浅谈Configure和Setting

Configure（配置）和Settin（设置），对于软件开发者来说应该是不太陌生的两个术语。但是，它们之间有什么关联和区别呢？在本文中，Anders Liu将和您一起分享一些想法。

首先，Configure和Setting都是与应用程序相关的数据，用于在运行时影响应用程序的行为。应用程序在启动后，往往会立即（或在某些时刻）读取Configure和Setting文件（或其他载体）中的内容，并根据具体配置和设置的不同，以不同的方式完成任务。

然而，它们的区别是什么呢？

Anders Liu认为，对于应用程序来说，Configure应该是只读的，而Setting是可读写的；对于用户来说，Configure应该是可读写的，而Setting应该是不可读写（甚至应该是隐藏的）。

例如，一个C/S架构的应用程序，需要访问数据库，因此需要一个连接字符串；同时，当应用程序的客户端关闭后，还能记录窗体最后的位置和大小。此场景涉及了3个数据——连接字符串、窗体位置和窗体大小。

那么，此时这些数据应该放在哪里呢？我们来分析一下这些数据的使用场景。

连接字符串通常由客户端管理员负责维护（可能是在程序安装时就写好一个默认值，之后可以由管理员修改，以适应不同的服务器环境）。而应用程序只需要在启动时（或需要连接数据库时）读取一下这个值即可。

窗体的位置和大小，通常在应用程序结束时，由应用程序对其进行持久化存储（文件、注册表等）；而在下一次启动应用程序时，由应用程序将其读取出来并根据其值修改窗体的位置和大小。

由此就看出两种数据的区别了，连接字符串"人来写，程序来读"，应该放在Configure中；窗体位置大小"程序写，程序读"，应该放在Setting中。

由此，Anders Liu想到了一些在设计和编程中的一些思路。

在设计过程中，应该对数据的使用场景进行思考（甚至进行建模，比如编写场景/方案文档），然后根据不同的场景决定将数据放在Configure中还是Setting中。

设计完毕后，在具体的编码中，如果遇到了需要修改Configure的情况，那么就应该暂停下来，仔细进行思考——是场景分析的有问题吗？如果是，考虑将数据放在Setting中，重新进行设计。如果不是，那么，应该考虑单独提供一个应用程序（配置工具），专门扶助管理员修改和保存配置。

而对于Setting，应该尽量隐藏Setting持久化存储的位置，尽可能隐藏（对用户透明）。但在读取Setting时，也需要进行严密的数据完整性检查，因为不可避免有用户手动修改Setting。（读取Configure的检查可以略弱一些，因为可以让他们"与管理员联系"嘛~ :D）

以上只是Anders Liu对于Configure和Setting的一些看法。很多软件开发框架实际上都对Configure提供了读和写的支持，但Setting一般都是靠开发者自己来实现。这就造成了一种很不好的编程习惯，就是将本来因该存放在Setting中的东西放到了Configure中，最终导致——从程序的角度看，Configure和Setting的概念变得混乱，难以理解和维护；从用户（应用程序管理员）的角度看，他们并不清楚哪些是可以改的配置，哪些是不应该去修改的设置。

总而言之，使用各种编程框架中提供的写Configure的API时一定要慎重，如果不是在编写"配置工具"，请尽力避免写Configure。二是自己写一些代码，实现自己的Setting框架。