

# Plugins TerraView

Última revisão: 12/12/32006

Versão TerraLib: 3.1.4

## Requisitos

- Código completo da TerraLib na estrutura de diretórios sugerida no site da TerraLib<sup>1</sup>.
- Código completo do TerraView na estrutura de diretórios sugerida no site do TerraView<sup>2</sup>.
- Um compilador C++ apropriado para a plataforma escolhida:
  - Microsoft Visual Studio .NET 2003 para o ambiente Windows;
  - GNU GCC 3.2.3 ou superior para o ambiente Linux (veja o site do GNU para maiores informações<sup>3</sup>).
- QT 3.3.4 ou superior (binários e ambiente de desenvolvimento). Necessário para a geração de Makefiles (veja o site do Qt<sup>4</sup> para maiores informações).

## Considerações iniciais

Todos as dependências externas para um plugin (código externo e bibliotecas externas) podem ser resolvidas utilizando o projeto Qt base fornecido. Para utilizar esse projeto deve ser mantida a organização de diretórios sugerida nas páginas da TerraLib e do TerraView, mostrada na Figura 1.

---

<sup>1</sup> Website TerraLib: [www.terralib.org](http://www.terralib.org)

<sup>2</sup> Website TerraView: [www.dpi.inpe.br/terraview](http://www.dpi.inpe.br/terraview)

<sup>3</sup> GNU Website: <http://www.gnu.org/>

<sup>4</sup> QT Website: <http://www.trolltech.com/>

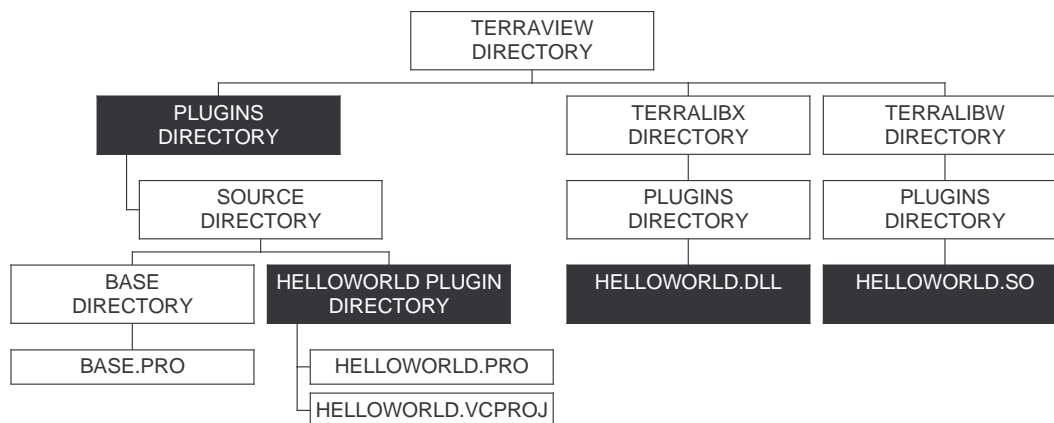


Figura 1 – Estrutura de diretórios da TerraLib, TerraView e Plugins.

## Construindo um Plugin “Hello World”

O suporte a Plugins para o TerraView usa uma biblioteca especialmente voltada para o gerenciamento de plugins chamada LibSPL (veja o site da LibSPL<sup>5</sup> para maiores informações), que resolve questões de portabilidade e de codificação. Assim sendo, todos os plugins para o TerraView devem ser linkados com a SPL (disponibilizada na estrutura de diretórios da TerraLib).

### *Criando o projeto do Plugin*

Para facilitar a geração de código, um arquivo contendo um projeto Qt é fornecido (ver o diretório “base” na Figura 1). Esse arquivo com o projeto básico pode ser incluído no projeto do novo plugin (**HelloWorld.pro**) de forma que todas as questões relativas a dependências e inclusão de arquivos estejam resolvidas. Com inclusão do projeto base (**base.pro**) os únicos arquivos que devem ser incluídos são aqueles específicos do plugin (*headers*, códigos fontes e arquivos de interface). O arquivo de geração do plugin é mostrado na Figura 2.

<sup>5</sup> Website LibSPL: <http://www.unitedbytes.de/go.php?site=spl>

```
include( ../base/base.pro )
FORMS += HelloworldPluginMainForm.ui
HEADERS += HelloworldPluginMainWindow.h HelloworldPluginCode.h
SOURCES += HelloworldPluginMainWindow.cpp HelloworldPluginCode.cpp
```

**Figura 2 – O arquivo HelloWorld.pro.**

O arquivo de projeto do plugin (**HelloWorld.pro**) pode ser usado para gerar o arquivo de projeto para cada versão do MS Visual Studio, ou Makefiles para usuários Linux utilizando a ferramenta “qmake”. Esta ferramenta permite a geração de arquivos de projeto para Visual Studio ou Makefiles a partir dos arquivos “.pro”. Para maiores informações consulte o site do QT.

### ***Criando o código principal do Plugin***

O uso da biblioteca SPL é muito simples. Apenas algumas macros devem ser usadas para criar o ponto de entrada principal do plugin. A primeira macro necessária (SPL\_DEFINE\_PLUGIN\_INFO) descreve a interface do Plugin e fornece todas informações sobre o Plugin que são o TerraView precisa no momento de carregamento do Plugin.

Seguindo com o exercício de construção de um Plugin HelloWorld, o conteúdo arquivo **HelloworldPluginCode.h** é mostrado na Figura 3.

```
#include <spl.h>
#if SPL_PLATFORM == SPL_PLATFORM_WIN32
    // Include windows.h - Needed for Windows projects.
    #include <windows.h>
#endif

#include <PluginParameters.h>
```

```

SPL_DEFINE_PLUGIN_EXPORTS();

SPL_DEFINE_PLUGIN_INFO(
    1, ///< The plugin build number.
    1, ///< The plugin major version (e.g. 1.xx).
    0, ///< The plugin minor version (e.g. 0.10).
    true, ///< Does this plugin show its arguments to the public?
    "HelloWorld", ///< The plugin's name.
    "INPE", ///< The plugin's vendor.
    "TerraView Sample Plugin", ///< The plugin's general description.
    PLUGINPARAMETERS_VERSION, ///< The expected parameters version.
    "http://www.dpi.inpe.br/terralib", ///< The plugin homepage.
    "terralib@dpi.inpe.br", ///< The plugin author email address.
    "TerraViewPlugin" ); ///< Plugin's UUID ( "TerraViewPlugin" ).
);

SPL_DEFINE_PLUGIN_DLLMAIN();
SPL_IMPLEMENT_PLUGIN_GETINFO();
SPL_PLUGIN_API bool SPL_INIT_NAME_CODE( slcPluginArgs* )
{
    return true;
}
SPL_PLUGIN_API bool SPL_SHUTDOWN_NAME_CODE( slcPluginArgs* )
{
    return true;
}

```

**Figura 3 – Código do arquivo HelloWorldPluginCode.h**

**NOTA:** as outras macros SPL são necessárias para a compilação dos PLUGINS mas sua implementação não é necessária.

**NOTA:** O uso da macro PLUGINPARAMETERS\_VERSION é necessário para evitar inconsistências entre os parâmetros de interface fornecidos pelo TerraView e os parâmetros de interface esperados pelo Plugin.

Outra macro que deve ser implementada define o código principal do plugin e é chamada `SPL_RUN_NAME_CODE` (o trecho de código que será chamado pelo TerraView quando um usuário ativa um dos plugins listados no menu “Plugins”). A mostra o código do sugerido para o arquivo **HelloWorldPluginCode.cpp**.

```
#include "HelloWorldPluginCode.h"
#include <HelloWorldPluginMainWindow.hpp>
#include <PluginBase.h>
#include <TeSharedPtr.h>
SPL_PLUGIN_API bool SPL_RUN_NAME_CODE( slcPluginArgs* a_pPluginArgs )
{
    void* arg_ptrs[ 1 ];
    a_pPluginArgs->GetArg( 0, arg_ptrs );

    PluginParameters* plug_pars_ptr =
        ( PluginParameters* ) arg_ptrs[ 0 ];

    // This is needed for windows - TeSingletons doesn't work with DLLs
    TeProgress::setProgressInterf( plug_pars_ptr->teprogressbase_ptr_ );

    /* Creating the plugin form instance statically ( just one form
    instance is created, even if the plugin is called many times ) */

    static TeSharedPtr<HelloWorldPluginMainWindow>    helloworldwindow_instance(
new HelloWorldPluginMainWindow(
    plug_pars_ptr->qtparent_widget_ptr_ ) );
    helloworldwindow_instance->update(plug_pars_ptr);
    helloworldwindow_instance->show();
    return true;
}
```

**Figura 4 – Código do arquivo HelloWorldPluginCode.cpp.**

**NOTA:** A cada execução do código definido pela macro `SPL_RUN_NAME_CODE` um ponteiro para uma instância para um objeto que contém parâmetros do Plugin (`slcPluginArgs*`) é passada. Esse objeto contém um ponteiro para uma instância de um objeto que contém os parâmetros de um Plugin TerraView (`PluginParameters*`) que

permite a interação com a interface do TerraView. Esse não o mesmo a cada execução do código definido pela macro `SPL_RUN_NAME_CODE` e é válido somente dentro de cada execução.

**NOTA par desenvolvedores QT:** no exemplo acima, uma instância de um formulário derivado do QT é criada e mostrada na tela (`HelloWorldPluginMainWindow`). O código principal do Plugin termina e o fluxo de execução retorna para o *loop* principal do TerraView, mas a instância do formulário irá permanecer ativa.

## ***Parâmetros do Plugin***

Os parâmetros disponibilizados pelo TerraView para os plugins permitem a comunicação bidirecional entre a aplicação principal (o TerraView) e os plugins. As questões de portabilidade (descritas abaixo) devem ser observadas com cuidado quando lidando com eles. Os parâmetros são (veja o código do arquivo **PluginParameters.h** para uma documentação detalhada):

### **Ponteiros para os objetos atualmente selecionados**

- Um ponteiro para a o banco corrente do TerraView (`current_database_ptr_`).
- Um ponteiro para o *layer* corrente ou ativo no TerraView (`current_layer_ptr_`).
- Um ponteiro para o vista corrente ou ativa no TerraView (`current_view_ptr_`).
- Um ponteiro para o tema ativo (`tethemeapplication_ptr_`).

## Ponteiros para os elementos de interface do TerraView

São fornecidos ponteiros para os elementos de interface do TerraView (mostrados na Figura 5). Estes ponteiros são sempre válidos para plugins modais e não modais. Os ponteiros disponibilizados são:

- Um ponteiro para a lista de bancos de dados conectados no TerraView (`teqtdatabaseslistview_ptr_`).
- Um ponteiro para a lista de vistas TerraView (`teqtviewslistview_ptr_`).
- Um ponteiro para a área de grade do Terraview (`teqtgrid_ptr_`).
- Um ponteiro para o canvas de desenho do Terraview (`teqtcanvas_ptr_`).

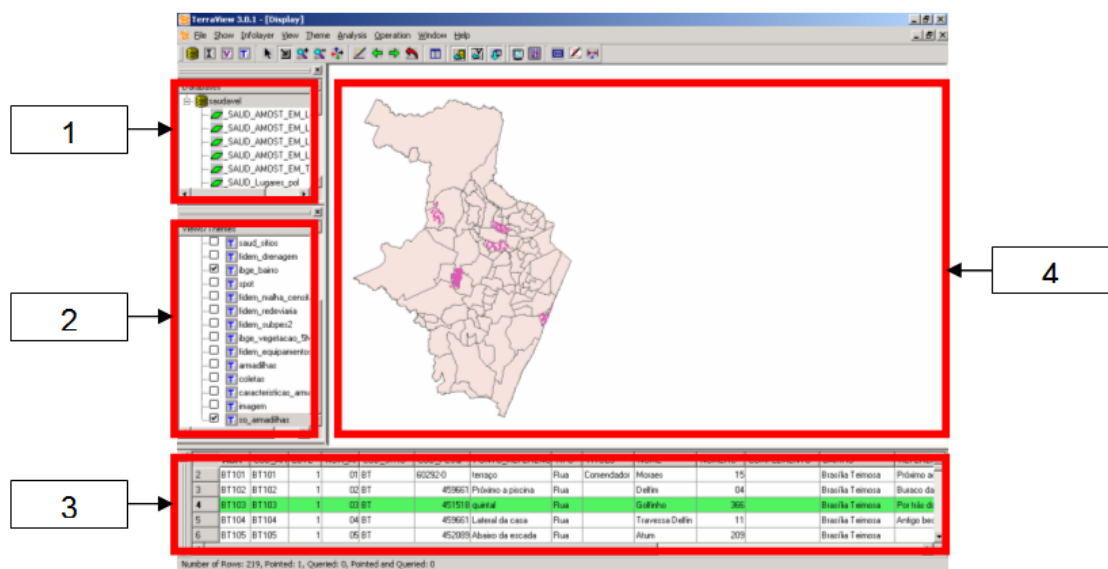


Figura 5 – Elementos de interface do TerraView.

## Ponteiros para instâncias de objetos auxiliares

Mais alguns ponteiros para outros objetos relacionados à interface gráfica do TerraView são fornecidos. Estes ponteiros são sempre válidos para plugins modais e não modais. Os ponteiros disponibilizados são:

- Um ponteiro para instância da janela principal do Qt (`qtparentwidget_prt_`). Deve ser usado para ligar a janela principal do plugin como filha da janela principal do TerraView.
- Um ponteiro para a instância da interface responsável por fornecer a barra de progresso do TerraView (`teprogressbase_prt_`).

## Ponteiros para funções auxiliares

Esses ponteiros apontam para funções internas do TerraView que desempenham funções específicas do lado da aplicação:

- `updateTVInterface`: atualiza a interface principal do TerraView, recarregando o conteúdo do banco de dados.

## Questões de Portabilidade

- **Modelo de memória do Windows**: devido ao modelo de memória das DLL's (Dynamic Linking Libraries) do Windows, algumas questões devem ser consideradas:
  - NUNCA deixe o código do Plugin destruir um objeto alocado dinamicamente pelo código da aplicação (TerraView). O caso inverso também deve ser evitado.
  - A tabela de alocação de memória do Pugin difere da tabela principal do TerraView e assim variáveis ou objetos globais estáticas também são as mesmas e devem ser tratadas com cuidado.
  - A classe `TeDatabase` detém o controle dos ponteiros para os planos de informação. Devido aos problemas mostrados acima, qualquer plano de informação criado do lado do Plugin não pode ser associado com uma instância de `TeDatabase` do TerraView, pois esses serão destruídos

quando essa instância for destruída. Para evitar erros, um clone da instância de `TeDatabase` deve ser criada do lado do Plugin (usando a classe `TeDatabaseFactory`), e todos os novos planos devem ser associados com à esta nova instância. Para evitar erros, o clone do `TeDatabase` deve ser destruído antes que processo de liberação (*unloading* da DLL).

## Questões freqüentes

1. Meu plugin é compilado corretamente, mas o TerraView não o reconhece. O que está errado?

Verifique se existem referências indefinidas durante a compilação. O carregador de códigos dinâmicos do sistema operacional não carrega bibliotecas onde faltam implementações de funções.

2. Meu plugin funcionava com a versão anterior do TerraView, mas agora a nova versão não o reconhece. O que posso fazer?

A estrutura de Plugins do TerraView verifica se a versão de interface é a mesma para o Plugin e para o TerraView. Se essa verificação falha, o plugin não será carregado. Recompile o código do seu plugin, atualizando-o o com a nova interface fornecida pelo TerraView.

3. Tudo está correto, mas quando eu fecho o TerraView (depois de ter usado meu plugin) ocorre um erro de falha de segmentação. O que está errado?

Possivelmente o TerraView está tentando destruir um objeto alocado no lado do Plugin. Verifique as questões de portabilidade descritas acima.