(MExtManagerCore REQUIRED)

SET(QT_DONT_USE_QTGUI 1)
INCLUDE(${QT_USE_FILE})
INCLUDE_DIRECTORIES(
    ${CMAKE_CURRENT_BINARY_DIR}
    ${CMAKE_CURRENT_SOURCE_DIR}/inc
    ${CMAKE_CURRENT_SOURCE_DIR}/../..../plugin
    ${MEXTMGR_CORE_INCLUDE_DIRS}
)

SET(org_ssdt_plugin_helloworld_src_files
    src/messageproviderimpl.cxx
    src/messagereceiverimpl.cxx
    src/org.ssdt.plugin.helloworld.cxx
)

SET(org_ssdt_plugin_helloworld_hdr_files
    inc/org.ssdt.plugin.helloworld.h
)

SET(org_ssdt_plugin_helloworld_qui_files
)

MEXTMGR_WRAP_UI(
    org_ssdt_plugin_helloworld_uis_files
    ${org_ssdt_plugin_helloworld_qui_files}
)
MEXTMGR_WRAP_XML(
    org_ssdt_plugin_helloworld_xml_files
    org.ssdt.plugin.helloworld.sipplugin
)
MEXTMGR_ADD_RESOURCES(
    org_ssdt_plugin_helloworld_qrc_files
    qrc/org.ssdt.plugin.helloworld.qrc
)
MEXTMGR_GEN_PLUGINHEADER(
    org_ssdt_plugin_helloworld_phd_files
    org.ssdt.plugin.helloworld.sipplugin
    ${CMAKE_CURRENT_BINARY_DIR}/plugin
)
MEXTMGR_GEN_PLUGINFEATURE(
    org_ssdt_plugin_helloworld_pfm_files
    org.ssdt.plugin.helloworld.sipplugin
)
MEXTMGR_ADD_TRANSLATION(
    org_ssdt_plugin_helloworld_qms_files
    ${CMAKE_CURRENT_SOURCE_DIR}/inc
    qrc/org.ssdt.plugin.helloworld_zh_CN.ts
    ${org_ssdt_plugin_helloworld_src_files}
    ${org_ssdt_plugin_helloworld_xml_files}
    ${org_ssdt_plugin_helloworld_qui_files}
)

```

```

)

ADD_LIBRARY(
    org.ssdut.plugin.helloworld_1.0.0 SHARED
    ${org_ssdut_plugin_helloworld_src_files}
    ${org_ssdut_plugin_helloworld_xml_files}
    ${org_ssdut_plugin_helloworld_qms_files}
    ${org_ssdut_plugin_helloworld_phd_files}
    ${org_ssdut_plugin_helloworld_pfm_files}
    ${org_ssdut_plugin_helloworld_xml_files}
    ${org_ssdut_plugin_helloworld_qrc_files}
    ${org_ssdut_plugin_helloworld_hdr_files}
    ${org_ssdut_plugin_helloworld_uis_files}
)
SET_TARGET_PROPERTIES(
    org.ssdut.plugin.helloworld_1.0.0 PROPERTIES
    POSTFIX "" PREFIX "" SUFFIX ".sep" AUTOMOC "1"
)
IF(WIN32)
    SET_TARGET_PROPERTIES(
        org.ssdut.plugin.helloworld_1.0.0
        PROPERTIES LINK_FLAGS "-mthreads"
        COMPILE_FLAGS "-frtti -fexceptions -mthreads"
    )
ENDIF(WIN32)
TARGET_LINK_LIBRARIES(
    org.ssdut.plugin.helloworld_1.0.0
    ${QT_LIBRARIES}
    ${MEXTMGR_CORE_LIBRARIES}
)

### Install
INSTALL(
    TARGETS org.ssdut.plugin.helloworld_1.0.0
    RUNTIME DESTINATION lib/plugins/${CMAKE_BUILD_TYPE}
    LIBRARY DESTINATION lib/plugins/${CMAKE_BUILD_TYPE}
    ARCHIVE DESTINATION lib/plugins/${CMAKE_BUILD_TYPE}
)
MESSAGE(STATUS "NOW path CMAKE_CURRENT_BINARY_DIR:${CMAKE_CURRENT_BINARY_DIR}")
INSTALL(
    DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}/plugin
    DESTINATION include
    FILES_MATCHING PATTERN "org.ssdut.plugin.helloworld.feature.*.h"
)
INSTALL(
    DIRECTORY qrc/
    DESTINATION share/translations/zh_CN
    FILES_MATCHING PATTERN "*_zh_CN.qm"
    PATTERN ".svn" EXCLUDE
)
INSTALL(

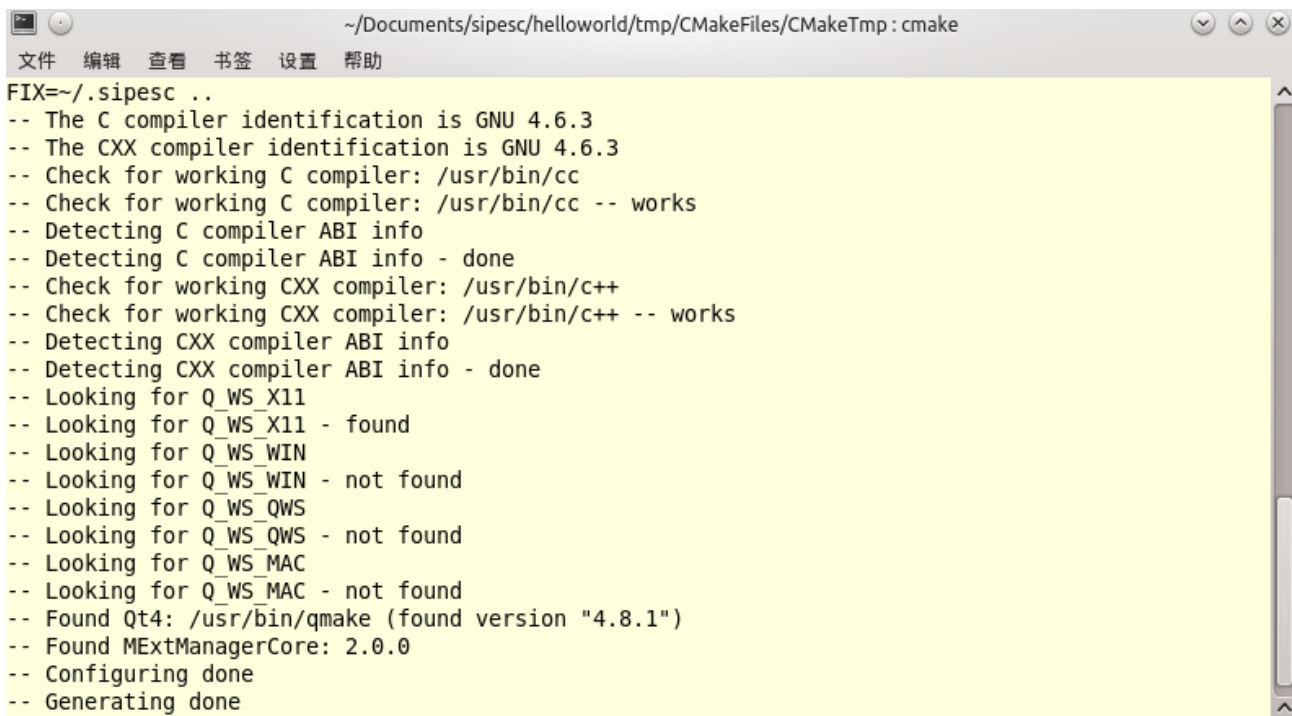
```

```
FILES $
{CMAKE_CURRENT_BINARY_DIR}/rg.ssdut.plugin.helloworld.features
DESTINATION share/features
)
```

在 helloworld 目录下新建一个目录 tmp，并在终端 进入 tmp 目录，执行如下命令：

```
$cd tmp
$cmake -DCMAKE_BUILD_TYPE=debug
-DCMAKE_INSTALL_PREFIX=~/.sipesc ..
```

命令效果如图：

A screenshot of a terminal window titled "~/Documents/sipesc/helloworld/tmp/CMakeFiles/CMakeTmp : cmake". The window shows the output of a CMake configuration command. The output includes compiler identification for GNU 4.6.3, checks for working C and CXX compilers, detection of compiler ABI info, and a search for Qt4 and MExtManagerCore. The search for Qt4 is successful, finding version 4.8.1. The search for MExtManagerCore is also successful, finding version 2.0.0. The configuration is complete, and the generation is done.

```
~/Documents/sipesc/helloworld/tmp/CMakeFiles/CMakeTmp : cmake
文件 编辑 查看 书签 设置 帮助
FIX=~/.sipesc ..
-- The C compiler identification is GNU 4.6.3
-- The CXX compiler identification is GNU 4.6.3
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Looking for Q_WS_X11
-- Looking for Q_WS_X11 - found
-- Looking for Q_WS_WIN
-- Looking for Q_WS_WIN - not found
-- Looking for Q_WS_QWS
-- Looking for Q_WS_QWS - not found
-- Looking for Q_WS_MAC
-- Looking for Q_WS_MAC - not found
-- Found Qt4: /usr/bin/qmake (found version "4.8.1")
-- Found MExtManagerCore: 2.0.0
-- Configuring done
-- Generating done
```

继续执行以下命令安装插件：

```
$make install
```

3、编写测试程序

在 helloworld 目录下创建测试工程目录：test

在 test 中创建以下目录：inc, src, tmp

将 helloworld/tmp/plugin 目录下的两个头文件复制到 test/inc 目录下

在 test/src 目录下新建源文件 main.cxx，代码如下：

```
#include <qapplication.h>
#include <mpluginmanager.h>
#include <mextensionmanager.h>

#include <org.ssdut.plugin.helloworld.feature.messageprovider.h>
```

```

#include <org.ssdut.plugin.helloworld.feature.messagereceiver.h>

using namespace org::ssdut::plugin::helloworld::feature;

int main(int argc, char *argv[]){

    QApplication app(argc, argv);
    MPluginManager::initialize(false);

    //获得插件管理器
    MPluginManager pmanager = MPluginManager::getManager();
    Q_ASSERT(!pmanager.isNull());
    bool ok = pmanager.loadAllPlugins();
    Q_ASSERT(ok);

    //获得全局扩展管理器
    MExtensionManager extManager = MExtensionManager::getManager();
    Q_ASSERT(!extManager.isNull());

    QString name = "org.ssdut.plugin.helloworld.feature.MessageProvider";
    //获得 provider 扩展
    MessageProvider provider = extManager.createExtension(name);

    //获得 receiver 扩展
    name = "org.ssdut.plugin.helloworld.feature.MessageReceiver";
    MessageReceiver receiver = extManager.createExtension(name);

    if(provider.isNull() || receiver.isNull()){
        return 0;
    }

    //从 provider 获得信息，并使用 receiver 显示。
    receiver.showMessage(provider.getMessage());

    return 0;
}

```

在 test 目录下新建 CMakeLists.txt，具体代码如下：

```

PROJECT(TEXT_HELLO)
SET(_version, 1.0.0)
SET(_soversion, 1.0.0)

CMAKE_MINIMUM_REQUIRED(VERSION 2.6.0)
IF(COMMAND CMAKE_POLICY)
    CMAKE_POLICY(SET CMP0003 NEW)
ENDIF(COMMAND CMAKE_POLICY)

FIND_PACKAGE(Qt4 REQUIRED)
FIND_PACKAGE(MExtManagerCore REQUIRED)

SET(QT_USE_QTTEST 1)

```



```
INCLUDE(${QT_USE_FILE})
INCLUDE_DIRECTORIES(
    ${CMAKE_CURRENT_BINARY_DIR}
    ${CMAKE_CURRENT_SOURCE_DIR}/inc
    ${MEXTMGR_CORE_INCLUDE_DIRS}
)

SET(
    src_file
    src/main.cxx
)

ADD_EXECUTABLE(
    main
    ${src_file}
)

TARGET_LINK_LIBRARIES(
    main
    ${QT_LIBRARIES}
    ${MEXTMGR_CORE_LIBRARIES}
)
```

从终端进入 test/tmp 目录，执行。

```
$cmake -DCMAKE_BUILD_TYPE=debug
-DCMAKE_INSTALL_PREFIX=~/.sipesc ..
$make
$./main
```

正确运行时输出:Hello world Plugin from Xiaodong.