Dokumentace úlohy CSV: CSV2XML v Python 3 do IPP 2015/2016

Jméno a příjmení: Filip Šťastný

Login: xstast24

1 Cíl projektu

Úkolem bylo vytvoření skriptu, který převede soubor z formátu CSV (stanovený normou RFC-4180) do formátu XML (verze 1.0), kde každý CSV řádek/záznam představuje rodičovský XML prvek pro všechny své dceřiné prvky, jejichž obsah je představován jednotlivými sloupci v CSV souboru. Neplatné XML znaky bylo potřeba převést do jejich platné podoby, v případě neplatného názvu prvku ukončit skript s chybou, nebo nahradit zadaným symbolem. Dle dalších parametrů bylo možné např. použít obsah sloupců prvního řádku/záznamu pro nazvání výstupních prvků, očíslovat výsledné prvky nebo celý obsah výstupního souboru obalit kořenovým prvkem.

2 Řešení, popis implementace

Chybové stavy

Jsou ošetřeny vhodně umístěnými podmínkami, které volají funkci print_error(). Ta bere jako parametr chybovou zprávu, kterou vytiskne na standardní chybový výstup, a kód chyby, se kterým ukončí běh programu.

Zpracování parametrů

Není použit žádný modul, argumenty zpracuje instance třídy Params, která postupně volá funkce get_params() pro načtení argumentů, a poté check_params() pro kontrolu platných argumentů a jejich kombinací nebo případné doplnění chybějících implicitních hodnot. Tento objekt také implementuje a případně volá funkci print_help() pro tisk nápovědy.

Zajištění platných XML prvků

Platné symboly jsou použity podle definice XML 1.0, avšak nejsou zahrnuty kontroly znaků s hodnotou přesahující kódování UTF-8, které by díky zadanému kódování souborů nemělo být potřeba řešit. Kontrolu platných symbolů v XML prvcích zajišťuje instance třídy TagValidator. Ta implementuje pouze dvě statické metody, což na jednu stranu vyvolává pocit „ne zcela nezbytné třídy“, ale na stranu druhou podporuje pěknou logickou soudržnost kódu. První metoda replace_problematic_symbols() je volána při potřebě nahrazení neplatných symbolů. Hledá v daném textu neplatné XML symboly a nahradí je symbolem předaným parametrem skriptu. Druhá metoda check_valid_tag() také hledá neplatné XML znaky, avšak nenahrazuje je, pouze ukončí běh programu. Je volána například po nahrazení neplatných symbolů první funkcí, zda je prvek po nahrazení platný, či ne.

Práce se soubory

Dostupnost a otevření vstupního souboru je ošetřeno ve funkci open_input_file(), výstupního souboru ve funkci open_output_file(). Načtení CSV souboru je zajištěno knihovnou „csv“, která je použita s výchozím nastavením, ani není nijak upravena. Zde nastal problém při čtení ze standardního vstupu, kdy je napřed potřeba použít „buffer“ pro načtení dat, až poté z něj číst knihovnou „csv“ (která iterativně volá čtení řádků, což na ne-přednačteném standardním vstupu působilo problémy). Pro tvorbu XML prvků byla použita knihovna „xml.etree.ElementTree“. Postupně v cyklech je zde vytvořen strom reprezentující CSV záznamy, který je nakonec převeden do textové podoby s kódováním UTF-8. Převod na textovou podobu a její zápis do souboru zajistí funkce write_xml_elements_to_file(). V případě chybějícího kořenového elementu netiskne celý strom, ale pouze iteruje po jeho přímých potomcích a až tyto zapíše do souboru.

Rozšíření

Skript podporuje také rozšíření PAD, které v případě číslovaných XML elementů provede zarovnání indexu na stejný počet cifer. Toto je zajištěno převedením číselného indexu na text a spočtením znaků, následným

vynásobením symbolu 0 (nula) rozdílem mezi počtem číslic aktuálního indexu a nejvyššího indexu a připojením aktuálního indexu.

3 Testování a závěr

Testování skriptu proběhlo na zadaných automatických testech, programem JExamXML, kde kvůli častému výstupu neplatného XML elementu byl problém s porovnáváním, a na doplňkových vlastních ručních testech. Chyby programu byly odladěny.

Projekt je funkční na základních testech, navíc s jedním rozšířením ze dvou možných.