Difference between Observer, Pub/Sub, and Data Binding

不同的观察者，酒吧/子，和数据绑定

Tags: model-view-controller design-patterns data-binding observer-pattern publish-subscribe source

标签: model-view-controller design-patterns data-binding observer-pattern publish-subscribe source

问题 (Question)

What is the difference between the **Observer Pattern**, **Publish/Subscribe**, and **data binding**?

I searched around a bit on SO and did not find any good answers.

What I have come to believe is that data binding is a generic term and there are different ways of implementing it such as the Observer Pattern or the Pub/Sub pattern. With the Observer pattern, an Observable updates its Observers. With Pub/Sub, 0-many publishers can publish messages of certain classes and 0-many subscribers can subscribe to messages of certain classes.

Are there other patterns of implementing "data binding"?

之间的区别是什么**观察者模式**，**发布/订阅**，和**数据绑定**?

我到处寻找一点所以并没有发现任何好的答案。

我终于相信，数据绑定是一个通用的术语有不同的实施如观察者模式或发布/订阅模式的方法。与观察者模式，观察更新其观察员。与Pub/Sub，出版商可以发布0-many某些类和用户信息可以订阅0-many某类信息。

有没有其他的实现“数据绑定”模式?

最佳答案 (Best Answer)

Here's my take on the three:

Data Binding:

Essentially, at the core this just means "the value of property X on object Y is semantically bound to the value of property A on object B. No assumptions are made as to how Y knows or is fed changes on object B.

Observer, or Observable/Observer:

A design pattern by which an object is imbued with the ability to notify others of specific events - typically done using actual events, which are kind of like slots in the object with the shape of a specific function/method. The observable is the one who provides notifications, and the observer receives those notifications. In .net, the observable can expose an event and the observer subscribes to that event with an "event handler"shaped hook. No assumptions are made about the specific mechanism which notifications occur, nor about the number of observers one observable can notify.

Pub/Sub:

Another name (perhaps with more "broadcast" semantics) of the Observable/Observer pattern, which usually implies a more "dynamic" flavor - observers can subscribe or unsubscribe to notifications and one observable can "shout out" to multiple observers. In .net, one can use the standard events for this, since events are a form of MulticastDelegate, and so can support delivery of events to multiple subscribers, and also support unsubscription. Pub/sub has a slightly different meaning in certain contexts, usually involving more "anonymity" between event and eventer, which can be facilitated by any number of abstractions, usually involving some "middle man" (such as a message queue) who knows all parties, but the individual parties don't know about each other.

Data binding, redux:

In many "mvc-like" patterns, the observable exposes some manner of "property changed notification" that also contains information about the specific property changed. The observer is implicit, usually created by the framework, and subscribes to these notifications via some binding syntax to specifically identify an object and property, and the "event handler" just copies the new value over, potentially triggering any update or refresh logic.

Data binding re redux:

An alternative implementation for data binding? Ok, here's a stupid one:

- a background thread is started that constantly checks the bound property on an object.
- if that thread detects that the value of the property has changed since last check, copy the value over to the bound item.

这是我在三以:

数据绑定:

基本上，在核心这只是意味着“财产x对y的值对象的语义约束的属性的对象没有假设了Y知道或价值是美联储改变对象B。

观察，或观察/观察者:

一个设计模式的对象具有特定的事件通知其他使用实际事件通常做的能力，这是种在一个特定的函数/方法的对象的形状的槽一样。观察到的是一个谁提供的通知，和观察者接收这些通知。在网上，可观察到。可以将事件和观察者订阅该事件与一个“事件处理程序”形钩。没有假设的具体机制，发生通知，也没有关于观察者观察的数量可以通知。

发布/订阅:

另一个名字（也许更多的“广播”的语义）的观察/观察者模式，这通常意味着更多的“动态”的味道——观察员可以订阅或退订通知和观察能喊出“多观察家。在。净，可以使用标准的事件，因为事件是一种形式的multicastdelegate，并能提供事件的多个用户，并支持退订。酒吧/子有一定语境中的含义有所不同，通常包括更多的“匿名”事件和事件之间，它可以由任何数量的抽象了，通常包括一些“中间人”（如消息队列）谁知道所有的当事人，但当事人不知道对方。

数据绑定，终极版:

在许多“MVC”模式，可观察到的暴露了某种“属性更改通知”，还包含有关特定属性的改变。观察者是隐性的，通常由框架创建，并订阅这些通知通过绑定语法具体确定的对象和属性，以及“事件处理程序”仅复制新的价值，有可能引发任何更新或刷新的逻辑。

数据绑定重新归来:

数据绑定的一个可选的实现？好吧，这是一个愚蠢的人:

- 一个后台线程启动，不断检查绑定属性的对象。
- 如果该线程检测到财产的价值从去年检查改变，值复制到绑定物品。

答案 (Answer) 2

Observer pattern is mostly implemented in a synchronous way, i.e. the observable calls the appropriate method of all its observers when some event occurs. The Pub/Sub pattern is mostly implemented in an asynchronous way (using message queue).

Also, in the observer pattern, the observers are aware of the observable. Whereas, in Pub/Sub, neither the publishers, nor the consumers need to know each other. They simply communicate with the help of message queues.

As you mentioned correctly, data binding is a generic term and it can be implemented using either Observer or Pub/Sub method. Data is the observable/publisher.

观察者模式主要是以同步的方式实现的，即观察到的调用相应的所有观察员的方法，当某些事件发生。发布/订阅模式主要是以异步的方式实现（使用消息队列）。

同时，在观察者模式，观察者有意识的观察。然而，在发布/订阅，无论是出版商，还是消费者需要知道对方。他们只是用消息队列的帮助沟通。

正如你所说的正确，数据绑定是一个通用的术语，它的实施可以使用观察者或发布/订阅的方法。数据是观察/出版商。

答案 (Answer) 3

I agree with your conclusion about both patterns, nevertheless, for me, I use Observable when I'm in the same process and I use the Pub/Sub in inter-process scenarios, where all parties only know the common channel but not the parties.

I don't know other patterns, or let me say this way, I've never needed another patterns for this task. Even most MVC frameworks and data binding implementations use usually internally the observer concept.

If you're interested in inter-process communication, I recommend you:

"Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions" (http://www.addison-wesley.de/9780321200686.html)

This book contains a lot of ideas about how to send messages between processes or classes that can be used even in intra-process communication tasks (it helped me to program in a more loose-coupled way).

I hope this helps!

我同意你对这两种模式，结论不过，对我来说，我观察到当我在相同的过程和我在进程间的方案使用发布/订阅，双方都只知道公共信道而不是当事人。

我不知道其他模式，或让我说这样的话，我从来都不需要另一个模式，这个任务。即使大多数的MVC框架和数据绑定实现使用通常内部观察者的概念。

如果你在进程间通信感兴趣，我建议你：

“企业集成模式：设计，建筑，和部署通信解决方案“http://www.addison-wesley.de/9780321200686.html）

这本书有很多想法如何发送消息的过程或类之间，甚至可以用在进程内通信任务（它帮助我计划在一个更松散耦合的方式）。

我希望这有助于！

相关问题

- Qt 5.2 Model-View-Pattern: How to inform model object about changes in underlying data structure [Qt 5.2 Model-View-Pattern:如何告知模型对象底层数据结构的变化]
- Should a model object instantiate inside a controller class? [该模型对象实例化内部类控制器？]
- PHP MVC - Why passing controller instance to view class? [PHP MVC为什么通过控制器实例视图类？]
- Where does the code go to link View to Model and View to Controller? [代码是否哪里去链接视图模型和视图控制器？]
- MVC, can Model have a Controller instance? [MVC模式，可以有一个控制器的例子吗？]
- Backend and frontend MVC [后端和前端MVC]
- Design Pattern - Objective-C - MVC Model View Controller [设计模式和MVC模型视图控制器]
- What type of View am I using in my MVC application? [什么类型的观点我使用在我的MVC应用程序吗?]
- How do I make a GUI using the model/view/controller method? [我如何使用模型/视图/控制器的方法使一个GUI？]
- Interacting with other objects in a text-based game in Java [在Java中其它对象交互的基于文本的游戏]

相关文章

- 谈谈关于JavaScript 中的 MVC 模式
- .NET Web开发之.NET MVC框架介绍
- ASP.NET小结之MVC, MVP, MVVM比较以及区别（一）
- 我认为JSP有问题(上)
- 指南：想成为一个JSP网站程序员吗？
- 困扰JSP的一些问题与解决方法
- 加快JDBC设计中JSP访问数据库
- Java学习的捷径
- PHP 开源框架22个简单简介
- ASP.NET MVC3-第02节-添加一个Controller (C#)

- Android照片墙应用实现，再多的图片也不怕崩溃
- Android高效加载大图、多图解决方案，有效避免程序OOM
- Android瀑布流照片墙实现，体验不规则排列的美感
- Android多点触控技术实战，自由地对图片进行缩放和移动
- 什么是领域驱动设计(Domain Driven Design)?
- 在 Linux 下你所不知道的 df 命令的那些功能
- 只有几百个字节大小的国际象棋程序
- Java 性能优化手册 -- 提高 Java 代码性能的各种技巧
- 5个最佳的Android测试框架（带示例）