Desenvolvendo Plugins para o SPRING

Versão Windows e Linux

SPRING Plugins

Fevereiro de 2010

INPE

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Sumário

SPRING Plugins

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.

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.

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.

Estrutura de Plugins do SPRING

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.

Camada de Informação

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

- O nome do plugin (plugin_name_)
- A identificação do plugin (plugin_type_)
- Versão da camada de interface (plugin_interface_version_) * Valor não pode ser modificado
- O numero de build do plugin (plugin_build_number_)
- A versão do plugin (plugin_major_version_, plugin_minor_version_)
- A descrição do plugin (plugin_description_)
- Quem desenvolveu o plugin (plugin_vendor_)
- E-mail do desenvolvedor do plugin (plugin_vendor_email_)
- Url de divulgação do plugin (plugin_url_)

Camada de Interface

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.

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 o Qt. Essa informação pode ser obtida no sobre do software SPRING.

Camada de Parâmetros

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*: Retorna o ponteiro do banco de dados ativo
- *getCurrentInfolayerPtr*: Retorna o ponteiro do plano de informação ativo
- *getCurrentCategoryPtr*: Retorna o ponteiro da categoria ativa
- *getCurrentProjectPtr*: Retorna o ponteiro do projeto ativo
- *getSelectedBox*: Retorna o box da seleção do cursor de área
- *getBoxDrawArea*: Retorna o box da área de desenho
- *updateControlPanel*: Atualiza o painel de controle
- *getQFileDialogPtr*: Retorna o ponteiro para o diálogo de arquivo
- *drawImage*: Desenha uma imagem na tela selecionada, se a imagem tiver apenas um canal, o mesmo será desenhado, caso tenha três canais, é gerado uma composição dos três canais
- *getAuxiliarCanvasPtr*: retorna o ponteiro do canvas da janela Auxiliar
- *getSErrorPtr*: Retorna o ponteiro do controle de erro e barra de progresso
- *getSPRINGDBPtr*: Retorna o caminho do banco de dados
- *activatedDatabase*: Ativa um banco de dados, dado um diretório e o nome do banco de dados

Camada de Gerenciamento

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.

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.

Desenvolvimento de Plugins

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

Cabeçalho e bibliotecas

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.

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.

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.

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.

Disponibilização do cabeçalho e bibliotecas

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).

Compiladores

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

- Ambiente Windows:
  - Visual Studio 2008
    - A versão Express, a mais simples, pode ser obtido gratuitamente no sitio http://www.microsoft.com/exPress/
- Ambiente Linux
  - GCC / G++
    - Normalmente esse compilador já vem instalado nas distribuições Linux

Qt

Para os ambientes Windows e Linux é necessário a instalação do Framework Qt responsável principalmente pelo gerenciamento das janelas. O Qt é disponibilizado pela Nokia e possui uma licença livre e pode ser obtido no sitio de internet http://qt.nokia.com/.

Uma necessidade da biblioteca Qt é a compilação da mesma de forma dinâmica e com suporte OpenGL. Abaixo uma demostração da compilação da biblioteca Qt. Esse passo é bem demorado.

- Descompacte o arquivo baixado (qt-win-opensource-src-<versão>.zip)
- Abra um Prompt de Comando(Windows) ou Terminal(Linux) que contenha as informações do compilador (Ex: Visual Studio 2008 Command Prompt)
- Entre no diretório, em que o Qt foi extraído
  - cd c:\Qt\qt-win-opensource-src-4.5.3
- Windows
  - Execute o comando configure e siga os passos para configurar a biblioteca do Qt
    - configure
  - Após configurado, compile e instale a biblioteca com o seguinte comando
    - Visual Studio 2008
      - nmake install
- Linux
  - A compilação do Qt no ambiente Linux é um pouco mais complicada, é necessário instalar uma série de bibliotecas antes de configurar a geração do Qt. As bibliotecas são:
    - OpenGl (mesa-libglu-devel)
    - LibXext (libXext-devel)

- LibXinerama (libXinerama-devel)
- LibXcursor (libXcursor-devel)
- LibXfixes (libXfixes-devel)
- LibXrandr (libXrandr-devel)
- LibXrender (libXrender-devel)
- LibXi (libXi-devel)
- FontConfig (fontconfig-devel)
- LibJpeg (libjpeg-devel)

além das bibliotecas as suas dependências também precisam ser instaladas. Os nomes das bibliotecas podem variar de acordo com a distribuição Linux.

Utilize o aplicativo "Instalar e Remover Programas" da distribuição Linux para instalar as bibliotecas requisitadas pelo Qt.

- Execute o comando configure e siga os passos para configurar a biblioteca do Qt
  - configure
    - Durante a configuração do Qt é apresentado uma lista de bibliotecas que o Qt utilizará. Verifique se as bibliotecas citadas acima estão com valores diferente de "no".
- Após configurado, compile e instale a biblioteca com o seguinte comando
  - GCC / G++
    - sudo make install

- Pronto Qt, compilado e instalado

Banco de Dados

Para o ambiente Windows não é necessário instalar as bibliotecas dos bancos de dados, as mesmas já se encontram no código fonte do SPRING, mas no ambiente Linux é necessário instalar o pacote de desenvolvimento do MySQL e PostgreSQL.

As bibliotecas de banco de dados necessárias no ambiente Linux são mysql-devel e postgresql-devel. Cada ambiente Linux possui uma maneira para instalar essas bibliotecas, utilize a que melhor se adéque a sua distribuição Linux. Abaixo uma demostração de instalação no ambiente Ubuntu e OpenSuse.

- Instalando as bibliotecas no Ubuntu 9.04
  - apt-get install libmysqlclient16-dev

    - apt-get install libpq-dev
- Instalando as bibliotecas no OpenSuse 11
    - yast2 --install libmysqlclient-devel
    - yast2 --install postgresql-devel

Exemplos de Plugins

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

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.

Analisando o plugin de Hello SPRING

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.

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

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

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

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

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

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.

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

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

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 {};

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

Definição do polimorfismo do método ExecuteSpringPlugin.

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

Determinar as informações do plugin. Exemplos:

```
_spring_plugin_info.plugin_name_ = "Hello SPRING";
_spring_plugin_info.plugin_vendor_ = "Raphael Meloni";
```

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

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();
```

Assistente de Criação de Banco de Dados

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.

Analisando o plugin de Assistente de Criação de Banco de Dados

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.

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

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

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.

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

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

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.

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

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 {};

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

Definição do polimorfismo do método ExecuteSpringPlugin.

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

Determinar as informações do plugin. Exemplos:

_spring_plugin_info.plugin_name_ = tr("Criação de Banco de Dados");

_spring_plugin_info.plugin_vendor_ = "Raphael Meloni";

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

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:

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

QString springdb = springparameters->getSPRINGDBPtr();

QFileDialog* fileDialog = springparameters->getQFileDialogPtr();

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

pluginParameters->activatedDatabase (databaseName , databaseDir);

Compilando e Executando um 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.

- Abra um Prompt de Comando(Windows) ou Terminal(Linux) que contenha as informações do compilador e do Qt (Ex: Qt <versão> Command Prompt)
- Windows
  - Entre no diretório do plugin Hello SPRING em que o SPRING Código Aberto foi instalado
    - cd <spring_codigo_aberto>\projetos\hellospringplugin
  - Crie o projeto do Visual Studio
    - qmake -tp vc hellospringplugin.pro
  - Abra no Visual Studio o arquivo gerado (hellospringplugin.vcproj)
    - devenv hellospringplugin.vcproj
  - Compile a versão Release do Plugin
    - Arquivo de plugin gerado <spring_codigo_aberto>\[distribuicao|distribuicaox64]\sprplugins\hellospringplugin.dll
- Linux
  - Entre no diretório do plugin Hello SPRING em que o SPRING Código Aberto foi instalado
    - cd <spring_codigo_aberto>/projetos/hellospringplugin
  - Makefile
    - qmake hellospringplugin.pro
  - Compile a versão Release do Plugin
    - make Release
    - Arquivo de plugin gerado <spring_codigo_aberto>\[distribuicao|distribuicaox64]\sprplugins\hellospringplugin.so
- 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 → Instalar Plugins...
- Execute o Plugin no SPRING
  - SPRING → Plugins → <Nome do Plugin>

Integração do Plugin com o SPRING Código Livre

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.

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".

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, não sendo necessário instalar o mesmo.