**Proxy（代理）**

意图

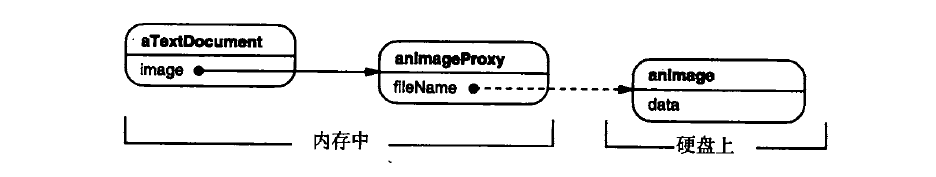
为其他对象提供一种代理以控制对这个对象的访问。

动机

对一个对象进行访问控制的一个原因是为了只有在我们确实需要这个对象时才对它进行创建和初始化。我们考虑一个可以在文档中嵌入图形对象的文档编辑器。有些图形对象（如大型光栅化图像）的创建开销很大。但是打开文档必须很迅速，因此我们在打开文档时应避免一次性创建所有开销很大的对象。因为并非所有这些对象在文档中都同时可见，所以也没有必要同时创建这些对象。

这一限制条件意味着，对于每一个开销很大的对象，应该根据需要进行创建，当一个图像变为可见时会产生这样的需要。但是在文档中我们用什么来代替这个图像呢？我们又如何才能隐藏根据需要创建图像这一事实，从而不会使得编辑器的实现复杂化呢？例如，这种优化不应影响绘制和格式化代码。

问题的解决方案是使用另一个对象，即图像Proxy，替代那个真正的图像。Proxy可以代替一个图像对象，并且在需要时负责实例化这个图像对象。



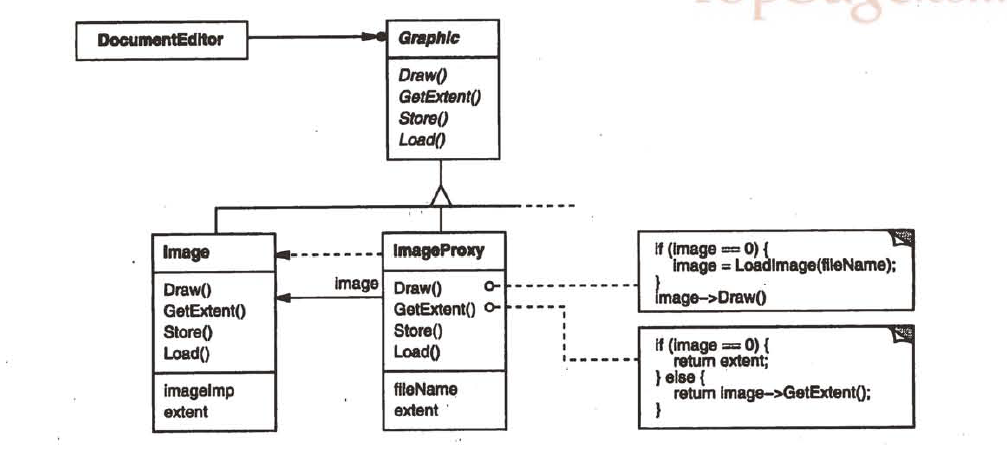
只有当文档编辑器激活图像代理的Draw操作以显示这个图像的时候，图像Proxy才创建真正的图像。Proxy直接将随后的请求转发给这个图像对象。因此在创建这个图像以后，它必须有一个指向这个图像的引用。

我们假设图像存储在一个独立的文件中。这样我们可以把文件名作为对实际对象的引用。Proxy还存储了图像的尺寸，即它的长和宽。有了图像尺寸，Proxy无需真正实例化这个图像就可以相应格式化程序对图像尺寸的请求。

以下类图更详细阐述这个例子。

文档编辑器通过抽象的Graphic类定义的接口访问嵌入的图像。ImageProxy是那些根据需要创建的图像的类，ImageProxy保存了文件名作为指向磁盘上的图像文件的指针。该文件名被作为一个参数传递给ImageProxy的构造器。

ImageProxy还存储了这个图像的边框以及对真正的Image实例的指引，直到代理实例化真正的图像时，这个指引才有效。Draw操作必须保证在向这个图像转发请求之前，它已经被实例化了。GetExtent操作只有在图像被实例化后才向他传递请求，否则，ImageProxy返回它存储的图像尺寸。



结构图

