

**Haute École LÉONARD DE VINCI  
INSTITUT PAUL LAMBIN  
Section Informatique**

Université Catholique de Louvain

**CLAROLINE : EXPORTATION DES  
PARCOURS PÉDAGOGIQUES AU FORMAT  
SCORM**

Mémoire présenté par  
**TIHON, Amand**  
en vue de l'obtention du diplôme de  
bachelier en informatique

Année académique 2004-2005



# Remerciements

En préambule à la lecture de ce mémoire, je voudrais vous faire partager mes remerciements aux personnes suivantes pour toute l'aide qu'elles ont pu m'apporter :

Pascal PAQUET, administrateur délégué du CERDECAM pour m'avoir accueilli.

Philippe DEKIMPE, mon promoteur, pour son assistance et ses remarques enthousiastes.

Colette DE MUYLDER, pour sa bénéfique opiniâtreté à me trouver un stage qui corresponde à mes attentes.

Merci également à Michel MARCHAND, mon maître de stage, pour tout le champ libre qu'il m'a laissé, peut-être à contre-cœur.

Toute l'équipe des développeurs du projet CLARODOC, nommément, Aurélien VAN HAMME, Matthieu LAURENT, Nicolas LEBLANC, Sébastien PIRAUX et Guillaume LEDERER, pour nos confrontations de points de vue productives et leurs avis éclairés.

Merci encore à l'équipe de développement de Claroline de l'UCL, pour leurs conseils avisés et retours d'impression sur mon code.

Matthieu VANDERDONCK, pour l'aide bien involontaire à la rédaction de ce mémoire.

Merci enfin à tous ceux que j'ai oubliés ou dont je ne connais pas le nom, et qui m'ont apporté leur aide de près ou de loin.

Je voudrais terminer par un merci particulier à Donald KNUTH pour avoir inventé  $\text{\TeX}$  ainsi qu'aux créateurs de  $\text{\LaTeX}$  et de ses nombreux modules, qui ont rendu possible la rédaction de ce mémoire.

# Sommaire

<b>Remerciements</b>	<b>iii</b>
<b>I Introduction</b>	<b>8</b>
<b>1 Claroline</b>	<b>9</b>
1.1 Présentation de la plateforme . . . . .	9
1.1.1 Fonctionnalités de Claroline . . . . .	9
1.1.2 Les parcours pédagogiques . . . . .	10
1.2 Ma mission . . . . .	10
<b>II Présentation du contexte</b>	<b>12</b>
<b>2 Présentation du CERDECAM</b>	<b>13</b>
2.1 Présentation générale . . . . .	13
2.2 Le projet Clarodoc . . . . .	13
<b>3 Claroline</b>	<b>14</b>
3.1 L'environnement de développement . . . . .	14
3.2 Intégration à l'équipe . . . . .	15
<b>III Application</b>	<b>17</b>
<b>4 Calendrier</b>	<b>18</b>
<b>5 Principe du parcours pédagogique</b>	<b>19</b>
5.1 Principes généraux . . . . .	19
5.2 Implémentation dans Claroline . . . . .	20
<b>6 Analyse de la base de données</b>	<b>23</b>
6.1 Vision globale . . . . .	23
6.1.1 Diagramme de la base de données . . . . .	23

## Sommaire

---

6.1.2	Conclusion . . . . .	25
6.2	Documents et liens externes . . . . .	25
6.3	Les exercices . . . . .	25
6.3.1	Les types de questions . . . . .	25
6.3.2	Diagramme . . . . .	26
6.3.3	Conclusion . . . . .	27
<b>7</b>	<b>Étude du format SCORM</b>	<b>28</b>
7.1	L'archive SCORM . . . . .	28
7.2	Dialogue avec le LMS . . . . .	30
<b>8</b>	<b>Réalisation de l'exportation</b>	<b>32</b>
8.1	Exportation des ressources . . . . .	32
8.1.1	Les documents . . . . .	32
8.1.2	Les liens . . . . .	33
8.1.3	Exportation des exercices . . . . .	34
8.1.4	Correction des exercices . . . . .	36
8.2	Cas d'un contenu SCORM existant . . . . .	39
8.3	Création du <i>manifest</i> . . . . .	39
8.3.1	Différences entre SCORM et Claroline . . . . .	41
<b>9</b>	<b>Intégration à la plateforme</b>	<b>42</b>
<b>IV</b>	<b>Conclusion</b>	<b>43</b>
<b>10</b>	<b>Concernant le code</b>	<b>44</b>
10.1	État actuel . . . . .	44
10.2	Regrets . . . . .	44
10.3	Améliorations possibles . . . . .	44
10.3.1	Automatisation des dépendances entre ressources . . . . .	45
10.3.2	Exercices en plusieurs pages . . . . .	45
10.3.3	Corrigé des exercices . . . . .	45
10.3.4	Duplication de code . . . . .	45
<b>11</b>	<b>Évaluation et critique</b>	<b>46</b>
<b>V</b>	<b>Annexes</b>	<b>47</b>
<b>A</b>	<b>Détail des tables utilisées</b>	<b>48</b>
A.1	Parcours pédagogique . . . . .	48
A.2	Exercices . . . . .	49
<b>B</b>	<b>Contenu du CD-ROM</b>	<b>51</b>

## Sommaire

---

<b>C</b>	<b>Code source de l'exportation</b>	<b>52</b>
C.1	scores.js . . . . .	52
C.2	learningPathList.php . . . . .	54
C.3	scormExport.inc.php . . . . .	55
	<b>Bibliographie</b>	<b>56</b>

# Table des figures

2.1	Logo CERDECAM . . . . .	13
5.1	Administration d'un parcours pédagogique . . . . .	21
5.2	Parcours pédagogique vu par l'étudiant . . . . .	21
5.3	Navigation . . . . .	22
6.1	Diagramme simplifié du parcours pédagogique . . . . .	24
6.2	Diagramme des exercices . . . . .	26
7.1	Exemple de fichier <code>imsmanifest.xml</code> . . . . .	29
8.1	Exemple de <i>frames</i> pour l'intégration d'un document . . . . .	33
8.2	Exemple de question à réponse unique . . . . .	35
8.3	Exemple de question à réponses multiples . . . . .	35
8.4	Exemple de question de mise en correspondance . . . . .	35
8.5	Exemple de texte à trous . . . . .	36
8.6	Tableau de correction des textes à trous . . . . .	36
8.7	Aperçu de la fonction <code>CalculateRawScore</code> du fichier <code>scores.js</code> . . . . .	37
8.8	Aperçu de la fonction <code>calcScore</code> des pages d'exercices exportés . . . . .	38
9.1	Administration des parcours pédagogiques . . . . .	42

# Première partie

## Introduction

# Chapitre 1

## Claroline

### 1.1 Présentation de la plateforme

Claroline<sup>1</sup>, dont le nom est issu de *Classroom Online*, est une plateforme pédagogique, destinée à l'enseignement à distance, et plus précisément *via* Internet.

Développé en langage PHP, Claroline est distribué sous la licence GPL<sup>2</sup>, ce qui en fait un Logiciel Libre. Notez l'utilisation ici de majuscules, qui indiqueront au fil de ce document, qu'il est question d'un logiciel dont la licence est considérée comme libre par la Free Software Foundation. Vous trouverez plus d'informations à ce sujet sur <http://www.fsf.org/licensing/licenses>.

Définir Claroline comme une plateforme d'enseignement à distance ne suffit pas à détailler les possibilités du système, car celles-ci sont variables d'un environnement à l'autre.

#### 1.1.1 Fonctionnalités de Claroline

De nombreux systèmes Internet de gestion de cours sont disponibles, que ce soit gratuitement ou non. Voici les principales caractéristiques de celui auquel nous nous intéressons :

- Publication de documents de tous formats, aussi bien textuels que multimédia, voire interactifs.
- Administration de forums de discussions publics ou privés.
- Classement des informations par facultés, puis par cours à l'intérieur d'une faculté, autorisant une gestion fine de l'accès des étudiants aux ressources mises à leur disposition.
- Compositions d'exercices.

---

<sup>1</sup> Site web officiel : <http://www.claroline.net/>

<sup>2</sup> Une copie de la licence est présente sur le CD-ROM en annexe

## Chapitre 1. Claroline

---

- Publication d'annonces et gestion d'un agenda reprenant les tâches en cours et leurs échéances.
- Possibilité pour les étudiants de soumettre leurs travaux.
- Statistiques de fréquentation et de réussite aux exercices.
- Structuration d'un ensemble de documents et d'exercices sous la forme d'un *parcours pédagogique*.
- Import de parcours pédagogiques réalisés sur d'autres plateformes.

La mise en œuvre de la plateforme ne nécessite qu'un hébergement web offrant la possibilité d'exécuter des fichiers PHP, et un système de gestion de bases de données. L'association logicielle typiquement retenue, sans être obligatoire pour autant, est le quartet Linux, Apache, MySQL et PHP, communément surnommé LAMP.

Une fois l'installation terminée, un navigateur suffit à réaliser l'intégralité de l'administration.

### 1.1.2 Les parcours pédagogiques

Il est nécessaire d'explicitier ici un terme utilisé tout au long de ce document, c'est celui de *parcours pédagogique*.

Un parcours pédagogique, dans le jargon de l'enseignement par Internet, se présente comme un ensemble organisé de documents, d'exercices, ou d'autres types de ressources.

Un ensemble, parce qu'il forme un tout, souvent distribué sous la forme d'un fichier `.zip`, même s'il est techniquement possible de récupérer les fichiers qui le composent.

Organisé, parce que le créateur du parcours a donné un ordre aux documents et exercices qui s'y trouvent. Il peut ainsi interdire l'accès à un document tant que l'étudiant n'a pas obtenu une note satisfaisante à un exercice.

Nous reviendrons par la suite plus en détail sur la structure et le fonctionnement des parcours pédagogiques, et sur les particularités de leur implémentation au sein de Claroline.

## 1.2 Ma mission

Lorsque j'ai rejoint l'équipe de développement de Claroline, le logiciel disposait déjà de trois fonctionnalités touchant aux parcours pédagogique :

- l'importation de parcours au format SCORM réalisés à l'aide d'un quelconque autre logiciel ;

## Chapitre 1. Claroline

---

- la création d'un parcours sur la plateforme elle-même, en réutilisant les ressources disponibles ;
- l'édition d'un parcours de manière à ajouter et supprimer des ressources, ou d'en changer l'ordre.

Une fonctionnalité importante manquait à l'appel : la possibilité d'exporter un parcours pédagogique. Ma tâche au sein de l'équipe de développement de Claroline fut d'implémenter cette fonctionnalité.

Deuxième partie  
Présentation du contexte

# Chapitre 2

## Présentation du CERDECAM

### 2.1 Présentation générale

Fondé en 1982, le CERDECAM, Centre de Recherche et de Développement de l'ECAM, promeut la recherche technologique à l'ECAM, aide les entreprises à relever les défis de l'innovation, maintient des liens entre l'ECAM et les industries, et collabore à différents projets de recherche et développement de laboratoires ou d'entreprises industrielles.

Pour ses activités de recherche et développement, le CERDECAM s'appuie sur les compétences des enseignants de l'ECAM, mais la réalisation de projets de grande envergure nécessite l'engagement de personnel permanent.



FIG. 2.1: Logo CERDECAM

### 2.2 Le projet Clarodoc

Le projet Clarodoc, commencé en 2003 et financé par la Région Wallone, dispose d'une équipe de chercheurs au sein du CERDECAM. S'appuyant dans une certaine mesure sur la plateforme Claroline, il vise à la réalisation d'une chaîne de production documentaire complète.

Dans cette équipe que j'ai intégrée durant mon stage, trois personnes travaillent à temps plein sur le moteur de Claroline proprement dit. Deux autres chercheurs s'occupent de ce qui a trait à la production et la gestion documentaire.

# Chapitre 3

## Claroline

Si le développement de Claroline fut initialement lancé par l'Université Catholique de Louvain, de nombreux développeurs y contribuent désormais. Parmi ceux-ci, outre les équipes de l'UCL et du CERDECAM, se retrouvent des professeurs, pédagogues, informaticiens et traducteurs du monde entier, qu'ils soient amateurs ou professionnels.

Aujourd'hui, Claroline est utilisé par plus de 430 organisations réparties à la surface du globe. Rien qu'en Belgique, l'on dénombre une quarantaine de campus virtuels<sup>1</sup>.

### 3.1 L'environnement de développement

Comme détaillé plus haut, la quasi totalité du développement de Claroline est réalisée par deux équipes géographiquement distantes. Cette situation nécessite l'utilisation de plusieurs outils, couramment mis en œuvre dans le cadre des Logiciels Libres.

**Accès au CVS** Lorsque plusieurs personnes travaillent sur une même application, un système de gestion des révisions est indispensable pour aider à la résolution d'éventuels conflits lors de l'édition du code.

Seuls les développeurs des deux équipes permanentes y ont un accès en écriture. Les personnes extérieures souhaitant inclure leurs améliorations et patches doivent alors les contacter.

De plus, il existe une liste de diffusion qui se charge de relayer par courriel toute mise à jour effectuée sur le dépôt CVS. Son accès est lui aussi réservé.

---

<sup>1</sup> Chiffres tirés de la section *Worldwide* du site officiel de Claroline : <http://www.claroline.net/worldwide.htm>

## Chapitre 3. Claroline

---

**Canal IRC** Le projet Claroline dispose d'un canal IRC<sup>1</sup> sur le principal réseau consacré aux Logiciels Libres, #claroline sur irc.freenode.net.

C'est le moyen de communication le plus utilisé par les équipes de développement. Son caractère instantané et multi-utilisateurs le rend particulièrement adapté à la situation et permet d'éviter de trop nombreuses réunions de coordination.

Les utilisateurs finaux pourront y trouver un support à l'installation et à la configuration de Claroline.

**Documentation** Il existe deux documentations bien distinctes, toutes deux basées sur un Wiki, principe de site collaboratif où tout visiteur peut éditer ou rédiger des pages. Le logiciel utilisé à cet effet est Mediawiki<sup>2</sup>.

La première documentation est à destination des utilisateurs. De celle-ci, peu à dire si ce n'est qu'elle est ordonnée, et régulièrement mise à jour.

La seconde documentation, destinée aux développeurs, alterne de rapides prises de notes avec des *roadmaps* personnelles, et des comptes rendus de réunion avec des liens vers d'autres outils du monde de l'e-learning.

**Un forum** Principalement destiné aux utilisateurs qui ont à installer ou administrer une plateforme Claroline.

Il est suprenant de ne pas trouver dans cette énumération, la traditionnelle liste de diffusion, système si répandu dans les communautés du Libre. Il apparaît cependant que forum et IRC suffisent à combler tout besoin de communication, tant entre développeurs qu'avec les utilisateurs.

### 3.2 Intégration à l'équipe

L'intégration à l'équipe du CERDECAM n'a pas posé le moindre problème. Les chercheurs y sont âgés de 20 à 30 ans et formaient déjà à mon arrivée une équipe soudée, dynamique et sympathique.

Toute liberté m'était laissée concernant le système d'exploitation à utiliser, et mon choix s'est tout naturellement porté sur la distribution GNU/Linux Debian<sup>3</sup>, à laquelle je suis habitué depuis de nombreuses années.

Aucun problème n'a non plus été rencontré en ce qui concerne les applications et le cycle de développement. Étant donné mon implication par le passé dans divers autres projets libres, j'y ai rapidement trouvé mes marques, ayant déjà eu l'occasion d'utiliser ces outils.

---

<sup>1</sup>IRC : Internet Relay Chat, protocole d'échange de messages, organisé en salons de discussion

<sup>2</sup>Moteur Wiki réalisé par l'équipe de Wikipedia, l'encyclopédie libre

<sup>3</sup><http://www.debian.org>

### Chapitre 3. Claroline

---

Néanmoins, la sortie d'une nouvelle version stable de Claroline était planifiée pour le courant de mon stage, ce qui m'a permis de découvrir une facette de CVS que je n'avais jamais eu l'occasion d'utiliser.

Afin de ne pas perturber les développements en cours, il a été décidé que mon travail serait effectué sur une branche séparée du dépôt CVS. Ce n'est qu'à la fin du stage que cette branche fut fusionnée au tronc, prête à être intégrée à la version stable suivante.

# Troisième partie

## Application

# Chapitre 4

## Calendrier

Vous trouverez ici un calendrier reprenant succinctement les grandes lignes de l'avancée du travail au long de mon stage.

**Le mois de février.** Ayant pour objectif de me familiariser avec le fonctionnement des exercices et les structures associées, j'ai réalisé ces premières semaines une fonction d'export des exercices et des questions au format IMS-QTI<sup>1</sup>, un format purement XML spécifiquement étudié pour les tests assistés par ordinateur.

**Trois premières semaines de mars.** Ce temps a été nécessaires à mon analyse approfondie du fonctionnement de SCORM.

Je voudrais préciser ici que, bien que très largement répandu, ce format est en réalité peu documenté, en dehors des textes « officiels » ardues et souvent trop touffus pour l'utilisation que j'en avais. Pour l'essentiel de mon travail, je me suis basé sur le livre *Cooking up a SCORM*<sup>2</sup> qui décrit différentes méthodes utilisées lors de la création de contenu SCORM.

**Fin mars, début avril.** Réalisation de l'exportation de parcours pédagogique ne contenant que des documents.

**Avril, début mai.** Exportation des exercices. Cette partie fut assurément la plus longue.

**Fin du stage.** Les derniers moments du stage ont été consacrés à la finalisation et à la correction des derniers bugs.

---

<sup>1</sup>Plus d'informations à ce sujet sur le site officiel : [http://www.imsglobal.org/question/quiv1p2/imsqti\\_oviewv1p2.html](http://www.imsglobal.org/question/quiv1p2/imsqti_oviewv1p2.html)

<sup>2</sup>Voir bibliographie en fin d'ouvrage

# Chapitre 5

## Principe du parcours pédagogique

Avant d'aller plus loin dans l'analyse du problème et de ses solutions, il est souhaitable de s'attarder sur le principe général du parcours pédagogique, et sur la manière dont il a été réalisé sous Claroline.

### 5.1 Principes généraux

Nous avons déjà entrevu dans les grandes lignes, durant l'introduction, les bases d'un parcours pédagogique et principes y relatifs. Pour correctement appréhender ce qui suivra, nous devons ici en approfondir notre étude. Au passage, nous définirons les termes spécifiques couramment utilisés dans ce contexte. Il est à noter que certains d'entre eux sont en langue anglaise, n'ayant pas d'équivalents francisés généralement acceptés.

Un parcours pédagogique est constitué de plusieurs fichiers, ou ressources. Toute information à destination directe de l'étudiant est une ressource, que ce soit la page HTML qu'il visionne, ou la feuille de style CSS utilisée pour mettre la page en forme. N'entrent pas dans cette catégorie les informations structurelles du parcours pédagogique, comme le fichier *manifest* utilisé par le format SCORM que nous détaillerons plus loin.

La structuration du parcours est réalisée par le biais de modules, constitués d'une ou plusieurs ressources formant un tout. Pour reprendre l'exemple de la page HTML, le module correspondant contiendrait en outre la feuille de style et les images nécessaires.

Ces modules peuvent ensuite être organisés de manière hiérarchique ou linéaire au sein du parcours.

## Chapitre 5. Principe du parcours pédagogique

---

Chaque module est capable d'interagir avec le *LMS*<sup>1</sup> pour lui signaler les avancées de l'étudiant. Il est dès lors possible de savoir quels sont les documents qui ont déjà été lus en totalité ou en partie.

Concernant les exercices, ces derniers sont habituellement de type à choix multiples, leur correction étant assumée par l'ordinateur.

Ce système de suivi de l'avancement permet la mise en place d'une gestion des prérequis nécessaires pour l'accès à un module, comme par exemple la lecture préalable d'un document donné et l'obtention d'une note supérieure ou égale à 12/20 à un exercice.

### 5.2 Implémentation dans Claroline

Claroline intègre tous les outils nécessaires à l'importation et à l'édition de parcours pédagogiques. La **figure 5.1** nous montre une capture d'écran de l'interface d'édition d'un parcours, telle que vue par l'administrateur du cours.

Nous y découvrons rapidement plusieurs choses :

- La structure hiérarchique du parcours, décomposé pour l'exemple en parties et en sections.
- Les différents types de modules sont différenciés (exercices, documents HTML ou PDF, liens internet, ...)
- Un module, celui des exercices, est indiqué comme étant bloquant. Cela signifie que le suivant ne sera accessible à l'étudiant que lorsqu'il aura correctement répondu aux exercices.
- Des liens sont prévus pour intégrer rapidement au parcours des ressources préalablement ajoutée sur la plateforme.

La **figure 5.2** représente la même page, mais telle qu'elle est vue cette fois par l'étudiant.

L'élève y a une vision rapide de sa progression dans le parcours pédagogique. Étant donné que la capture d'écran rend invisible cette information, il est utile de signaler que si les trois premiers modules sont accessibles, le dernier ne l'est pas. Il ne le deviendra que lorsque une note suffisante aura été obtenue aux exercices.

---

<sup>1</sup>Learning Management System, ou système de gestion de cours, terme qui reviendra de manière récurrente par la suite

## Chapitre 5. Principe du parcours pédagogique

Mode de vue : Etudiant | **Gestionnaire de cours**

**Parcours pédagogique**

**Titre du parcours pédagogique**  
✎

Description du parcours pédagogique  
✎ ✖

Utiliser un document | Utiliser un exercice | Utiliser un module de ce cours | Créer un titre

Module	Modifier	supprimer	Bloquer	Visibilité	Déplacer	Ordre
<b>Première partie</b>	✎	✖		👁	📄↔📄	▼
📄 Document_1.htm	✎	✖	⬇	👁	📄↔📄	▼
🔗 Lien_vers_site_externe.url	✎	✖	⬇	👁	📄↔📄	▲ ▼
<input checked="" type="checkbox"/> Exemple d'exercice	✎	✖	🚫	👁	📄↔📄	▲
<b>Seconde partie</b>	✎	✖		👁	📄↔📄	▲
<b>Section 1</b>	✎	✖		👁	📄↔📄	▼
📄 autre_document.pdf	✎	✖	⬇	👁	📄↔📄	
<b>Section 2</b>	✎	✖		👁	📄↔📄	▲

FIG. 5.1: Administration d'un parcours pédagogique

Module	Progression
<b>Première partie</b>	
📄 Document_1.htm	██████████ 100%
🔗 Lien_vers_site_externe.url	██████████ 0%
<input checked="" type="checkbox"/> Exemple d'exercice	██████████ 0%
<b>Seconde partie</b>	
<b>Section 1</b>	
📄 autre_document.pdf	██████████ 0%
<b>Section 2</b>	
Progression du parcours pédagogique :	███ 25%

FIG. 5.2: Parcours pédagogique vu par l'étudiant

## Chapitre 5. Principe du parcours pédagogique

---

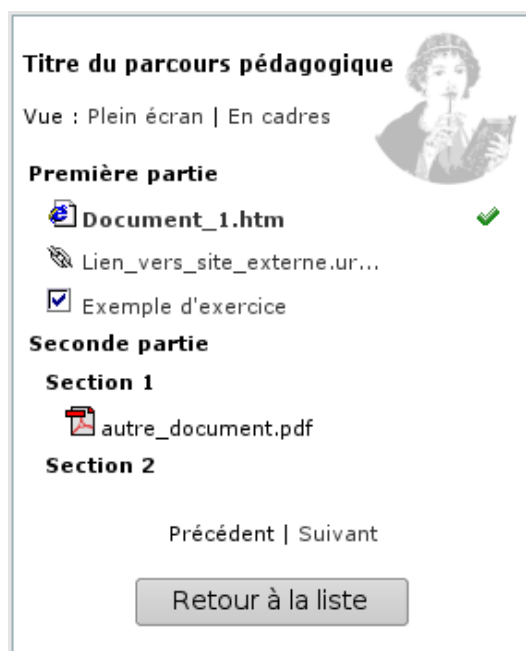


FIG. 5.3: Navigation

Tout au long de sa progression au sein du parcours pédagogique, une boîte de navigation est à disposition immédiate de l'étudiant, sur la partie gauche de la fenêtre de son navigateur. Celle-ci lui permet de naviguer aisément d'un module à l'autre. Les *coches* vertes indiquent les modules terminés avec succès.

Notez que, comme sur la **figure 5.2**, l'intitulé du document PDF n'est pas un lien.

# Chapitre 6

## Analyse de la base de données

### 6.1 Vision globale

Ma première réaction au moment de commencer l'analyse des structures de données fut de tenter d'imaginer une version simplifiée de la base de données qui pourrait se trouver derrière Claroline. Quelques heures d'utilisation assidue de la plateforme m'en ont donné une vision, certes simpliste, mais largement plus intuitive que ne l'aurait permis une étude théorique de la base de données.

Il va de soi que cela ne suffit pas, et de loin, à appréhender toute la complexité de la base de données. Cette étude « pratique » a rapidement été accompagnée par de fréquentes investigations dans la structure elle-même.

Les grandes lignes sont très vite apparues lors de cette première analyse. Examinons-les point par point.

- Les modules repris dans un parcours pédagogique sont issus des documents, liens et exercices existants.
- L'analyse de la base de données nous apprend que les titres sont également des modules.
- Nous déduisons du dernier point qu'une relation parent-enfant existe entre les modules pour réaliser une hiérarchie.

#### 6.1.1 Diagramme de la base de données

Nous ne nous intéressons pour le moment qu'à l'organisation du parcours pédagogique, aussi appelé *Learning path* en anglais.

Pour une meilleure lisibilité, ce diagramme a été simplifié et ne comporte que les relations qui en donnent la structure générale.

Détaillons le contenu utile des différentes tables.

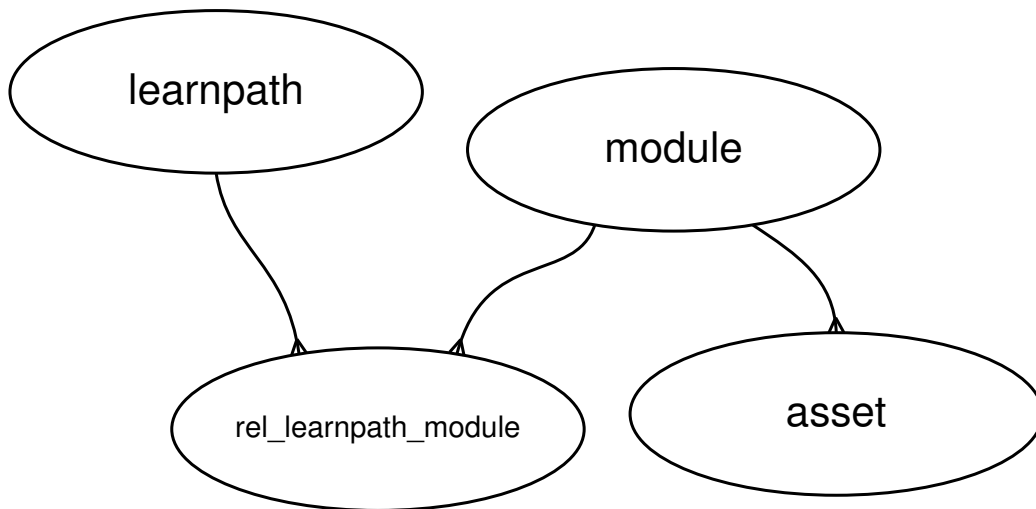


FIG. 6.1: Diagramme simplifié du parcours pédagogique

**learnpath** La table principale identifiant les parcours pédagogiques. Il y a peu à en dire.

**asset** Autre table simple, dont les champs importants sont d'une part la référence au module, et d'autre part le champ **path**. Celui-ci est soit un chemin d'accès au fichier proprement dit, soit une référence à un exercice.

**module** Cette table contient, outre les noms et description sommaire des modules, deux champs dignes d'intérêt. Le premier, **contentType** nous indique le type de ressource, à savoir un document, un exercice ou encore un titre.

Le second est le champ **startAsset\_id** qui indique, en prévision du cas où un module serait constitué de plusieurs assets, par lequel commencer. Les parcours créés au sein de Claroline ne permettent pas cette particularité, mais des parcours créés avec d'autres outils avant d'être importés sur la plateforme en sont capables.

**rel\_learnpaath\_module** est plus qu'une classique table de jointure. C'est elle qui détermine si un module est bloquant et s'il est visible. C'est ici aussi, grâce au champ **rank** qu'est spécifié l'ordre des modules. Nous retrouvons également la mise en place de la structure arborescente avec le champ **parent** qui référence un module. Enfin, le champ **raw\_to\_pass**, nécessaire uniquement pour les exercices, indique la note minimum à obtenir pour que l'exercice soit considéré comme réussi.

### 6.1.2 Conclusion

En résumé, les trois tables **learnpath**, **module** et **rel\_learnpath\_module** donnent la structure du parcours pédagogique tandis que la table **asset** contient les références aux matériaux du parcours.

## 6.2 Documents et liens externes

Le cas des documents et des liens est très facile à traiter. Le champ **path** de la table **asset** contient simplement un chemin vers le document présent sur le disque dur. Les liens ne sont rien de plus que des documents particuliers au format HTML, provoquant une redirection vers l'URL demandée.

## 6.3 Les exercices

L'analyse des tables relatives aux exercices fut assez rapide, vu leur relative simplicité.

C'est également une section de Claroline réalisée par un étudiant stagiaire, et le style de programmation est notablement différent du reste.

C'est l'une des rares parties de la plateforme dont le code est totalement orienté objet. Les classes mises à disposition reflètent très fort la structure des tables dans la base de données et leur organisation.

### 6.3.1 Les types de questions

Claroline propose quatre types de questions, dont trois sont une extension du principe du choix multiple.

1. Le choix multiple à réponse unique, une seule réponse possible parmi plusieurs.
2. Le choix multiple à réponses multiples, similaire au précédent mais pour lequel il est possible de sélectionner plusieurs réponses.
3. La mise en correspondance, où les éléments de deux listes doivent être reliés entre eux, par exemple une liste de villes et une de pays, où l'étudiant appliquera la relation « est située dans le pays. »
4. Le texte à trous, où l'élève se voit présenter un texte dont il doit remplir les blancs.

Pour chacun de ces types de questions, chaque réponse rapporte un nombre définissable de points.

### 6.3.2 Diagramme

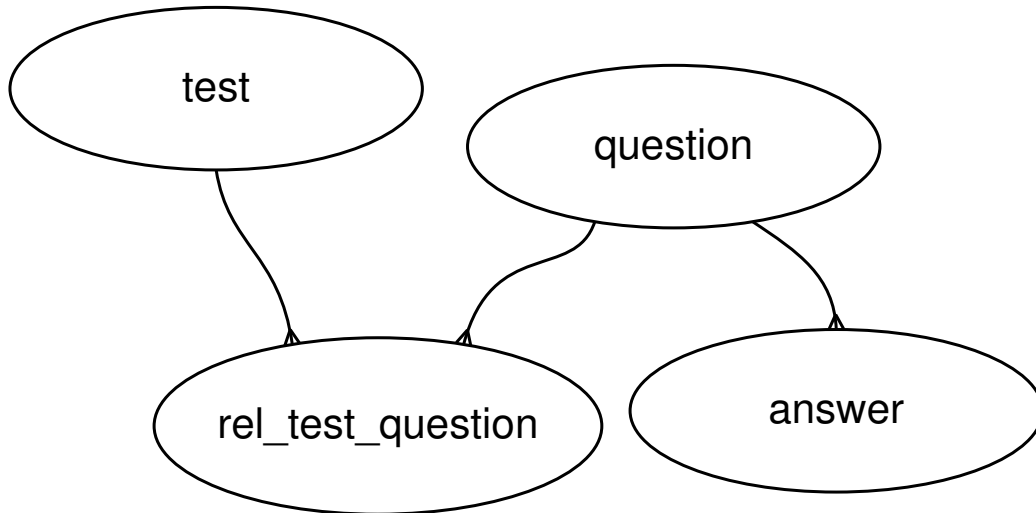


FIG. 6.2: Diagramme des exercices

Voyons le contenu de ces quatre tables dans le détail.

**test** Est la référence générale à un exercice. Cette table contient de nombreux champs décrivant les propriétés de l'exercice, il n'a pas été possible de tous les respecter lors de l'exportation SCORM.

**random** indique, s'il est positif, un nombre de question à sélectionner au hasard parmi celles constituant l'exercice.

**max\_time** permet de spécifier une durée limite pour la résolution de l'exercice.

**max\_attempt** peut limiter le nombre de tentatives de l'élève.

**start\_date** et

**end\_date** précisent le moment durant lequel l'exercice est accessible.

**question** Constitue la question elle-même, éventuellement avec un fichier attaché (image, animation Flash, ...). Y est repris le type de question, parmi les quatre décrits plus haut, ainsi que, par le biais du champ **ponderation**, un facteur de pondération des points obtenus.

**answer** Représente les réponses possibles à une question, et spécifie les points rapportés par chacune d'elles.

**rel\_test\_question** n'est rien de plus qu'une table de jointure réduite à sa plus simple expression.

Les tables **question** et **answer** possèdent également un champ définissant l'ordre d'appartition.

### 6.3.3 Conclusion

La structure est telle qu'un exercice est constitué d'une ou plusieurs questions, chacune ayant une ou plusieurs réponses.

Les questions apparaissent toutes dans la *banque de questions* de Claroline et peuvent être réutilisées d'un exercice à l'autre.

Étant donné l'existence de classes déjà écrites pour la manipulation des exercices, il va de soi que j'en ai fait usage durant mon travail.

# Chapitre 7

## Étude du format SCORM

Le format SCORM<sup>1</sup>, créé par le projet ADL<sup>2</sup> est destiné à favoriser la création de contenus d'apprentissage réutilisables sous forme d'« objets d'enseignement » dans un cadre technique défini pour l'apprentissage assisté par ordinateur et sur le Web.

Un contenu SCORM se présente sous la forme d'une archive au format `.zip`, dont nous allons détailler, d'abord le contenu, ensuite le fonctionnement.

### 7.1 L'archive SCORM

Si nous ouvrons un objet SCORM à l'aide d'un programme d'archivage<sup>3</sup>, nous y découvrons les éléments suivants :

- Les documents, images, et autres ressources composant le contenu d'apprentissage proprement dit. Ceux-ci peuvent être rangés dans d'éventuels sous-répertoires.
- Obligatoire dans tout objet SCORM, et toujours à la racine de l'archive, un fichier `imsmanifest.xml`. Celui-ci décrit de manière détaillée et complète, à destination du LMS, les modules et leur organisation.
- Les schémas XML nécessaires à l'analyse du précédent fichier.
- Des fichiers XML reprenant les métadonnées relatives aux ressources. Elles sont facultatives et peuvent indifféremment se trouver dans des fichiers à part, ou directement intégrées au *manifest*.
- Éventuellement quelques fichiers *javascript*, destinés à faciliter le dialogue des pages de contenu avec le LMS.

---

<sup>1</sup>Sharable Content Object Reference Model

<sup>2</sup>*Advanced Distributed Learning*, projet lancé en 1997 par le Département de la défense américain et le White House Office of Science and Technology

<sup>3</sup>en somme, n'importe quel logiciel capable de décompresser une archive `.zip`

## Chapitre 7. Étude du format SCORM

---

Le fichier `imsmanifest.xml`, ou plus souvent appelé *manifest*, est composé de deux parties principales. Commençons pour plus de facilité par la seconde partie. Celle-ci n'est autre qu'une liste de références à tous les fichiers nécessaires au bon fonctionnement du parcours, ainsi que les dépendances entre eux. En reprenant l'exemple de la page HTML contenant une image, une ressource dans le *manifest* correspondra à l'image. Une autre, correspondant au fichier HTML, aura en outre une référence à l'image. Chaque ressource doit impérativement indiquer quels fichiers lui sont nécessaires.

La première partie du fichier réalise la structure arborescente des modules, représentés ici par des *item*, chacun référençant sa ressource.

La **figure 7.1** donne un aperçu du fichier `imsmanifest.xml`.

```
1 <?xml version="1.0" ?>
  <manifest identifieur="Exemple" version="1.2">
    [...]
5   <organizations default="One">
     <organization identifieur="One">
       <title>Exemple de parcours d'une page</title>
       <item identifieur="item1" identifieurref="fichier1">
10        <title>Ajout d'un titre inutile</title>
       </item>
     </organization>
   </organizations>

15  <resources>
     <resource identifieur="fichier1" type="web content"
       adlcp:SCORMtype="SCO" href="page.html">
       <file href="page.html" />
       <file href="image.gif" />
       <file href="autre_image.jpg" />
20     </resource>
   </resources>

</manifest>
```

**FIG. 7.1:** Exemple de fichier `imsmanifest.xml`

Il peut aussi contenir de très nombreuses *métadonnées*, ou faire référence à des fichiers XML externes les contenant. Celles-ci définissent des informations diverses relatives la plupart des éléments présents dans le *manifest*.

Nous pouvons par exemple y trouver une description détaillée du parcours, la langue dans lequel il est écrit, les informations concernant son auteur, ou encore la licence sous laquelle il est distribué.

Peuvent y figurer également les commentaires associés aux modules et aux ressources, ou encore leurs auteurs respectifs.

## Chapitre 7. Étude du format SCORM

---

Un exemple de *manifest* contenant d'innombrables métadonnées est présenté en page 49 du livre *Cooking up a SCORM*, disponible sur le CD-ROM annexe.

### 7.2 Dialogue avec le LMS

Chaque module d'un parcours pédagogique doit signaler au LMS l'état d'avancement de l'étudiant. Ceci est réalisé par le biais d'une API<sup>1</sup> normalisée en langage JavaScript.

Seul un nombre réduit de fonctions est nécessaire au bon fonctionnement de tout cela. De plus, un fichier `APIWrapper.js`, librement téléchargeable, contient toute une série de fonctions destinées à faciliter le dialogue entre un module SCORM et le LMS sur lequel il est utilisé.

Une communication avec la plateforme ressemblera au schéma suivant :

1. Recherche de l'objet correspondant à API SCORM exportée par le LMS. Cet objet peut avoir été créé dans une autre fenêtre du navigateur ou dans un autre cadre de la page. La fonction `getApiHandle()` se charge entièrement de cette recherche.
2. Initialisation de l'API LMS. Il est nécessaire d'initialiser la communication avec le LMS, pour signaler que l'on désire entamer le dialogue. La fonction `doLMSInitialize()` est disponible à cet effet. Une autre fonction, `LMSIsInitialized()` permet de savoir si le LMS a déjà été initialisé.
3. Communication avec le LMS. Deux fonctions sont capables d'enregistrer ou de récupérer des paires *clé, valeur* : `LMSGetValue(clé)` et `LMSSetValue(clé, valeur)`.
4. Valider les modifications effectuées. Les valeurs précédemment envoyées au LMS ne seront réellement enregistrées qu'après appel à la fonction `doLMSCommit()`.
5. Fermer la communication, en appelant `DoLMSFinish()`.

Quelques fonctions supplémentaires sont disponibles pour la gestion des erreurs, comme `doLMSGetLastError()` et `doLMSGetErrorString()`.

Remarquez qu'un module ne doit pas s'identifier auprès de la plateforme lorsqu'il désire communiquer avec elle. C'est le LMS qui est chargé de savoir ce qu'il est en train d'afficher à l'étudiant.

---

<sup>1</sup>Application Programming Interface : liste des fonctions qu'une entité, application ou bibliothèque, met à disposition du programmeur

## Chapitre 7. Étude du format SCORM

---

Les clés intéressantes dans le cas qui nous occupe concernent l'état d'avancement dans une leçon. Le champ **cmi.core.lesson\_status** représente cet état et peut prendre les valeurs textuelles suivantes :

**"not attempted"** : L'élève n'a jamais commencé ce module. C'est la valeur par défaut assignée à tous les modules lorsque l'élève entame un parcours.

**"incomplete"** : L'élève a commencé la lecture du module, mais n'est pas encore arrivé au bout de celui-ci. Ce cas peut se présenter lorsqu'un module est constitué de plusieurs pages et que l'élève n'a pas atteint la dernière.

**"completed"** : L'élève a terminé ce module.

**"passed"** : Uniquement utilisé pour les exercices, cette valeur signifie que l'exercice a été réalisé avec succès.

**"failed"** : également utilisé pour les exercices, indique un échec.

Quelques autres champs donnent les indications nécessaires pour les exercices :

**cmi.core.score.raw** : La note obtenue à l'exercice par l'étudiant.

**cmi.core.score.max** : La note maximale qu'il est possible d'atteindre en répondant correctement à toutes les questions de l'exercice.

**cmi.core.score.min** : La note minimale, souvent égale à 0.

**cmi.core.score.raw\_to\_pass** : La note nécessaire à la réussite, ramenée sur 100.

Je voudrais vous faire partager ici une petite réflexion au sujet de l'utilisation du terme *raw*.

Il s'agit à première vue des points « bruts » obtenus à l'exercice, de la même manière qu'un examen noté sur 48 est ramené à une cote sur 20 en fin d'année. Et c'est effectivement le cas en ce qui concerne le champ **cmi.core.score.raw**, et dans Claroline de manière générale.

Contre toute attente, le champ **cmi.core.score.raw\_to\_pass** requiert, lui, que la note soit ramenée sur 100.

Ce manque de cohérence et le manque de documentation à ce sujet m'ont induit en erreur lors de mes premières versions des fonctions d'exportation. Heureusement, en réimportant les parcours exportés dans Claroline, il est aisé de tester le bon (ou mauvais) fonctionnement de ceux-ci.

# Chapitre 8

## Réalisation de l'exportation

Ayant à présent terminé l'étude de l'existant, nous pouvons nous lancer dans l'analyse et la réalisation de l'application.

Dans ce chapitre, nous commencerons par expliciter les méthodes mises en œuvre pour réaliser l'exportation des ressources, puis nous détaillerons la création du fichier manifest. Enfin, nous passerons rapidement en revue quelques différences que l'on peut trouver entre le fonctionnement sous Claroline et le standard SCORM.

### 8.1 Exportation des ressources

#### 8.1.1 Les documents

La réalisation de l'exportation des documents fut assez aisée, bien que quelques points spécifiques méritent une attention toute particulière.

Pour commencer, les modules du parcours pédagogique ont l'obligation de signaler au LMS tout changement d'état. Si un document doit être lu pour passer au suivant, il doit inévitablement signaler au LMS s'il a été visionné ou non.

S'il est facile de le faire depuis une page HTML, capable d'exécuter les fonctions javascript idoines, la tâche se complique lorsque le document est d'un autre format, comme le PDF.

La solution retenue est celle proposée par ADL : l'intégration du document, sans aucune distinction de son type, dans une page HTML générée, composée de cadres, ou *frames*. Un cadre intègre le document en question. La mise à jour de l'état (**cmi.core.lesson\_status**) se fait immédiatement au chargement de la page.

La valeur de l'attribut *onload* utilisé dans la description des cadres est la fonction javascript à appeler au chargement de la page. Cette fonction, **immediateCom-**

## Chapitre 8. Réalisation de l'exportation

---

```
1 <html><head>
  <script src="APIWrapper.js"
    type="text/javascript" language="JavaScript">
  </script>
5  <title>Default Title</title>
</head>
<frameset border="0" rows="100%,*" onload="immediateComplete()">
  <frame src="Documents/autre_document.pdf" scrolling="auto">
  <frame src="">
10 </frameset>
</html>
```

FIG. 8.1: Exemple de *frames* pour l'intégration d'un document

`plete()`, a été ajoutée par mes soins dans le fichier `APIWrapper.js` et gère tout le nécessaire pour signaler au LMS que le module doit passer à l'état "**completed**".

Le second cadre, en réalité inutile, est laissé vide.

Une deuxième subtilité vient compliquer légèrement la tâche de l'utilisateur. Dans Claroline, tous les fichiers propres à un cours sont situés dans un répertoire bien précis. De ce fait, si l'utilisateur rédige un document HTML contenant des images, il n'y a aucun problème pour savoir où se trouvent ces images, relativement au document.

Les choses se compliquent si ce document est utilisé dans un parcours pédagogique que l'on désire exporter, car seules les ressources listées dans le parcours seront présentes dans le contenu SCORM.

La tâche incombe à l'utilisateur d'ajouter manuellement les images et autres objets au parcours pédagogique, et de les indiquer comme étant « cachées ». Ainsi, elles n'apparaîtront pas dans la liste des modules, mais seront présentes dans l'archive SCORM.

Étant donné la grande diversité des méthodes d'intégration de contenu dans une page HTML, aucune solution automatique n'a été retenue à ce sujet.

Tous les documents sont disposés dans un répertoire nommé `Documents` à l'intérieur de l'archive SCORM en respectant l'arborescence définie dans Claroline. Les cadres HTML sont créés à la racine de l'archive.

### 8.1.2 Les liens

Les liens vers des sites externes étant constitués, sous Claroline, d'une page HTML provoquant une redirection, ils sont traités de manière identique aux documents.

## Chapitre 8. Réalisation de l'exportation

---

### 8.1.3 Exportation des exercices

Cette partie du stage fut sans conteste la plus ardue de toutes, à cause des différences fondamentales entre les exercices natifs et les exercices au format SCORM. C'était aussi la partie considérée comme la plus importante du stage.

Les exercices natifs fonctionnent en intégration totale avec la base de données, et c'est le moteur de Claroline qui est chargé de la correction et du calcul des points. À l'inverse, un exercice au format SCORM doit inclure toutes les données et fonctions nécessaires à sa correction, la plateforme ne gérant que la note obtenue et la réussite ou non.

Il a donc été nécessaire, lors de la génération des pages HTML contenant les questions d'un exercice, de générer également les fonctions javascript de correction et de calcul des points. Heureusement, une propriété du format HTML s'est avérée fort utile : les attributs *id* et *name* applicables aux balises.

Une base commune est utilisée pour la génération des différents types d'exercices. Le format de destination étant une page HTML, les composants représentant les questions sont des balises de formulaires.

Chaque balise HTML représentant une réponse se voit affecter deux attributs utilisés lors de la correction. Tout d'abord un attribut *id* de la forme « scorm\_[compteur] », la valeur du compteur étant chaque fois incrémentée. L'attribut *name* contient une chaîne de caractères constituée comme suit : « [Type de question]\_[numéro de la question]\_[numéro de la réponse] ».

Voyons à présent de quelle manière se présente le code généré de chaque question. Chacune des descriptions sera accompagnée d'un exemple, épuré de ses balises de mise en page.

**Choix multiple à réponse unique.** Les réponses possibles sont rendues sous forme de boutons *radios* (cases à cocher mutuellement exclusives). Les balises HTML correspondantes disposent d'un attribut *value* contenant le nombre de points obtenus lorsque la réponse est cochée.

La chaîne représentant le type de question à l'intérieur de la balise *name* est « unique ».

Remarquez que les trois réponses ont une balise *name* identique dans laquelle le numéro de réponse est remplacé par la lettre « x ». Cette simplification est rendue nécessaire par le comportement des navigateurs qui groupent un ensemble de boutons *radios* en fonction de cet attribut.

**Choix multiple à réponses multiples** Similaire au précédent, si ce n'est que les réponses sont représentées par des *checkboxes* (boîtes à cocher). C'est également l'attribut *value* qui indique le nombre de points à assigner lorsque la case est cochée.

Le type de question est « multiple ».

## Chapitre 8. Réalisation de l'exportation

---

```
1 Sélectionnez la réponse correcte.  
  
<input type="radio" name="unique_1_x" id="scorm_1" value="10"> Correct  
<input type="radio" name="unique_1_x" id="scorm_2" value="0"> Incorrecte  
5 <input type="radio" name="unique_1_x" id="scorm_3" value="5"> Mitigée
```

FIG. 8.2: Exemple de question à réponse unique

```
1 Sélectionnez la ou les réponses correctes.  
  
<input type="radio" name="multiple_2_1" id="scorm_4" value="10">  
Réponse correcte  
5 <input type="radio" name="multiple_2_2" id="scorm_5" value="-10">  
Réponse absolument incorrecte  
<input type="radio" name="multiple_2_3" id="scorm_6" value="5">  
Réponse mitigée
```

FIG. 8.3: Exemple de question à réponses multiples

**Mise en correspondance** Ce type de question, plus complexe, est représenté par deux colonnes, séparées par des « drop down », ou boîtes de sélection, figurées par la balise *select* et remplies par une suite de balises *option* indiquant les relations et les points correspondants.

L'exemple ne reprend qu'une seule mise en correspondance pour une plus grande clarté. Dans un exercice réel, le bloc *select* sera présent vis-à-vis de chaque entrée de la première colonne.

Le type de question est « matching ».

```
1 Indiquez pour chaque ville dans quel pays elle est située.  
  
<!-- Dix villes numérotées séquentiellement dans la première colonne, 4 pays  
numérotés de 'A' à 'D' dans la seconde.  
5 Face à chaque nom de ville se trouve une construction de ce genre, dans  
laquelle les points donnés en attribut "value" changent. -->  
  
<select name="matching_3_1" id="scorm_7">  
10 <option value="0">--</option> <!-- pour ne pas répondre -->  
<option value="10"> A </option>  
<option value="0"> B </option>  
<option value="0"> C </option>  
<option value="-5"> D </option> <!-- pour un pays inexistant -->  
</select>
```

FIG. 8.4: Exemple de question de mise en correspondance

**Texte à trous** Ce genre d'exercice est un cas particulier, parce que les informations que nous pouvons intégrer aux balises ne suffisent plus. Il est nécessaire de construire un tableau associatif indexé sur l'attribut *name* des zones de

## Chapitre 8. Réalisation de l'exportation

---

texte, et reprenant la réponse correcte ainsi que les points que cette dernière rapporte. Une fois la totalité des questions passées en revue, ce tableau est exporté dans la page HTML sous forme d'un tableau javascript appelé *fillAnswerList*.

Le type de question est « fill »

```
1 Conjuguez les verbes mis entre parenthèses.  
  
Demain, vous <input type="text" name="fill_4_1" id="scorm_8"> (manger,  
2 points) avec les personnes qui <input type="text" name="fill_4_2"  
5 id="scorm_9"> (aller, 3 points) vous engager.
```

FIG. 8.5: Exemple de texte à trous

Voici à présent de quelle manière ce tableau est généré dans la page résultante.

```
1 <script type="text/javascript" language="javascript">  
    var fillAnswerList = new Array();  
    fillAnswerList ['fill_4_1 '] = new Array('mangez', 2);  
    fillAnswerList ['fill_4_2 '] = new Array('vont', 3);  
5    [...]  
</script>
```

FIG. 8.6: Tableau de correction des textes à trous

### 8.1.4 Correction des exercices

Maintenant que les questions des exercices ont été exportées d'une manière qui simplifie leur correction, il s'agit de les regrouper dans la page HTML et de réaliser le calcul des points obtenus. À cet effet, un bouton est inséré dans le bas de la page et les calculs sont effectués lorsque l'utilisateur clique dessus pour indiquer qu'il a terminé.

Le calcul des points bruts est réalisé par la fonction **CalculateRawScore()** présente dans le fichier `scores.js`<sup>1</sup>, inclus dans chaque page d'exercices.

Cette fonction demande trois arguments : la référence DOM<sup>2</sup> de l'objet document, le nombre de réponses à vérifier, et une référence au tableau *fillAnswerList* précédemment construit.

Il est alors trivial de retrouver toutes les réponses à analyser ainsi que, à l'aide d'une expression rationnelle, le type de chacune d'elles.

---

<sup>1</sup>La version complète de cette fonction se trouve en annexe

<sup>2</sup>Document Object Model, permettant d'interagir avec la structure d'un document HTML depuis le javascript

## Chapitre 8. Réalisation de l'exportation

---

Malheureusement, cette solution à base d'expressions rationnelles ne m'est apparue que fort tard, me retardant longtemps, faute de trouver une méthode acceptable qui permette de retrouver, dans la page d'exercices, toutes les informations relatives aux réponses données par l'étudiant.

```
1  function CalculateRawScore(objDoc, idCount, fillin)
   {
     [...]
     var myRegexp = /^(.*)_(.*)_(.*)/
5   var eltId;
     var element;
     var i;
     var questionType;

10  // Boucle sur tous les éléments correspondant à une réponse
     for (i=0; i<idCount; i++)
     {
       eltId = 'scorm_' + i;
       element = objDoc.getElementById(eltId);
15  questionType = myRegexp.exec(element.name)[1];

       switch (questionType)
       {
20         // Traitement de la réponse suivant le type de question.
           [...]
       }
     }

     return score;
25  }
```

FIG. 8.7: Aperçu de la fonction CalculateRawScore du fichier scores.js

Les valeurs des attributs *value* des réponses sélectionnées sont ajoutées à la variable *score*. Pour les textes à trous, les réponses sont comparées avec celles fournies dans le tableau généré (voir **figure 8.6**), et les points sont obtenus si elles concordent.

## Chapitre 8. Réalisation de l'exportation

---

Pour savoir si une note satisfaisante a été obtenue, nous devons préalablement calculer quelques valeurs qui seront exportées « en dur » dans le code javascript généré. La table de correction des textes à trous est l'une d'elles, voici les autres.

- Nous devons connaître la note minimale, sur 100, indiquant la réussite. Celle-ci est disponible *via* notre objet Exercice et se nomme **raw\_to\_pass**.
- Le nombre brut des points sur lesquels est noté l'exercice. Ce nombre correspond simplement à la somme des points de chaque question, et se nomme **weighting**.
- Le nombre total de réponses dans l'exercice, qui nous permettra de retrouver tous les éléments dont l'*id* est « scorm\_x »

Il reste à exporter dans la page une fonction javascript qui sera chargée, après appel à **CalculateRawScore()**, de ramener le score sur 100 et de signaler au LMS les notes brutes obtenue et maximale, ainsi que la réussite ou non de l'exercice. C'est cette fonction **calcScore()** qui est appelée lorsque l'utilisateur clique sur le bouton indiquant qu'il a terminé de répondre à l'exercice.

```
1  function calcScore() {
    var rawScore = CalculateRawScore(document, nbAnswer, fillAnswerList);
    var score = Math.max(Math.round(rawScore * 100 / weighting), 0);

5  // Bornes des points pour l'exercice
    doLMSSetValue("cmi.core.score.max", weighting);
    doLMSSetValue("cmi.core.score.min", 0);

    // Points obtenus
10   doLMSSetValue("cmi.core.score.raw", rawScore);

    if (score >= raw_to_pass) {
        doLMSSetValue("cmi.core.lesson_status", "passed");
    } else {
15   doLMSSetValue("cmi.core.lesson_status", "failed");
    }
}
}
```

**FIG. 8.8:** Aperçu de la fonction calcScore des pages d'exercices exportés

Finalement, les pages HTML générées sont placées à la racine de l'archive SCORM. Si des fichiers étaient attachés aux questions, ceux-ci sont placés dans le répertoire **Exercises**. Si parmi les fichiers attachés se trouve un ou plusieurs fichiers MP3<sup>1</sup>, un lecteur écrit en Flash et disponible dans Claroline, est également ajouté à la racine de l'archive.

Le fichier **score.js** est placé à la racine de l'archive.

---

<sup>1</sup>Format de compression audiophonique

### 8.2 Cas d'un contenu SCORM existant

Si Claroline permettait déjà l'importation de contenus SCORM, ces derniers sont fortement différents des parcours pédagogiques créés de toutes pièces sur la plateforme.

Lors de l'importation dans Claroline, le parcours est décompressé dans un répertoire qui lui est propre. Le fichier `imsmanifest.xml` est ensuite analysé et les entrées nécessaires sont ajoutées dans la base de données.

L'utilisateur peut alors appliquer les modifications qu'il désire à la structure du parcours en ajoutant ou retirant des modules, ou en changeant l'ordre.

Le problème se pose alors de la ré-exportation de ce parcours. . .

Il a été décidé à l'unanimité des développeurs d'exporter la totalité du contenu SCORM original dans un répertoire nommé `OrigScorm`. Cette méthode nous assure que tout le contenu nécessaire sera présent, au risque de rendre le paquet résultant inutilement trop volumineux.

Les modules étant déjà conformes au standard, il n'y a pas d'autre opération à effectuer que de les référencer dans le manifeste, de la même manière que les documents classiques.

### 8.3 Création du *manifest*

La création du fichier `imsmanifest.xml` ne pose pas de problème particulier. C'est une simple mise en forme dans un flux XML d'éléments tirés de la base de données, contenant directement toutes les informations nécessaires.

Après avoir constitué une table des ressources et de leurs dépendances, la présentation hiérarchique des modules est réalisée à l'aide d'une classique fonction récursive.

Voici le fichier `imsmanifest.xml` généré par Claroline pour le parcours pédagogique utilisé en exemple dans le chapitre 5.

```
1 <?xml version="1.0" encoding="iso-8859-1" ?>
  <manifest identifier="SingleCourseManifest" version="1.1"
    xmlns="http://www.imsproject.org/xsd/ims_cp_rootv1p2"
    xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_rootv1p2"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_rootv1p2p1"
    xsi:schemaLocation="http://www.imsproject.org/xsd/ims_cp_rootv1p2
    ims_cp_rootv1p2.xsd
10    http://www.imsglobal.org/xsd/imsmd_rootv1p2p1 imsmd_rootv1p2p1.xsd
    http://www.adlnet.org/xsd/adlcp_rootv1p2 adlcp_rootv1p2.xsd">
  <organizations default="A1">
  <organization identifier="A1">
```

## Chapitre 8. Réalisation de l'exportation

---

```
15 <title>Titre du parcours pédagogique</title>
<item identifiant="I_3" isvisible="true" >
  <title>Première partie</title>
  <item identifiant="I_4" isvisible="true" identifiantref="R_4" >
    <title>Document_1.htm</title>
  </item>
20 <item identifiant="I_5" isvisible="true" identifiantref="R_5" >
  <title>Lien_vers_site_externe.url</title>
  </item>
  <item identifiant="I_2" isvisible="true" identifiantref="R_2" >
    <title>Exemple d'exercice</title>
25 </item>
</item>
<item identifiant="I_6" isvisible="true" >
  <title>Seconde partie</title>
  <item identifiant="I_7" isvisible="true" >
    <title>Section 1</title>
    <item identifiant="I_9" isvisible="true" identifiantref="R_9" >
      <title>autre_document.pdf</title>
      <adlcp:prerequisites type="aicc_script">I_2</adlcp:prerequisites>
    </item>
30 </item>
  <item identifiant="I_8" isvisible="true" >
    <title>Section 2</title>
  </item>
35 </item>
</organization>
40 </organizations>

<resources>
<resource identifiant="R_4" type="webcontent" adlcp:scormtype="sco" href="
frame_for_4.html">
45 <file href="frame_for_4.html" />
  <file href="Documents/Document_1.htm" />
</resource>
<resource identifiant="R_5" type="webcontent" adlcp:scormtype="sco" href="
frame_for_5.html">
  <file href="frame_for_5.html" />
50 <file href="Documents/Lien_vers_site_externe.url" />
</resource>
<resource identifiant="R_2" type="webcontent" adlcp:scormtype="sco" href="
quiz_1.html" >
  <file href="quiz_1.html" />
</resource>
55 <resource identifiant="R_9" type="webcontent" adlcp:scormtype="sco" href="
frame_for_9.html">
  <file href="frame_for_9.html" />
  <file href="Documents/autre_document.pdf" />
</resource>
60 </resources>
</manifest>
```

Le fichier a été épuré des métadonnées pour plus de lisibilité. Bien qu'elles soient facultatives, nous avons décidé d'en réaliser l'export. Elles correspondent aux commentaires que l'utilisateur peut ajouter, sous Claroline, aux modules et aux ressources.

## Chapitre 8. Réalisation de l'exportation

---

Nous voyons, de la ligne 15 à la ligne 39, la structure de la présentation du parcours pédagogique. En ligne 33, nous pouvons apercevoir la balise `<adlcp :prerequisites>` qui définit le prérequis : pour avoir accès au fichier PDF (*item* « I\_9 »), il faut au préalable avoir terminé le module dont la référence est donnée, soit « I\_2 », ce qui correspond à l'exercice.

Il est parfaitement envisageable d'avoir plusieurs prérequis spécifiés pour un même *item*.

Les lignes 43 à 59 listent la totalité des ressources utilisées par ce parcours. Les ressources de type documents sont chaque fois composées de deux fichiers : le document proprement dit, et la page HTML de *frames* qui le contiendra et dont le rôle est de dialoguer avec le LMS, comme vu au chapitre 8.

### 8.3.1 Différences entre SCORM et Claroline

Un point malgré tout reste à signaler, car il s'agit d'une différence fondamentale entre Claroline et SCORM.

Sur Claroline, un module non terminé peut rendre inaccessible la totalité des modules suivants. Cependant, cette gestion des prérequis fonctionne en sens inverse dans la description d'un parcours SCORM : un module est inaccessible tant que certains autres n'ont pas été terminés.

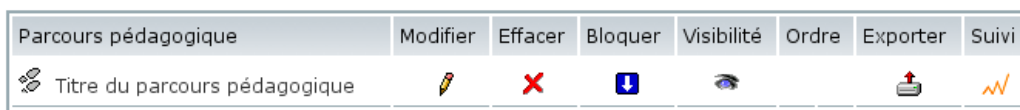
La solution retenue est que lorsqu'un module « bloquant » est rencontré pendant l'exportation, celui-ci est ajouté aux prérequis de tous ceux qui le suivent.








Précisons encore que Claroline ne gère pas les prérequis SCORM lors de l'importation de contenu, ce qui nous assure de ne pas nous trouver face à une telle configuration durant l'exportation.

# Chapitre 9

## Intégration à la plateforme

Les modifications apportées au code existant de Claroline sont réellement minimes. Elles consistent en l'ajout d'une colonne intitulée « Exporter » dans la page d'accès aux parcours pédagogiques, assisté d'une gestion des erreurs qui pourraient se produire lors de la création de l'archive SCORM.



Parcours pédagogique	Modifier	Effacer	Bloquer	Visibilité	Ordre	Exporter	Suivi
 Titre du parcours pédagogique							

**FIG. 9.1:** Administration des parcours pédagogiques

Lorsque l'utilisateur clique sur l'icône d'exportation du parcours, la création du parcours est réalisée et le résultat est « zippé » puis envoyé au navigateur pour téléchargement. En cas d'erreur survenant durant le processus, celle-ci est mise en valeur sur la page, appliquant le style des erreurs de Claroline.

Tout le reste du code a pu être réalisé dans des fichiers nouvellement créés pour l'occasion.

Vous trouverez en **annexe C** une liste détaillée des fichiers qui ont été nécessaires à cette réalisation et leur emplacement dans l'arborescence des sources de Claroline.

Quatrième partie  
Conclusion

# Chapitre 10

## Concernant le code

### 10.1 État actuel

L'exportation des parcours pédagogiques au format SCORM depuis Claroline est à présent pleinement fonctionnelle et parfaitement utilisable. Tout parcours pédagogique peut être exporté, en ce compris les exercices.

Mon travail a été intégré à la branche principale du CVS à la fin de mon stage et sera disponible dans la prochaine version de Claroline, prévue pour le mois de septembre.

La simplicité d'accès pour l'utilisateur entre judicieusement dans la ligne de conduite générale de Claroline.

Pour autant, il n'en reste pas moins de nombreux points sur lesquels j'aurais aimé pouvoir passer un peu plus de temps. Le fonctionnement n'est pas absolument identique entre les exercices réalisés sous Claroline et leur version exportée, et certaines facilités qu'il m'aurait plu d'intégrer sont absentes.

### 10.2 Regrets

J'ai déjà précisé, dans le calendrier, que près de trois semaines m'avaient été nécessaires pour comprendre les tenants et aboutissants du format SCORM. Malheureusement, il m'a aussi fallu régulièrement partir en chasse d'informations sur des points de détail durant les semaines qui ont suivi.

Ceci a beaucoup contribué à me faire perdre un temps précieux.

### 10.3 Améliorations possibles

Voici une liste, certainement non exhaustive, des améliorations envisageables que l'on pourrait apporter à mon travail.

## Chapitre 10. Concernant le code

---

### 10.3.1 Automatisation des dépendances entre ressources

J'en ai déjà parlé plus haut mais permettez-moi de revenir sur le sujet.

Si une image est incluse dans une page HTML, l'image doit également faire partie du parcours pédagogique et être notée comme étant un module « caché. »

Il serait parfaitement possible de réaliser un *parser* HTML capable de recenser, à l'intérieur d'un document, toutes les références aux images et objets embarqués dans la page. Cela rendrait très certainement service aux utilisateurs.

Il existe déjà dans Claroline une fonction de recherche de références aux images, que je n'ai malheureusement pas eu le temps d'intégrer.

### 10.3.2 Exercices en plusieurs pages

Claroline permet de créer des exercices dont chaque question est disposée sur sa propre page HTML.

Ce type de rendu n'est pas disponible pour les exercices exportés, où toutes les questions sont disposées sur une seule et même page.

### 10.3.3 Corrigé des exercices

Les exercices réalisés sous Claroline offrent la possibilité d'afficher la correction une fois remplis.

Il m'a été impossible de trouver de quelle manière une telle chose pouvait être effectuée en SCORM. Néanmoins, je sais que SCORM dispose d'énormes capacités, et je doute que ce ne soit pas réalisable.

### 10.3.4 Duplication de code

La génération des exercices au format SCORM duplique énormément de code provenant des classes de gestion des exercices.

Un important travail de refactorisation pourrait être effectué de manière à partager un seul code pour l'affichage des exercices dans la plateforme et pour la génération des pages d'exercices exportés.

Malheureusement, de nombreuses différences subsistent, et la somme de travail est conséquente.

# Chapitre 11

## Évaluation et critique

Dès le début du mois de mars, après avoir commencé l'étude du format SCORM, je me suis vite rendu compte que, même si je devais normalement avoir le temps de terminer ce qui m'était assigné, le temps imparti était calculé au plus juste.

C'est ce qui s'est avéré, et ce n'est que de justesse que tout a été terminé dans les temps, mais pas pour les raisons que j'imaginai. Je me rends compte, avec le recul, que j'ai passé trop longtemps à tenter de comprendre certaines subtilités de SCORM, alors que j'avais en main tout le nécessaire pour entamer la programmation.

Ceci est probablement dû au fait que, si j'ai déjà participé à de nombreux projets Libres, j'ai toujours été motivé par le désir d'apprendre quelque chose de neuf. Je serais bien mal avisé de ne pas remarquer l'apport positif que ces contributions ont eues sur mon expérience dans le développement, mais il est désormais évident qu'il manquait quelque chose.

Je souhaite malgré tout souligner que mon estimation de la durée nécessaire était correcte à quelques jours près — dans le bon sens — nonobstant les écueils et impondérables rencontrés durant ces trois mois et demi.

Ces quelques semaines de stage m'ont été très bénéfiques parce qu'elles m'ont fait me rendre compte qu'apprendre n'est pas tout, il faut aussi mettre ses connaissances fraîchement acquises en application si l'on veut être capable de respecter les délais inhérents à tout travail de développeur !

Tout cela a contribué à une progressive remise en cause de certaines de mes méthodes de travail.

Je voudrais également signaler que l'intégration au quotidien d'une équipe de développement m'a énormément apporté. Certes je travaillais principalement sur un module totalement déconnecté de leurs préoccupations journalières, mais j'ai rapidement été impliqué dans les discussions et prises de décision, exactement au même titre que les développeurs permanents.

# Cinquième partie

## Annexes

# Annexe A

## Détail des tables utilisées

### A.1 Parcours pédagogique

**TAB. A.1:** Structure de la table learnPath

Champ	Type	Null	Défaut
<i>learnPath_id</i>	int(11)	Non	
name	varchar(255)	Non	
comment	text	Non	
lock	enum('OPEN', 'CLOSE')	Non	OPEN
visibility	enum('HIDE', 'SHOW')	Non	SHOW
rank	int(11)	Non	0

**TAB. A.2:** Structure de la table module

Champ	Type	Null	Défaut
<i>module_id</i>	int(11)	Non	
name	varchar(255)	Non	
comment	text	Non	
accessibility	enum('PRIVATE', 'PUBLIC')	Non	PRIVATE
startAsset_id	int(11)	Non	0
contentType	enum('CLARODOC', 'DOCUMENT', 'EXERCISE', 'HANDMADE', 'SCORM', 'LABEL')	Non	CLARODOC
launch_data	text	Non	

## Annexe A. Détail des tables utilisées

---

**TAB. A.3:** Structure de la table rel\_learnPath\_module

Champ	Type	Null	Défaut
<i>learnPath_module_id</i>	int(11)	Non	
learnPath_id	int(11)	Non	0
module_id	int(11)	Non	0
lock	enum('OPEN', 'CLOSE')	Non	OPEN
visibility	enum('HIDE', 'SHOW')	Non	SHOW
specificComment	text	Non	
rank	int(11)	Non	0
parent	int(11)	Non	0
raw_to_pass	tinyint(4)	Non	50

**TAB. A.4:** Structure de la table asset

Champ	Type	Null	Défaut
<i>asset_id</i>	int(11)	Non	
module_id	int(11)	Non	0
path	varchar(255)	Non	
comment	varchar(255)	Oui	NULL

## A.2 Exercices

**TAB. A.5:** Structure de la table test

Champ	Type	Null	Défaut
<i>id</i>	mediumint(8)	Non	
titre	varchar(200)	Non	
description	text	Non	
type	tinyint(4)	Non	1
random	smallint(6)	Non	0
active	tinyint(4)	Non	0
max_time	smallint(5)	Non	0
max_attempt	tinyint(3)	Non	0
show_answer	enum('ALWAYS', 'NEVER', 'LASTTRY')	Non	ALWAYS
anonymous_attempts	enum('YES', 'NO')	Non	YES
start_date	datetime	Non	0000-00-00 00 :00 :00
end_date	datetime	Non	0000-00-00 00 :00 :00

## Annexe A. Détail des tables utilisées

---

**TAB. A.6:** Structure de la table question

Champ	Type	Null	Défaut
<i>id</i>	mediumint(8)	Non	
question	varchar(200)	Non	
description	text	Non	
ponderation	float	Oui	NULL
q_position	mediumint(8)	Non	1
type	tinyint(3)	Non	2
attached_file	varchar(50)	Oui	

**TAB. A.7:** Structure de la table rel\_test\_question

Champ	Type	Null	Défaut
<i>question_id</i>	mediumint(8)	Non	0
<i>exercice_id</i>	mediumint(8)	Non	0

**TAB. A.8:** Structure de la table answer

Champ	Type	Null	Défaut
<i>id</i>	mediumint(8)	Non	0
<i>question_id</i>	mediumint(8)	Non	0
reponse	text	Non	
correct	mediumint(8)	Oui	NULL
comment	text	Oui	NULL
ponderation	float	Oui	NULL
r_position	mediumint(8)	Non	1

# Annexe B

## Contenu du CD-ROM

Vous trouverez ici la liste des fichiers disponibles sur le CD-ROM accompagnant le présent document. Le contenu de ce CD-ROM est également disponible à l'adresse <http://allergy.alrj.org/TFE/> ainsi que sur demande à [amand.tihon@alrj.org](mailto:amand.tihon@alrj.org)

- /Claroline** Les sources de la plateforme d'apprentissage en ligne Claroline.
- /claroline-CVS** Un *snapshot* de Claroline tel qu'il était à la fin de mon stage.
- claroline162.zip** L'archive officielle de la dernière version stable, pour les personnes qui souhaiteraient l'installer. Notez que cette version ne propose pas encore l'exportation SCORM.
- claroline170b.zip** L'archive officielle de la version **bêta** de Claroline, incluant l'exportation SCORM.
- /Mémoire** Ce mémoire aux formats **.dvi**, **.pdf** et **.ps**.
- /Sources mémoire** Les sources **L<sup>A</sup>T<sub>E</sub>X** du présent mémoire, accompagnées de tous les fichiers nécessaires à sa reconstruction.
- CookingUpASCORM.pdf** La version électronique du document qui m'a suivi tout au long du stage (voir bibliographie).
- GPL\_V2.txt** Le texte de la GNU General Public License, version 2, sous laquelle est publié Claroline.

# Annexe C

## Code source de l'exportation

Voici la liste des fichiers concernés par l'exportation des parcours pédagogiques au sein de l'arborescence des sources de Claroline. Ceux marqué d'une astérisque ont été repris tels quels et ne sont partant pas de moi. Ce sont les schémas XML utilisés pour l'analyse du fichier `imsmanifest.xml` généré, ainsi que le fichier `APIWrapper.js`.

Tous ces fichiers sont bien sûr disponibles sur le CD-ROM proposé avec le mémoire, dans le répertoire `Claroline/claroline-CVS`.

```
/Claroline/claroline-CVS/claroline/learnPath
|-- export
|   |-- APIWrapper.js          *
|   |-- adlcp_rootv1p2.xsd    *
|   |-- ims_xml.xsd           *
|   |-- imscp_rootv1p1p2.xsd  *
|   |-- imsmd_rootv1p2p1.xsd  *
|   '-- scores.js
|-- include
|   '-- scormExport.inc.php
'-- learningPathList.php
```

### C.1 scores.js

```
1  /*
   *
   * Check the answers of a scorm quiz.
   *
5  */
var DEBUG = false;

function CalculateRawScore(objDoc, idCount, fillin)
```

## Annexe C. Code source de l'exportation

---

```
10 {
11     var i;
12     var eltId;
13     var element;
14
15     var score = 0;
16
17     var questionType;
18     var questionNum;
19     var answerNum;
20
21     var myRegexp = /^(.*)_(.*)_(.*)/
22     var myMatch;
23
24     // Loop over every element with an interesting id ("scorm_*")
25     for (i=0; i<idCount; i++)
26     {
27         eltId = 'scorm_' + i;
28         element = objDoc.getElementById(eltId);
29
30         myMatch = myRegexp.exec(element.name);
31         questionType = myMatch[1];
32         questionNum = myMatch[2];
33         answerNum = myMatch[3];
34
35         switch (questionType)
36         {
37             case 'unique':
38             case 'multiple':
39                 if (element.checked)
40                 {
41                     score += (+element.value);
42                 }
43                 break;
44
45             case 'matching':
46                 score += (+element.value);
47                 break;
48
49             case 'fill':
50                 var textIn = element.value;
51                 if (textIn.toUpperCase() == fillin[element.name][0].
52                     toUpperCase())
53                 {
54                     var w = fillin[element.name][1];
55                     score += (+w);
56                 }
57                 break;
58         }
59     }
60
61     if (DEBUG) alert('Score:\n' + score + ' points');
62     return score;
63 }
64
65 }
```

## Annexe C. Code source de l'exportation

---

### C.2 learningPathList.php

Trois zones de ce fichier ont été modifiée par mes soins. Voici la première, qui effectue l'export à poprement parler.

```
91  if ( $cmd == 'export' )
    {
      include ( 'include/scormExport.inc.php' );
95  $scorm = new ScormExport($_REQUEST['path_id']);
      if ( !$scorm->export() )
      {
        $dialogBox = '<b>Error exporting SCORM package</b><br><ul>';
100         foreach( $scorm->getError() as $error)
            {
              $dialogBox .= '<li>' . $error . '</li>';
            }
          }
      } // endif $cmd == export
```

Les deux autres zone concernent la création d'une colonne supplémentaire dans la liste, contenant l'icône d'export

Nous avons ici la ligne 459, insérée par mes soins, qui correspond au titre de la colonne.

```
451  if($is_AllowedToEdit)
    {
      // Titles for teachers
      echo "<th>". $langModify . "</th>"
455         . "<th>". $langDelete . "</th>"
          . "<th>". $langBlock . "</th>"
          . "<th>". $langVisibility . "</th>"
          . "<th colspan='2'>". $langOrder . "</th>"
          . "<th>". $langExport . "</th>"
460         . "<th>". $langTracking . "</th>";
    }
```

Les lignes suivantes créent les liens vers l'exportation apparaissant devant chaque parcours pédagogique de la liste.

```
806  // EXPORT links
      echo '<td><a href="' . $_SERVER['PHP_SELF'] . '?cmd=export&
path_id=' . $list['learnPath_id'] . '" >'
          . '</a></td>' . "\n";
```

## **C.3 scormExport.inc.php**

Ce fichier est le plus important de tous. C'est dans celui-ci que réside l'intégralité des fonctions d'exportation. Il est abondamment commenté, mais en anglais, les contributeurs à Caroline venant du monde entier.

Étant donné sa longueur, il n'est pas repris ici. Vous le trouverez néanmoins sur le CD-ROM en annexe.

# Bibliographie

- [1] Claude Ostyn, *Cooking up a SCORM*, Clic2Learn, 2003
- [2] *Claroline development wiki*, <http://www.claroline.net/wiki/>
- [3] Steve Lay, *IMS Question and Test Interoperability : Item Implementation*, University of Cambridge, 2004, [http://www.imsglobal.org/question/qti\\_item\\_v2p0pd/implementation.html](http://www.imsglobal.org/question/qti_item_v2p0pd/implementation.html)
- [4] ADL Technical Team, *Sharable Content Object Reference Model (SCORM) Version 1.2*, <http://www.adlnet.org/downloads/120.cfm>
- [5] Karl Fogel and Moshe Bar, *Open Source Development with CVS, 3rd Edition*, The Coriolis Group, 2000, <http://cvsbook.red-bean.com/cvsbook.html>
- [6] *The PHP Manual*, Gabor Hojtsy, 2005, <http://www.php.net/manual/en/>