RFC 208 Metatype Annotations

Draft

21 Pages

Abstract

This RFC introduces annotations for the Metatype specification which can be use to annotate Java types so that tools can generate Meta Type Resources from the type declaration.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

0.2 Trademarks

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 1.

`Source code is shown in this typeface.`

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	2013-11-19	Initial draft. BJ Hargrave, IBM
2nd	2013-11-20	Updated after feedback from Peter Kriens. Replaced Designate annotation with designate and designateFactory elements on the ObjectClassDefinition annotation. Also added icon element (and Icon annotation) to ObjectClassDefinition. BJ Hargrave, IBM

1 Introduction

The Metatype specification defines a Meta Type Resource format which can be used by Meta Type Service implementations. These resources are XML documents which conform to the Meta Type Resource XML Schema. RFC 190 introduces annotation configuration types to DS so that developers can access their configuration (component properties) in a type safe way. Since the configuration is now describable as a Java type, this RFC will also allow the type to document the Meta Type information about the configuration so that tools can generate Meta Type Resources from the Java type.

2 Application Domain

OSGi has long had the Meta Type specification which defines meta type information for configurations which are stored in Configuration Admin service. The Meta Type definitions are useful by GUIs to allow users to define actual configurations by providing information about the expected data types and values including localized information for a GUI. Meta Type specification also defines a Meta Type Resource format which is an XML document that can be contained in a bundle and processed by the Meta Type service.

Declarative Services uses configurations from Configuration Admin service as component properties for components. RFC 190 is updating DS to allow the component properties to be “shaped” into annotation types to provide components type-safe access to their component properties.

RFC 179 “DS Updates for Configurable” is an RFC which is no longer being worked but which contains the seed of the design now being using in RFC 190 for the configuration annotation types. RFC 179 is based upon RFC 178 “Configurable” which includes design ideas on annotations of these types for Meta Type support.

Bnd has also provided support for Meta Type annotations. See http://www.aqute.biz/Bnd/MetaType. The Meta.OCD and Meta.AD annotations were inputs to RFC 178.

3 Problem Description

Writing Meta Type Resource documents requires the programmer to author an XML document which both conforms to the Meta Type XML schema and accurately reflects the data and data types in the configuration. The programmer must keep changes to the program using the configuration and the XML document in sync. This can be difficult during refactoring and hard to validate during testing to avoid allowing errors from being propagated.

4 Requirements

MTA-0100 – Meta Type resource information must be able to be described in Java source code. This allows for compiler checking of types and refactoring support.

MTA-0200 – Must be able to mark a configuration annotation type (from RFC 190) as a source for Meta Type information.

MTA-0300 – Defaults for meta type information must be derivable from the marked type.

MTA-0400 – The programmer must be able to supply meta type information to override the defaults.

MTA-0500 – Tools must be able to process the meta type information specified in the source so that Meta Type Resource XML documents can be automatically generated.

MTA-0600 – Meta type information from the source files must also be present in the generated class files so tools do not need to process the source files.

5 Technical Solution

5.1 Introduction

A~~Meta-a~~nnotations are defined that can applied to the configuration annotation types from RFC 190. An example from RFC 190:

```
@interface Config {
    boolean enabled() default true;
    String[] names() default {"a", "b"};
    String topic() default MyComponent.DEFAULT_TOPIC_PREFIX + "/topic";
}

@Component
public class MyComponent {
    static final String DEFAULT_TOPIC_PREFIX = "topic.prefix";
    protected void activate(Config configuration) {
        String t = configuration.topic();
    }
}
```

In this example, the Config annotation type is used as a configuration type which is used by the activate method. The Config type describes the “shape” of the configuration and can be used to also describe the meta type information. If we annotate the Config type with the new @ObjectClassDefinition annotation,

```
@ObjectClassDefinition
@interface Config {
    boolean enabled() default true;
    String[] names() default {"a", "b"};
    String topic() default MyComponent.DEFAULT_TOPIC_PREFIX + "/topic";
}
```

a tool (like bnd) processing the bundle can automatically generate a Meta Type Resource XML document from the information in the Config type. This is the main purpose of these annotations: to generate Meta Type Resource XML documents from the configuration annotations. However, the id element of the AttributeDefinition annotation is also needed at runtime for SCR to map the annotated element name to the component property name since the value of the id element is used as the name of the component property instead of the default conversion from the element name.

In this larger example:

```
~~@Designate("test.pid")~~
@ObjectClassDefinition(localization = "OSGI-INF/l10n/test",
                       description = "%test.description",
                       name = "%test.name"
                       designate = “test.pid”,
                       icon = @Icon(resource = "icon/test-32.png", size = 32))
public @interface Test {
        @AttributeDefinition(type = AttributeType.PASSWORD,
                             description = "%test.password.description",
                             name = "%test.password.name")
        public String _password();

        @AttributeDefinition(options = {
                             @Option(label = "%strategic", value = "strategic"),
                             @Option(label = "%principal", value = "principal"),
                             @Option(label = "%contributing", value = "contributing")
        },
                             defaultValue = "contributing",
                             description = "%test.membertype.description",
                             name = "%test.membertype.name")
        public String memberType();

        @AttributeDefinition(id = "my.prop")
        public String blahblah();
}
```

we can see more extensive use of the new annotations. ~~@Designate defines the pid to be associated with the meta type resource.~~ @ObjectClassDefinition marks the Test type as a configuration type for which a meta type resource should be generated. It further defines meta type information including the description and name which are to be localized using the specified resource as well as a pid and an icon resource. @AttributeDefinition marks elements of the Test type to provide meta type information. If meta type information is not provided by the annotation declaration, default information must be generated from the annotated type.

This RFC is tied to RFC 190 in that the annotations defined here are to be applied to the configuration annotation types defined by RFC 190.

5.2 @ObjectClassDefinition

The ObjectClassDefinition annotation is applied to a configuration annotation type to mark it for processing into a Meta Type Resource XML document.

The ObjectClassDefinition annotation can be applied without defining any element values as defaults can be generated from the annotated type. The following elements are defined:

- name – (String) A human readable name of the object, can be localized if it starts with a % sign. The default is a string derived from the id where _, $, or camel casing is used to provide spaces. The name becomes the value of the name attribute of the OCD element in the generated Meta Type Resource XML document.
- id – (String) The id of the object, the default is the fully qualified name of the type with a $ as separator for nested classes. This is not to be confused with a PID which can be specified by an @Designate annotation. The id becomes the value of the id attribute of the OCD element in the generated Meta Type Resource XML document.
- localization – (String) The localization resource of the object. This refers to a resource property entry in the bundle that can be augmented with locale information. The default is the fully qualified name of the class in the OSGI-INF/l10n folder. The localization becomes the value of the localization attribute of the OCD element in the generated Meta Type Resource XML document.
- description – (String) A human readable description that can be localized when it starts with %. Default is the empty string. The description becomes the value of the description attribute of the OCD element in the generated Meta Type Resource XML document.
- designate – (String[]) The PIDs associated with the ObjectClassDefinition. The default is no associated PIDs. The designate information becomes a set of Designate elements for each pid which reference the OCD element in the generated Meta Type Resource XML document.
- designateFactory – (String[]) The factory PIDs associated with the ObjectClassDefinition. The default is no associated factory PIDs. The designateFactory information becomes a set of Designate elements for each factoryPid which reference the OCD element in the generated Meta Type Resource XML document.
- icon – (Icon[]) Specify icons (resource name and size). The default is no icon information. The icon information becomes a set of Icon elements of the OCD element in the generated Meta Type Resource XML document.

Each element of the configuration annotation type annotated by ObjectClassDefinition is mapped to an AD child element of the OCD element in the generated Meta Type Resource XML document. The AttributeDefinition annotation only needs to be applied if values other than the defaults are desired.

5.3 @AttributeDefinition

The AttributeDefinition annotation is an optional annotation which can applied to elements in a configuration annotation type annotated by ObjectClassDefinition. Each element of the configuration annotation type annotated by ObjectClassDefinition is mapped to an AD child element of the OCD element in the generated Meta Type Resource XML document. The AttributeDefinition annotation only needs to be applied if values other than the defaults are desired. The following elements are defined:

- name – (String) A human readable name of the attribute, can be localized if it starts with a % sign. The default is a string derived from the method name where _, $, or camel casing is used to provide spaces. The name becomes the value of the name attribute of the AD element in the generated Meta Type Resource XML document.

- id – (String) The id of the attribute. The id is used as the ~~and~~ name of the configuration property. By default, this is the name of the element converted to a property name as specified in RFC 190 section 5.6.2 (e.g. removal of dollar sign and converting underscore to dot). The id becomes the value of the id attribute of the AD element in the generated Meta Type Resource XML document.

- description – (String) A human readable description that can be localized if it starts with %. Default is the empty string. The description becomes the value of the description attribute of the AD element in the generated Meta Type Resource XML document.

- type – (AttributeType) The type of the attribute. This must be one of the types defined in the Metatype specification. The default is derived from the type of the element. Class and Enum types are mapped to String. Annotation types are not supported. A tool processing the annotation should declare an error during processing in this case. The type is used to select the value of the type attribute of the AD element in the generated Meta Type Resource XML document.

- cardinality - (int) The cardinality of the attribute. The default is 0 if the element is not an array and a large positive number if the element is an array type. The cardinality becomes the value of the cardinality attribute of the AD element in the generated Meta Type Resource XML document.

- min – (String) The minimum value allowed for this attribute. There is no default. The min becomes the value of the min attribute of the AD element in the generated Meta Type Resource XML document.

- max – (String) The maximum value allowed for this attribute. There is no default. The max becomes the value of the max attribute of the AD element in the generated Meta Type Resource XML document.

- defaultValue – (String[]) The default values. The defaultValues are concatenated into a comma delimited list to becomes the value of the default attribute of the AD element in the generated Meta Type Resource XML document.

- required – (boolean) Indicates if this attribute is required. The default is true. The required becomes the value of the required attribute of the AD element in the generated Meta Type Resource XML document.

- options - (@Option[]) Specify options (value and optional label). There is only a default if the element type is an Enum or Enum[] in which case the label is the enum element toString() output and the value is the enum element name() output. The options information becomes a set of Option elements of the AD element in the generated Meta Type Resource XML document.

Also, the id element of the AttributeDefinition annotation is needed at runtime for SCR to map the annotated element name to the component property name since the value of the id element is used as the name of the component property instead of the default conversion from the element name. This is why the retention policy for AttributeDefinition must be RUNTIME.

5.4 ~~@Designate~~

~~The Designate annotation can further mark a type annotated by an ObjectClassDefinition annotation to generate a Designate element referencing the OCD element in the generated Meta Type Resource XML document.~~

5.5 @Option

The Option annotation is only used for the options element of the AttributeDefinition annotation to allow specifying label/value pair for an AttributeDefinition.

5.6 @Icon

The Icon annotation is only used for the icon element of the ObjectClassDefinition annotation to allow specifying a icon resource/size pair.

5.7 Other Changes

Since this RFC will modify the Meta Type Specification and bump its version to 1.3, we can also pick up some minor Meta Type bugs awaiting a specification version change. The metatype package should also be updated to use the new package and type annotations from RFC 197.

5.7.1 Bug 2436

The schema is fixed to use “Character” instead of “Char” to match the proper Java type name and other OSGi specifications like DS and RSA.

5.7.2 Bug 2540

The schema is modified to allow more flexible ordering of elements.

5.8 Open Issues

### Do we need a special value for the default? For example default “” is an empty String but the annotation could be written with an empty String.

### Currently ObjectClassDefinition is a meta-annotations and can only be applied to annotations which are the configuration types used in RFC 190. Should we relax the target type to all types?

6 Javadoc

More Javadoc detail will be added after the first review round of this RFC.

OSGi Javadoc

11/20/13 4:33 PM

Package Summary		*Page*
org.osgi.service.metatype.annotations	Metatype Annotations Package Version 1.3.	*12*

Package org.osgi.service.metatype.annotations

```
@org.osgi.annotation.versioning.Version(value="1.3")
```

Metatype Annotations Package Version 1.3.

See:
  Description

Enum Summary		Page
AttributeType	Types for `AttributeDefinition` annotation.	15

Annotation Types Summary		Page
AttributeDefinition		13
Icon		17
ObjectClassDefinition		18
Option		20

Package org.osgi.service.metatype.annotations Description

Metatype Annotations Package Version 1.3.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.metatype.annotations; version="[1.3,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.metatype.annotations; version="[1.3,1.4)"
```

Annotation Type AttributeDefinition

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value=ElementType.METHOD)
public @interface AttributeDefinition
```

Required Element Summary		Page
int	**cardinality**	14
String[]	**defaultValue**	14
String	**description**	13
String	**id** id	13
String	**max**	14
String	**min**	14
String	**name**	13
Option[]	**options**	14
boolean	**required**	14
AttributeType	**type**	13

Element Detail

name

`public abstract String name`

Default:
""

description

`public abstract String description`

Default:
""

id

`public abstract String id`

id

Default:
""

type

`public abstract AttributeType type`

Default:
AttributeType.STRING

---

cardinality

```
public abstract int cardinality
```

Default:
0

---

min

```
public abstract String min
```

Default:
""

---

max

```
public abstract String max
```

Default:
""

---

defaultValue

```
public abstract String[] defaultValue
```

Default:
{}

---

required

```
public abstract boolean required
```

Default:
true

---

options

```
public abstract Option[] options
```

Default:
{}

Enum AttributeType

org.osgi.service.metatype.annotations

```
java.lang.Object
  └java.lang.Enum<AttributeType>
      └org.osgi.service.metatype.annotations.AttributeType
```

All Implemented Interfaces:

Comparable<AttributeType>, Serializable

```
public enum AttributeType
extends Enum<AttributeType>
```

Types for `AttributeDefinition` annotation.

Enum Constant Summary	Page
BOOLEAN	16
BYTE	16
CHARACTER	16
DOUBLE	15
FLOAT	16
INTEGER	16
LONG	15
PASSWORD	16
SHORT	16
STRING	15

Method Summary		Page
String	**toString**()	16
static AttributeType	**valueOf**(String name)	16
static AttributeType[]	**values**()	16

Enum Constant Detail

STRING

```
public static final AttributeType STRING
```

LONG

```
public static final AttributeType LONG
```

DOUBLE

```
public static final AttributeType DOUBLE
```

FLOAT

```
public static final AttributeType FLOAT
```

INTEGER

```
public static final AttributeType INTEGER
```

BYTE

```
public static final AttributeType BYTE
```

CHARACTER

```
public static final AttributeType CHARACTER
```

BOOLEAN

```
public static final AttributeType BOOLEAN
```

SHORT

```
public static final AttributeType SHORT
```

PASSWORD

```
public static final AttributeType PASSWORD
```

Method Detail

values

```
public static AttributeType[] values()
```

valueOf

```
public static AttributeType valueOf(String name)
```

toString

```
public String toString()
```

Overrides:

`toString` in class `Enum`

Annotation Type Icon

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value={})
public @interface Icon
```

Required Element Summary		Page
String	resource	17
int	size	17

Element Detail

resource

```
public abstract String resource
```

size

```
public abstract int size
```

Annotation Type ObjectClassDefinition

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value=ElementType.ANNOTATION_TYPE)
public @interface ObjectClassDefinition
```

Required Element Summary		Page
String	description	18
String[]	designate	19
String[]	designateFactory	19
Icon[]	icon	19
String	id id	18
String	localization	18
String	name	18

Element Detail

id

```
public abstract String id
```

id

Default:
""

name

```
public abstract String name
```

Default:
""

localization

```
public abstract String localization
```

Default:
""

description

```
public abstract String description
```

Default:
""

designate

```
public abstract String[] designate
```

Default:
{}

---

designateFactory

```
public abstract String[] designateFactory
```

Default:
{}

---

icon

```
public abstract Icon[] icon
```

Default:
{}

Annotation Type Option

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value={})
public @interface Option
```

Required Element Summary		Page
String	label	20
String	value	20

Element Detail

value

```
public abstract String value
```

label

```
public abstract String label
```

Default:
""

Java API documentation generated with DocFlex/Doclet v1.5.6

DocFlex/Doclet is both a multi-format Javadoc doclet and a free edition of DocFlex/Javadoc. If you need to customize your Javadoc without writing a full-blown doclet from scratch, DocFlex/Javadoc may be the only tool able to help you! Find out more at www.docflex.com

7 Considered Alternatives

7.1 @Designate

The Designate annotation was removed and replaced by the designate and designateFactory elements on the ObjectClassDefinition annotation.

The Designate annotation can further mark a type annotated by an ObjectClassDefinition annotation to generate a Designate element referencing the OCD element in the generated Meta Type Resource XML document.

8 Security Considerations

The annotations do not have any security considerations.

9 Document Support

9.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

9.2 Author's Address

Name	BJ Hargrave
Company	IBM

9.3 Acronyms and Abbreviations

9.4 End of Document