POLITECNICO
MILANO 1863

A.Y 2015-2016
Software Engineering 2

Code Inspection Document
"Glassfish 4.1"
Version 1.0

Massimiliano Paci (mat. 852720)
Giovanni Patruno (mat. 852658)

January 5th 2016

Contents

1 Introduction

This document contains the code inspection of one file and some method of the Glassfish Appserver 4.1.1 source code. We focused on specific aspects of the code such as the elegance and the order that the developer showed, avoiding the *brutish programming*.
We will not provide ways to improve these aspects rather we just highlight the points that are not in line with an assigned inspection checklist.

2 Assigned methods

We were assigned four methods of the *InterceptorManager* class to ispect(located at: appserver/ejb/ejb-container/src/main/java/com/sun/ejb/containers/interceptors/InterceptorManager.java) :

- buildInterceptorChain(InterceptorInfo interceptorInfo)
  line: 453
- initEjbCallbackIndices()
  line: 513
- initCallbackIndices(List < InterceptorDescriptor > callbackList , CallbackType callbackType)
  line: 592
- createCallbackInterceptors(CallbackType eventType , InterceptorDescriptor inter , ClassLoader classLoaderToUse)
  line: 634

3 Functional Role of the assigned methods

We tried to understand the aim of these methods with the Javadoc associated, but it was not particularly meaningful and so we had to do an accurate reading of the source code and general java EE documentation.

Interceptors are used to allow developers to invoke interceptor methods on an associated target class, in conjunction with method invocations or lifecycle events simulating, in a certain way, a client-server interaction. Common uses of interceptors are logging, auditing, and profiling.
An interceptor can be defined within a target class as an interceptor method, or in an associated class.
Our assigned class (InterceptorManager) should manage all the logic to allow developers to create interceptors and recieve callbacks for lifecycle events (post-create or pre-destroy of a target class).

3.1 buildInterceptorChain(InterceptorInfo interceptorInfo)

This method initializes all the interceptors and the callback indices. It manages to find the target classes from the interceptorInfo. For the event types: AROUND-CONSTRUCT, POST-CONSTRUCT and PRE-DESTROY initializes the callback indices.

3.2 initEjbCallbackIndices()

This method initializes all the Ejb (Enterprise Java Bean) callback indices also scanning and loading lifecycle methods.

3.3 initCallbackIndices(List < InterceptorDescriptor > callbackList, CallbackType callbackType)

This method initialize all the callback indices of a certain type.

3.4 createCallbackInterceptors(CallbackType eventType , InterceptorDescriptor inter , ClassLoader classLoaderToUse)

This method retrieves the callback descriptors from the interceptor descriptor taken in input than creates the callback interceptor for each target method that can be either a bean callback interceptor or a simply callback interceptor and then add the interceptor in a list returned by the method.

4 Issues

In this section we want to give a precise description of the file and method issues, we are referring to the checklist issues points in the *Assignment 3.pdf*. The checklist points are highlited in red and the referring row in green.
All the checklist point omitted are respected or there are no correlations with the code in analysis.

4.1 File issues

In this section we will show the general issues relative of the whole file under inspection

Naming Conventions

- [6] Class variables don't follow a specific convention. e.g. the variable in line 93 begin with an underscore ('_') and the others simply with a lowercase letter.

Braces

- [10] : in all the file is used the "Kernighan and Ritchie" bracing style

Java Source Files

- [23] The class doesn't have a meaningful associated Javadoc
- The file contains too much classes (five). It could be correct because there is only one public class and is the first of the file, but in a file of more then 1000 lines is better to reduce the number of classes.

Class and Interface Declarations

- [25.E] line 113 should be before the private variables
- [27] The InterceptorManager class is too big and with too much methods
- [27] Methods too much long such as "getAroundInvokeChain" (line 213) of 53 lines

4.2 buildInterceptorChain issues

Naming Convention

- [1] [465] : the variable *clazz* has a confusing name
- [1] [475] : the method *getHasTargetClassAroundInvoke()* has a confusing name
- [1] [477] : to what does the variable *size* refer to?

File Organization

- [13] [480-482] : this three lines exceed 80 characters (around 100)

Comments

- [18] Not enough comments to explain the scope of the method

Initialization and Declaration

- [33] [436] : at each repetition of the for-loop is instantiated the same object (clazz) and is not substituted (the declaration should be at the beginning of the block)
- [33] [477] : the variable *size* should be initialized at the beginning of the method

Computation, Comparisons and Assignments

- [46] [474] : it's not used a specific parenthesis convention; in this statement the first condition is between parenthesis and the second one is without.

Exceptions

- Bad exception managing: the exception spread up too much and is difficult to know the source

4.3 initEjbCallbackIndices issues

Naming Convention

- [1] [516] : to what does the variable *size* refer to?
- [1] [518] : the boolean variable *scanFor2xLifecycleMethods* has a confusing name and appears to be a method (starts with a verb)
- [1] [521] : to what does the variable *index* refer to?

Wrapping Lines

- [15] [585] : line break not clear: it's better after a comma or an operator

Java Source Files

- [23] : the method is not present in the Javadoc

Class and Interface Declarations

- [26] : The method is so long because is thought by scope rather than by functionality

Object Comparison

- [40] [525, 526] : comparison with "==" instead of "equals"

Computation, Comparisons and Assignments

- [44] [524, 531] : created a useless Boolean variable instead of a simple Boolean expression

4.4 initCallbackIndices issues

Naming Convention

- [1] [597] : to what does the variable *index* refer to?
- [1] [601] : the variable *interceptorClass* has a confusing name
- [1] [604] : the variable *inters* has a meaningless name

Wrapping Lines

- [15] [622] : line break after a cast is confusing
- [17] [605] : new line not aligned with the beginning of the others parameters

Comments

- [18] : method and code blocks totally free of comments
- [19] [619] : the only commented out code doesn't explain anything

Java Source Files

- [23] : the method is not present in the Javadoc

Initialization and Declaration

- [31] [597] : *index* initialized at the beginning and not used after the initialization
- [33] [601-605] : *interceptorClass, classLoaderToUse* and *inters* should be declared at the beginning of the for-loop

Object Comparison

- [40] [602] : *null* and *interceptorClass* compared with "!="

Computation, Comparisons and Assignments

- [44] [599] : The method is thought by scope rather than by functionality (it should be invoked another method to add the interceptors)

4.5 createCallbackInterceptors issues

Naming Convention

- [1] [635] : the variable *inter* has a confusing name
- [1] [659] : to what does the variable *index* refer to?

Wrapping Lines

- [15] [650] : line break should occur after the "+" operator

Comments

- [18] : method and code blocks totally free of comments

Java Source Files

- [23] : the method is not present in the Javadoc

Initialization and Declaration

- [31] [637] : *callbackList* initialized at the beginning of the method and used only in a for-loop at the end of the method

Object Comparison

- [40] [659] : comparison with "==" instead of "equals"

5 Appendix

5.1 Working Hours

The Code Inspection Document was written in more or less 20 hours divided between the two elements of the group:

- Massimiliano Paci: 10 hours
- Giovanni Patruno: 10 hours

Each element of the group didn't work on specific sections but worked on the whole document.