WisdomPluginFramework

插件框架设计说明

及使用手册

手册说明基于框架版本 Ver1.6.1

IceAir 写于 8 月广西南宁

一、框架简介

插件框架 WisdomPluginFramework 是融合 OSGI 微内核理念 + Eclipse 的扩展点概念而精心设计的轻量级插件框架，一切尽求朴素地简单，由 Delphi 实现，但可以使用于 Delphi、BCB、VC++中，提供面向接口的非常强大灵活的插件调度能力，让你充分享受插件式编程的乐趣。

期望后续能合集众人之能量，汇闪智慧之光芒！

项目地址：

https://code.csdn.net/IceAir/wisdompluginframework

git://code.csdn.net/IceAir/wisdompluginframework.git

二、框架特点及功能

（一）微内核设计

什么是微内核？就是插件核心实现小如原子，仅完成插件管理及接口调度，其它一切皆是插件，通过插件的相互关联、堆叠和构织，构建出大千世界。

（二）面向接口编程

插件核心、插件服务全部以纯接口方式暴露，任何插件在任何时候可以获取并使用任何其它插件所声明提供的接口服务，只需尽兴取用，插件如何调用、插件的生命周期管理、插件间复杂的关系处理就交给框架吧。

（三）扩展点概念让插件无限扩展

每个插件实现自己梦想的同时，能力大的还可以留给别人实现梦

想的空间，这个空间就是扩展点，每个插件都可以向别的插件声明扩展空间，任何其它插件都可以来实现这个扩展点，就如同一个插头又预留有插头口一样，插头和插头间可以互插而互连形成树状获得无限扩展能力，但从任何一个插件角度看去，其它插件又是平面化的，就像人与人一样，你可以与任何一个人你见到的人打交道，而不需要一定有介绍人。

扩展点概念是 Eclipse 的设计精髓之一，通过扩展点，每个插件通过树状的散射而具备无限的扩展能力。

（四）框架核心皆可动态更换

框架各核心接口能在运行时动态更换为你私人定制的服务，如为了能载入只存在于内存的 DLL 插件，你可以私制一个实现了 IPluginLoader 核心接口的载入器，在运行时动态替换默认核心载入器，即实现了又享用 DLL 的便利又不带 DLL 的纯净系统。

（五）支持 C++的 DLL 插件

已测试可支持 VC++、BCB 等编译的 DLL 插件，各编译器做的 DLL 插件间通过框架可以无压力随意调用。

（六）同时支持 BPL 和 DLL 插件

Delphi 的 BPL 插件确实独具一格，特别是要把复杂界面元素放在插件中时，BPL 比 DLL 有更大优势。框架支持 BPL 和 DLL 两种插件共用混用模式，随你喜好使用。

（七）插件运行时动态更新/更换

这功能在不能停的服务端相当有用，框架可以不需停止运行即可

安全完成对指定插件的动态更新，哪怕插件正处于运行状态。

（八）插件懒加载，务求最少资源占用

插件框架提供总是、按需、自动管理三种方式载入插件，插件 DLL 只有在被呼叫时才载入内存，否则就只留在磁盘中，务求最少内存资源占用。

（九）支持插件手动式和配置式管理

提供全套 API 让你可在程序中手动获取、管理、卸载插件 DLL，同时也提供 xml 配置的方式自动管理插件。

（十）可能支持 Linux

框架只用到了 D7 的语言特性，因此，小改后可能可以在 Linux 下用 Lazarus 跑起来，但我是 Linux 菜鸟，没有发言权，所以只能说可能，各老鸟们自行上斧调改。

三、框架源码目录及文件

（一）目录说明

序号	目录名	说明
1	bin	可执行文件目录，在这儿有编译生成的各 Demo 可执行文件和插件管理器工具软件。
2	Demo	用于展示框架特性的各类 Demo 根据地
3	Doc	与框架相关的文档资料
4	Source	框架核心源文件区
5	Tools	各种方便框架使用的工具软件源码目录

（二）框架核心文件说明

序号	文件名	说明
1	ClassDefineForC.inc	为了支持 C++的插件，函数加了 stdcall，参数使用 PWideChar 的声明定义文件。
2	ClassImplementForC.inc	为了支持 C++的插件，这是匹配楼上单元的实现文件，只是做个桥梁，里面都是转手调用 WisdomFramework.pas 中的实现函数。
3	Def.inc	全局条件编译用的定义文件，每个 pas 文件一开头都要包含它，这样更改编译条件时，只需在这儿更改一下就好了，不用到处加条件，一般情况下，不要乱动这个文件。
4	MyString.inc	框架源码中所有涉及字符串部分都替换为 MyString 和 PMyChar 符号，最终具体是用的是 AnsiString、UnicodeString 还是 WideString 就在这文件中定义了。
5	ResultCode.pas	框架运行不抛出异常，只返回错误代码，此文件是错误代码和对应文字说明。
5	WisdomCoreInterfaceForC.pas	框架核心接口的声明，为了支持 C++另外进行的声明文件，一般不用看不用管，这些辛苦枯燥的工作由我做了。
6	WisdomCoreInterfaceForD.pas	根正苗红的框架核心接口的声明，框架都有哪些接口和服务函数，就看这文件！
7	WisdomFramework.pas	框架的核心实现类，精华所在，仅 1000 余行代码，朴素简单却有力量。
8	XmlConfig.pas	框架插件配置信息存取接口的实现，不想用 xml，想用 json 或其它类型记录配置的，重实现一个就行啦。

实际上，框架所有的核心精华就只在以上表格中标色的两个文

件，如要研读源码，就看这两个足够，其它只是饰品都可以不看。

四、框架设计原理

要淋漓尽致地发挥出框架的能量，对框架的设计及实现原理必须要清晰明了，因此这一节建议认真看，也有助于阅读和理解源码。

（一）框架核心三个主要概念

1．微内核

框架的实现核心非常小，小到恰到其是刚刚好地实现插件和接口服务的调度管理，除此之外，功能上不再增减一丝一毫，微内核的设计使得源码非常简单纯洁而稳定，框架就像初生的婴儿，在其上构建的插件就像后天的教育和引导，最终框架能成就什么，就像婴儿最终能成长为什么样的人，是由成长环境和教育决定，对应到框架则是由插件的功能和构织决定，因此，微内核的设计可令框架具备无穷的灵活扩展力。

2．一切皆接口

通过接口进行功能调用可令模块间获得高内聚低耦合的巨大好处是长期实践证明和众所周知的，因此框架当然设计为一切面向接口了，这是毫不迟疑的。

3．扩展点

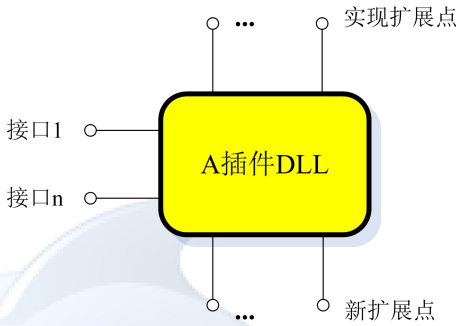
顾名思义，就是可以扩展的地方，在我们的生活中，电源插排就是一种“扩展点”，很多“扩展”（也就是电源插头）可以插在它上面，而电源插头自身又可以提供某种插口让别人可以接入，这样通过扩展点可形成一种无穷无尽的树形结构，也就获得无穷无尽的扩展能力。

（二）插件、接口服务、扩展点的关系

1、每个 DLL 称为一个插件。

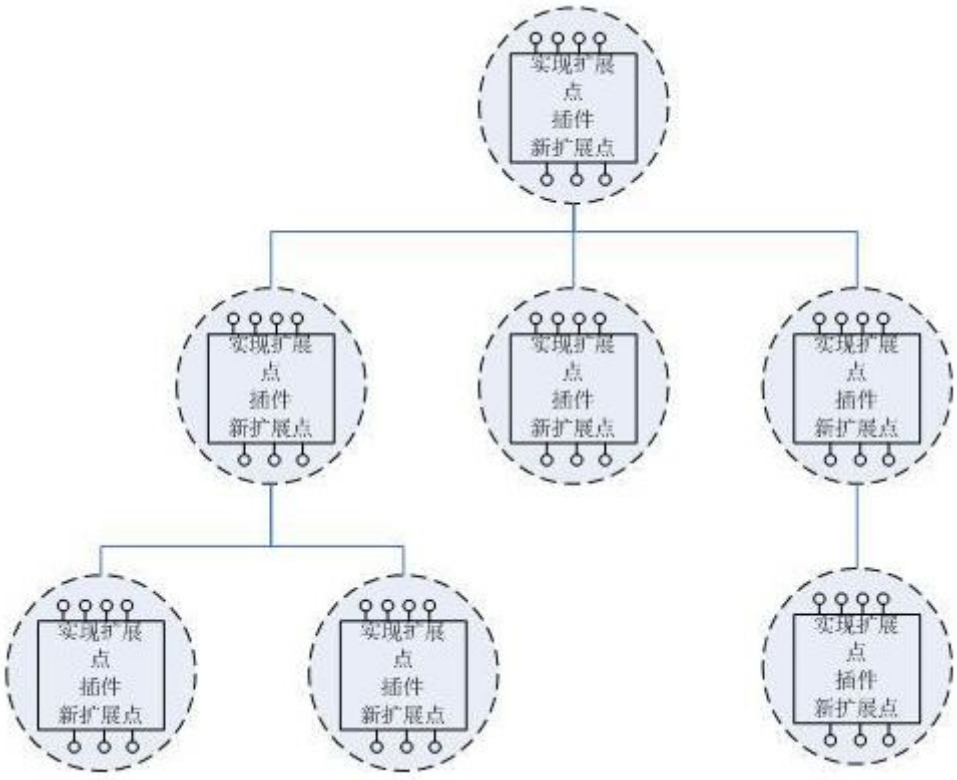
2、每个 DLL 可提供 0..N 个接口服务，包括新服务或实现别人扩展点。

3、每个 DLL 可向外声明 0..N 个扩展点，扩展点不能实例化。



以图示例为：

（三）插件 DLL 间通过扩展点互插连接成树状关系



一个插件系统就是众多“可供插入的地方”（扩展点）和“可以插入的东西”（扩展）共同组成的集合体。

（四）框架核心接口说明

框架的设计尽求简单直接,核心实现非常朴素,仅包含 6 个接口：

```
IServiceInfo = interface;

IPluginInfo = interface;

IServiceManager = interface;

IPluginLoader = interface;

IPluginManager = interface;

IConfig = interface;
```

各接口分别说明如下：

1、IServiceInfo

每个插件提供的服务由此接口保存描述信息，如接口 ID、版本、作者、接口类类型、是否单例、对象创建的回调函数等，框架即是根据此描述信息进行接口对象的创建和调度。1 个接口服务由 1 个 IServiceInfo 描述，一个插件 DLL 可以提供 0..n 个接口服务，因此，框架从 DLL 中获取服务接口时会创建多个 IServiceInfo，并由 IPluginInfo 负责进行生命期管理。

2、IPluginInfo

一个插件 DLL 由 IPluginInfo 负责描述，包括插件 ID、插件版本、作者等信息，此接口最重要的任务是负责 IServiceInfo 的生命周期管理，此外还保存一个 IPluginLoader 的引用。

3、IPluginLoader

实现这个接口的对象我一般称为插件载入器，专门负责插件 DLL

的载入和解析，每个 IPluginInfo 内部会创建一个此接口用于插件的载入和获取插件与服务信息，通过定制此接口即可实现各种各样规格文件的载入，因此，插件未必就是 DLL 文件哦，甚至一个“插件”有可能实际就只是一个 txt 文件哦。

4、IPluginManager

插件管理器接口，负责管理所有 IPluginInfo 的生命周期，负责插件的调度管理，如插件的载入、卸出、动态更新等。

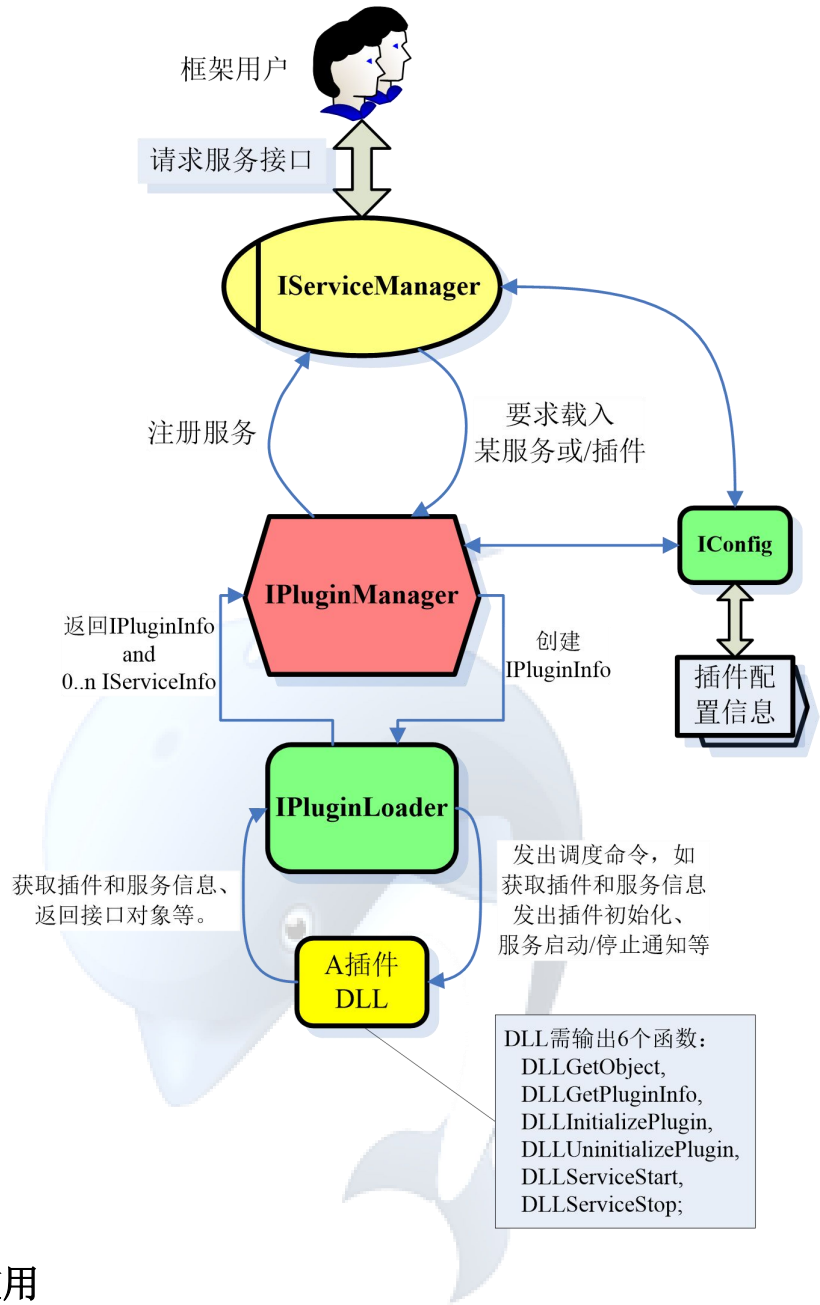
5、IServiceManager

服务接口管理器，负责所有插件中所提供的接口服务调度，框架用户即是通过此管理器获取到各种接口服务来使用，管理器提供了服务注册注销、启动停止服务、获取服务信息等功能，服务通过扩展点形成的树状相连即是在此管理器内部分析及拼接完成的哦。

6、IConfig

插件配置信息管理接口，框架默认提供以 XML 进行记录的实现，可以通过定制此接口记录到其它载体，如 JSON、数据库中。

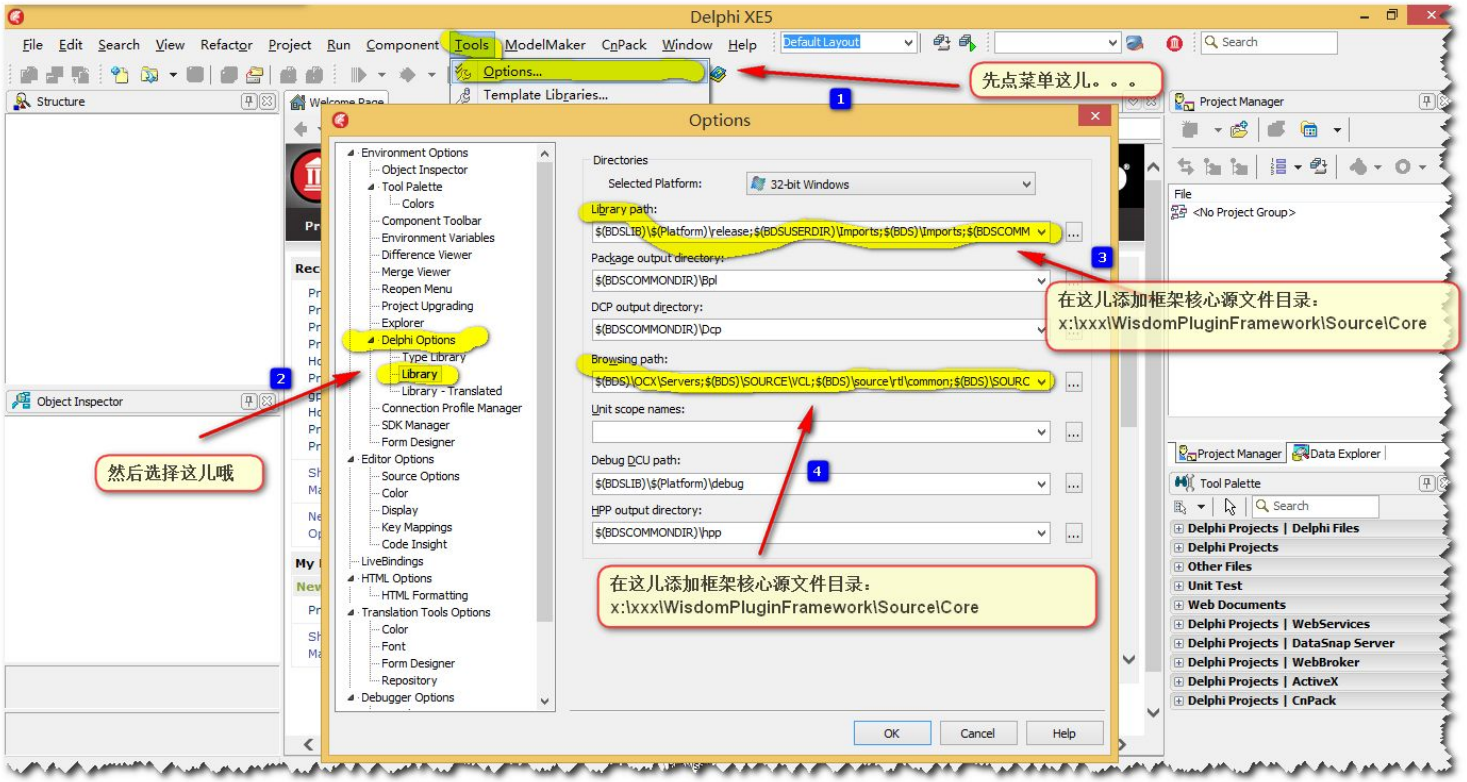
各接口间的调度关系图形化展示为：



五、框架使用

（一）编译环境配置

只需要把框架的源码目录设置到 Delphi 的搜索路径即可，可以设项目局部搜索路径，也可以设置全局搜索路径，我一般设置为全局的，只需以下几步，如下图：



（二）编译条件选项设置

1、VC++插件支持

框架默认只支持 Delphi 编译的各 DLL 插件，如果要开启 C++编译的 DLL 插件支持，需要打开编译条件选项“_USE_FOR_CPP”，请自行在 Host 程序的 Project->Options->Directories/Conditionals 下的 conditionals defines 中添加。

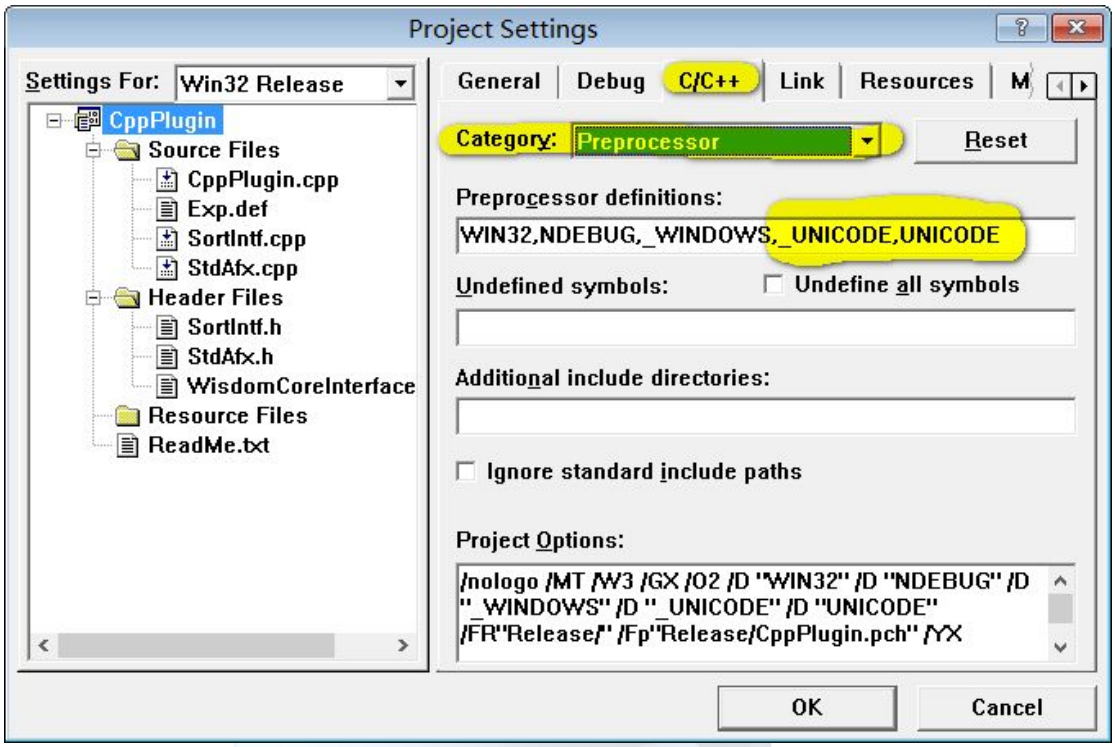
2、字符串类型选择

框架在字符串类型使用上支持三个选项，分别为_FORCE_ANSI、_FORCE_UNICODE 和_USE_WSTRING，默认使用_USE_WSTRING，这样可以支持 D7 以上各版本 Delphi 编译的 DLL 插件。

3、使用 VC++编译器

除非 Host 程序的编译选项设置为_FORCE_ANSI，否则需要设置编

译选项为 UNICODE 和_UNICODE，如下图：



（三）框架使用

1、对于 Host 程序

右键点击 Host 项目，在弹出菜单里选择“View Source”，然后在第一个单元中包含“WisdomFramework”单元的引用，当然不一定要放在此处，只是建议，你可以放在任何其它地方，如在主窗口的 Uses 单元中，只要 User 了此单元，框架即能在 Host 启动时完成初始化，但此时不会载入插件。

包含“WisdomCoreInterfaceForD”单元在需要使用框架核心接口的地方，该单元包含了一个 IServiceManager 接口引用的全局变量 GServiceManager，在框架初始化时赋予了初值，可直接使用，不要再另外创建。

在需要按配置载入插件的地方运行“TWisdomFramework.Run”，

框架即载入插件开跑，**如果插件中有界面元素载到 Host 显示的，在 Host 的主窗口退出前务必调用一次“TWisdomFramework.Stop”，否则会因资源不能正确释放出现内存访问错误。**

现在，即可通过 GServiceManager 根据插件 ID 自由获取各服务接口尽意使用了。

2、对于 DLL 插件

框架调用 DLL 插件前，会先获取插件信息，此时会调用 DLL 的输出函数 DLLGetPluginInfo（在 C++插件中为 GetPluginInfo），并传入已初始化的 IPluginInfo 接口，你需要在此时提供插件的描述信息和实现了的服务、扩展点信息，代码类如：

```
procedure DLLGetPluginInfo(const im: IPluginInfo); stdcall;
begin
  im.SetAuthor('IceAir'); //插件模块作者
  im.SetComment('插件说明信息');//插件简要说明
  im.SetPluginID('{A27A7515-4AA0-4BD9-B18F-FCB43831C277}');//插件 ID
  im.SetPluginName('extPointImp');//插件名称
  im.SetVersion('1.0.0');//插件版本
  im.SetUpdateHistory('2014 年 6 月 22 日 1:22 分……');//插件更新历史

  //提供的服务接口 1 信息
  with im.AppendServiceInfo(IID_ExtToolButton) do begin
    SetAuthor('IceAir'); //服务接口作者
    SetComment('扩展点 IToolBarExt 的一个具体实现。');//服务功能特点简要说明
    //SetIsExtendPoint(True); //如果是扩展点必须加此句
    SetSingleton(True);//服务是否要以单例模式运行
    SetServiceClass(TExtToolButton);//服务接口的实现类，Delphi 下专用
    SetServiceName('IExtToolButton');//服务名称
    SetImplementServiceID(IID_ToolBarExt);//服务实现的扩展点 ID
  end;
  //提供的服务接口 2 信息
  with im.AppendServiceInfo(IID_ExtToolButton) do begin
    ……
  end;
end;
```

此处插件提供的 Singleton（单例）要求主要用于配置插件 XML 信息时参考，但可以在 XML 中改动，框架最终只以 XML 中配置的为准。

框架获取插件信息成功后，会调用函数 DLLInitializePlugin，并传入 IServiceManager 引用，你需要在此处保存该接口的引用，同时可以做一些插件模块正常运行所需要的初始化操作，该函数被调用的时机是：插件 DLL 已被框架载入，但未激活(包括插件中的服务项)。

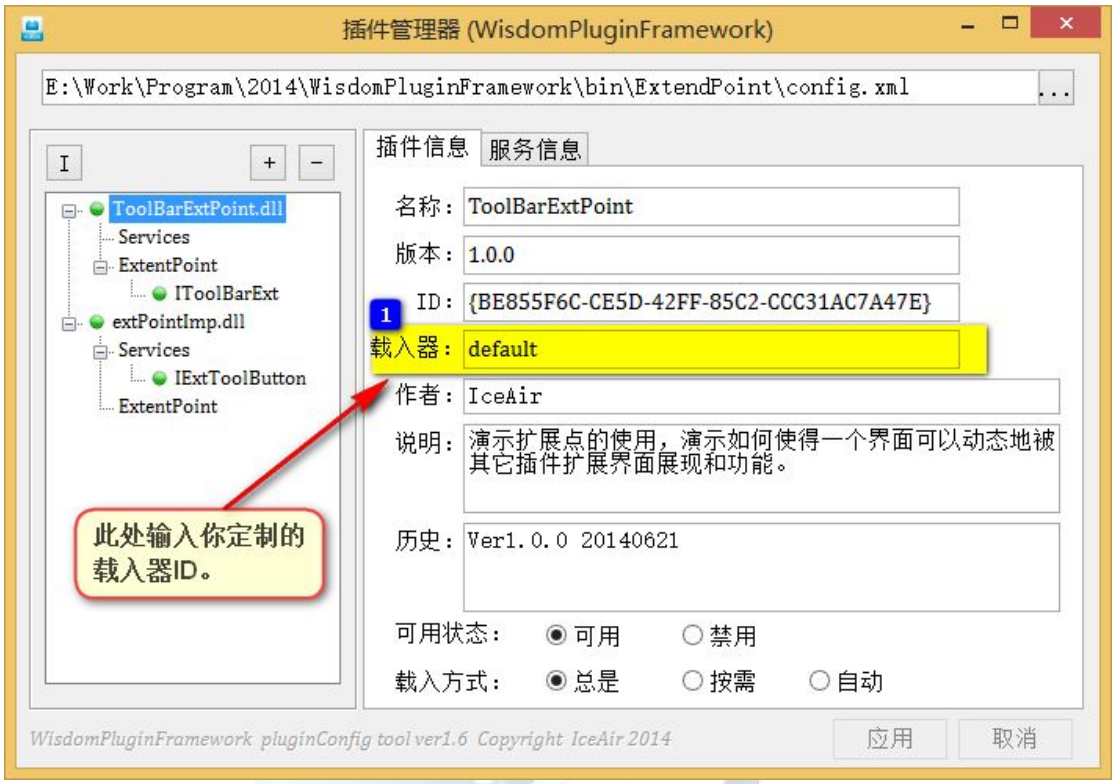
插件初始化后，框架会注册插件提供的服务信息，每注册并激活一个服务信息后会触发 DLL 插件的输出函数 DLLServiceStart(注意，包括自己的服务注册和别的 DLL 插件的服务注册均会触发)，你可以在此灵活地做些你想要的任何处理。

框架在任何一个服务停止前会调用输出函数 DLLServiceStop，并传入将被停止的服务 ID，**如果你引用并正在使用此 ID 的服务，必须立刻无条件释放此服务的引用，框架不检测谁还在持有该服务，因而在全部插件通知一遍后即关停服务并卸出内存，无条件立刻释放接口引用是必须遵守的规则，如果你想框架能够支持动态更新 DLL 的插件的话。**

最后一个输出函数 DLLGetObject 一般用于 C++的插件中，当框架发现 IServiceInfo 的服务描述信息中，既没有记录 TXxClass，又没有 TCreateObjectFunc 的回调函数时，就会调用 DLLGetObject 并传入服务 ID，意思是说，请你创建该服务 ID 的接口对象并返回给我。

以上输出函数是框架默认的实现标准，如果你定制实现了你的 IPluginLoader，可以不用这一套，DLL 的工作结构完全由你决定。

对于必须由你定制的载入器载入的插件，在使用 PluginConfigTool 配置插件时，要记得必须指定载入器 ID 哦，如图：



六、关于 Demo

各 Demo 演示界面上均有该 Demo 特点的说明，此处就不再敷述了。

七、期望

个人精力难以持续支持框架紧跟 Delphi 步伐发展，个人经验、灵感和智力实也非常有限，因此殷诚恳切期请各位觉此框架确有价值的高手，能够反馈改进建议，最好能够出手斧正精刻，共同完善框架，令其成为造福大众之精品。

相信合集众人之能量，必能闪耀璀璨迷人的智慧光芒！