Advice in AspectJ

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Advice

- It specifies behaviour when a join point is reached
  - Advice is similar to a method in OOP
- Advice can be execute
  - *Before* the joint point
  - *After* the joint point
  - Before and after the joint point (*around*)

Example of Advice

```
pointcut logCredit() :
    call (* Account*.credit(double));

after(): logCredit() {
    System.out.println("Credit done.");
}
```

The code of this advice is executed *after* a call to the credit method

Alternative *After* Advice

- **After returning**
  - It is executed only when the method returns its normal execution
- **After throwing**
  - It is executed only when the method returns an exceptional state (*exception*)

Examples of After Advice

```
pointcut logWithdraw() :
    call (* Account*.withdraw(double));

after() returning: logWithdraw() {
    System.out.println("Withdraw done.");
}

after() throwing: logWithdraw() {
    System.out.println("Withdraw fails.");
}
```

Getting Context Information

- Pointcuts can get context information for the advice execution
  - **args()** : parameters of the method call
  - **target()** : object being called (method) or accessed (attribute)
  - **this()** : object calling the method or accessing the attribute

Example of Context (*target*)

```
pointcut logCredit(Account c) :
     call (* Account*.credit(double)) &&
          target(c);

after(Account c): logCredit(c) {
     System.out.println("Credit done.");
     System.out.println("Account: "+c.getNumber);
}
```

Account being called

Output the account number.

Example of Context (*args*)

```
pointcut logCredit(Account c, double v) :
    call (* Account*.credit(double)) &&
         target(c) && args(v);

after(Account c, double v): logCredit(c, v) {
    System.out.println("Credit done.");
    System.out.println("Account: "+c.getNumber);
    System.out.println("Value: "+v);
}
```

Credit parameter

Output the credit value.

Around Advice

- It replaces the joint point
  - **proceed()** is used to execute the joint point

```
pointcut logWithdraw(Account c, double v) :
    call (* Account.withdraw(double)) &&
    target(c) && args(v);

void around(Account c, double v): logWithdraw(c, v) {
    if (v > c.getBalance())
      System.out.println("No funding to withdraw.");
    else
      proceed(c,v);
}
```

Proceed calling the withdraw method only if account has balance

Example of *Logging* Aspect

```
public aspect LogAccount {

  pointcut logCredit() :
    call (* Account.credit(double));

  pointcut logWithdraw() :
    call (* Account.withdraw(double));

  after(): logCredit() {
    System.out.println("Credit done.");
  }

  after() returning: logWithdraw() {
    System.out.println("Withdraw done.");
  }
}
```

Bibliography

- R. LADDAD. **AspectJ in Action**, 2ª Ed. 2010.
  - Part 1 Understanding AOP and AspectJ