

aspect OutcomeVotable depends on OutcomeAware, Terminatable

structural view

IOutcomeVotableParticipant

OutcomeVotableContext

~ OutcomeVotableContext create()
+ pauseContext()
+ continueContext()
+ terminateContext(IOutcomeVotableParticipant)
+ registerVote(Outcome, IOutcomeControllingParticipant)
+ boolean alreadyVoted(IOutcomeVotableParticipant)
- Outcome computeOutcome()
- boolean isOutcomeDecided()

votes 1
1 IOutcomeVotableParticipant,
Outcome

Map

~ Map<OutcomeVotableParticipant, Outcome> create()
~ put(IOutcomeVotableParticipant, Outcome)
~ boolean containsKey(IOutcomeVotableParticipant)
~ Collection<Outcome> values()

IOutcomeVotableParticipant

+ leaveContext()
+ voteAndLeaveContext(Outcome)

OutcomeAware instantiation

IOutcomeControllingParticipant →
IOutcomeVotableParticipant

Terminatable instantiation

ITerminatableParticipant →
IOutcomeVotableParticipant

Terminatable binding

OutcomeVotableContext →
TerminatableContext

leaveContext → leaveContext

pauseContext → pauseContext

continueContext → continueContext

terminateContext → terminateContext

OutcomeAware binding

OutcomeVotableContext →
ContextWithOutcome

message view voteAndLeaveContext affected by OutcomeAware.getContext, Terminatable.leaveContext

Pointcut

caller: Caller

target:
IOutcomeVotable
Participant

voteAndLeaveContext(v)

Advice

caller: Caller

target:
IOutcomeVotable
Participant

voteAndLeaveContext(v)

Default
Instantiation
caller → *
Caller → *
target → *

myContext :=
getContext()

myContext:
OutcomeVotable
Context

registerVote(v, target)

leaveContext()

message view registerVote affected by OutcomeAware.setContext, Terminatable.terminateContext

Pointcut

caller: Caller

target:
IOutcomeVotable
Context

registerVote(v, p)

Advice

caller: Caller

target:
IOutcomeVotable
Context

registerVote(v, p)

votes:
Map<OutcomeVotable
Participant, Outcome>

votes.put(v,p)

decided :=
isOutcomeDecided()

opt [decided] /
outcome :=
computeOutcome()

setOutcome(outcome)

terminateContext()

Default
Instantiation
caller → *
Caller → *
target → *

message view alreadyVoted

Pointcut

caller: Caller

target:
IOutcomeVotable
Context

voted :=
alreadyVoted(p)

Advice

caller: Caller

target:
IOutcomeVotable
Context

voted :=
alreadyVoted(p)

votes:
Map<OutcomeVotable
Participant, Outcome>

containsKey(p)

Default Instantiation
caller → *, Caller → *, target → *

message view computeOutcome affected by OutcomeAware.BinaryOutcome.create

Advice

Pointcut

caller: Caller

target:
IOutcomeVotable
Context

outcome :=
computeOutcome()

caller: Caller

target:
IOutcomeVotable
Context

outcome :=
computeOutcome()

votes:
Map<OutcomeVotable
Participant, Outcome>

outcome := create(true)
votes := values()

outcome:
BinaryOutcome

loop [v in votes]

v:
Outcome

positive := isPositive()

opt [not positive]

outcome := create(false)

outcome:
BinaryOutcome

Default
Instantiation
caller → *
Caller → *
target → *