This sample demonstrates how can Interceptors be used to block or filter calls between components dynamically, and change the result, based on the values passed as arguments or based on the return value. It only adds a few new elements to the Interceptor sample.

Project name: "L.CallFiltering"

For information on how to get the code, and run the sample, please see About Basic Samples .

Description

This sample is built by modifying the previous sample and adding another interceptor to it to demonstrate the call filtering. Otherwise, it behaves and works the same way.

Similar to LogToConsoleListener and LogToConsoleInterceptor classes, two more classes are added to this project's code called BigNumberFilterListener and BigNumberFilterInterceptor.

The new interceptor requires to perform following tasks:

- If the caller tries to add two numbers, that are both larger than 10, the call should be blocked. (Filtered based on the method input)
- If the caller tries to multiply two numbers and the result is larger than 100, the call should be blocked. (Filtered based on the method result)
- Otherwise, the method call should go through.
- In any blocked call, the blocked method should return "-1" as the result.

For the first requirement, BeforeCall method of the interceptor is used to implement the logic, because arguments are available to the interceptor before performing the actual call. But for the second requirement, the return value is not available in BeforeCall method. So, AfterCall is used to check if the result is small enough to return to the caller.

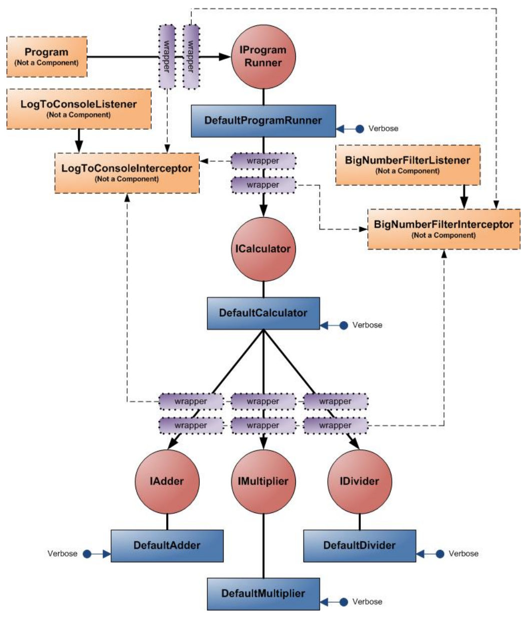
BigNumberFilterListener wraps all components in dynamically created components, similar to LogToConsoleListener, to allow connecting them to the interceptor. This listener is similarly registered in CalculatorComposition.xml file.

Note:

- Blocking the Multiply call when the result is greater than 100, actually means blocking the result from getting to the caller. Since the logic is implemented in the AfterCall, the method call is performed.
- Boh BeforeCall and AfterCall change the result of the method call to "-1" when blocking.
- Setting callInfo.Completed flag in the BeforeCall prevents the method call to reach the wrapped component.

The output of the sample shows that performing 67 + 12 calculation is blocked, as both numbers are greater than 10, and 67 * 12 is blocked, since the result (804) is greater than 100. In these cases, -1 is returned to the program runner.

Dependency Diagram



Sample Output

```
 LISTENER     - LogToConsoleListener.OnComponentRetrieved(IClassEmitter)
LISTENER     - LogToConsoleListener.OnComponentComposed(<null>)
CONSTRUCTOR  - DefaultProgramRunner
LISTENER     - LogToConsoleListener.OnComponentCreated(IProgramRunner)
           - Wrapping component for logging.
LISTENER     - BigNumberFilterListener.OnComponentCreated(IProgramRunner)
           - Wrapping component for filtering big numbers.
CONSTRUCTOR  - DefaultCalculator
LISTENER     - LogToConsoleListener.OnComponentCreated(ICalculator)
           - Wrapping component for logging.
LISTENER     - BigNumberFilterListener.OnComponentCreated(ICalculator)
           - Wrapping component for filtering big numbers.
CONSTRUCTOR  - DefaultAdder
LISTENER     - LogToConsoleListener.OnComponentCreated(IAdder)
           - Wrapping component for logging.
LISTENER     - BigNumberFilterListener.OnComponentCreated(IAdder)
           - Wrapping component for filtering big numbers.
SET CONFIG   - DefaultAdder.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(IAdder)
NOTIFICATION - DefaultAdder: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(IAdder)
SET PLUG     - DefaultCalculator.Adder(IAdder7)
CONSTRUCTOR  - DefaultMultiplier
LISTENER     - LogToConsoleListener.OnComponentCreated(IMultiplier)
           - Wrapping component for logging.
LISTENER     - BigNumberFilterListener.OnComponentCreated(IMultiplier)
           - Wrapping component for filtering big numbers.
SET CONFIG   - DefaultMultiplier.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(IMultiplier)
NOTIFICATION - DefaultMultiplier: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(IMultiplier)
SET PLUG     - DefaultCalculator.Multiplier(IMultiplier9)
CONSTRUCTOR  - DefaultDivider
LISTENER     - LogToConsoleListener.OnComponentCreated(IDivider)
           - Wrapping component for logging.
LISTENER     - BigNumberFilterListener.OnComponentCreated(IDivider)
           - Wrapping component for filtering big numbers.
SET CONFIG   - DefaultDivider.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(IDivider)
NOTIFICATION - DefaultDivider: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(IDivider)
SET PLUG     - DefaultCalculator.Divider(IDivider11)
SET CONFIG   - DefaultCalculator.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(ICalculator)
```

```
NOTIFICATION - DefaultCalculator: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(ICalculator)
SET PLUG     - DefaultProgramRunner.Calculator(ICalculator5)
SET CONFIG   - DefaultProgramRunner.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(IProgramRunner)
NOTIFICATION - DefaultProgramRunner: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(IProgramRunner)
BEFORE       - IProgramRunner.Run()

METHOD CALL  - DefaultProgramRunner.Run()

BEFORE       - ICalculator.Add(67, 12)
METHOD CALL  - DefaultCalculator.Add(67, 12)
BLOCKED      - Not authorized to add big numbers.
AFTER        - ICalculator.Add(67, 12) -> -1
67 + 12 = -1

BEFORE       - ICalculator.Subtract(67, 12)
METHOD CALL  - DefaultCalculator.Subtract(67, 12)
ALLOWED      - Authorized to add small numbers.
BEFORE       - IAdder.Add(67, -12)
METHOD CALL  - DefaultAdder.Add(67, -12)
AFTER        - IAdder.Add(67, -12) -> 55
AFTER        - ICalculator.Subtract(67, 12) -> 55
67 - 12 = 55

BEFORE       - ICalculator.Multiply(67, 12)
METHOD CALL  - DefaultCalculator.Multiply(67, 12)
BEFORE       - IMultiplier.Multiply(67, 12)
METHOD CALL  - DefaultMultiplier.Multiply(67, 12)
AFTER        - IMultiplier.Multiply(67, 12) -> 804
BLOCKED      - Result is too large.
AFTER        - ICalculator.Multiply(67, 12) -> -1
67 * 12 = -1

BEFORE       - ICalculator.Divide(67, 12)
METHOD CALL  - DefaultCalculator.Divide(67, 12)
BEFORE       - IDivider.Divide(67, 12)
METHOD CALL  - DefaultDivider.Divide(67, 12)
AFTER        - IDivider.Divide(67, 12) -> 5
AFTER        - ICalculator.Divide(67, 12) -> 5
BEFORE       - ICalculator.Remainder(67, 12)
METHOD CALL  - DefaultCalculator.Remainder(67, 12)
BEFORE       - IDivider.Remainder(67, 12)
METHOD CALL  - DefaultDivider.Remainder(67, 12)
AFTER        - IDivider.Remainder(67, 12) -> 7
AFTER       - ICalculator.Remainder(67, 12) -> 7
67 / 12 = 5 (with remainder = 7)

AFTER       - IProgramRunner.Run() -> <null>
```