Quick Reference

Version: 1.0

Table of Contents

Chapter 1. Conventions

This document gives a quick overview of how to use the XML and annotation constructs available in JBoss AOP for users who have read through the rest of the documentation. It uses a somewhat non-standard notation, based on how you would actually use an xml snippet or annotation in your code, with optional and alternates marked. The values for attributes and elements have been left out, and for annotations taking only one value, the value should not be specified.

`{a}`
: Item a between curly braces is optional.

`[a]`
: Groups the items between the braces

`[a | b]`
: Use a or b

`multiplicity`
: The multiplicity symbols are the same as for a normal DTD

- `none` - item should occur exactly once
- `?` - item should not occur, or occur once
- `*` - item is optional and can occur several times
- `+` - item must occur one or more times

The parts outside these markers should just be typed in as is.

For the following XML "definition":

```
<test {optional} [choiceA | choiceB] compulsory >
   [ <subA></subA> | <subA></subA> ]?
</test>
      </programlisting>
      These would be valid XML:
      <programlisting>
<test choiceA="y" compulsory="c" >
   <subA>4</subA>
</test>
<test optional="x" choiceB="y" compulsory="c" >
   <subB>4</subB>
</test>
<test optional="x" choiceB="y" compulsory="c" />
      </programlisting>
        These would not be valid:
      <programlisting>
<test choiceA="y" choiceB="y" compulsory="c" >
   <subA>4</subA>
</test>

<test choiceA="y" >
   <subA>4</subA>
</test>

<test choiceA="y" choiceB="y" compulsory="c" >
   <subA>4</subA>
</test>
```

```
<test choiceA="y" compulsory="c" >
   <subA>4</subA>
   <subB>4</subB>
</test>
```

For the following annotation "definitions":

```
@AnnA ( compulsory, [choiceA | choiceB] {, optional1})
@AnnB (expr)
```

These would be valid uses, bearing in mind that for this example compulsory is a String, choiceA and choiceB are integers, optional1 is an array of Strings, and expr is a String:

```
@AnnA ( compulsory="a", choiceA=3, optional1="a")
@AnnA ( compulsory="a", choiceB=3, optional1="a", "b")
@AnnA ( compulsory="a", choiceA=3)
@AnnB ("a")
```

These would not be valid:

```
@AnnA (choiceA=3, optional1={"a", b=""})
@AnnA ( compulsory="a", choiceB=3, choiceB=3, optional1={"a", "b"})
@AnnB (3)
```

Chapter 2. Quick ref

2.1. Aspects

XML

```
<aspect {name} [class |factory] scope>
   [<attribute name></attribute>]*
   [<advisor-attribute  name ></advisor-attribute>]
   [<instance-advisor-attribute name ></instance-advisor-attribute>]
   [<joinpoint-attribute  name ></joinpoint-attribute>]
   [random xml]
</aspect>
```

Annotation

Usage: class

```
@Aspect ([class|factory], scope)
```

2.2. Interceptors

XML

```
<interceptor {name} [class|factory] scope />
        [<attribute name ></attribute>]*
        [<advisor-attribute name ></advisor-attribute>]
        [<instance-advisor-attribute name></instance-advisor-attribute>]
        [<joinpoint-attribute name ></joinpoint-attribute>]
        [random xml]
<interceptor>
```

Annotation

Usage: class

```
@InterceptorDef ([class|factory], scope)
```

2.3. Bindings

XML

```
<bind {name} pointcut {cflow} >
```

```
     [ <interceptor-ref name/> |
       <stack-ref name/> |
       <advice aspect name/>  ]*
   </bind>
```

Annotation

Usage: class (if annotated @InterceptorDef), method (if class annotated @Aspect)

```
Bind (pointcut {, cflow} )
```

2.4. Stacks

XML

```
<stack name>
   [ <interceptor-ref name /> |
     <stack-ref name /> |
     <interceptor [class |factory] scope /> |
     <advice aspect name />  ]+
</stack>
```

Annotation

Usage: N/A

2.5. Pointcuts

XML

```
<pointcut name expr />
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class

```
@PointcutDef (expr)
```

2.6. Interface Introductions

XML

```
<introduction [class|expr] >
        <interfaces></interfaces>
</introduction>
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class

```
@Introduction ([target|typeExpression], interfaces)
```

2.7. Mixins

XML

```
<introduction [class|expr] >
   [
        <mixin {transient} >
            <interfaces></interfaces>
            <class></class>
            <construction></construction>
        </mixin>
   ]*
</introduction>
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class. The method returns the mixin class type

```
@Mixin ([target|typeExpression], interfaces, isTransient)
```

2.8. Annotation Introductions

XML

```
<annotation-introduction expr invisible ></annotation-introduction>
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class.

```
@AnnotationIntroductionDef (expr, invisible, annotation)
```

2.9. CFlow Stack

XML

```
<cflow-stack name>
   [<called></called> | <not-called></not-called>]*
<cflow-stack>
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class.

```
@CFlowStackDef (cflows)
```

Where the cflows entries must be of type

```
@CFlowDef (expr, called)
```

2.10. Typedefs

XML

```
<typedef name expr />
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class.

```
@TypeDef (expr)
```

2.11. Dynamic CFlow

XML

```
<dynamic-cflow name class />
```

Annotation

*Usage:*class.

```
@DynamicCFlowDef (expr)
```

2.12. Prepare

XML

```
<prepare expr />
```

Annotation

*Usage:*field in @Aspect or @InterceptorDef annotated class.

```
@Prepare (expr)
```

2.13. Metadata

XML

```
<metadata tag class>
   [<default>[any xml tags with content]*</default>]*
   [<class>[any xml tags with content]*</class>]*
   [<method expr>[any xml tags with content]*</method>]*
   [<constructor expr>[any xml tags with content]*</constructor>]*
   [<field expr>[any xml tags with content]*</field>]*
</metadata>
```

Annotation

Usage: N/A.

2.14. Metadata Loaders

XML

```
<metadata-loader tag class>
   [arbitrary xml]
</metadata-loader>
```

Annotation

Usage: N/A.