XML Parser

Теодор Костадинов 0MI0700242

Курсов проект

GitHub

Документация

Съдържание:

1. Увод

1.1. Описание и идея на проекта

Проектът представлява програма за работа с XML файлове, тяхната сериализация и десериализация, извършване на различни операции върху тяхното съдържание и правенето на прости XPath заявки.
Програмата поддържа операции като селектиране на атрибути на даден елемент, достъп до текста на елемент, изтриване на атрибути, принтиране и т.н.

1.2. Цел и задачи на разработка

Целта на проекта е да бъде създадена програма за десериализация на XML файлове, която съдържа прост интерфейс за възможно най-улеснена работа с XML файловете

Основните задачи на проекта са:

- Прочитане на XML файл и запазване на информацията, записана в него, под формата на дървовидна структура
- Поддръжка на уникален идентификатор за всеки прочетен XML елемент
- Записване на XML документи във файлове
- Извеждане на екрана на прочетената информация от XML файла по добре форматиран начин
- Извеждане на стойност на атрибут по даден идентификатор на елемента и ключ на атрибута
- Добавяне на нов наследник на елемент
- Изтриване на атрибут на елемент по ключ
- Изпълнение на прости XPath 2.0 заявки към кореновия елемент

1.3. Структура на документацията

Документацията разглежда следните теми:

- Общо описание, цели и задачи на проекта, както и структурата на документацията
- Описание на техническите изисквания и спецификации на програмата
- Информация за реализацията на различните операции и алгоритми
- Постигнати резултати и предложения за бъдещо развитие
- Използвана литература

2. Преглед на предметната област

2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

2.1.1. Маркиращ език (език за маркиране) - система от правила, указваща структурата, форматирането на документ и в някои случаи - взаимоотношенията между отделните елементи

2.1.2. *XML* (*Extensible Markup Language* – широкообхватен маркиращ език) - маркиращ език и файлов формат, използван за съхранение и пренос на данни. За него е дефинирано множество от синтактични и семантични правила, които го правят четим както от хора, така и от машини. Обичайното разширение за него е *.xml*

2.1.3. *XPath* - език за създаване на заявки към XML документи

2.1.4. *XPath* ос - ключова дума в една XPath заявка, която дефинира взамодействието между различните елементи в XML дървото. Осите могат да бъдат:

- *Child* - избират се всички деца на текущия елемент
- *Descendant* - избират се всички наследници на текущия елемент
- *Parent* - избира се родителят на текущия елемент
- *Ancestor* - избират се предшествениците на текущия елемент
- *Following-sibling* - избират се всички братя на текущия елемент, които са след него
- *Preceding-sibling* - избират се всички братя на текущия елемент, които са преди него

2.1.5. Таг - маркираща конструкция, която започва с < и завършва с >, като има 3 типа тагове:

- Стартов (начален) таг - например <*product*>
- Затварящ таг - например </*product*>
- Таг за празен елемент (самозатварящ се таг) - например <*br*/>

2.1.6. Елемент - логически компонент, който или започва с начален таг и завърша със затварящ, или съдържа само таг за празен елемент

2.1.7. Съдържание на елемент - текста между отварящия и затварящия таг на елемента (ако такива съществуват)

2.1.8. Дете на елемент - елемент, който е дефиниран в съдържанието на друг елемент

2.1.9. Атрибут - логически компонент, представляващ наредена двойка ключ-стойност, която съществува в рамките на определен елемент

2.1.10. Именно пространство (*namespace*) - начин за избягване на конфликти при имената на елементи в XML (и не само). Именните пространства в XML се дефинират като атрибут, за който е задължително ключът му да започва с *“xmlns:”*

2.1.11. Сериализация - процесът на преобразуване на дадена колекция или обект във формат, който може да бъде съхраняван

2.1.12. Десериализация - процесът на прочитане на файл и преобразуването на данните му в дадена колекция или обект

2.1.13. Дърво - абстрактен тип данни, представляващ йерархична дървовидна структура, съдържаща множество от свързани елементи (възли, върхове). В зависимост от вида на дървото, то може да има много деца, но е задължително всеки елемент да има точно един родителски елемент. Иначе казано, дървото представлява ацикличен, свързан, неориентиран граф.

2.1.14. Обхождане в дълбочина (*depth-first search*) - алгоритъм за обхождане на структури от данни (най-вече дървета и графи). Този алгоритъм представлява основният алгоритъм, използван за обхождането на XML дървото в този проект. Той работи по следния начин:

- Започва се от кореновия елемент на дървото, който се маркира като посетен
- Всеки непосетен съседен възел на текущия се добавя в стек
- Връщане назад се извършва, когато достигнем до възел, който няма непосетени съседи

2.2. Дефиниране на проблеми

2.2.1. Сериализация и десериализация - Създаване на ефективен формат за съхранение и четене на XML файлове, който позволява лесно управление и манипулация на данните.

2.2.2. Уникални идентификатори - Генериране и поддържане на уникални идентификатори за всеки XML елемент, за да се гарантира точност при изпълнение на операции.

2.2.3. Присвояване на стойност на атрибут - Имплементация на механизъм за промяна на стойността на даден атрибут.

2.2.4. Управление на паметта - Имплементация на добър механизъм за управление на паметта в XML дървото.

2.2.5. Обработка на грешки - Осигуряване на валидация на подадения вход и хвърлянето и обработката на подходящи грешки в дадени ситуации.

2.3. Сложност на поставената задача

Разработването на програма за управление на XML файлове е сложна задача, поради факта, че трябва да бъдат съобразени синтактичните и семантичните правила на езика XML (тоест, трябва да се погрижим за правилната обработка на тагове, атрибути, именни пространства и други). Управлението на паметта на една дървовидна структура от елементи е от ключово значение, като основен приоритет трябва да бъде построяването и използването на дървовидна структура, без загуба на излишна памет. Обработката на XPath заявки е трудоемка задача, тъй като трябва да бъдат съобразени синтаксиса и семантиката на езика, както и различните XPath оси.

2.4. Подходи, методи за решаване на поставените проблеми

При решаването на проблемите, свързани с управлението на XML файлове, се използват различни подходи и методи, като:

- Алгоритми за обхождане на дървовидни структура: Използване на алгоритми като *търсене в дълбочина* за намиране на даден елемент в XML дървото
- Поддръжка на уникални идентификатори за всеки елемент

3. Проектиране

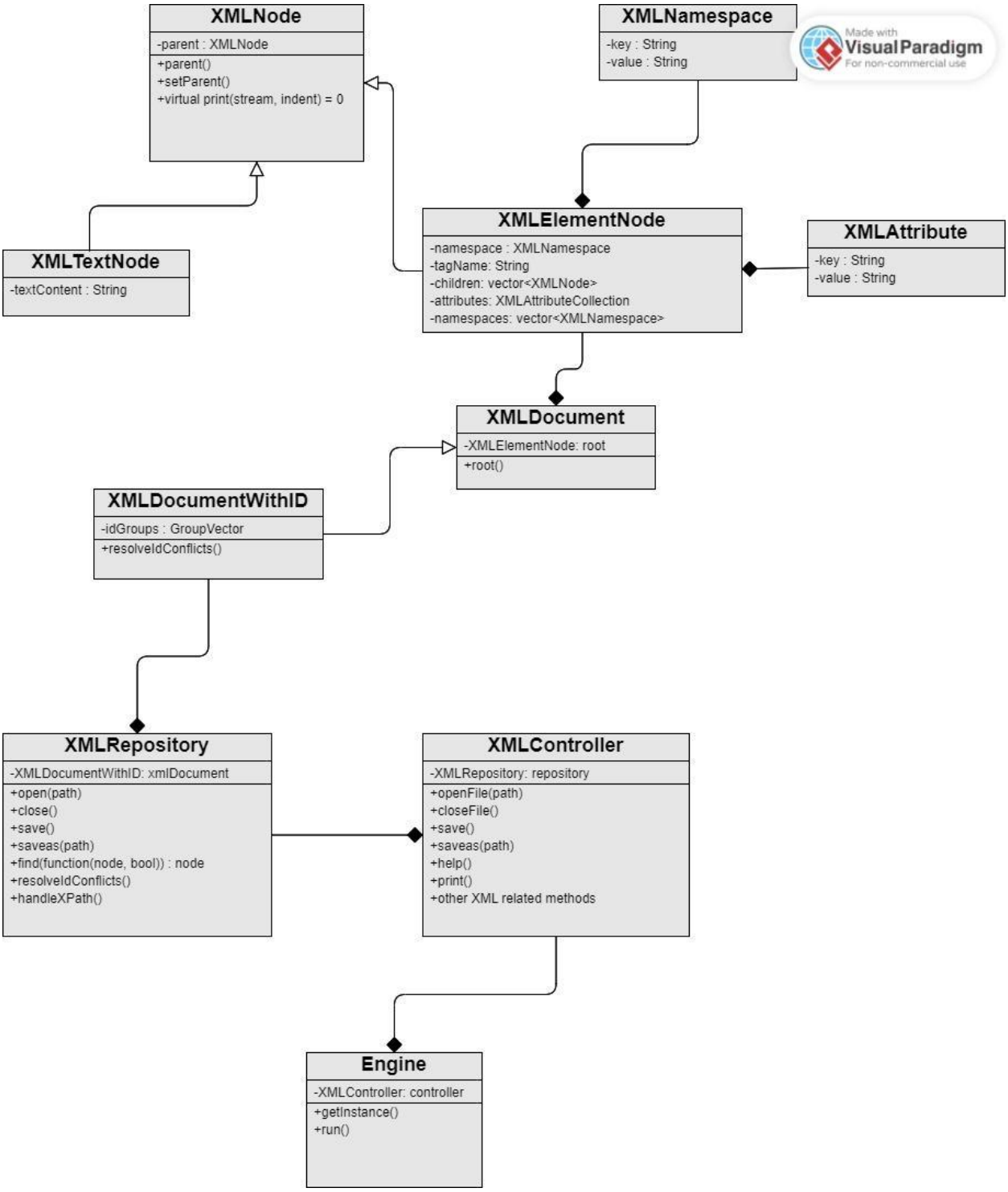
3.1. Обща архитектура – ООП дизайн

Системата за управление на XML файлове е проектирана спазвайки добрите практики на обектно-ориентираното програмиране (ООП).

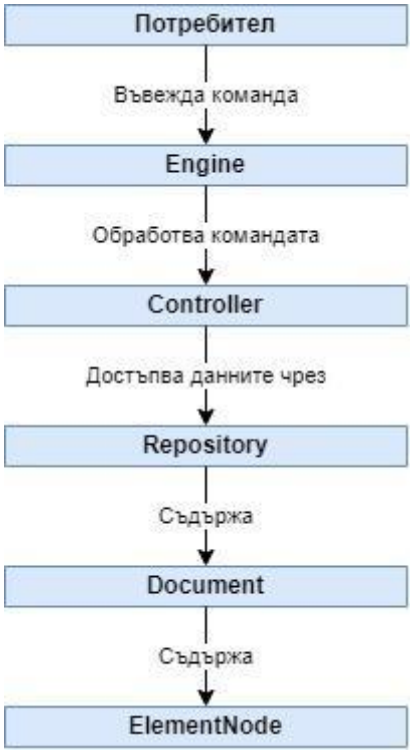
Основните класове са:

- *Engine* за обработка на потребителския вход
- *XMLController* за обработка на подадените от потребителя заявки
- *XMLRepository* за достъп до данните на XML дървото
- *XMLSerializer* и *XMLDeserializer* за сериализация и десериализация на XML дървото
- *XMLAttribute, XMLNode, XMLElementNode, XMLElementNodeWithID, XMLNamespace, XMLTextNode* за различни съставни части на XML дървото
- *XPathQuery* за *XPath* заявките

3.2. Диаграми



Диаграма 1: Структурна диаграма



Диаграма 2: Поведенческа диаграма

4. Реализация, тестване

4.1. Реализация на класове

Класът *Engine* използва *Singleton design pattern*

```
class Engine {

	XMLController _xmlController;

	Engine() = default;

	void handleCommandLine(const MyVector<MyString>&
commandLine);

public:

	Engine(const Engine&) = delete;

	Engine& operator=(const Engine&) = delete;

	static Engine& getInstance();
```

```
	void run();

};

class XMLController

{

	XMLRepository _repository;

	MyString _currentFilePath;

	MyString _currentFileName;

public:

	void openFile(const MyString& path);

	void closeFile();

	void save();

	void saveAs(const MyString& path);

	void help() const;

	void print() const;

	void selectAttribute(const MyString& nodeId, const MyString&
attributeName) const;

	void changeAttributeValue(const MyString& nodeId, const
MyString& attributeName, const MyString& newValue);

	void printChildrenOfNode(const MyString& nodeId) const;

	void printNthChild(const MyString& nodeId, int childIndex)
const;

	void printInnerText(const MyString& nodeId) const;

	void deleteAttribute(const MyString& nodeId, const MyString&
attributeName);
```

```
    void addChild(const MyString& nodeId, const MyString&
newChildName);

    void handleXPath(const MyString& query) const;

};
```

4.2. Управление на паметта и алгоритми. Оптимизации.

Основният алгоритъм за обхождане на XML дървото е *търсене в дълбочина*. Управлението на паметта в програмата е извършено чрез имплементация на така наречените *умни указатели* (*Shared pointer* и *Weak pointer*), които съдържат брояч, отчитащ броя на използващите ресурса, към който сочи указателя.

5. Заключение

5.1. Обобщение на изпълнението на началните цели

Въз основа на изпълнението на началните цели, системата е успешно разработена и предоставя широк набор от функционалности за управление на XML файлове. Бяха постигнати ключовите цели за сериализация, десериализация, работа с XML файловете.

5.2. Насоки за бъдещо развитие и усъвършенстване

В бъдеще системата може да бъде разширена с добавяне на допълнителни функционалности като:

- Поддръжка на *following-sibling* и *preceding-sibling* XPath осите
- Поддръжка на XML декларации
- Поддържане на повече функционалности на *XPath* заявките

6. Използвана литература

1. Jain, Sandeep. “Depth First Search or DFS for a Graph.” *GeeksforGeeks*, 16 February 2024, https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/.
2. W3Schools. “XPath Tutorial.” *W3Schools*, https://www.w3schools.com/xml/xpath_intro.asp.
3. Wikipedia. “XML.” *Wikipedia*, https://en.wikipedia.org/wiki/XML.