JAVA INSIDE
TP 02 - Reflection, Annotation, JSON

- Reflection
- Introspection
- Annotation
- ClassValue

Mélissa DA COSTA

Réflexion

La Réflexion (Reflection en anglais) est l'introspection des classes.

Cela permet d'accéder aux informations d'une classe pour généraliser du code et ne pas avoir à le dupliquer car :

« A kitten dies each time you duplicate a bug ! »

Utilisation de la réflexion

java.lang.Class.getMethods() : retourne un tableau de Method contenue dans la classe.

Cette méthode est très lente car les objets Method sont mutable donc elle doit cloner tous ses objets.

Method.getName() : récupère le nom de la méthode.

Utile dans notre cas pour filtrer seulement les méthodes commencant par « get » :

```
Arrays.stream(object.getClass().getMethods())
.filter(method->method.getName().startsWith("get"))
```

method.invoke(object)

Permet d’exécuter la méthode sur l’objet passé en paramètre.

Gestion des exceptions :

```
try {
    return method.invoke(obj);
// Pas les exceptions qui héritent de runtime : elles se propagent
} catch (IllegalAccessException e) {  // Si la méthode est inaccessible
    throw new IllegalStateException(e);
} catch (InvocationTargetException e) { // Si la méthode appelée renvoie une
exception
    var cause = e.getCause();
    if (cause instanceof RuntimeException)
        throw (RuntimeException) cause; // Re-propage l'exception
    if (cause instanceof Error)
        throw (Error) cause;

    throw new UndeclaredThrowableException(cause);

}
```

Création d’une annotation

Une annotation est une métadonnée qui donne des informations sur du code.

```
@Documented // Spécifie l'annotation dans la javadoc
@Target(ElementType.METHOD) // C'est les méthodes qu'on annote
@Retention(RetentionPolicy.RUNTIME) // Visible a l'exécution

public @interface JSONProperty {
String value() default "";      // Valeur donnée à l’annotation d’une méthode
}
```

Utilisation des annotations

Exemple d'annotation d'une méthode :

```
    @JSONProperty() // Valeur de l'annotation = « planet »
        public String getPlanet() {
            return planet;
        }
```

Idem :

```
    @JSONProperty(« planet »)
        public String getPlanet() {
            return planet;
        }
```

Pour récupérer la valeur d'une annotation d'une méthode :

```
var valueAnnotation=method.getAnnotation(JSONProperty.class).value();
```

Gestion du cache avec ClassValue

ClassValue permet de stocker des valeurs.

Il faut redéfinir la méthode computeValue qui récupère les données à stocker :

```
 private static final ClassValue<Method[]> cache =new ClassValue<>() {

//  Redéfinie la méthode car ClassValue est une classe abstraite

  @Override protected Method[] computeValue(Class<?> type) {
    return  type.getMethods();
    }
};
```

La méthode get retourne les données stockées ou les récupère avec computeValue.

```
var st =cache.get(object.getClass()).entrySet().stream();
```