Algorithms

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

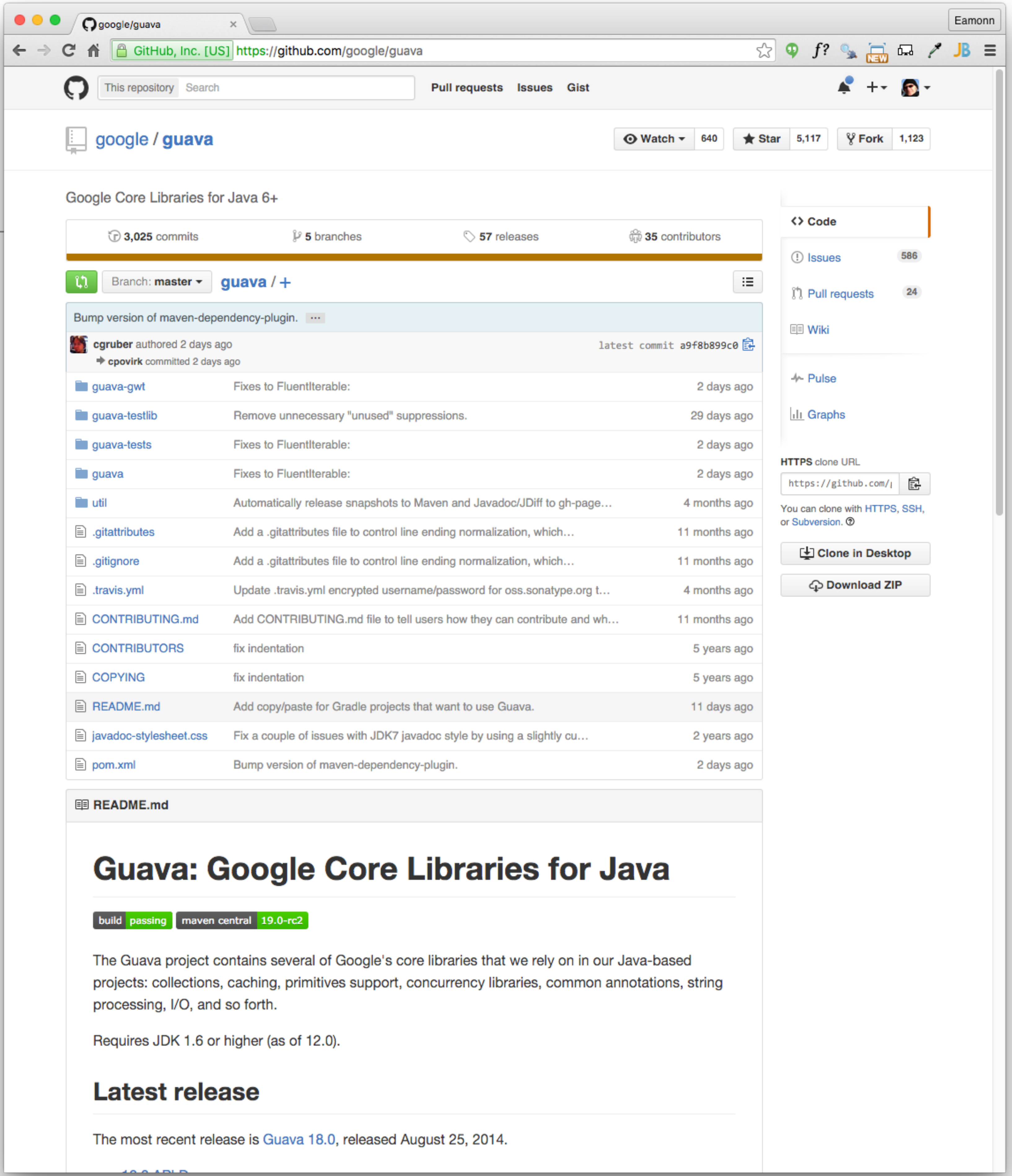
http://www.wit.ie

http://elearning.wit.ie

Pacemaker Lab02

Guava

- Google's Java Libraries
- Consider it an extension to the JDK to be included in all your projects



google/guava

GitHub, Inc. [US] https://github.com/google/guava

google / guava

Watch 640 · Star 5,117 · Fork 1,123

Google Core Libraries for Java 6+

3,025 commits · 5 branches · 57 releases · 35 contributors

Branch: master · guava / +

Bump version of maven-dependency-plugin. ...

cgruber authored 2 days ago · cpovirk committed 2 days ago · latest commit a9f8b899c0

guava-gwt	Fixes to FluentIterable:	2 days ago
guava-testlib	Remove unnecessary "unused" suppressions.	29 days ago
guava-tests	Fixes to FluentIterable:	2 days ago
guava	Fixes to FluentIterable:	2 days ago
util	Automatically release snapshots to Maven and Javadoc/JDiff to gh-page…	4 months ago
.gitattributes	Add a .gitattributes file to control line ending normalization, which…	11 months ago
.gitignore	Add a .gitattributes file to control line ending normalization, which…	11 months ago
.travis.yml	Update .travis.yml encrypted username/password for oss.sonatype.org t…	4 months ago
CONTRIBUTING.md	Add CONTRIBUTING.md file to tell users how they can contribute and wh…	11 months ago
CONTRIBUTORS	fix indentation	5 years ago
COPYING	fix indentation	5 years ago
README.md	Add copy/paste for Gradle projects that want to use Guava.	11 days ago
javadoc-stylesheet.css	Fix a couple of issues with JDK7 javadoc style by using a slightly cu…	2 years ago
pom.xml	Bump version of maven-dependency-plugin.	2 days ago

README.md

Guava: Google Core Libraries for Java

build passing · maven central 19.0-rc2

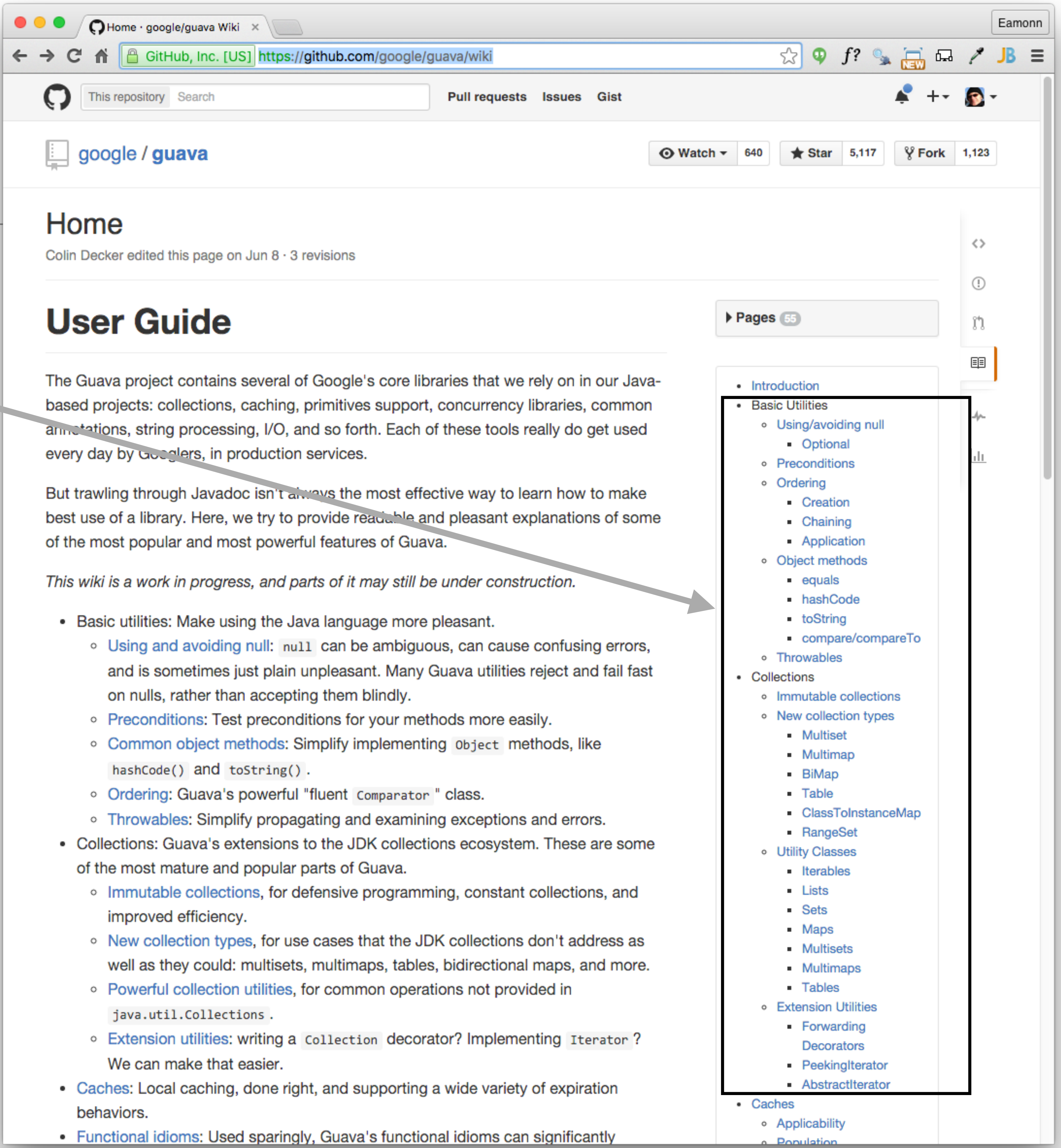
The Guava project contains several of Google's core libraries that we rely on in our Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, and so forth.

Requires JDK 1.6 or higher (as of 12.0).

Latest release

The most recent release is Guava 18.0, released August 25, 2014.

- Dominated but revisions and extensions the collections libraries



Home

Colin Decker edited this page on Jun 8 · 3 revisions

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.

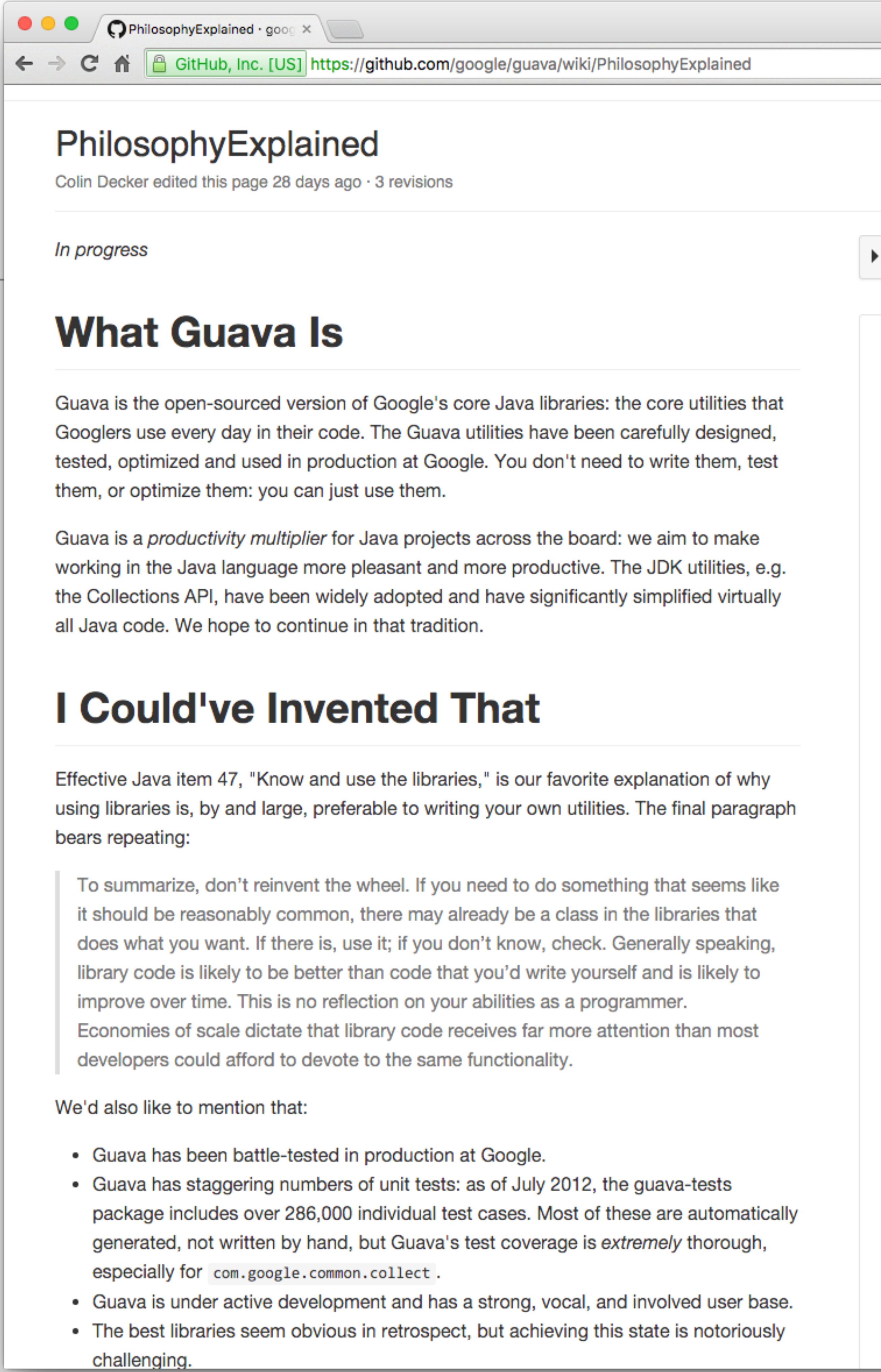
- Basic utilities: Make using the Java language more pleasant.
  - Using and avoiding null: `null` can be ambiguous, can cause confusing errors, and is sometimes just plain unpleasant. Many Guava utilities reject and fail fast on nulls, rather than accepting them blindly.
  - Preconditions: Test preconditions for your methods more easily.
  - Common object methods: Simplify implementing `Object` methods, like `hashCode()` and `toString()`.
  - Ordering: Guava's powerful "fluent `Comparator`" class.
  - Throwables: 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, for defensive programming, constant collections, and improved efficiency.
  - New collection types, 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, for common operations not provided in `java.util.Collections`.
  - Extension utilities: writing a `Collection` decorator? Implementing `Iterator`? We can make that easier.
- Caches: Local caching, done right, and supporting a wide variety of expiration behaviors.
- Functional idioms: Used sparingly, Guava's functional idioms can significantly

Pages 55

- Introduction
- Basic Utilities
  - Using/avoiding null
    - Optional
  - Preconditions
  - Ordering
    - Creation
    - Chaining
    - Application
  - Object methods
    - equals
    - hashCode
    - toString
    - compare/compareTo
  - Throwables
- Collections
  - Immutable collections
  - New collection types
    - Multiset
    - Multimap
    - BiMap
    - Table
    - ClassToInstanceMap
    - RangeSet
  - Utility Classes
    - Iterables
    - Lists
    - Sets
    - Maps
    - Multisets
    - Multimaps
    - Tables
  - Extension Utilities
    - Forwarding Decorators
    - PeekingIterator
    - AbstractIterator
- Caches
  - Applicability
  - Population

Guava Philosophy

"Guava is a **productivity multiplier** *for Java projects across the board: we aim to make working in the Java language more pleasant and more productive. The JDK utilities, e.g. the Collections API, have been widely adopted and have significantly simplified virtually all Java code. We hope to continue in that tradition."*



PhilosophyExplained · goog
GitHub, Inc. [US] https://github.com/google/guava/wiki/PhilosophyExplained

PhilosophyExplained

Colin Decker edited this page 28 days ago · 3 revisions

In progress

What Guava Is

Guava is the open-sourced version of Google's core Java libraries: the core utilities that Googlers use every day in their code. The Guava utilities have been carefully designed, tested, optimized and used in production at Google. You don't need to write them, test them, or optimize them: you can just use them.

Guava is a *productivity multiplier* for Java projects across the board: we aim to make working in the Java language more pleasant and more productive. The JDK utilities, e.g. the Collections API, have been widely adopted and have significantly simplified virtually all Java code. We hope to continue in that tradition.

I Could've Invented That

Effective Java item 47, "Know and use the libraries," is our favorite explanation of why using libraries is, by and large, preferable to writing your own utilities. The final paragraph bears repeating:

> To summarize, don't reinvent the wheel. If you need to do something that seems like it should be reasonably common, there may already be a class in the libraries that does what you want. If there is, use it; if you don't know, check. Generally speaking, library code is likely to be better than code that you'd write yourself and is likely to improve over time. This is no reflection on your abilities as a programmer. Economies of scale dictate that library code receives far more attention than most developers could afford to devote to the same functionality.

We'd also like to mention that:

- Guava has been battle-tested in production at Google.
- Guava has staggering numbers of unit tests: as of July 2012, the guava-tests package includes over 286,000 individual test cases. Most of these are automatically generated, not written by hand, but Guava's test coverage is *extremely* thorough, especially for `com.google.common.collect`.
- Guava is under active development and has a strong, vocal, and involved user base.
- The best libraries seem obvious in retrospect, but achieving this state is notoriously challenging.

User

```java
package models;

import static com.google.common.base.MoreObjects.toStringHelper;
import com.google.common.base.Objects;

public class User
{
  static Long   counter = 0l;

  public Long   id;
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  public User()
  {
  }

  public User(String firstName, String lastName, String email, String password)
  {
    this.id         = counter++;
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
  }

  public String toString()
  {
    return toStringHelper(this).addValue(firstName)
                               .addValue(lastName)
                               .addValue(password)
                               .addValue(email)
                               .toString();
  }

  @Override
  public int hashCode()
  {
     return Objects.hashCode(this.lastName, this.firstName, this.email, this.password);
  }
}
```

PacemakerAPI

- Store users indexed by email and id.

```
public class PacemakerAPI
{
  private Map<Long, User>     userIndex       = new HashMap<>();
  private Map<String, User>   emailIndex      = new HashMap<>();

  public Collection<User> getUsers ()
  {
    return userIndex.values();
  }

  public  void deleteUsers()
  {
    userIndex.clear();
    emailIndex.clear();
  }

  public User createUser(String firstName, String lastName, String email, String password)
  {
    User user = new User (firstName, lastName, email, password);
    userIndex.put(user.id, user);
    emailIndex.put(email, user);
    return user;
  }

  public User getUserByEmail(String email)
  {
    return emailIndex.get(email);
  }

  public User getUser(Long id)
  {
    return userIndex.get(id);
  }

  public void deleteUser(Long id)
  {
    User user = userIndex.remove(id);
    emailIndex.remove(user.email);
  }
}
```

Main

```
public class Main
{
  public static void main(String[] args) throws IOException
  {
    PacemakerAPI pacemakerAPI = new PacemakerAPI();

    pacemakerAPI.createUser("Bart", "Simpson",   "bart@simpson.com",  "secret");
    pacemakerAPI.createUser("Homer", "Simpson",  "homer@simpson.com", "secret");
    pacemakerAPI.createUser("Lisa", "Simpson",   "lisa@simpson.com", “ secret");

    Collection<User> users = pacemakerAPI.getUsers();
    System.out.println(users);

    User homer = pacemakerAPI.getUserByEmail("homer@simpson.com");
    System.out.println(homer);

    pacemakerAPI.deleteUser(homer.id);
    users = pacemakerAPI.getUsers();
    System.out.println(users);
  }
}
```

Firsts test + Fixture

```
public class Fixtures
{
  public static User[] users =
  {
    new User ("marge", "simpson", "marge@simpson.com",  "secret"),
    new User ("lisa",  "simpson", "lisa@simpson.com",   "secret"),
    new User ("bart",  "simpson", "bart@simpson.com",   "secret"),
    new User ("maggie","simpson", "maggie@simpson.com", "secret")
  };
}
```

```
public class UserTest
{
  User homer = new User ("homer", "simpson", "homer@simpson.com",  "secret");

  @Test
  public void testCreate()
  {
    assertEquals ("homer",               homer.firstName);
    assertEquals ("simpson",             homer.lastName);
    assertEquals ("homer@simpson.com",   homer.email);
    assertEquals ("secret",              homer.password);
  }

  @Test
  public void testIds()
  {
    Set<Long> ids = new HashSet<>();
    for (User user : users)
    {
      ids.add(user.id);
    }
    assertEquals (users.length, ids.size());
  }

  @Test
  public void testToString()
  {
    assertEquals ("User{" + homer.id + ", homer, simpson, secret, homer@simpson.com}", homer.toString());
  }
```