CS 683 Emerging Technologies
Spring Semester, 2003
Doc 6 AspectS

Contents

AspectS .................................................................................................................. 2
Advise supported ................................................................................................. 3
Types of Pointcuts .............................................................................................. 4

References

http://www.prakinf.tu-ilmenau.de/~hirsch/Projects/Squeak/AspectS/

2003 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA.
OpenContent (http://www.opencontent.org/pl.shtml) license defines the copyright on this
document.
AspectS

Aspect-oriented programming in Smalltalk

• No language extension used
• No precompiler used
• No rearranging of source code
• Aspects can be turned on & off dynamically
Advise supported

• Handler
  Advice for dealing with exceptions

• Before-After
  Advice run before and/or after a method

• Around
  Advice that can bypass a method call

• Introduction
  Introduces new behavior

All advice as access to
• Receiver
• Sender
• Method arguments
Types of Pointcuts

• Receiver Class Specific

All receivers of the message that are an instance of a certain class are affected.

• Receiver Instance Specific

Only specific receivers of the message that are an instance of a certain class are affected.

• Sender Class Specific

Receivers of the message that are an instance of a certain class are going to be affected if the sender is of a certain class or its subclasses.

• Sender Instance Specific

Receivers of the message that are an instance of a certain class are going to be affected only if the sender is known to the advice.
Cflow Pointcuts

- Class First
- Class All-But-First
- Instance First
- Instance All-But-First
- Super First
- Super All-But-First
Examples

Hello class

Smalltalk.AspectS defineClass: #Hello
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: "
   classInstanceVariableNames: "
   imports: "
   category: 'AspectS-Examples Counter'

hello
   Transcript
      show: 'Hello';
      cr
HelloAspect

Smalltalk.AspectS defineClass: #HelloAspect
  superclass: #{AspectS.AsAspect}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: "
  category: 'AspectS-Examples Counter'

adviceAnnounceBefore
  ^AsBeforeAfterAdvice
  qualifier: (AsAdviceQualifier attributes: #(#receiverClassSpecific))
  pointcut:
    [OrderedCollection
      with: (AsJoinPointDescriptor targetClass: Hello targetSelector: #hello)]
  beforeBlock:
    [:receiver :arguments :aspect :client |
      Transcript
      show: 'Before';
      cr]
  afterBlock:
    [:receiver :arguments :aspect :client :return |
      Transcript
      show: 'After';
      cr]
Test Program

| greeter aspect |
| greeter := Hello new. |
| aspect := HelloAspect new. |
| greeter hello. |
| aspect install. |
| greeter hello. |
| aspect uninstall. |
| greeter hello |

Output In Transcript

Hello
Before
Hello
After
Hello
Instance Specific Advice

Smalltalk.AspectS defineClass: #HelloAspect

adviceAnnounceBefore
  ^AsBeforeAfterAdvice
  qualifier: (AsAdviceQualifier attributes: #(#receiverInstanceSpecific))
  pointcut:
    [OrderedCollection
      with: (AsJoinPointDescriptor targetClass: Hello targetSelector: #hello)]
  beforeBlock:
    [:receiver :arguments :aspect :client |
      Transcript
      show: 'Before';
      cr]
  afterBlock:
    [:receiver :arguments :aspect :client :return |
      Transcript
      show: 'After';
      cr]
Test Program

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>a := Hello new.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b := Hello new.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspect := HelloAspect new.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspect addReceiver: a.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspect install.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b hello.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transcript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>show: 'End b';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cr.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a hello.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspect uninstall.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output

Hello
End b
Before
Hello
After
Recursive Example

Method Added to Integer Class

factorial2
"Answer the factorial of the receiver. Object-recursive."

self = 0 ifTrue: [^ 1].
self > 0 ifTrue: [^ self * (self - 1) factorial2].
self error: 'Not valid for negative integers'.

Method in AsFactorialTraceAspect Class

adviceFactorialInOutFirst

^[AsBeforeAfterAdvice
 qualifier: (AsAdviceQualifier attributes: #(receiverClassSpecific cfFirstClass))
 pointcut: [OrderedCollection
 with:
   (AsJoinPointDescriptor targetClass: Integer targetSelector: #factorial2)]
 beforeBlock: [:receiver :arguments :aspect :client |
   Transcript
   show: '#factorial-in: ', receiver printString;
   cr]
 afterBlock: [:receiver :arguments :aspect :client :return |
   Transcript
   show: '#factorial-out(', receiver printString, '):', return printString;
   cr]
Test Program

aspect := AsFactorialTraceAspect new.
aspect install.
4 factorial2.
aspect uninstall

Output

#factorial-in: 4
#factorial-in: 3
#factorial-in: 2
#factorial-in: 1
#factorial-in: 0
#factorial-out(0): 1
#factorial-out(1): 1
#factorial-out(2): 2
#factorial-out(3): 6
#factorial-out(4): 24
Cflow Example

**Method in AsFactorialTraceAspect Class**

```smalltalk
adviceFactorialInOutFirst

^ AsBeforeAfterAdvice
  qualifier: (AsAdviceQualifier attributes: #(receiverClassSpecific cfFirstClass))
  pointcut: [OrderedCollection
    with: [(AsJoinPointDescriptor targetClass: Integer targetSelector: #factorial2)]
  beforeBlock: [:receiver :arguments :aspect :client |
    Transcript
    show: '#factorial-in: ', receiver printString;
    cr]
  afterBlock: [:receiver :arguments :aspect :client :return |
    Transcript
    show: '#factorial-out(', receiver printString, ': ', return printString; cr]

Test Program

| aspect |
aspect := AsFactorialTraceAspect new.
aspect install.
4 factorial2.
aspect uninstall

Output

#factorial-in: 4
#factorial-out(4): 24