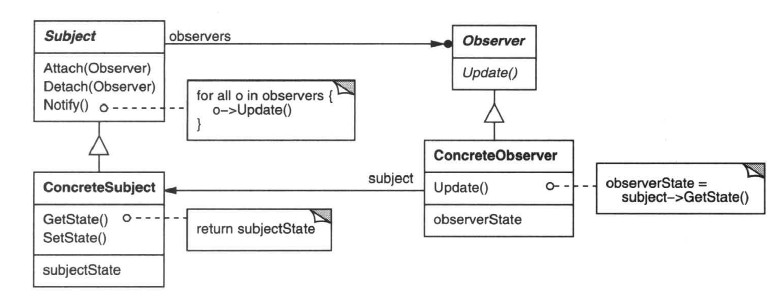
# 一、事件基础概念

## 简介

触发事件的对象称为事件发送者（目标）；接收事件的对象称为事件接收者（观察者）。**事件模型一般用到了观察者模式。**



1. SUBJECT（目标）

目标知道它的观察者，可以有任意多个观察者观察同一个目标，提供注册和删除观察者对象的接口。

1. Observer（观察者）

为那些在目标发生改变时需获得通知的对象定义一个更新接口。

1. ConcreteSubject（具体目标）

将有关状态存入各ConcreteObserver对象，当它的状态发生改变时,向它的各个观察者发出通知。

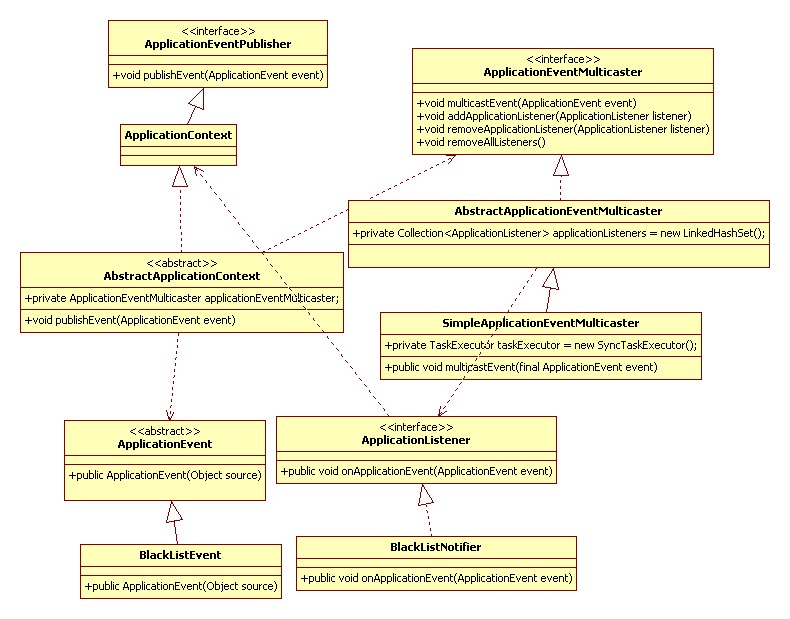
1. ConcreteObserver（具体观察者）

维护一个指向ConcreteSubject对象的引用。

存储有关状态，这些状态应与目标的状态保持一致。

实现Observer的更新接口以使自身状态与目标的状态保持一致。

## Spring事件机制



1. ApplicationListener 就是Observer(观察者)，需要到容器中注册，他要关心他所关心的ApplicationEvent 。一般有如下代码：if (event instanceof BlackListEvent) {}
2. ApplicationEventMulticaster是Subject(目标)一个代理，负责管理 ApplicationListener 。
3. ApplicationEvent 是事件，就是传递的媒介，充当介质的作用。
4. 在spring中,容器管理所有的 bean。是ApplicationEvent 驱动的，一个ApplicationEvent publish了，观察这个事件的监听者就会送到通知。

# 二、Spring的事件处理

## 1. 工作流程

1. 当加载 beans 时，ApplicationContext （目标）发布某些类型的事件。
2. 通过 ApplicationEvent 类（介质）和 ApplicationListener （观察者）接口来提供在 ApplicationContext 中处理事件。
3. 如果一个 bean 实现 ApplicationListener（观察者），那么每次 ApplicationEvent 被发布到 ApplicationContext 上，那个 bean 会被通知。

(4) ApplicationContext发布事件，ApplicationEvent事件类，ApplicationListener监听器接口。

## 2. Spring内置事件

1. 上下文更新事件（ContextRefreshedEvent）：

该事件会在ApplicationContext被初始化或者更新时发布。也可以在调用ConfigurableApplicationContext 接口中的refresh()方法时被触发。

1. 上下文开始事件（ContextStartedEvent）：

当容器调用ConfigurableApplicationContext的Start()方法启动ApplicationContext时触发该事件。你可以调查你的数据库，或者你可以在接受到这个事件后重启任何停止的应用程序。

1. 上下文停止事件（ContextStoppedEvent）：

当使用 ConfigurableApplicationContext 接口中的 stop() 方法停止 ApplicationContext时触发该事件，在接受到这个事件后做必要的清理的工作。

1. 上下文关闭事件（ContextClosedEvent）：

当使用 ConfigurableApplicationContext 接口中的 close() 方法关闭 ApplicationContext 时，该事件被发布。容器被关闭时，其管理的所有单例Bean都被销毁。一个已关闭的上下文到达生命周期末端；它不能被刷新或重启。

1. 请求处理事件（RequestHandledEvent）：

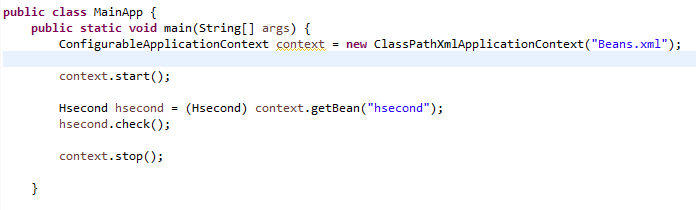
在Web应用中，当一个http请求（request）结束触发该事件，告诉所有 bean HTTP 请求已经被停止。

1. **由于 Spring 的事件处理是单线程的，所以如果一个事件被发布，直至所有的接收者得到的该消息，否则该进程被阻塞并且流程将不会继续。**

## 监听上下文事件

1. 为了监听上下文事件，一个 bean 应该实现只有一个方法 onApplicationEvent() 的 ApplicationListener 接口。

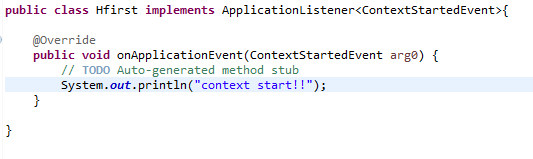
（Main类）

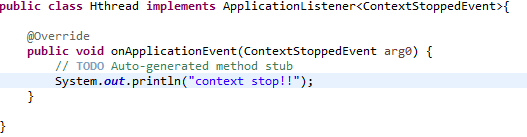


（Bean.xml配置）



（监听类）





# 三、自定义事件

## 1. 事件相关类

事件类（extends ApplicationEvent）

事件发布类（implements ApplicationEventPublisherAware）

事件监听类（implements ApplicationListener<CustomEvent>）

bean配置（声明事件监听类和事件发布类）

执行类（获取bean，发布事件）

## 2. 执行流程

1) 声明一个事件类

2) 声明一个事件发布类，编写事件发布方法，初始化事件类

3) 声明一个事件监听类，监听发布的事件

4) 在bean配置文件中注册事件监听类和事件发布类

5) 在执行类中获取bean，调用事件发布方法

## 3. 具体代码：

（事件类）



（事件监听类）



（事件发布类）



（执行类）



（bean配置文件）

