DGen Home

DGen is a tool for generating documenting annotations based on Javadoc.

GitHub Repository	https://github.com/Devexperts/dgen
Public Maven repo	https://bintray.com/devexperts/Maven/dgen
License	GPLv3
Contact	dxlab@devexperts.com

README.md (source from dgen)

Dgen -- @Description generator

Sometimes we would lke to have some documentation at **runtime** (for example to print help messages). This framework automatically annotates specified classes, methods and fields with **@Description**, that contains documentation from theirs **Javadoc**. The framework is implemented as annotation processor and used in **compilation** phase. Just add it to classpath during compilation.

Javadoc processing

As mentioned in the previous section, the description framework generates description from element's Javadoc. It reads documentation and passes it to `@Description` annotation. For methods, framework reads `@param` tags and if possible creates annotations for method parameters too. If element have no Javadoc or first sentence in it (e.g. Javadoc has only tags or it is empty) then no annotation is created.

For example, method

```
/**
 * Method documentation
 *
 * More detailed documentation
 *
 * @param b second
 *          argument
 */
void f(int a, int b) { ... }
```

will be transformed to

```
@com.devexperts.annotation.Description("Method documentation")
void f(
    @com.devexperts.annotation.Description(name = "a", value = "") int a,
    @com.devexperts.annotation.Description(name = "b", value = "second argument") int b
) { ... }
```

Configuration

In file

Configuration represented by a list of class rules. Class rule may contain a list of class matching predicates and will be applied only if a class matches all of them. If no predicates are specified then the rule matches all classes. During execution first suitable rule is chosen. Class rule may contain sub-rules for methods and fields as well. Each of them contains list of predicates with execution logic like in class rule. No method or field annotations generated by default. If several method/field rules are specified then the first matching will be applied.

Default configuration file path is `./dgen.config`. Custom configuration file path may be specified in `dgen.config` annotation processor property: `-Adgen.config=<filename>`.

Configuration file example:

```
class {
    name = ".*\.[AB]"; # matches all classes with name that is matched with ".*\.[AB]" regular expression.

    method {
        isStatic = true; # matches all static methods.
    }
}

class {

    extendsOrImplements = java.util.Collection; # matches all classes that implements java.util.Collection
interface.

    # Contains 2 field rules. Matches field if it's static or have private or package-default access.
    field { # matches all fields with private or package-default access.
        access = private | default;
    }

    field {
        isStatic = true;
    }
}

class { # matches all protected classes with name that is matched with ".*\.G[AB]" regular expression.
    name = ".*\.G[AB]";
    access = protected;
}
```

For more details you can see antlr4 grammar for this configuration in source code.

In Javadoc

You can add **@dgen.annotate** tag in Javadoc to generate annotations for specified element. For classes, in tag value you can add method and field rules like in file configuration. For methods and fields tag value is ignored.

For example, the next class

```
/**
 * My class
 *
 * @annotate field { access = public; }
 */
public class A {

    /**
     * Field x
     */
    public int x;

    /**
     * Field y
     * @annotate
     */
    private int y;

    /**
     * Field z
     */
    int z;
}
```

is transformed to

```
@com.devexperts.annotation.Description("My class")
public class A {

    @com.devexperts.annotation.Description("Field x")
    public int x;

    @com.devexperts.annotation.Description("Field y")
    private int y;

    int z;
}
```

In most cases this configuration is more clear.

Priorities

If element have **@Description** annotation, it isn't replaced. If class have **@dgen.annotate** tag in Javadoc it completely supersedes other configuration rules.

Options

Following options may be specified in rules:

- **retrieveStrategy** – defines strategy for extracting description from Javadoc. Possible values: `firstSentence`, `firstParagraph`, `returnTag`, a`ll`. First sentence by default.
- **annotateClass** – defines should class to be annotated or not. Possible values: `true`, `false`. True by default. May be used for exclusions (shall be specified before more general rules).

Options should be passed in options block and can be passed to class, method and field rules.

Predicates

We can filter processing elements via predicates (see Configuration section). A predicate matches or mismatches element and can be applied only for specified types of elements (e.g. Extends Or Implements).

Following predicates are supported:

Name

Checks that element's name matches with specified *regular expression*. Uses *qualified* name for classes and simple name for other elements.

```
name = ".*\.[AB]"; # matches classes A and B from any packages.
```

Access Modifier

Checks that element's access modifier matches with *ANY* of specified.

Possible access modifiers:

- private
- default
- protected
- public

```
access = private|default; # matches if element have private or default access modifier.
```

Is Static

Checks that element has static modifier or not.

```
isStatic = true; # matches all static methods.
```

Extends Or Implements

Checks that class extends (or implements) specified class (or interface) in declaration. Can be used **only for class rules**.

For example, if class A extends java.util.ArrayList then for class A this predicate, parametrized with *java.util.ArrayList* returns true, but parametrized with *java.util.List* returns false.

```
extendsOrImplements = java.util.Collection; # matches all classes that implements java.util.Collection
interface.
```

Using with Maven

To use with Maven build Dgen should be added as dependency in your **pom.xml**.

```
<dependency>
    <groupId>com.devexperts.dgen</groupId>
    <artifactId>dgen</artifactId>
    <version>1.0</version>
    <scope>compile</scope>
</dependency>
<!-- For @Description -->
<dependency>
    <groupId>com.devexperts.qd</groupId>
    <artifactId>dxlib</artifactId>
    <version>3.232</version>
    <scope>compile</scope>
</dependency>
```

Related articles

Project Lombock

The Hacker's Guide to Javac