

AspectJ

What is AspectJ?

AspectJ is an extension built for IDEs such as Eclipse or Netbeans that allows users to use Java like programming to form code as aspect oriented programming (AOP). All Java programs are AspectJ programs, however, AspectJ allows for programmers to define constructs called aspects. These aspects can contain several entities that are unavailable to standard classes.

What is Aspect Oriented Programming (AOP)?

Aspect-oriented programming is a paradigm which allows increased modularization of code by cross-cutting of concerns.

Terminology

Concerns - Are what need to be added to the code, most often in multiple places.

Aspects - Encapsulates the new concepts that are being added.

Join Point - A *join point* is a well-defined point in the program flow. A *pointcut* picks out certain join points and values at those points. Before method calls or after a method is return would be typical join points. A piece of *advice* is code that is executed when a join point is reached. These are the dynamic parts of AspectJ.

Advice - The code that is to be executed at a certain point.

AspectJ For Netbeans

<http://aspectj-netbeans.sourceforge.net/> has an AspectJ plugin for Netbeans. However it is no longer being actively developed. This plugin was developed for Netbeans 3.5.x. There are also a few from the Netbeans plugin page <http://plugins.netbeans.org/>. One developed for Netbeans 6.0 and another 6.7, and after messing around, trying to go through the comments, trying to follow some instructions I've decided to abandon trying to use it in Netbeans.


AspectJ For Eclipse

In light of the problems experienced with netbeans we are just going to make a demo project in Eclipse using AspectJ.

Install

1. Load Eclipse 3.6 or 3.7
2. Under **Help** select **Install New Software...**
3. Click the **Add** button and enter <http://download.eclipse.org/tools/ajdt/36/update> OR <http://download.eclipse.org/tools/ajdt/37/update> depending on your version (3.6 or 3.7)
4. Select **AspectJ Development Tools (Required)** and click **Next**
5. In the **Install Details** click **Next** to install
6. Accept the license agreement and click **Finish**
7. Once installation is complete you will be asked to restart the IDE. Once you do we are good to go.

Tutorial

1. In the Eclipse toolbar click the  button or navigate to **File > New > Project** and select **AspectJ Project**.
2. Fill out **Project Name** and click **Finish**
3. Create a new "helloWorld" java package
4. Create a new "Hello" java class
5. Add a simple main method and another method...

```

public static void main(String[] args) {
    sayHello();
}
public static void sayHello() {
    System.out.print("Hello");
}

```

6. *Let's create an Aspect!* Right click on your **helloWorld** package and create a **New > Other > Aspect**. Call the aspect "World" and click **Finish**.
7. Add the following to the Aspect code.

```

public aspect World {
    pointcut greeting() : execution(* Hello.sayHello(..));

    after() returning() : greeting() {
        System.out.println(" World!");
    }
}

```

8. You can now run **Hello** as a AspectJ/Java Application and the output is "Hello World!"

Crosscutting Information

The cross referencing information is a great way to see how different aspects interact with the different methods. To view the cross referencing information in Eclipse go to **Window > Show View > Other > Cross Referencing**.

Join Point Categories

- Method join points
 - call (* Hello.sayHello(..))
 - execution (* Hello.sayHello(..))
- Constructor join points
 - call (* Hello.new(..))
 - execution (* Hello.new(..))
- Field access join points
 - return "Saying ..." + _say; _say = "Hello"
- Class initialization join points
 - staticinitialization(Hello)
- Object Preinitialization
- Advice Execution

Uses

Aspect Oriented Programming has a lot of uses:

- Untangle complex concepts that are divided across multiple methods into one aspect
 - Makes code readability easier reducing maintenance costs.
 - Easier to pinpoint location for software change
 - Developers can focus on one concern
 - More reuse of code
 - Easy modifications, one aspect across multiple methods
- Simple uses
 - Easily implement logging to multiple methods.
 - Clean input data before all database writes.

Basically anytime a feature needs to be implemented across numerous methods it would be easier to use AOP compared to going to each method and adding code, which could introduce new bugs and is there for more risky.