.NET Refactoring

Wayne Allen
Corillian

Refactoring – What?

- Making changes to a body of code in order to improve its internal structure, without changing its external behavior.
- Improving internal consistency and clarity
- Remove dead code
- Make it easier for human maintenance in the future

Refactoring – What Not

- Does not fix bugs
- Does not add new functionality

Refactoring – Why?

- Refactoring improves the design.
- What is the business case of good design?
- To me, it's that you can make changes to the software more easily in the future.

Refactoring – Why?

- The corollary is that it's pointless to refactor a system you will never change, because you'll never get a payback.

Refactoring – Why?

- You refactor not because it's fun, but because "there are things you expect to be able to do with your programs if you refactor that you just can't do if you don't refactor."

Refactoring & Unit Tests

- Testing is a very important underpinning to refactoring.
- If you want to refactor, the essential precondition is having solid tests.

Refactoring Catalog (Partial)

- Add Parameter
- Change Bidirectional Association to Unidirectional
- Change Reference to Value
- Change Unidirectional Association to Bidirectional
- Change Value to Reference
- Collapse Hierarchy
- Consolidate Conditional Expression
- Consolidate Duplicate Conditional Fragments
- Convert Dynamic to Static Construction by Gerard M. Davison
- Convert Static to Dynamic Construction by Gerard M. Davison
- Decompose Conditional
- Duplicate Observed Data
- Eliminate Inter-Entity Bean Communication (Link Only)
- Encapsulate Collection
- Encapsulate Downcast
- Encapsulate Field
- Extract Class
- Extract Interface
- Extract Method
- Extract Package by Gerard M. Davison
- Extract Subclass
- Extract Superclass
- Form Template Method
- Hide Delegate
- Hide Method
- Hide presentation tier-specific details from the business tier (Link Only)
- Inline Class
- Inline Method
- Inline Temp
- Introduce A Controller (Link Only)
- Introduce Assertion
- Introduce Business Delegate (Link Only)
- Introduce Explaining Variable
- Introduce Foreign Method
- Introduce Local Extension
- Introduce Null Object
- Introduce Parameter Object
- Move Business Logic to Session (Link Only)
- Move Class by Gerard M. Davison
- Move Field
- Move Method
- Parameterize Method
- Preserve Whole Object
- Pull Up Constructor Body
- Pull Up Field
- Pull Up Method
- Push Down Field
- Push Down Method
- Reduce Scope of Variable by Mats Henricson
- Refactor Architecture by Tiers (Link Only)
- Remove Assignments to Parameters
- Remove Control Flag
- Remove Double Negative by Ashley Frieze and Martin Fowler
- Remove Middle Man
- Remove Parameter
- Remove Setting Method
- Rename Method
- Replace Array with Object
- Replace Assignment with Initialization by Mats Henricson
- Replace Conditional with Polymorphism
- Replace Conditional with Visitor by Ivan Mitrovic
- Replace Constructor with Factory Method
- Replace Data Value with Object
- Replace Delegation with Inheritance
- Replace Error Code with Exception
- Replace Exception with Test
- Replace Inheritance with Delegation
- Replace Iteration with Recursion by Dave Whipp
- Replace Magic Number with Symbolic Constant
- Replace Method with Method Object
- Replace Nested Conditional with Guard Clauses
- Replace Parameter with Explicit Methods
- Replace Parameter with Method
- Replace Record with Data Class

Rename Method

- The name of a method does not reveal its purpose.
- Change the name of the method.
- Very common

Extract Method

- Turn fragments into a method whose name explains the purpose of the method.
- Turn large, unmanageable methods into a series of small, logical and functional methods.
- Reduce duplication
- Very common
- **Inverse refactoring:** Inline Method

Extract Method - How

- “Quick & Dirty” Version
- “By the Book” Version
- Resharper
- VS.NET 2005

Introduce Null Object

- You have repeated checks for a null value.
- Replace the null value with a null object.

Split Loop

- You have a loop that is doing two things
- Duplicate the loop

Resources

- *Refactoring*, Martin Fowler
- http://www.refactoring.com
- http://en.wikipedia.org/wiki/Refactoring

Refactoring Problems

- Not doing it enough (code hygiene)
- Refactoring in batches
- Refactoring for “fun”
- Meaningless style changes (m_Var vs. _var vs. var)
- Lack of tests
- Shared libraries (refactoring space)

3rd Party Tools

- Resharper (JetBrains) http://www.jetbrains.com/resharper/
- C# Refactory (XtremeSimplicity) http://www.xtreme-simplicity.net
- Refactor!™ Pro (Developer Express) http://www.devexpress.com/Products/NET/IDETools/
- JustCode! (Omnicore) http://www.omnicore.com/justcode.htm

3rd Party Tools

- Code Explorer (ModelMaker Tools) http://www.modelmakertools.com/code-explorer-vs/in
- devAdvantage (Anticipating Minds) http://www.anticipatingminds.com/Content/Products/d
- .NET Refactor (Know Dot Net) http://knowdotnet.com/articles/netrefactorproducthome.htm

Questions?

Thanks!

wallen@corillian.com
weblogs.asp.net/wallen