Université de Lille

C3P - **Citezen**

By Gaci Noufel & Saadi Nadine

Université de Lille

Sommaire

- C'est quoi, Citezen ?
- Exemple d'utilisation
- Dépendances du projet
- Packaging du projet
- Rétro-ingénierie
- Améliorations proposées
- Difficultés rencontrés
- Conclusion/Questions

Université de Lille

- **C'est quoi, Citezen?**

Une gamme d'outils pour parser, valider, trier et afficher des références bibliographiques, et qui sert majoritairement à gérer des publications scientifiques telles que des articles, des livres, des thèses de master/doctorat, etc...
Il prend également en charge la génération de page web, de CV, etc ...

- **Exemple d'utilisation**

rmod.bib

Fichier qu'on veut generer en format HTML

C: > Citezen > rmod.bib

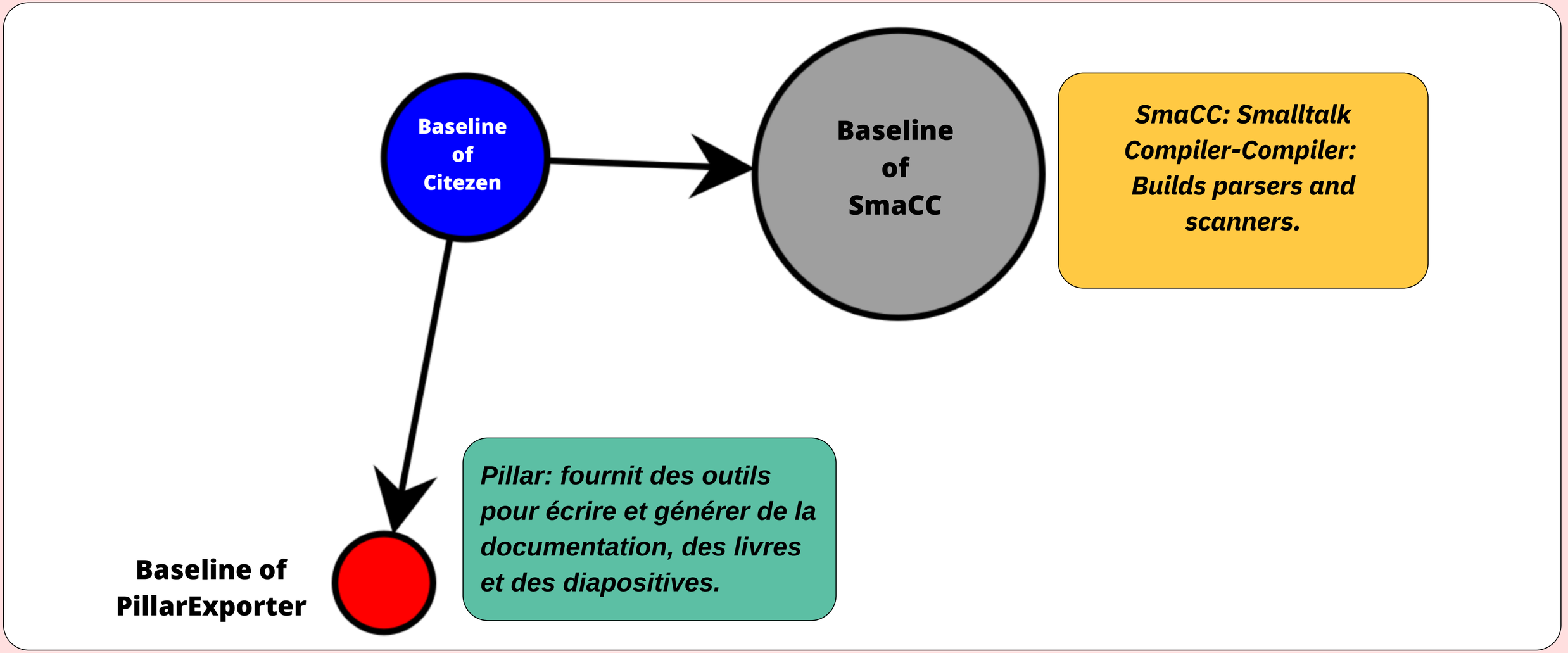
```
1
2    @article{Duca10a,
3                    author = {St\'ephane Ducasse and Damien Pollet},
4                    title = {Fingerprints},
5                    journal = {Journal of Information System},
6                    year = { 2010 }}
```

Script à exécuter

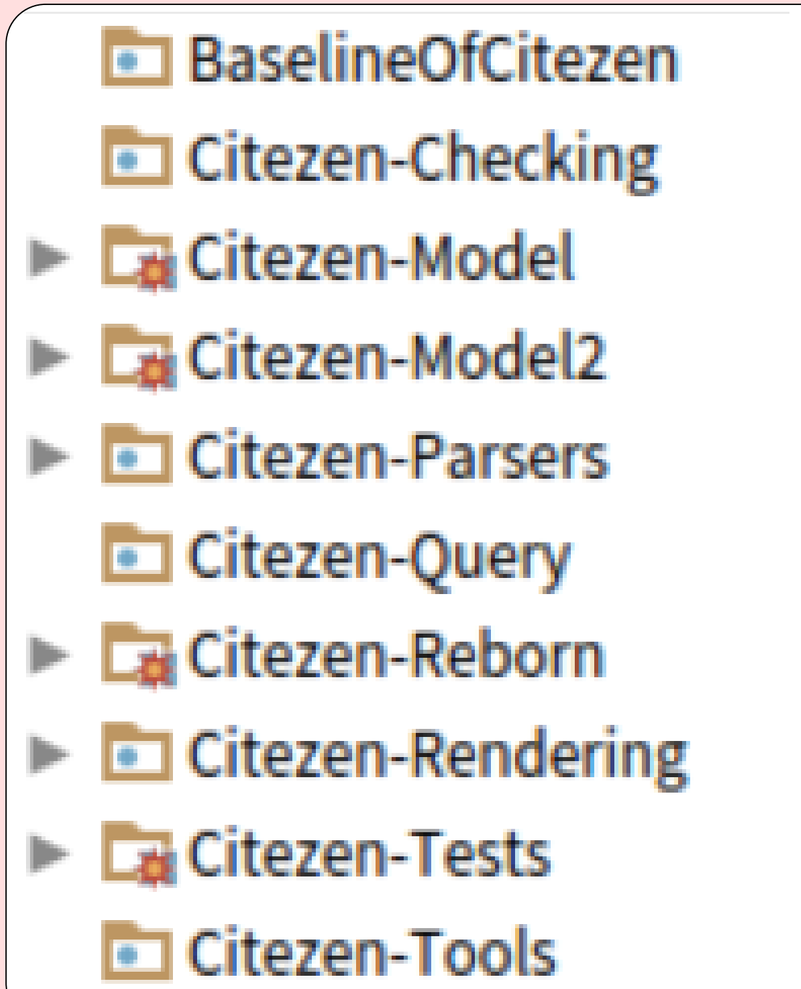
```
| bibset generator |
bibset := CZBibParser parse: ('/Citezen/rmod.bib' asFileReference) contents.
bibset scope: CZSet standardDefinitions.
generator := CZHTMLGenerator new.
generator save: bibset to: '/Citezen/output.html'.
```

On passe le contenu présent dans le fichier rmod.bib en paramètre à la fonction "parse".

- **Dépendances du projet**



- **Packaging du projet**



Citezen-Checking: Effectue des vérifications sur les fichier BibTeX et détecte les anomalies (Erreurs de syntaxe, entrées invalides, champs vides, champs dupliqués etc ...) et définie plusieurs règles à respecter.
Citezen-Model: Définie les champs et entrées bibliographiques.
Citezen-Parsers: Comme son nom l'indique, il contient les différents parsers utilisé dans Citezen.
Citezen-Queries: Crée et définie les différents queries utilisés dans le projet.
Citezen-Reborn: Contient des métadonnées utilisées par différents outils
Citezen-Tests: Contient les classes de tests
Citezen-Tools: Contient les différents outils qui servent à traiter les références bibliographiques.

Université de Lille

Rétro-ingénierie

- *Dépendances du projet (cf. slide n°5).*
- *Classes les plus référencées: CZEntry, CZField.*
- *Classes les plus testées: CZEntry, CZField, différents Parsers*

Nous avons donc cherché:

- *Les implémetations des classes CZEntry et CZField.*
- *Les envois de messages relatifs à ces deux classes.*
- Les références à ces classes-là, notamment les endroits remplis de conditions.

Améliorations proposées

- *Implémenter une nouvelle hiérarchie pour les entrées <CZEntry> (Article, Book, InProceedings, PhDThesis, Techreport ...).*

- *Implémenter une nouvelle hiérarchie pour les champs <CZField> (Author, Title, Year, Note, Publisher ...)*

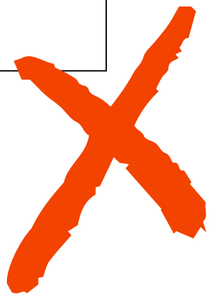
Université de Lille

BEFORE

CZEntries

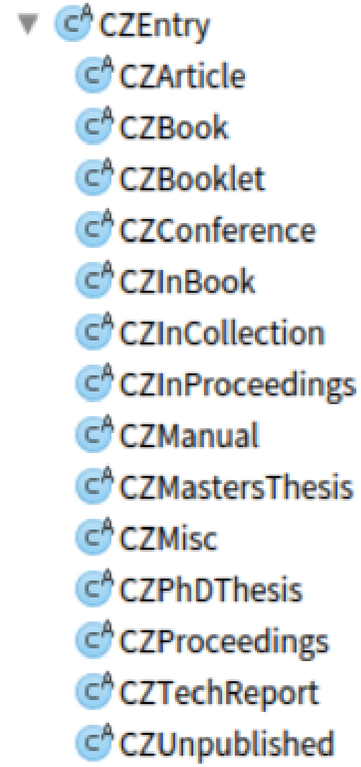
Avant	Problèmes?

• **Les types d'entrées <CZEntry> étaient stockés dans CZSet. Une *SharedPool* nommé CZTypePool qui contient les types d'entrées à aussi été crée mais n'est pour le moment pas utilisée.**
 	
`SharedPool subclass: #CZTypePool`
`  instanceVariableNames: ''`
`  classVariableNames: 'Article Book Booklet Conference Editor InBook InCollection InProceedings Manual MastersThesis Misc PhDThesis Proceedings TechReport Unpublished'`
`  package: 'Citezen-Model'`
	• **A chaque utilisation d'un(e) entrée/champ, il fallait faire plusieurs conditions pour faire les traitements voulus selon le type d'entrée/champ utilisé(e). (Beaucoup d'utilisations de la méthode at:put)**



Université de Lille

AFTER

Aprés	Solution
• On n'a plus besoin de stocker les types d'entrées dans CZSet. • Chaque entrée peut rédiger ses propres spécifications, et l'utilisation de celles-ci ne nécessitera plus de conditions grâce au dispatch/double dispatch. • Il suffit simplement de créer une nouvelle classe si l'on décide d'ajouter un type d'entrée (Il ne sera pas nécessaire de passer par tous les envois de messages et références à CZEntry pour rajouter une condition relative à notre nouvelle entrée).	• Chaque type d'entrée est representé par une sous classe de CZEntry. 

Université de Lille

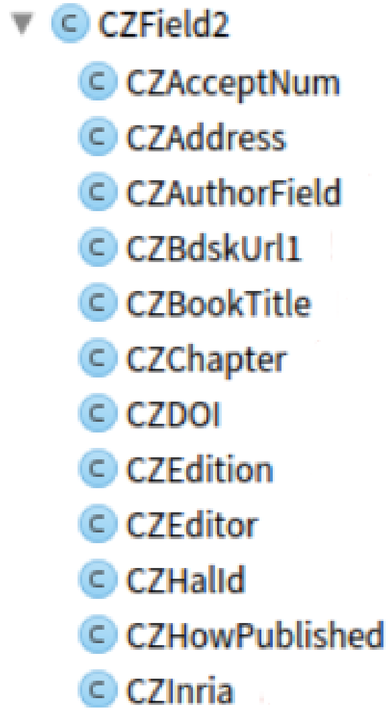
BEFORE

CZFields

Avant	Problèmes?

• **Tous les champs étaient énumérées dans une *SharedPool* nommé CZFieldPool.**
 	
`SharedPool subclass: #CZFieldPool`
`instanceVariableNames: ''`
`classVariableNames: 'Address Annote Author Book BookTitle Chapter DOI DefaultFields Editor HAL HALID HALURL ImpactFactor Institution Journal Misc Month Note Number PDF Pages Publisher School Series Title URL Volume Year'`
`package: 'Citezen-Model'`
	• **A chaque utilisation d'un champ, il fallait faire plusieurs conditions pour faire des traitements selon le type de champ.**
	• **(Exemple extrait:**
	aField key = CZFieldPool volume
	ifTrue: [self outputStream
	nextPutAll: 'Volume '].
	aField key = CZFieldPool pages
	ifTrue: [self outputStream
	nextPutAll: 'pp. '].)

Université de Lille

Aprés	Solution
• On n'a plus besoin de stocker les types d'entrées dans CZFieldPool. • Chaque champ peut rédiger ses propres spécifications, et l'utilisation de ceux-ci ne nécessitera plus de conditions grâce au dispatch/double dispatch. • Il suffit simplement de créer une nouvelle classe si l'on décide d'ajouter un type de champ (Il ne sera pas nécessaire de passer par tous les envois de messages et références à CZField pour rajouter une condition relative à notre nouvelle entrée).	• Chaque type d'entrée est representé par une sous classe de CZFields. 

Exemple de nouvelle CZEntry: CZArticle

Avant (implémentation dans CZEntry)	Après (CZArticle)

```
initializeNeededFields
    "no used for now."
    "self initializeNeededFields"
    neededFieldsDictionary := Dictionary new.
    "this is super boring!!!!
    With a nice hierarchy each entry would declare its fields an
    neededFieldsDictionary
        at: Article put: {
            CZFieldPool author.
            CZFieldPool title.
            CZFieldPool journal.
            CZFieldPool year. };
        at: Book put: {
            CZFieldPool author .
            CZFieldPool editor .
            CZFieldPool title .
            CZFieldPool publisher .
            CZFieldPool year. };
        at: Booklet put: { CZFieldPool title };
        at: Conference put: {
            CZFieldPool author.
            CZFieldPool title.
            CZFieldPool booktitle.
            CZFieldPool year. };
        at: InBook put: {
            CZFieldPool author.
            CZFieldPool editor.
            CZFieldPool title.
            CZFieldPool chapter.
            CZFieldPool pages.
            CZFieldPool publisher.
```

```
initializeNeededFields

    self neededFields: #(author title journal year)
```

Pareil pour les autres entrées avec leurs propres champs obligatoires

Exemple de nouvelle CZEntry: CZArticle

Avant (implémentation dans CZEntry)

```
initializeOptionalFields

optionalFieldsDictionary := Dictionary new.
   optionalFieldsDictionary
      at: Article put: #(volume number pages month note key), self
extendedFieldKeys;
      at: Book put: #(volume series address edition month note key), self
extendedFieldKeys;
      at: Booklet put: #(author howpublished address month year note key)
self extendedFieldKeys;
      at: Conference put: #(editor pages organization publisher address
month note key), self extendedFieldKeys;
      at: InBook put: #(volume series address edition month note key),
self extendedFieldKeys;
      at: InCollection put: #(editor pages organization publisher address
month note key), self extendedFieldKeys;
      at: InProceedings put: #(editor series pages organization publisher
address month note key), self extendedFieldKeys;
      at: Manual put: #(author organization address edition month year
note key), self extendedFieldKeys;
      at: MastersThesis put: #(address month note key), self
extendedFieldKeys;
      at: Misc put: #(author title howpublished month year note key), sel
extendedFieldKeys;
      at: PhDThesis  put: #(aaddress month note key), self
extendedFieldKeys;
      at: Proceedings put: #(editor publisher organization address month
note key), self extendedFieldKeys;
      at: TechReport put: #(type number address month note key), self
extendedFieldKeys;
```

Après (CZArticle)

```
initializeOptionalFields

   self optionalFields: #(volume number pages month note key), self extendedFieldKeys
```

Pareil pour les autres entrées avec leurs propres champs optionnels

Toutes les sous-classes de CZEntry peuvent bénéficier des deux méthodes communes à toutes : ***initializeFieldKeysToRemove, initializeExtendedFieldKeys*** *et definir chacune leurs propres méthodes :* ***initializeNeededFields, initializeOptionalFields***

Template Method Supremacy !

CZEntry

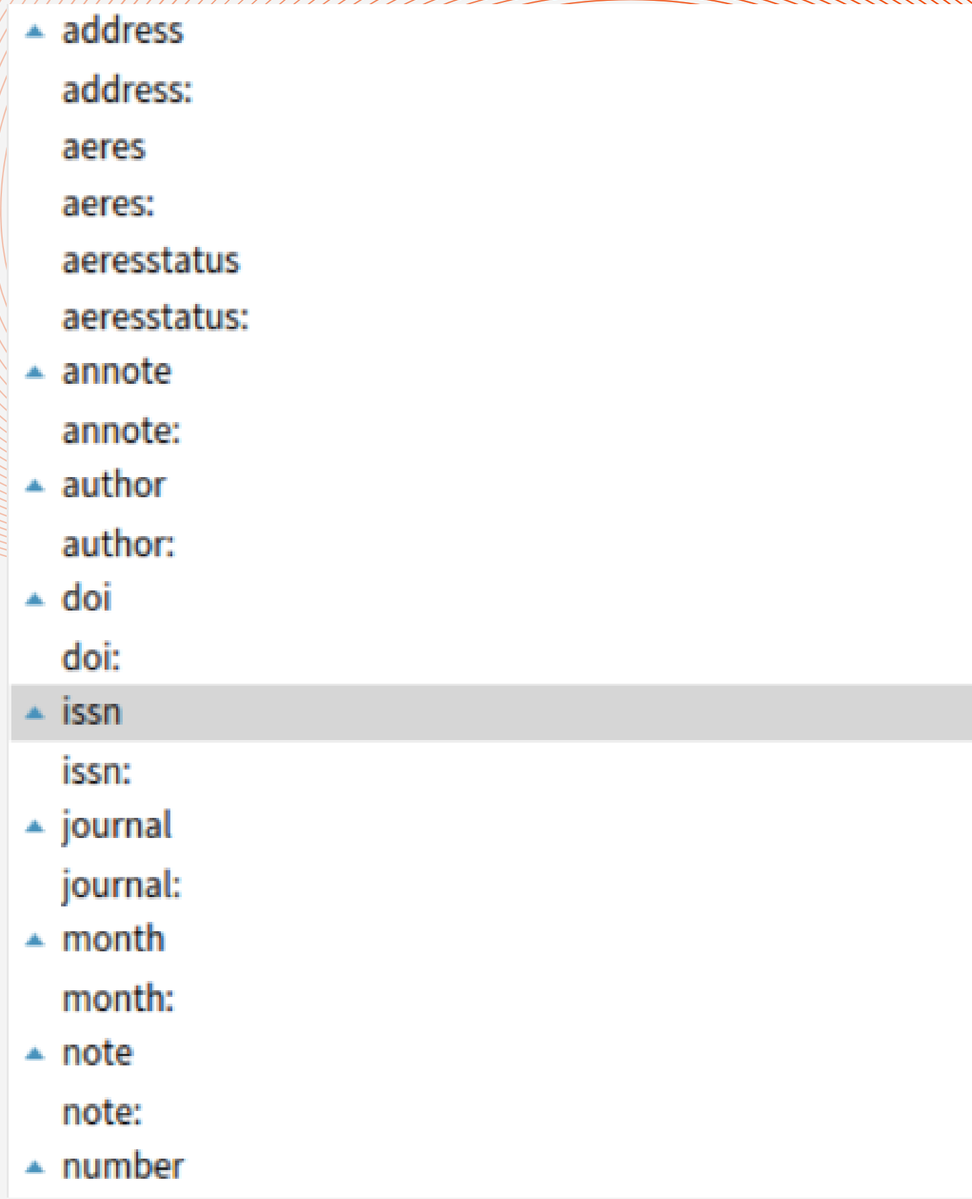
```
initialize
    "self initialize"
    self initializeFieldKeysToRemove.
    self initializeExtendedFieldKeys.
    self initializeNeededFields.
    self initializeOptionalFields
```

Nous avons également généré les accesseurs de chaque entrée, qui représentent majoritairement les champs (CZFields) de chacune d'entre elles

Exemple avec CZArticle

```
CZEntry subclass: #CZArticle
   instanceVariableNames: 'author journal month note number pages title
volume year Aeres Aeresstatus Annote issn doi publisher address'
   classVariableNames: 'neededFields optionalFields'
   package: 'Citezen-Model'
```

Longue vie au *Ctrl+Shift+N qui nous a beaucoup aidé dans la recherche des différents champs de chaque entrée.*



...

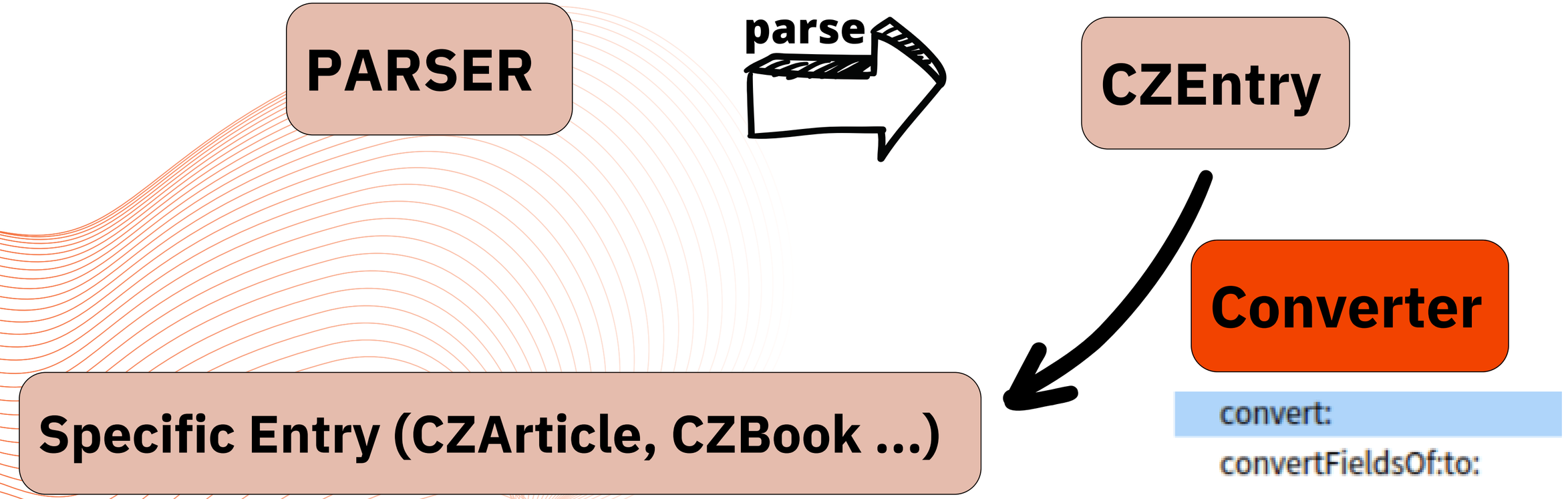
Exemple de changements après l'implémentation de la hiérarchie

```
entry := (CZEntry type: CZEntry article)
   key: #Duca10a;
   at: #author
      put: 'St\''ephane Ducasse and Damien Pollet';
   at: CZFieldPool title put: 'Fingerprints';
   at: CZFieldPool journal
      put: 'Journal of Information System';
   at: CZFieldPool year put: '2010';
   yourself.
entry2 := (CZEntry
   type: CZEntry inproceedings)
   key: #Abde08a;
   at: #author
      put:
      'Hani Abdeen and Ilham Alloui and St\''ephane Ducasse and Damien Pollet and Mathieu Suen';
   at: CZFieldPool title
      put:
      'Package Reference Fingerprint: a Rich and Compact Visualization to Understand Package Relationships';
   at: #Aeres put: 'ACT';
   at: #Aeresstatus put: 'aeres08';
   at: #Annote
      put: 'internationalconference stefPub';
   at: CZFieldPool booktitle
      put:
      'European Conference on Software Maintenance and Reengineering (CSMR)';
   at: #Inria put: 'ADAM';
   at: #Keywords put: 'moose-pub';
   at: #'Hal-Id' put: 'hal-1234';
   at: #Location put: 'Athens, Greece';
   at: #Misc
```

Exemple de changements après l'implémentation de la hiérarchie

```
entry := (CZArticle new)
    key: #Duca10a;
    author: 'St\''ephane Ducasse and Damien Pollet';
    title: 'Fingerprints';
    journal: 'Journal of Information System';
    year: '2010';
    yourself.
entry2 := (CZInProceedings new)
    key: #Abde08a;
    author: 'Hani Abdeen and Ilham Alloui and St\''ephane Ducasse and Damien Pollet and Mathieu Suen';
    title:'Package Reference Fingerprint: a Rich and Compact Visualization to Understand Package Relationships';
    aeres: 'ACT';
    aeresstatus: 'aeres08';
    annote: 'internationalconference stefPub';
    booktitle: 'European Conference on Software Maintenance and Reengineering (CSMR)';
    inria: 'ADAM';
    keywords: 'moose-pub';
    halid: 'hal-1234';
    location: 'Athens, Greece';
    misc: 'Acceptance rate: 24/87 = 27\%';
    acceptTotal: 87;
    acceptNum: 24;
    pages: '213--222';
    publisher: 'IEEE Computer Society Press';
    rate: '27%';
    selectif: 'oui';
    url: 'http://scg.unibe.ch/archive/external/Abde08a.pdf';
    year: 2008;
    bdskurl1: 'http://scg.unibe.ch/archive/external/Abde08a.pdf'.
```

Petit aperçu du Converter



Quelques tests sur le Converter ✓

setUp

```
   articleEntry := (CZBibParser
      parse:
         '  @article{Herm00a,
 author = {Herman, Ivan and Melan\c{c}on, Guy and Marshall, M. Scott},
 title = {Graph Visualization and Navigation in Information Visualization: A Survey},
 journal = {IEEE Transactions on Visualization and Computer Graphics},
 volume = {6},
 number = {1},
 year = {2000},
 issn = {1077-2626},
 pages = {24--43},
 doi = {10.1109/2945.841119},
 publisher = {IEEE Educational Activities Department},
 address = {Piscataway, NJ, USA}
 }') entries first.

   techreportEntry := (CZBibParser
      parse:
         '
@techreport{Dias14b,
   Annote = {technicalReport},
   Author = {Mart\''in Dias and Uquillas G\''{o}mez, Ver\''onica  and Damien Cassou and St\''ephane Ducasse},
   Inria = {RMOD},
   Institution = {INRIA Lille},
   Keywords = {lse-pub},
   Title = {Software Integration Questions: A Quantitative Survey with \&},
   Url = {https://hal.inria.fr/hal-01093496},
   Year = {2014}}
') entries first.
```

Quelques tests sur le Converter ✓

- testConvertArticle
- testConvertBook
- testConvertInProceedings
- testConvertMastersThesis
- testConvertMisc
- testConvertPhDThesis
- testConvertTechReport
- testConvertUnpublished

testConvertArticle

```
| article |
article := (CZEntryConverter new convert: articleEntry).
self assert: article class equals: CZArticle2.
self assert: article author equals: articleEntry author.
self assert: article title equals: articleEntry title.
self assert: article journal equals: articleEntry journal.
self assert: article volume equals: articleEntry volume.
self assert: article number equals: articleEntry number.
self assert: article year equals: articleEntry year.
self assert: article issn equals: articleEntry issn.
self assert: article pages equals: articleEntry pages.
self assert: article doi equals: articleEntry doi.
self assert: article publisher equals: articleEntry publisher.
self assert: article address equals: articleEntry address.
```

Difficultés rencontrés

Bonus : Pull Request

Citezen Project - Nadine and Noufel #27

Open **NadineSS** wants to merge 10 commits into `Ducasse:master` from `NadineSS:master`

Conversation 16 | Commits 10 | Checks 0 | Files changed 1,127

NadineSS commented 5 days ago

Beginning of the implementation of the new hierarchy of entries and fields

- The converter <Entry -> specific entry> seems to work pretty well, some tests are provided.
- The different Entry fields have been created while testing so there might be some fields missing in the new entry classes.
- I still haven't been able to use the fields objects. I tried to edit some tests with the new classes but i have a difficulty in using the fields, as it is done in the parsing method which is redefined in BibParser, but i don't understand the code in it yet.

Université de Lille

Conclusion

N'hésitez pas à me dire si vous voulez des Carambars, j'en ai !
Sinon, **avez-vous des questions ?**