Higher Order Messaging (HOM)

Marcel Weiher

www.metaobject.com

Overview

- Intro
- Motivation/Development
- Usage and Implementation
- Outlook

HOM in a Nutshell

- Message taking a Message as an Argument
- Example

```
[[stringArray collect]
  stringByAppendingString:@"suffix"];
```

(this returns the result of sending the message "stringByAppendingString" with the argument @"suffix" to each of theobjects in stringArray)

Motivation

- Pattern avoidance (laziness)
- The "loop" pattern

```
id nextObject;
id enumerator=[array objectEnumerator];
id result=[NSArray array];
while ( nil!=(nextObject=[enumerator nextObject] ) {
      [result addObject:[nextObject
            stringByAppendingString:@"suffix"]];
}
```

(most of this code is unchanging, it gets repeated in every 'invocation' of the pattern. We *should* be able to write it once and then 'call' it with the arguments given in **bold**)

Inspiration

- Iteration in Smalltalk, ST-72
- **forall** in Postscript
- Functors in FP (HOF)
- Unix Pipes and Filters
- NSEnumerator

Perspiration

- makeObjectsPerformSelector:
- Filter-Enumerators: enumerator-based, stackable, practical

```
result=[array filterWithSelector:
          @selector(stringByAppendingString:)
          withObject:@"suffix" withObjet:nil];
```

Problems

- proliferation of filter-methods
- dispersed message-send

```
result=[array filterWithSelector:
          @selector(stringByAppendingString:)
          withObject:@"suffix" withObjet:nil];
```

(the parts in **bold** together form a message send)

- order obscured

```
result=[array filterArgumentWithSelector:
          @selector(stringByAppendingString:)
          withTarget:@"prefix" withObjet:nil];
```

(@"prefix" is the receiver of the dispersed message send,
array is the argument, but the order is reversed in code)

Insight

- use existing syntax

```
result=[object stringByAppendingString:@"suffix"];
```

- generalize message send
- use runtime as 'parser'

HOM

- prefix message (the HOM)

```
[[array collect] stringByAppendingString:@"suffix"];
```

- argument-message

```
[[array collect] stringByAppendingString:@"suffix"];
```

Usage

- HOMs:

  `do,collect,select,selectWhere:,reduce`

- Arbitrary multiple iterations

```
[[prefixes collect] stringByAppendingString:@"suffix"];
```

(iterates through receiver array, single argument)

```
[[@"prefix" collect] stringByAppendingString:[suffixes each]];
```

(single receiver, iterates through argument array)

```
[[prefixes collect] stringByAppendingString:[suffixes each]];
```

(iterates through both receiver array and argument array in lock-step)

Advantages

- Saves code
- Declarative, intentional style
- Chunk more, do more
- Easy conceptual model
- Works across bridges

Implementation

- Trampolines
- Filter Enumerators
- Optimization

Trampolines

- NSInvocation
- forwardInvocation:
- Target + selector to bounce

Filter-Enumerators

- Setup
- nextObject
- Results Processing
- Running the Iteration

Optimization

- Aggressive IMP-caching
- Encapsulated/Localized
- Compensate Overhead

Issues

- Overhead
- Obfuscation
- Portability
- **select** and calling conventions

Outlook

- Expose stackability
- Optimize implementation
- Eliminate more patterns
- DBs, Threads, Futures,
- Idea of message transport

Availability

- Part of MPWFoundation
- unit tests, heavy use, > 4 years

  (PS/PDF Interpreter, Workflow-Engine, etc.)
- Free for the asking

  www.metaobject.com

Summary

- EnumFilters eliminate loops
- Obvious interface via HOM
- Very general technique
- Tested, free:
  www.metaobject.com