DirectShow 专业著作

DirectShow 开发指南

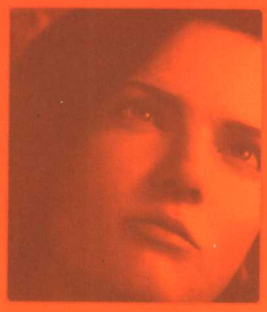
99%

实用性、综合性

- 深入剖析DirectShow应用架构
- 掌握先进的多媒体应用开发技术
- 全面、深刻、通俗易懂

陆其明 编著
金邦飞 审校

清华大学出版社



- 本书完全忠实于 DirectX SDK 的帮助文档以及基类源代码，并结合作者多年的实践，经过提炼而成；涉及的内容几乎涵盖了在 Windows 平台上使用 DirectShow 进行 C++ 编码的方方面面
- 本书条理清晰，实用性强，适合广大的流媒体应用开发人员，以及对 Windows 平台上多媒体处理感兴趣的编程爱好者学习和参考

ISBN 7-302-07650-2

9 787302 076506 >

定价：38.00 元

封面设计：谭洁红

读者联系电话：（010）82896448，82896450

网　址：http://www.khp.com.cn

E-mail：khp@khp.com.cn

DirectShow 开发指南

陆其明　编著
金邦飞　审校

清华大学出版社
北京

内 容 简 介

本书以 DirectX SDK 9.0 版为蓝本，内容几乎涵盖了在 Windows 平台上使用 DirectShow 进行 C++编码的方方面面。全书共分 4 个部分。第 1 部分详细介绍了 DirectShow 的基础知识。第 2 部分重点讨论了 Filter 的开发，以及 DirectShow 应用程序的开发，包括目前非常流行的音视频采集、数码摄像机的支持、非线性编辑等应用。第 3 部分深入分析了 DirectShow SDK 提供的部分典型源代码。第 4 部分结合作者个人的一些开发实践，通过案例和开放源码分析，进一步介绍 DirectShow 的实务应用。

本书内容丰富，条理清晰，实用性强。适合广大的流媒体应用开发人员，以及对 Windows 平台上多媒体处理感兴趣的编程爱好者学习、参考。

图书在版编目（CIP）数据

DirectShow 开发指南 / 陆其明编著.
—北京：清华大学出版社，2003
ISBN 7-302-07650-2

Ⅰ. D… Ⅱ. 陆… Ⅲ. 多媒体—软件工具, DirectShow Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字（2003）第 103827 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社总机：010-62770175　　**客户服务**：010-62776969

组稿编辑：科海
文稿编辑：安靖　潘秀燕
封面设计：谭洁红
版式设计：杨月静
印 刷 者：北京市耀华印刷有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　　**印张**：23.25　　**字数**：565 千字
版　　次：2003 年 12 月第 1 版　　2004 年 3 月第 2 次印刷
书　　号：ISBN 7-302-07650-2/TP・5612
印　　数：5001~6000
定　　价：38.00 元

前　言

流媒体的处理以其复杂性和技术性而闻名，一向广受业界的关注。特别是伴随着因特网的普及，流媒体在网络上广泛应用，怎样使流媒体的处理变得简单而富有成效逐渐成为了焦点问题。选择一种合适的应用方案，将会事半功倍。此时，微软公司的 DirectShow 给了我们一个不错的选择。

本书以 DirectX SDK 9.0 版为蓝本，内容几乎涵盖了在 Windows 平台上使用 DirectShow 进行 C++编码的方方面面。全书共分 4 个部分。第 1 部分详细介绍了 DirectShow 的基础知识。第 2 部分重点讨论了 Filter 的开发，以及 DirectShow 应用程序的开发，包括目前非常流行的音视频采集、数码摄像机支持、非线性编辑等应用。第 3 部深入分析了 DirectShow SDK 提供的部分典型源代码例子。第 4 部分结合作者个人的一些开发实践，通过案例和开放源码分析，进一步介绍 DirectShow 的实务应用。

促使我写作这本书的原因是多方面的。首先，我自己学习 DirectShow 的时候并不轻松，因为当时除了 DirectX 的帮助文档外，找不到更多的资料，国内市场上也找不到一本有关 DirectShow 的参考书；而且当时国内从事相关开发的公司或个人甚少，难得有交流的机会。其次，因为我在网上发表了很多有关 DirectShow 的文章，很多朋友认为我是这方面的“高手”。（其实我不是所谓的高手，但我很愿意拿我的平生所学来与大家交流。）于是，经常有人向我咨询如何学习 DirectShow，以及要求我协助解决 DirectShow 应用中的很多实际问题。看到 DirectShow 越来越受欢迎，我一边兴奋，一边也萌发了这样一个念头：“是出一本 DirectShow 书的时候了”。

本书提供 DirectX SDK 9.0 的安装程序，以及书中涉及的所有例子的源代码，下载地址为 http://hqtech.nease.net/DSDeveloper/DsBkSource.zip。

本书是集体智慧的结晶。我首先要感谢彭木昌先生、黄重来先生，以及出版社的夏非彼老师、李才应编辑；没有这些朋友的鼓励和帮助，就不会有这本书的问世。本书初稿完成后，还特邀了敏递软件（上海）有限公司的工程部经理金邦飞先生，进行了严格的技术审校。值得一提的是，本书的封面由我的爱人谭洁红设计。感谢她出色的工作，以及长期以来对本人无微不至的关怀。由于时间短促，再加之本人的水平有限，书中的错误以及疏漏之处在所难免，望广大专家、同行批评指正。

陆其明

2003 年 11 月于上海

目　录

第 1 部分　DirectShow 基础知识

第 2 部分　DirectShow 开发与应用

第 3 部分 SDK 典型源码分析

第 4 部分 开放源码分析

第1部分 DirectShow基础知识

第1章 系统概述

1.1 DirectX大家族

DirectX软件开发包是微软公司提供的一套在Windows操作平台上开发高性能图形、声音、输入、输出和网络游戏的编程接口。微软将DirectX定义为“硬件设备无关性”，即使用DirectX可以用与设备无关的方法提供设备相关的（高）性能。

事实上，DirectX已经成为一种标准，它可以为应用程序（特别是游戏）开发人员和硬件厂商之间的关系“解耦”。DirectX标准的建立，可以为硬件开发提供策略，硬件厂商不得不按照这一标准进行产品改进；同时，通过使用DirectX所提供的接口，开发人员可以尽情地利用硬件可能带来的高性能，而无需关心硬件的具体执行细节。

另外，DirectX采用了组件对象模型（COM）标准，因此不同对象的版本可以有不同的接口，这使用DirectX开发的程序即使在未来也能得到完全的兼容和支持。

DirectX是一项卓越的技术。那么，它为什么称为DirectX呢？其实也不难理解，Direct是直接的意思，X可以代表很多东西，合在一起就是具有共性的一组东西（这个共性就是直接）。DirectX是一个大家族，并且随着DirectX版本的不断更新，家族成员也在不断地发展壮大。

下面是DirectX 9.0家族的所有成员：

- DirectX Graphics：集成了以前的DirectDraw和Direct3D技术。DirectDraw主要负责2D加速，以实现对显卡内存和系统内存的直接操作；Direct3D主要提供三维绘图硬件接口，它是开发三维DirectX游戏的基础。
- DirectInput：主要支持输入服务（包括鼠标、键盘、游戏杆等），同时支持输出设备。
- DirectPlay：主要提供多人网络游戏的通信、组织功能。
- DirectSetup：主要提供自动安装DirectX组件的API功能。
- DirectMusic：主要支持MIDI音乐合成和播放功能。
- DirectSound：主要提供音频捕捉、回放、音效处理、硬件加速、直接设备访问等功能。
- DirectShow：为在Windows平台上处理各种格式的媒体文件的回放、音视频采集等高性能要求的多媒体应用，提供了完整的解决方案。
- DirectX Media Objects：DirectShow Filter的简化模型，提供更方便的流数据处理方案。

其实，DirectShow 开始并不是 DirectX 家族中的一员，它是经过 DirectX 6.0 中的 DirectX Media 发展而来的。DirectShow 集成了 DirectX 家族中其他成员（如 DirectDraw、DirectSound 等）的技术，可以说是 DirectX 中的一位“集大成者”。经过几个版本的发展，DirectShow 架构日趋成熟。而 DirectX Media Objects 是从 DirectX 8.1 的 DirectShow 中分离出来的，成为了另一种高效率的流数据处理解决方案。

下面将介绍 DirectShow 系统，使大家能够对 DirectShow 有一个总体了解。

1.2 DirectShow 简介

为什么需要 DirectShow？DirectShow 到底能够做什么？带着这两个问题，我们先一起来看多媒体应用开发所面临的挑战：

（1）多媒体数据量巨大，应如何保证数据处理的高效性；

（2）如何让音频和视频时刻保持同步；

（3）如何用简单的方法处理复杂的媒体源问题，包括本地文件、计算机网络、广播电视以及其他一些数码产品等；

（4）如何处理各种各样的媒体格式问题，包括 AVI、ASF、MPEG、DV、MOV 等；

（5）如何支持目标系统中不可预知的硬件。

DirectShow 的设计初衷就是尽量要让应用程序开发人员从复杂的数据传输、硬件差异、同步性等工作中解脱出来，总体应用框架和底层工作由 DirectShow 来完成，这样，基于 DirectShow 框架开发多媒体应用程序就会变得非常简单！

1.2.1 DirectShow 系统

如图 1.1 所示，图中央最大的一块即是 DirectShow 系统，虚线以下是 Ring 0 特权级别的硬件设备，虚线以上是 Ring 3 特权级别的应用层。DirectShow 系统位于应用层中。它使用一种叫 Filter Graph 的模型来管理整个数据流的处理过程；参与数据处理的各个功能模块叫做 Filter；各个 Filter 在 Filter Graph 中按一定的顺序连接成一条“流水线”协同工作。

按照功能来分，Filter 大致分为 3 类：Source Filters、Transform Filters 和 Rendering Filters。Source Filters 主要负责获取数据，数据源可以是文件、因特网计算机里的采集卡（WDM 驱动的或 VFW 驱动的）数字摄像机等，然后将数据往下传输；Transform Filters 主要负责数据的格式转换，例如数据流分离/合成、解码/编码等，然后将数据继续往下传输；Rendering Filters 主要负责数据的最终去向——将数据送给显卡、声卡进行多媒体的演示，或者输出到文件进行存储。

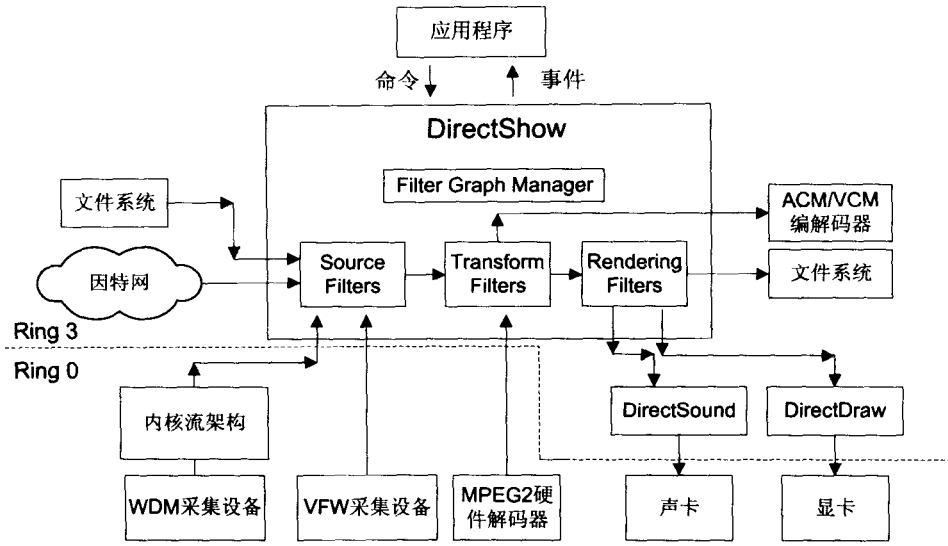


图 1.1　DirectShow 系统

1.2.2　播放第一个媒体文件

DirectShow 提供了大量的 Filter 用以支持最基本的应用。根据实际需要，也可以定制自己的 Filter（如何开发自己的 Filter 将在后续章节逐步介绍）。它的最基本的应用莫过于回放一个媒体文件。如图 1.2 所示是一条典型的 AVI 文件回放的链路。

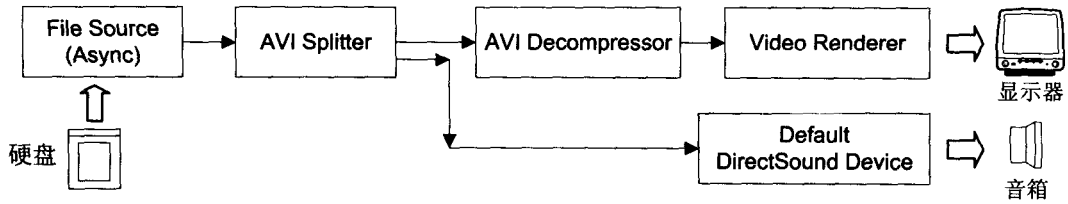


图 1.2　本地 AVI 文件的回放链路

其中，箭头方向即是数据的流向。对照图 1.1 不难发现，File Source（Async）属于 Source Filters，它用于管理硬盘上指定的播放文件，并根据 AVI Splitter 的要求提供数据。AVI Splitter 和 AVI Decompressor 属于 Transform Filters，其中，AVI Splitter 负责向 File Source（Async）索取数据，并将取得数据的音频和视频进行分离，然后分别从各自的输出 Pin 输出；AVI Decompressor 负责视频的解码。Video Renderer 和 Default DirectSound Device 属于 Rendering Filters，其中，Video Renderer 负责向视频窗口输出图像；Default DirectSound Device 负责同步播放声音。

提示

如果你已经安装了 DirectX SDK（可以到微软公司的网站上去下载），运行 SDK 安装目录下的 bin\DXUtils\graphedt.exe，然后选择菜单项 File | Render Media Filter，在弹出的对话框中选择一个媒体文件，可以看到一条回放文件的 Filter 链路。选择菜单项 Graph | Play，开始播放 Filter Graph。

1.3 COM 编程基础

DirectX 采用了 COM 标准。而 DirectShow 是一套完全基于 COM 的应用系统。要想深入学习 DirectShow，掌握一些 COM 编程的基础知识是必不可少的。

对于 DirectShow 应用程序开发人员来说，对 COM 知识的了解并不要求很高。因为 DirectShow 应用程序实际上是一种 COM 组件的客户程序，只是 COM 组件的“使用”问题。这些问题包括如何创建 COM 组件、如何得到组件对象上的接口以及调用接口方法、如何管理组件对象（即需要熟悉 COM 的引用计数机制）等。下面的代码是最一般的步骤。

```
CoInitialize(NULL);                    // COM 库初始化
// Do something...
// ...

IUnknown *pUnk    = NULL;
IObject *pObject = NULL;
// 创建组件对象
HRESULT hr = CoCreateInstance(CLSID_Object,
CLSCTX_INPROC_SERVER, NULL, IID_IUnknown, (void**)&pUnk);
if (SUCCEEDED(hr))
{
       // 查询得到组件对象上的接口
     hr = pUnk->QueryInterface(IID_IOjbect,(void**)&pObject);
     if (SUCCEEDED(hr))
     {
       // 调用接口方法
         pObject->SomeMethod();
         pObject->Release();
       }
       pUnk->Release();
}
// ...
CoUninitialize();          // 释放 COM 库使用的资源
```

而对于 Filter 开发人员来说，需要掌握的 COM 知识就要多一点。因为 Filter 本身是一种 COM 组件，开发 Filter 牵涉到了 COM 组件的“实现”问题。

那么，COM 组件到底是什么？COM 本身只是一种规范，而不是实现。但是当使用 C++来实现时，COM 组件就是一个 C++类，而接口都是纯虚类。可以用如下的 C++代码来简单描述一个 COM 组件。

```
class Ifunction
{
public:
virtual Method1(...) = 0;
virtual Method2(...) = 0;
    // ...
};
```

```
class MyObject : public Ifunction
{
public:
virtual Method1(...){...}
virtual Method2(...){...}
    // ...
};
```

其中，Ifunction 就是我们常说的接口，而 MyObject 就是 COM 组件。

COM 规范规定，任何组件或接口都必须从 IUnknown 接口中继承而来。IUnknown 定义了 3 个重要函数，分别是 QueryInterface、AddRef 和 Release。其中，QueryInterface 负责组件对象上的接口查询，AddRef 用于增加引用计数，Release 用于减少引用计数。引用计数是 COM 中的一个非常重要的概念，它很好地解决了组件对象的生命周期问题，即 COM 组件到底在什么时候被销毁，以及由谁来销毁的问题。

提示　COM 组件一般会采用一个“自销毁”的策略。可以看到很多 IUnknown::Release 函数的实现：当引用计数器值为 0 时，组件对象就会调用 delete this 指令。

除了 IUnknown 接口外，还有另外一个重要的接口，即 IClassFactory。COM 组件实际上是一个 C++类，对于组件的外部使用者来说，这个类名一般不可知，那么如何创建这个类的实例？由谁来创建？COM 规范规定，每个组件都必须实现一个与之相对应的类工厂（Class Factory）。类工厂也是一个 COM 组件，它实现了 IClassFactory 接口。在 IClassFactory 的接口函数 CreateInstance 中，才能使用 new 操作生成一个 COM 组件类对象实例。

COM 组件有 3 种类型：进程内组件、本地进程组件和远程组件。Filter 一般是一种进程内组件，以 DLL（动态链接库）的形式提供服务。

接下来看一下怎么来实现一个 COM 组件。如图 1.3 所示是 COM 组件的创建过程。

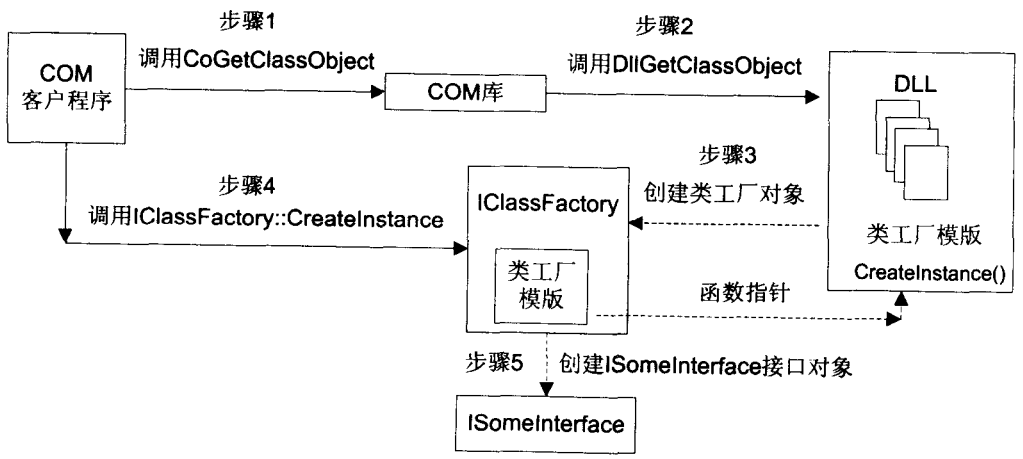


图 1.3　COM 组件的创建过程

下面的代码是 CoCreateInstance 函数实现的伪代码。

```
CoCreateInstance(...)
{
// Do something...
// ...

  IClassFactory *pClassFactory = NULL;
  CoGetClassObject(CLSID_Object, CLSCTX_INPROC_SERVER, NULL,
  IID_IClassFactory, (void **)&pClassFactory);pClassFactory->Create-
  Instance(NULL, IID_IUnknown, (void**)&pUnk);pClassFactory->Release();

  // ...
}
```

每个 COM 组件都使用一个 GUID 来惟一标识。当创建一个 COM 组件时，总是首先通过这个 GUID（上例中为 CLSID_Object）调用 CoGetClassObject 来获得创建这个组件对象的类工厂。然后调用类工厂的接口方法 IClassFactory::CreateInstance，就能真正地创建 CLSID_Object 标识的组件对象了。下面再给出一段 CoGetClassObject 函数以及其他相关函数实现的伪代码。

```
CoGetClassObject(...)
{
// 通过查询注册表 CLSID_Object 得知组件 DLL 文件路径
// 装入 DLL 库（调用 LoadLibrary）
// 使用函数 GetProcAddress(...)得到 DLL 中函数 DllGetClassObject 的函数指针
// 调用 DllGetClassObject 得到类工厂对象指针
}

DllGetClassObject(...)
{
// ...
// 创建类工厂对象
CFactory* pFactory= new CFactory;
// 查询得到 IClassFactory 指针
pFactory->QueryInterface(IID_IClassFactory, (void**)&pClassFactory);
pFactory->Release();
// ...
}

CFactory::CreateInstance(...)
{
// ...
// 创建 CLSID_Object 对应的组件对象
CObject *pObject = new CObject;
pObject->QueryInterface(IID_IUnknown, (void**)&pUnk);
pObject->Release();
// ...
}
```

其中，DllGetClassObject 是在 COM 组件 DLL 中必须实现的一个（导出）函数，它的作用是根据指定的组件 GUID 创建相应的类工厂对象，并返回这个类工厂的 IClassFactory 接

口；CreateInstance 是 IClassFactory 接口的一个接口方法，负责最终创建组件对象实例；CObject 就是我们的 COM 组件类，它实现了 COM 框架以外的真正的组件功能。

一个典型的自注册 COM 组件 DLL 所必需的 5 个（导出）函数如下：

- DllMain：DLL 的入口函数（DirectShow 实现的是 DllEntryPoint）；
- DllGetClassObject：用于获得类工厂指针；
- DllCanUnloadNow：系统空闲时会调用这个函数，以确定是否可以卸载 DLL；
- DllRegisterServer：将 COM 组件注册到注册表中；
- DllUnregisterServer：删除注册表中 COM 组件的注册信息。

提示　其实，SDK 基类源代码中已经实现了 Filter 中的 COM 基本特性。这样，在 SDK 基础上开发 Filter 就没有那么费劲了。了解 COM 组件的创建过程关键是为了理解 Filter 框架的实现。

第 2 章　Filter 原理

2.1　Filter 概 述

Filter 是 DirectShow 中最基本的概念。DirectShow 使用 Filter Graph 来管理 Filter（管理者叫做 Filter Graph Manager）。Filter Graph 是 Filter 的“容器”，而 Filter 是 Filter Graph 中的最小功能模块。

Filter 一般由一个或多个 Pin 组成，Filter 之间通过 Pin 相互连接，构成一条顺序的链路。在第 1 章中曾经提到，Filter 根据实现功能的不同大致可分为 3 类：Source Filters、Transform Filters 和 Rendering Filters。另一种实用的判别方法是根据 Filter 包含的输入 Pin 或者输出 Pin 的数量来判断。

如图 2.1 所示，仅含有输出 Pin，没有输入 Pin 的 Filter 为 Source Filter；既有输入 Pin 又有输出 Pin 的 Filter 为 Transform Filter；仅有输入 Pin，没有输出 Pin 的 Filter 为 Rendering Filter。

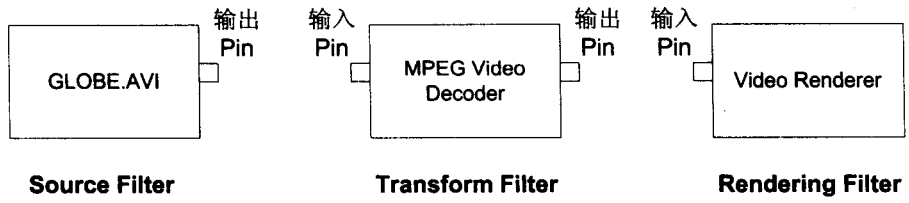


图 2.1　3 种 Filter 类型

Filter 是一种 COM 组件。为了实现在 Filter Graph 中的统一操作，每个 Filter 上都至少实现了 IBaseFilter 接口。实现 Filter 的文件一般是一个 DLL，扩展名可以是.dll，但更多的时候是.ax。跟普通的 COM 组件一样，Filter 的创建是通过 API 函数 CoCreateInstance 来完成的，代码如下：

```
STDAPI CoCreateInstance(
  REFCLSID rclsid,      // Class identifier (CLSID) of the object
  LPUNKNOWN pUnkOuter, // Pointer to controlling IUnknown
  DWORD dwClsContext,  // Context for running executable code
  REFIID riid,         // Reference to the identifier of the interface
  LPVOID * ppv         // Address of output variable that receives
                       // the interface pointer requested in riid
);
```

其中，参数 rclsid 指定要创建的 Filter 的 CLSID；因为绝大多数情况下创建的 Filter 不是被“聚合”（Aggregation）的，所以 pUnkOuter 指定为 NULL；dwClsContext 可以指定为 CLSCTX_INPROC_SERVER，以创建进程内组件对象；riid 在创建 Filter 成功后获得的接口的 ID，一般为 IID_IBaseFilter，也可以是其他特殊的接口；ppv 用于获得接口对象的指针。