

GESTIONI CORSI XML

Autore Ing. Antonio Di Fluri

In collaborazione con



www.futuresoftware.it

info@futuresoftware.it

Sommario

DESCRIZIONE.....	3
STRUMENTI UTILIZZATI.....	3
APPROCCIO AL PROBLEMA.....	4
1. ANALISI E SPECIFICA DEI REQUISITI.....	5
• Requisiti Funzionali	5
• Requisiti Non Funzionali	5
• Use Case.....	5
Figura 1: Use Case Diagram	5
2. PROGETTAZIONE.....	6
Figura 2: View of Packages	6
Figura 3: Collaboration Diagram	7
Figura 4: Sequence Diagram	7
3. IMPLEMENTAZIONE.....	8
Figura 5: package structure	8
Figura 6: package print.....	8
Figura 7: package event.....	8
Figura 8: package constants	9
Figura 9: package xml2pdfcourse	9
Figura 10: package parse.....	10
4. TESTING E VALIDAZIONE	11
Figura 11: frame principale	11
Figura 12: parser e visualizzazione xml	11
Figura 13: conversione da xml a pdf.....	11

DESCRIZIONE

Il software da sviluppare consiste in una applicazione che consenta la creazione e la gestione di una presentazione di un corso universitario tramite linguaggio xml.

La presentazione di un corso universitario ha alcune informazioni come il nome della facoltà, il nome del professore, gli argomenti trattati ecc. Tali informazioni dovranno essere memorizzate in un file xml con la possibilità di essere all'occorrenza modificate e, quindi, salvate sempre in un file xml.

Oltre alla gestione sopra descritta, il software dovrà convertire il contenuto del file xml in formato pdf.

STRUMENTI UTILIZZATI

- Librerie Xerces-2.5.0
- Librerie Fop-0.20.5
- Librerie Log4j-1.2.13
- Librerie Apache-Ant-1.6.5
- XmlSpy
- XmlStyle
- Eclipse 3.1.2
- NetBeans 5.0
- JIndent

APPROCCIO AL PROBLEMA

Il primo passo è stato quello di creare un file xml che potesse rispecchiare una generica presentazione di un corso universitario. A tale scopo è stato usato il programma XmlSpy che consente, a partire dal file xml, di creare il corrispondente schema xml ossia il file xsd. Nel passo successivo sono state progettate delle classi java che, come il file xml, rappresentano tutte le caratteristiche della presentazione di un corso universitario. L'ultimo passo consiste nel mappare i dati del file xml nelle classi java discusse sopra e ciò avviene tramite una classe che esegue il parser del file xml.

Infine per convertire il file xml in pdf è stato creato un file di stile secondo le specifiche FO (formatting object) e una classe java che, a partire dal file xml e dal file di stile, genera il file pdf.

Esempio di file creati:

- corso.xml:dati presentazione di un corso universitario
- corso.xsd:schema presentazione di un corso universitario
- corso.xsl:stile per corso.xml per la presentazione su browser
- corso-fo.xsl:stile FO per corso.xml per la conversione in pdf

Nel dettaglio, questo documento è strutturato secondo l'approccio utilizzato durante lo sviluppo:

- Analisi e specifica dei requisiti
- Progettazione
- Implementazione
- Testing e validazione

1. ANALISI E SPECIFICA DEI REQUISITI

- Requisiti Funzionali

RF1: il sistema deve eseguire il parser del file xml

RF2: il sistema deve memorizzare i dati ottenuti dal parsing del file xml

RF3: il sistema deve modificare i dati ottenuti dal parsing del file xml

RF4: il sistema deve salvare in un file xml i dati modificati

RF5: il sistema deve convertire il file xml in un file pdf

- Requisiti Non Funzionali

RNF1: il sistema deve salvare il file xml con codifica ISO-8859-1

RNF2: il sistema deve eseguire la validazione dello schema xml

- Use Case

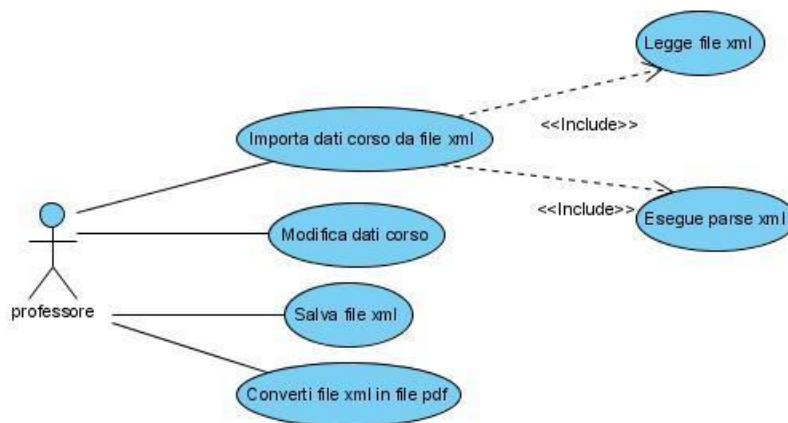


Figura 1: Use Case Diagram

2. PROGETTAZIONE

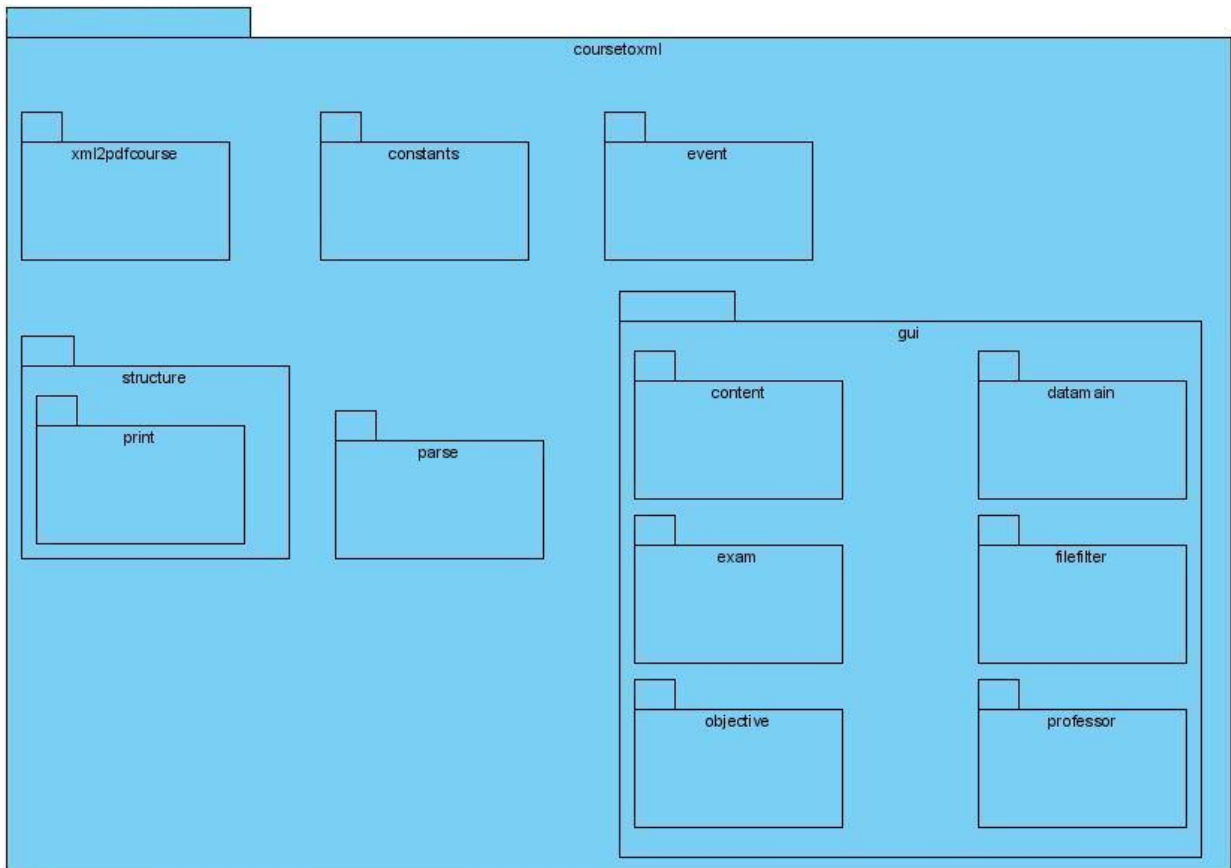


Figura 2: View of Packages

Il package “coursetoxml.structure” contiene le classi che descrivono tutte le caratteristiche del corso.

Il package “coursetoxml.structure.print” contiene la classe che permette di stampare in un file xml il contenuto delle classi del package sopra descritto.

Il package “coursetoxml.gui” e i sotto-package contengono le classi per la definizione dell’interfaccia grafica del software sviluppato.

Il package “coursetoxml.parse” contiene le classi che eseguono il parser del file xml.

Il package “coursetoxml.xml2pdfcourse” contiene la classe che converte il file xml in file pdf.

Il package “coursetoxml.event” contiene le classi che definiscono gli eventi per la gui.

Il package “coursetoxml.constants” contiene le classi che definiscono le principali costanti del software.

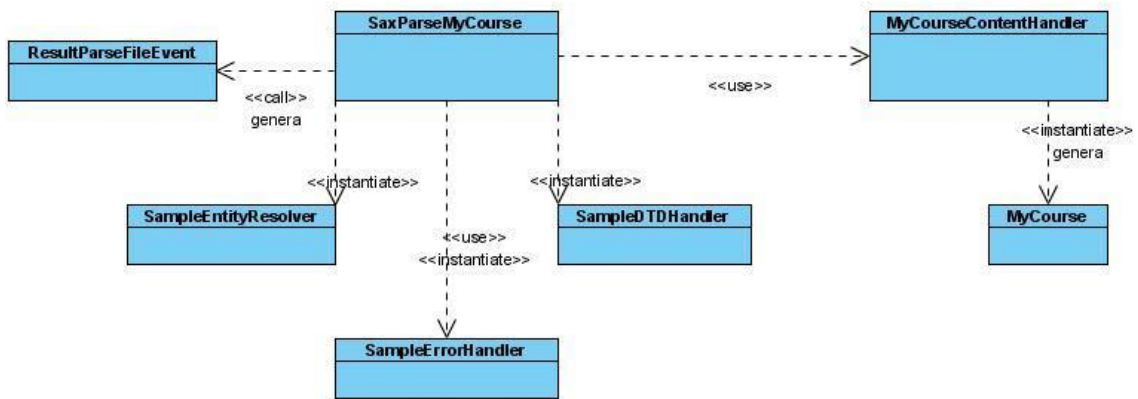


Figura 3: Collaboration Diagram

La classe SaxParseMyCourse è la classe più importante perché avvia il parser del file xml e crea il la classe MyCourse che contiene tutti i dati estrapolti dal file xml.

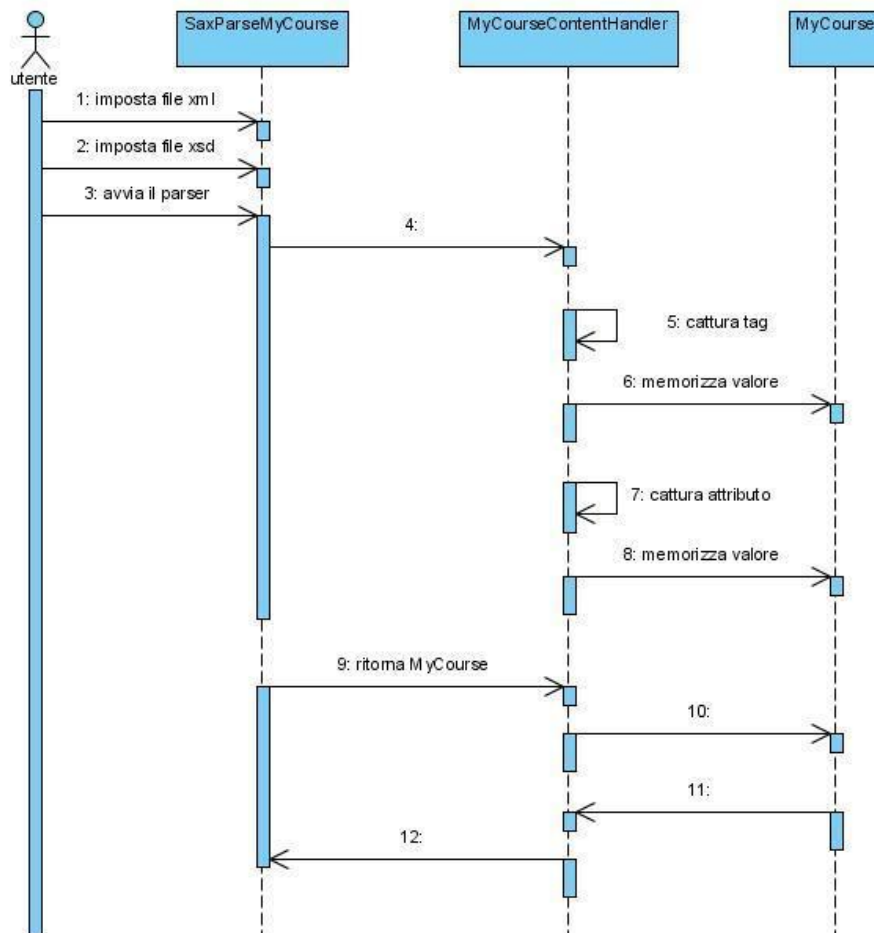


Figura 4: Sequence Diagram

3. IMPLEMENTAZIONE

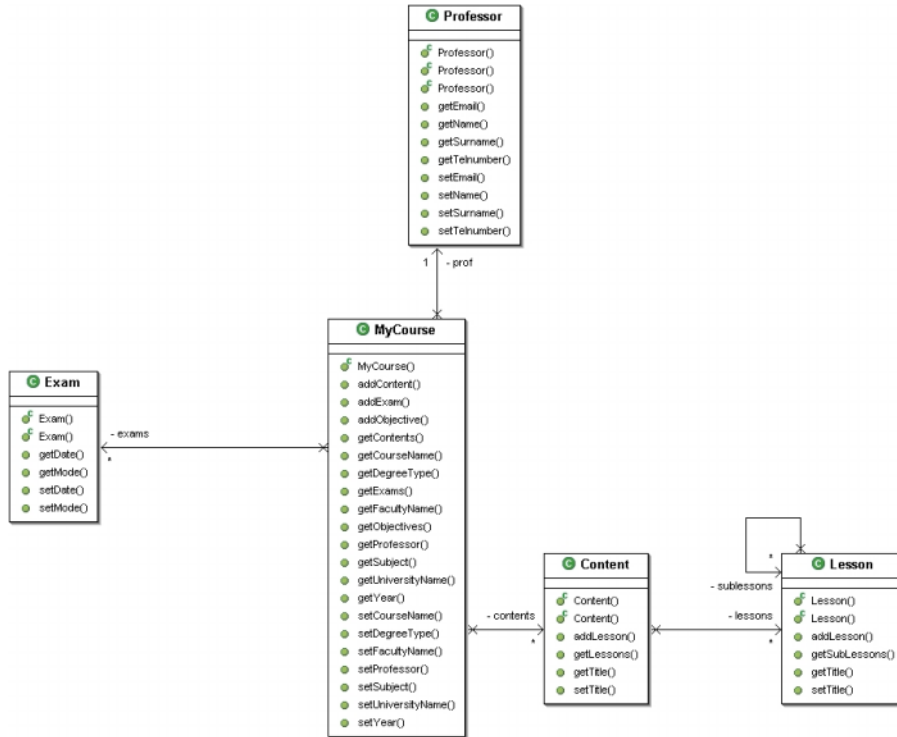


Figura 5: package structure

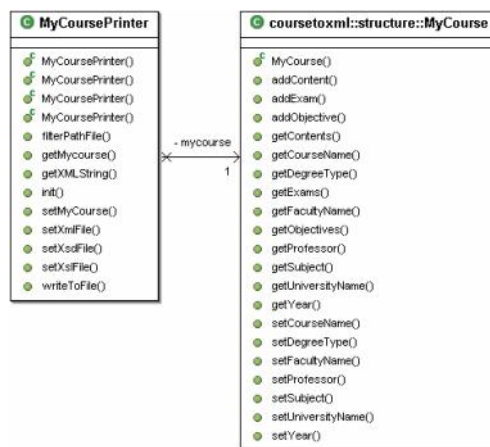


Figura 6: package print

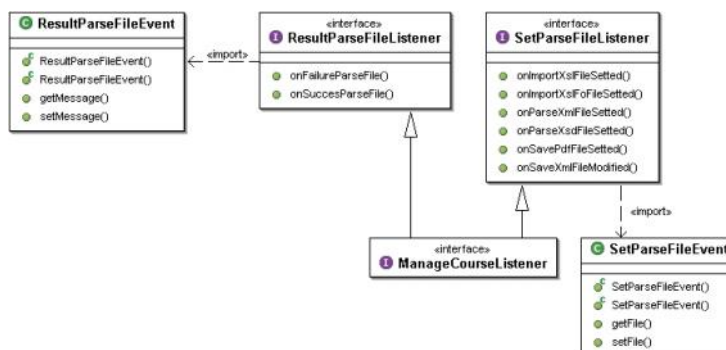


Figura 7: package event

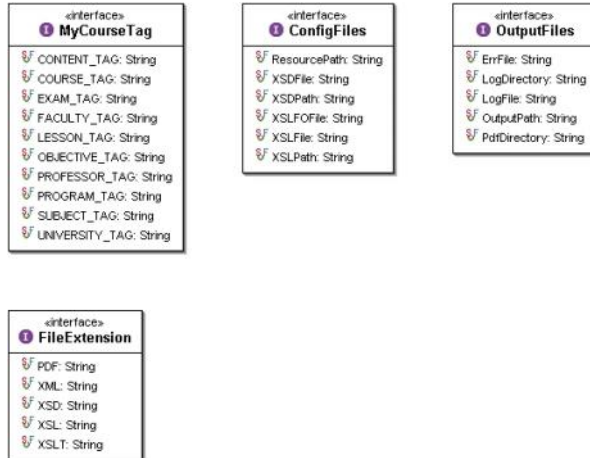


Figura 8: package constants

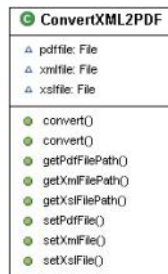


Figura 9: package xml2pdfcourse

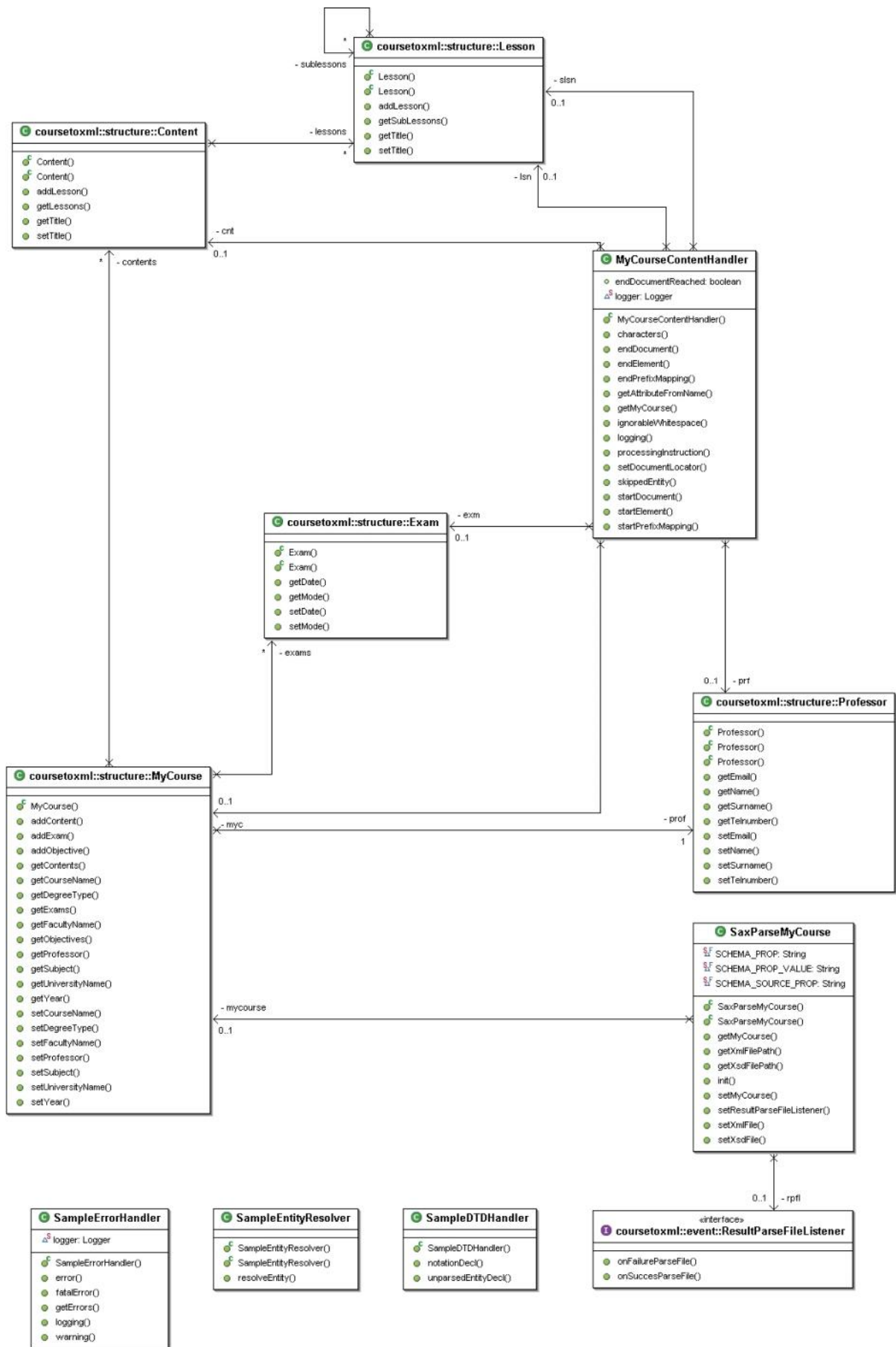


Figura 10: package parse

4. TESTING E VALIDAZIONE

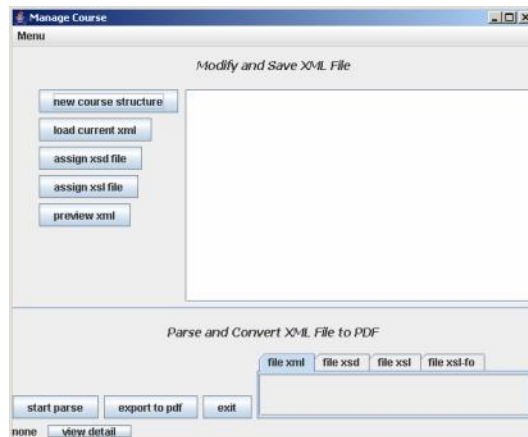


Figura 11: frame principale

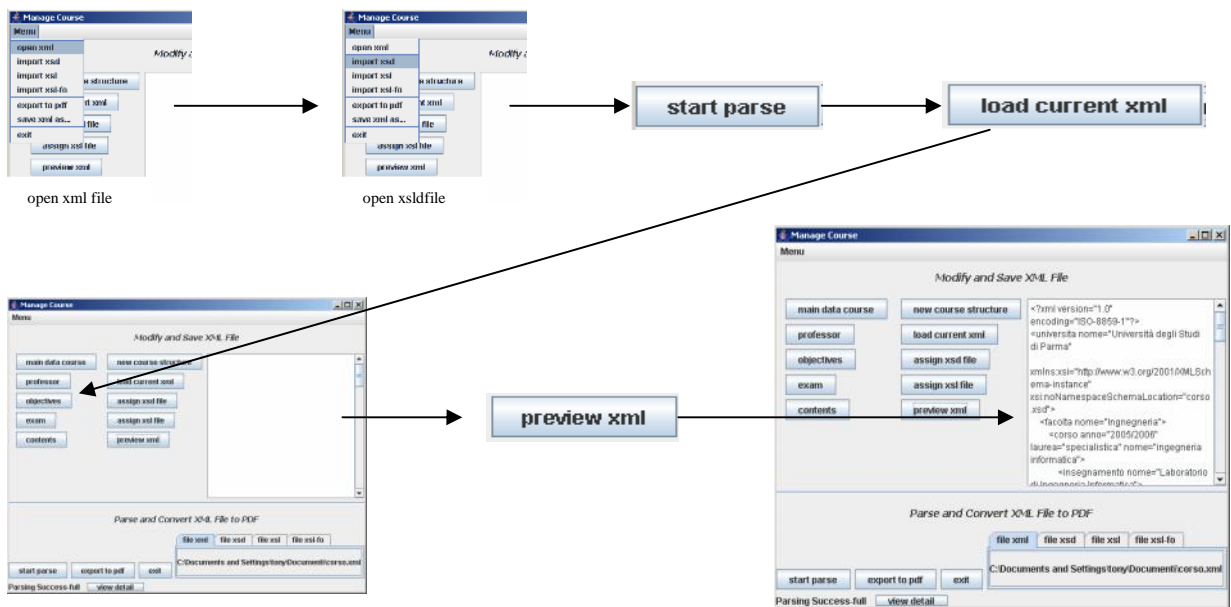


Figura 12: parser e visualizzazione xml

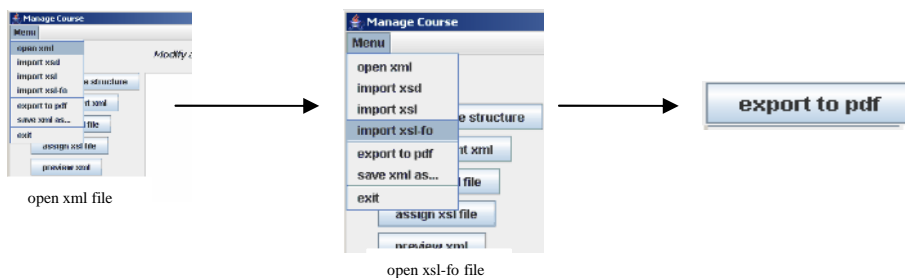


Figura 13: conversione da xml a pdf