**Builder（生成器）**

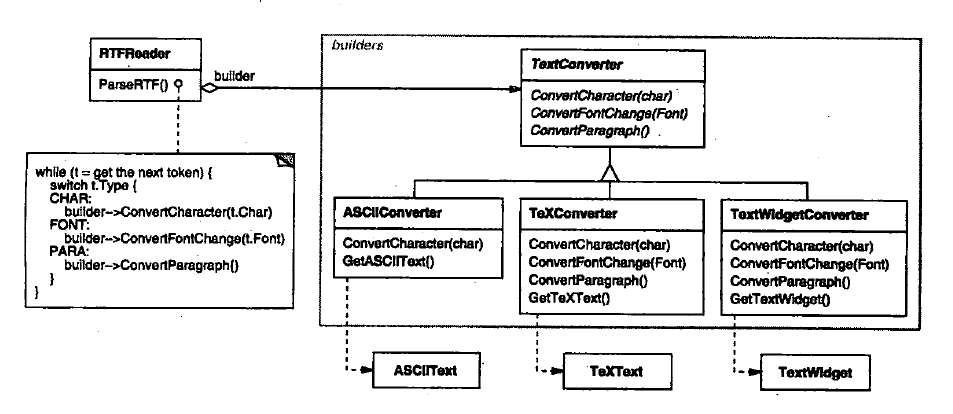
意图

将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

动机

一个RTF（Rich Text Format）文档交换格式的阅读器应能将RTF转换为多种正文风格。该阅读器可以将RTF文档转换成普通ASCII文本或转换成一个能以交互方式编辑的正文窗口组件。但问题在于可能转换的数目是无限的。因此要能够很容易实现新的转换的增加，同时却不改变RTF阅读器。

一个解决方法是用一个可以将RTF转换成另一种正文表示的TextConverter对象配置这个RTFReader类。当RTFReader对RTF文档进行语法分析时，它使用TextConverter去做转换。无论何时RTFReader识别一个RTF标记（或是普通正文或是一个RTF控制字），它都发送一个请求给TextConverter去转换这个标记。TextConverter对象负责进行数据转换以及用特定格式表示该标记，如下图所示。



TextConverter的子类对不同转换和不同格式进行特殊处理，例如，一个ASCIIConverter只负责转换普通文本，而忽略其他转换请求。另一方面，一个TeXConverter将会为实现对所有请求的操作，以便生成一个获取正文中所有风格信息的TEX表示。一个TextWidgetConverter将生成一个复杂的用户界面对象以便用户浏览和编辑正文。

每种转换器类将创建和装配一个复杂对象的机制隐含在抽象接口的后面。转换器独立于阅读器，阅读器负责对一个RTF文档进行语法分析。

Builder模式描述了所有这些关系。每一个转换器类在该模式中被称为生成器，而阅读器则称为向导器。在上面的例子中，Builder模式将分析文本格式的算法与描述怎样创建和表示一个转换后格式的算法分离开来。这使我们可以重用RTFReader的语法分析算法，根据RTF文档创建不同的正文表示—仅需使用不同的TextConverter子类配置该RTFReader即可。

结构图

