Lazy Load Pattern

Design Patterns

Steve Smith

SteveSmithBlog.com

Motivating Example

Lazy Load Pattern

- You work with objects that must be loaded from a persistent store (e.g. database)
- You want to avoid loading portions of the object's state that you don't necessarily need
- You don't want the client code to have to know if or when it needs to load additional state – the objects themselves should manage this
- The *lazy load pattern* describes several strategies to achieve these goals.

Intent

Lazy Load Pattern

"A Lazy Load interrupts the object loading process for the moment, leaving a marker in the object structure so that if the data is needed it can be loaded only when it is used."

Patterns of Enterprise Application Architecture

"As many people know, if you're lazy about doing things you'll win when it turns out you don't need to do them at all."

Martin Fowler

Variants: *lazy initialization, virtual proxy, value holder, ghost*

Applicability

Lazy Load Pattern

Use the *Lazy Load* Pattern when:

- Fetching an object requires an extra call for the data, and the data you're fetching isn't used when the main object is used.
- Avoid using it unless or until you need it – use it as a tuning mechanism
- Need to balance amount of data being fetched with number of data requests being made

Lazy Initialization

Lazy Load Pattern

- **Simplest approach**
- **Every property checks to see if its backing field has been initialized**
  - If so, it calculates or fetches the value for the field before returning
  - Violates Don't Repeat Yourself principle (see Principles of OO Design course)
- **Requires all access to the value to go through the property**
  - Within the class, nothing but the property should access the field
- **Requires knowledge of whether the field has been calculated**
  - Typically check if the field is null
  - If null is a legal value, then another approach is required

Demo

Lazy Initialization

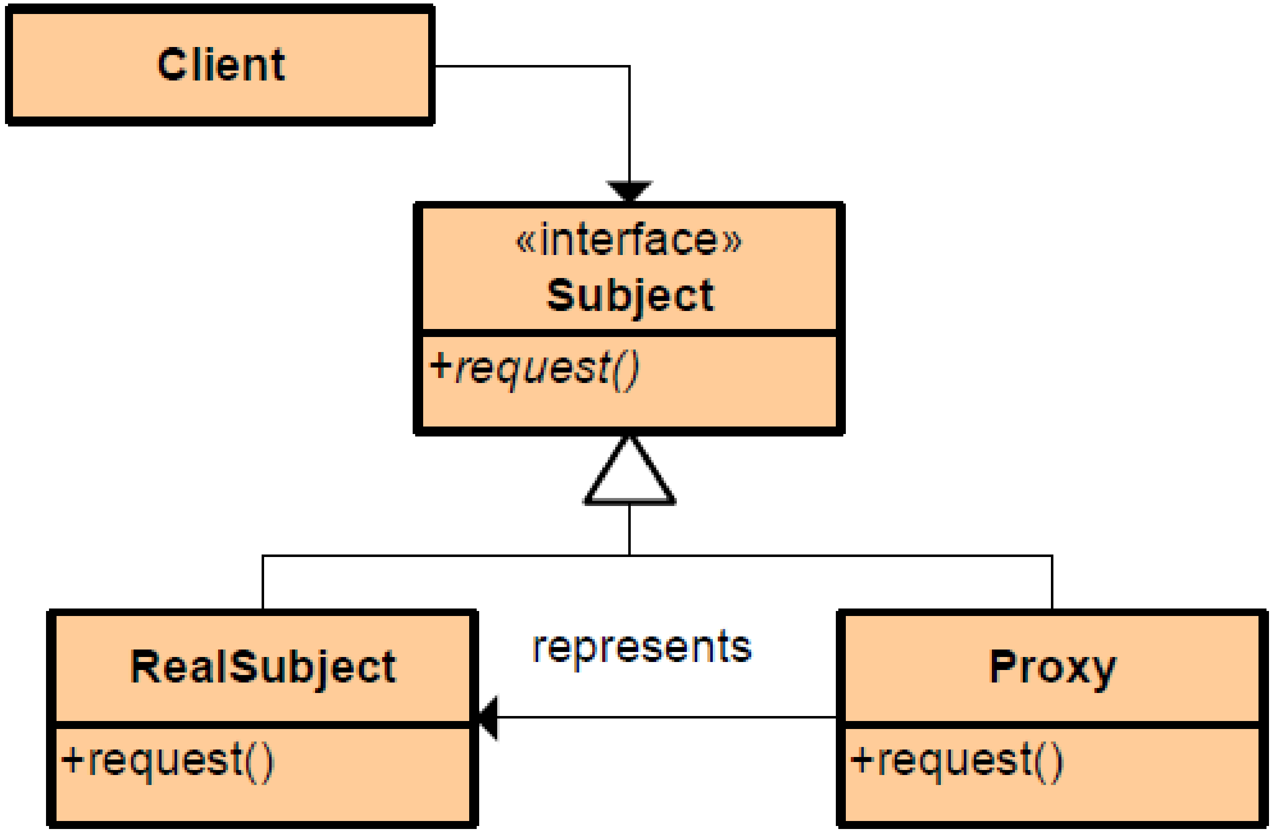
Virtual Proxy

Lazy Load Pattern

- **Proxy looks just like the real object**
- **Learn more about the *Proxy Design Pattern* in the Patterns Library**
- **Can introduce identity problems**
  - Proxy isn't really the object – different object identities and states
  - Override the equality method (Equals())
- **May need to create many virtual proxies**
  - Best done via some kind of code generation
  - Many OR/M tools create *dynamic proxies* at runtime(nHibernate, Entity Framework)

Structure – Virtual Proxy

Lazy Load Pattern



Demo

Virtual Proxy

Virtual Proxy Consequences

Lazy Load Pattern

- **Managing Identity can present challenges**
- **Need to create many virtual proxies, one for each class that is proxied**
- **These problems are not present for *collections***
  - Collections are value objects and do not have an identity
  - Typically there are far fewer collections than objects
  - Collections typically provide largest performance benefit from lazy loading

Value Holder

Lazy Load Pattern

- **Provides lazy load functionality without encapsulation**
- **Calling code knows it is working with a Value Holder type**
- **Requires creating several new types**
  - ValueHolder
  - IValueLoader
  - Factory or Mapper classes
- **ValueHolder uses IValueLoader via *Strategy Pattern* to load value when accessed**
- **ValueHolder and IValueLoader are reusable**

Demo

Value Holder

Ghosts

Lazy Load Pattern

- A *ghost* is a real object in a partial state.
- Initially, the ghost only contains its id.
- Whenever *any* property is accessed, the ghost class loads *all* of its state from the persistence
- Essentially, the object is its own *virtual proxy*
  - Note, this violates the Single Responsibility Principle (see Principles of OO Design course)
- Avoids identity concerns of virtual proxy technique

Demo

Ghosts

How Lazy Loading Is Used

Lazy Load Pattern

- **Depending on implementation client may or may not be aware of whether the object is initialized or not**

- **Typically client access the object as if it were the real object**
  - Object loads its state on demand if needed

- **Consider using Lazy<T> type**
  - Available in .NET 4.0
  - Thread-safe, simple to use method of implementing lazy instantiation on any given type

Consequences

Lazy Load Pattern

- **Can greatly improve performance by eliminating unnecessary calls to data store**

- **Keeping domain objects ignorant of mapping and persistence may require significant code and complexity**

- **Beware of ripple loading when using lazy load**
  - For instance, a collection of objects which are iterated, with each one lazy loaded via a database request for a single row
  - Much more efficient to fetch all rows in a single request

- ***Tune your application to load the data it needs when it needs it, with as few requests to the data source as possible***

Related Patterns

Lazy Load Pattern

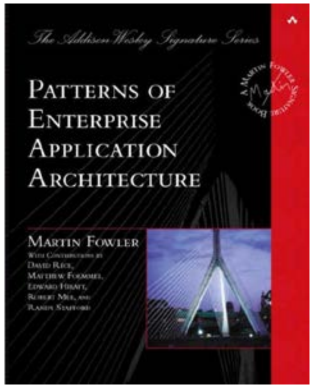
- **Proxy**
  - Lazy Load is often implemented via a *virtual proxy* or other proxy variant
- **Strategy**
  - Used by ValueHolder
- **Template Method**
  - Used by Ghost implementation

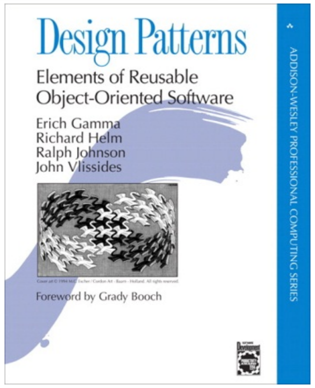
You can learn more about these patterns in the Pattern Library on PluralSight

References

Lazy Load Pattern

- **Books**
  - Patterns of Enterprise Application Architecture
    http://amzn.to/MFPoEAA
  - Design Patterns
    http://amzn.to/95q9ux


The Addison-Wesley Signature Series
Patterns of Enterprise Application Architecture
Martin Fowler


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
Addison-Wesley Professional Computing Series

- **Online**
  - http://www.martinfowler.com/eaaCatalog/lazyLoad.html

Summary

Lazy Load Pattern

- **Apply the Lazy Load pattern to reduce data access calls for unneeded data**

- **Avoid premature optimization – add lazy loading only when performance needs require it**

- **Several approaches, each with different tradeoffs**
  - Lazy Initialization – Simple but Repetitive (not DRY)
  - Virtual Proxy – Effective, but can create identity problems
  - Value Holder – Less abstract and more complex, but no identity problems
  - Ghosts – Less chatty but more complex, and violate SRP

- **Consider simple use of Lazy<T> where needed as a first step when optimizing**

For more in-depth online developer training visit

pluralsight
see what you can learn

on-demand content from authors you trust

Blog: SteveSmithBlog.com
Twitter: @ardalis