This sample demonstrates how a custom component factory can be implemented and added to the Composer. It is based on the Call Filtering sample.

Project name: "M.CustomFactory"

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

Description

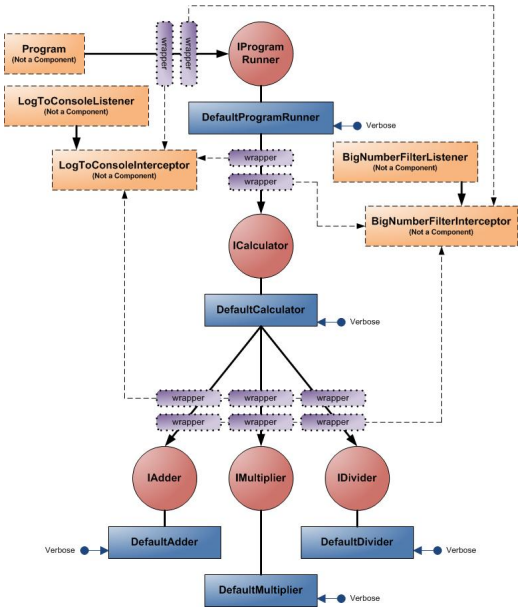
This sample runs and functionally behaves exactly similar to the previous sample , but for one of the components (DefaultCalculator) it uses a custom factory instead of using default factories of Composer.

The class CalculatorFactory implements IComponentFactory interface, and implements a custom way that a component should be instantiated:

- In the GetContracts method, it returns an array of the contracts provided by the factory (ICalculator).
- In the GetComponentInstance method, it instantiates, initializes and returns an instance of the DefaultCalculator component. It also shows how you can use Composer to get references to other components for initializing the component instance in hand.

The CalculatorFactory class is registered in the component context in the applications Main method (Program class).

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
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(IAdder5)
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(IMultiplier7)
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(IDivider9)
SET PLUG     - DefaultProgramRunner.Calculator(DefaultCalculator)
SET CONFIG   - DefaultProgramRunner.Verbose(True)
LISTENER     - LogToConsoleListener.OnComponentComposed(IProgramRunner)
NOTIFICATION - DefaultProgramRunner: OnCompositionComplete.
LISTENER     - LogToConsoleListener.OnComponentRetrieved(IProgramRunner)
BEFORE       - IProgramRunner.Run()

METHOD CALL  - DefaultProgramRunner.Run()

BLOCKED      - Not authorized to add big numbers.
67 + 12 = -1

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

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

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

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