Developing Plugins to SPRING

Windows and Linux Version

SPRING Plugins

2010 February

INPE

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Index

SPRING Plugins

This document shows SPRING plugin technology and through of the plugin technology the users can incorporate your implementations in SPRING software. The plugin technology is used to add functions in software, including some special or specific functionality. Usually is small and light, and used only by demand.

Este documento apresenta a tecnologia de plugins para o SPRING e através dessa tecnologia os usuários poderão incorporar suas implementações ao software SPRING. A tecnologia de plugins é utilizada para adicionar funções a softwares, provendo alguma funcionalidade especial ou muito específica. Geralmente é pequeno e leve, e utilizado somente sob demanda.

SPRING supports plugin technology to allow external developers to increase the functionality of SPRING. These users can deliver their search and implementations, evolving more and more the SPRING software.

O SPRING possui o suporte a tecnologia de plugins para permitir que desenvolvedores externos aumentem as funcionalidades do SPRING e disponibilizem seus estudos e implementações, evoluindo cada vez mais o software SPRING.

The SPRING plugin structure has four layers: Information, Interface, Parameters and Management, only the three first layers are necessary for external users develops your studies, and put on SPRING.

A estrutura de plugins do SPRING, possui quatro camadas: Informação, Interface, Parâmetros e Gerenciamento, apenas as três primeiras são necessárias para que os usuários externos incorporem suas implementações ao SPRING.

SPRING Plugin Structure

This chapter shows the four layers of SPRING plugin structure. The SPRING software uses these layers to define, add and execute plugins installed. Below, detail of each layers presents in SPRING plugins structure.

Neste capítulo apresentamos as camadas presentes na estrutura de plugins do SPRING. O SPRING se utiliza dessas quatro camadas para incorporar e executar os plugins desenvolvidos. Abaixo um detalhamento de cada camada da tecnologia de plugins.

Information Layer

The information layer is responsible to define basics information of plugin. These information are:

A camada de informação é responsável por definir informações básicas do plugin. Essas informações são:

- Plugin name (plugin_name_)
- Plugin identification (plugin_type_)
- Interface layer version (plugin_interface_version_) * This value can't be modified
- Plugin build number (plugin_build_number_)
- Plugin version (plugin_major_version_, plugin_minor_version_)
- Plugin description (plugin_description_)
- Plugin developer (plugin_vendor_)
- Email of plugin developer (plugin_vendor_email_)
- Url of plugin (plugin_url_)

Interface Layer

Interface layer has the information layer and a plugin execution method. New plugins are sons of this layer always, in other words, SPRING plugins ever will inherit this layer. Inherits the interface layer, is necessary implement the method *ExecuteSpringPlugin*, this method was called to execute the plugin.

A camada de interface contém a camada de informação e contém o método de execução do plugin. Os plugins são sempre filhos dessa camada, ou seja, os plugins do SPRING sempre herdarão dessa camada. Ao herdar a camada de interface, é obrigatório implementar o

método *ExecuteSpringPlugin*, pois esse será o método chamado ao executar um plugin.

Also, this layer has the plugin declaration necessary by Qt, framework that control the plugin load, and the interface layer is responsible to validate a plugin. Only valid plugins are loaded. A valid plugin is when this is compiled in same version of SPRING and Qt. This information can be find in SPRING About.

Além disso, essa camada contém a declaração de plugin necessária pelo Qt, framework que controla o carregamento do plugin.

E a camada de Interface também é responsável por validar um plugin e apenas os plugins válidos serão carregados. Um plugin é válido quando o mesmo for gerado(compilado) na mesma versão que o SPRING e Qt do executável SPRING. Essa informação pode ser obtida no sobre do software SPRING.

Parameters Layer

Parameters layer is responsible to available methods that execute some operation on SPRING or return controls variables of SPRING. The operations and variables available are:

A camada de parâmetros é responsável por disponibilizar métodos que executam alguma operação no SPRING ou retornam variáveis de controle do SPRING. Dentre as operações e variáveis disponíveis são:

- *getCurrentDatabasePtr*: Return active database pointer
- *getCurrentInfolayerPtr*: Return active infolayer pointer
- *getCurrentCategoryPtr*: Return active category pointer
- *getCurrentProjectPtr*: Return active project pointer
- *getSelectedBox*: Return selection box of area cursor
- *getBoxDrawArea*: Returna box of draw area
- *updateControlPanel*: Update the control panel
- *getQFileDialogPtr*: Return file dialog pointer
- *drawImage*: Draw image on selected screen, if image has one channel, this are painter, if image has three channels is generated a composition of channels
- *getAuxiliarCanvasPtr*: Return canvas pointer of Assistant window
- *getSErrorPtr*: Return the pointer of error and progress bar control
- *getSPRINGDBPtr*: Return the database path
- *activatedDatabase*: Activate a database, passing a directory and database name

Management Layer

Management Layer is used only by SPRING software, that search by ".dll"(Windows) or ".so"(Linux) files on directory "sprplugins". For each plugin found is verified if is a valid plugin and load the plugin.

A camada de gerenciamento é utilizada apenas pelo software SPRING, que procura no diretório "sprplugins" por arquivos ".dll"(Windows) ou ".so"(Linux), e para cada plugin encontrado, verifica sua validade e carrega o plugin.

Also, the management layer creates a menu "Plugins" on main window of SPRING. This menu "Plugins" has a plugin installation item and adds a item for each loaded plugin. Through these items the plugins are executed.

Além disso, a camada de gerenciamento cria o menu "Plugins" na interface principal do SPRING que possui um item de instalação de plugins e adiciona um item para cada plugin carregado. Através desses itens que os plugins serão executados.

Plugin Development

To develop plugins for SPRING are necessary some prerequisites. These prerequisites are details below, in this chapter.

Para desenvolver plugins para o SPRING são necessários alguns pré-requisitos. Os pré-requisitos são detalhados abaixo, nesse capítulo.

Headers and libraries

The headers files of SPRING functionality and libraries are required in plugin compilation process. Headers files are written on C++ language, because the SPRING be developed in this programming language.

Os arquivos de cabeçalho das funcionalidades do SPRING e as bibliotecas são necessários no processo de compilação do plugin. Os arquivos que formam o cabeçalho estão na linguagem C++, devido o SPRING ser desenvolvido nesta linguagem de programação.

Headers files has classes, methods, variables and identifiers declarations. Including the files paths on plugin code, a external developers can use classes, methods and variables of SPRING.

Os arquivos de cabeçalhos contém as declarações das classes, rotinas, variáveis e identificadores. Através da inclusão desses arquivos no código do plugin, será possível utilizar as classes, os métodos e variáveis presentes no SPRING.

Libraries (.lib or .so files) are required for phase linking in the plugin generation. If the plugin uses any classes, methods or variables of SPRING, will be necessary link the library that has these classes, method or variable. The examples chapters explain how realize this step.

As bibliotecas (arquivos .lib ou .so) são necessárias para a fase de lincagem ou ligação do plugin. Quando o plugin usufruir de alguma classe ou método do SPRING será necessário ligar ou lincar a biblioteca que contém a classe ou método utilizado. Nos exemplos do próximo capítulo explicaremos como realizar essa etapa.

Along with the required files for plugin creation, is available a classes documentation of SPRING. This documentation is based on existing implementations in SPRING. Through this, is possible manipulate and work with all existing features. Examples are: image and vector manipulations, infolayer manipulations, access method of digital image processing and another implementations.

Além dos arquivos necessários para a criação do plugins, é disponibilizada a documentação de classes do SPRING, esta documentação fundamenta as implementações existentes no SPRING. Através desta, é possível manipular e trabalhar todas as características

presentes. Como exemplos podemos citar: manipulação de imagens e dados vetoriais, manipulação do plano de informação, acessar métodos de processamento digital de imagem e diversas outras implementações.

Header and libraries Availability

Required files for SPRING plugin creation are headers and libraries, described in the previous chapter. There are two ways to get those files, download a headers and libraries installer in SPRING internet site(www.dpi.inpe.br/spring) or by SPRING Open Source (www.spring-gis.org).

Os arquivos necessários para a criação de um plugin do SPRING são os cabeçalhos e bibliotecas, descritos no capítulo anterior. Existem duas formas de obter esses arquivos, através do download de um instalador com os cabeçalhos e bibliotecas disponíveis no sitio de internet do SPRING (www.dpi.inpe.br/spring) ou pelo SPRING Código Aberto (www.spring-gis.org).

Compilers

The tested compilers in SPRING software are:

Os compiladores testados para compilar o conjunto de aplicativos SPRING são:

- Windows:
  - Visual Studio 2008
    - Express version, is a more simpler version and can be obtained in internet site http://www.microsoft.com/exPress/
- Linux
  - GCC / G++
    - This compiler is already installed on Linux distributions

Qt

For Windows and Linux environment Qt is needed. It's freely available from Nokia at the website http://qt.nokia.com.

It's necessary to compile it as a shared library and with OpenGL support. It may take sometime.

- Get Qt and unzip it to a suitable directory (qt-win-opensource-src-<version>.zip)
- Open a command prompt (Windows – Visual Studio 2008 Command Prompt) or

a Terminal (Linux)

- Go to the directory where Qt has been extracted
  - cd c:\Qt\qt-win-opensource-src-4.5.3
- Run the configure executable and follow the steps
  - configure -opengl
- After this, install it
  - Visual Studio 2008
    - nmake install
  - GCC / G++
    - make install
- Finished

Database

On Windows it's not necessary to install MySQL and PostgreSQL libraries, but on Linux both MySQL and PostgreSQL development libraries should be installed first using the package manager.

Examples:

- Installing on Ubuntu 9.04
  - apt-get install libmysqlclient16-dev
  - apt-get install libpq-dev
- Installing on OpenSUSE 11
  - yast2 --install libmysqlclient-devel
  - yast2 --install postgresql-devel

Plugins Examples

In this chapter, shows examples how uses the Information, Interface and Parameter layers of SPRING plugin methodology. Introduce a Hello SPRING and a database creation examples. This examples will be distributed with SPRING.

Neste capítulo mostraremos exemplos de como utilizar as camadas de Informação, Interface e Parâmetros da metodologia de plugins para o software SPRING. Apresentaremos os exemplos de Hello SPRING, bem mais simples e a criação de banco de dados. Esses exemplos serão distribuídos junto ao SPRING.

Hello SPRING

Hello SPRING is more simple, but has all steps for a plugin development and uses almost all functionality offered by parameters layers.

O exemplo Hello SPRING apesar de bem simples, contém todos os passos necessários para o desenvolvimento de um plugin e utiliza grande quantidade das funcionalidades oferecidas pela camada de parâmetros.

Analyzing Hello SPRING Plugin

The "*hellospringplugin.pro*" file in the "*projetos/hellospringplugin*" path is a project file, through these are generated ".vcproj" or "Makefile" files. This file consists by SPRING plugin definitions and which files will be part of this plugin.

O arquivo "*hellospringplugin.pro*" no diretório "*projetos/hellospringplugin*" é um arquivo de projetos, que através deste são gerados os arquivos ".vcproj" ou "Makefile". O conteúdo desse arquivo é composto pela definição de plugin do SPRING e quais arquivos farão parte desse plugin.

LIB_NAME variable is required, this variable define a plugin file name. Example "LIB_NAME = hellospringplugin".

A variável LIB_NAME é obrigatória, essa variável define o nome do arquivo do plugin (.dll ou .so). Exemplo "LIB_NAME = hellospringplugin".

The "include(../plugindefines.pri)" is required, this include says that this project is a plugin.

A linha "include(../plugindefines.pri)" é obrigatória, esse include determina que esse projeto será um Plugin.

The “include(../libfreetype/libfreetypepath.pri)” , “include(../libcodebase/libcodebasepath.pri)” and “include(../libspring/libspringpath.pri)” are lines that adds to INCLUDEPATH the headers path.

As linhas “include(../libfreetype/libfreetypepath.pri)” , “include(../libcodebase/libcodebasepath.pri)”, “include(../libspring/libspringpath.pri)” adicionam ao INCLUDEPATH (caminho de inclusão) os diretórios dos cabeçalhos do SPRING.

The “include($${LIB_NAME}.inc)” line is that adds the “hellospringplugin.inc” file, and will be explain following.

A linha “include($${LIB_NAME}.inc)”, adiciona o arquivo “hellospringplugin.inc”, que será explicado a seguir.

The line “include(../basespringplugin.inc)” is required, this line adds necessary files for plugin creation.

A linha “include(../basespringplugin.inc)” é obrigatória, pois agrega arquivos necessários para a criação do Plugin.

The “LIBS += $${SPRDESTLIBS}/$${LIBPREFIX}libglobal.$${LIBSUFFIX} \

$${SPRDESTLIBS}/$${LIBPREFIX}libspring.$${LIBSUFFIX}” line defines which SPRING libraries will be linked with the plugin. This lines are required when the parameter layer uses some classes or variable of SPRING.

A linha “LIBS += $${SPRDESTLIBS}/$${LIBPREFIX}libglobal.$${LIBSUFFIX} \

$${SPRDESTLIBS}/$${LIBPREFIX}libspring.$${LIBSUFFIX}” define quais as bibliotecas do SPRING serão lincadas ao plugins, essas linhas são necessárias quando o plugin utiliza a camada de parâmetros ou qualquer outra classe do SPRING.

The “*hellospringplugin.inc*” file in the “*projetos/hellospringplugin*” path defines the files that will be part of the Hello SPRING plugin.

O arquivo “*hellospringplugin.inc*” no diretório “*projetos/hellospringplugin*” define quais são os arquivos que farão parte do plugin Hello SPRING.

The “*helloplugin.h*” file in the “*src/pluginspr/helloplugin*” path is the header that define the Hello SPRING plugin.

O arquivo “*helloplugin.h*” no diretório “*src/pluginspr/helloplugin*” é o cabeçalho que define o plugin Hello SPRING.

For SPRING plugin creation is required that the plugin class inherits SpringPluginInterface, that is the Interface layer, and inherits QObject, that is object definition of Qt. Example: class HelloSPRINGPlugin : public QObject, SpringPluginInterface {};

Para a criação de plugins no SPRING, é necessário que a classe do plugin

herde de SpringPluginInterface, que é a camada de interface, e herde de QObject, que é a definição de objeto do Qt. Exemplo: class HelloSPRINGPlugin : public QObject, SpringPluginInterface {};

Adds Q_INTERFACES macro, required by Qt. Example:Q_INTERFACES(SpringPluginInterface).

Inclusão da macro Q_INTERFACES, requisitada pelo Qt. Exemplo: Q_INTERFACES(SpringPluginInterface).

Polymorphism definition of ExecuteSpringPlugin method.

Definição do polimorfismo do método ExecuteSpringPlugin.

The "*helloplugin.cpp*" file in the "*src/pluginspr/helloplugin*" path is the implementation of Hello SPRING plugin.

O arquivo "*helloplugin.cpp*" no diretório "*src/pluginspr/helloplugin*" é a implementação do plugin Hello SPRING.

Sets the plugin information. Examples:

Determinar as informações do plugin. Exemplos:

```
_spring_plugin_info.plugin_name_ = "Hello SPRING";

_spring_plugin_info.plugin_vendor_ = "Raphael Meloni";
```

Adds Q_EXPORT_PLUGIN2 macro, required by Qt: Example: Q_EXPORT_PLUGIN2(helloplugin, HelloSPRINGPlugin);

Inclusão da macro Q_EXPORT_PLUGIN2, requisitada pelo Qt. Exemplo: Q_EXPORT_PLUGIN2(helloplugin, HelloSPRINGPlugin);

Polymorphism implementation of ExecuteSpringPlugin method. The ExecuteSpringPlugin method uses the parameters layer to show the plugin information(), active database, active project, projection of active project, active infolayer, infolayer representations, select box from cursor area and draw area. Examples:

```
DataBase* pgDb = spp->getCurrentDatabasePtr();

InfoLayer* pgIl = spp->getCurrentInfolayerPtr();
```

Implementação do polimorfismo do método ExecuteSpringPlugin. A implementação do método ExecuteSpringPlugin utiliza a camada de parâmetros, apresentando as informações do plugin (nome, criador e versão), o banco de dados ativo, o projeto ativo, a projeção do projeto, o plano de informação ativo, as representações presentes no plano de informação ativo, o box selecionado pelo cursor de área e a área de desenho. Exemplos:

```
DataBase* pgDb = spp->getCurrentDatabasePtr();

InfoLayer* pgIl = spp->getCurrentInfolayerPtr();
```

Database Creation Wizard

The database creation wizard example is more complex that last example. This plugin uses the plugins methodology and the parameters layer.

O exemplo do Assistente de Criação de Banco de Dados é bem mais complexo que o exemplo anterior e também utiliza a metodologia de plugins, usufruindo da camada de parâmetros.

Analyzing Database Creation Wizard Plugin

The "*databasewizardplugin.pro*" file in the "*projetos/databasewizard*" path is a project file, through these are generated ".vcproj" or "Makefile" files. This file consists by SPRING plugin definitions and which files will be part of this plugin.

O arquivo "*databasewizardplugin.pro*" no diretório "*projetos/databasewizard*" é um arquivo de projetos, que através deste são gerados os arquivos ".vcproj" ou "Makefile". O conteúdo desse arquivo é composto pela definição de plugin do SPRING e quais arquivos farão parte desse plugin.

LIB_NAME variable is required, this variable define a plugin file name. Example "LIB_NAME = databasewizardplugin".

A variável LIB_NAME é obrigatória, essa variável define o nome do arquivo do plugin (.dll ou .so). Exemplo "LIB_NAME = databasewizardplugin".

The "include(../plugindefines.pri)" is required, this include says that this project is a plugin.

A linha "include(../plugindefines.pri)" é obrigatória, esse include determina que esse projeto será um Plugin.

The "include(../libfreetype/libfreetypepath.pri)" , "include(../libcodebase/libcodebasepath.pri)" and "include(../libspring/libspringpath.pri)" are lines that adds to INCLUDEPATH the headers path.

As linhas "include(../libfreetype/libfreetypepath.pri)" , "include(../libcodebase/libcodebasepath.pri)", "include(../libspring/libspringpath.pri)" adicionam ao INCLUDEPATH (caminho de inclusão) os diretórios dos cabeçalhos do SPRING.

The "include($${LIB_NAME}.inc)" line is that adds the "databasewizardplugin .inc" file, and will be explain following.

A linha "include($${LIB_NAME}.inc)", adiciona as informações do arquivo "databasewizardplugin.inc".

The line “include(../basespringplugin.inc)” is required, this line adds necessary files for plugin creation.

A linha “include(../basespringplugin.inc)” é obrigatória, pois agrega arquivos necessários para a criação do Plugin.

The “LIBS += $${SPRDESTLIBS}/$${LIBPREFIX}libglobal.$${LIBSUFFIX} \

$${SPRDESTLIBS}/$${LIBPREFIX}libspring.$${LIBSUFFIX} \

$${SPRDESTLIBS}/libimp.$${LIBSUFFIX} \

$${SPRDESTLIBS}/libtiff.$${LIBSUFFIX}” line defines which SPRING libraries will be linked with the plugin. This lines are required when the parameter layer uses some classes or variable of SPRING.

A linha “LIBS += $${SPRDESTLIBS}/libglobal.$${LIBSUFFIX} \

$${SPRDESTLIBS}/libspring.$${LIBSUFFIX} \

$${SPRDESTLIBS}/libimp.$${LIBSUFFIX} \

$${SPRDESTLIBS}/libtiff.$${LIBSUFFIX}” define quais as bibliotecas do SPRING serão lincadas ao plugins, essas linhas são necessárias quando o plugin utiliza a camada de parâmetros ou qualquer outra classe do SPRING.

The “databasewizardplugin.*inc*” file in the “*projetos/databasewizard*” path defines the files that will be part of the Database Creation Wizard plugin.

O arquivo “databasewizardplugin *.inc*” no diretório “*projetos/databasewizard*” define quais são os arquivos que farão parte do plugin de Assistente de Criação de Banco de Dados.

The “*databasewizardplugin.h*” file in the “*src/pluginspr/databasewizard*” path is the header that define the Database Creation Wizard plugin.

O arquivo “*databasewizardplugin.h*” no diretório “*src/pluginspr/databasewizard*” é o cabeçalho que define o plugin Criação de Banco de Dados.

For SPRING plugin creation is required that the plugin class inherits SpringPluginInterface, that is the Interface layer, and inherits QObject, that is object definition of Qt. Example: class DatabaseWizardPlugin : public QObject, SpringPluginInterface {};

Para a criação de plugins no SPRING, é necessário que a classe do plugin herde de SpringPluginInterface, que é a camada de interface, e herde de QObject, que é a definição de objeto do Qt. Exemplo: class DatabaseWizardPlugin : public QObject, SpringPluginInterface {};

Adds Q_INTERFACES macro, required by Qt. Example:Q_INTERFACES(SpringPluginInterface).

Inclusão da macro Q_INTERFACES, requisitada pelo Qt. Exemplo: Q_INTERFACES(SpringPluginInterface).

Polymorphism definition of ExecuteSpringPlugin method.

Definição do polimorfismo do método ExecuteSpringPlugin.

The "*databasewizardplugin.cpp*" file in the "*src/pluginspr/databasewizard*" path is the implementation of Database Creation Wizard plugin.

O arquivo "*databasewizardplugin.cpp*" no diretório "*src/pluginspr/databasewizard*" é a implementação do plugin Criação de Banco de Dados.

Sets the plugin information. Examples:

Determinar as informações do plugin. Exemplos:

_spring_plugin_info.plugin_name_ = tr("Database Creation Wizard");

_spring_plugin_info.plugin_vendor_ = "Raphael Meloni";

Adds Q_EXPORT_PLUGIN2 macro, required by Qt: Example: Q_EXPORT_PLUGIN2(helloplugin, DatabaseWizardPlugin).

Inclusão da macro Q_EXPORT_PLUGIN2, requisitada pelo Qt. Exemplo: Q_EXPORT_PLUGIN2(databasewizardplugin, DatabaseWizardPlugin).

Polymorphism implementation of ExecuteSpringPlugin method. The ExecuteSpringPlugin method open a wizard window to database creation. The plugins files uses the parameter layer and SPRING libraries for database creation, project creation(optional) and active the database(optional). Examples:

Implementação do polimorfismo do método ExecuteSpringPlugin. A implementação do método ExecuteSpringPlugin abre uma janela em forma de assistente, para a criação de bancos de dados. Os arquivos, que compõem esse plugins, utilizam a camada de parâmetros e a biblioteca do SPRING com o objetivo de criar o Banco de Dados, criar o Projeto (opcional) e ativar o banco de dados criado (opcional). Exemplos:

The " *databasewizardpages.cpp*" file in the "*src/pluginspr/databasewizard*" has:

No arquivo "*databasewizardpages.cpp*" do diretório "*src/pluginspr/databasewizard*":

QString springdb = springparameters->getSPRINGDBPtr();

QFileDialog* fileDialog = springparameters->getQFileDialogPtr();

The " *databasewizarddlg.cpp*" file in the "*src/pluginspr/databasewizard*" has:

No arquivo "*databasewizarddlg.cpp*" do diretório "*src/pluginspr/databasewizard*":

pluginParameters->activatedDatabase (databaseName , databaseDir);

Compile and Execute a Plugin

In this module shows how to compile, install and execute a plugin on SPRING software. In following steps, we uses the Hello SPRING plugin as example. To share your plugin, available only the dll file of plugin.

Neste módulo apresentaremos como compilar, instalar e executar um plugin no software SPRING. Nos passos a seguir, tomaremos como base o exemplo do plugin Hello SPRING. Para distribuir seu plugin, apenas disponibilize a dll gerada.

- Open a Command Prompt(Windows) or a Terminal(Linux) that has a compiler and Qt definitions. (Example: Qt <version> Command Prompt)
- Windows
  - Go to plugin Hello SPRING path where SPRING Open Source was installed
    - cd <spring_open_source>\projetos\hellospringplugin
  - Creates a Visual Studio project
    - qmake -tp vc hellospringplugin.pro
  - Open the generate file(hellospringplugin.vcproj) on Visual Studio
    - devenv hellospringplugin.vcproj
  - Compile the Release version of plugin
    - Dll file generated for the plugin <spring_open_source>\[distribuicao|distribuicaox64]\sprplugins\hellospringplugin.dll
- Linux
  - Go to plugin Hello SPRING path where SPRING Open Soure was installed
    - cd <spring_open_source>/projetos/hellospringplugin
  - Makefile
    - qmake hellospringplugin.pro
  - Compile the Release version of plugin
    - make Release
    - Dll file generated for the plugin <spring_open_source>\[distribuicao|distribuicaox64]\sprplugins\hellospringplugin.dll
- Uses the Install Plugins tool of SPRING to add your plugin at SPRING, If the plugin has some incompatibility it will be presented.
- Use a ferramenta Instalar Plugins do SPRING, para adicionar o seu plugin ao SPRING, caso o plugin possua alguma incompatibilidade a mesma será apresentada

    - SPRING → Plugins → Install Plugins...

- Execute your Plugin at SPRING
    - SPRING → Plugins → <Plugin Name>

Integration of Plugin and SPRING Open Source

If you want to include your plugin in your SPRING version adds the plugin generation on SPRING generation. For this is required only one modification on project file (.vcproj e Makefile) generation. Again we take the example of the Hello SPRING plugin.

Caso queira incluir o seu plugin a sua versão do SPRING adicione a geração do seu plugin na geração do SPRING, para isso é necessário apenas uma alteração na criação dos arquivos de projeto (.vcproj e Makefile). Novamente tomaremos como exemplo o plugin Hello SPRING.

Adds the plugin compilation on "*projetos.pro*" file in the "projetos" path, for this includes the line "SUBDIRS += hellospringplugin" between the commentary lines "#add plugins here".

Inclua no arquivo "*projetos.pro*" do diretório "*projetos*" a compilação do plugin, para isso adicione a linha "SUBDIRS += hellospringplugin" entre as linhas que contém o comentário "#add plugins here".

After this, recreate the project files and compile the SPRING software and the plugin. Hello SPRING plugin will be saved on SPRING plugins path,

Após isso recrie os arquivos de projeto e compile o SPRING e o plugin. O plugin já será criado na pasta de plugins do SPRING, is not required to install the plugin.