Proxy Pattern

**Motivation**

Sometimes we need the ability to control the access to an object. For example if we need to use only a few methods of some costly objects we'll initialize those objects when we need them entirely. Until that point we can use some light objects exposing the same interface as the heavy objects. These light objects are called proxies and they will instantiate those heavy objects when they are really need and by then we'll use some light objects instead.

This ability to control the access to an object can be required for a variety of reasons: controlling when a costly object needs to be instantiated and initialized, giving different access rights to an object, as well as providing a sophisticated means of accessing and referencing objects running in other processes, on other machines.

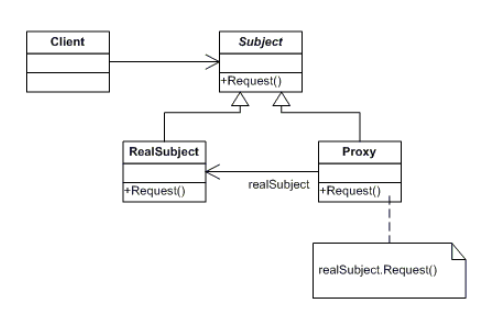
Consider for example an image viewer program. An image viewer program must be able to list and display high resolution photo objects that are in a folder, but how often do someone open a folder and view all the images inside. Sometimes you will be looking for a particular photo, sometimes you will only want to see an image name. The image viewer must be able to list all photo objects, but the photo objects must not be loaded into memory until they are required to be rendered.

**Intent**

The Proxy design pattern provides a surrogate or placeholder for another object to control access to it.

**代理模式**是一种结构型设计模式， 让你能够提供对象的替代品或其占位符。 代理控制着对于原对象的访问， 并允许在将请求提交给对象前后进行一些处理。

代理是一种结构型设计模式， 让你能提供真实服务对象的替代品给客户端使用。 代理接收客户端的请求并进行一些处理 （访问控制和缓存等）， 然后再将请求传递给服务对象。



* Proxy
  + maintains a reference that lets the proxy access the real subject. Proxy may refer to a Subject if the RealSubject and Subject interfaces are the same.
  + provides an interface identical to Subject's so that a proxy can be substituted for for the real subject.
  + controls access to the real subject and may be responsible for creating and deleting it.
  + other responsibilites depend on the kind of proxy:
    - *remote proxies* are responsible for encoding a request and its arguments and for sending the encoded request to the real subject in a different address space.
    - *virtual proxies* may cache additional information about the real subject so that they can postpone accessing it. For example, the ImageProxy from the Motivation caches the real images's extent.
    - *protection proxies* check that the caller has the access permissions required to perform a request.
* Subject
  + defines the common interface for RealSubject and Proxy so that a Proxy can be used anywhere a RealSubject is expected.
* RealSubject
  + defines the real object that the proxy represents.

**场景**：

在需要用比较通用和复杂的对象指针代替简单的指针的时候，使用 Proxy 模式。

下面是一些可以使用 Proxy 模式常见情况：

* 远程代理（Remote Proxy ）为一个对象在不同的地址空间提供局部代表。
* 虚代理（Virtual Proxy ）根据需要创建开销很大的对象。
* 保护代理（Protection Proxy ）控制对原始对象的访问。保护代理用于对象应该有不同的访问权限的时候。
* 智能指引（Smart Reference ）取代了简单的指针，它在访问对象时执行一些附加操作。

它的典型用途包括：

*  对指向实际对象的引用计数，这样当该对象没有引用时，可以自动释放它。
*  当第一次引用一个持久对象时，将它装入内存。
*  在访问一个实际对象前，检查是否已经锁定了它，以确保其他对象不能改变它。