Proxy Pattern

Tomás Felipe Melli

June 10, 2025

Índice

1 Intent

El patrón **Proxy** proporciona un sustituto u objeto representante (proxy) de otro objeto para controlar el acceso a él. También se conoce como **Surrogate**.

2 Motivation

Queremos insertar imágenes en un documento de texto, pero cargarlas en memoria solo cuando sea necesario (por ejemplo, cuando se dibujan por primera vez). El objeto `ImageProxy` actúa como intermediario entre el documento y el objeto `Image` real.

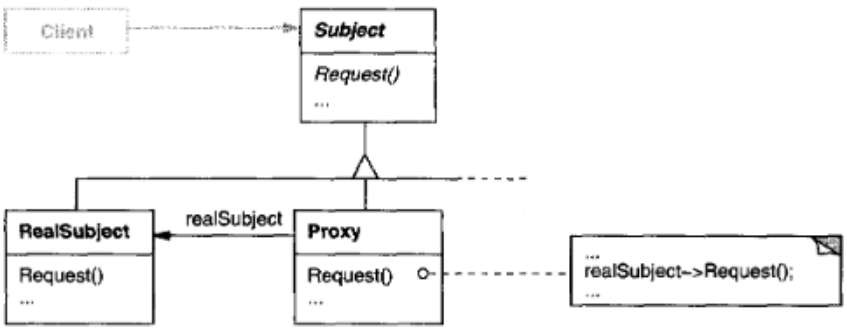
El proxy guarda la ruta al archivo, las dimensiones de la imagen, y crea la instancia real solo cuando se necesita.

3 Applicability

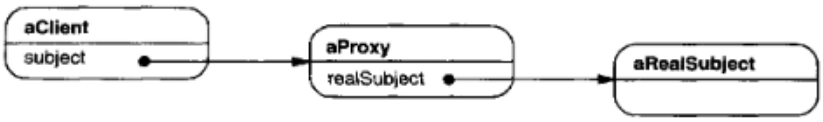
Usar Proxy cuando:

- Querés controlar el acceso a un objeto complejo o costoso de instanciar.
- Querés agregar lógica adicional al acceder al objeto (como autenticación, carga diferida, o control remoto).
- Querés proteger o encapsular el acceso a un objeto real desde el cliente.

4 Structure



Here's a possible object diagram of a proxy structure at run-time:



5 Participants

- **Proxy**: (`ImageProxy`)
  - Mantiene una referencia al `RealSubject`.
  - Implementa la interfaz `Subject` para que el cliente no note la diferencia.
  - Controla el acceso al objeto real. Por ejemplo:
    - Creando el objeto sólo cuando es necesario (proxy virtual).
    - Verificando permisos de acceso (proxy de protección).
    - Controlando acceso remoto (proxy remoto).
    - Añadiendo conteo de referencias u otras funcionalidades (proxy inteligente).
  - Puede realizar tareas antes o después de delegar la solicitud al objeto real.

- **Subject**: Declara la interfaz común entre el Proxy y el RealSubject. (`Graphic`)
- **RealSubject**: El objeto real al cual el proxy representa (`Image`).

6 Example

Contexto del problema

Un editor de texto permite insertar imágenes en un documento, pero muchas imágenes pueden ocupar memoria innecesariamente si se cargan todas de entrada. Queremos posponer la carga de la imagen real hasta que se necesite dibujarla.

Interfaz común (Subject)

```
class Graphic {
public:
    virtual void Draw(const Point& position) = 0;
    virtual Point GetExtent() = 0;
    virtual ~Graphic() {}
};
```

Objeto real (RealSubject)

```
class Image : public Graphic {
public:
    Image(const std::string& fileName);

    void Draw(const Point& position) override {
        // C digo para dibujar la imagen real
    }

    Point GetExtent() override {
        // Retorna el tama o real
    }
};
```

Proxy

```
class ImageProxy : public Graphic {
private:
    Image* _image;
    std::string _fileName;
    Point _extent;

public:
    ImageProxy(const std::string& fileName)
        : _image(nullptr), _fileName(fileName), _extent(100, 100) {}

    void Draw(const Point& position) override {
        LoadImage()->Draw(position);
    }

    Point GetExtent() override {
        return _extent; // Valor estimado antes de cargar
    }

private:
    Image* LoadImage() {
        if (!_image) {
            _image = new Image(_fileName);
        }
        return _image;
    }
};
```

Resultado

El cliente usa el objeto ‘Graphic‘, que puede ser un ‘Image‘ o un ‘ImageProxy‘ indistintamente:

```
Graphic* img = new ImageProxy("foto.jpg");

// No carga a n la imagen
Point size = img->GetExtent();

// Ahora s  la carga y la dibuja
img->Draw(Point(50, 100));
```