Guava

Google + Java

Guava

- Librarie générale comblant des lacunes de Java
- Trois parties
  - Primitives
  - I/O
  - Collections

Guava vs Apache Commons

Guava

- Fait après Java 1.5 (pensé avec les génériques)
- Compatible avec GWT

Apache Commons

- Fait avant Java 1.5 (les génériques sont venus après)

PARTIE I

Primitives

Primitives

- Classes utilitaires:
  - Objects
  - Strings

La classe Objects

- Problème: comment vérifier si deux objets o1 et o2 sont égaux?
  - `if (o1.equals(o2)) {`

    NullPointerException if o1 == null
  - `if (o1 != null && o1.equals(o2)) {`

    false if o1 and o2 == null (incorrect)
  - `if ((o1 == null) && (o2 == null)) || (o1 != null && o1.equals(o2))) {`

    oui! Mais qui est assez discipliné pour faire cela?

La classe Objects

- Solution: la classe utilitaire Objects de Guava
  - `if (Objects.equals(o1, o2) {`

  - returns true if both objects are null (correct)
  - returns false if only one object is null (correct)
  - else, returns o1.equals(o2) (correct)

Note: Objects implanté dans JDK7:

- http://marxsoftware.blogspot.com/2011/03/jdk-7-new-objects-class.html
- http://marxsoftware.blogspot.com/2011/06/java-7-objects-powered-compact-equals.html

La classe Objects

- La méthode equals() doit être:
  - reflective: o1.equals(o1) must be true
  - symétrique: if o1.equals(o2), then o2.equals(o2)
  - transitive: if o1.equals(o2) and o2.equals(o3) then o1.equals(o3)

- Guava vérifie la réflectivité de l'égalité
- Apache Commons définit un reflectionEquals(o1, o2), qui vérifie si o1.equals(o2) == o2.equals(o1). Exécution plus longue, mais résultat plus certain. Intéressant.
- Guava définit un Objects.deepEquals(o1, o2)

La classe Objects

- A chaque fois que l'on implémente equals(), on devrait implémenter hashCode()
- Guava fournir Objects.hashCode :

```
class Employee {
    private String firstName, lastName;
    private Date birthDate;


    @Override
    public boolean equals(Object that) {
        boolean equal = Objects.equal(this.firstName, that.firstName) &&
            Objects.equal(this.lastName, that.lastName) && ..
        return equal;
    }

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

La classe Objects

- Chaque classe (contenant des données) devrait implémenter toString().
- Guava fournir Objects.toStringHelper :

```
class Employee {
    private String firstName, lastName;
    private Date birthDate;

    ..

    @Override
    public String toString {
        String s = Objects.toStringHelper(this)
          .addValue(this.firstName).addValue(this.lastName).addValue(..
        return s;
    }
```

La classe Strings

- La classe Strings
  - La méthode Strings.isNullOrEmpty()
- La classe Splitter

PARTIE II

I/O

La classe Files

- La classe Files

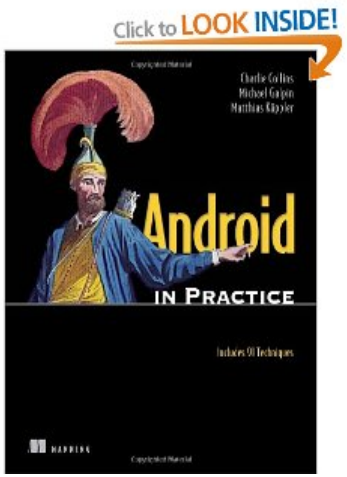
PARTIE III

Collections

Guava Collections

- La classe Filter
- Programmation fonctionnelle

On y parle de Guava



References

- Google Guava vs Apache Commons for Argument Validation
  - http://piotrjagielski.com/blog/google-guava-vs-apache-commons-for-argument-validation/
- 5 Reasons to use Guava
  - http://insightfullogic.com/blog/2011/oct/21/5-reasons-use-guava/
  -

Popularity

Guava : historique

- 2000