Log4j converter - Závěrečná zpráva

Michal
Pochobradský

Log4j converter

Log4j converter

Cílem projektu bylo vytvořit konzolovou aplikaci na koverzi konfigurace log4j mezi formáty souborů properties a XML. Výsledná aplikace se skládá ze tří tříd:

- Log4JConveter - hlavní třída aplikace, stará se o uživatelský vstup.
- XMLToProperties - stará se o konverzi z XML do properties
- PropertiesToXML - stará se o konverzi z properties do XML

Třída PropertiesToXML

Třída PropertiesToXML

Mým hlavním přínosem k projektu byla implementace třídy PropertiesToXML. Jak již bylo uvedeno její úlohou je konverze nastavení Log4j z, properties souboru do xml. Konvertovaný soubor je třídě předložen jako parametr konstruktoru. Samotnou konverzi zajišťuje metoda convert, která má dvě varianty, buď s jedním vstupním parametrem, kterým je název výstupní soubor do něhož se má výsledek konverze uložit, nebo bez parametru, pak se výsledek konverze vypíše na standartní výstup.

Vsstupní soubor je zpracováván pomocí třídy Properties a průběžné výsledky jsou udržovány ve formě DOM stromu prostředky prostředí Java. Konverze třídy je rozdělena do jednotlivých metod zpracovávajících konkrétní prvky konfigurace. Obecně lze tyto metody rozdělit na dvě skupiny podle toho kde se v XML nachází jimi generovaný element na prvky "hlavní" a "vedlejší".

- Hlavní prvky - jsou prvky, které jsou přímo potomky kořenového elementu. Jejich metody jsou tvaru void convertNazevPrvku(Properties properties, SortedSet<String> keys, Element configurationElemnt), kde *properties* obsahuje vstupní data, *keys* jsou setřazené klíče do *properties* (viz níže) a *configurationElemnt* je kořenový element. Každá z těchto metod zná pevně danou předponu klíče podle které určí klíče, které náleží jí zpracovávanému prvku. Tyto metody jsou volány pouze metodou doConvert().
- Vedlejší prvky - jsou prvky, jež jsou částí některého hlavního prvku, tedy jsou potomky jeho elementu.Jejich metody jsou tvaru String processNazevPrvku(Properties properties, SortedSet<String> keys, String currentPrefix, Element element), kde význam *properties* a *keys* je stejný jako u hlavních prvků, *currentPrefix* je předpona klíče prvku zpracovávaného volající funkcí a *element* je XML element prvku generovaný volající funkcí. Tyto metody vytváří při každém volání novou předponu připojením svého "klíčovho slova" k přeponě *currentPrefix*. Tyto metody jsou volány jednak z metod hlavních prvků, ale také se mohou volat mezi sebou.
- Vyjímky - metoda convertConfig,která generuje kořenový element log4j:configuration, je obdobou metod hlavních prvků, ale celkem logicky nemá atribut *configurationElemnt*, naopak jej vrací. Metoda processParams je obdobou metod vedleších prvků, ale navíc má atributy *position* a *ignored*, kde *position* je pozice, kam se mají do *element*u vložit vygenerované elementy

param a *ignored* je seznam klíčových slov patřících jiným prvkům právě generovaného prvku.

Asi největším problémem, na který jsem narazil, byl v tom, jak XML a properties přistupují k řazení prvků. Zatímco v properties může být pořadí prakticky libovolné, DTD log4j klade poměrně velmi striktní podmínky na pořadí jednotlivých elemntů.
Rešení:

- Properties - klíče si ukládám do lexikograficky setřízené množiny, kterou pak předávám jednotlivým metodám (viz výše). Toto mi umožňuje vzít podmnožinu obsahující klíče větší nebo rovny nějakému řetězci, v mém případě předponě náležející generovanému prvku a tu procházet do okamžiku, kdy narazím na klíč nemající danou předponu, pak vím že jsem prošel všechny klíče patřící danému prvku.
- XML - prořadí vygenerovaných elemntů je dáno pořadím volání generujících metod, vyjímkou je metoda processParams, která umožňuje vložit elemnty na konkrétní pozici (viz výše)

Závěr
Závěr
Výsledkem naší práce je použitelná aplikace.