Projet Citezen

Nos actions et ce que nous avons appris

1. Introduction

Présentation de Citezen et de son utilisation

Citezen : outil pour gérer et générer des bibliographies

Exemple Conversion : Entrée Bib

```
@inproceedings{Gael04c,
  author = {Markus Gaelli and Oscar Nierstrasz and St\'ephane Ducasse},
  title = {One-Method Commands: Linking Methods and Their Tests},
  booktitle = {OOPSLA Workshop on Revival of Dynamic Languages},
  year = {2004},
  pdf = {http://rmod-files.lille.inria.fr/Team/Texts/Papers/Gael04c-LinkingMethodsAndTests.pdf},
  abstract = {Although unit testing is essential for programming, current languages only barely support the developer
  annote = {internationalworkshop},
  keywords = {snf05 scg-pub skip-doi gaelli kzChecking},
  month = oct,
  pdf-second = {http://rmod-files.lille.inria.fr/Team/Texts/Papers/Gael04cLinkingMethodsAndTests.pdf}}
```

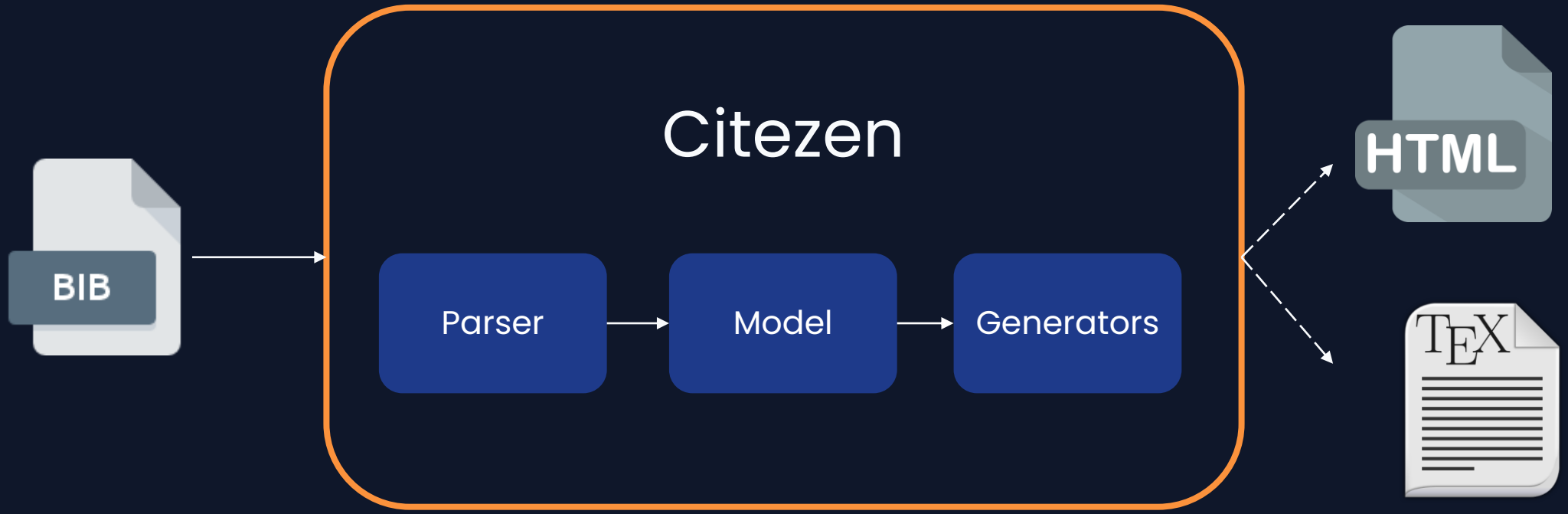
Exemple Conversion : Traitement par Citezen

```
| bibset generator |
bibset := CZBibParser parse: ('rmod.bib' asFileReference) contents.
bibset scope: CZSet standardDefinitions.
generator := CZHTMLGenerator new.
generator save: bibset to: 'rmod-Generated.html'.
```

Exemple Conversion : Sortie HTML

1. Steven Costiou, Mickaël Kerboeuf, Clotilde Toullec, Alain Plantec and Stéphane Ducasse, Object Miners: Acquire, Capture and Replay Objects to Track Elusive Bugs, Journal of O
2. Pablo Tesone, Guillermo Polito, Luc Fabresse, Noury Bouraqadi and Stéphane Ducasse, Preserving Instance State during Refactorings in Live Environments, Future Generation Co
3. Pablo Tesone, Stéphane Ducasse, Guillermo Polito, Luc Fabresse and Noury Bouraqadi, A new modular implementation for Stateful Traits, Science of Computer Programming, Els
4. Guido Chari, Diego Garbervetsky, Stefan Marr and Stéphane Ducasse, Fully Reflective Execution Environments: Virtual Machines for More Flexible Software, Transaction on Soft
5. Pablo Tesone, Guillermo Polito, Luc Fabresse, Noury Bouraqadi and Stéphane Ducasse, Dynamic Software Update from Development to Production, Journal of Object Technology
6. Guillermo Polito, Stéphane Ducasse, Luc Fabresse and Camille Teruel, Scoped Extension Methods in Dynamically-Typed Languages, The Art, Science, and Engineering of Progra
7. Damien Pollet and Stéphane Ducasse, A critical analysis of string APIs: the case of Pharo, Science of Computer Programming, 2017, http://rmod-files.lille.inria.fr/Team/Texts/Pape
8. Guillermo Polito, Luc Fabresse, Noury Bouraqadi and Stéphane Ducasse, Run-Fail-Grow: Creating Tailored Object-Oriented Runtimes, The Journal of Object Technology, 16, Cha
9. Nick Papoulias, Marcus Denker, Stéphane Ducasse and Luc Fabresse, End-User Abstractions for Meta-Control: Reifying the Reflectogram, Science of Computer Programming, 14
10. Clément Béra, Eliot Miranda, Marcus Denker and Stéphane Ducasse, Practical Validation of Bytecode to Bytecode JIT Compiler Dynamic Deoptimization, Journal of Object Techn
11. Vincent Blondeau, Anne Etien, Nicolas Anquetil, Sylvain Cresson, Pascal Croisy and Stéphane Ducasse, Test Case Selection in Industry: An Analysis of Issues Related to Static Ap

Conception Citezen



2. Nos Actions

Notre travail sur le projet

2.a. Lancer le Projet

Comment convertir un fichier bib en HTML

Objectif

- Trouver une utilisation classique du projet

Recherches : README.md

Citezen

Citezen is a bib parser and tool suite.

build failing | coverage unknown | license MIT | Pharo 6.1 | Pharo 7.0 | Pharo 8.0

Tests are all green on my machine. No idea why some are failing on travis

Installing for use

```
$ git clone git@github.com:Ducasse/Citezen.git
$ cd Citezen
$ ./scripts/build.sh
```

Loading for dev

```
Metacello new
   baseline: 'Citezen';
   repository: 'github://Ducasse/Citezen';
   load.
```

Recherches : Commentaires de Classes

Class: CZHTMLGenerator

```
A CZHTMLGenerator is generating nice html for us.
```

visitor bibset

```
bibset := CZBibParser parse: (FileStream readOnlyFileNamed: 'rmod.bib') contents.
bibset scope: CZSet standardDefinitions.
visitor := CZHTMLGenerator new filename: 'rmod-Generated.html'.
visitor visit: bibset.
```

Actions

- Rendre les exemples compatibles avec Pharo 9+
  API FileStream déprécié, utiliser « asFileReference » à la place
- Ajouter un script d’exemple dans le README.md

2.b. Générer un Petit .bib

Une tâche plus difficile que prévue...

Objectifs

- Le .bib d'exemple est trop gros
  10729 lignes, 650 entrées
- Générer un .bib le plus petit possible pour faciliter le travail

Fichier .bib

```
@inproceedings{Anqu20a,
    author = {Nicolas Anquetil and Anne Etien and Houekpetodji, Mahugnon Honor\'e ar
    title = {Modular Moose:  A new generation of software reengineering platform},
    booktitle = {International Conference on Software and Systems Reuse, ICSR2020},
    year = {2020},
    annote = {internationalconference},
    keywords = {kzEvolution},
    month = dec
}
```

Problème

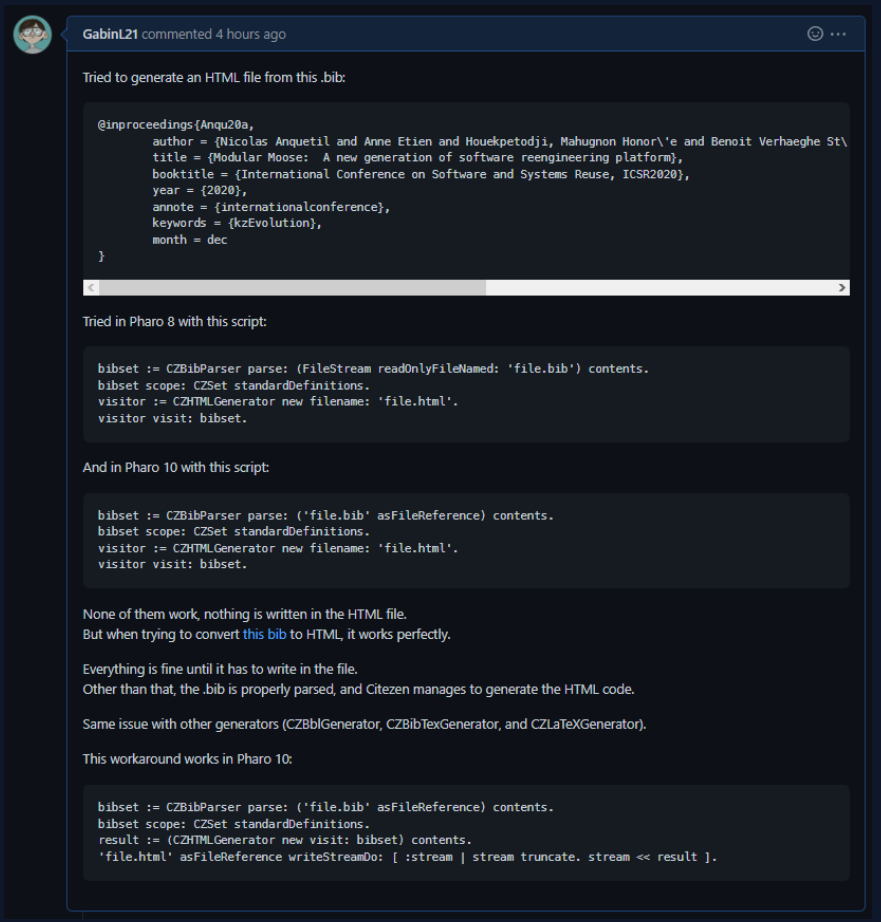
- Problème : les fichiers générés sont vides
- Le problème survient seulement au moment où le Generator écrit le code dans le fichier

Solution

Récupérer le code sous forme de String, puis l'écrire dans le fichier

```
| bibset html |
bibset := CZBibParser parse: ('in.bib' asFileReference)
contents.
bibset scope: CZSet standardDefinitions.
html := (CZHTMLGenerator new visit: bibset) contents.
'out.html' asFileReference writeStreamDo:
  [ :stream | stream truncate. stream << html ].
```

Git Issue



GabinL21 commented 4 hours ago

Tried to generate an HTML file from this .bib:

```
@inproceedings{Anqu20a,
        author = {Nicolas Anquetil and Anne Etien and Houekpetodji, Mahugnon Honor\'e and Benoit Verhaeghe St\
        title = {Modular Moose:  A new generation of software reengineering platform},
        booktitle = {International Conference on Software and Systems Reuse, ICSR2020},
        year = {2020},
        annote = {internationalconference},
        keywords = {kzEvolution},
        month = dec
}
```

Tried in Pharo 8 with this script:

```
bibset := CZBibParser parse: (FileStream readOnlyFileNamed: 'file.bib') contents.
bibset scope: CZSet standardDefinitions.
visitor := CZHTMLGenerator new filename: 'file.html'.
visitor visit: bibset.
```

And in Pharo 10 with this script:

```
bibset := CZBibParser parse: ('file.bib' asFileReference) contents.
bibset scope: CZSet standardDefinitions.
visitor := CZHTMLGenerator new filename: 'file.html'.
visitor visit: bibset.
```

None of them work, nothing is written in the HTML file.
But when trying to convert this bib to HTML, it works perfectly.

Everything is fine until it has to write in the file.
Other than that, the .bib is properly parsed, and Citezen manages to generate the HTML code.

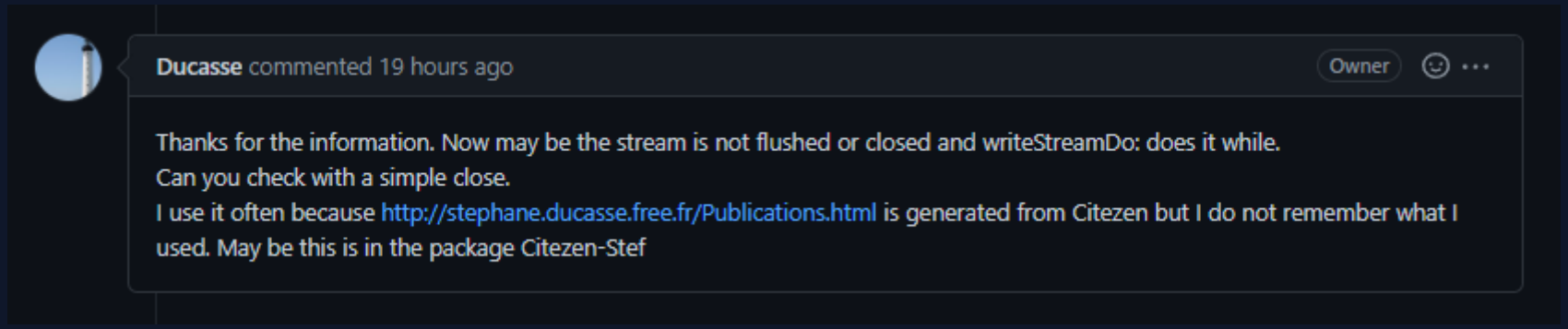
Same issue with other generators (CZBblGenerator, CZBibTexGenerator, and CZLaTeXGenerator).

This workaround works in Pharo 10:

```
bibset := CZBibParser parse: ('file.bib' asFileReference) contents.
bibset scope: CZSet standardDefinitions.
result := (CZHTMLGenerator new visit: bibset) contents.
'file.html' asFileReference writeStreamDo: [ :stream | stream truncate. stream << result ].
```

Commentaire

- Suggestion : la stream n’est pas fermé correctement



Avant le Fix

Avant, 2 envois de message :

- Définit le fichier de sortie
- Visite (génère le contenu)

```
visitor := CZHTMLGenerator new.
visitor filename: 'rmod.bib'.
visitor visit: bibset.
```

Après le Fix

Après, 1 envoi de message :

- Définit le fichier de sortie
- Visite (génère le contenu)
- Ferme le fichier (écrit correctement)

```
visitor := CZHTMLGenerator new.

visitor save: bibset to: 'rmod.bib'.
```

```
save: bibset to: aString

  [ self
    filename: aString;
    visitBibSet: bibset ] ensure: [ self close ]
```

Leçons Apprises

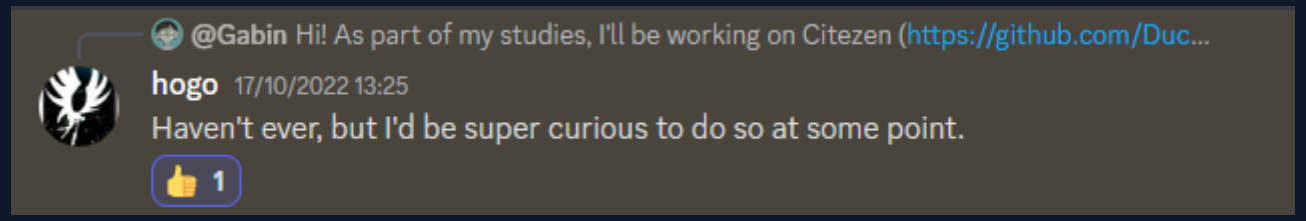
- Nous aurions dû faire la Git Issue plus tôt
- Nous avons tout de même appris à nous servir du debugger et à investiguer un bug

2.c. Retravailler le Checker

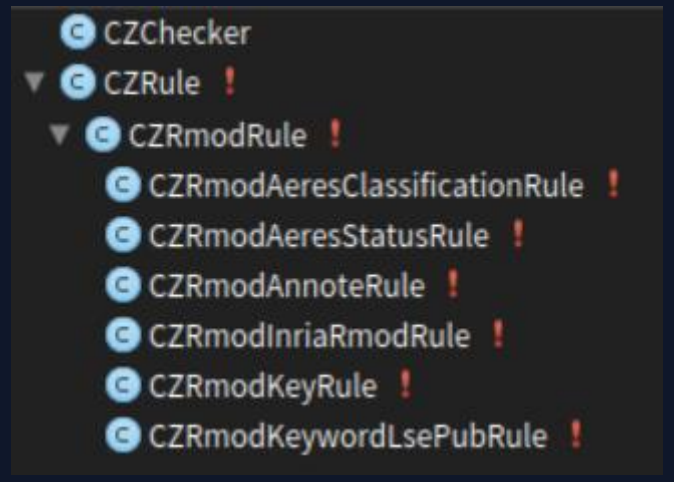
Refonte du package « Citezen-Checking »

Pourquoi le Checker ?

- Volonté de travailler sur du refactoring



Fonctionnement du Checker




```
| checker |
checker := CZChecker new.
checker addRule: CZRmodKeyRule new.
checker checkFile: 'rmod.bib'
```

Avertissement

Class: CZChecker

CZ so ugly and terrible

La Méthode « checks »

```
checks
   | newEntries newFields newMacros toBeDeleted toBeInclude error |

   keys := Dictionary new.

   macros addAll: (CZSet standardDefinitions macros).

   "Don't display that some month macros are defined but not used"
   (CZSet standardDefinitions macros) do: [ :each | macrosUsage at: each key put: true ].

   newEntries := OrderedCollection new.
   macroKeys := macros collect: [ :each | each key ].

   entryKeys := entries collect: [ :each | each key ].

   index := 0.
   entries do: [ :entry |
      toBeInclude := true.
      error := false.

      index := index + 1.

      "checks if the type of the entry is a correct bibtex type"
      (self checkBibtexType: entry)
         ifFalse: [
            error := true.
            toBeInclude := false ].

      newFields := OrderedCollection new.
      fieldKeys := OrderedCollection new.
      entry fields do: [ :field |
         toBeDeleted := false.

         "checks the macros referenced in the entry and removes fields with undefined macros"
         (self checkMacroReferenced: field)
            ifFalse: [ toBeDeleted := true ].

         toBeDeleted ifFalse: [
            "removes empty fields"
            (self checkEmptyField: field)
               ifFalse: [ toBeDeleted := true ] ].

         "removes duplicate fields"
         (self checkDuplicateField: field)
            ifFalse: [ toBeDeleted := true ].

         toBeDeleted
            ifFalse: [ newFields add: field ]
            ifTrue: [ error := true ]
      ].
      entry fields: newFields.

      "checks if the entry contains all the fields needed by it's type"
      ((typeFieldsMatching at: (entry type)) value: (entry fields collect: [ :each | each key ]))
         ifFalse: [
            errorStream nextPutAll: 'Error: Entry does not contain all needed fields'; cr.
            error := true.
            toBeInclude := false ].
```

```
      "checks if the key is unique and changes it if it's not"
      "keys at: entry key put: ((keys at: entry key ifAbsent: 0) + 1).
      ((keys at: entry key) > 1)
         ifTrue: [ | key |
            key := entry key.
            entry key: '###',(entry key, '_',(keys at: entry key) asString,'###' ).
            errorStream nextPutAll: 'Warning: duplicate key: ', key, ' changed to: ', entry key; cr.
      error := true ]."

      "checks if the entry respects all the rules"
      rules do: [ :rule |
         "(rule value value: entry)"
         (rule check: entry)
            ifFalse: [
               errorStream nextPutAll: 'Error : entry does not respect the rule: ', rule explanation; cr.
               error := true.
               toBeInclude := false ] ].

      "displays the errors and creates the collection with safe entries"
      error ifTrue: [
         ((toBeInclude not) | (warnings & toBeInclude))
         ifTrue: [
            errorStream
               nextPut: Character tab;
               nextPutAll: 'for :';
               cr;
               nextPutAll: "assoc value; "

               "(CZBibtexOutputer new entryToBibtexString:" entry  ")";
               cr.
         ].
         toBeInclude
            ifTrue: [
               warnings ifTrue: [
                  newEntries add: entry.
                  errorStream nextPutAll: '(There are only warnings, the entry is not deleted)'; cr; cr ] ]
            ifFalse: [
               errorStream nextPutAll: '(There are errors, the entry is deleted)'; cr; cr ]
      ]
   ].
   entries := newEntries.

   "removes unused macros"
   newMacros := OrderedCollection new.
   macros do: [ :macro |
      (macrosUsage at: macro key)
         ifFalse: [ errorStream nextPutAll: 'Unused macro:'; cr; nextPutAll: macro; cr; cr ]
         ifTrue: [  newMacros add: macro ]
   ].
   macros := newMacros.

   errorStream close.

   Smalltalk at: #Console ifPresent: [ :console | console print: (self errorString copyReplaceAll: String cr with: String crlf) ].

   ^ self errorString
```

Reverse-engineering

- Portée : la méthode checks
- Utilisation :
  - Senders (References et Implementors pas très utiles ici)
  - Debugger

Fonctionnement de « checks »

- Même parser, juste avec un retour des erreurs

- 2 types de règles :
  - Des règles « en dur » dans la méthode
  - Des règles optionnelles ajoutables à la demande

Nettoyer cette Méthode

- Ajouter des tests
- Régler les warnings de Pharo
- Séparer en petits messages

Premières Actions

- Amélioration du code d’exemple dans le commentaire de classe
- Clean du code (warnings Pharo)
- Ajout de quelques tests

Problème

- Séparer les responsabilités est compliqué sans repartir de zéro…

```
Object subclass: #CZChecker
   instanceVariableNames: 'fileContent rules eof errorStream entries typeFieldsMatching keys macros macrosUsage bibtexTypes macroKeys fieldKeys entryKeys index warnings parserClass'
   classVariableNames: ''
   package: 'Citezen-Checking'
```

- Début d’une nouvelle conception « CZChecker2 »

Buts de la nouvelle Conception

- Regrouper le concept de règles
- Séparer les responsabilités
- Améliorer la modélisation des informations remontées (erreurs, warnings, etc.)

« Checking-Reborn »


```
| checker |
checker := CZChecker2 new.
checker addRule: (CZDuplicateKeysRule new).
checker addRule: (CZEmptyFieldRule new).

checker checkFile:
'C:\Users\w123575\Downloads\Citezen\bad.bib'.
```

```
[WARN] "Dias13c" entry key is duplicated
[WARN] "inria" of entry "Dias14b" is empty
[WARN] "hal-id" of entry "Dias14c" is empty
```

CZChecker2

- Analyse le bib set -> CZAnalyzedSet
- Délègue à chaque règle de vérifier le bib set analysé

```
checkAnalyzedBibSet: anAnalyzedBibSet

    rules do: [ :rule | logs add: (rule check: anAnalyzedBibSet) ].
    ^ logs
```

CZRule2

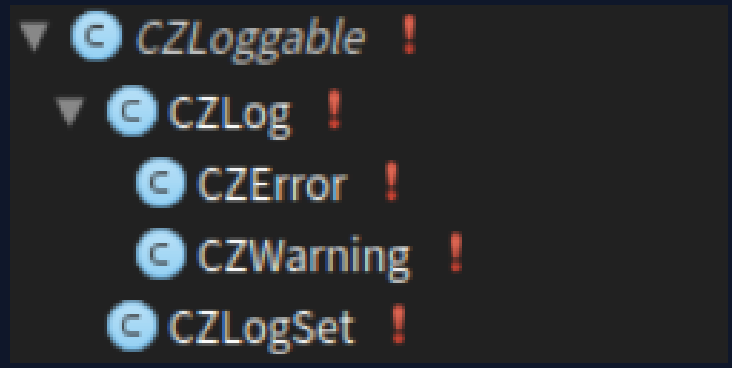
- Design pattern Command
- Exemple de CZDuplicateKeysRule :

```
check: anAnalyzedBibSet

    | entryKeys |
    entryKeys := anAnalyzedBibSet entryKeys.
    entryKeys keysDo: [ :key |
        (entryKeys at: key) > 1
            ifTrue: [ self logEntryKey: key ]].
    ^ logs
```

CZLoggable

- Design pattern Composite
  Utile pour « printOn: »



- Exemple de CZError :

```
printOn: aStream

    aStream nextPutAll: '[ERROR] '.
    super printOn: aStream.
```

Méthodologie

- XTDD
- Commit à chaque « état stable »
- Processus itératif : essayer puis améliorer

Leçons Apprises

- Nous aurions dû passer à une nouvelle conception plus tôt
- Voir/revoir les design patterns Command et Composite
- Se questionner sur de la conception objet

Conclusion

Questions ?