Structural Pattern: Proxy

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Coming Up

Describing the proxy pattern

- Remote API call

Structure of the proxy pattern

Variations of the proxy pattern

Coming Up

Use cases for this pattern

Pattern consequences

Related patterns

Describing the Proxy Pattern

Creational

Structural

Behavioral

Proxy

The intent of this pattern is to provide a surrogate or placeholder for another object to control access to it

Describing the Proxy Pattern

Accessing a remote API from an application

- “Add service reference” generates a proxy to interact with an API

Proxy is responsible for controlling the actual access to the remote API

- Provides an interface identical to the actual call (although this is sometimes diverted from…)

```
HttpClient client = new();
var response = await client.SendAsync(“api/remotecall”, …);
```

Describing the Proxy Pattern

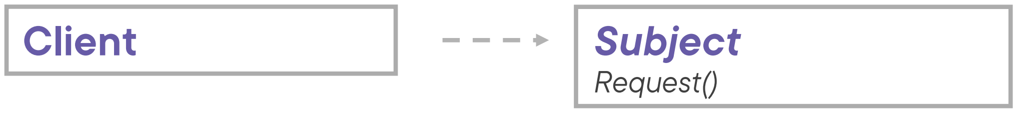
```
HttpClient client = new();
// some code
var response = await client.SendAsync(“api/remotecall”, …);
// more code
```

Describing the Proxy Pattern

Proxy can take on the responsibility of executing code before or after calling the API

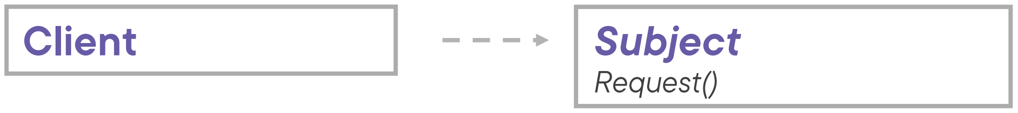
Structure of the Proxy Pattern

Structure of the Proxy Pattern

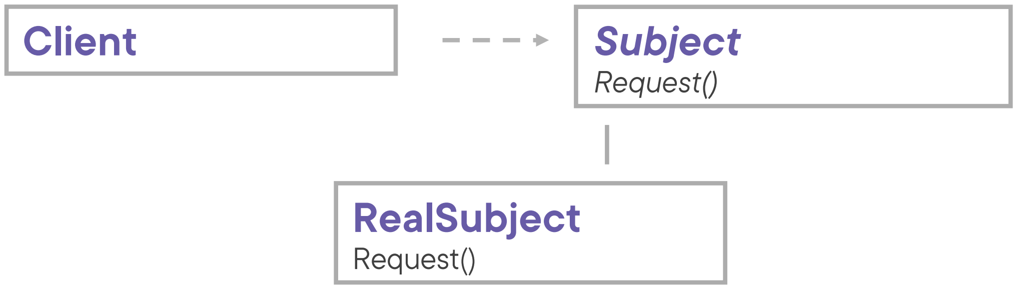


Subject defines the common interface between the **RealSubject** and the **Proxy**

Structure of the Proxy Pattern

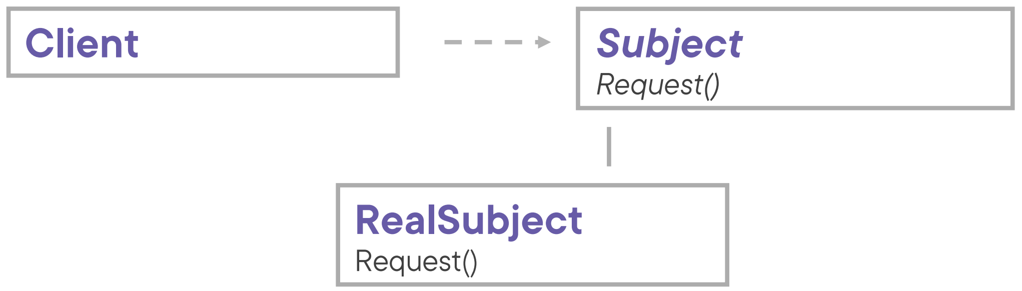


Structure of the Proxy Pattern

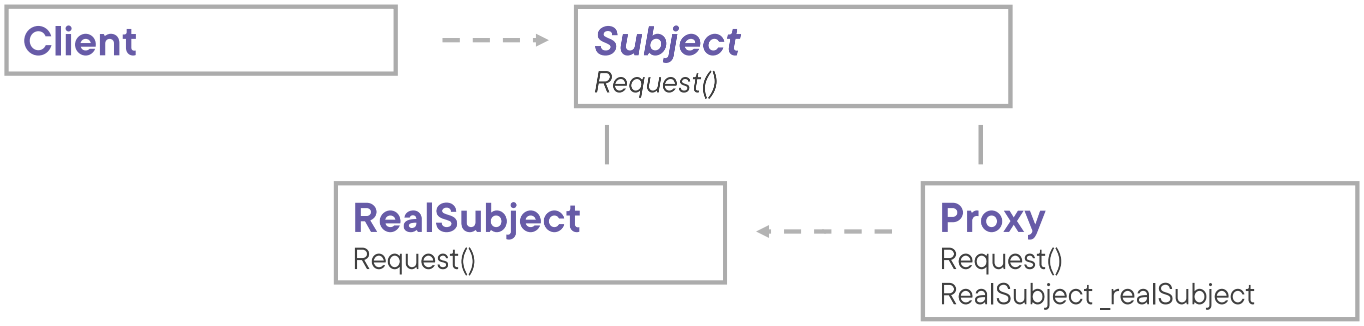


RealSubject defines the real object that the proxy presents

Structure of the Proxy Pattern



Structure of the Proxy Pattern



Proxy provides an interface identical to the **Subject**. It maintains a reference to and controls access to the **RealSubject**.

Variations of the Proxy Pattern

Remote proxy
Client can communicate with the proxy, feels like local resource

Virtual proxy
Allows creating expensive objects on demand

Smart proxy
Allows adding additional logic around the subject

Protection proxy
Used to control access to an object

Variations of the Proxy Pattern

Separation isn’t always clear

- Not all code generators play nice…

Chain proxies when multiple variations are required for your use case

Proxy Pattern Demo Description

Document object scenario

- Virtual proxy to ensure creation only happens when needed
- Protection proxy to add a restriction on who can access the document

Demo

Implementing the proxy pattern

Demo

Chaining proxies

Use Cases for the Proxy Pattern

Use the pattern when you need to add some form of access control to an actual object

Use Cases for Proxy Pattern Variations

Remote proxy
When you want to provide a local representative

Virtual proxy
When you only want to create expensive objects on demand

Smart proxy
When you're in need of a caching or locking scenario

Protection proxy
When objects should have different access rules

Pattern Consequences for Proxy Pattern Variations

Remote proxy
Hides the fact that an object resides in a different network space

Virtual proxy
The object can be created on demand

Smart proxy
Additional housekeeping tasks can be executed when an object is accessed

Protection proxy
Additional housekeeping tasks can be executed when an object is accessed

Pattern Consequences

It allows introducing new proxies without changing the client code: open/closed principle

Added complexity because of additional classes

Performance impact of passing through additional layers

Related Patterns

Adapter
Adapter provides a different interface, proxy provides the same interface

Decorator
Decorator adds responsibilities to an object, while proxy controls access to an object

Summary

Intent of the proxy pattern:

- Provide a surrogate or placeholder for another object to control access to it

Summary

Pattern variations:

- Remote proxy
- Virtual proxy
- Smart proxy
- Protection proxy

Up Next:

Structural Pattern: Flyweight