Atitit **跨平台 跨语言 通用api 的三种实现策略API 映射 API 模拟 1GUI 模拟 2**

目录

[API 映射 1](#_Toc11318)

[API 模拟 1](#_Toc27650)

[GUI 模拟 2](#_Toc281)

## **API 映射**

API 映射是说，界面库使用同一套 API，将其映射到不同的底层平台上面。大体相当于将不同平台的 API 提取公共部分。比如说，将 Windows 平台上的按钮控件和 Mac OS 上的按钮组件都取名为 Button。当你使用 Button 时，如果在 Windows 平台上，则编译成按钮控件；如果在 Mac OS 上，则编译成按钮组件。

1. 优点：   
   所有组件都是原始平台自有的，外观和原生平台一致。
2. 缺点：   
   编写库代码的时候需要大量工作用于适配不同平台，并且，只能提取相同部分的 API。比如 Mac OS 的文本框自带拼写检测，但是 Windows 上面没有，则不能提供该功能。
3. 代表：   
   这种策略的典型代表是 wxWidgets。这也是一个标准的 C++ 库，和 Qt 一样庞大。它的语法看上去和 MFC 类似，有大量的宏。据说，一个 MFC 程序员可以很容易的转换到 wxWidgets 上面来。

## **API 模拟**

API 映射会“缺失”不同平台的特定功能，而 API 模拟则是解决这一问题。不同平台的有差异 API，将使用工具库自己的代码用于模拟出来。按照前面的例子，Mac OS 上的文本框有拼写检测，但是 Windows 的没有。那么，工具库自己提供一个拼写检测算法，让 Windows 的文本框也有相同的功能。

1. 优点：   
   API 模拟最大优点是，应用程序无需重新编译，即可运行到特定平台上。
2. 代表：   
   API 模拟的典型代表是 wine —— 一个 Linux 上面的 Windows 模拟器。它将大部分 Win32 API 在 Linux 上面模拟了出来，让 Linux 可以通过 wine 运行 Windows 程序。   
   另外一个例子是微软提供的 DirectX，这个开发库将屏蔽掉不同显卡硬件所提供的具体功能。使用这个库，你无需担心硬件之间的差异，如果有的显卡没有提供该种功能，SDK 会使用软件的方式加以实现。

## **GUI 模拟**

任何平台都提供了图形绘制函数，例如画点、画线、画面等。有些工具库利用这些基本函数，在不同绘制出自己的组件，这就是 GUI 模拟。

1. 优点：   
   可以很方便的修改组件的外观，只要修改组件绘制函数即可。
2. 缺点：   
   GUI 模拟的工作量无疑是很大的，因为需要使用最基本的绘图函数将所有组件画出来；并且这种绘制很难保证和原生组件一模一样。
3. 代表：   
   很多跨平台的 GUI 库都是使用的这种策略，例如 gtk+（这是一个 C 语言的图形界面库。使用 C 语言很优雅地实现了面向对象程序设计。不过，这也同样带来了一个问题——使用大量的类型转换的宏来模拟多态，并且它的函数名一般都比较长，使用下划线分割单词，看上去和 Linux 如出一辙。gtk+ 并不是模拟的原生界面，而有它自己的风格，所以有时候就会和操作系统的界面格格不入。），Swing 以及 Qt。