 根据伺服对象的状态把它们进行分类后，现在基于内存管理来定义Corba对象的分类。显然，这个讨论和伺服对象的生命周期密切相关。这里侧重于分类，所以只是简单地讨论一下内存管理问题，并在下面引入一通用管理模式。 1. 静态Corba对象 静态对象是在系统整个生命时期中存在的Corba对象。通常，这些对象是组件入口点例如， Corba命名服务必须提供一根命名上下文，用来创建新的命名层次。这个根命名上下文可归类为静态的，因为它始终存在。从内存管理的角度来看，静态对象的实现是很简单的。通常，静态对象可通过在服务器主线中实例化伺服对象来实现，并把它直接绑定到相关的Corba对象（即早期绑定）。 2. 瞬态Corba对象 瞬态Corba对象并不和任何持久的状态相关联-它确实是瞬态的。通常，瞬态Corba对象绑定到有状态伺服对象，即对象的状态仅由伺服对象包含。不幸的是，这意味着瞬态Corba对象的生命周期紧密绑定到伺服对象的生命周期：瞬态Corba对象的创建必然导致伺服对象的立即激活，以实现对象。另一方面，伺服对象的销毁会立即导致相关Corba对象的删除，因为所有的状态都随着伺服对象而消失。 瞬态对象的一个恰当例子是迭代器，它使用户能反复查看查询的结果集合。迭代器对象并不和任何持久的状态绑定，因为查询结果通常是瞬态的。为瞬态Corba对象找出好的内存管理策略会是很困难的。 3. 持久Corba对象 最后，持久Corba对象和其他持久状态相关联，并由数据管理系统来维护。这

使用户可以为这些对象的实现应用十分灵活的内存管理策略，因为我们可以使用后期绑定来动态激活和冻结伺服对象。

4. 伺服对象池模式 对Corba对象实现的分类，说明了从内存管理的角度来看，不同的对象有不同的需求。伺服对象池模式为伺服对象管理定义了一个通用的框架。它的基本思想是包含一个池管理器，来管理激活伺服对象所在的池。每个伺服对象和一驱逐策略相关联。该策略描述了伺服对象何时被逐出。池管理器有两个角色：保持器和驱逐器。保持器保证对象在需要时存在。例如，瞬态对象不能重新创建，所以它必须保持到客户机对它的请求完成。驱逐器必须保证伺服对象是经常被逐出的，以避免不必要的资源消耗。 