GUERROUDJ David
BERTOLINI Alban

RAPPORT XML

Sommaire

Sommaire **2**

MAKEFILE - DAVID **3**

XML **4**

DTD **5**

Organigramme simplifié de la DTD : 5

Les identifiants : 6

XSD **7**

Les indexes : 7

Les restrictions ajoutées : 7

XSL - DAVID **8**

XQUERY - DAVID **9**

JAVA **11**

Algorithme 11

8. TIDY **12**

1. MAKEFILE - DAVID

Notre makefile se compose de la façon suivante :

all	Execute dans l'odre : clean -> mkdir_web -> mkdir_traces -> dtd -> noEnt -> xsd -> web -> tidy -> xq -> javaNom
mkdir_traces	Permet de créer le répertoire traceExecution qui contient la trace d'exécution de toutes nos fonctions.
mkdir_web	Permet de créer le répertoire www dans lequel toutes les pages HTML se trouvent.
dtd	Permet d'exécuter la fonction xmllint et de créer une trace.
noEnt	Permet de formatter notre xml en y enlevant les entités.
xsd	Permet d'exécuter la fonction xmllint -valid -schema et de générer un schéma.
web	Permet d'exécuter la fonction xsltproc et de générer des pages Web à partir du XML dans le répertoire www.
tidy	Permet de vérifier la validité des pages web en exécutant la commande tidy et de créer une trace. Exécute un script en Bash, qui permet d'appliquer la commande tidy à tous les fichiers .html du répertoire www.
xq	Permet d'utiliser une requête XQ et de créer une trace. Se déplace d'abord dans le répertoire XQ et utilise un autre makefile qui compile et exécute notre code en XQ.
javaNom	Permet d'utiliser un programme en JAVA et de créer une trace. Se déplace d'abord dans le répertoire JAVA et utilise un autre makefile qui compile et exécute notre code en JAVA.

2. XML

Nous avons opté pour la structure de XML suivante :

```
<master>
	<intervenants>
		Contient toutes les informations sur un intervenant.
	</intervenants>
	<enseignements>
		Contient toutes les informations sur un enseignement.
	<enseignements>
	<blocks>
		<block>
			Block contenant un ID et une liste de références vers un
enseignement.
		</block>
	</blocks>
	<semestres>
		Contenant la liste de chaque semestre
		<semestre>
			Contenant un titre et une référence vers des blocks.
		</semestre>
	</semestres>
	<specialites>
		<specialite>
			Contenant toutes les informations des différentes spécialités.
		</specialite>
	</specialites>
	<parcours>
		<parcour>
			Contenant toutes les informations des différents parcours.
		</parcour>
	</parcours>
</master>
```

3. DTD

Organigramme simplifié de la DTD :

(Les éléments HTML et PCDATA se trouve dans les feuilles de l'organigramme)

master				
	intervenants			
		identifiant		
		nom		
		mail*		
		siteWeb		
		telephone		
		adresse?		
		etablissement		
		autre?		
			titre	
			p	
			ANY*	
	enseignements			
		nom		
		identifiant		
		nombreCredit		
		professeurs?		
		resume		
		pre-requis?		
		plan?		
		etablissement*		
		ref-intervenant*		
		adresse*		
		autre?		
	blocks			
		block+		
			ref-enseignement+	
	semestres			
		semestre*		
			titre	
			ref-blocks+	
	specialites			
		specialite*		
			nom	
			identifiant	
			code	
			finalite?	
			etablissement	
			responsable	
			description	

			competences	
			connaissances?	
			ref-semestres+	
			politiqueDesStages	
			aspects?	
			modalites	
			conditionAdmission	
			debouches	
			pousuitesEtudes	
	parcours			
		parcour*		
			nom	
			description?	
			ref-specialite+	
			autre*	
				titre
				p
				ANY*
	description			

Les identifiants :

Les éléments **Intervenant**, **Enseignement**, **Specialite**, **Parcour**, **Semestre** et **Block** possèdent un @id en attribut.

4. XSD

Les indexes :

- INTERVENANT
- ENSEIGNEMENT
- SPECIALITE
- SEMESTRE
- BLOCK

Les restrictions ajoutées :

- Restriction à deux semestres dans une spécialité
- Une adresse peut contenir maximum 100 caractères. Les blancs de début de fin de chaîne sont supprimés.
- Un nom peut contenir de 2 à 80 caractères . Les blancs de début de fin de chaîne sont supprimés.
- Un titre peut contenir de 1 à 15 caractères . Les blancs de début de fin de chaîne sont supprimés.
- Les crédits sont des entiers positifs.
- Un numéro de téléphone doit matcher notre regex(Accepte les +33 et les numéro séparés d'un espace ou non).
- Un type ID doit matcher notre regex (Accepte majuscule et underscore).
- Un email doit matcher notre regex (Accepte de la forme blabla@nomdedomaine).
- Un site web doit matcher notre regex(Accepte si l'entête comporte http://).

5. XSL - DAVID

Notre document XSL est unique, il permet de générer toute l'arborescence HTML de notre site WEB.
Il se compose de la façon suivante :

`<xsl:document href="www/parcours/listeEnseignements.html">` Permet de créer une page Web qui liste toutes Unités d'enseignement du site.
`<xsl:document href="www/parcours/listeIntervenants.html">` Permet de créer une page Web qui liste tous les Enseignants du site.
`<xsl:for-each select="//intervenant">` Permet de créer une page Web par intervenant du Site.
`<xsl:for-each select="//specialite">` Permet de créer une page Web par spécialité du Site.
`<xsl:for-each select="//enseignement">` Permet de créer une page Web par enseignant du Site.
`<xsl:for-each select="//parcour">` Permet de créer une page Web par parcours du Site.
`<xsl:template name="menu">` Permet de générer un menu. Cette fonction prend en argument le lien de la page courante afin de générer un menu dynamique pour chaques pages.
`<xsl:template name="genererSpecialite">` Permet de générer les spécialités
`<xsl:template match="ref-intervenant">` Permet de générer les références intervenants
`<xsl:template match="intervenant">` Permet de remplir le contenu de chaque page Intervenant.
`<xsl:template match="enseignement">` Permet de remplir le contenu de chaque page Enseignement.
`<xsl:template name="genererSemestre">` Permet de générer les semestres du Site.
`<xsl:template name="afficherNomEnseignement">` `Fonction permettant d'afficher le nom d'un enseignant en fonction de son ID.`
`<xsl:template match="specialite">` `Permet de générer la page Spécialité du Site.`

6. XQUERY - DAVID

Exécute une requête qui affiche la liste de tous les enseignants ainsi que la liste de toutes les UE qu'il enseigne. N'ayant malheureusement pas trouvé la référence de tous les enseignants sur les UE du site internet du Master de Luminy, nous avons manuellement entrées des références vers des enseignants aléatoire afin de tester notre requête.

Algorithme :

```
Pour chaque nom d'intervenant dans <Intervenant>
  On ordonne par intervenant
  On retourne la liste des noms d'intervenants
  Pour chaqu'une des matières dans <enseignement>
    On récupère la référence de l'enseignement associé.
    On ordonne par id
    Si l'id de l'intervenant est égal à celui de la référence.
      Alors on retourne le nom de l'enseignement.
    FinSi
    Sinon
      On ne fait rien
    FinSinon
  FinPour
FinPour
```

Afin d'appliquer cet algorithme, nous utilisons la technologie XPath2.0.
Nous faisons appel à :

- Des Boucles FOR
- Des Conditions SI
- Des Déclarations de Variables locales : let $variable
- La fonction **fn:distinct-values** qui permet de ne sélectionner que des valeurs distinctes
- la fonction **fn:compare** qui permet de comparer deux à deux des valeurs contenues dans des variables
- La fonction **string** qui permet de caster le contenu d'une variable en string.

Nous utilisons des déclarations d'option afin de créer un document xHTML valide :

```
declare namespace saxon="http://saxon.sf.net/";
 declare option saxon:output "method=xml";



 declare option saxon:output "doctype-public=-//W3C//DTD XHTML 1.0 Strict//EN";



 declare option saxon:output "doctype-system=http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd";



 declare option saxon:output "omit-xml-declaration=no";


declare option saxon:output "indent=yes";
```

Nous n'avons pas de soucis, tout fonctionne correctement dans cette partie.

7. JAVA

Algorithme

```
noeudEnseignement = false
                noeudNom = false
                Tant qu'il existe des noeuds
                        si noeud courant == <enseignement>
                                noeudEnseignement = true
                        si noeud courant == <nom> && noeudEnseignement == true
                                noeudNom = true
                        si noeud noeudEnseignement == true && noeudNom = true
                                charger noeud dans notre arbre XML
                        noeudEnseignement = false
                                noeudNom = false
```

Nous n’avons pas de soucis, tout fonctionne correctement dans cette partie.

8. TIDY

Afin de pouvoir exécuter Tidy sur l'ensemble de nos fichiers, nous avons opté pour la création d'un script bash qui explore chaque répertoire du répertoire WWW et y applique notre commande.
Voici le code de notre Script :

```
#/!bin/bash
compt=0
for FILE in `find ../www/ -name "*.html" `;
do
    execPath=$PWD/${FILE}
    executionTidy="tidy -q --show-warnings false -utf8 $execPath"
    $executionTidy
    compt=$((compt+1))
done
compt=$((compt+1))
echo 'Fichiers tidy check : ' $compt > ../traceExecutions/nbFichiersTidy.txt
```