OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

JEP Process

Source code
Mercurial
GitHub

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects
(overview)
Amber
Annotations Pipeline 2.0
Audio Engine
Build Infrastructure
CRaC
Caciocavallo
Closures
Code Tools
Coin
Common VM Interface
Compiler Grammar
Detroit
Developers' Guide
Device I/O
Duke
Font Scaler
Framebuffer Toolkit
Graal
Graphics Rasterizer
HarfBuzz Integration
IcedTea
JDK 6
JDK 7
JDK 7 Updates
JDK 8
JDK 8 Updates
JDK 9
JDK (... 18, 19, 20)
JDK Updates
JavaDoc.Next
Jigsaw
Kona
Kulla
Lambda
Lanai
Leyden
Lilliput
Locale Enhancement
Loom
Memory Model Update
Metropolis
Mission Control
Modules
Multi-Language VM
Nashorn
New I/O
OpenJFX
Panama
Penrose
Port: AArch32
Port: AArch64
Port: BSD
Port: Haiku
Port: Mac OS X
Port: MIPS
Port: Mobile
Port: PowerPC/AIX
Port: RISC-V
Port: s390x
Portola
SCTP
Shenandoah
Skara
Sumatra
ThreeTen
Tiered Attribution
Tsan
Type Annotations
XRender Pipeline
Valhalla

JEP 120: Repeating Annotations

Author	Joseph D. Darcy
Owner	Alex Buckley
Type	Feature
Scope	SE
Status	Closed / Delivered
Release	8
Component	specification / language
JSRs	269 MR, 337
Discussion	enhanced dash metadata dash spec dash discuss at openjdk dot java dot net
Effort	S
Duration	M
Depends	JEP 104: Type Annotations
Endorsed by	Brian Goetz
Created	2011/10/17 20:00
Updated	2015/02/13 19:40
Issue	8046110

Summary

Change the Java programming language to allow multiple application of annotations with the same type to a single program element.

Goals

Improve the readability of source code which logically applies multiple instances of the same annotation type to a given program element.

Motivation

Frequently-used idioms of programming with annotations in EE and elsewhere awkwardly use a container annotation just to simulate the ability to apply multiple annotations. Building in support for repeated annotations will improve the readability of source code.

Description

The basic approach to implement the language feature is to desugar repeated annotations of a base type into a single container annotation; the container annotation has a `value` method which returns an array of the base annotation type. For repeated annotations to be enabled for a particular annotation type, the declaration of the base annotation type will need to include a new meta-annotation, say `@ContainerAnnotation`, to declare which other annotation type should be used as a container. Warnings and errors should be issued if the container is not suitability compatible with the base annotation, including problematic differences in retention policy or target.

Open design issues include whether or not multiple levels of compiler-generated containers will be supported. For example, should

```
@A(1)
@A(2)
@AContainer
@AContainerContainer
foo();
```

be treated as logically equivalent to

```
@AContainerContainer(@AContainer({@A(1), @A2}), @AContainer)
foo();
```

or a compilation error after one level of nesting to

```
@AContainer({@A(1), @A(2)})
@AContainer
@AContainerContainer
foo();
```

At a libraries level, the implementations of the reflective APIs in the platform, including core reflection and `javax.lang.model`, will need to be updated to handle the repeated annotation information. For example, the `AnnotatedElement.getAnnotation(BaseAnnotation.class)` method will be redefined to look for in a container annotation if the base annotation is not directly present. One or more methods to query for the presence of absence of annotations may be added to the `AnnotatedElement` interface. If multiple levels of compiler-generated nesting is supported, the library changes will be more extensive.

Testing

As with all language changes, the corresponding compiler JCK tests will need to be updated.

Risks and Assumptions

One risk is the possibility of currently unforeseen interactions between this language feature and existing library semantics or between this language feature other language features present in Java SE 8. In particular, the interaction, if any, between repeated annotations and annotations on types will need to be defined.

An assumption is that existing EE annotation types which use the manual container annotation pattern will migrate to use the repeating annotations pattern and thereby validate the feature through usage.

If the various Java IDEs do not support this language change during development, experimentation with the feature and validation of its design will be slowed.

Dependences

The interaction between repeating annotations and JEP 104: Annotations on Java Types, if any, needs to be defined.

Impact

- Other JDK components: Direct use of repeated annotations in the JDK code base is expected to be minimal.
- Compatibility: Interfaces not usually expected to have implementations outside of the JDK may have methods added to them. Defender methods from Project Lambda can help limit the source compatibility impact.
- Performance/scalability: There should be no degradation in compilation speed if the feature is not used.