Aide Correction

Delhomme Fabien

20 novembre 2022

1 Objectif

Avoir une interface qui permette de m'aider pour la correction de contrôle.
J'aimerai notamment :

- comptage des points
- statistiques de réussite
- grille de réussite pour chaque question ("propreté", "justification")
- Détection automatique d'erreur systématique chez les élèves, pour les notifier efficacement.
- exporter en csv pour analyser les données dans un tableur.

2 Étape 1

2.1 On définit l'ensemble des élèves dans une variable globale

Nous avons donc le code lisp suivant :

```
(defparameter *classe* '((ABDOUNE Maxence)
                         (AGAMAKOU Axel)
                         (ATTYE Charlotte)
                         (BEMBA Séréna)
                         (BRESSY Jade)
                         (CALEJO Alex)
                         (CHARLES Maeva)
                         (CORREIA Joana)
                         (DE-LUISE Enzo)
                         (DE-PERCIN Kristopher)
                         (FOURDRINIER Thomas)
                         (GALLOY Ronan)
                         (GARRUZZO Rafael)
                         (GERMAIN Willem)
                         (GORSSE Clara)
                         (LABAYE Elodie)
                         (LEOCADIE Enzo)
                         (LEVI-VALENSI Margaux)
                         (MADAOUI Lyma)
                         (MAHROUG Walid)
                         (MARTIN Emma)
                         (MOUROU Aya)
                         (POUMAROUX Alyssia)
                         (PUPILLI Lola)
                         (RACON Kassidi)
                         (RAMTANI Cerina)
                         (RENZI Ambre)
                         (SAHNOUN Halima)
                         (SALEM Ibrahim)
                         (SALLENAVE Helene)
                         (SLIMANI Shirine)
                         (TRAN Matthieu)
                         (TRAORE Saïd)
                         (TSHISELEKA MUTETENU Justino)
                         (VAREILLE Mathieu)
                         (ZEKA Kerene)))
```

`*CLASSE*`

Avoir des cons-listes ou des listes ne changent pas grand chose à mon avis. Autant faire au plus simple avec des listes.

2.2 Exemple

On peut parcourir l'ensemble des élèves par :

```
(loop for (nom prenom) in *classe* do
  (print nom)
  (princ prenom))

(format t "~%La classe comporte ~S élèves" (length *classe*))
```

```
ABDOUNE MAXENCE
AGAMAKOU AXEL
ATTYE CHARLOTTE
BEMBA SÉRÉNA
BRESSY JADE
CALEJO ALEX
CHARLES MAEVA
CORREIA JOANA
DE-LUISE ENZO
DE-PERCIN KRISTOPHER
FOURDRINIER THOMAS
GALLOY RONAN
GARRUZZO RAFAEL
GERMAIN WILLEM
GORSSE CLARA
LABAYE ELODIE
LEOCADIE ENZO
LEVI-VALENSI MARGAUX
MADAOUI LYMA
MAHROUG WALID
MARTIN EMMA
MOUROU AYA
POUMAROUX ALYSSIA
PUPILLI LOLA
RACON KASSIDI
RAMTANI CERINA
RENZI AMBRE
SAHNOUN HALIMA
SALEM IBRAHIM
SALLENAVE HELENE
SLIMANI SHIRINE
TRAN MATTHIEU
TRAORE SAÏD
TSHISELEKA MUTETENU
VAREILLE MATHIEU
ZEKA KERENE
La classe comporte 36 élèves
```

3 Mise en forme des contrôle dans lisp

3.1 Définition d'un contrôle.

Un contrôle est une évaluation qui comporte un certain nombre de **questions**. Chaque question vérifie que l'élève sait faire un certain nombre de **compétences**. Chaque compétences acquises rapporte un certain nombre de **points**.
Plusieurs pistes pour implémenter un contrôle dans lisp :

- La programmation orientée objet.
- Une liste de liste. On peut utiliser des clé voire des tableaux de hashage.

La solution la plus souple me paraît être la deuxième option. Je ne connais pas assez les objets dans lisp pour m'y fier.

3.2 Programmer un exemple de contrôle.

On définit au passage une fonction qui ajoute un contrôle à la liste des contrôles.

```
(defparameter *controles* '())

(defparameter *controle-1* '(
                             (
                                (:enonce "Combien vaut 1 + 1 ?" :competence (:calcul 1
 ↪  :presentation 0.5))
                                (:enonce "Montrer que $\sqrt{2}$ est irrationnel."
 ↪  :competence (:demonstration 2 :proprete 1 :logique 1)))))

(defun ajouter-controle (controle)
    (append controle *controles*))

(defun reset-controles ()
    (setf *controles '()))

(ajouter-controle *controle-1*)
```

3.3 Accéder aux caractéristiques d'un contrôle

On souhaite accéder aux nombres de points de la première question. On définit deux fonctions qui permettent d'accéder aux compétences d'une question d'un contrôle, et une autre qui retourne le nombre de points associés à cette question.
On avait débuté par faire des accès «absolus» aux questions, mais ça sert à rien. Les fonctions acceptent une liste qu'elles comprennent comme une question. Beaucoup plus souple, et pas besoin de traîner des indices partout. On voit ça notamment dans la ligne 23.

```lisp
;; (defun liste-competence-point-question (controle exercice question)
;;   (let ((question-selectionne (nth question (nth exercice controle))))
;;     (getf question-selectionne :competence)))

(defun liste-competences-points-question (question)
  (getf question :competence))

;; (defun nombre-point-question (controle exercice question)
;;   (let ((ensemble-competence (liste-competence-point-question controle exercice
 ↪  question)))
;;     (reduce '+ (remove-if-not 'numberp ensemble-competence))))

(defun nombre-points-question (question)
  "En donnant une question, retourne la somme des points associés aux compétences."
  (reduce '+ (remove-if-not 'numberp question)))

(defun nombre-points-controle (controle)
  (loop
    for exercice in controle
    sum (loop
            for question in exercice
            sum (nombre-points-question (liste-competences-points-question question)))))

;(nombre-point-question *controle-1* 0 0)
(nombre-points-controle *controle-1*)
```

```
5.5
```

3.4 Coder une interface pour ajouter un nouveau contrôle

4 Macro

4.1 Parcourir toutes les questions d'un contrôle

Une idée à peaufiner pour plus tard.
On notera qu'il peut y avoir des soucis à ne pas avoir utiliser `gemsym` pour le nom d'une variable (ici exercice) qui parcourt les exercices.
Pour l'instant le code ne fonctionne pas.

```lisp
(defmacro pour (question controle &body)
    `(loop
        for exercice in ,controle
            do
                (loop
                    for ,question in exercice
                        do
                            &body)))
```

5 Corriger un contrôle

5.1 Proposer une interface qui intervient pour chaque élève.

Pour l'instant je ne sais pas créer une copie du modèle contrôle qui ne contiennent que des compétences initialisée à 0 sans modifier la liste des compétences de chaque question.
J'ai testé de définir une liste des compétences pour chercher parmis elles, mais dans ce cas on crée des propriétés pour celle qui n'y était pas, ce n'est pas ce que l'on veut.
Une autre idée : une plist est une liste. Donc, on peut tout simplement tout parcourir et mettre tous les nombres à 0.
Je suis en train d'implémenter cette idée dans la fonction `reset-nombre-zero` mais c'est pas si évident.

```
;; Il faudrait une copie neuve du contrôle
;; qui représente la copie de l'élève.

(defparameter *liste-competences* '(:calcul :raisonement :logique) "Liste des
 ↪  compétences d'une question.")

(defun reset-nombre-zero (liste)
  (let ((element (car liste)))
    (cond
      ((numberp element)
        (progn
                (setf (car liste) 0)
                (reset-nombre-zero (cdr liste))))
      ((listp element)
        (reset-nombre-zero (car element))))
    liste))

(defun reset-notes (question)
    (loop
        for competence in *liste-competences*
        do
        (let ((champ-competence (getf question :competence)))
            (when (getf champ-competence competence) (setf (getf champ-competence
 ↪  competence) 0))
            (print champ-competence)))
    question)

(defun copie-eleve (controle)
  "Donne une copie viege du contrôle, prete à être corrigée"
  ;; C'est là que les objets aurait pu être utile...
    (loop
        for exercice in controle
            collect (loop
                        for question in exercice
                            (reset-notes (copy-list question)))))




;;On parcourt chaque eleve

(loop
  for eleve in *classe*)
```

```
NIL
```

6 Commandes pour faciliter la correction

6.1 En vrac

— search pour chercher un élève dans la liste, et le sélectionner pour modifier ses notes

— stat pour sortir les statistiques par questions du contrôle

*