DP4Dummies – Chương 4: Observer, Chain of Responsibility

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 kì 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 gởi 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 gởi 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

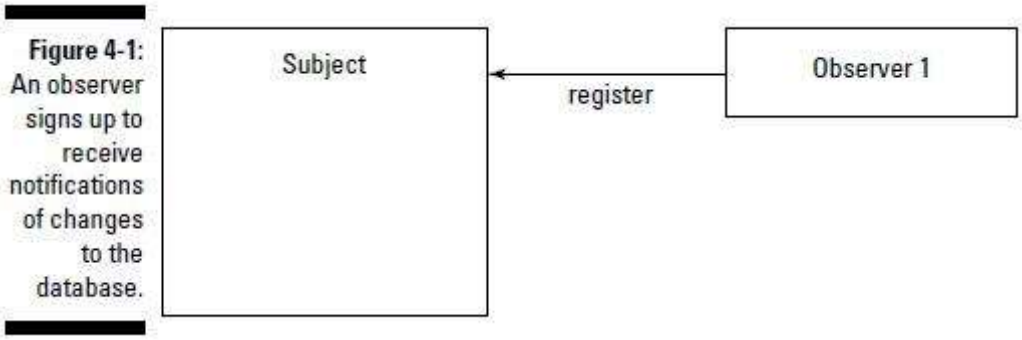

Figure 4-1: An observer signs up to receive notifications of changes to the database.

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:

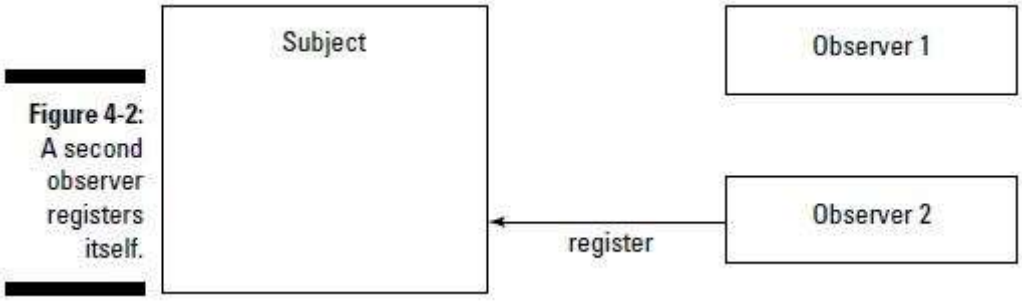

Figure 4-2: A second observer registers itself.

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 gởi thông báo cho 2 Observer như hình dưới:

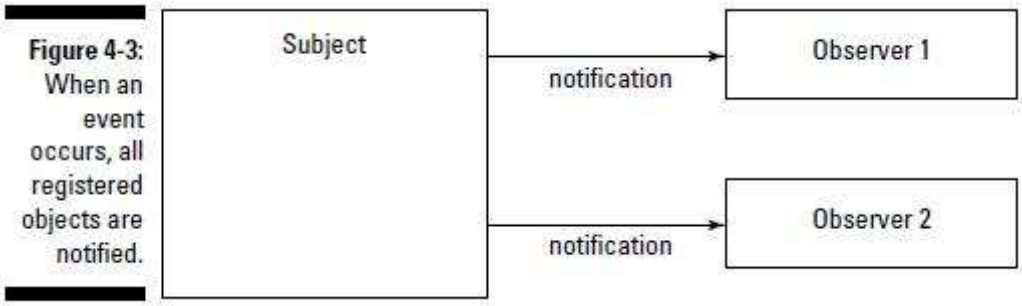

Figure 4-3: When an event occurs, all registered objects are notified.

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:

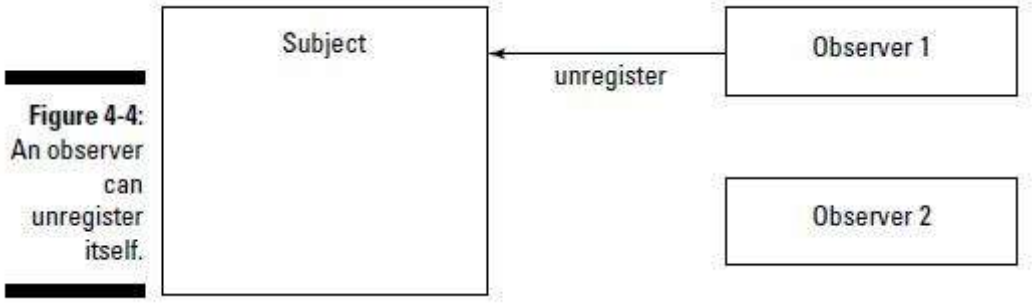

Figure 4-4: An observer can unregister itself.

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 gởi thông báo cho Observer 2 như hình sau:

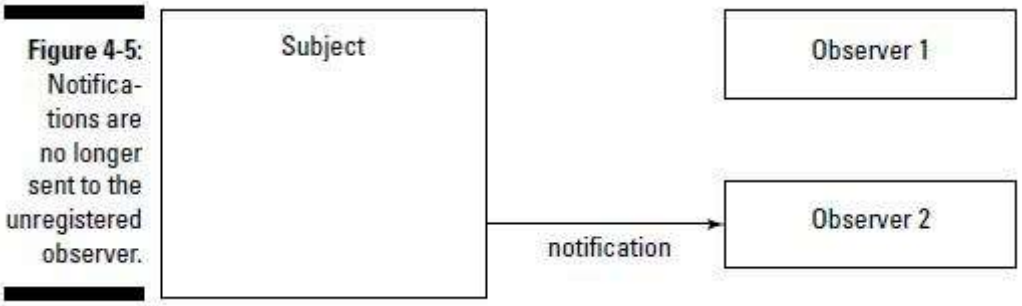

Figure 4-5: Notifications are no longer sent to the unregistered observer.

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 thức 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 đọc 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.

Design Patterns For Dummies
Design Patterns
< DP4Dummies – Chương 3: Decorator, Factory
> DP4Dummies – Chương 5: Singleton, Flyweight