Guava Release 15.0: Release Notes

- Release 15.0 was release on September 06, 2013.
- Release 15.0-rc1 was released on August 26, 2013.

(See ReleaseHistory.)

Full API Documentation

Using Guava in your project

This release will be identified in the Maven Central repository as com.google.guava:guava:15.0 and com.google.guava:guava-gwt:15.0

See UseGuavaInYourBuild for help integrating Guava into your build environment.

If you don't use managed dependencies, you can also just manually download JARs of the classes, sources and documentation from:

- guava-15.0.jar
- guava-gwt-15.0.jar (for GWT users)
- guava-15.0-javadoc.jar (Javadoc)
- guava-15.0-sources.jar (Source)

A note on JEE6 / CDI 1.0

A workaround added in Guava 15.0 to make it compatible with CDI 1.1 (used in JEE7 containers) caused problems for Guava with CDI 1.0 (used in JEE6 containers).

If you're using Guava in a CDI 1.0 environment, you should use guava-15.0-cdi1.0.jar instead of the normal Guava jar. In Maven, the dependency can be specified as:

```
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>15.0</version>
  <classifier>cdi1.0</classifier>
</dependency>
```

Issues resolved

53 issues are resolved in this release.

API Changes

Full JDiff Report of changes since release 14.0.1

To build a combined report of the API changes between release 15.0 and any older release, check out our docs tree and run `jdiff/jdiff.sh` with the previous release number as argument (example: `jdiff.sh 5.0`).

Significant API additions

common.escape (new) `Escaper`, `Escapers`, various simple `Escaper` implementations.

common.html (new) `HtmlEscapers`

common.xml (new) `XmlEscapers`

common.base `StandardSystemProperty`

`Splitter.splitToList`

common.collect `TreeTraverser`, `BinaryTreeTraverser`

`EvictingQueue`

`Multimaps.asMap`

`Queues.synchronizedDeque`

`Sets.newConcurrentHashSet`

common.hash `Funnels.sequentialFunnel`

common.io `ByteSource.concat`, `empty`, `isEmpty`

`CharSource.concat`, `empty`, `isEmpty`

`CharStreams.nullWriter`

`Files.fileTreeTraverser`, `isDirectory`, `isFile`

common.math `DoubleMath.mean`

common.net `UrlEscapers`

common.reflect `TypeResolver`

common.util.concurrent `ListenableScheduledFuture`

Significant API changes

The `Stopwatch` constructors have been deprecated in favor of static `createStarted()` and `createUnstarted()` methods.

The `Constraint` interface and methods in `Constraints` have been deprecated.

The static methods in `HashCodes` have been moved to `HashCode`.

`HashFunction.hashString`, `Hasher.putString` and `Funnels.stringFunnel` overloads that do not take a `Charset` have been renamed to `hashUnencodedChars`, `putUnencodedChars` and `unencodedCharsFunnel`, respectively.

`ByteSource`, `ByteSink`, `CharSource` and `CharSink` have temporarily been changed to implement `InputSupplier` and `OutputSupplier`. Additionally, `ByteStreams.asByteSource(InputSupplier)`, `ByteStreams.asByteSink(OutputSupplier)`, `CharStreams.asCharSource(InputSupplier)` and `CharStreams.asCharSink(OutputSupplier)` methods have been added to adapt existing Suppliers to Sources and Sinks. These changes are all intended to help make migration easier and will be reverted in a future release.

`ByteStreams.asByteSource(byte[])` has been moved to `ByteSource.wrap(byte[])`. `CharStreams.asCharSource(String)` has been moved to `CharSource.wrap(CharSequence)`.

`ListeningScheduledExecutorService` now returns the new `ListenableScheduledFuture` type from its `schedule*` methods. To implement this, methods on `MoreExecutors.listeningDecorator` executors have been changed to no longer directly call the corresponding methods on the delegate. For example, the decorator's `schedule(Callable, long, TimeUnit)` now calls the delegate's `schedule(Runnable, long, TimeUnit)`.

Some changes have been made to the `Service` interface. `start()`, `startAndWait()`, `stop()` and `stopAndWait()` have been deprecated in favor of new `startAsync()`, `stopAsync()`, `awaitRunning()` and `awaitTerminated()` methods.

Other notable changes

`Ordering.natural` now always delegates directly to `compareTo` without checking for identical inputs first. This should affect only broken classes whose `compareTo` implementations treat an object as unequal to itself (that is, that are non-reflexive).