Các mẫu Proxy được sử dụng khi bạn cần để biểu diễn một đối tượng phức tạp hay tốn thời gian để tạo ra bằng một thứ đơn giản hơn.

Nếu việc tạo ra một đối tượng mới mất nhiều thời gian hay tài nguyên máy tính, Proxy cho phép chúng ta trì hoãn quá trình tạo đối tượng cho đến khi chúng ta cần đối tượng thực sự.

Một mẫu Proxy thường có chung các phương thúc giống như đối tượng mà nó đại diện, và một khi đối tượng được nạp vào bộ nhớ, các lời gọi hàm đến đối tượng thực sự này sẽ được chuyển tiếp qua Proxy.

Vài trường hợp mà mẫu Proxy có thể hữu dụng:

1. Một đối tượng, chẳng hạn như một file ảnh, mất quá nhiều thời gian để load.

2. Kết quả của một tính toán cần nhiều thời gian để hoàn thành, và bạn cần hiển thị các kết quả ngay lập tức trong khi quá trình tính toán vẫn đang tiếp tục.

3. Đối tượng nằm ở trên một máy remote, và tải nó thông qua mạng có thể mất nhiều thời gian, đặc biệt trong những thời điểm mạng lag.

4. Quyền truy cập đối tượng bị hạn chế, và proxy có thể xác nhận quyền truy cần của người dùng.

Các Proxy có thể được sử dụng để phân biệt giữa việc yêu cầu một thể hiện của đối tượng và quá trình truy xuất thực sự vào thể hiện đó.

Ví dụ, chương trình khởi tạo có thể tạo ra một số lượng các đối tượng mà một trong số chúng có thể chưa được sử dụng ngay. Trong trường hợp này, một proxy có thể nạp một đối tượng thực sụ chỉ khi cần thiết.

Giả sử trong trường hợp chương trình cần nạp và hiển thị một file ảnh có dung lượng lớn. Khi chương trình khởi động, cần có một vài dấu hiệu cho người dùng thấy rằng bức ảnh đó sẽ được hiển thị trên màn hình và nằm đúng vị trí, nhưng thực sự thì việc hiển thị hình ảnh sẽ bị trì hoãn lại cho đến khi nào việc nạp bức ảnh hoàn tất.

Điều này đặc biệt quan trọng trong các chương trình xử lí văn bản hay các trình duyệt Web khi chúng đặt các dòng chữ nằm xung quanh các bức ảnh thậm chí trước khi cả tấm ảnh đó được hiển thị.

Một proxy hình ảnh có thể ghi chứ lại tấm ảnh và bắt đầu nạp ngầm nó trong khi vẽ ra một khung hình chữ nhật đơn giản hay bất cứ biểu tượng cho thấy rẳng bức ảnh đang được hiển thị trên màn hình trước khi nó xuât hiện. Proxy thậm chí còn có thể trì hoãn việc nạp bức ảnh cho đến chỉ khi nó nhận được yêu cầu vẽ tấm ảnh đó ra sau đó bắt đầu công việc của mình.

***Code Mẫu***

Trong ví dụ này, chúng ta tạo ra một chương trình đơn giản nhằm hiển thị một tấm ảnh khi nó được load lên bằng Image control. Thay vì nạp trực tiếp tấm ảnh, chúng ta sẽ sử xài một class mà ta gọi là ImageProxy để hoãn lại việc này lại và chỉ vẽ lên một hình chữ nhật cho đến khi nó hoàn tất.

private void init() {

imageProxy = new ImageProxy ();

}

public Form() {

InitializeComponent();

init();

}

private void buttonDraw\_Click(object sender, EventArgs e) {

Pic.Image = imageProxy.getImage();

}

Chú ý rằng chúng ta tạo ra một thể hiện của ImageProxy như mong muốn cho Image của chúng ta. Lớp ImageProxy sẽ chăm sóc việc tạo tải ảnh và tạo một đối tượng Imager trong quá trình tải. Nó sẽ trả về một lớp hiện thực giao diện Imager.

public interface Imager {

Image getImage() ;

}

Trong trường hợp đơn giản này, lớp ImageProxy chỉ việc delay 5 giây và sau đó chuyển từ hình ảnh đang dang dở sang hình hoàn chỉnh. Nó làm điều đó bằng cách sử dụng một thể hiện của lớp Timer. Timerđược quản lý bởi một lớp TimerCallback nơi định nghĩa một phương thức gọi là time ticks.

Điều này cũng y như việc chúng ta sẽ thêm một event handler khác mà thôi. Và phương thưc callback này, "timerCall", đặt lại cờ và tắt timer đi.

public class ImageProxy {

private bool isDone;

private Timer timer;

public ImageProxy() {

//create a timer thread and start it

timer = new Timer(new TimerCallback(timerCall), this, 5000, 0);

}

//called when timer completes

private void timerCall(object obj) {

isDone = true;

timer.Dispose();

}

public Image getImage() {

Imager img;

if (isDone)

img = new FinalImage();

else

img = new QuickImage();

return img.getImage();

}

}

Chúng ta hiện thực giao diện Imager bằng hai lớp dễ thương mà chúng ta sẽ gọi chúng là QuickImage và FinalImage. Nhừ tên gọi, một bé sẽ lưu trữ một tấm ảnh gif và đứa kia sẽ lưu trữ tấm ảnh (to bự và load chậm) jpeg.

Trong C#, Image là một lớp trừu tượng, và các lớp Bitmap, Cursor, Icon, Metafile được dẫn xuất từ nó. Do đó lớp thực sự mà chúng ta sẽ trả về sẽ là một lớp được dẫn xuất từ Image. Lớp QuickImage trả về một Bitmap từ file gif, và tấm ảnh cuối cùng từ file jpeg.

public class QuickImage : Imager {

public QuickImage() {}

public Image getImage() {

return new Bitmap ("Box.gif");

}

}

public class FinalImage : Imager {

public FinalImage() {}

public Image getImage() {

return new Bitmap("flowrtree.jpg");

}

}

Khi chương trình thực hiện, chúng ta hiển thị hình ảnh từ file Dox.gif (hay quick image theo định nghĩa lớp), và sau đó 5 giây, chúng ta gọi phương thức getImage() lần nữa, ta sẽ có được tấm hình sau cùng (flowrtree.jpg).

Hai trạng thái của chương trình được mô tả như hình 20-1

<< chụp hình 2o - 1 trong sách bỏ vô >>

***Các Proxy trong C#***

Bạn sẽ thấy có nhiều hành vi tương tụ như proxy trong C# hơn các ngôn ngữ khác, bởi vì chúng được tạo ra nhằm mục đích network và Internet.

Chẳng hạn, tất cả các lớp database connection trong ADO.NET đều là các proxy hoạt động hiệu quả.

***Copy-on-Write***

Bạn cũng có thể sử dụng proxy để giữ các bản copy của cac đối tượng kích thước lơn mà có thể chúng sẽ không thay đổi trong suốt quá trình hoạt động của chương trình. Nếu bạn tọa ra một thể hiện mới của một đối tượng đắt giá (về dung lượng hay về tốc độ), một Proxy có thể quyết định rằng chẳng có lý gì để tạo ra thêm một copy cả.

Nó chỉ đơn giản là sử dụng lại đối tượng gốc đã có sẵn. Nếu chương trình thay đổi trong một bản sao mới, Proxy có thể copy đối tượng gốc và cập nhập các thay đổi này trên một thể hiện mới của đối tượng. Điều này có thể tiết kiệm rất nhiều thời gian và bộ nhớ khi các đôi tượng không thương xuyên thay đổi sau khi chúng được khởi tạo.

***So sánh với các mẫu liên quan***

Cả Adapter và Proxy đều tạo nên một lớp wrapper bao quanh đối tượng. Tuy nhiên, Adapter cung cấp một giao diện khác cho đối tượng, trong khi Proxy tạo ra một giao diện giống nhau cho đối tượng nhưng lại xen vào khi nó có cơ hội nhằm trì hoãn quá trình hay vì hiệu suất truyền tải.

Decorator cũng có các giao diện tương tự như đối tượng được bao, tuy nhiên mục đich chính của nó là thêm vào (đôi khi trực quan) các chức năng cho đối tượng gốc. Một Proxy, ngược lại, kiểm soát việc truy xuất đến lớp được nó bao bọc.

***Câu hỏi để suy nghĩ***

1: Bạn đã thiết kế một máy chủ để kết nối đến CSDL, nếu như một vài server kết nối đến máy chủ của bạn cùng một lúc, Proxy sẽ có thể giúp chúng ta như thế nào?