

DAJ User's Guide

by John J. Sung

Introduction

The integration between AspectJ and Demeter concepts came about as my Master's Thesis topic. We wanted to gain a better understanding of how the concepts applied in AspectJ and Demeter would interact. Importantly, we wanted to understand the fundamental concepts applied and their interactions.

These concepts and their interactions are important in analyzing the tools based on these concepts and in planning a directed development of these tools. In this analysis, we should be able to determine the source of the expressive power of the tools, types of problems it solves and potential complications.

Installation

Requirements

DAJ was developed with and compatible with java, aspectj and demeterj. The specific versions used are listed below.

- java sdk 1.3.0 or later from www.java.sun.com
- aspectj 1.0 or later from www.aspectj.org
- demeterj 0.8.4 or later from www.ccs.neu.edu/research/demeter/DemeterJava

Downloading and unpacking DAJ

You may download the latest version of DAJ from <http://www.ccs.neu.edu/home/jsr/daj/>. You should use your browser to download the jar file. Then use the "jar xf daj[version].jar" to unpack the DAJ in the desired location. It will create a directory structure that looks like this:

```
daj[version]
  daj.jar
  basket
    BasketMain.java
    BasketMainCount.java
    BasketTraversal.trv
  ccg
    CreateClassGraph.java
```

For unix workstations, you might need to unpack the daj.jar file, because ajc might not be able to load the .jar file.

```
jar xf daj.jar
```

Setting the Environment

You need to add these paths to your class path:

[fullpath]/daj.jar or if unpacked: [fullpath]/daj[version]

You also need to make sure that your environment is setup for java, aspectj, and demeterj.

Running DAJ

To run DAJ you need to pass DAJ as the class to execute the main method. DAJ requires 2 options, -ccg and -main. -ccg is the option to specify the location of the CreateClassGraph.java file in the ccg directory. This file is compiled with the user code to generate the implementation of the traversals specified within the trv files. The -main option specifies the class that contains the main method for the user code. This class is passed to the java vm in the traversal generation process.

The DAJ command line specification:

```
java DAJ -ccg CreateClassGraph.java -main Main [options] [java files] [trv files]
```

java files - files with .java extension that contains AspectJ code or pure java code.

trv files - files with .trv extension that contains the traversal specification. See Traversal Files Format section for syntax and semantics of the traversal specification.

Required

-ccg [java location] - Location of the CreateClassGraph.java file

-main [class name] - The class name of the class that contains the static main method

Optional

-gpa - This option generates printing advice for each of the traversals that outputs a trace of the calls to the traversal methods.

-dtrv [directory path] - Outputs the generated traversal files to the specified directory.

-ajc "ajc compiler" - The string containing the actual command line for the ajc compiler. This is used for using other than "ajc" for the compilation.

Compiling and Running the Basket Example

This example run of the basket example is for windows.

```
>cd basket
```

```
>basket.bat
```

```
Compiling Basket Example
```

```
%I - Generating Stubs
```

```
prefix for stub: trav
```

```
Generating Stub file: trav/BasketTraversal.java
```

```
%I - traversal generation compilation
```

```
ajc.bat d:\development\daj\dev\ccg\CreateClassGraph.java  
BasketMain.java BasketMainCount.java trav/BasketTraversal.java
```

```
%I - traversal generation
```

```
java BasketMain -gpa -d trav BasketTraversal.trv
```

```
%I - traversal compilation
```

```
ajc.bat BasketMain.java BasketMainCount.java  
trav/BasketTraversal.java
```

```
Running the Basket Example
```

```
call(void Basket.t2())
```

```
call(void Basket.t2_crossing_f())
```

```
call(void Fruit.t2())
```

```
call(void Fruit.t2_crossing_w())
```

```
call(void Weight.t2())
```

```
Total weight of basket = 5
```

Traversal File Format

The traversal file format is designed such that it is syntactically similar to AspectJ. The idea is that this extension to AspectJ could be integrated into the compiler in the future. Thus, it uses aspect declaration, declare key words, etc. within it. In the traversal File, one can declare class graph slices and traversals. DAJ takes these and generates AspectJ code that implements the traversal.

The ClassGraph declaration can have two different forms, default or a class graph slice. The default declaration is the ClassGraph object that is obtained from CreateClassGraph using DJ. It has the form:

```
ClassGraph cg1;  
ClassGraph defaultCG;  
ClassGraph oneMoreDefaultCG1;
```

The Class Graph Slice form of the Class Graph declaration looks like a DJ Class Graph constructor with a Class Graph and a strategy.

```
ClassGraph cg2 = new ClassGraph(cg1, "from A to B via C");  
ClassGraph cg3 = new ClassGraph(cg2,  
                                "from Me via Telephone to You");
```

Notice that the Class Graph Slice form of the Class Graph declaration needs to match the previous declaration of the ClassGraph with the first argument of the ClassGraph constructor. This is designed to make this look very much like DJ code.

The Traversal Declaration has two forms as well, a default traversal and a traversal with class graph as an argument. The default traversal basically implements the traversal with the default class graph and has the form:

```
declare traversal t1: "from A to B via C";  
declare traversal myTraversall: "from Me to You via EMail";
```

The traversal with class graph specification takes in a class graph and produces the traversal with it. It has the form:

```
declare traversal t2(cg2): "from Here to There via Points";  
declare traversal myTrav(default): "from A via X to B";
```

The last language feature is the aspect declaration. This is designed to look like AspectJ code. If interested in AspectJ, check out www.aspectj.org. The aspect declaration should contain a list of ClassGraph declarations and traversal declarations. Here is an example of an aspect declaration.

```
aspect MyTraversal {
    ClassGraph defaultCG;
    ClassGraph cg1 = new ClassGraph(defaultCG,
        "from * bypassing {java.lang.String} to *");
    declare traversal t1: "from CompoundFile to SimpleFile";
    declare traversal t2(cg1): "from CompoundFile to *";
}
```

A file should contain one and only aspect declaration and the name of the aspect should match the file name before the .trv extension.

Warning! Strategies must have a valid class name for the start of the traversal specified by the string after the keyword "from".

Compilation Process

The compilation process that DAJ executes has four steps listed below.

- Generation of Stubs based on the traversal
- Compilation of CreateClassGraph.java, AspectJ files and Traversal Files
- Execution of the compiled files with the class specified by the -main option. Traversal implementation in AspectJ code is generated using DJ.
- Compilation of AspectJ Files with generated traversal implementation.

During the generation of stub files, the traversal files with extension .trv is processed. A file with the same name, but with .java extension is created and a method for the traversal is introduced for the class that is the starting point for the traversal specified. This is needed for the first compilation of the AspectJ code for traversal generation.