

Huỳnh Thanh Hải – haihth

About Huỳnh Thanh Hải

DP4Dummies – Chương 4: Observer, Chain of Responsibility

Posted on [Tháng Hai 22, 2013](#)

Chương 4: “Chuyện gì đang diễn ra” mẫu Observer và mẫu Chain of Responsibility

Trong chương này, chúng ta sẽ nói về:

- Sử dụng mẫu Observer (Mẫu quan sát)
- Tạo một đối tượng quan sát Observer và đối tượng bị quan sát Subject
- Sử dụng lớp Observer Java
- Sử dụng giao diện Observer Java
- Sử dụng mẫu Chain of Responsibility

Ông chủ bước vào văn phòng và nói: “Ai đó đang chỉnh sửa dữ liệu trên máy chủ, sao tôi không được thông báo?”

“Bằng cách nào?” Bạn hỏi lại: “Ông muốn chúng tôi thông báo khi có bất kỳ sự thay đổi nào trên máy chủ à?” Bạn biết ông chủ vừa mới tiếp cận hệ thống này, nhưng không ngờ ông chủ lại hỏi những câu kì lạ vậy.

“Đúng vậy”. Ông chủ nhấn mạnh. “Tôi muốn được thông báo khi có bất kỳ sự thay đổi nào trên dữ liệu ở máy chủ. Tôi muốn biết việc gì đang diễn ra quanh đây!”

“Ý ông muốn nói là gọi ông một báo cáo?”

“Đúng”

“Ok” Bạn nói “Tôi có một ý tốt hơn. Ông nghĩ sao khi tôi sử dụng mẫu thiết kế Observer (Người quan sát), và đăng ký ông với máy chủ dữ liệu như là một người quan sát dữ liệu Database Observer.

“Là sao?” Ông chủ hỏi.

“Ông sẽ được thông báo bất cứ khi nào có sự thay đổi dữ liệu trên máy chủ.” Bạn nói “Không cần báo cáo gì cả. Tất cả đều được viết tự động trong mã nguồn”

“Đó là tất cả những gì tôi muốn,” Ông chủ nói và đi khỏi.

Bạn mỉm cười với chính mình và tự hỏi. Không biết khuôn mặt ông chủ sẽ thế nào khi hàng ngày nhận được 200,000 thông báo về sự thay đổi dữ liệu ở máy chủ. Thế nhưng, với mẫu Observer, việc viết mã sẽ không có khó khăn gì.

Chương này nói về một đối tượng cần quan sát. Khi đối tượng có thay đổi, nó sẽ thông báo cho tất cả các Observer (đối tượng quan sát) những thay đổi đó. Chúng ta nói về hai mẫu, mẫu Observer và mẫu Chain of Responsibility.

Mẫu Observer cho phép nhiều đối tượng quan sát Observer nhận được thông báo, khi đối tượng bị quan sát có thay đổi. Từng Observer phải đăng ký với chủ thể, khi chủ thể thay đổi, nó sẽ thông báo tới tất cả các Observer. Các Observer được thông báo cùng lúc.

Mẫu Chain of Responsibility tương tự vậy, nhưng các Observer được thông báo theo thứ tự trước sau. Hết Observer này tới Observer khác, giúp cho từng Observer xử lý thông báo của theo cách của riêng mình.

Thông báo cho Observer với mẫu Observer.

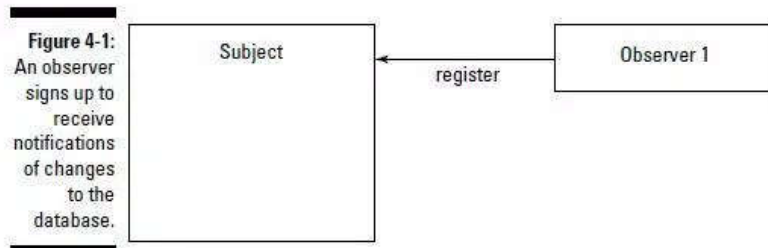
Ông chủ muốn được báo cáo mỗi khi dữ liệu trên máy chủ có thay đổi. Bạn cũng muốn ghi nhận lại sự thay đổi này. Người thực hiện việc thay đổi cũng muốn biết việc thay đổi có thành công hay không?

Vì vậy, bạn cần có một tập hợp nhiều Observer để quan sát những gì đang diễn ra. Điều đó đúng với tên gọi của mẫu là “Người quan sát”. Cho phép đối tượng bị quan sát (máy chủ), thông báo tới các đối tượng quan sát (ông chủ, người sử dụng...) những gì đang diễn ra.

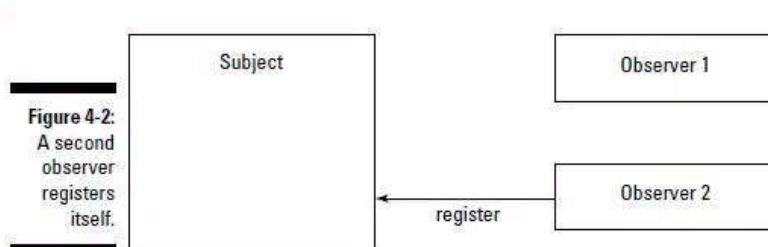
Mẫu thiết kế Observer nói về việc gửi thông báo cập nhật tới các đối tượng quan sát. Bạn có thể thêm hoặc xóa 1 đối tượng quan sát Observer trong khi hệ thống thực thi. Khi có sự thay đổi diễn ra, tất cả các Observer đều nhận được thông báo.

Theo sách của GOF (Design Patterns: Elements of Reusable Object-Oriented Software, xuất bản năm 1995) định nghĩa mẫu Observer như sau: “Xây dựng mối quan hệ một nhiều giữa các đối tượng, và khi đối tượng chủ thay đổi, tất cả các đối tượng phụ thuộc sẽ được thông báo một cách tự động”

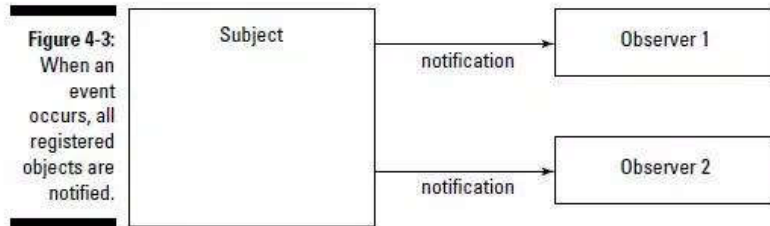
Đây là cách thức mẫu Observer hoạt động. Một observer đăng ký với chủ thể như hình dưới



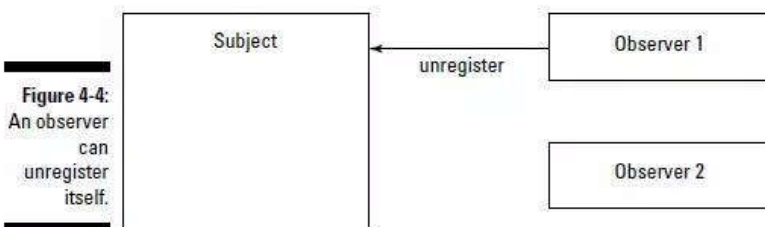
Chủ thể lưu lại thông tin của Observer này. Sau đó một observer khác, Observer 2 cũng đăng ký với chủ thể như hình dưới:



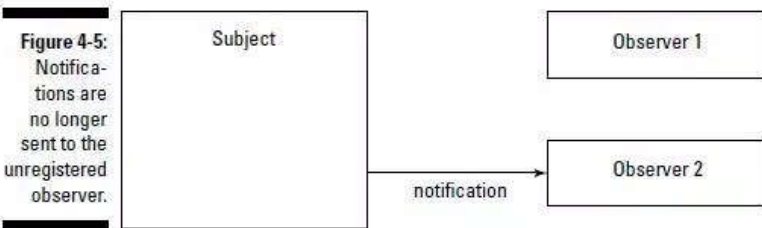
Vào thời điểm này, chủ thể (máy chủ dữ liệu), đang lưu trữ thông tin của 2 Observer. Khi có sự thay đổi trên máy chủ (dữ liệu bị thay đổi...), nó sẽ tự động gọi thông báo cho 2 Observer như hình dưới:



Đối tượng quan sát Observer (ông chủ, người quản trị...) có thể gỡ đăng ký ra khỏi chủ thể (máy chủ) bất cứ lúc nào nó muốn. Lúc đó nó sẽ không nhận được thông báo từ chủ thể nữa. Theo như hình sau:



Bây giờ Observer 1 sẽ không nhận được bất kỳ thông báo nào từ máy chủ nữa. Khi máy chủ có sự thay đổi, nó chỉ gọi thông báo cho Observer 2 như hình sau:



Tuy nhiên, Observer 1 có thể tái đăng ký với máy chủ bất cứ khi nào cần thiết. Khi đó nó lại được đưa vào danh sách nhận thông báo.

Cách tạo một giao diện chủ thể (Subject Interface)

(ND: Interface là một khái niệm trừu tượng trong lập trình hướng đối tượng, là một khái niệm tổng quát hơn, thô sơ hơn của một lớp. Nó định nghĩa hình dáng của một lớp. Còn được coi như một hợp đồng. Thông thường nó quy định lớp nào hiện thực interface này phải có những thuộc tính nào, phương thức nào...)

Khi bạn hiện thực một mẫu thiết kế, phương pháp tốt nhất thường là bắt đầu từ việc tạo một interface. Điều này đảm

bảo cho những lớp bạn định tạo luôn phù hợp với mẫu thiết kế. Đặc biệt là trong trường hợp bạn phải tạo nhiều lớp khác nhau, việc tạo một interface gần như bắt buộc. Hiện thực(implement) một interface giúp cho mã của bạn sáng sủa và dễ hiểu hơn.

Khi viết mã cho mẫu Observer, bạn tạo một interface hay abstract class cho đối tượng Observer. Điều này giúp cho việc tạo nhiều Observer luôn nhất quán với nhau.

(ND: Abstract class, còn gọi là lớp trừu tượng, là một khái niệm trong lập trình hướng đối tượng, nó nằm ở giữa interface và class. Nó chi tiết hơn Interface, nhưng tổng quát cao hơn Class. Interface chỉ định nghĩa hình dáng của lớp, Abstract class vừa định nghĩa hình dáng một lớp, vừa chứa đựng cả phần hiện thực của lớp)

Trong ví dụ sau, tôi sẽ tạo một interface cho chủ thể Subject (Đối tượng cần quan sát), interface này liệt kê các phương thức của Subject cần có. Subject này cần phải có ít nhất 3 hàm. Hàm registerObserver để đăng ký Observer, hàm removeObserver để gỡ đăng ký 1 Observer, và hàm notifyObservers để thông báo cho tất cả các Observer biết khi có sự thay đổi trên Subject.

```
public interface Subject
{
    public void registerObserver(Observer o);
    public void removeObserver(Observer o);
    public void notifyObservers();
}
```

Interface trên liệt kê các hàm mà chủ thể (máy chủ dữ liệu), cần phải có. Tiếp theo, ta sẽ định nghĩa 1 interface cho Observer.

Tạo một giao diện quan sát Observer Interface

Việc này tương đối dễ dàng, interface cho Observer chỉ cần 1 hàm, hàm này sẽ thực hiện khi Observer nhận được thông báo từ máy chủ. Tôi đặt tên cho nó là hàm cập nhật update. Trong ví dụ này, bạn chuyển những thay đổi của máy chủ (như xóa, sửa, tạo ...) và tên record đã thay đổi vào hàm update này.

```
public interface Observer
{
    public void update(String operation, String record);
}
```

(ND: tôi giữ nguyên chữ record trong văn bản gốc. Thông thường các database bao gồm nhiều bảng, mỗi bảng bao gồm nhiều hàng, nhiều cột. Một record chính là 1 hàng trong bảng. Khi lập trình ta thường ánh xạ 1 bảng thành 1 lớp. 1 hàng trong bảng, chính là 1 đối tượng. Khi ta thay đổi đối tượng này, chính là đang thay đổi 1 record. Tôi luôn phân vân khi dịch một từ chuyên ngành tin học sang tiếng việt, vì thực sự chúng ta chưa xây dựng một từ điển chuẩn cho tin học, điều này cần phải huy động lực lượng quốc gia. Tôi thấy rằng để nguyên từ tiếng anh thì dễ hiểu và chuẩn mực. Bạn nghĩ sao khi tôi dịch từ implement thành hiện thực, instance thành thể hiện, record thành bản ghi...Có gì đó không ổn, nên có lúc tôi giữ nguyên từ tiếng anh, có lúc tôi chuyển ngữ, mong rằng các độc giả thông cảm bỏ qua)

Khi observer hiện thực hàm update này, máy chủ có thể chuyển tới observer tên record và những thay đổi trên record đó.

Mọi việc bắt đầu tốt đẹp. Tiếp theo tôi sẽ tạo một đối tượng Database(máy chủ dữ liệu), làm đối tượng bị quan sát, nó sẽ lưu trữ thông tin về các observer (đối tượng quan sát), và thông báo cho observer biết khi có gì thay đổi diễn ra.

Tạo một đối tượng cần quan sát – chủ thể Subject

Chủ thể cho phép một Observer đăng ký, và phải thông báo cho Observer biết khi có sự kiện xảy ra. Theo như giao diện Subject Interface ở trên, có 3 hàm mà Subject cần phải hiện thực là registerObserver, removeObserver, notifyObserver.

Về quảng cáo

Share this:



Be the first to like this.

Related

DP4Dummies - Chương 8: Iterator, Composite
In "Design Patterns For Dummies"

DP4Dummies - Chương 6: Adapter, Facade
In "Design Patterns For Dummies"

DP4Dummies - Chương 9: State, Proxy
In "Design Patterns For Dummies"

[Theo dõi](#)

This entry was posted in [Design Patterns For Dummies](#)

Follow “Huỳnh Thanh Hải – haihth”

Click the [permalink](#).

Có 18 phản hồi tại DP4Dummies

Get every new post delivered to your Inbox.

Join 30 other followers



thunder55 nói:

Tháng Tư 9, 2013 lúc 2:34 sáng

Mình mong chờ chương 4 update

[Trả lời](#)

Build a website with WordPress.com

er, Chain of Responsibility

ẻ học hỏi của chủ blog thật đáng học hỏi,thank chủ blog..



haihth nói:

Thứ Tư 23, 2013 lúc 10:11 sáng

Chào bạn thunder55, mình dịch series này theo cảm hứng và thời gian rảnh rỗi trên cty. Mình đã post tiếp, bạn xem thử nha. Vẫn còn nữa. Đang dịch.

[Trả lời](#)



thunder55 nói:

Thứ Năm 4, 2013 lúc 5:53 chiều

Vẫn luôn theo dõi update mới từ blog của bạn haihthcảm ơn những chia sẻ quý báu của bạn.

[Trả lời](#)



Nhật Tiến nói:

Thứ Năm 5, 2013 lúc 4:15 chiều

chào bạn mình hiện đang thực hiện đề tài giới thiệu quát về Design Patter ,tình cờ thì gặp những tài liệu tham khảo của bạn....nhưng mình vẫn còn vài chỗ không hiểu lắm...mong bạn có thể trao đổi chia sẻ vs mình qua

y!m: nhattien_o1_bd

gmail: nhattien.hnt@gmail.com

cảm ơn bạn trước rất nhiều

hoặc bạn có thể reply lại mail của bạn để thuận tiện cho việc trao đổi

[Trả lời](#)



haihth nói:

Thứ Sáu 6, 2013 lúc 2:03 chiều

Hi

Mình đã email cho bạn.

[Trả lời](#)



izplay nói:

Thứ Sáu 12, 2013 lúc 3:51 chiều

Sao chưa có update nữa vậy anh

[Trả lời](#)



haihth nói:

Thứ Bảy 14, 2013 lúc 11:30 sáng

Chưa có time em ơi.

Còn chương 4 và chương 9. Thông cảm nha.

[Trả lời](#)



Lập Trình .NET nói:

Thứ Bảy 23, 2013 lúc 9:28 sáng

Nếu các bạn không chê thì tôi có thể dịch tiếp chương này

[Trả lời](#)



haihth nói:

Tháng Năm 23, 2013 lúc 9:39 sáng

Rất hoan nghênh bạn dịch tiếp.

Thân mến

[Trả lời](#)



Gara nói:

Tháng Năm 28, 2013 lúc 11:38 chiều

Thanks a Thanh Hải nhiều, bài viết rất hay.

[Trả lời](#)



Minh nói:

Tháng Sáu 10, 2013 lúc 3:25 chiều

Bài dịch của a tương đối dễ hiểu.

mà e muốn hỏi thêm 1 số điều liên quan đến parttern. E có thể liên hệ với a hải qua yahoo được ko ạ.

yahoo : echip_aslonely@yahoo.com.vn

email : ericnguyen275@gmail.com

Khi nào có thời gian thì a email hoặc liên hệ vs yahoo cho e.

Thanks a Hải

[Trả lời](#)



haihth nói:

Tháng Sáu 11, 2013 lúc 7:18 sáng

Chào em,

Email của anh: huynhthanhhai@gmail.com

[Trả lời](#)



thong nói:

Tháng Bảy 21, 2013 lúc 10:45 sáng

Bài dịch rất hay. Nhưng vẫn chưa có update à anh. Thanks

[Trả lời](#)



tùng nói:

Tháng Tám 9, 2013 lúc 5:41 chiều

đang mong bài update của a

[Trả lời](#)

**haihth** nói:

Tháng Tám 10, 2013 lúc 7:51 sáng

Dạo này a bận quá, hi vọng sẽ sớm có thời gian update cho xong C4, C9. Kết thúc quyền này.

[Trả lời](#)**Nguyễn Thế Hoàng** nói:

Tháng Tám 28, 2013 lúc 3:56 chiều

Ừa mình có đọc thấy phần giới thiệu có nói 12 chương mà ^^

**javadp** nói:

Tháng Mười 13, 2013 lúc 11:20 chiều

anh ơi dịch nốt đi, đang đọc hay, cái này toàn tài liệu tiếng anh thôi ạ.

[Trả lời](#)**Hùng Sang** nói:

Tháng Mười 22, 2013 lúc 10:58 chiều

Update đi a ơi. Đang hay mà a.

[Trả lời](#)

Huỳnh Thanh Hải – haihth*The Twenty Ten Theme. Tạo một website miễn phí hoặc 1 blog với WordPress.com.*