Java SE程序设计 北京圣思园科技有限公司

主讲人　张龙

观察者模式(Observer)

- 观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态上发生变化时，会通知所有观察者对象，让他们能够自动更新自己

观察者模式(Observer)

- 观察者模式的组成
  - 抽象主题角色：把所有对观察者对象的引用保存在一个集合中，每个抽象主题角色都可以有任意数量的观察者。抽象主题提供一个接口，可以增加和删除观察者角色。一般用一个抽象类或接口来实现。
  - 抽象观察者角色：为所有具体的观察者定义一个接口，在得到主题的通知时更新自己。

观察者模式(Observer)

- 观察者模式的组成
  - 具体主题角色：在具体主题内部状态改变时，给所有登记过的观察者发出通知。具体主题角色通常用一个子类实现。
  - 具体观察者角色：该角色实现抽象观察者角色所要求的更新接口，以便使本身的状态与主题的状态相协调。如果需要，具体观察者角色可以保存一个指向具体主题角色的引用。通常用一个子类实现

观察者模式(Observer)

- 实现自己的观察者模式

观察者模式(Observer)

- 观察者模式在Java语言中的地位极其重要
- **JDK也提供了对观察者模式的内置支持**

Observable（观测）

- Observable类用于创建可以观测到你的程序中其他部分的子类。当这种子类的对象发生变化时，观测类被通知。观测类必须实现定义了update( )方法的Observer接口。当一个观测程序被通知到一个被观测对象的改变时，update( )方法被调用。

Observable （观测）

- 一个被观测的对象必须服从下面的两个简单规则。第一，如果它被改变了，它必须调用setChanged( )方法。第二，当它准备通知观测程序它的改变时，它必须调用notifyObservers( )方法。这导致了在观测对象中对update( )方法的调用。注意——当对象在调用notifyObservers( )方法之前，没有调用setChanged( )方法，就不会有什么动作发生。在update( )被调用之前，被观测对象必须调用setChanged( )和notifyObservers( )两种方法。

Observable （观测）

- 注意notifyObservers()有两种形式：一种带有参数而另一种没有。当用参数调用notifyObservers( )方法时，该对象被传给观测程序的update( )方法作为其第二个参数。否则，将给update( )方法传递一个null。可以使用第二个参数传递适合于你的应用程序的任何类型的对象。

观测接口

- 为了观测一个可观测的对象，必须实现Observer接口。这个接口仅仅定义了如下所示的一个方法。
  - void update(Observable observOb, Object arg)
  - 这里，observOb是被观测的对象，而arg是由notifyObservers( )方法传递的值。当被观测对象发生了改变，调用update( )方法

观测程序举例

- 这里是一个说明可观测对象的例子。该程序创建了一个叫做Watcher的类，该类实现了Observer接口。被监控的类叫做BeingWatched，它继承了Observable。在BeingWatched里，是counter( )方法，该方法仅是从一个指定的值开始递减计数。每次计数改变时，notifyObservers( )方法被调用，而当前的计数被作为参数传递给notifyObservers( )方法。这导致了Watcher中的update( )方法被调用，显示当前的计数值。在main( )内，分别调用observing和observed的Watcher和BeingWatched对象被创建。然后，observing被增加到对observed的观测程序列表。这意味着每次counter( )调用notifyObservers( )方法时，observing.update( )方法将被调用

观测程序举例

- 参见程序 ObserverDemo.java

观测程序举例

- 有多个对象可以用作观测程序。例如下面程序实现了两个观测类并且将每个类中的一个对象增加到BeingWatched观测程序列表中。第二个观测程序等待直到计数为0

观测程序举例

- 参见程序 TwoObservers.java