

I. INTRODUCTION

Ce rapport présente les aspects techniques concernant les applications développées lors de mon stage à l'ULP Multimédia. Pour des informations plus générales, veuillez vous référer au rapport de stage.

Note :

Les termes définis dans le glossaire sont marqués en bleu lors de leur première apparition dans le rapport.

II. TABLE DES MATIERES

I. Introduction.....	2
II. Table des matières.....	3
III. Technologies utilisées.....	5
A. Langages.....	5
1. XML.....	5
2. Java.....	5
3. XSLT.....	5
4. ASP.....	5
5. XHTML/CSS.....	5
B. Logiciels, API et Middlewares.....	6
1. API.....	6
a) Xerces.....	6
b) Windows Media SDK.....	6
c) RMI.....	6
d) Commons Net.....	6
2. Logiciels.....	6
a) XSLTProc.....	6
3. Middlewares.....	6
a) XML DBMS.....	6
C. Normes.....	7
1. Mpeg-7.....	7
D. Documentation.....	7
E. Divers.....	7
1. Arbres DOM.....	7
IV. Existant.....	8
A. VidéoCours : fonctionnement.....	8
V. Travail effectué.....	9
A. Portage VidéoCours en ligne / hors ligne.....	9
1. Fonctionnement.....	9
a) Situation initiale : environnement en ligne.....	9
b) Tâches à accomplir.....	9
c) Processus de transformation.....	9
d) Langages et logiciels utilisés durant le processus de portage.....	10
e) Fonctionnement et fonctionnalités des applications.....	10
(1) Interface graphique.....	10
(2) Exportation des données XML.....	10
(3) Transformation du fichier XML.....	10
(4) Récupération des fichiers distants.....	11
(5) Transformation du fichier vidéo.....	11
(6) Génération de l'interface graphique.....	11
(7) Inscription des cours exportés dans la base des exports.....	11
f) Situation finale : environnement hors ligne.....	14
B. Annotation / Hiérarchisation des diapositives d'un cours.....	14
1. Fonctionnement.....	14
2. Fonctionnement technique.....	14
a) Annotation.....	14
b) Hiérarchisation.....	15

c)	Environnement Client-Serveur	17
d)	Correspondance représentation graphique – représentation mémoire	19
(1)	Modèle	19
(2)	Ordonnancement des noeuds	20
(3)	Type de noeuds	21
e)	Conception « Modèle – Vue – Contrôleur »	21
f)	Glisser-Déposer.....	23
3.	Sécurité	23
C.	Mise en ligne des cours.....	24
1.	Fonctionnement	24
2.	Fonctionnement technique.....	24
3.	Sécurité	25
VI.	Conclusion	26
VII.	Glossaire.....	27
VIII.	Bibliographie	30
A.	Livres	30
B.	Internet	30
IX.	Table des illustrations	31

III. TECHNOLOGIES UTILISEES

Un nombre relativement important de technologies différentes a été utilisé durant mon travail. Cette partie va les présenter succinctement.

A. Langages

1. XML

(eXtended Markup Language)

XML est un [méta-langage](#) créé par le web consortium (W3C) et finalisé en 1997. Il s'agit en fait d'un sous-ensemble de [SGML](#). XML est désigné pour être le successeur de HTML. Il est actuellement de plus en plus utilisé comme format standard d'échange de données.

2. Java

Java est un langage de programmation objet développé par Sun. Sa syntaxe est dérivée de celle du C++. Le Java est un langage multi-plateforme, c'est-à-dire que les programmes, une fois écrits, peuvent s'exécuter indifféremment sous différents environnements. Pour ce faire, les fichiers sources Java ne sont pas compilés directement en langage machine, mais en bytecode (fichiers Class). Ces fichiers seront identiques quel que soit l'environnement utilisé. A l'exécution du programme, ils seront interprétés par une machine virtuelle Java, qui elle, va varier en fonction de l'environnement.

Java étant un langage relativement jeune, il a été conçu dès le départ pour intégrer des fonctionnalités sécurité et réseau avancées. De plus, il bénéficie d'une API très vaste, permettant au programmeur de réaliser facilement un grand nombre de tâches.

3. XSLT

(eXtensible Style Language Transformation)

C'est un langage qui permet de transformer un document XML. Le document produit peut être un document XML, XHTML, texte, ou théoriquement de n'importe quel format.

4. ASP

(Active Server Page)

Langage de script côté serveur développé par Microsoft qui permet de générer dynamiquement des pages web. L'utilisation la plus répandue d'un tel langage est l'interfaçage avec une base de données pour y récupérer texte, images, documents.

5. XHTML/CSS

(eXtended HyperText Markup Language / Cascading Sheet Style)

XHTML est une reformulation de HTML (HyperText Markup Language) suivant la syntaxe XML. Couplé à CSS, il permet de ne décrire que le contenu dans une page web. Les informations de présentation relèvent alors uniquement de la feuille de style CSS.

B. Logiciels, API et Middlewares

1. API

a) Xerces

Xerces est un [parseur](#) XML implémentant DOM et SAX et développé par le Apache XML Project.

b) Windows Media SDK

Le format vidéo utilisé dans VidéoCours est le format Windows Media Video de Microsoft. Le Windows Media SDK fournit un ensemble de fonctions nécessaires pour travailler avec les vidéos de ce format.

c) RMI

(Remote Method Invocation : Invocation de méthodes à distance)

Dispositif client/serveur utilisable avec le langage Java permettant d'instancier des objets et d'invoquer leurs méthodes sur un ordinateur distant.

d) Commons Net

Commons Net est une API Java développée par le Jakarta Project. Elle permet de faciliter la programmation réseau en proposant des classes pour manipuler de nombreux protocoles (http, ftp, smtp, pop3, telnet, rlogin, ...).

2. Logiciels

a) XSLTProc

XSLTProc est un processeur XSLT. C'est un programme qui permet d'appliquer des transformations XSLT sur un fichier XML. Il est basé sur la librairie libxslt.

3. Middlewares

a) XML DBMS

(eXtended Markup Language – DataBase Managment System)

XML-DBMS est un [middleware](#) permettant de transférer des données entre des documents XML et des bases de données. Il utilise pour cela un langage objet-relationnel de conversion basé sur XML. Le fichier utilisant ce langage (appelé fichier Map) permet de faire correspondre (mapper) les tables et colonnes de la base de données avec des éléments ou attributs XML correspondants. XML-DBMS utilise un pont [JDBC-ODBC](#) pour pouvoir se connecter à différents types de bases de données sans avoir à se soucier de leurs formats.

C. Normes

1. Mpeg-7

Mpeg-7 est un standard ISO développé par MPEG (Moving Picture Expert Group). Contrairement à ses prédécesseurs (Mpeg-1, Mpeg-2, Mpeg-4), Mpeg-7 n'est pas un standard de compression audio-visuelle. Egalement connu sous le nom de « Multimedia Content Description Interface » (Interface de description de contenu multimédia), Mpeg-7 est un standard servant à décrire différents types d'informations multimédia, à associer cette description au contenu et à décrire l'organisation des événements.

D. Documentation

La totalité du code Java a été commenté en utilisant la syntaxe Javadoc pour que la documentation des différentes applications puisse être générée automatiquement.

E. Divers

1. Arbres DOM

DOM (Document Object Model) est un modèle permettant de représenter et de manipuler un document XML en mémoire. Le document est alors construit sous forme d'arbre. A noter qu'il existe également SAX (Simple API For XML) qui propose une approche événementielle de manipulation des documents XML.

IV. EXISTANT

A. VidéoCours : fonctionnement

L'enseignant, muni du système de poursuite infra-rouge, enregistre son cours. Le support de cours informatique est également capturé à chaque clic de l'enseignant. La vidéo est ensuite transférée à un serveur d'encodage qui va réaliser l'indexation de la vidéo. Les **scripts** et **marqueurs** sont ajoutés au fichier vidéo. Ensuite, la vidéo est envoyée par **FTP** à un serveur de contenu (en l'occurrence, il s'agit ici d'un serveur du CINES : Centre Informatique National de l'Enseignement Supérieur). De la même manière, les diapositives sont placées sur un serveur. Finalement, le cours est référencé dans une base de données, le rendant ainsi accessible *via* une interface web.

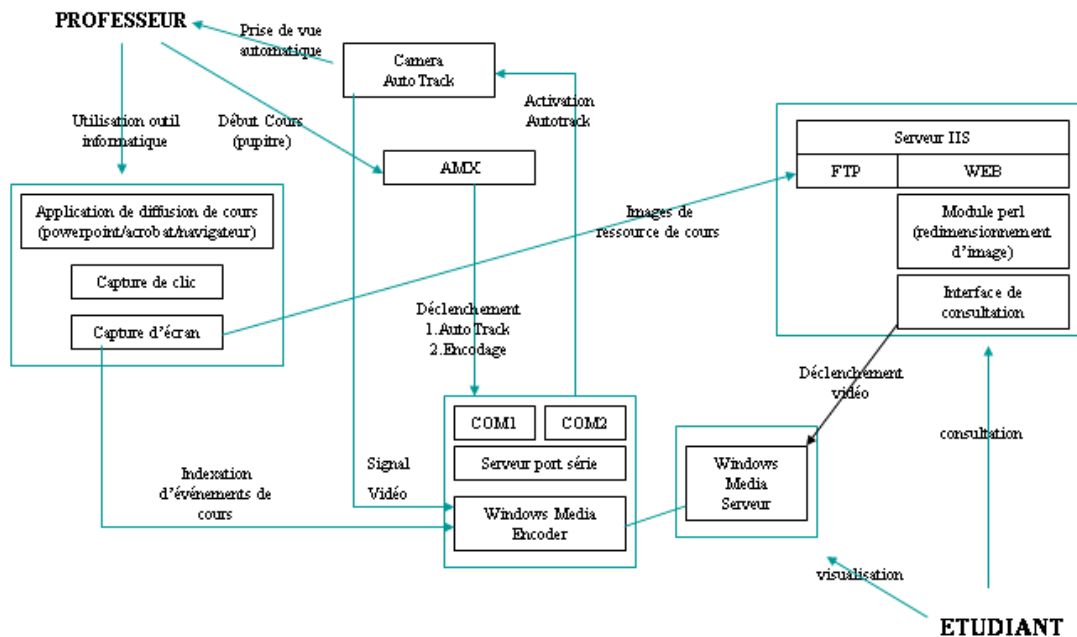


Figure 1 : Schéma de fonctionnement de VidéoCours

V. TRAVAIL EFFECTUE

A. Portage VidéoCours en ligne / hors ligne

Le but de cette application est de pouvoir exporter des cours de VidéoCours.

1. Fonctionnement

a) Situation initiale : environnement en ligne

Au départ, dans l'environnement en ligne, une base de données contient les informations concernant les différents cours (*méta-données*). Pour chaque cours, il existe une référence vers un fichier vidéo (le cours) situé sur un serveur de diffusion. Le fichier vidéo contient dans une couche script des références vers des fichiers image (les diapositives de cours) situés sur un serveur de diapositives.

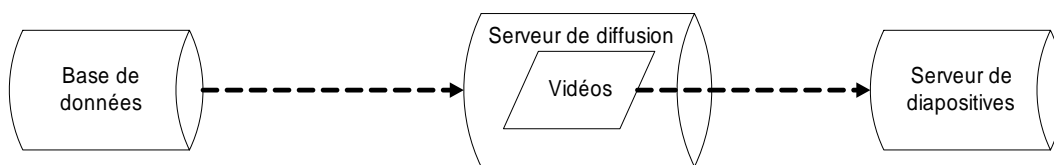


Figure 2 : Situation initiale : environnement en ligne

b) Tâches à accomplir

En observant le schéma ci-dessus, on peut dégager les différents obstacles à une utilisation dans un environnement hors ligne. Tout d'abord, l'utilisation d'une base de données est délicate. En effet, pour pouvoir utiliser celle-ci, il faut pouvoir disposer du système de gestion de base de données (SGBD) correspondant. La base de données sera donc remplacée par un fichier XML pour chaque cours. D'autre part, la base de données contient l'instance de la vidéo à utiliser, qui est représentée par une **URI**. Cette URI est **absolue** et pointe vers le serveur de vidéos. Elle doit donc être changée pour être **relative**, ce qui est plus adapté à un fonctionnement statique au sein d'un système de fichiers. Ceci sera fait par une manipulation du fichier XML obtenu à partir de la base de données. Ensuite, la couche script de la vidéo contient elle aussi des URIs absolues pointant vers les diapositives stockées sur un serveur de diapositive. Il faut donc également manipuler le fichier vidéo pour obtenir des URIs relatives. Finalement, il faut pouvoir disposer d'une interface graphique statique.

c) Processus de transformation

Voici les différentes étapes effectuées pour arriver à un environnement hors ligne :

- Sélection des cours à récupérer
- Pour chaque cours
 - o Exportation des données de la base concernant ce cours dans un fichier XML
 - o Transformation du fichier XML obtenu afin que les références vers la vidéo et les diapositives pointent vers le système de fichiers local et non vers les serveurs
 - o Récupération des fichiers distants : vidéo et diapositives

- Modification de la couche script de la vidéo afin que les références vers les diapositives pointent vers le système de fichiers local et non vers le serveur de diapositives
- Génération de l'interface graphique de consultation à partir du fichier XML
- Inscription des cours dans une base de données recensant les exports

L'ensemble de ce processus est englobé dans une interface web ASP permettant une utilisation à distance. D'autre part, un système de verrous a été mis en place pour que deux utilisateurs ne puissent pas exporter le même cours en même temps, ce qui pourrait provoquer des conflits lors de la manipulation des données créées.

d) Langages et logiciels utilisés durant le processus de portage

Etape	Langage	API - Middleware
Interface graphique Appel des différents programmes	ASP	
Exportation des données en XML	Java	XML-DBMS
Transformation du fichier XML	Java	Xerces
Récupération des fichiers distants	Java	Commons Net
Transformation de la vidéo	C/C++	Windows Media SDK
Génération de l'interface graphique	XSLT	Xsltproc
Inscription du cours dans la base des exports	ASP	

Figure 3 : Langages et logiciels utilisés pour chaque étape du processus de portage

e) Fonctionnement et fonctionnalités des applications

(1) Interface graphique

L'interface ASP permet de choisir un ou plusieurs cours parmi ceux disponibles. Tous les paramètres nécessaires au fonctionnement des différentes applications utilisées lors du processus de portage sont centralisés dans un fichier de configuration propre à l'application, permettant ainsi un déploiement et une maintenance plus aisée.

(2) Exportation des données XML

Un programme Java de transfert est livré avec XML-DBMS et permet d'utiliser ce middleware sans passer par son API. C'est donc ce programme qui est appelé directement par l'interface graphique.

Paramètres : l'identifiant d'un cours, les différents paramètres de connexion à la base de données

Sortie : un fichier XML contenant la description du cours

(3) Transformation du fichier XML

Le fichier XML est mis en mémoire sous forme d'arbre DOM. Les différentes URLs des vidéos et des diapositives sont modifiées. Finalement, l'arbre DOM est sérialisé sous forme d'un nouveau fichier XML.

Entrée : un fichier XML contenant la description d'un cours

Paramètres : les répertoires relatifs pour les vidéos et les diapositives

Sortie : un fichier XML modifié, un fichier « batch » (traitement par lot) contenant les URLs des vidéos et des diapositives avant modification

(4) Récupération des fichiers distants

Le fichier « batch » précédemment créé va permettre à cette application de récupérer toutes les ressources distantes. En réalité, seules les diapositives sont accessibles publiquement sur un serveur, tandis que les vidéos doivent être rapatriées *via* FTP. L'application va donc lire le fichier contenant les URLs des fichiers distants, et les récupérer soit par [HTTP](#) soit par FTP.

Ce programme utilise la librairie Net Commons du projet Apache Jakarta, qui fournit une riche API pour l'utilisation des protocoles réseaux. On peut s'interroger sur la nécessité d'utiliser une API externe, puisque l'API Java propose également des classes destinées à l'utilisation de HTTP et FTP. Cependant, la classe utilisée pour FTP fait partie du [package](#) `sun.*`, qui englobe toutes les classes qui ne sont pas supportées officiellement, et qui sont donc susceptibles de changer sans préavis.

(5) Transformation du fichier vidéo

Ce programme va modifier le fichier vidéo de manière à ce que les scripts qu'il contient ne comporte plus d'URLs absolues. La modification est en fait semblable à celle effectuée sur le fichier XML, mais appliquée directement à la vidéo.

(6) Génération de l'interface graphique

L'interface graphique de consultation en ligne de VidéoCours est basée sur ASP, qui génère du HTML, du [Javascript](#) pour permettre l'interactivité de la lecture, et du CSS pour la mise en forme. En ce qui concerne l'interface hors ligne, il n'est plus possible d'utiliser ASP, qui est un langage dynamique. L'interface doit donc être générée pour chaque cours afin d'arriver à un résultat statique. Le langage XSLT est donc utilisé pour produire ce résultat à partir du fichier XML de description du cours. Les pages HTML et le Javascript sont générés, tandis que les fichiers CSS sont prédéfinis et simplement recopiés dans le répertoire de destination.

(7) Inscription des cours exportés dans la base des exports

A l'issue du processus, les informations sur les cours exportés sont insérées dans une base de données, ainsi que la date des exports. Ceci permettra de savoir s'il est nécessaire d'exporter un cours donné dans le cas d'une sauvegarde.

```
vid131103084756
|
|   diaFrame.htm
|   diaThumb.htm
|   index.htm
|   mediaFrame.htm
|   vid131103084756.xml
|
+---GUI
|   |
|   |   fonctions.js
|   |   style.css
|   |   styleDiaFrame.css
|   |   styleDiaThumb.css
|   |   styleMediaFrame.css
|   |
+---images
|   |
|   |   next.jpg
|   |   pause.gif
|   |   play.gif
|   |   previous.jpg
|   |   stop.gif
|   |
+---video
|   |
|   |   AT7_131103_081239.asf
|   |
+---diapos
|   |
|   |   AT7131103081738.jpg
|   |   AT7131103081746.jpg
|   |   AT7131103082201.jpg
|   |   AT7131103082930.jpg
|   |   AT7131103083603.jpg
|   |   AT7131103083727.jpg
|   |   AT7131103084318.jpg
|   |   AT7131103084605.jpg
|   |   AT7131103085217.jpg
```

Figure 4 : Arborecence obtenue à l'issue du processus de portage

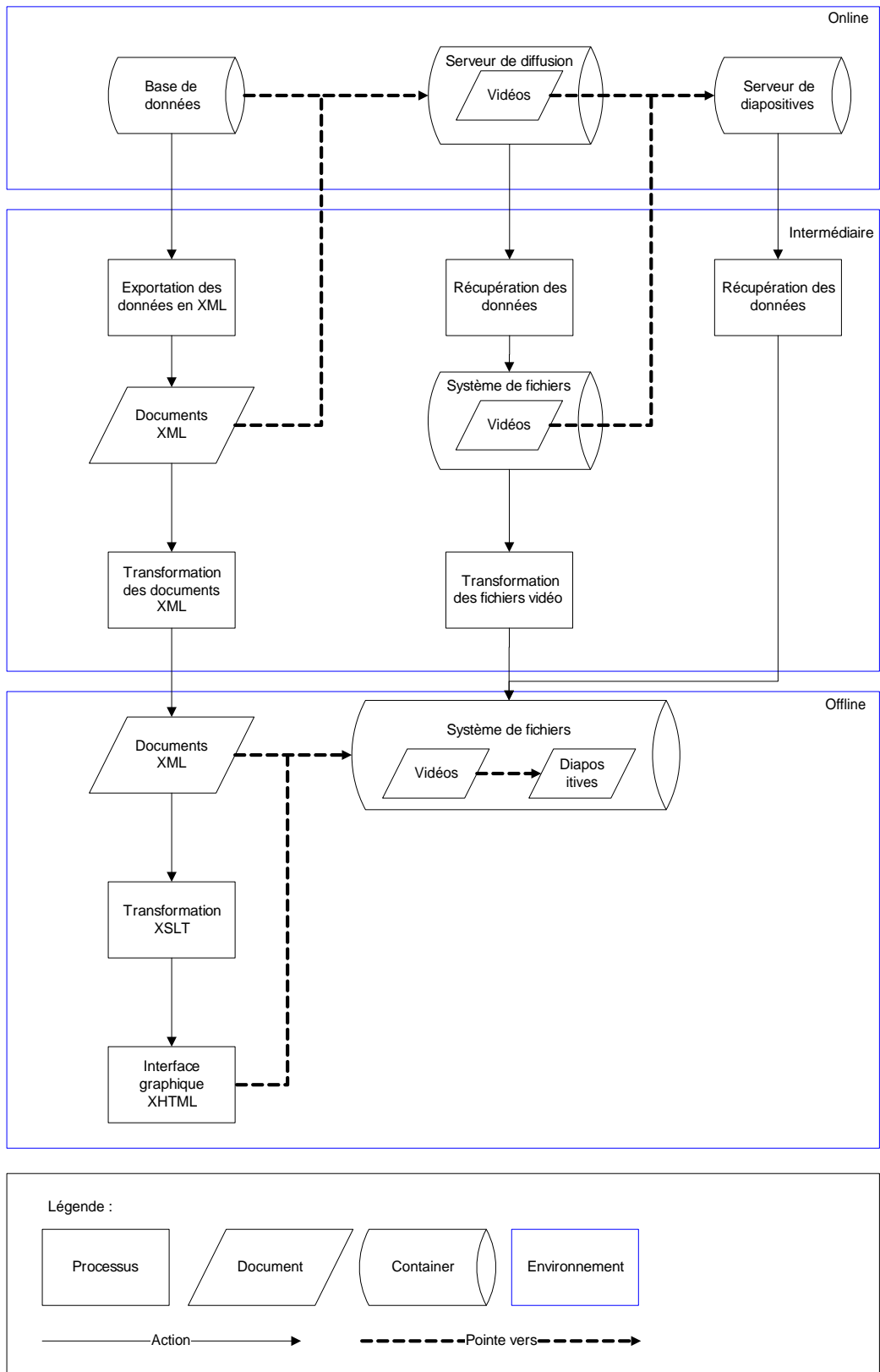


Figure 5 : Processus de portage de VidéoCours d'un environnement en ligne à un environnement hors ligne

f) Situation finale : environnement hors ligne

Nous arrivons donc dans une consultation au sein d'un environnement hors ligne : l'interface graphique statique générée pointe vers la vidéo correspondant au cours avec une URL relative. Les scripts de la vidéo pointent vers les diapositives correspondantes avec des URL relatives.

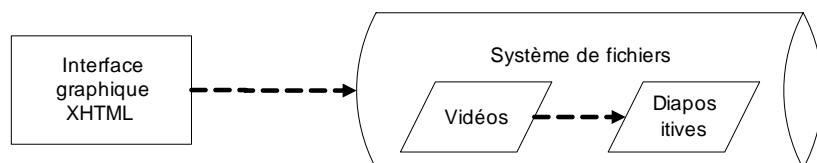


Figure 6 : Situation finale : environnement offline

Les cours exportés peuvent désormais être consultés sur un PC local ou gravés sur un CD-ROM.

B. Annotation / Hiérarchisation des diapositives d'un cours

Le but de cette application est de permettre aux enseignants de donner un titre aux diapositives de cours, et de les hiérarchiser de sorte à matérialiser le découpage du cours en différentes parties et sous-parties.

1. Fonctionnement

Le cours à éditer est sélectionné par un paramètre passé à l'applet. Les informations concernant le cours sont récupérées sous forme de document XML via XML-DBMS. Ensuite, le document XML est mis en mémoire sous forme d'arbre DOM. Une interface fait correspondre cet arbre DOM avec une représentation arborescente affichée à l'écran. Ainsi, toute modification sur la représentation se répercute sur l'arbre DOM lui-même. Finalement, l'arbre DOM modifié est sérialisé sous forme de document XML, qui est renvoyé en base.

2. Fonctionnement technique

a) Annotation

Suivant la norme Mpeg-7, une diapositive est représentée sous la forme d'un Segment.

```
<Segment id="dia1-131103084756">
  <TimeSecond>0</TimeSecond>
  <TimeHour>0</TimeHour>
  <MediaURI>video/diapos/AT7131103081738.jpg</MediaURI>
  <TimeFrame>0</TimeFrame>
  <TimeMinute>0</TimeMinute>
  <StructuredAnnotation id="annot1-131103084756">
    <WhatObject>Diapositive 1</WhatObject>
  </StructuredAnnotation>
</Segment>
```

Figure 7 : Représentation XML d'une diapositive suivant Mpeg-7

Différents éléments contenus au sein de la balise `Segment` permettent de localiser la diapositive temporellement : `TimeHour` (heures), `TimeMinute` (minutes), `TimeSecond` (secondes), et `TimeFrame` (fractions de secondes). L'élément `MediaURI` permet quant à lui de localiser la ressource associée à la diapositive. A noter que dans l'exemple précédent l'URL est relative, mais elle peut également être absolue dans l'environnement en ligne. Finalement, l'élément `StructuredAnnotation` contient une balise `WhatObject` qui est le titre de la diapositive. Ainsi, pour changer le titre d'une diapositive, il suffit de changer la valeur de l'élément `WhatObject`.

On peut s'interroger sur l'utilité de l'élément `StructuredAnnotation`. En fait, cette décomposition permet de prévoir les évolutions futures du système, pour lesquelles d'autres éléments que le titre d'une diapositive pourraient faire partie des méta-données.

b) Hiérarchisation

Implémenter la hiérarchisation a sans conteste été la partie la plus ardue de mon travail, et ceci pour différentes raisons. La première est l'idée d'arbre liée à ce concept. En effet, se limiter à une annotation des diapositives aurait été beaucoup plus simple, puisqu'une structure linéaire de type liste aurait suffi pour la représentation des diapositives. De plus, dans ce cas, un fonctionnement statique aurait été possible : la structure de données utilisée pour un composant graphique linéaire aurait pu être complètement dissociée de l'arbre DOM. C'est à dire que les données concernées auraient pu être extraites de l'arbre DOM puis y être réinjectées après modification.

Cependant, dans le cas de la hiérarchisation, modifier l'arbre DOM après avoir modifié un composant graphique représentant des données arborescentes aurait été non seulement délicat, mais assez illogique, eut égard au fait que les deux structures de données auraient été du même type. C'est pourquoi l'idée de faire correspondre directement la représentation graphique et l'arbre DOM a été choisie. Cela a cependant été la seconde difficulté de cette implémentation.

Conformément à la norme Mpeg-7, les différentes diapositives sont encapsulées dans un élément `SegmentDecomposition`, qui matérialise un ensemble au niveau du découpage :

```
<SegmentDecomposition id="segdec131103084756">
  <Segment id="dia1-131103084756">
    ...
  </Segment>
  <Segment id="dia2-131103084756">
    ...
  </Segment>
  <Segment id="dia3-131103084756">
    ...
  </Segment>
</SegmentDecomposition>
```

Figure 8 : Les éléments `Segment` sont encapsulés dans un élément `SegmentDecomposition`

Pour pouvoir matérialiser les différentes sous-parties d'un cours, des éléments `SegmentDecomposition` sont insérés, chacun matérialisant une sous-partie, à l'exception de celui de plus haut niveau, qui englobe l'ensemble du cours :

```

<SegmentDecomposition id="segdec131103084756">
  <SegmentDecomposition id="segdec1-131103084756">
    <Segment id="dia1-131103084756">...</Segment>
    <Segment id="dia2-131103084756">...</Segment>
    <Segment id="dia3-131103084756">...</Segment>
  </SegmentDecomposition>
  <SegmentDecomposition id="segdec4-131103084756">
    <Segment id="dia4-131103084756">...</Segment>
    <Segment id="dia5-131103084756">...</Segment>
    <Segment id="dia6-131103084756">...</Segment>
  </SegmentDecomposition>
  <SegmentDecomposition id="segdec7-131103084756">
    <Segment id="dia7-131103084756">...</Segment>
    <Segment id="dia8-131103084756">...</Segment>
    <SegmentDecomposition id="segdec9-131103084756">
      <Segment id="dia9-131103084756">...</Segment>
      <Segment id="dia10-131103084756">...</Segment>
    </SegmentDecomposition>
    <SegmentDecomposition id="segdec11-131103084756">
      <Segment id="dia11-131103084756">...</Segment>
      <Segment id="dia12-131103084756">...</Segment>
    </SegmentDecomposition>
  </SegmentDecomposition>
</SegmentDecomposition>

```

Figure 9 : Imbrication des éléments SegmentDecomposition

De cette manière, on peut créer un nombre infini de sous-parties au sein du cours. On notera que l'identifiant attribué à chaque `SegmentDecomposition` se voit ajouté un numéro entre la chaîne `segdec` et l'identifiant du cours, à la manière des diapositives. En effet, il est commun de décider que l'ordre des éléments d'un même niveau n'a pas d'importance au sein d'un document XML. Ce numéro permet donc de définir l'ordre des diapositives afin de les ordonner. Il remplit la même fonction pour les sous-parties. On notera que le numéro choisi pour les `SegmentDecomposition` est le numéro de la première diapositive de la sous-partie. Cela impose une limitation : une sous-partie doit contenir au moins une diapositive. En d'autres termes, elle ne peut être vide ni ne contenir que des sous-parties.

Naturellement, cette représentation arborescente ne convient pas à un stockage dans une base de données relationnelle. La hiérarchie des `SegmentDecomposition` est donc aplatie avant de mettre le document XML en base. Pour cela, un nouvel élément `Relations` est ajouté au document XML :

```

<Relations>
  <Contenant id="segdec131103084756">
    <Contenu id="segdec1-131103084756">
    <Contenu id="segdec4-131103084756">
    <Contenu id="segdec7-131103084756">
  </Contenant>
  <Contenant id="segdec7-131103084756">
    <Contenu id="segdec9-131103084756">
    <Contenu id="segdec11-131103084756">
  </Contenant>
</Relations>

```

Figure 10 : Eléments XML ajoutés pour représenter la hiérarchie des sous-parties de manière linéaire

D'autre part, tous les éléments `SegmentDecomposition` et leurs enfants sont placés au même niveau dans le document. Finalement, même si cela est superflu, pour des raisons pratiques, chaque élément `SegmentDecomposition` se voit rajouter un attribut `father` qui précise l'identifiant de son père. Nous avons ainsi une représentation linéaire de la hiérarchie des sous-parties convenant à une mise en base : (les identifiants ont été raccourcis par souci de lisibilité)

```
<SegmentDecomposition id="segdec" />
<SegmentDecomposition id="segdec1" father="segdec">
  <Segment id="dia1">...</Segment>
  <Segment id="dia2">...</Segment>
  <Segment id="dia3">...</Segment>
</SegmentDecomposition>
<SegmentDecomposition id="segdec4" father=" segdec">
  <Segment id="dia4">...</Segment>
  <Segment id="dia5">...</Segment>
  <Segment id="dia6">...</Segment>
</SegmentDecomposition>
<SegmentDecomposition id="segdec7" father="segdec">
  <Segment id="dia7">...</Segment>
  <Segment id="dia8">...</Segment>
</SegmentDecomposition>
<SegmentDecomposition id="segdec9" father="segdec7">
  <Segment id="dia9">...</Segment>
  <Segment id="dia10">...</Segment>
</SegmentDecomposition>
<SegmentDecomposition id="segdec11" father="segdec7">
  <Segment id="dia11">...</Segment>
  <Segment id="dia12">...</Segment>
</SegmentDecomposition>
```

Dans la base de données, une table `Relations` contenant deux champs `Contenant` et `Contenu` a été rajoutée ainsi qu'une colonne `Father` dans la table `SegmentDecomposition`.

De cette manière, l'arbre DOM est préparé à une sauvegarde au sein d'une base de données relationnelle.

A contrario, l'imbrication des sous-parties au sein du document XML est recréée à partir de ces informations lors de la récupération des informations de la base de données.

c) Environnement Client-Serveur

En réalité, il n'est pas possible de réaliser les traitements de conversion des données (de la base de données vers le format XML et inversement) directement au sein de l'applet, car la base de données utilisée pour stocker les méta-données est de type Microsoft Access et n'est pas un serveur de base de données à proprement parler. Ainsi, Jet, le pilote utilisé par **ODBC** pour accéder à la base ne supporte pas les accès distants.

Ces traitements sont donc effectués côté serveur, en utilisant la technologie RMI, qui permet à un programme Java d'instancier et d'utiliser des objets distants de manière relativement transparente. Au final, l'applet gère uniquement la modification du fichier XML.

RMI fonctionne de manière relativement simple pour le programmeur qui souhaite l'utiliser. D'une part, le client et le serveur partagent les interfaces des classes dont on souhaite pouvoir invoquer les objets à distance. D'autre part, seul le serveur possède l'implémentation de ces classes. Lorsque le serveur démarre, il doit inscrire les objets qu'il souhaite mettre à disposition auprès du RMI Registry, un serveur fourni avec Java. Quand le client souhaite invoquer un objet distant, il demande l'emplacement de l'objet au RMI Registry, qui lui renvoie une référence sur cet objet. Il peut ensuite l'instancier et l'utiliser.

Il s'agit bien évidemment de la partie visible de RMI. En réalité, d'autres classes sont nécessaires à Java pour pouvoir faire communiquer clients et serveurs. Cependant, comme leur utilisation est transparente pour l'utilisateur dans les dernières versions de Java, nous ne détaillerons pas leur fonctionnement ici.

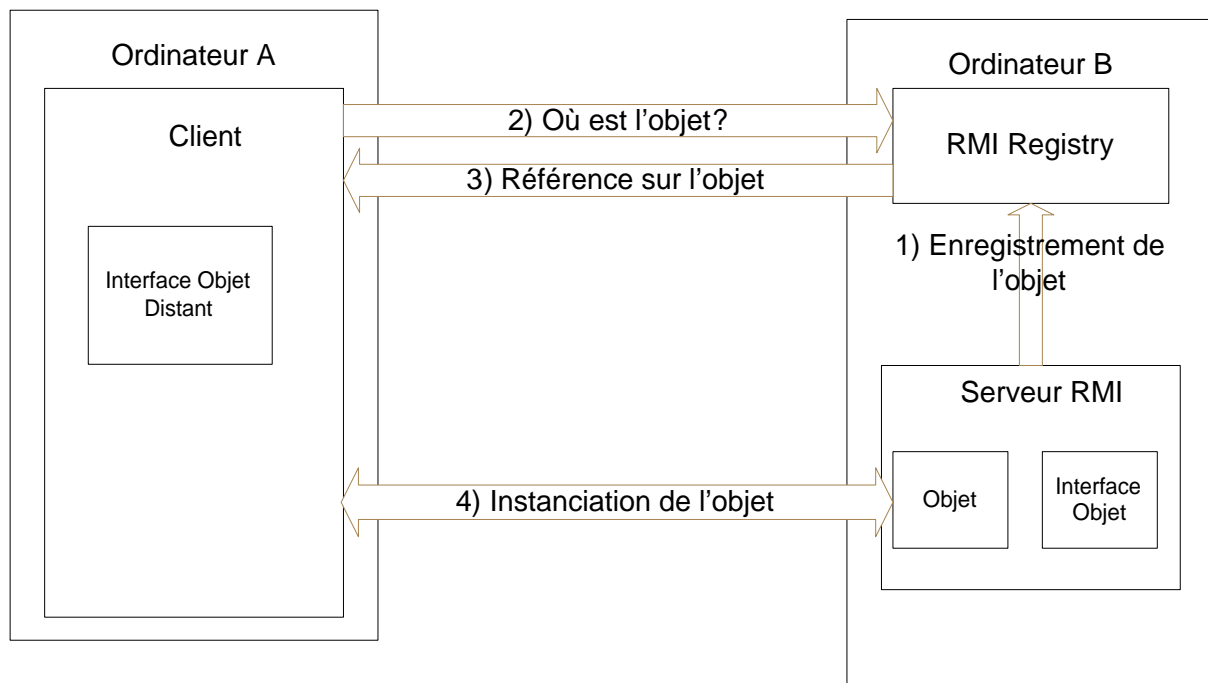


Figure 11 : Fonctionnement simplifié de Java RMI

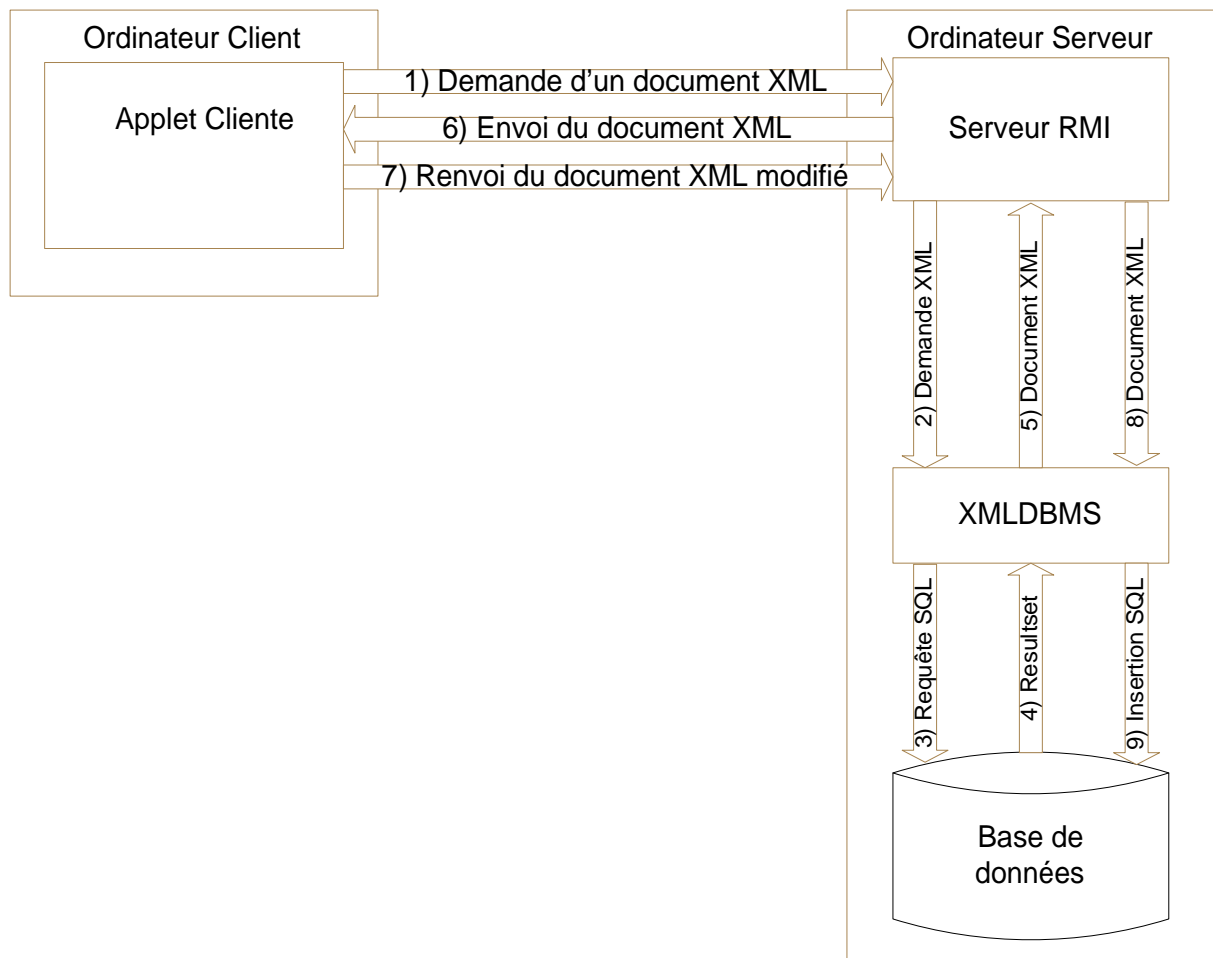


Figure 12 : Fonctionnement du processus d'annotation et de hiérarchisation des diapositives de VidéoCours

d) Correspondance représentation graphique – représentation mémoire

Le composant Java utilisé pour représenter l'arborescence des diapositives est un JTree, qui est un des composants graphiques les plus riches de Swing (l'API graphique de Java).

Dans son utilisation la plus simple, JTree propose un modèle de représentation des données ainsi qu'un type de nœud par défaut. Il est cependant possible de personnaliser complètement son fonctionnement en définissant soi-même le modèle de représentation des données ainsi que le type de nœud à utiliser. C'est l'approche utilisée pour faire correspondre la représentation graphique (le JTree) et la représentation mémoire des données (l'arbre DOM). Pour ce faire, Java fournit une interface nommée `TreeModel`, qui permet à une classe l'implémentant de servir de modèle de données à un JTree.

(1) Modèle

La classe `DomToTreeModelAdapter` implémente l'interface `TreeModel`, qui impose de redéfinir différentes fonctions afin que le JTree puisse comprendre le modèle qui lui est fourni :

- `Object getChild(Object parent, int index)` : permet d'obtenir le fils d'un nœud à un index donné.

- `Int getChildCount(Object parent)` : permet de connaître le nombre de fils d'un nœud
- `Int getIndexOfChild(Object parent, Object child)` : permet de savoir à quel index se situe le fils du père passés en paramètre.
- `Object getRoot()` : permet de connaître la racine de l'arbre
- `Boolean isLeaf(Object node)` : indique si le nœud passé en paramètre est une feuille.

C'est en définissant ces fonctions que nous allons pouvoir faire correspondre l'arbre DOM et l'arbre JTree. En effet, l'arbre DOM contient des informations que nous n'avons pas à représenter. Nous allons donc masquer dynamiquement certaines parties de l'arbre DOM pour les rendre invisible au JTree.

Prenons l'exemple de la représentation d'une diapositive (`Segment`) au sein de l'arbre DOM :

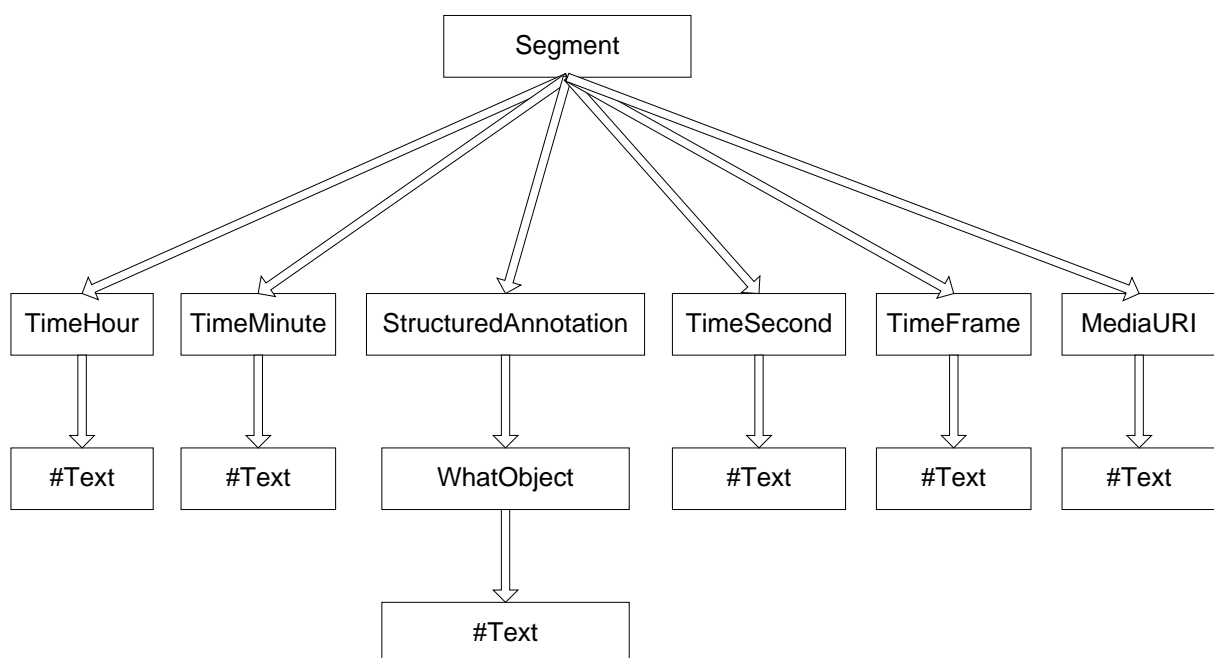


Figure 13 : Diapositive représentée sous forme d'un sous-arbre DOM

En réalité, la seule information que nous avons à afficher pour une diapositive est le nœud de type `Text` fils de `WhatObject`, qui est le titre de la diapositive. Quand le `JTree` va demander au modèle quels sont les fils de `Segment`, nous allons lui mentir en lui disant que `Segment` n'a pas de fils, mais que la valeur du nœud est en fait le titre de la diapositive.

(2) Ordonnement des nœuds

D'autre part, les diapositives peuvent apparaître dans un ordre indéterminé au sein du document XML et donc de l'arbre DOM. Leurs identifiants permettent de les trier pour les afficher dans l'ordre du cours. C'est également en mentant au `JTree` que nous allons pouvoir effectuer ce tri. En effet, quand le `JTree` demande le fils d'un nœud à un index donné, nous n'allons pas renvoyer le nœud issu de l'ordre d'apparition des nœuds dans l'arbre DOM, mais trier la liste des nœuds fils en premier lieu, puis renvoyer le bon nœud. Le tri n'est pas effectué par l'applet elle-même, mais par Java, qui propose des fonctions de tri optimisées. La

classe matérialisant les nœuds (`Diapo`) implémente pour cela l'interface `Comparable`, qui permet à Java de savoir comment trier une collection d'objets. Cependant, cette manière de procéder atteint ses limites lorsque le nombre de fils d'un nœud devient trop grand, puisque le temps de calcul utilisé pour trier devient important, et ralentit de manière significative le temps d'exécution de l'application. Pour pallier cette limitation, une optimisation a été mise en place. Celle-ci consiste à ne plus effectuer ces tris de manière totalement dynamique, mais d'effectuer le tri au départ, puis de mettre en cache le résultat. Ainsi, un nouveau tri ne sera effectué que lorsque c'est nécessaire, à savoir lorsque la structure de l'arbre change.

(3) Type de nœuds

Le type de nœud utilisé conjointement avec notre modèle de données est le type issu de la classe `Diapo`. Celui-ci encapsule un objet de type `Node` (le type utilisé pour représenter les nœuds d'un arbre DOM). Cela permet d'utiliser un type générique de nœud qui est raffiné en fonction de nos besoins. Il comporte en plus une propriété qui indique si le nœud est une diapositive ou une sous-partie. S'il s'agit d'une diapositive, son titre et l'image qui lui est associée y sont également stockés.

e) Conception « Modèle – Vue – Contrôleur »

Dans le cadre de son utilisation par défaut, le fonctionnement du `JTree` est basé sur le modèle de conception « Modèle – Vue – Contrôleur ». Cette approche permet de séparer les traitements inhérents au programme de ceux appartenant à l'interface graphique proprement dite, pour rendre l'application la plus indépendante possible et évolutive vis-à-vis des bibliothèques d'affichage. La vue correspond aux classes responsables de l'affichage. Le modèle correspond aux classes responsables des données utilisées. Le contrôleur correspond aux classes responsables des interactions avec l'utilisateur. La redéfinition du modèle du `JTree` permet de conserver cette approche.

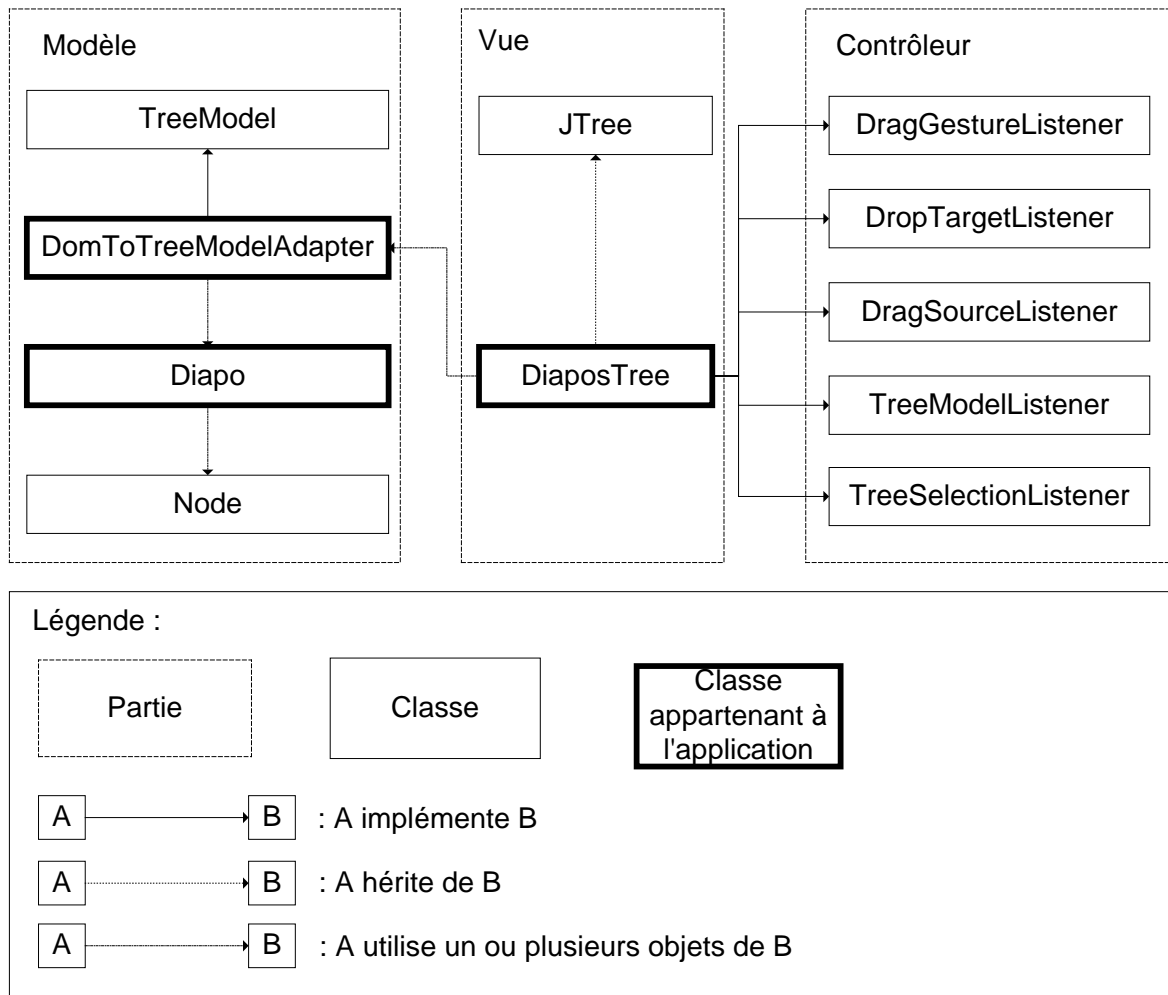


Figure 14 : Diagramme des classes représentant la conception Modèle-Vue-Contrôleur de l'applet

Voici la description des différentes classes mises en jeu dans ce schéma :

Modèle :

TreeModel est responsable de la représentation des données dans l'arbre.

DomToTreeModelAdapte implémente TreeModel pour faire correspondre l'arbre DOM et l'arbre JTree.

Diapo est le type de nœud utilisé par DomToTreeModelAdapter.

Node est un type de nœud simple, que Diapo utilise.

Vue :

JTree est le composant Java permettant d'afficher un arbre.

DiaposTree hérite de JTree pour afficher un arbre adapté à nos besoins.

Contrôleur :

DragGestureListener, DropTargetListener et DragSourceListener sont responsables des événements de glisser-déposer.

TreeModelListener est responsable des événements sur les données.

TreeSelectionListener est responsable des événements de sélection sur la vue.

f) Glisser-Déposer

Le `DiaposTree` implémente également différentes interfaces pour permettre à l'utilisateur de manipuler le cours par glisser-déposer (Drag'n'drop).

Deux actions sont possibles par glisser-déposer :

Déplacement d'une sous-partie à l'intérieur d'une autre (la sous-partie concernée est déplacée, ainsi que l'ensemble de ses fils).

Déplacement d'une ou plusieurs diapositives à l'intérieur d'une sous-partie.

L'utilisation du glisser-déplacer induit un certain nombre d'actions à effectuer pour maintenir la cohérence de l'arbre :

Il faut tout d'abord tester si l'endroit où l'on souhaite déposer un objet est valide. En effet, il n'est pas nécessaire de déplacer un objet sur lui-même. D'autre part, la destination du déplacement ne peut pas être un descendant de la source : on ne peut pas déplacer un objet dans une partie de lui-même. Et évidemment, la destination doit faire partie de l'arbre.

Ensuite, lors du déplacement d'une ou plusieurs diapositives hors d'une sous-partie, si la sous-partie d'origine devient vide suite au déplacement, elle est alors supprimée du cours, puisqu'une sous-partie ne peut pas être vide.

Finalement, après un déplacement dans l'arbre, il faut pouvoir prévenir le composant responsable de l'affichage de cet arbre que des changements ont eu lieu et qu'il faut actualiser l'affichage. C'est pourquoi il est possible de s'inscrire auprès du modèle pour être prévenu des éventuels changements : c'est ce que le `JTree` va faire auprès du `DomToTreeModelAdapter`. Ainsi, après changement, le modèle va prévenir la vue qu'un changement a eu lieu en précisant son type (des nœuds ont changé, des nœuds ont été ajoutés, des nœuds ont été supprimés, la structure de l'arbre a changée) ainsi que la partie de l'arbre concernée.

3. Sécurité

Par défaut, les Applet possèdent très peu de droits lors de l'exécution sur le poste client. En fait, leur exécution est confinée dans une `SandBox` (bac à sable). Or, l'applet d'annotation et de hiérarchisation des cours nécessite des droits supplémentaires, ne serait-ce que pour pouvoir se connecter au serveur RMI. Pour ce faire, l'applet est signée numériquement. En effet, cette procédure permet à l'utilisateur de s'assurer que le contenu de l'applet qu'il s'apprête à exécuter n'a pas été modifié et que l'auteur de l'applet est bien celui qu'il prétend être. Ainsi, lors du démarrage de l'applet, l'utilisateur est invité à choisir s'il veut confiner l'applet dans le bac à sable, ou lui donner tous les privilèges.

Note : pour le moment, l'applet est signée par l'ULP multimédia. N'importe qui peut donc signer une applet en prétendant être l'ULP Multimédia. Pour que la signature de l'applet soit sûre, la démarche serait d'acheter un certificat de signature à un tiers de confiance assurant que les applets signées à l'aide de ce certificat émanent bien de l'ULP Multimédia. Ceci est rendu possible par le fait que le certificat lui-même est signé par le tiers de confiance.

Note 2 : Il est également possible d'accorder tous les droits aux applets en mettant en place un fichier de règles de sécurité sur le poste client. Cependant, il peut s'avérer délicat d'exiger cette manipulation des utilisateurs. Le fait de signer l'applet permet de s'affranchir de cette contrainte.



Figure 15 : Boîte de dialogue de sécurité lors du chargement de l'applet VidéoCours

C. Mise en ligne des cours

Le but de cette applet est de permettre aux enseignants de mettre en ligne des cours enregistrés sur un PC local.

1. Fonctionnement

L'applet de mise en ligne des cours recherche le logiciel VidéoCours sur l'ordinateur. Si celui-ci n'est pas trouvé dans le répertoire d'installation par défaut, son emplacement est demandé à l'utilisateur. Une fois le logiciel trouvé, les différents cours présents sur la machine sont listés. L'utilisateur peut alors en choisir un et le mettre en ligne.

2. Fonctionnement technique

Lors de la création du cours, un fichier XML au format Mpeg-7 contenant les métadonnées est généré. C'est ce fichier qui servira à la mise en base des informations. Il est utilisé par l'applet pour récupérer les données nécessaires à l'affichage du cours dans la liste : auteur, date, sujet.

Cependant, ce fichier contient des URLs locales. Il sera donc transformé avant d'être envoyé au serveur. Ensuite, les diapositives, la vidéo et le fichier XML sont envoyés par FTP au serveur pour y être traités. En réalité, en fonction de leur type, les fichiers sont envoyés sur différents serveurs : les vidéos sont envoyées sur le serveur de diffusion, les diapositives sur le serveur de diapositives, et le fichier XML sur le serveur de bases de données pour qu'il y soit mis en base.

En ce qui concerne la modification du fichier XML, le programme Java utilisé lors du processus de portage en environnement hors-ligne est réutilisé.

3. Sécurité

Comme l'applet d'annotation et de hiérarchisation de VidéoCours, l'applet d'envoi a besoin de permissions plus importantes que celles disponibles dans la Sandbox. Elle est donc également signée.

VI. CONCLUSION

Le développement effectué au cours de ce stage s'est révélé être relativement riche, de par le nombre de technologies différentes, mais également de par les concepts employés.

Ainsi, la modularité d'une application est un critère décisif quand les technologies hétérogènes se multiplient. Développer de manière modulaire est l'assurance de pouvoir faire coopérer ces technologies, et un avantage pour pouvoir en intégrer de nouvelles.

D'autre part, les concepts liés à la manipulation d'arbres DOM et l'utilisation d'XML comme format d'échange de données sont en passe de devenir des standards, et semblent déjà largement utilisés. De fait, les avoir utilisés ne peut qu'être bénéfique pour mes expériences futures.

VII. GLOSSAIRE

Absolue (URL ou URI) :

Une URL ou URI est dite absolue quand le chemin qu'elle définit ne tient pas compte de l'endroit où l'on se trouve.

Exemple :

<http://serveur.ext/répertoire1/répertoire2/fichier.ext> pointera toujours vers la même ressource, quel que soit l'endroit où on se trouve.

API : Application Programming Interface.

Une API est un jeu de fonctions que l'on peut utiliser pour travailler avec un composant, une application, un langage ou un système d'exploitation.

Etendue (classe) :

Etendre une classe revient à utiliser les capacités de la classe étendue, et à en rajouter d'autres dans la classe qui étend.

FTP : File Transfer Protocol.

Protocole de transfert de fichiers.

HTTP : HyperText Transfer Protocol.

C'est le protocole utilisé par les navigateurs pour afficher les pages web.

Interface (classe) :

Implémenter une interface revient à programmer une ou plusieurs fonctions qui sont spécifiées par l'interface. L'interface permet de préciser ce qui doit être fait. La classe qui implémente l'interface spécifie comment cela est fait.

Javascript :

Langage de script côté client, qui permet au navigateur d'interagir avec les pages web.

JDBC : Java DataBase Connectivity.

Protocole utilisé pour accéder aux bases de données avec Java. Malheureusement, tous les systèmes de gestion de bases de données ne supportent pas JDBC. On est donc souvent obligé d'utiliser un pont JDBC-ODBC, qui va se charger de convertir les appels JDBC en ODBC.

Machine virtuelle :

La machine virtuelle Java est le logiciel qui permet d'exécuter le bytecode Java commun à toutes les plateformes. Les programmes Java sont donc susceptibles d'être exécutés sur n'importe quelle plateforme. C'est le « Write Once, Run Anywhere » (Ecrivez une fois, Exécutez n'importe où).

Marqueurs :

Les marqueurs permettent de découper la vidéo en séquences. On peut comparer les marqueurs d'une vidéo aux plages d'un CD audio : grâce à eux, on peut accéder directement à différentes parties du média. Dans VidéoCours les marqueurs sont associés au déclenchement des diapositives. Cela permet à l'utilisateur de piloter la lecture en fonction des diapositives.

Métadonnées :

Information sur l'information. Les métadonnées sont généralement des informations qui ne vont pas apparaître dans un document, mais qui vont permettre de le décrire.

Métalangage :

Langage permettant de définir d'autres langages. XML est un métalangage dans le sens où l'utilisateur peut définir de nouvelles balises en fonction de ses besoins.

Middleware :

Un middleware est un logiciel ou une API faisant le lien entre deux logiciels ou composants. Dans une architecture client-serveur, le middleware est souvent l'intermédiaire entre les applications et le transport des données. On parle alors d'architecture 3-tiers.

Il n'existe pas de traduction officielle pour middleware, c'est pourquoi le terme anglais est utilisé dans ce rapport. Une traduction possible peut être « logiciel médian ».

ODBC : Open Database Connectivity.

Protocole standard d'accès aux bases de données relationnelles reposant sur SQL.

Package :

Ensemble de fonctions ou de classes ayant un thème commun. L'API Java est découpée en différents packages, correspondant aux différents domaines abordés (interface graphique, mathématiques, entrées/sorties...). On parle également de « paquetage », ou plus généralement, de bibliothèque.

Parseur :

Un parseur est un programme permettant de parcourir un fichier (généralement, il s'agit de code source) pour faire l'analyse de son contenu. On parle également d'analyseur syntaxique.

Relative (URI ou URL) :

Une URI ou URL est dite relative quand le chemin est défini à partir de l'endroit courant.

Exemple : repertoire1/repertoire2/fichier.ext

Scripts :

Les scripts au sein d'une vidéo permettent de déclencher une action à un moment donné de la vidéo. Ils peuvent être de différents types. Les scripts utilisés au sein de VidéoCours sont de type URL : l'action engendrée est alors d'ouvrir la ressource pointée par la valeur du script. Il s'agit des diapositives de cours dans ce cas.

SDK : Standard Development Kit

Un SDK est un ensemble d'éléments (logiciels, API, ...) permettant à un programmeur d'utiliser un langage ou une API donnée.

SGML : Standard Generalized Markup Language

Langage à balises utilisé pour l'écriture de documents. Il est plus général, plus ancien et plus complexe que HTML et XML.

SQL : Simple Query Language.

Langage qui permet d'interroger une base de données relationnelle.

URI : Uniform Resource Identifier.

Comme l'URL, l'URI permet d'accéder à une ressource, mais son champ d'application est plus vaste. Elle permet notamment d'accéder à une ressource d'un système de fichiers.

URL : Uniform Resource Locator.

Chaîne de caractères permettant d'accéder à une ressource sur Internet. Celle-ci est formée suivant la syntaxe <protocole>://<serveur>/<repertoires>/<fichier>

VIII. BIBLIOGRAPHIE

A. Livres

Au cœur de Java 2 : Fonctions avancées / Cay S. Horstmann & Gary Cornell ; trad. Nathalie Le Guillon De Penanros.- Paris : CampusPress, 2002 (CampusPress Référence)

C++ / Kyle Loundon ; trad. Vivian Girel.- Paris : O'Reilly, 2003 (Précis & concis).

Introduction to Mpeg-7 : Multimedia Content Description Interface / B.S Manjunath, Philippe Salembier, Thomas Sikora.- Chichester : Wiley, 2002

B. Internet

Java :

Documentation Java : <http://java.sun.com/docs>

VidéoCours :

ULP Multimédia : <http://ulpmultimedia.u-strasbg.fr>

VidéoCours : <http://videocours.u-strasbg.fr>

Windows Media SDK & ASP :

Microsoft Developer Network : <http://msdn.microsoft.com/>

Xerces :

Apache XML project : <http://xml.apache.org/>

XML :

<XML>fr : <http://xmlfr.org/>

W3C : <http://www.w3.org/XML/>

XML-DBMS :

XML-DBMS : <http://www.rpbouret.com/xmldbms/>

XML-DBMS Mailing List : <http://groups.yahoo.com/group/xml-dbms/>

XSLT :

W3C : <http://www.w3.org/TR/xslt>

XSLTProc : <http://xmlsoft.org/XSLT/>

Commons Net :

Jakarta Commons Net : <http://jakarta.apache.org/commons/net/>

IX. TABLE DES ILLUSTRATIONS

Figure 1 : Schéma de fonctionnement de VidéoCours	8
Figure 2 : Situation initiale : environnement en ligne	9
Figure 3 : Langages et logiciels utilisés pour chaque étape du processus de portage.....	10
Figure 4 : Arborescence obtenue à l'issue du processus de portage	12
Figure 5 : Processus de portage de VidéoCours d'un environnement en ligne à un environnement hors ligne.....	13
Figure 6 : Situation finale : environnement offline.....	14
Figure 7 : Représentation XML d'une diapositive suivant Mpeg-7	14
Figure 8 : Les éléments Segment sont encapsulés dans un élément SegmentDecomposition.....	15
Figure 9 : Imbrication des éléments SegmentDecomposition	16
Figure 10 : Eléments XML ajoutés pour représenter la hiérarchie des sous-parties de manière linéaire	17
Figure 11 : Fonctionnement simplifié de Java RMI	18
Figure 12 : Fonctionnement du processus d'annotation et de hiérarchisation des diapositives de VidéoCours	19
Figure 13 : Diapositive représentée sous forme d'un sous-arbre DOM.....	20
Figure 14 : Diagramme des classes représentant la conception Modèle-Vue-Contrôleur de l'applet.....	22
Figure 15 : Boite de dialogue de sécurité lors du chargement de l'applet VidéoCours	24

