目　　录

HelloWorld插件设计向导

1.1 概述

我们准备编写一个插件，在控制台中运行时，输出“Hello World!”信息。这个插件具有两个扩展，MessageProvider 和 MessageReceiver。MessageProvider 扩展包含一个方法 getMessage()，该方法提供“Hello World ！”字符串；扩展 MessageReceiver 也包含一个方法 showMessage()，该方法在控制台输出信息。

要实现该插件，我们可以使用插件接口设计器或者通过 SiPESC 平台建立插件项目，设计出 MessageProvider 和 MessageReciever 这两个扩展，以及它们各自包含的方法，getMessage() 和 showMessage()，编译链接，生成插件，再将该插件安装到平台中。然后写一个脚本文件，调用该插件，实现在控制台中输出“Hello World!”信息的功能。

注意 1：插件本身是不能运行的，我们需要在 SiPESC 平台上调用这个插件。另外，上面说的扩展，其实就是 C++中的类，而方法则是类的接口函数。

注意 2：必须在环境变量中添加 HOME 变量。

1.2 设计插件

插件的设计的方法两种，一种是通过插件接口设计器，另一种是通过在 SiPESC 平台上建立插件项目，接下来，详细介绍这两种方法。

1.2.1 使用插件接口设计器设计插件

1.准备工作。

事先参照相关文档，安装、配置 SiPESC 平台。创建 HelloWorld 插件的工作目录，示例代码中的工作目录为~/workspaces/workspace_sipesc/example_helloworld，其中~代表用户目录。在 example_helloworld 文件夹下创建 doc 文件夹。

注：为了方便起见，后面以 example_helloworld 为当前工作目录进行叙述。

打开插件设计器，如图 1。

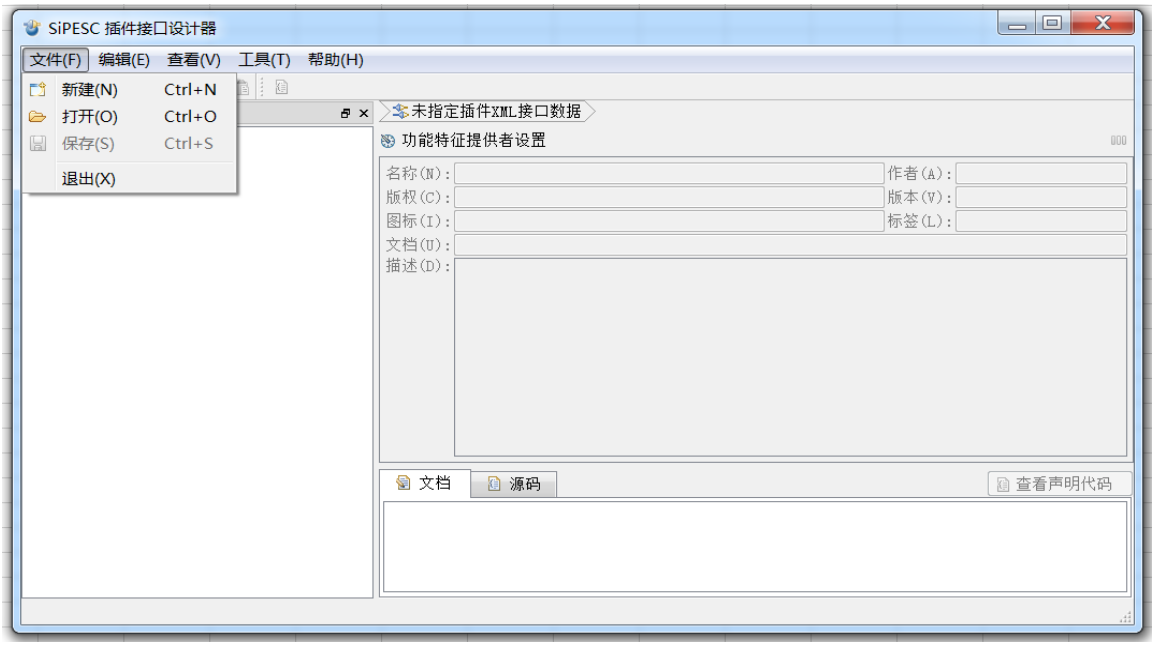

图1

点击工具栏上的“新建”按钮，按照图 2 填写完整各项信息。点击“保存”，将插件设计文件保存到 example_helloworld/doc 文件夹中。

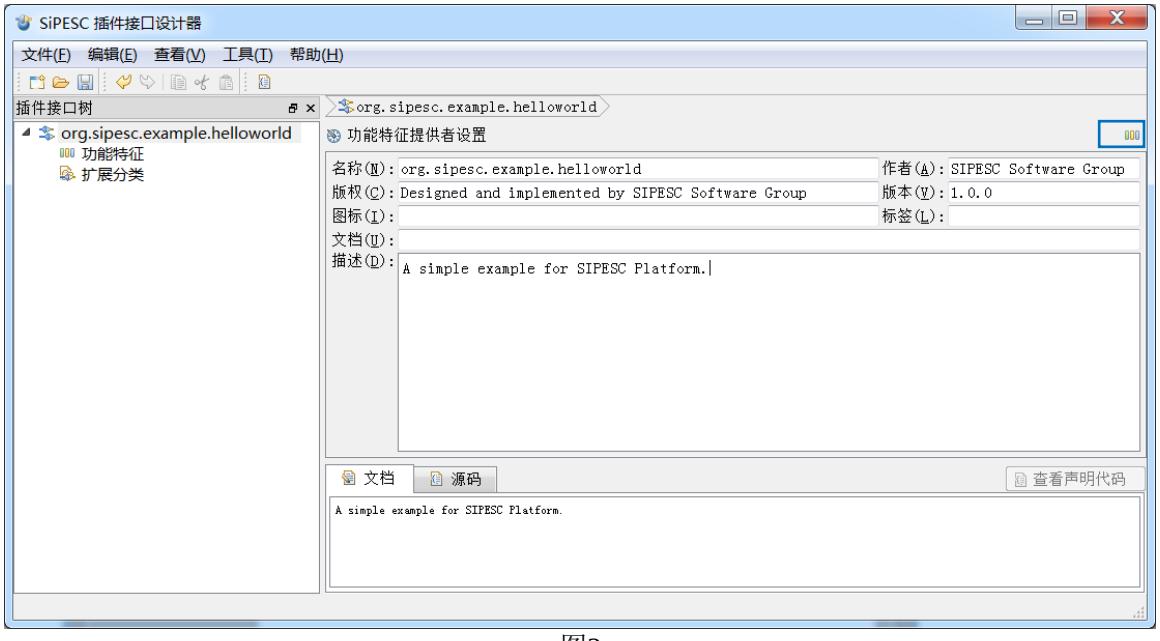

图2

注意：选定插件设计文件的保存目录，在弹出的对话框中填写文件名称，建议此处文件名称与插件的名称保持一致，否则在插件后续编译使用过程中可能会出错，例如这里命名（org.sipesc.example.helloworld），保存类型应为*.sipplugin，如图 3。

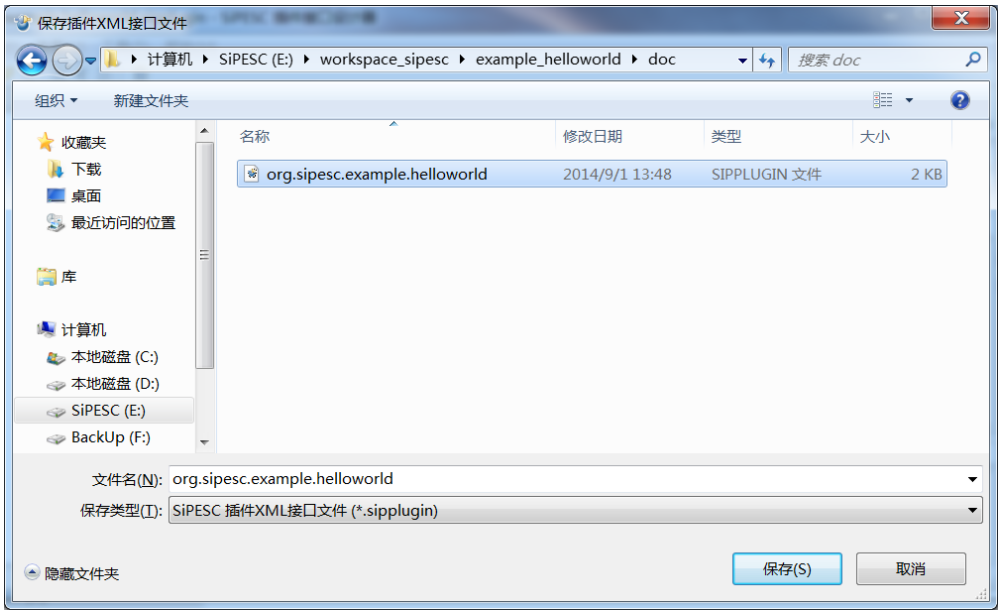

图3

2．如图 2，点击 “功能特征列表”按钮，弹出功能特征列表图（图 4）。

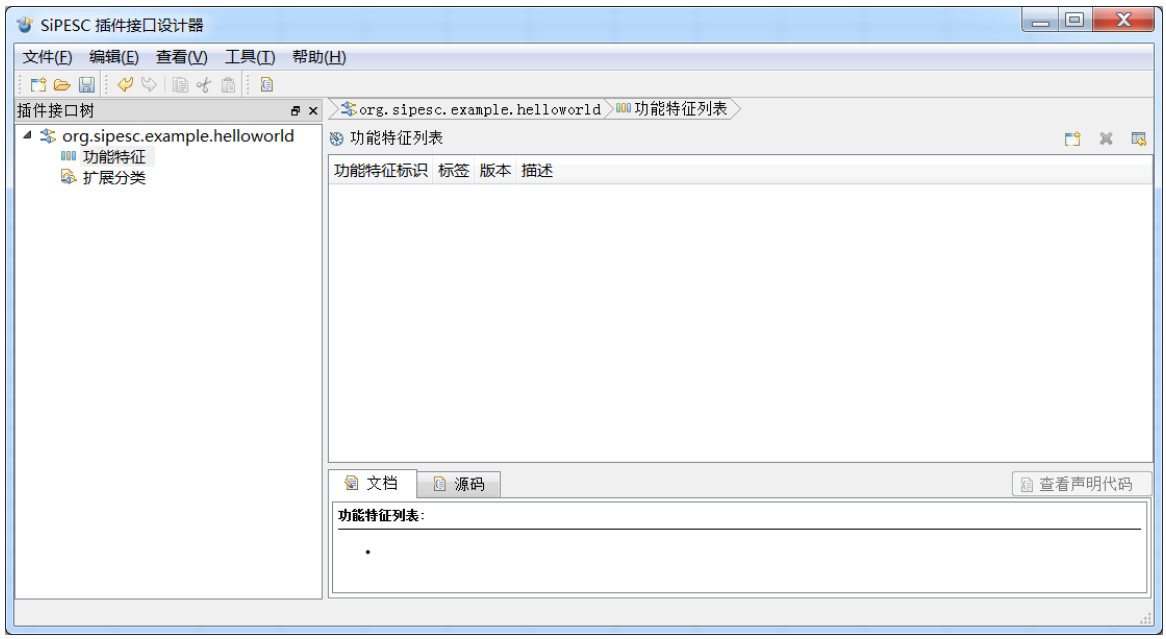

图4

3．功能特征列表上，点击 “添加功能特征”按钮，在功能特征设置中，添加功能特征信息，如图 5。

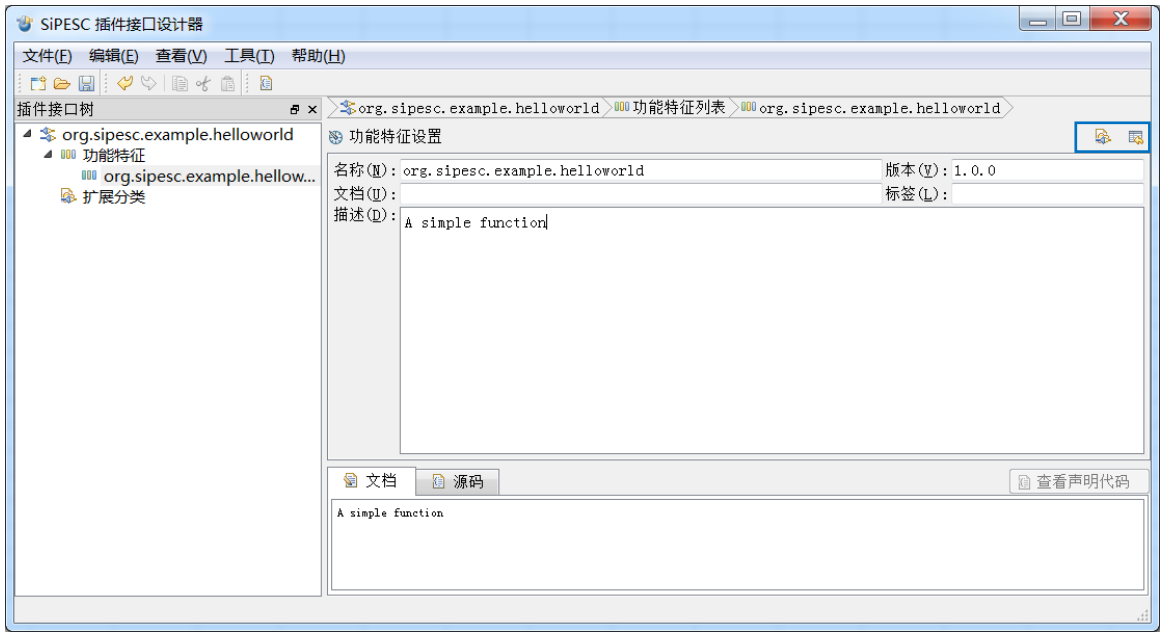

图5

4.点击“扩展列表”按钮，进入扩展列表页面，点击“添加扩展”，创建 MessageProvider 扩展。按照图 6 填写扩展设置页面里的名称、分类和描述信息，选中“可创建”复选框。

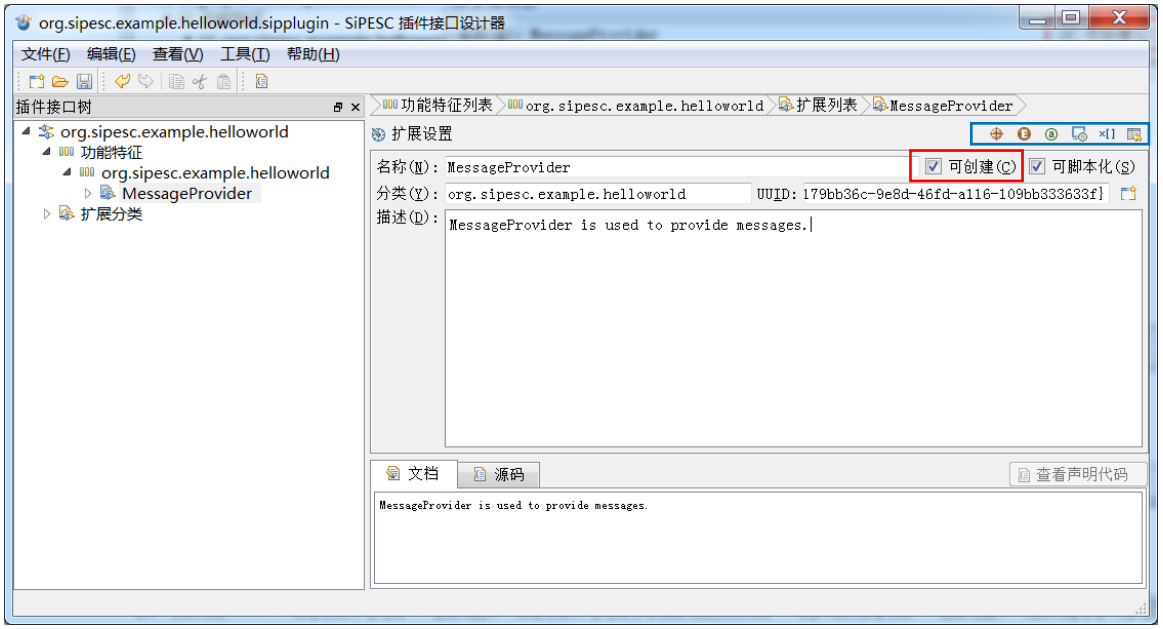

图6

5. 点击“属性列表”按钮，属性列表页面里点击“添加属性”按钮，按照图 7 填写属性设置页面里的名称、类型、默认和描述信息，注意“只读”。至此，MessageProvider 扩展

设计完成。系统自动为每个属性创建一个 get 方法，用于返回该属性的值。这里自动创建 message 的 getMessage()方法。

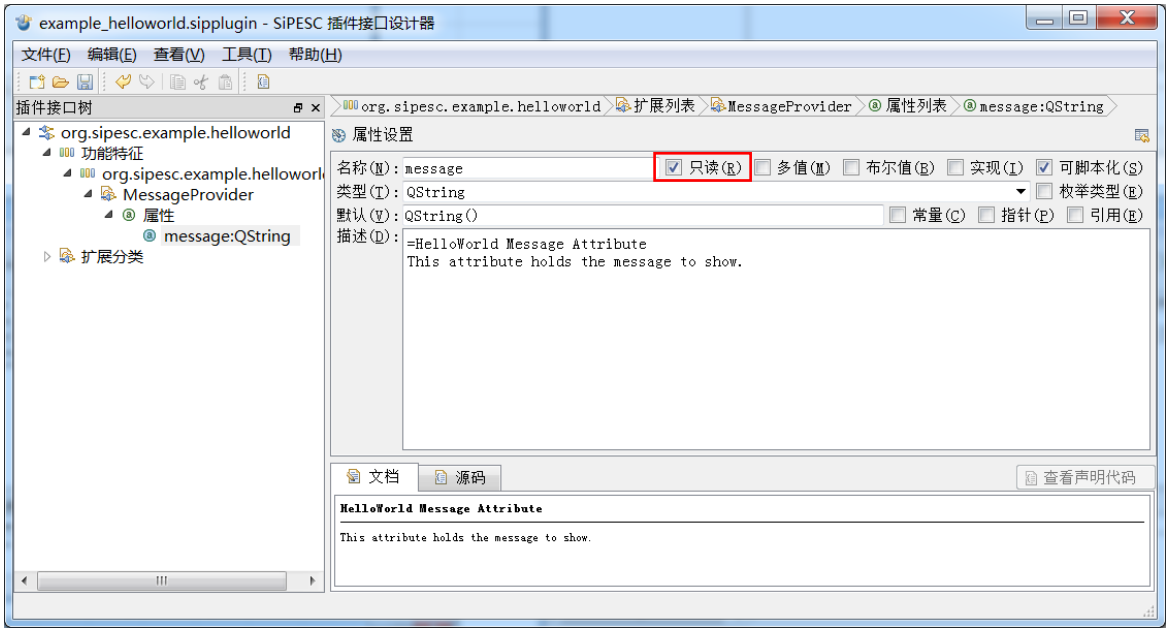

图7

6. 返回到扩展列表页面，如图 8，点击 “添加扩展”按钮，创建 MessageReceiver 扩展。

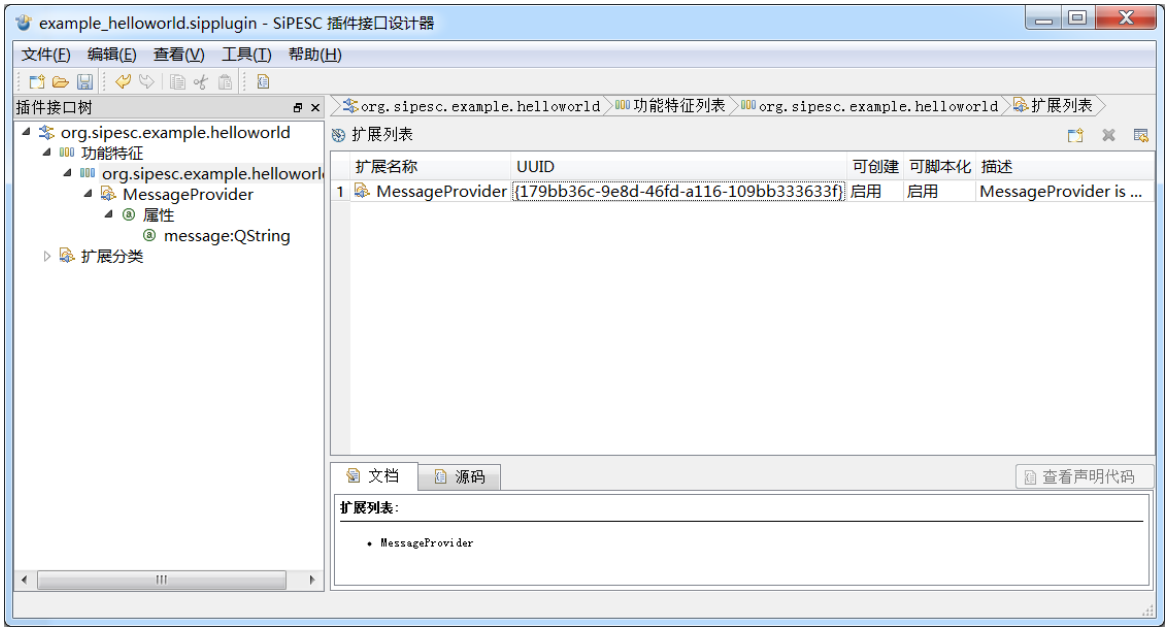

图8

按照图 9 填写扩展设置页面里的扩展名称，分类和描述信息，选中“可创建”复选框。

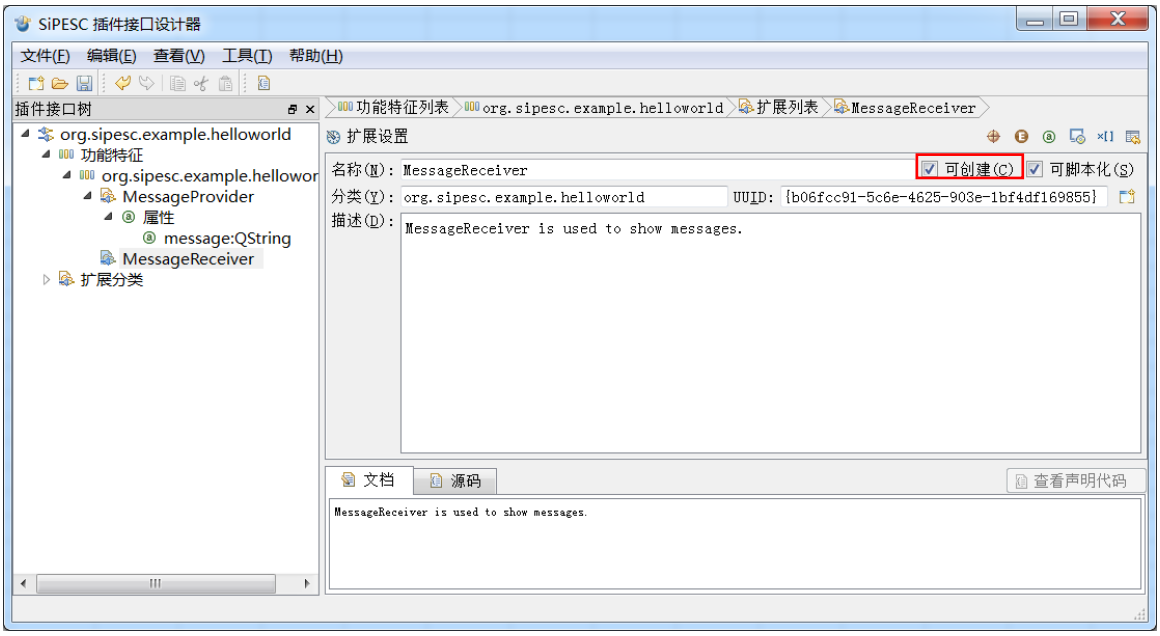

图9

7. 点击方法列表页面里的创建方法按钮，创建 showMessage()方法。按照图 10 填写方法设置页面里的名称和描述，选中“常量方法”复选框。

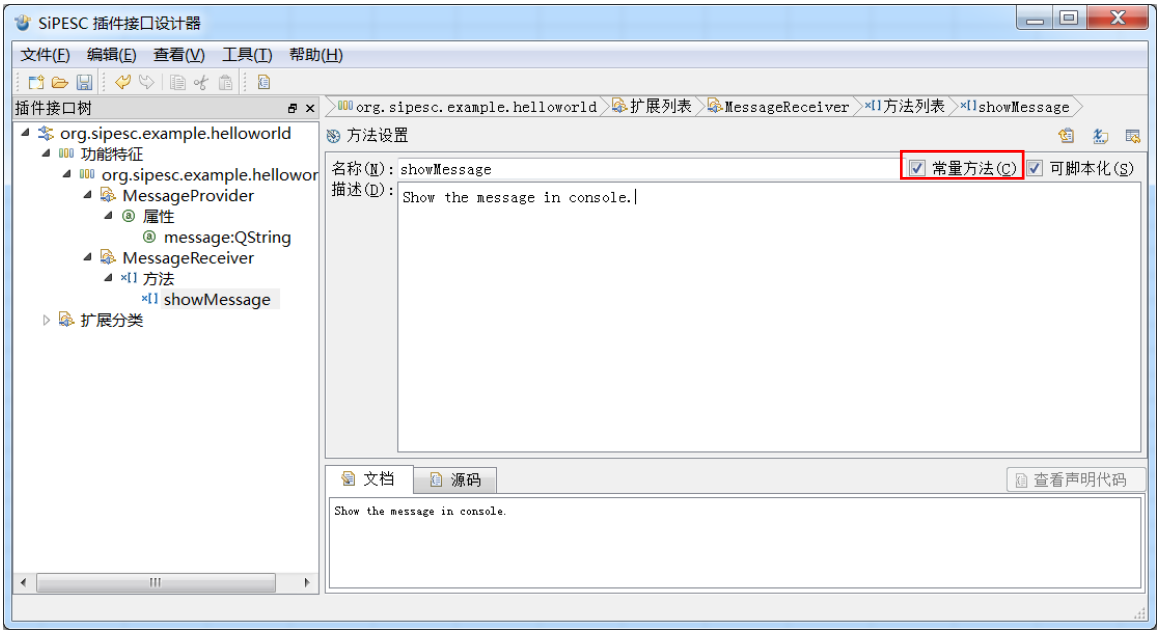

图10

该方法没有返回值，但是有参数。点击“参数列表”按钮，参数列表页面点击“添加参数”按钮，按照图 11 填写名称，类型等信息。MessageReceiver 扩展设计完成，注意保存。

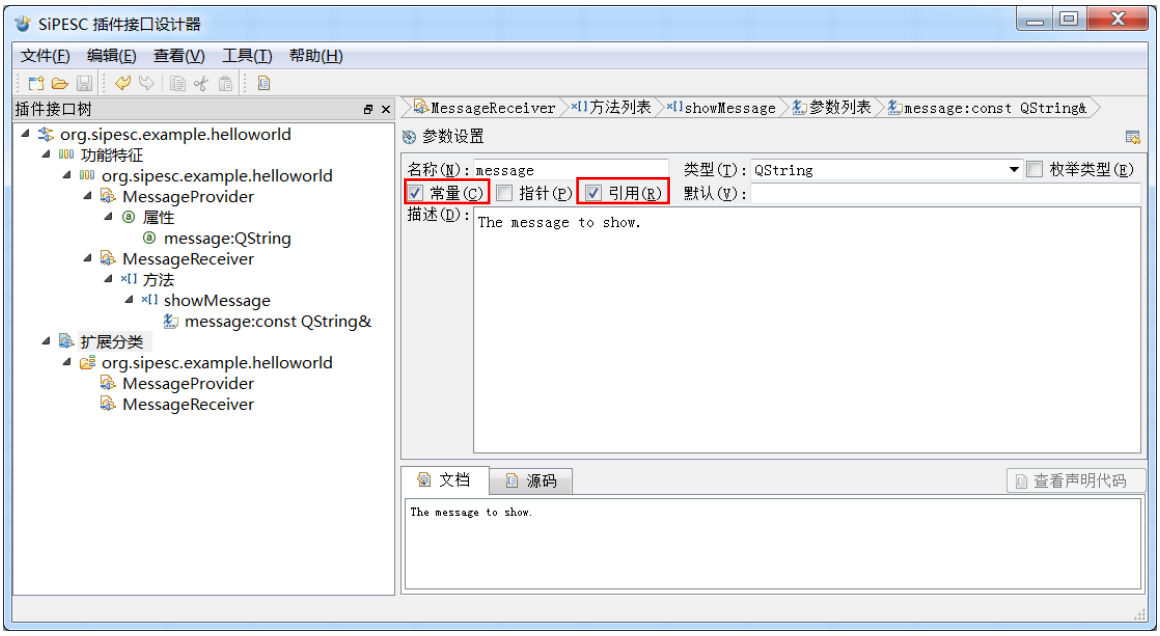

图11

8. 现在我们已经完成 HelloWorld 插件的设计工作，点击工具栏从插件信息生成代码按钮，按照图 12 填写信息（工程路径设为~/workspaces/workspace_sipesc/example_helloworld）。点击“生成代码”按钮。

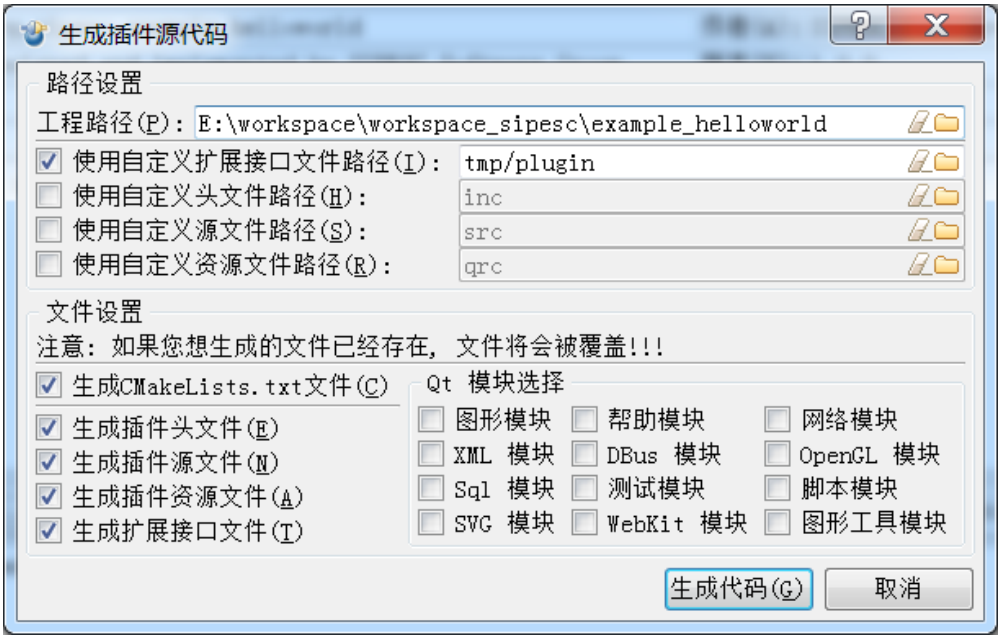

图12

注：

(1) 需要改变生成路径时，选中其复选框，在可编辑栏指定路径

(2) 根据需要，在“文件设置”中选择生成的文件

(3) Qt 功能模块选择群组框里面的选项，与 Qt 的各个模块对应。例如，我们的插件中用到 Qt 的界面模块，就需要选中“图形模块”复选框。

(4) 由于平台版本的原因，执行“生成插件源代码”后，需要在 doc 文件夹下新建名为 design 的文件夹，并把后缀.sipplugin 的插件原文件剪切到 design 中。

1.2.2 通过 SiPESC 平台设计插件

1. 打开 SiPESC 平台，选择 “插件项目”，建立 HelloWorld 插件项目，如图 13。

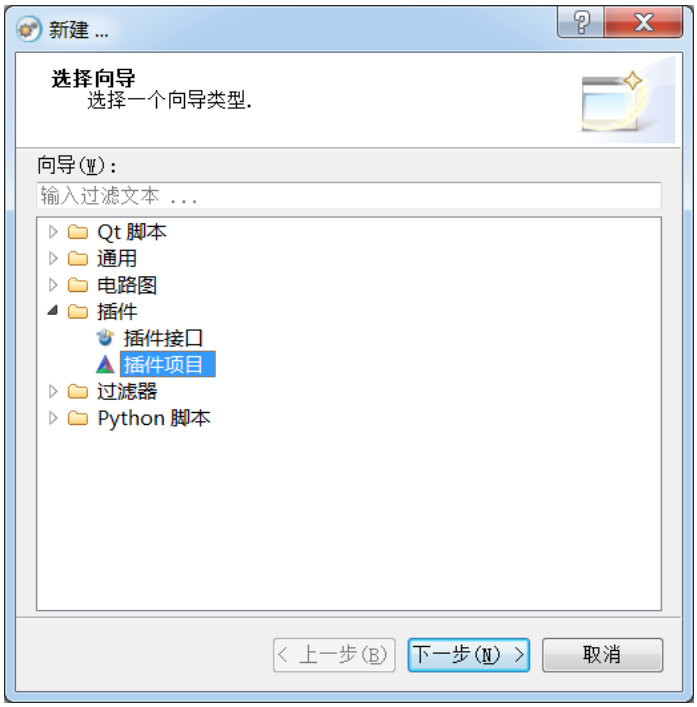

图13

2. 如图 14，添加插件名称、标签等信息。点击“下一步”。

注：“项目目录”要选择在 SiPESC 平台的工作空间下的某个目录，否则会出错。

按照图 15，选择插件包含的模块资源。例如，如果我们的插件中用到 Qt 的界面模块，就需要选中“使用 Qt 图形模块”复选框。

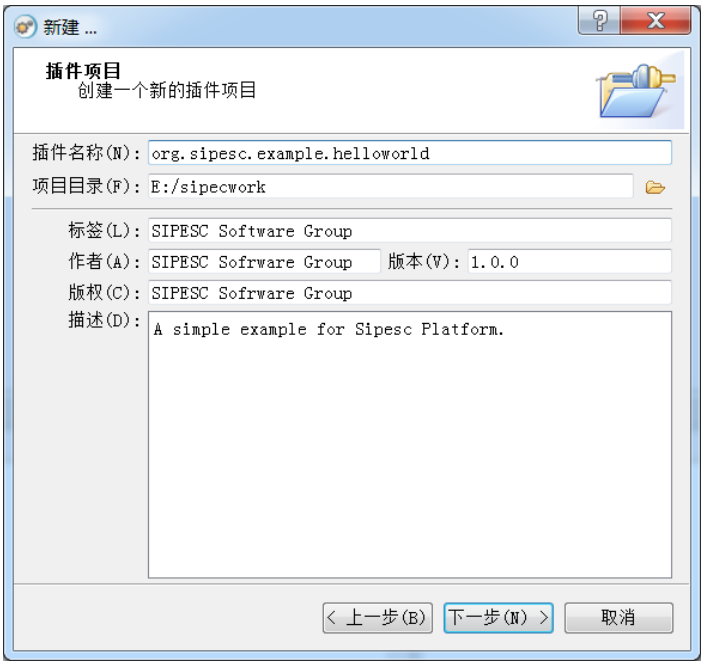

图14

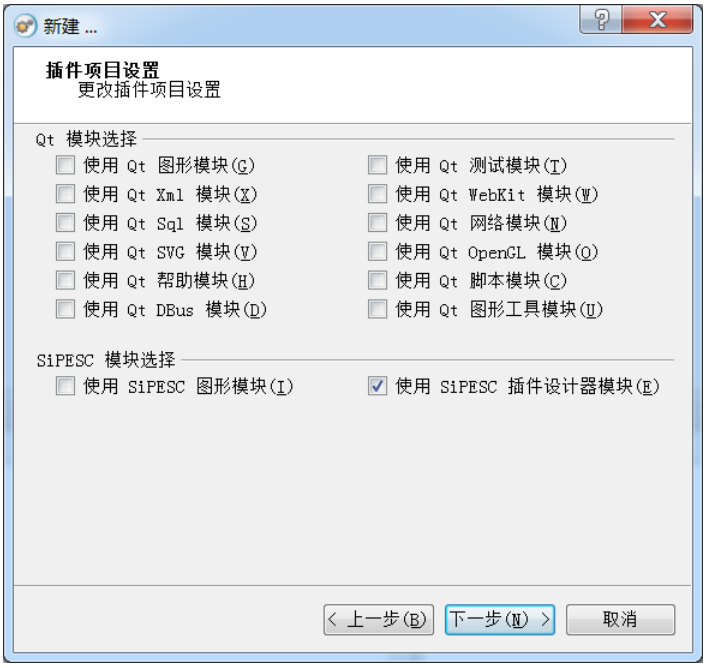

图15

3. 配置插件项目，如图 16。

注：安装目录，即插件生成后安装的位置，选择用户目录，例如：C：/User/jifx/.sipesc。构建类型，即生成插件的版本。调试版本对应的是 debug 版，发行版本对应 release 版。

点击“下一步”，系统构建工程，生成 CMakeLists 等文件，如图 17，点击“完成”。

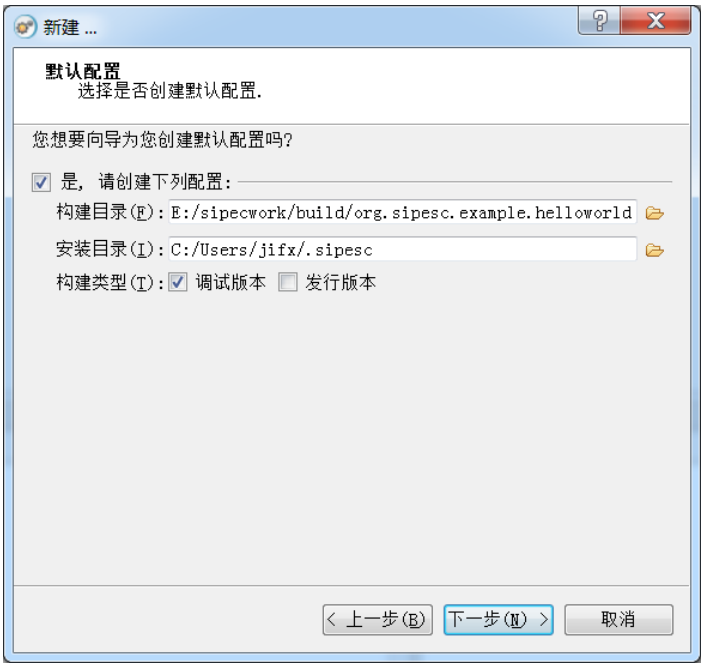

图16

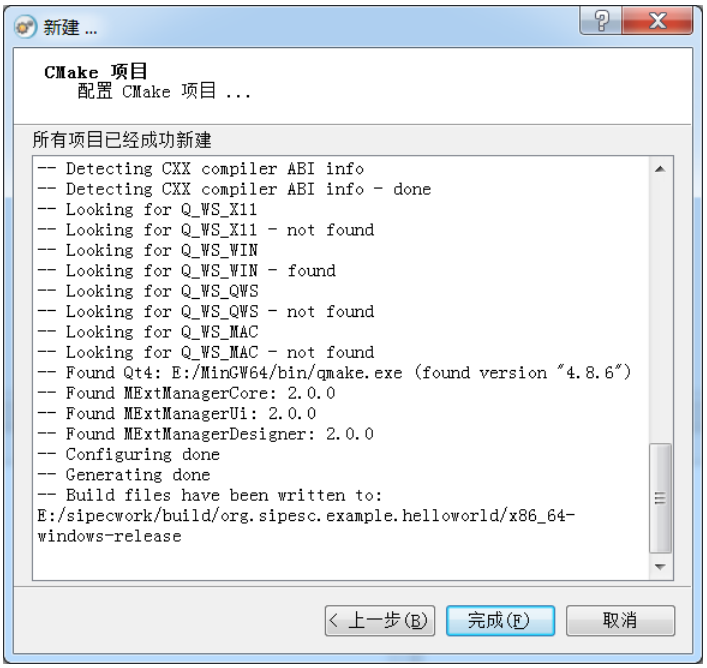

图17

4. 至此，插件项目建立完成，如图 18，在项目浏览器中，打开 org.sipesc.example.helloworld 插件，对 HelloWorld 插件进行设计。

接下来的步骤如图 2，与使用“插件接口设计器”设计插件相同，这里不再赘述。

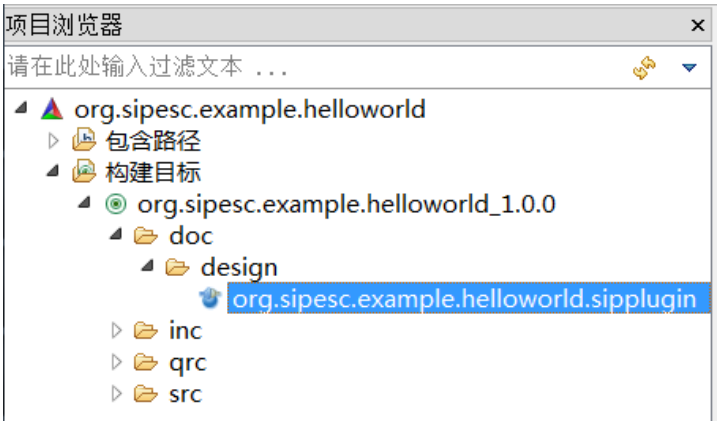

图18

5. 生成源代码文件

完成 HelloWorld 插件的设计工作后，点击平台工具栏“生成源代码文件”，如图 19、图 20，配置生成代码信息。点击“生成代码”。

注：其中功能特征选项卡中添加的是目标插件所依赖的功能特征，例如依赖数据库，可添加“org.sipesc.core.engdbs”。HelloWorld 插件没有依赖的功能特征，所以不需要添加。

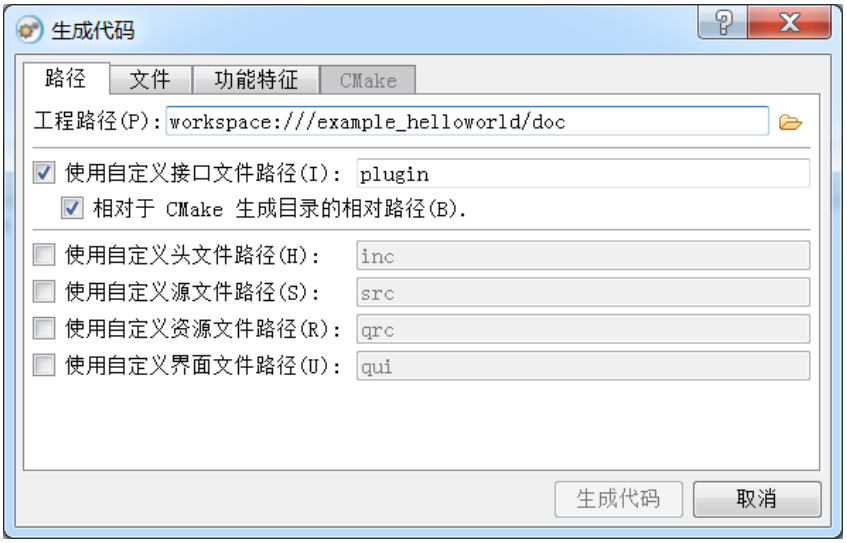

图19

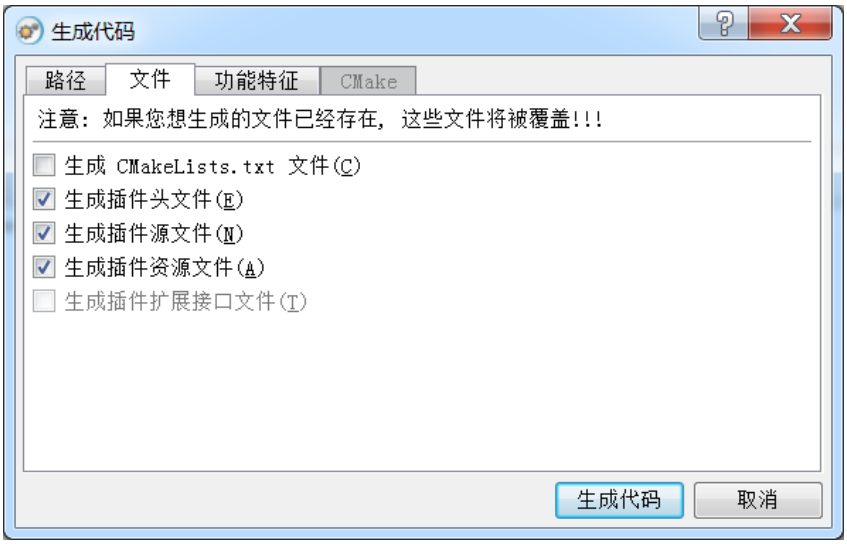

图20

1.3 实现插件

使用“插件接口设计器”生成代码之后，默认会在 example_helloworld 目录下生成 inc，src，tmp 和 qrc 四个文件夹，以及一个 CMakeLists.txt 文件（这些跟图 12 中的各个文件的路径相对应）。

同样，通过 SiPESC 平台“插件项目”生成代码之后，默认会在 org.sipesc.example.helloworld 目录下生成 inc，src 文件夹，相对应的构建文件，则生成在“构建目录”（图 17）目录下。

inc 文件夹中包含一个插件接口头文件 org.sipesc.example.helloworld.h，src 文件夹中包含一个插件接口源文件 org.sipesc.example.helloworld.cxx，在 tmp/plugin 文件夹下，有两个扩展接口文件 org.sipesc.example.helloworld.messageprovider.h 和 org.sipesc.example.helloworld.messagereceiver.h。

注：生成代码后，使用 QtCreator 导入刚生成的 cmake 项目，新建文件操作使用 QtCreator 的“新建-〉C++-〉Class”。这么做可以保证项目中文件的编码为 UTF-8，否则会出现乱码问题。

1.3.1 实现扩展方法

a. 实现 MessageProvider 扩展接口

我们可以看到，在 org.sipesc.example.helloworld.messageprovider.h 扩展接口文件中，类 MessageProviderInterface 包含一个虚方法 getMessage()，这就是使用插件设计器设计出来的接口。在 inc 文件夹下新建一个头文件，命名为 messageproviderimpl.h，创建 MessageReceiverImpl 类，该类继承 MessageProviderInterface 类，实现 getMessage()方法。具体代码如下：

```
#ifndef MESSAGEPROVIDERIMPL_H_
#define MESSAGEPROVIDERIMPL_H_

#include <org.sipesc.example.helloworld.messageprovider.h>
/**
 * 命名空间。
 */
using namespace org::sipesc::example::helloworld;

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

#endif /* MESSAGEPROVIDERIMPL_H_ */
```

在 src 文件夹下新建一个源文件，命名为 messageproviderimpl.cxx，具体代码如下：

```
#include <messageproviderimpl.h>

MessageProviderImpl::MessageProviderImpl()
{
    _message="Hello World!";
}
MessageProviderImpl::~MessageProviderImpl()
{
}
QString MessageProviderImpl::getMessage() const
{
    return _message;
}
```

b.实现 MessageReceiver 扩展接口

在 org.sipesc.example.helloworld.messagereceiver.h 扩展接口文件中，类 MessageReceiverInterface 包含一个虚方法 showMessage()，也是使用插件设计器设计出来的接口。在 inc 文件夹下新建一个头文件，命名为 messagereceiverimpl.h，创建 MessageReceiverImpl 类，该类继承 MessageReceiverInterface 类，实现 showMessage()方法，具体代码如下：

```
#ifndef MESSAGERECEIVERIMPL_H_
#define MESSAGERECEIVERIMPL_H_
```

```
#include <org.sipesc.example.helloworld.messagereceiver.h>
/**
 * 命名空间。
 */
using namespace org::sipesc::example::helloworld;
class MessageReceiverImpl:public MessageReceiverInterface
{
public:
    MessageReceiverImpl();
    ~MessageReceiverImpl();
public:
    void showMessage(const QString& message) const;
};

#endif /* MESSAGERECEIVERIMPL_H_ */
```

在 src 文件夹下新建一个源文件，命名为 messagereceiverimpl.cxx，具体代码如下：

```
#include <qtextstream.h>
#include <messagereceiverimpl.h>
#include <QDebug>

MessageReceiverImpl::MessageReceiverImpl()
{
}
MessageReceiverImpl::~MessageReceiverImpl()
{
}
void MessageReceiverImpl::showMessage(const QString& message) const
{
    qDebug()<<message;
}
```

1.3.2 实现插件接口

example_helloworld/src/文件夹中的插件接口文件 org.sipesc.example.helloworld.cxx，需要修改的地方有：添加头文件#include <messageproviderimpl.h>和#include <messagereceiverimpl.h>，修改 createOrgSipescExampleHelloworldMessageReceiver()接口和 createOrgSipescExampleHelloworldMessageProvider()接口。具体代码如下：

```
#include <org.sipesc.example.helloworld.h>
#include <messageproviderimpl.h>
#include <messagereceiverimpl.h>

OrgSipescExampleHelloworldPlugin::OrgSipescExampleHelloworldPlugin()
  {
```

```
  _running=false;
  }

OrgSipescExampleHelloworldPlugin::~OrgSipescExampleHelloworldPlugin()
  {
  }

bool OrgSipescExampleHelloworldPlugin::initialize()
  {
  return true;
  }

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

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

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

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

MExtensionObject* OrgSipescExampleHelloworldPlugin
::createOrgSipescExampleHelloworldMessageProvider()
  {
  //return 0;
  return new MessageProviderImpl;
  }
MExtensionObject* OrgSipescExampleHelloworldPlugin
::createOrgSipescExampleHelloworldMessageReceiver()
  {
  //return 0;
  return new MessageReceiverImpl;
  }
```

注：插件接口文件 org.sipesc.example.helloworld.cxx 中的函数 getRequiredFeatures()是用来获得需要初始化的功能特征，HelloWorld 插件设计过程中并没有用到其他插件的内容，因此，

上述例子函数中没有添加功能特征。如果在插件设计过程中用到其他插件的内容，需要将用到的插件功能特征添加到 QStringList 中并返回，以便系统对其进行初始化，否则会报错。例如：

```
QStringList OrgSipescExampleHelloworldPlugin::
getRequiredFeatures() const
  {
    QStringList RequiredFeatures;  //QList<QString>
    RequiredFeatures << "org.sipesc.core.application";    //功能特征名
    RequiredFeatures << "org.sipesc.experiments";
    RequiredFeatures << "org.sipesc.utility.table";
    return RequiredFeatures;
  }
```

1.3.3 修改 CMakeList.txt 文件

通过“插件接口设计器”设计的 HelloWorld 插件，需要手动修改 CMakeList 文件，而通过 Sipesc 平台设计的插件，只需要“添加包含文件”和“添加源码文件”，对应的 CMakeList 文件将自动被修改。

a. 手动修改 CMakeList 文件部分如下：

SET(org_sipesc_example_helloworld_src_files
    src/messageproviderimpl.cxx
    src/messagereceiverimpl.cxx
    src/org.sipesc.example.helloworld.cxx
)
SET(org_sipesc_example_helloworld_hdr_files
    inc/messageproviderimpl.h
    inc/messagereceiverimpl.h
    inc/org.sipesc.example.helloworld.h
)

具体代码如下：

```
PROJECT(org.sipesc.example.helloworld)

CMAKE_MINIMUM_REQUIRED(VERSION 2.8.0)

IF(COMMAND CMAKE_POLICY)
    CMAKE_POLICY(SET CMP0003 NEW)
ENDIF(COMMAND CMAKE_POLICY)

SET(CMAKE_MODULE_PATH "${home_path}/.sipesc/share/cmake")

FIND_PACKAGE(Qt4 REQUIRED)
FIND_PACKAGE(MExtManagerCore REQUIRED)
```

```
FIND_PACKAGE(MExtManagerDesigner REQUIRED)

SET(QT_DONT_USE_QTGUI 1)
INCLUDE(${QT_USE_FILE})

INCLUDE_DIRECTORIES(
    ${CMAKE_CURRENT_BINARY_DIR}
    ${CMAKE_CURRENT_SOURCE_DIR}/inc
    ${CMAKE_CURRENT_BINARY_DIR}/../plugin
    ${MEXTMGR_CORE_INCLUDE_DIRS}
    ${MEXTMGR_DESIGNER_INCLUDE_DIRS}
)

SET(org_sipesc_example_helloworld_src_files
    src/org.sipesc.example.helloworld.cxx
    src/messageproviderimpl.cxx
    src/messagereceiverimpl.cxx
)

SET(org_sipesc_example_helloworld_hdr_files
    inc/org.sipesc.example.helloworld.h
    inc/messageproviderimpl.h
    inc/messagereceiverimpl.h
)

MEXTMGR_ADD_RESOURCES(
    org_sipesc_example_helloworld_qrc_files
    qrc/org.sipesc.example.helloworld.qrc
)

SET(org_sipesc_example_helloworld_plg_file
    doc/design/org.sipesc.example.helloworld.sipplugin
)

MEXTMGR_GEN_PLUGINHEADER(
    org_sipesc_example_helloworld_phd_files
    ${org_sipesc_example_helloworld_plg_file}
    ${CMAKE_CURRENT_BINARY_DIR}/../plugin
)

MEXTMGR_GEN_PLUGINFEATURE(
    org_sipesc_example_helloworld_pfm_files
    ${org_sipesc_example_helloworld_plg_file}
)

MEXTMGR_WRAP_XML(
    org_sipesc_example_helloworld_xml_files
    ${org_sipesc_example_helloworld_plg_file}
)

MEXTMGR_ADD_TRANSLATION(
    org_sipesc_example_helloworld_qms_zh_cn_files
    ${CMAKE_CURRENT_BINARY_DIR}
```

```
    ${CMAKE_CURRENT_SOURCE_DIR}/inc
    ${CMAKE_CURRENT_BINARY_DIR}/../plugin
    ${MEXTMGR_CORE_INCLUDE_DIRS}
    ${MEXTMGR_DESIGNER_INCLUDE_DIRS}
    "qrc/org.sipesc.example.helloworld_zh_CN.ts"
    ${org_sipesc_example_helloworld_src_files}
    ${org_sipesc_example_helloworld_hdr_files}
    ${org_sipesc_example_helloworld_xml_files}
)

ADD_LIBRARY(
    org.sipesc.example.helloworld_1.0.0 SHARED
    ${org_sipesc_example_helloworld_src_files}
    ${org_sipesc_example_helloworld_hdr_files}
    ${org_sipesc_example_helloworld_qrc_files}
    ${org_sipesc_example_helloworld_phd_files}
    ${org_sipesc_example_helloworld_pfm_files}
    ${org_sipesc_example_helloworld_xml_files}
    ${org_sipesc_example_helloworld_qms_zh_cn_files}
)

SET_TARGET_PROPERTIES(
    org.sipesc.example.helloworld_1.0.0 PROPERTIES
    SUFFIX ".sep"
    PREFIX ""
    AUTOMOC "1"
    POSTFIX ""
)

TARGET_LINK_LIBRARIES(
    org.sipesc.example.helloworld_1.0.0
    ${QT_LIBRARIES}
    ${MEXTMGR_CORE_LIBRARIES}
    ${MEXTMGR_DESIGNER_LIBRARIES}
)

INSTALL(DIRECTORY
    "qrc/"
    DESTINATION "share/translations/zh_CN"
    FILES_MATCHING PATTERN "*_zh_CN.qm"
    PATTERN ".svn" EXCLUDE
)

INSTALL(FILES
    "${CMAKE_CURRENT_BINARY_DIR}/org.sipesc.example.helloworld.features"
    DESTINATION "share/features"
)

INSTALL(TARGETS
    "org.sipesc.example.helloworld_1.0.0"
    ARCHIVE DESTINATION "lib/plugins/${CMAKE_BUILD_TYPE}"
    LIBRARY DESTINATION "lib/plugins/${CMAKE_BUILD_TYPE}"
    RUNTIME DESTINATION "lib/plugins/${CMAKE_BUILD_TYPE}"
```

)

b. SiPESC 添加 HelloWorld 插件包含文件和源码文件操作如下：

如图 21、图 22，在构建目标名称上，右键选择“添加包含文件”，在弹出的对话框中选中刚刚建立的头文件（messageproviderimpl.h 和 messagereceiverimpl.h），点击“确定”，完成添加。同样的方式“添加源码文件”（messageproviderimpl.cxx 和 messagereceiverimpl.cxx）。

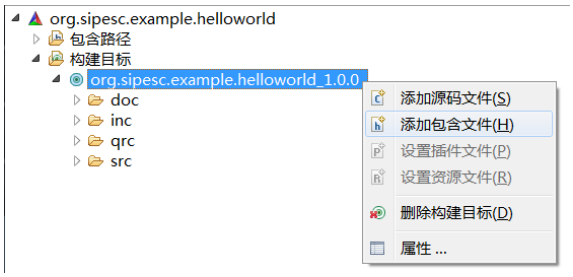

图21

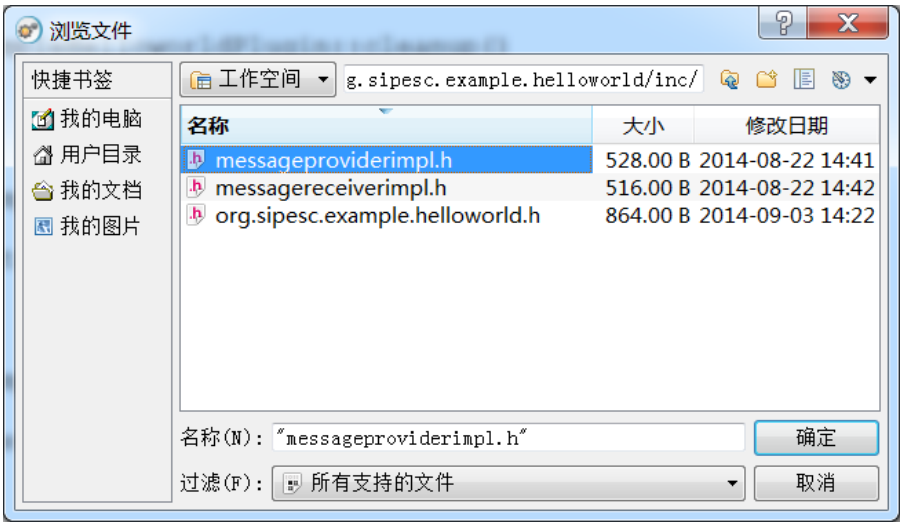

图22

1.4 编译链接，安装插件

至此，插件代码实现部分已经完成，下一步就是编译链接。插件实质上是一个动态链接库，编译链接时有两个版本可供选择，debug 版和 release 版，这里我们已 debug 版为例。

对于插件的编译和安装主要有两种方式，分别是通过命令行编译安装和通过 SiPESC 平台编译安装。

1.4.1 通过命令行编译、安装插件

命令行编译、安装插件，主要是针对于通过“插件接口设计器”设计生成的插件。具体的实现方法如下：

(1) Linux 操作系统下，从控制台进入到 example_helloworld/tmp 目录下，

编译 debug 版插件输入：cmake -DCMAKE_BUILD_TYPE=debug ..（或者编译 release 版输入：cmake -DCMAKE_BUILD_TYPE=release ..）

若没有错误，再在控制台输入:

cmake -DCMAKE_INSTALL_PREFIX=$ENV{HOME}/.sipesc ..

或者直接在控制台输入：（如图 23）

cmake –DCMAKE_BUILD_TYPE=debug -DCMAKE_INSTALL_PREFIX=$ENV{HOME}/.sipesc ..

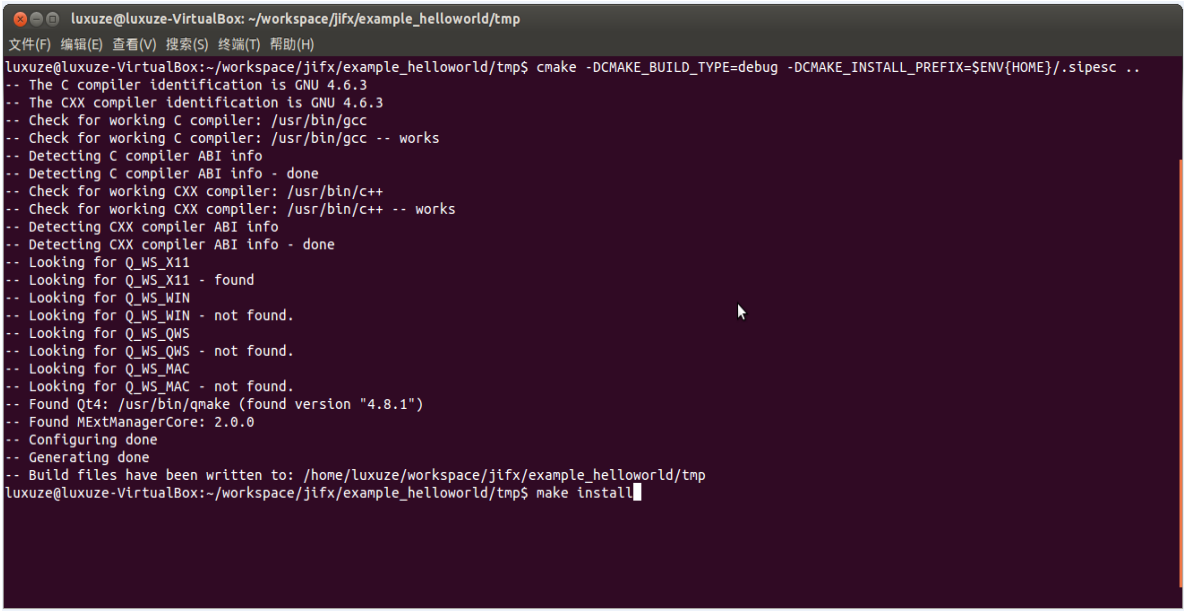

图23

在控制台输入:make install

若没有错误，

在~/example_helloworld/tmp/{HOME}/.sipesc/lib/plugins/debug 目录下会生成插件 org.sipesc.example.helloworld_1.0.0.sep。

在~/example_helloworld/tmp/{HOME}/.sipesc/share/feature 目录下会生成特征文件 org.sipesc.example.helloworld.features。

在~/example_helloworld/tmp/{HOME}/.sipesc/share/translations/zh_CN 目录下会翻译文件（用于国际化）org.sipesc.example.helloworld_zh_CN.qm。

(2) Windows 操作系统下，通过-G 指定编译器。

从开始菜单输入“cmd”进入控制台，再进入到 example_helloworld/tmp 目录下，编译 debug 版输入：cmake –G ”MinGW Makefiles” –DCMAKE_BUILD_TYPE=debug ..（或者编译 release 版输入：cmake –G ”MinGW Makefiles” –DCMAKE_BUILD_TYPE=release ..）

若没有错误，再在控制台输入:

cmake –G ”MinGW Makefiles” -DCMAKE_INSTALL_PREFIX=C:Users\jifx\.sipesc ..

或者直接在控制台输入：（如图 24）

cmake –G ”MinGW Makefiles” –DCMAKE_BUILD_TYPE=debug -DCMAKE_INSTALL_PREFIX=C:Users\jifx\.sipesc ..

注：这里的“jifx”是当前电脑的用户名。

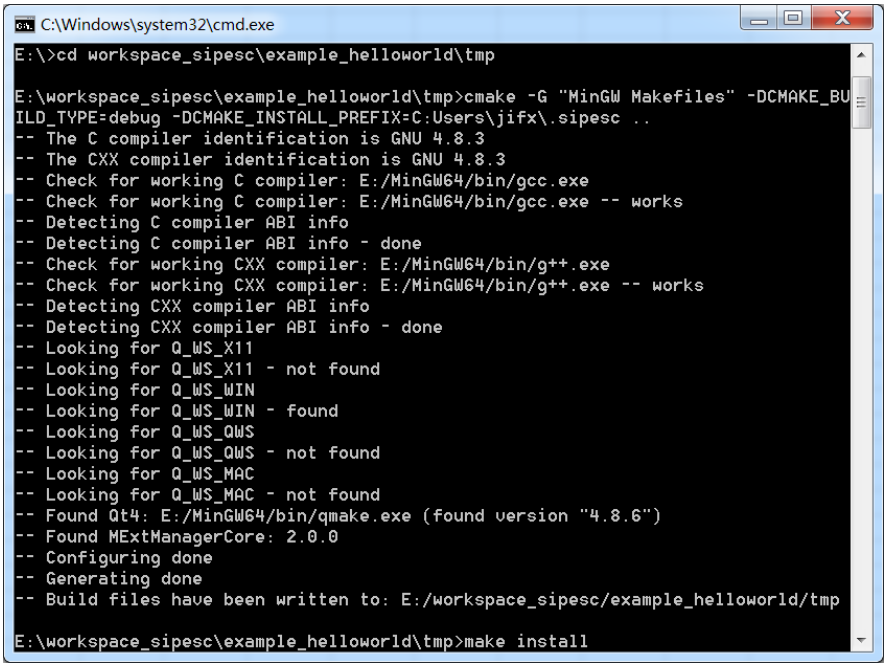

图24

在控制台输入:make install

若没有错误，

在 C:\Users\jifx\.sipesc\lib\plugins\debug 目录下会生成插件 org.sipesc.example.helloworld_1.0.0.sep。

在 C:\Users\jifx\.sipesc\share\features 目录下会生成特征文件 org.sipesc.example.helloworld.features。

在 C:\Users\jifx\.sipesc\share\translations\zh_CN 目录下会生成翻译文件（用于国际化）org.sipesc.example.helloworld_zh_CN.qm。

注：当成功制作一个插件后，向其他使用者提供的文件包括插件文件（.sep 文件）、特征文件（.features 文件）和翻译文件（.qm 文件）。

1.4.2 通过 SiPESC 平台编译、安装插件

1. 准备工作

通过 SiPESC 平台设计的插件可以直接通过 SiPESC 平台进行编译、安装，而使用“插件接口设计器”设计的插件，如果想通过 SiPESC 平台进行编译、安装，需要先将插件导入 SiPESC 平台，使之成为 SiPESC 平台项目，具体操作如下：

SiPESC 平台点击“文件”->“导入”，如图 25，选择 Cmake 项目。

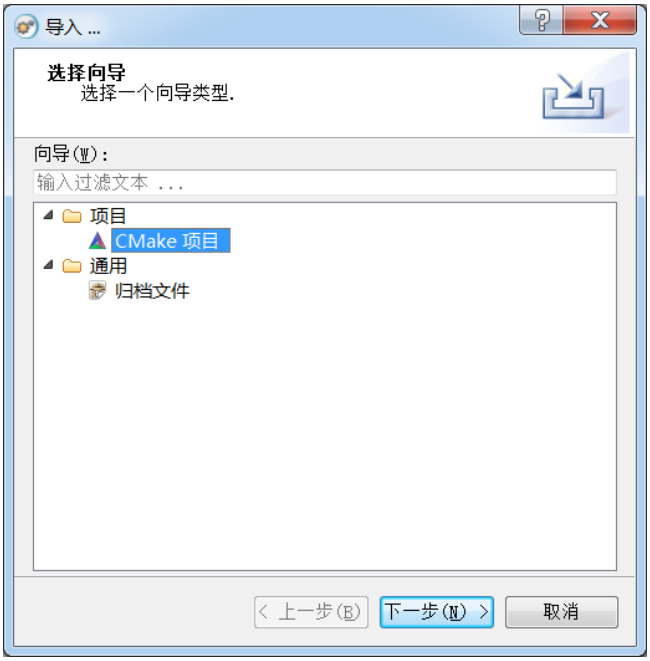

图25

如图 26，点击“浏览”，选中~/example_helloworld 路径向的 CMakeLists.txt 文件，点击“确定”，导入 HelloWorld 插件。

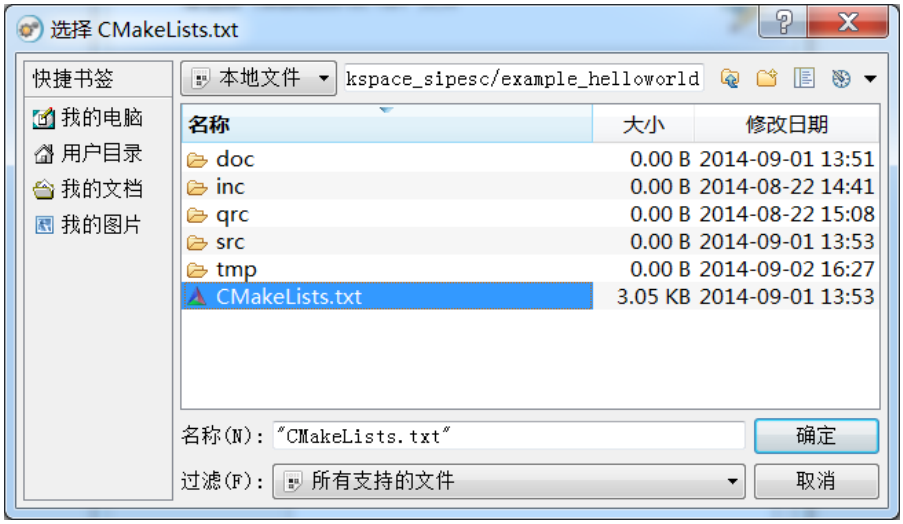

图26

接下来是配置 HelloWrold 插件项目，如图 27。

注：安装目录，即插件生成后安装的位置，选择用户目录，例如：C：/User/jifx/.sipesc。构建类型，即生成插件的版本。调试版本对应的是 debug 版，发行版本对应 release 版。

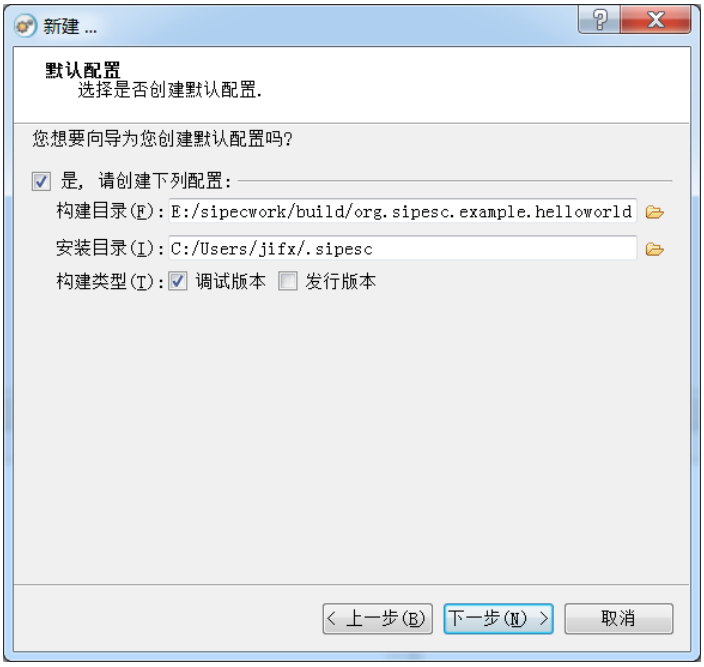

图27

点击“下一步”->“完成”，至此，插件已成功导入，成为 SiPESC 平台项目。

2. 插件的编译、安装

如图 28，图 29 在“项目浏览器”中，项目名称“org.sipesc.example.helloworld”上，右键->构建项目，点击“x86_64_windows-debug[Windows 64 位调试版本]”。对插件项目进行构建之后，在“布署项目”中点击“x86_64_windows-debug[Windows 64 位调试版本]”。

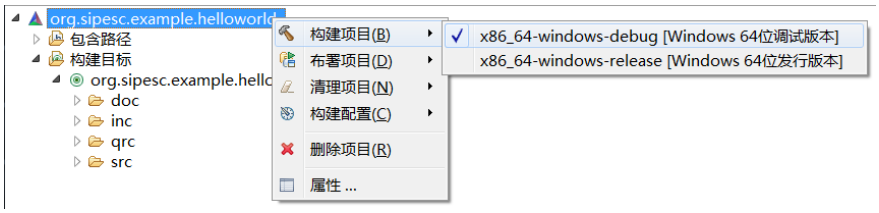

图28

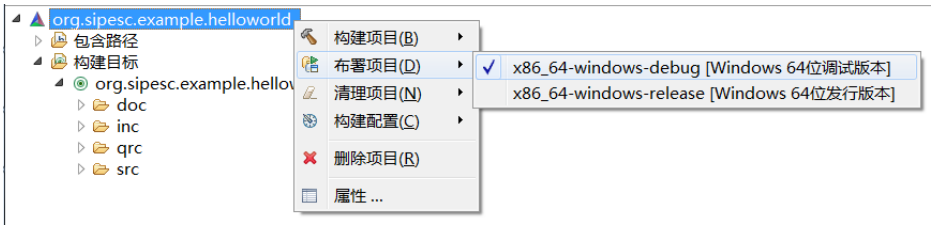

图29

至此，HelloWorld 插件已完成通过 SiPESC 平台进行编译、安装，成功之后，在 C:\Users\jifx\.sipesc\lib\plugins\debug 目录下生成插件 org.sipesc.example.helloworld_1.0.0.sep。在 C:\Users\jifx\.sipesc\share\features 目录下会生成特征文件 org.sipesc.example.helloworld.features。

1.5 利用 SiPESC 平台测试插件

1. 打开 SiPESC 平台，配置“org.sipesc.example.helloworld”插件,如图 30。

注意：打开的 SiPESC 平台版本与插件编译生成的版本要一致，

如果插件编译的是 debug 版本，则通过 mextmgr-launcherd.exe 打开 debug 版本的平台，

如果插件编译的是 release 版本，则通过 mextmgr-launcher.exe 打开 release 版本的平台

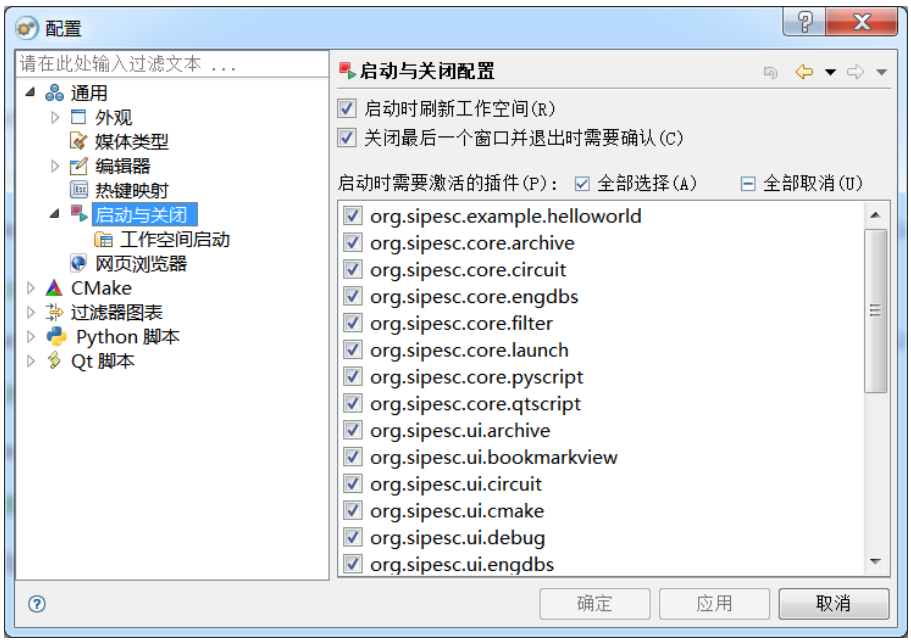

图30

2. 如图 31，“新建”->“其他”，建立 Qt 脚本文件（也可以建立 Python 脚本文件，脚本文件的代码与 Qt 脚本文件略有不同，这里以 Qt 脚本文件为例）。

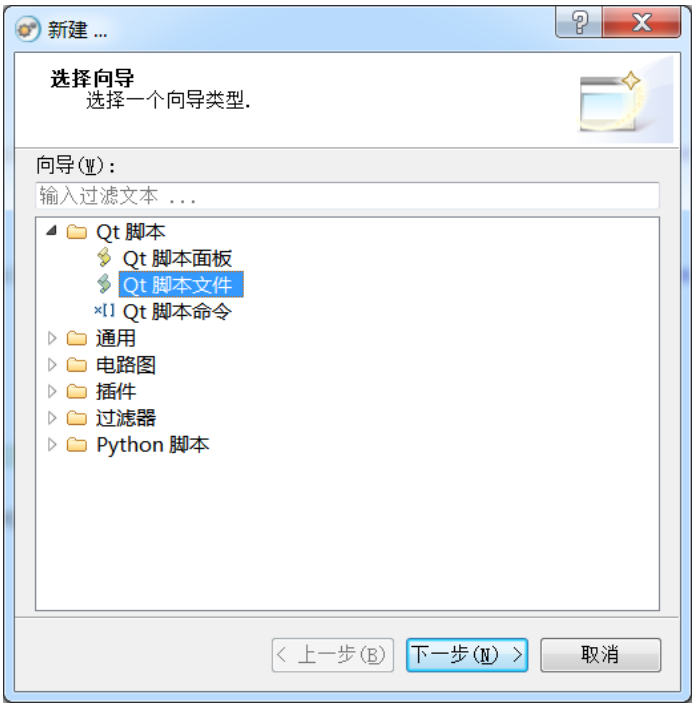

图31

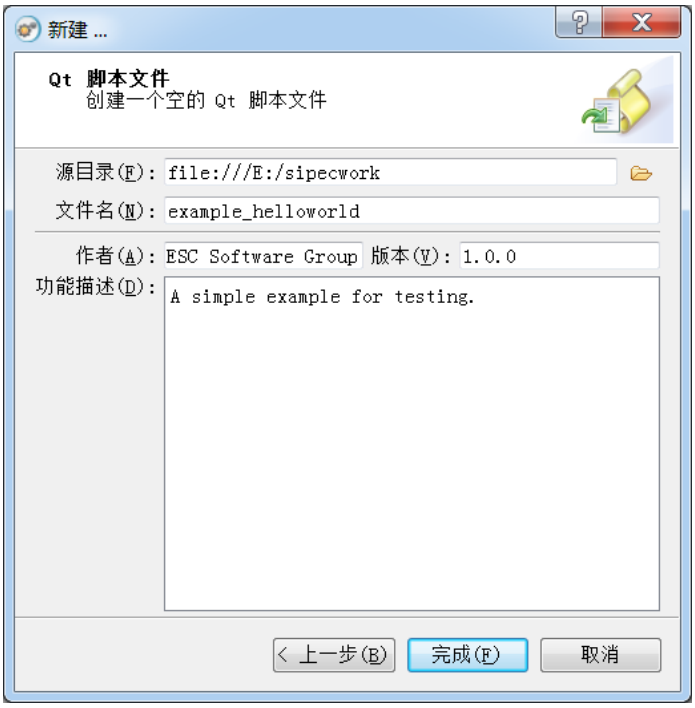

图32

添写脚本信息，如图 32，单击“完成”。在编辑器中输入以下代码：

```
/**************************************************************************
* A simple example for testing.
*
```

```
* @author SIPESC Software Group
* @version 1.0.0
*/
/**
* 获取全局扩展管理器。
*/
var manager = new MExtensionManager;
/**
* 由扩展管理器创建MessageProvider对象。
*/
var provider =
manager.createExtension("org.sipesc.example.helloworld.MessageProvider");
/**
* 由扩展管理器创建MessageReceiver对象。
*/
var receiver =
manager.createExtension("org.sipesc.example.helloworld.MessageReceiver");
/**
* 从provider获取信息，并在控制台显示。
*/
print(provider.getMessage());
/**
* 从provider获取信息，再由receiver显示该信息。
*/
receiver.showMessage(provider.getMessage());
```

3. 保存脚本文件，点击平台工具栏  图标中的倒三角，取消“在新进程中运行”，勾选为在相应的版本中运行。本例中选为“使用调试版本”。然后点击  中的小黑人“开始运行当前脚本”按钮。如图 33、图 34，控制台、调试台都输出“Hello World!”，其中控制台的输出来自语句“print(provider.getMessage());”，调试台的输出来自语句“receiver.showMessage(provider.getMessage());”

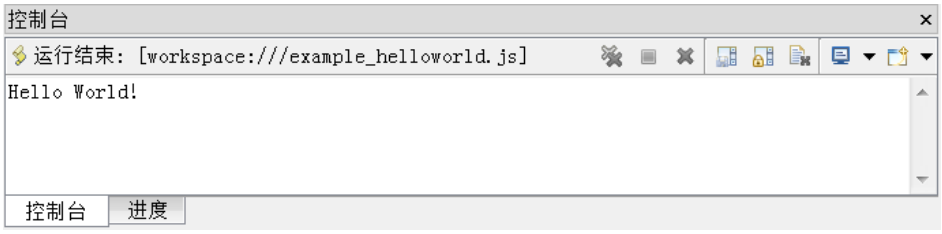

图 33

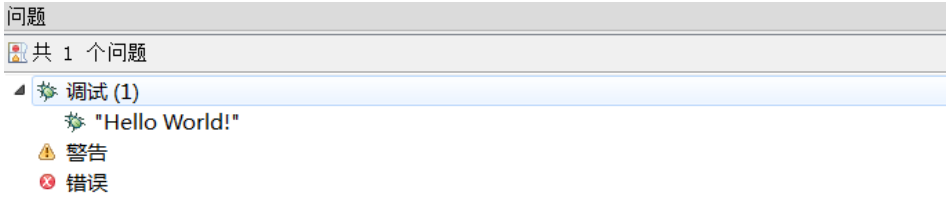

图 34

1.6 平台插件安装位置介绍

在我们编译插件的时候，命令中有这样一段：“-DCMAKE_INSTALL_PREFIX=c:Users/登陆的用户名/.sipesc”，这里的“c:Users/登陆的用户名/.sipesc”就是我们编写插件的安装目录，如图 35。

图 35

这四个文件夹中：

Etc：不用管。

Include：所有插件的头文件。

Lib：分为 debug 和 release 文件夹，分别存放两个版本的插件。

Share：分为 features 和 translations 两个文件夹，分别存放特征文件和翻译文件。

拷贝插件的时候只需要有 lib 和 share 两个文件夹，插件就能正常使用了。这里 lib 文件夹如图 36。

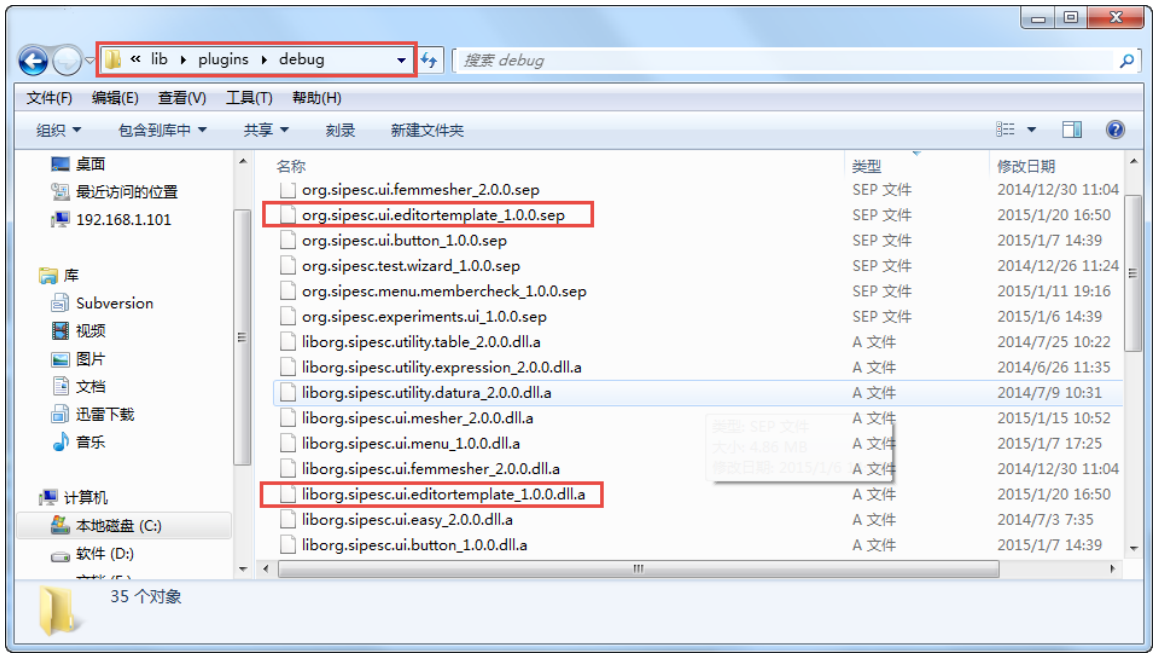

图 36

执行过 make install 后，就会把 temp 中的插件拷贝到这里来。

Org.sipesc.ui.editortemplate_1.0.0.sep 是插件文件

Org.sipesc.ui.editortemplate_1.0.0.a 不用管

所以安装和拷贝插件，拷的就是“.sep”、“.feature”和“.qm”三个文件。