**Chain of Responsibility (职责链)**

意图

使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到一个对象处理它为止。

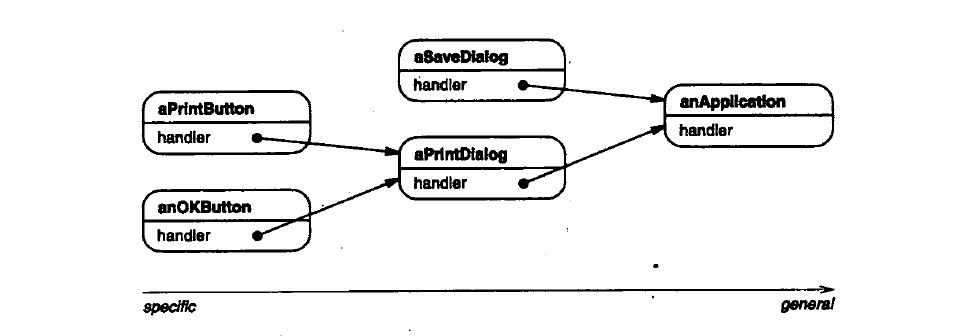
动机

考虑一个图形用户界面中的上下文有关的帮助机制。用户在界面的任一部分上点击就可以得到帮助信息，所提供的帮助依赖于点击的是界面哪一部分以及其上下文。例如，对话框中的按钮的帮助信息就可能和主窗口中的类似的按钮不同。如果对那一部分界面没有特定的帮助信息，那么帮助系统应该显示一个关于当前上下文的较一般的帮助信息—比如说，整个对话框。

因此很自然地，应根据普遍性即从最特殊到最普通的顺序来组织帮助信息。而且，很明显，在这些用户界面对象中会有一个对象来处理帮助请求；至于是哪一个对象则取决于上下文以及可用的帮助具体到何种程度。

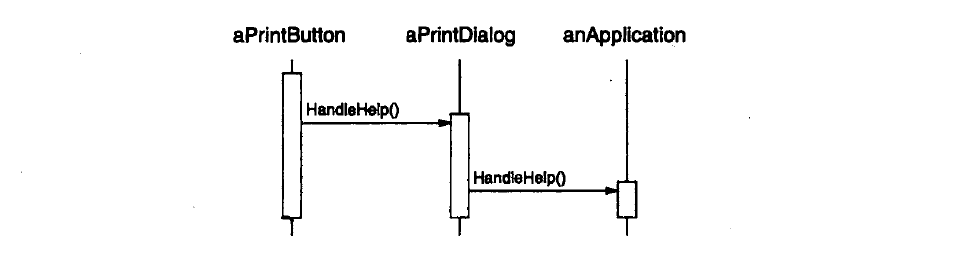
这儿的问题是提交帮助请求的对象（如按钮）并不明确知道谁是最终提供帮助的对象。我们要有一种办法将提交帮助请求的对象与可能提供帮助信息的对象解耦。Chain of Responsibility模式告诉我们应该怎么做。

这一模式的想法是，给多个对象处理一个请求的机会，从而解耦发送者和接收者。该请求沿对象链传递直至其中一个对象处理它，如下图所示。



从第一个对象开始，链中收到请求的对象要么亲自处理它，要么转发给链中下一个候选者。提交请求的对象并不明确地知道哪一个对象将会处理它—我们说该请求有一个隐式的接收者。

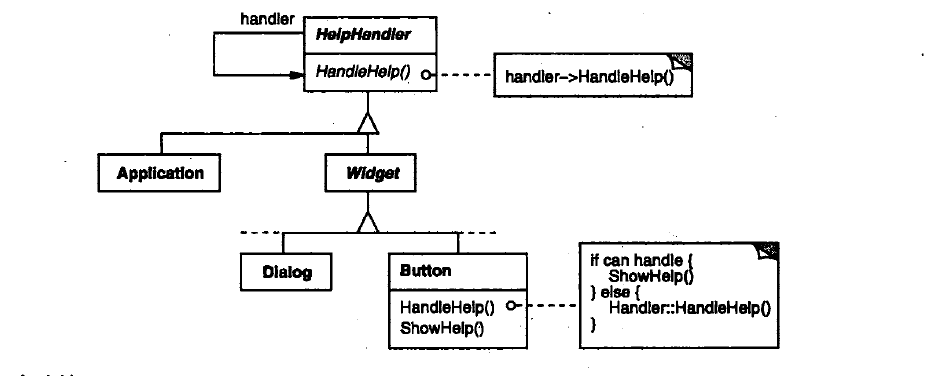
假设用户在一个标有“Print”的按钮窗口组件上单击帮助，而该按钮包含在一个PrintDialog的实例中，该实例知道它所属的应用对象。下面的交互框图说明了帮助请求怎样沿链传递：



在这个例子中，既不是aPrintButton也不是aPrintDialog处理该请求；它一直被传递给anApplication,anApplication处理它或忽略它。提交请求的客户不直接应用最终响应它的对象。

要沿着链转发请求，并保证接收者为隐式的，每个在链上的对象都有一致的处理请求和访问链上后继者的接口。例如，帮助系统可定义一个带有相应的HandleHelp操作的HelpHandler类。HelpHandler可为所有候选对象类的父类，或者它可被定义为一个混入类。这样想处理求帮助请求的类就可将HelpHandler作为其一个父类，如下图所示。

按钮、对话框和应用类都使用HelpHandler操作来处理帮助请求。HelpHandler的HandleHelp操作缺省的是将请求转发给后继。子类可从定义这一操作以在适当的情况下提供帮助；否则它们可使用缺省实现转发该请求。



结构图

