Projekt TKOM - Dokumentacja

Konrad Miziński

16 stycznia 2013

Spis treści

1 Treść projektu

Biblioteka java do obsługi plików konfiguracyjnych XML.

2 Cel projektu

Celem projektu jest dostarczenie zestawu klas języka Java pozwalających na:

- Odczyt z pliku konfiguracyjnego właściwości (reprzezentowanych jako pary nazwa-wartość) pogrupowanych w sekcje.
- Dodawanie, usuwanie, modyfikacja właściwości. W szczególności tworzenie własnych obiektów zawierających zdefiniowane przez użytkownika właściwości.
- Serializajca i zapis do pliku zmodyfikowanych/utworzonych obiektów.

3 Wymagania funkcjonalne

- Pliki konfiguracyjne są formatu *.xml o sztywno zdefiniowanej strukturze(opisanej w dalszej części dokumentacji).
- Właściwości zawarte w pliku konfiguracyjnym reprezentowanie są przez pary nazwa-wartość.
- Właściwości są pogrupowane w sekcje.
- Sekcje reprezentowane są przez unikalne nazwy sekcji.
- Dopuszczalne są sekcje puste.
- Nazwy właściwości są unikalne w ramach sekcji.
- Wartości właściwości mogą być puste.
- Pliki konfiguracyjne reprezentowane są przez klasy implementujące interfejs PropertiesFile, sekcje przez klasy implementujące interfejs Section, a właściwości przez klasy imlementujące intefejs Property; Kody źródłowe tych interfejsów zostały zamieszczone w dalszej części dokumentacji.
- Dostęp do klasy reprezentującej plik właściwości można uzyskać za pomocą obiektu serwisowgo, udostępniającego metody odczytujące zapisujące do pliku *.xml obiekty typu PropertiesFile.
- Za tworzenie obiektów klasy PreprtiesFile odpowiada osobna klasa pełniąca rolę fabryki obiektów.
- Do biblioteki dołączona jest dokumentacja wygenerowana przez javadoc.

4 Wymiagania niefunkcjonalne

- Biblioteka jest dostarczona w postaci archiwum typu JAR.
- Wraz z biblioteką jest dostarczony prosty program demonstrujący możliwości biblioteki(w postaci wykonywlnego plku JAR). Program bedzie pracował w trybie tekstowym.

5 Przykładowy plik konfiguracyjny

```
<?xml version="1.0" encoding="ASCII"?>
<properties>
        <section name="section1">
                <property name="name1">value1</property>
                <property name="name2">value2</property>
                <property name="name3"></property>
                <property name="name4"/>
        </section>
        <section name="section2">
                <property name="name1">value3</property>
                <property name="name5">value5</property>
        </section>
        <section name="section3"></section>
        <section name="section4"/>
</properties>
```

6 Projekt realizacji

6.1 Podział na moduły

Biblioteka składa się z 2 podstawowych modułów:

- Moduł obsługi plików xml - oddpowiada za analizę leksykalną i składniową plików xml. Służy do tworzenia struktur danych odpowiadających poszczególnym składowym drzewa xml. Odpowiada za odczyt i zapis do pliku *.xml. Jest niewidoczny dla użytkownika biblioteki.
- Moduł API użytkownika - dostarcza metod implementujących zadania biblioteki. Wykorzystuje obiekty z modułu pierwszego.

6.2 Moduł obsługi plików xml

6.2.1 Analizator leksykalny(Lekser)

Lexer ma za zadanie rozbić wczytany z pliku text na leksemy. Na ich podstawie zostają ropoznane tokeny, którym w szczególnych przypadkach przypisywane są wartości. W poniższej tabeli zamieszczono rozpoznawane przez lexer tokeny wraz z przykładami dla podanego fragmentu pliku xml:

```
<?xml version="1.0" encoding="ASCII"?>
...
<property name="nazwa">value1</property>
<property/>
```

Leksem	Token	Wartość
<property	OpenStartTag	property
<\property	OpenEndTag	property
>	CloseTag	
\>	CloseEmptyElementTag	
name	AttributeName	name
=	Equals	
”nazwa”	AttributeValue	nazwa
value1	Text	value1
<?xml version=”1.0” encoding=”ASCI”?>	Prolog	

Dopuszcza się stosowanie znaków specjalnych takich jak <, >, ” , ’, & zapisanych jako odpowiednio: &lt, &gt, &quot, &apos, &amp

Przyporządkowanie lexemów do tokenów nie zależy tylko od spełnienia wyrażenia regularnego, ale również od stanu w jakim znajdował się lexer po dopasowaniu ostatniego lexema (np. rozpoznanie tokena ”Text” może mastąpić tylko jeśli poprzednio został rozpoznany token ”CloseTag”).

6.2.2 Analizator składniowy(Parser)

Po analizie leksykalnej dane są poddawane analizie składniowej. Parser stara się utworzyć drzewo składni na podstawie następującej gramatyki (uproszczonej gramatyki drzewa xml):

```
Document = Prolog, Element;
Element = OpenStartTag, {Attribute}, (CloseTag, Content, EndTag) | CloseEmptyElementTag
Attribute = AttributeName, Equals, AttributeValue;
StartTag = OpenStartTag, {Attribute}, CloseTag;
EndTag = OpenEndTag, CloseTag;
Content = {Element} | Text;
```

Elementy drzewa xml są opakowane w klasy o następujących interfejsach:

```
package pl.waw.mizinski.xmlproperties.xml;

import java.util.List;

import pl.waw.mizinski.xmlproperties.exceptions.CanNotUpdateElementException;
import pl.waw.mizinski.xmlproperties.exceptions.MissingObjectException;

public interface XMLElement
{
        String getName();

        void setName(String name);

        boolean isComplexElement();

        boolean isEmty();

        List<XMLAttribute> getAttributes();

        void setAttribute (XMLAttribute attribute);

        void removeAttribute (XMLAttribute attribute) throws MissingObjectExc

        List<XMLElement> getChildElements();

        void addChildElement(XMLElement element) throws CanNotUpdateElementEx

        void removeChildElement(XMLElement element) throws MissingObjectExcep

        String getValue();

        void setValue(String value) throws CanNotUpdateElementException;
}
```

```
package pl.waw.mizinski.xmlproperties.xml;

public interface XMLAttribute
{
        String getName();

        void setName(String name);

        String getValue();

        void setValue(String value);
}
```

6.3 Moduł API użytkownika

Moduł ten ma za zadanie dostarczyć użytkownikowi klas pozwalających dodawać/usuwać/edytować właściwości zawarte w pliku konfiguracyjnym. Buduje on swoje klasy w oparciu o dane odczytne z plików i zapisane w klasach modułu pierwszego. Podczas zapisu do pliku dane najpierw są zamienieniane na klasy plików xml(klasy modułu pierwszego), a następnie zapisywane na dysk. Fakt korzystania z formatu xml jakie prawie całkowicie ukryty przed użytkownikiem. Klasy wchodzące w skład tego modułu implementują następujące interfejsy:

```
package pl.waw.mizinski.xmlproperties.properties;

import java.util.List;

import pl.waw.mizinski.xmlproperties.exceptions.CanNotUpdateElementException;
import pl.waw.mizinski.xmlproperties.exceptions.MissingObjectException;

public interface PropertiesFile
{
        List<Section> getSections();

        Section getSectionByName(String name);

        void addSection (Section section) throws CanNotUpdateElementException

        void removeSection(Section section) throws MissingObjectException;
}
```

```
package pl.waw.mizinski.xmlproperties.properties;

import java.util.List;

import pl.waw.mizinski.xmlproperties.exceptions.MissingObjectException;

public interface Section
{
        String getName();

        void setName(String name);

        boolean isEmpty();

        List<Property> getProperties();

        Property getPropertyByName(String name);

        String getPropertyValue(String name);

        void setProperty(Property property);

        void setProperty(String name, String value);

        void removeProperty(Property property) throws MissingObjectException;

        void removeProperty(String propertyName) throws MissingObjectExceptio
}

package pl.waw.mizinski.xmlproperties.properties;

public interface Property
{
        String getName();

        void setName(String name);

        String getValue();

        void setValue(String value);

}
```