

导航

博客园

首 页

新随笔

订 阅 

XML

管 理

<

2018年1月

>

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

公告

昵称：目不识丁  
园龄：2年10个月  
粉丝：41  
关注：12  
+加关注

搜索

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

随笔分类

ASP.NET(9)

CLR via C#(4)

MVC(1)

Redis(1)

程序员的自我修养(9)

设计模式(2)

深入理解C#(11)

数据结构与算法分析(6)

数据库

搜索引擎(1)

随笔档案

2018年1月 (1)

2017年12月 (1)

2017年10月 (5)

2017年9月 (3)

2017年8月 (5)

2017年5月 (1)

2017年2月 (1)

2016年11月 (4)

2016年1月 (11)

2015年11月 (1)

2015年10月 (4)

2015年9月 (2)

2015年8月 (6)

2015年7月 (2)

2015年5月 (1)

文章分类

设计模式

积分与排名

积分 - 12694

排名 - 26417

## 设计模式之观察者模式

### 目录

- 什么是观察者模式
- 设计原则
- 图解观察者模式
- 例子
  - 定义观察者模式：类图
  - 设计气象站
  - 实现气象站

### 正文

### 目录

- 什么是观察者模式
- 设计原则
- 图解观察者模式
- 例子
  - 定义观察者模式：类图
  - 设计气象站
  - 实现气象站

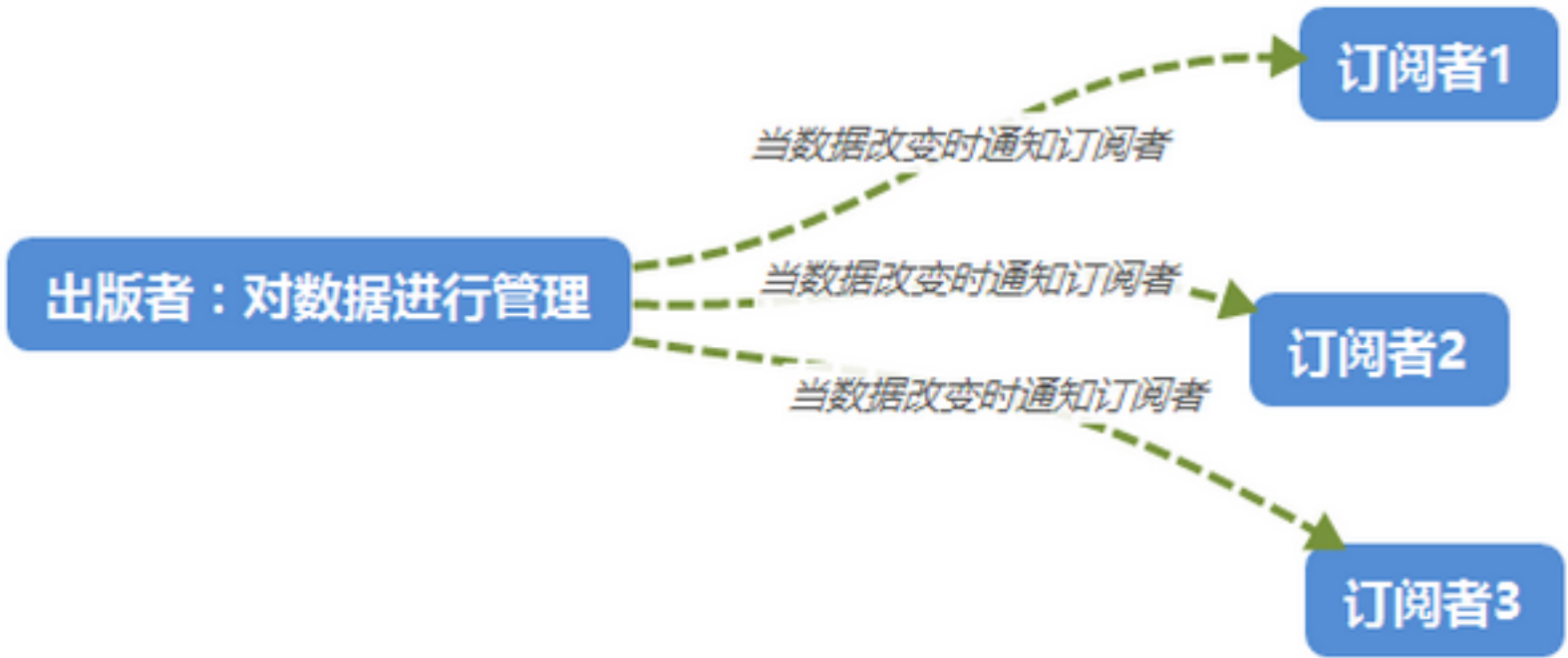
### 正文

回到顶部

回到顶部

## 什么是观察者模式

它定义了对象之间的一对多依赖，这样一来，当一个对象改变状态时，它的所有依赖者都会收到通知。



未订阅者：数据改变不会产生任何影响

回到顶部

回到顶部

## 设计原则

为了交互对象之间的松耦合设计而努力：松耦合的设计之所以能让我们建立有弹性的OO系统，能够应对变化，是因为对象之间的互相依赖降到了最低。

回到顶部

回到顶部

## 图解观察者模式

最新评论

1. Re:设计模式之观察者模式

@追风的狼谢支持...

--目不识丁
2. Re:设计模式之观察者模式支持

--追风的狼
3. Re:设计模式之策略模式

@PatrickLiu多谢支持...

--目不识丁
4. Re:设计模式之策略模式写的不错。有一个是使用模式的过程，这样写很好

--PatrickLiu
5. Re:数据结构之优先队列清晰，明了！

--夜里码代码

阅读排行榜

1. 程序员的自我修养—温故而知新(3641)
2. 程序员的自我修养十内存(858)
3. ASP.NET WebForm(827)
4. 程序员的自我修养四静态链接(651)
5. 程序员的自我修养九Windows下的动态链接(620)

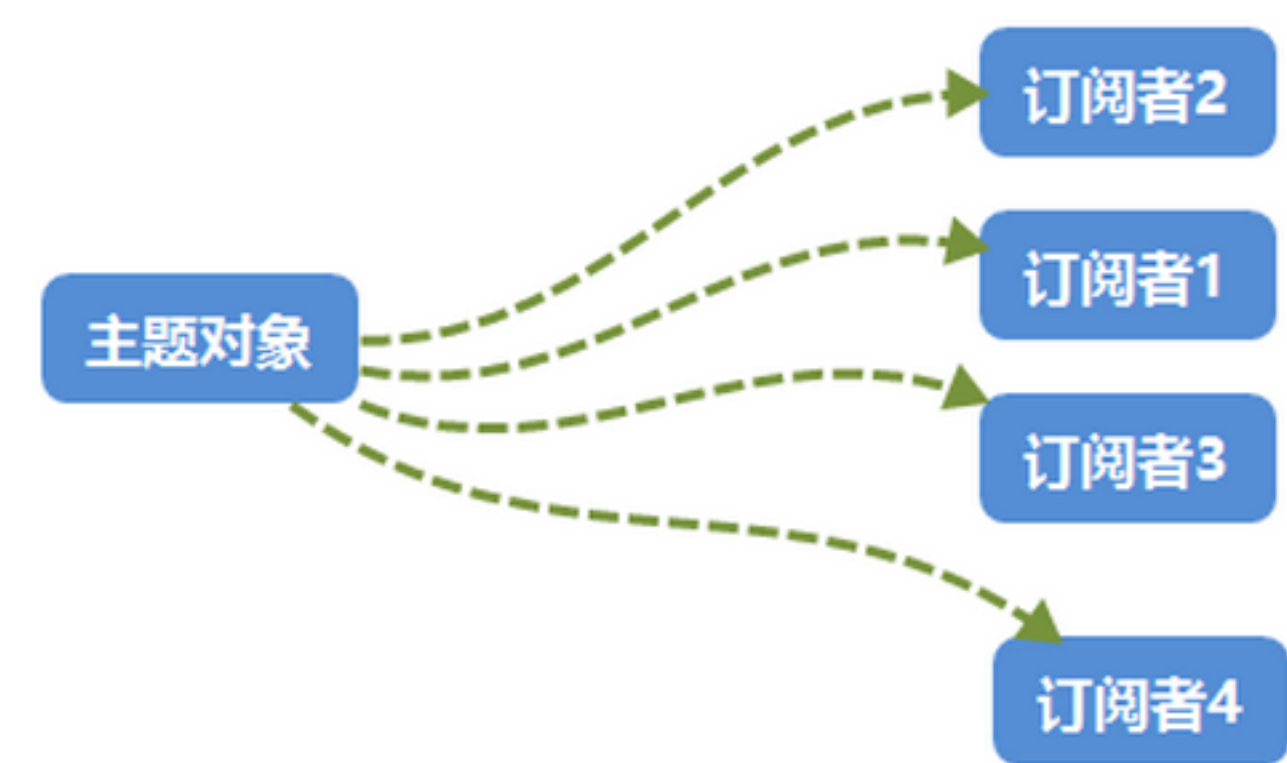
推荐排行榜

1. 程序员的自我修养—温故而知新(59)
2. 程序员的自我修养十内存(7)
3. 程序员的自我修养九Windows下的动态链接(7)
4. 程序员的自我修养六可执行文件的装载与进程(4)
5. 程序员的自我修养二编译和链接(4)

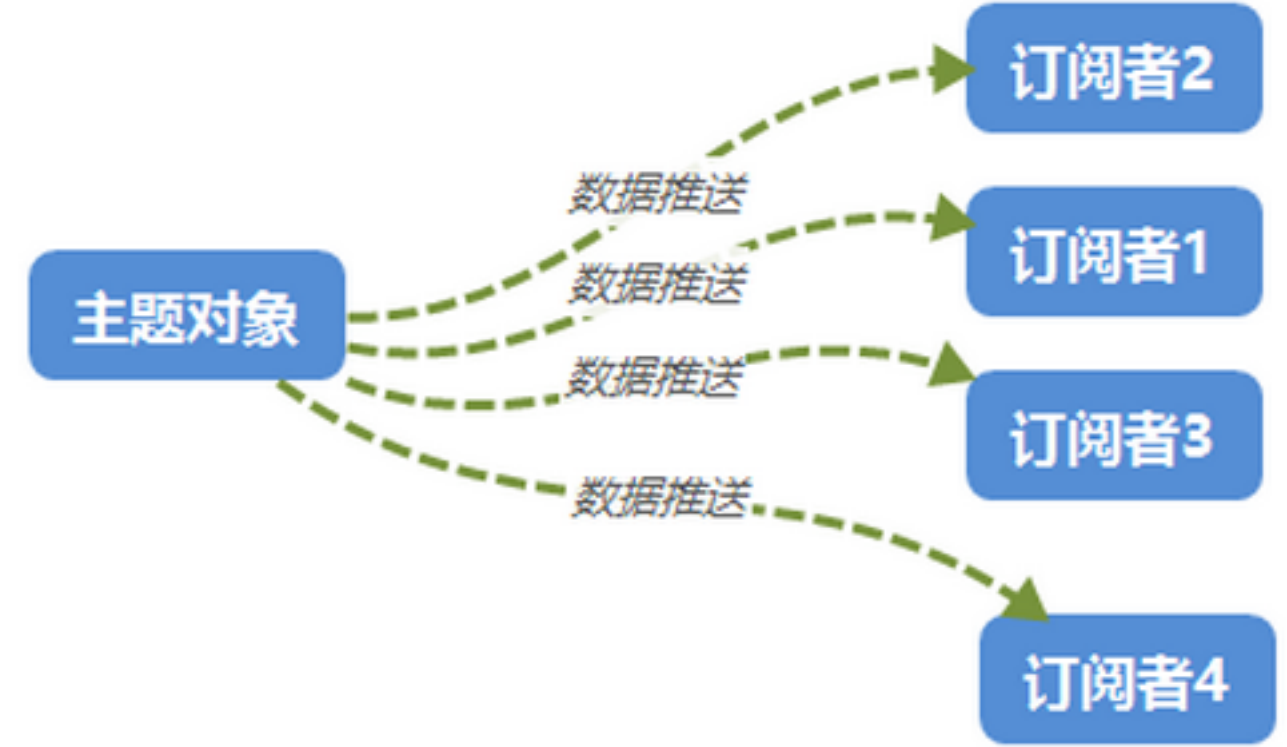
一个新的对象需要订阅：当有一个新的需要订阅的对象时，需要先向主题对象调用注册接口注册订阅信息。



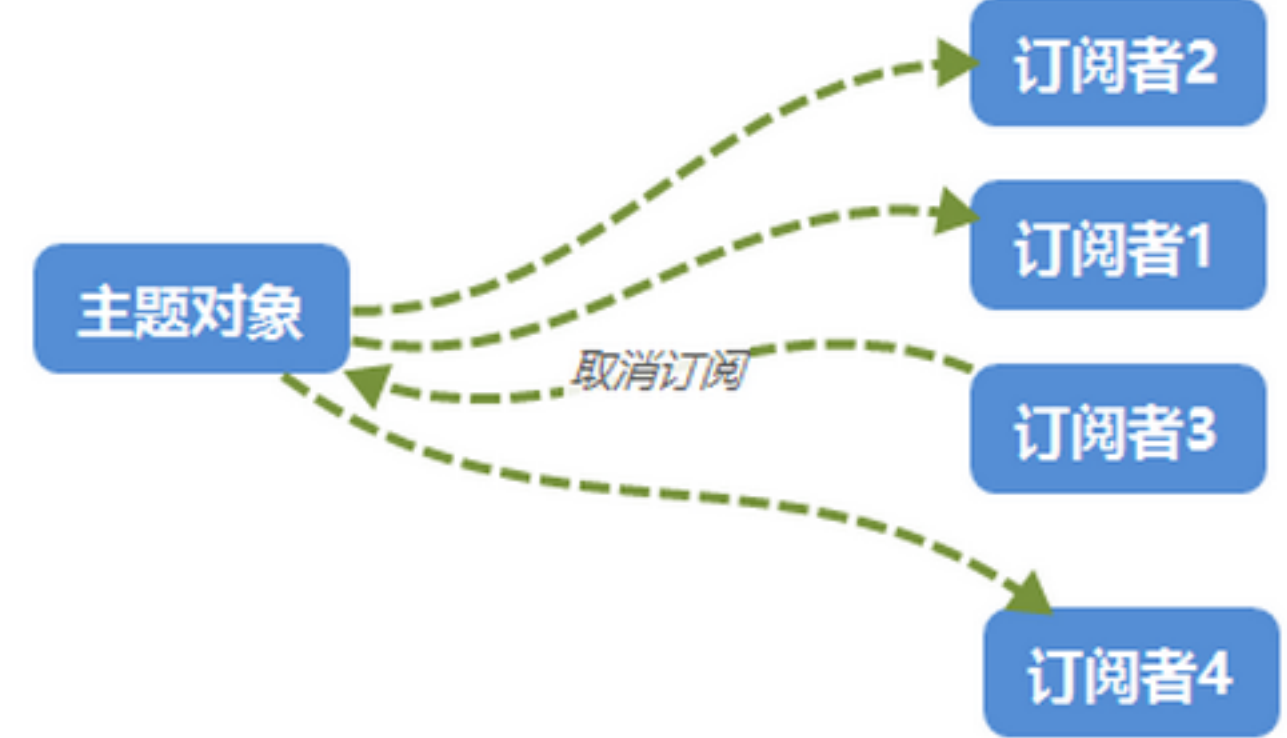
新的订阅者：有了新的订阅者后，主题对象将向订阅者4也推送订阅消息。



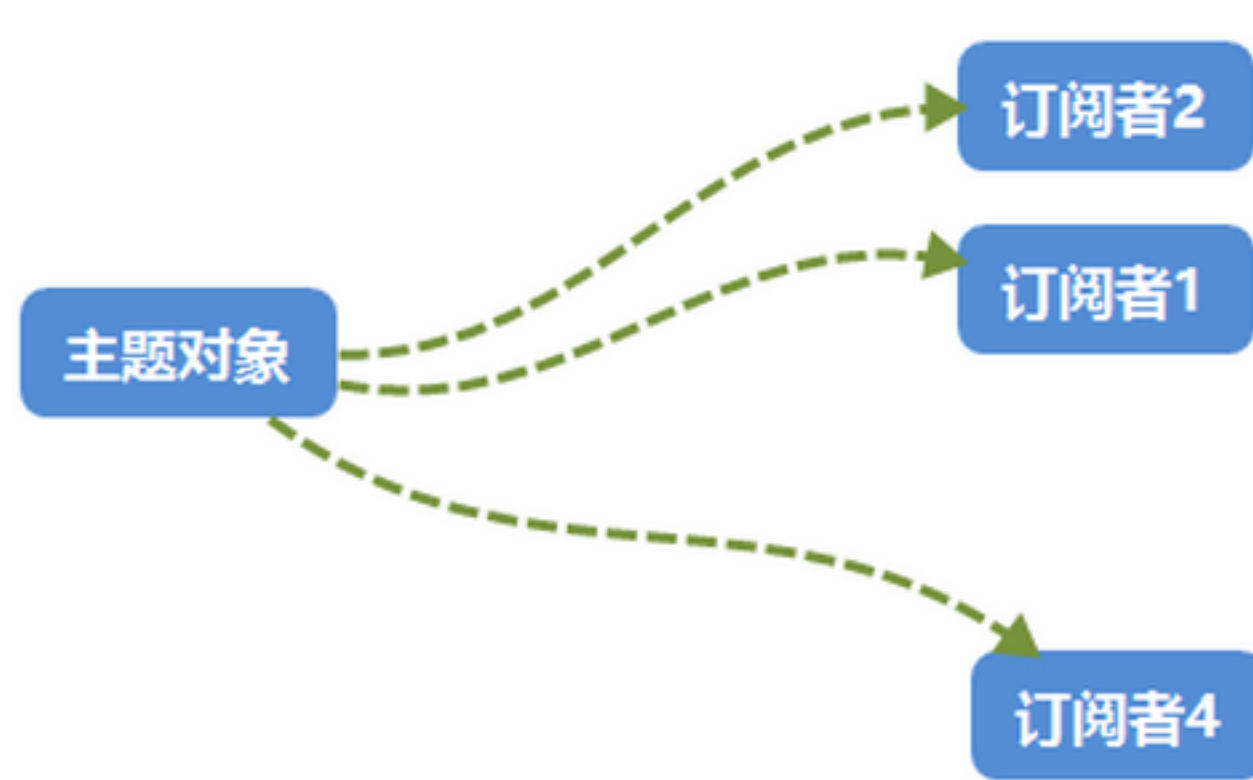
数据更新：一旦数据有更新，将向所有订阅者更新信息。



有订阅者需要离开：有订阅者需要离开时，将该订阅者在订阅者列表中移除。



离开后：不再向离开的订阅者推送消息。

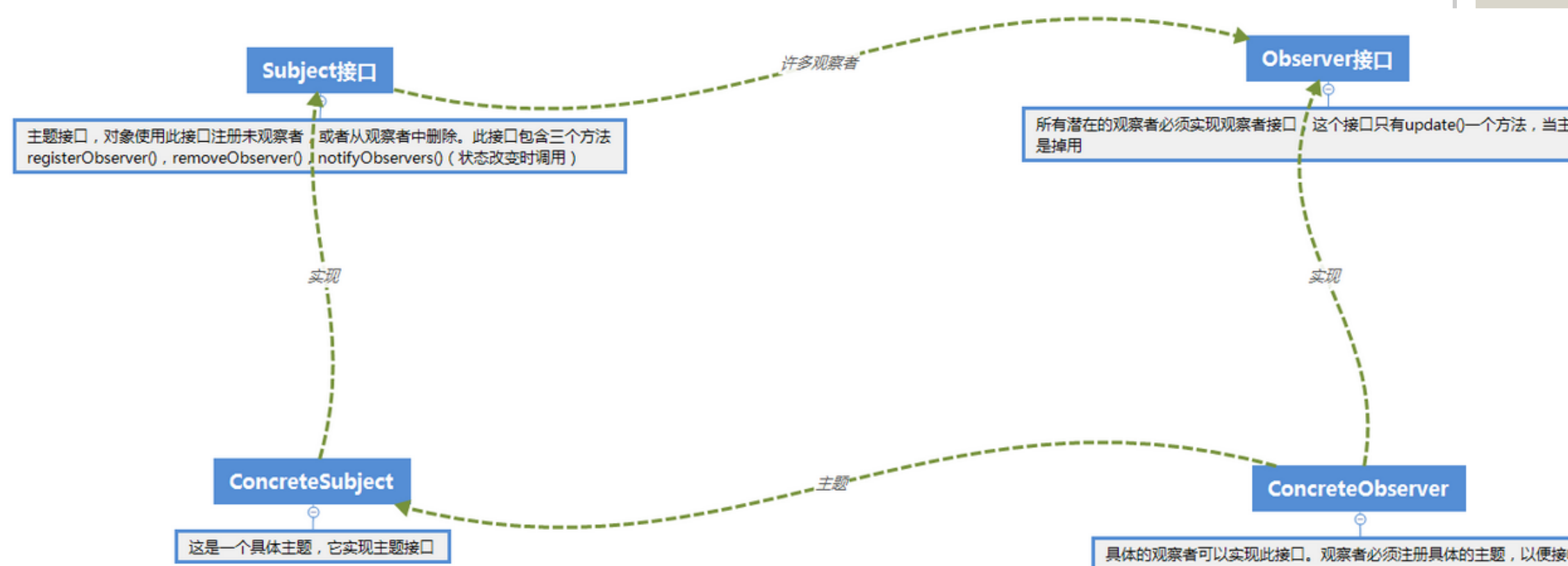


[回到顶部](#)  
[回到顶部](#)

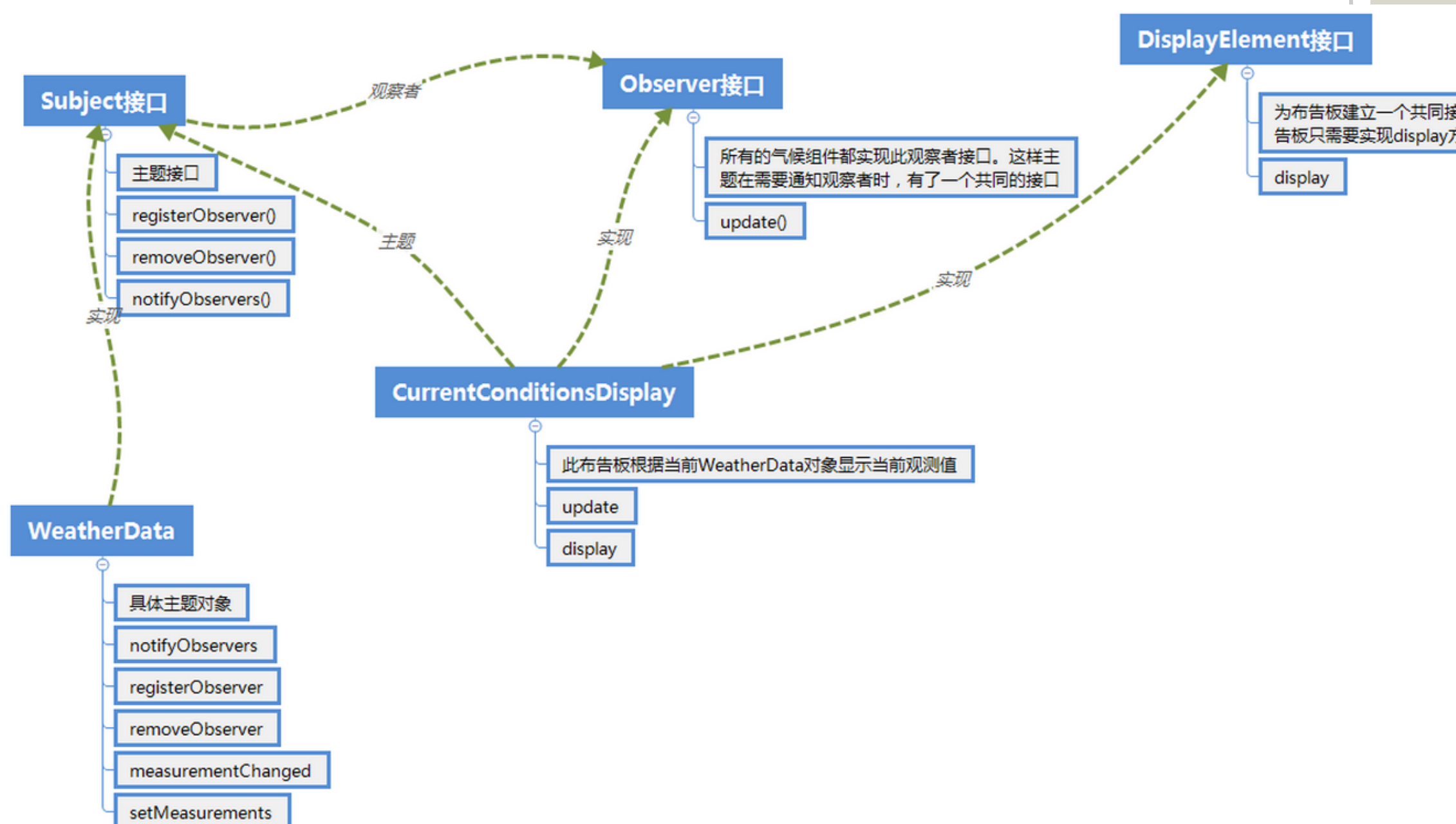
## 例子

现在有一个气象站和布告站（湿度，温度，气压），WeatherData对象（获取气象站数据，更新布告站），我们的工作就是建立一个应用，利用WeatherData对象取得数据，并更新三个布告板：目前状况，气象统计和天气预报。

## 定义观察者模式：类图





## 设计气象站







主题接口：







```
public interface Subject
{
    void registerObserver(Observer o); //注册
    void removeObserver(Observer o); //删除
    void notifyObservers(); //改变状态通知
}
```







观察者接口：





```
public interface Observer
{
    /// <summary>
    /// 当气象观察者改变时，主题把这些状态值当作方法参数，传送给观察者。所有观察者都是实现此方法。
    /// </summary>
    void update(float temp, float humidity, float pressure);
}
```





显示接口（非必需）：



```
public interface DisplayElement
{
    void display(); //当布告栏需要显示时，调用此方法
}
```



主题对象：





```
public class WeatherData : Subject //实现主题接口
{
    private List<Observer> observers; //用来记录观察者
    private float temperature;
    private float humidity;
    private float pressure;
    public WeatherData()
    {
        observers = new List<Observer>();
    }
    public void registerObserver(Observer o) //注册观察者时，添加
    {
        observers.Add(o);
    }
    public void removeObserver(Observer o) //注销观察者时，删除
    {
        observers.Remove(o);
    }
    public void notifyObservers() //把消息通知所有观察者，因为它们都实现了update
    {
        foreach (Observer Observer in observers)
        {
            Observer.update(temperature, humidity, pressure);
        }
    }
}
```

```
}

public void measurementChanged() //更新数据, 通知观察者
{
    notifyObservers();
}

public void setMeasurements(float temp, float humidity, float pressure)
{
    this.temperature = temp;
    this.humidity = humidity;
    this.pressure = pressure;
    measurementChanged();
}

}
```



观察者对象:



```
public class CurrentConditionsDisplay : Observer, DisplayElement //实现观察者接口和显示接口
{
    private float temperature;
    private float humidity;
    private Subject weatherData;
    public CurrentConditionsDisplay(Subject weatherData)
    {
        this.weatherData = weatherData;
        weatherData.registerObserver(this); //在构造方法中注册 (订阅)
    }

    public void display()
    {
        Console.WriteLine($"temperature:{temperature},humidity:{humidity}");
    }

    public void update(float temp, float humidity, float pressure) //当此方法被调用时, 更新温度和湿度并显示
    {
        this.temperature = temp;
        this.humidity = humidity;
        display();
    }
}
```



测试:



```
static void Main(string[] args)
{
    WeatherData weatherData = new WeatherData(); //建立一个主题对象
    CurrentConditionsDisplay currentConditions = new CurrentConditionsDisplay(weatherData); //注册 (订阅)
    //模拟数据
    weatherData.setMeasurements(80, 54, 34);
    weatherData.setMeasurements(1, 2, 3);
    weatherData.setMeasurements(12, 23, 34);
    Console.ReadKey();
}
```



运行结果:

```
temperature:80, humidity:54
temperature:1, humidity:2
temperature:12, humidity:23
```


如本文对您有帮助请移步右下角，推荐本文，谢谢大家的点赞，因为您的支持是我最大动力

分类：[设计模式](#)

好文要顶

关注我

收藏该文



目不识丁

关注 - 12

粉丝 - 41

+加关注

« [上一篇：设计模式之策略模式](#)  
» [下一篇：设计模式之装饰者模式](#)

posted on 2018-01-04 14:55 目不识丁 阅读(333) 评论(2) 编辑 收藏

## 评论

### #1楼

支持

支持(0) 反对(0)

2018-01-05 01:00 | [追风的狼](#)

### #2楼[楼主]

@ [追风的狼](#)  
谢谢支持

支持(0) 反对(0)

2018-01-05 09:00 | [目不识丁](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

努力加载评论框中...

【推荐】超50万VC++源码：大型工控、组态\仿真、建模CAD源码2018！  
【推荐】加入腾讯云自媒体扶持计划，免费领取域名&服务器

腾讯官方

前端NEXT学位

总监授课，免费试学，  
入职BAT不遥远！

立即前往



#### 最新IT新闻：

- 圆通速递新标志亮相：新图标更加简洁 开启“二次创业”
  - 推出自己的加密货币很可能是Facebook的下一步举措
  - 大股东软银敲定Uber两位董事 含运营商Sprint之CEO
  - 知乎VS悟空问答：精英们的投票体系和快手们的问答社区
  - Python2.7 “退休”倒计时 预计2020年不再提供维护
- » [更多新闻...](#)

告别高昂运维费用 云计算全面助力

40+款核心产品免费半年 再+8000津贴任意采购

立即申请

广告

#### 最新知识库文章：

- 步入云计算
- 以操作系统的角度述说线程与进程

- [软件测试转型之路](#)
- [门内门外看招聘](#)
- [大道至简，职场上做人做事做管理](#)
- » [更多知识库文章...](#)

2

 推荐

0

 反对