User Guide

The Guava project contains several of Google's core libraries that we rely on in our Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, and so forth. Each of these tools really do get used every day by Googlers, in production services.

But trawling through Javadoc isn't always the most effective way to learn how to make best use of a library. Here, we try to provide readable and pleasant explanations of some of the most popular and most powerful features of Guava.

This wiki is a work in progress, and parts of it may still be under construction.

- Basic utilities: Make using the Java language more pleasant.
  - [[Using and avoiding null|UsingAndAvoidingNullExplained]]: `null` can be ambiguous, can cause confusing errors, and is sometimes just plain unpleasant. Many Guava utilities reject and fail fast on nulls, rather than accepting them blindly.
  - [[Preconditions|PreconditionsExplained]]: Test preconditions for your methods more easily.
  - [[Common object methods|CommonObjectUtilitiesExplained]]: Simplify implementing `Object` methods, like `hashCode()` and `toString()`.
  - [[Ordering|OrderingExplained]]: Guava's powerful "fluent `Comparator`" class.
  - [[Throwables|ThrowablesExplained]]: Simplify propagating and examining exceptions and errors.
- Collections: Guava's extensions to the JDK collections ecosystem. These are some of the most mature and popular parts of Guava.
  - [[Immutable collections|ImmutableCollectionsExplained]], for defensive programming, constant collections, and improved efficiency.
  - [[New collection types|NewCollectionTypesExplained]], for use cases that the JDK collections don't address as well as they could: multisets, multimaps, tables, bidirectional maps, and more.
  - [[Powerful collection utilities|CollectionUtilitiesExplained]], for common operations not provided in `java.util.Collections`.
  - [[Extension utilities|CollectionHelpersExplained]]: writing a `Collection` decorator? Implementing `Iterator`? We can make that easier.
- [[Graphs|GraphsExplained]]: a library for modeling graph-structured data, that is, entities and the relationships between them. Key features include:
  - Graph: a graph whose edges are anonymous entities with no identity or information of their own.
  - ValueGraph: a graph whose edges have associated non-unique values.
  - Network: a graph whose edges are unique objects.
  - Support for graphs that are mutable and immutable, directed and undirected, and several other properties.
- [[Caches|CachesExplained]]: Local caching, done right, and supporting a wide variety of expiration behaviors.
- [[Functional idioms|FunctionalExplained]]: Used sparingly, Guava's functional idioms can significantly simplify code.
- Concurrency: Powerful, simple abstractions to make it easier to write correct concurrent code.
  - [[ListenableFuture|ListenableFutureExplained]]: Futures, with callbacks when they are finished.
  - [[Service|ServiceExplained]]: Things that start up and shut down, taking care of the difficult state logic for you.
- [[Strings|StringsExplained]]: A few extremely useful string utilities: splitting, joining, padding, and more.
- [[Primitives|PrimitivesExplained]]: operations on primitive types, like `int` and `char`, not provided by the JDK, including unsigned variants for some types.
- [[Ranges|RangesExplained]]: Guava's powerful API for dealing with ranges on `Comparable` types, both continuous and discrete.

- [[I/O|IOExplained]]: Simplified I/O operations, especially on whole I/O streams and files, for Java 5 and 6.
- [[Hashing|HashingExplained]]: Tools for more sophisticated hashes than what's provided by `Object.hashCode()`, including Bloom filters.
- [[EventBus|EventBusExplained]]: Publish-subscribe-style communication between components without requiring the components to explicitly register with one another.
- [[Math|MathExplained]]: Optimized, thoroughly tested math utilities not provided by the JDK.
- [[Reflection|ReflectionExplained]]: Guava utilities for Java's reflective capabilities.
- Tips: Getting your application working the way you want it to with Guava.
    - [[Philosophy|PhilosophyExplained]]: what Guava is and isn't, and our goals.
    - [[Using Guava in your build|UseGuavaInYourBuild]], with build systems including Maven, Gradle, and more.
    - [[Using ProGuard|UsingProGuardWithGuava]] to avoid bundling parts of Guava you don't use with your JAR.
    - [[Apache Commons equivalents|ApacheCommonCollectionsEquivalents]], helping you translate code from using Apache Commons Collections.
    - [[Compatibility|Compatibility]], details between Guava versions.
    - [[Idea Graveyard|IdeaGraveyard]], feature requests that have been conclusively rejected.
    - [[Friends|FriendsOfGuava]], open-source projects we like and admire.
    - [[HowToContribute|HowToContribute]], how to contribute to Guava.

NOTE: To discuss the contents of this wiki, please just use the guava-discuss mailing list.