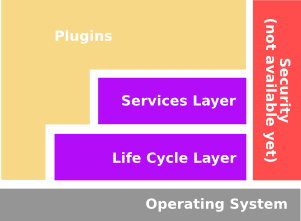
1. **CTK介绍**
2. 简述

CTK Plugin Framework 可以简单的描述为 C++ 的动态组件系统。

1. 设计

CTK Plugin Framework 的设计（参考：[设计文档](http://www.commontk.org/index.php/Documentation/PluginFramework_DesignDoc)）受到了 OSGi（Java 的动态组件系统）的极大启发，并且提供了一种开发模型，其中应用程序（动态地）由许多不同的（可重用）组件组成。该模型允许通过服务进行通信，服务是特定于组件之间的对象。

框架的分层模型，如下图所示：



插件（Plugins）：是由开发人员创建的 CTK 组件。

服务层（Services Layer）：通过为 C++ 对象提供一个 publish-find-bind 模型，以动态方式连接插件。

生命周期层（Life Cycle Layer）：用于安装、启动、停止、更新和卸载插件的 API。

安全性（Security）：处理安全方面（目前还不可用）

1. 插件

CTK 插件是它的核心，一个基于 Qt Plugin 系统的共享库。此外，CTK 库中的符号默认隐藏在所有平台上。这是迈向模块化的第一步，它关系到保持本地化和不共享。你分享的东西越少，错误的假设就会越少。然而，如果没有共享，就无法进行协作。CTK 插件通常只共享符号（类和函数）来支持 CTK 的服务模型。

1. 服务

C++ 中的协作模型通常使用工厂模式来实现。不同的工具包使用不同的模式和 API 来访问这样的工厂。通常，影响工厂使用哪一种实现的决策是非常重要的。此外，实现代码通常不能通告其可用性，用户也不能列出可能的实现，并选择最合适的实现。工厂通常也不是动态的，一旦一个实现的实例被注册，就不能被撤销。最后，如果使用了许多不同的工厂，则没有对代码绑定的实现进行集中的概述。

解决这些问题的一个解决方案是 CTK service registry。一个插件可以创建一个对象，并在一个或多个接口（接口通常是一个只有纯虚方法的 C++ 类）下使用 CTK service registry 注册它。其他插件可以要求 registry 列出在特定接口下注册的所有服务（对象）。一个插件甚至可以等待一个特定的服务出现，然后收到回复。

因此，一个插件可以注册一个服务，它可以获得一个服务，并侦听服务的出现或消失。任意数量的插件可以在相同的接口下注册服务，并且任意数量的插件都可以得到相同的服务，如下图所示。



如果多个插件在同一个接口下注册对象，则可以通过其属性进行区分。每个服务注册都有一套标准的自定义属性。你可以使用一个富有表现力的过滤语言来仅选择感兴趣的服务。属性也可以被用于应用程序级的其他角色。

由于服务是动态的，一个插件可以决定从 registry 中退出其服务，而其他插件仍然在使用它。使用这样一个服务的插件必须确保它们不再使用服务对象，并删除任何指向它的指针。乍一看，这听起来可能很复杂，但是使用像 ctkServiceTracker这样的帮助类，以及像 Declarative Services（待开发）这样的框架，可以使这个过程变得非常简单，并且所获得的优势相当大。服务的动态特性允许在其他插件保持功能的情况下动态地安装和卸载插件。它也更贴近现实世界的问题，因为大多数情况下，这些问题不是静态的。例如，在分布式环境中，服务可以对连接端点进行建模，如果连接到远程计算机，则可以取消该服务。此外，动态解决了初始化问题。使用 CTK 插件的应用程序不需要在插件中使用特定的启动顺序。

尽管 service registry 接受任何基于 QObject 的对象作为服务，但是实现重用的最佳方式是在（标准的）接口下注册这些对象，从而将实现与客户端代码解耦。因此，CTK Plugin Framework 提供了许多标准接口，这些接口与 [OSGi Service Platform Release 4 Compendium Specification](https://www.osgi.org/developer/specifications/) 发布的服务规范紧密相关。这些标准服务在规范和这个 Wiki （https://www.osgi.org/developer/specifications/）中都有详细的描述。

1. 部署

CTK Plugin Framework 可被用作你的所有应用逻辑的主容器，但它也可以被嵌入到你的现有框架中。该框架的管理是标准化的，它提供了一个简单的 API，允许插件安装、启动、停止和更新其他插件，以及列举插件和它们的服务使用。这个 API 可以被所谓的 management agents 用来控制插件框架。Management agents 可以是多种多样的，像命令 shell、丰富的图形桌面应用程序、或 AJAX 应用程序。

1. 优点  
   CTK Plugin Framework 基于 OSGi 原理和 API。正因如此，它继承了一种非常成熟和完全设计的组件系统，这在 Java 中用于构建高度复杂的应用程序。它将这些好处带给了本地（基于 Qt 的）C++ 应用程序。

* 降低复杂性   
  使用 CTK Plugin Framework 开发意味着插件的开发。它们隐藏了内部实现，并通过已经定义好的服务来和其他插件通信，隐藏内部实现意味着以后可以自由地更改实现。这不仅有助于 bug 数量的减少，还使得插件的开发变得更加简单，因为只需要实现已经定义好的一定数量的功能接口即可。
* 复用   
  标准化的组件模型，使得它在应用程序中可以非常简单的使用第三方组件。
* 现实情况   
  CTK Plugin Framework 是一个动态框架。它可以动态地更新插件和服务。现实世界中有很多场景都和动态服务模型相匹配。因此，应用程序可以在其所属的领域中强有力的重用 service registry（注册，获取，具有展现力的过滤语言列表，等待服务的出现或消失）。这不仅可以保存正在实现的代码，而且还对全局提供了可见性、调试工具和更多的功能而不是需要实现过时的问题。在这样的动态环境下写代码听起来似乎是个噩梦，但是幸运的是，它有主要的支持类和框架，如果不是这样，那将非常痛苦。
* 开发简单   
  CTK Plugin Framework 不仅仅是组件的标准，它还指定了如何安装和管理组件。这个 API 可以被插件用来提供一个 management agent。这个 management agent 可以非常简单，如命令 shell、图形桌面应用程序、Amazon EC2 的云计算接口、或 IBM Tivoli 管理系统。标准化的管理 API 使得在现有和未来的系统中集成 CTK Plugin Framework 变得非常容易。
* 动态更新   
  使用的 OSGi 组件模型是一个动态模型。插件可以在不需要停止整个系统的情况下被安装、启动、停止、更新和卸载。
* 自适应   
  OSGi 的组件模型设计原则是允许组件的混合和匹配。这就需要组件的依赖关系被指定，并且需要组件在一个并不总是可用的可选的依赖项环境中生存。service registry 是一个动态注册的软件包，其中插件可以注册、获取和监听服务。这种动态服务模型允许插件找出系统中可用的功能，并调整它们所能提供的功能。这使得代码更加灵活，并且能够更好地适应变化。
* 透明性   
  插件和服务是 CTK 插件环境中的一等公民。管理 API 不但提供了对插件的内部状态的访问，而且也提供了如何去和其他插件做对接。有部分应用为了调试一个确切的问题而被停止，或者引入软件插件。
* 版本控制   
  在 CTK Plugin Framework 中，所有的插件都是经过严格的版本控制，只有能够协作的插件才会被连接在一起。
* 简单   
  CTK 插件 API 非常简单。核心 API 不到 25 个类。这个核心 API 足以编写插件、安装、启动、停止、更新和卸载，并包含所有的监听类。
* 懒加载   
  懒加载是软件中一个很好的点，OSGi 技术有很多的机制来保证只有当类真正需要的时候才开始加载他们。例如，插件可以用饿汉式启动，但是它们也可以被配置，仅当另一个插件使用它们时才启动。服务可以被注册，但只有在使用时才创建。这些懒加载场景，可以节省大量的运行时成本。
* 非独占性   
  CTK Plugin Framework 不会接管整个应用程序，你可以选择性地将所提供的功能暴露给应用程序的某些部分，或者甚至可以在同一个进程中运行多个框架实例。
* 非侵入   
  在一个 CTK 插件环境中，不同插件均有自己的环境设置。他们可以使用任何设施，框架对此并无任何限制。CTK 服务不需要任何特殊的接口，甚至每一个 QObject 都可以充当一个服务，每一个类（也包括非 QObject）都可以充当一个接口。

1. **CTK动态库编译**
2. 依赖（基于Windows 64bit环境）

* Qt

版本最低要求≥5.0.1，为了统一，目前采用最新版本Qt5.9.2，公司内网Qt安装包的路径为：

* VS2015
* CMake

版本最低要求≥2.8.9，为了统一，目前采用最新版本CMake3.9.0，公司内网CMake安装包的路径为：

* CTK 源码包

不作版本要求，为了统一，目前采用最新版本CTK，公司内网CTK源码的路径为：

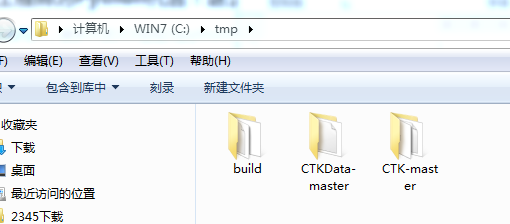
* CTKData数据文件

在编译CTK源码时，需要到外网下载CTK数据，而我是通过github离线下载的。公司内网CTKData数据的存放路径为（后面编译出现错误时会用到）：

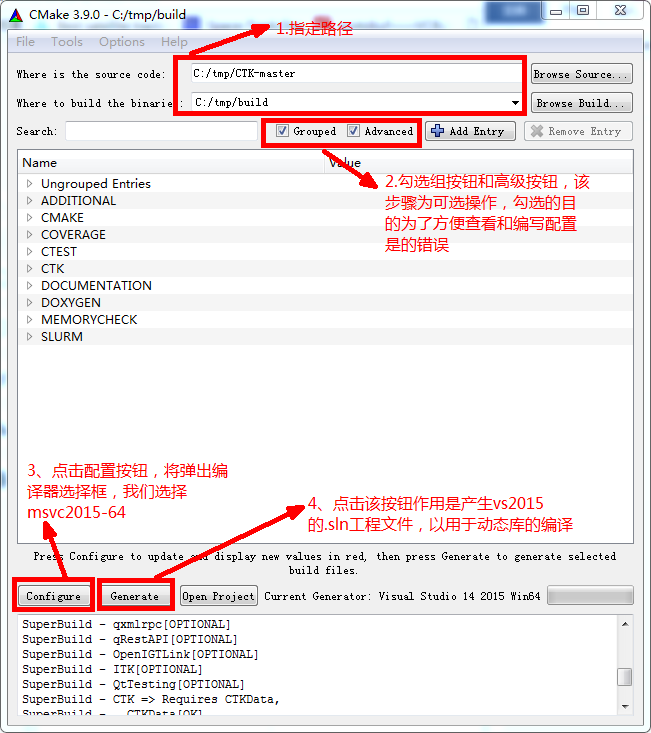
1. 编译

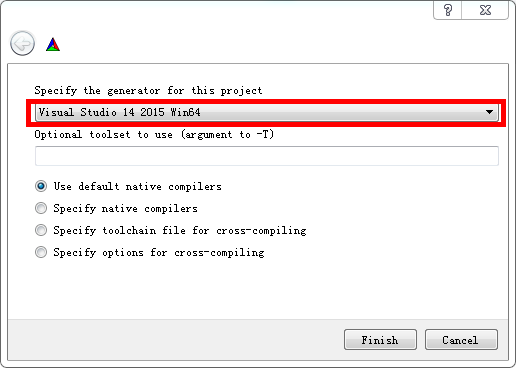
第一步：安装Qt5.9.2开发包，在安装时，需要注意勾选MSVC2015或者MSVC2015-64bit，后面安装教程以编译msvc2015-64bit的动态库来讲解（如果已经安装，此步可以跳过）。

第二步：解压CTK源码包和CTKData数据包至任意文件夹，注意尽量不要出现中文路径（现以C:/tmp为例），并创建编译生成目录build，如下图所示。

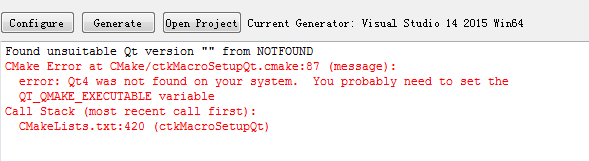


第三步：解压CMake至任意目录（以解压到C：盘为例讲解），CMake为绿色软件（免安装），我们进入到“C:\cmake-3.9.0-win64-x64\bin”，并打开cmake-gui.exe程序。将看到如下图界面，我们按照下图所示步骤操作，1、指定源码路径及编译产生目录，2、勾选Group及Advanced按钮。3、点击“Configure”按钮，4、点击 “Generate”来将生成VS2015的工程文件。

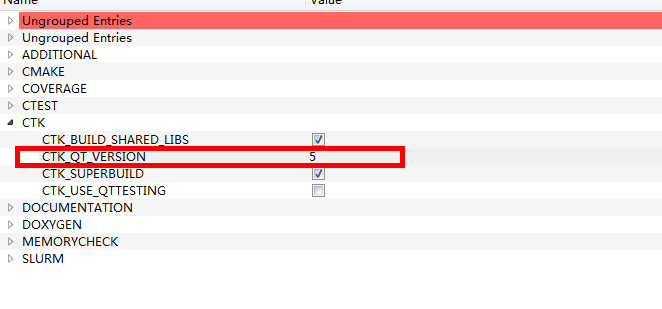


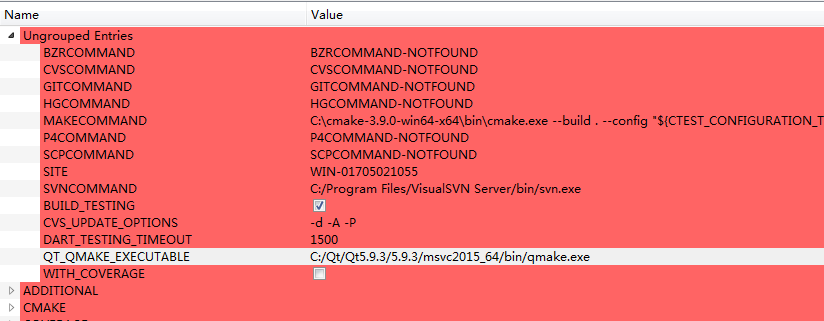


1. 遇到的问题及处理
2. 在点击“Configure”按钮配置后，会出现如下错误：

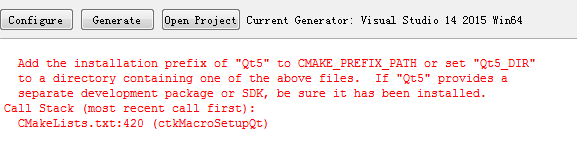


处理：指定qmake的路径及Qt的版本为5，如下图所示。

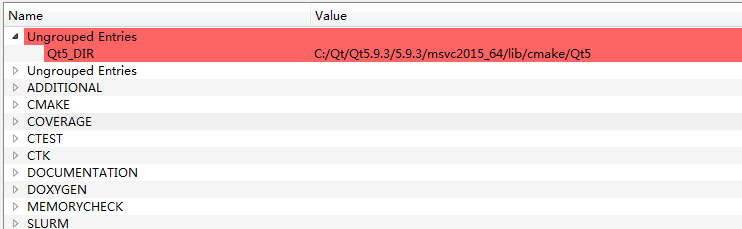




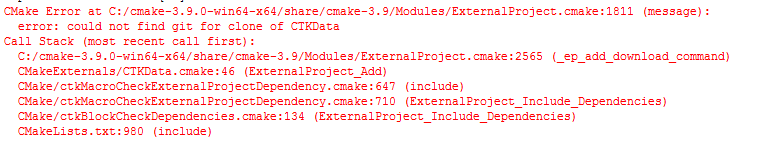
1. 再次点击“Configure”按钮，配置运行后出现如下错误：



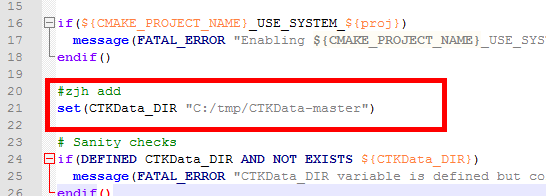
处理：指定Qt下CMake的路径，如下如图所示：



1. 再次点击“Configure”按钮，配置运行后出现如下错误：

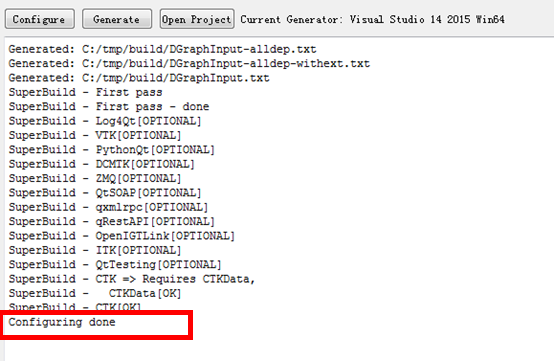


处理：在C:\tmp\CTK-master\CMakeExternals\下的CTKData.cmake文件中添加如下一句，如下图所示，再次点击“Configure”，系统将不再在线更新CTKData数据，而是从本地读取CTKData数据。

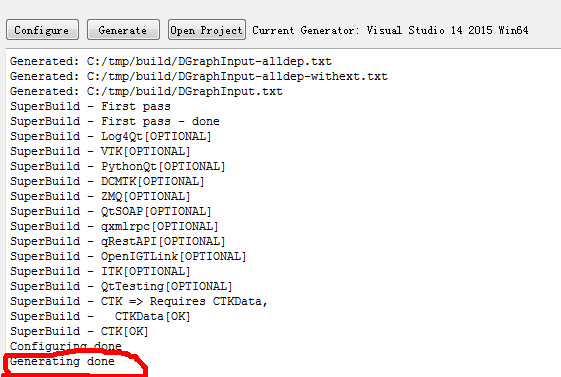


1. 结果

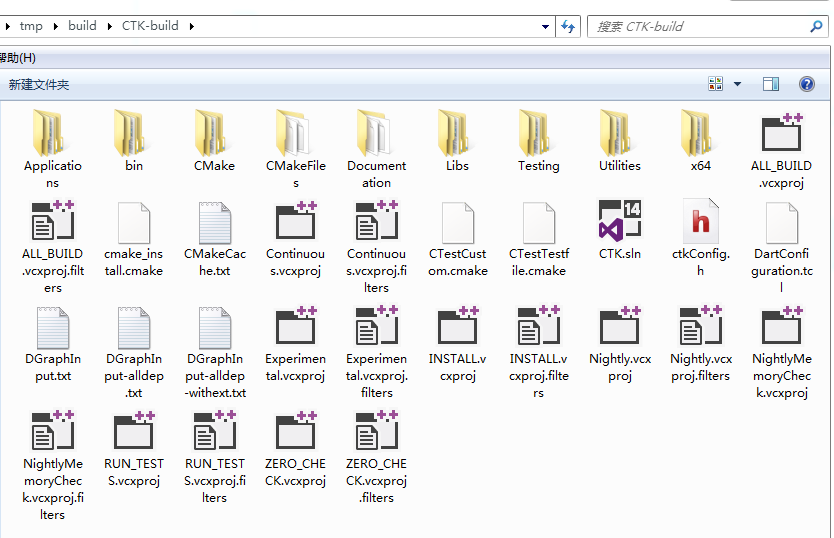
点击“Configure”按钮，配置通过会出现如下图所示信息：



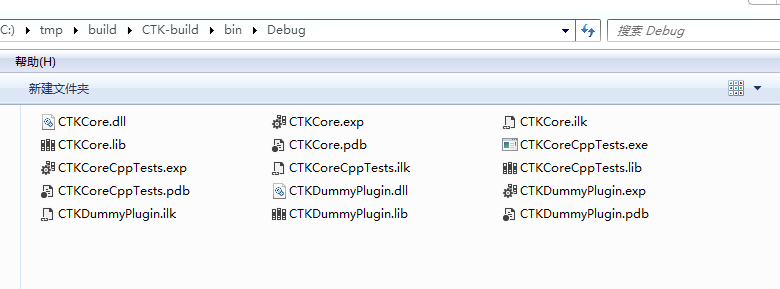
点击“Generate”按钮，创建工程通过会出现如下图所示信息：



在build目录下生产对应的工程文件，如下图所示：



通过vs2015加载工程编译，将能够得到我们所需的动态库。如下图所示：



1. **CTK插件编译及使用**