Niklas Teichmann, Wolfgang Otto 29.4.2013

Dokumentation für Entwickler

Release 1.1 Vokabular Erstellung und .pickle-.ttl-Konvertierungs-Tool (TTLConverter)

„Vokabular und .pickle KVT“

Anforderung:
/LF130/ Produktfunktion: Überführung der HTML-Daten in eine Datenbank und Erfassung des Vokabulars

TTLConverter

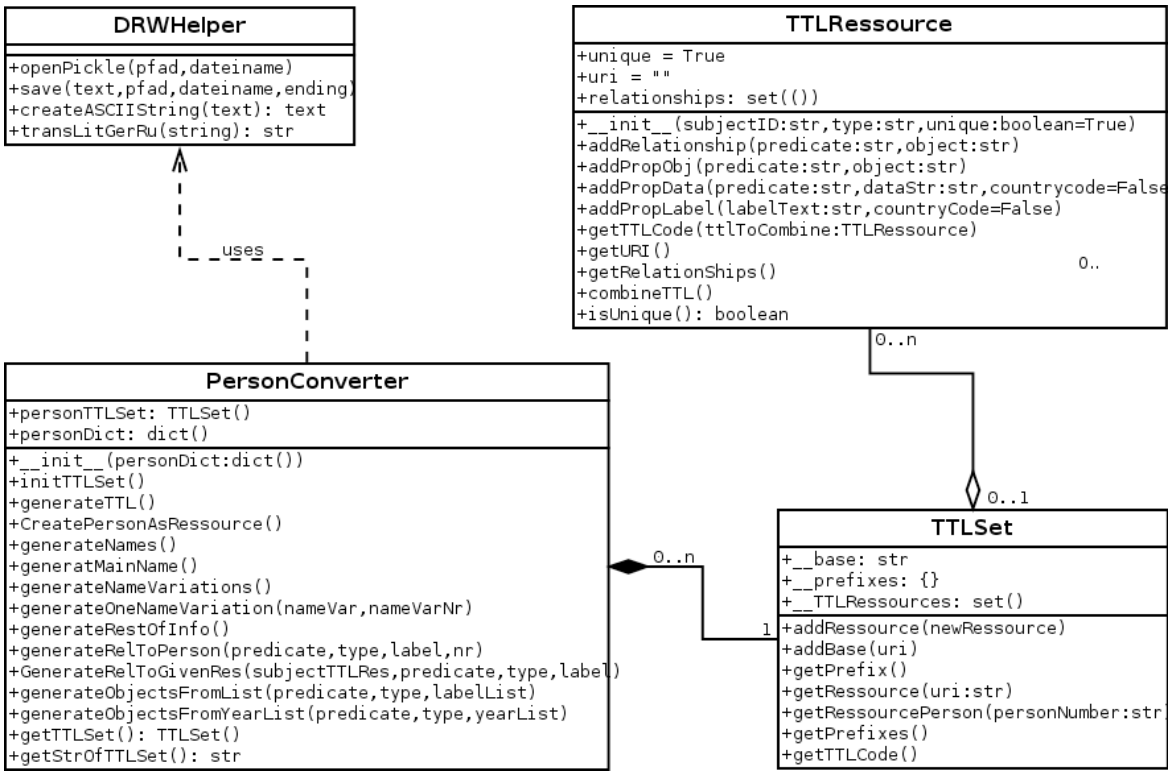
Beschreibung:

Das Modul ist in der Lage Informationen, die in einer dictionary Struktur der Programmiersprache Python formuliert ist und in .pickle Dateien gespeichert ist in das semantische .turtle Format zu überführen.

Die Pickle Dateien liegen vor.

Enwurfsbeschreibung:

UML Diagramm TTLConverter



Der Kern des Moduls ist der PersonConverter, der aus den eingelesenen Dictionaries .TTL text formuliert.

Er speichert die Daten in einem TTLSet Objekt in Form von TTRessource Objekten.

Er geht die Dictionarz durch und ordnet jedem Element die richtigen Beziehungen zu.

Beispielsweise befindet sich im Dictionary unter dem Key „SL“ die Liste der Sekundaerliteratur.

Das Programm erstellt dann aus den Listenobjekten Ressourcen und verknuepft diese mit der passenden Relation mit der Ressource der Person. Als Label wird der Inhalt des Listenobjekts verwedet.

Die Klassen TTLSet und TTLRessource dienen zur Verwaltung der semantischen Daten.

Es sind generische Klassen.

TTLRessource verwaltet eine Ressource, die man mit Angabe von Label und type initialisieren kann.

Ein TTLSet verwaltet mehrere TTLRessourcen.

Desweiteren kann eine base und prefixe defniert werden.

Die Funktion getTTLCode gibt dann base, Prefixe und alle gespeicherten Ressourcen mit ihren Verknüpfungen aus.

Aufgaben, die noch nicht gelöst werden konnten

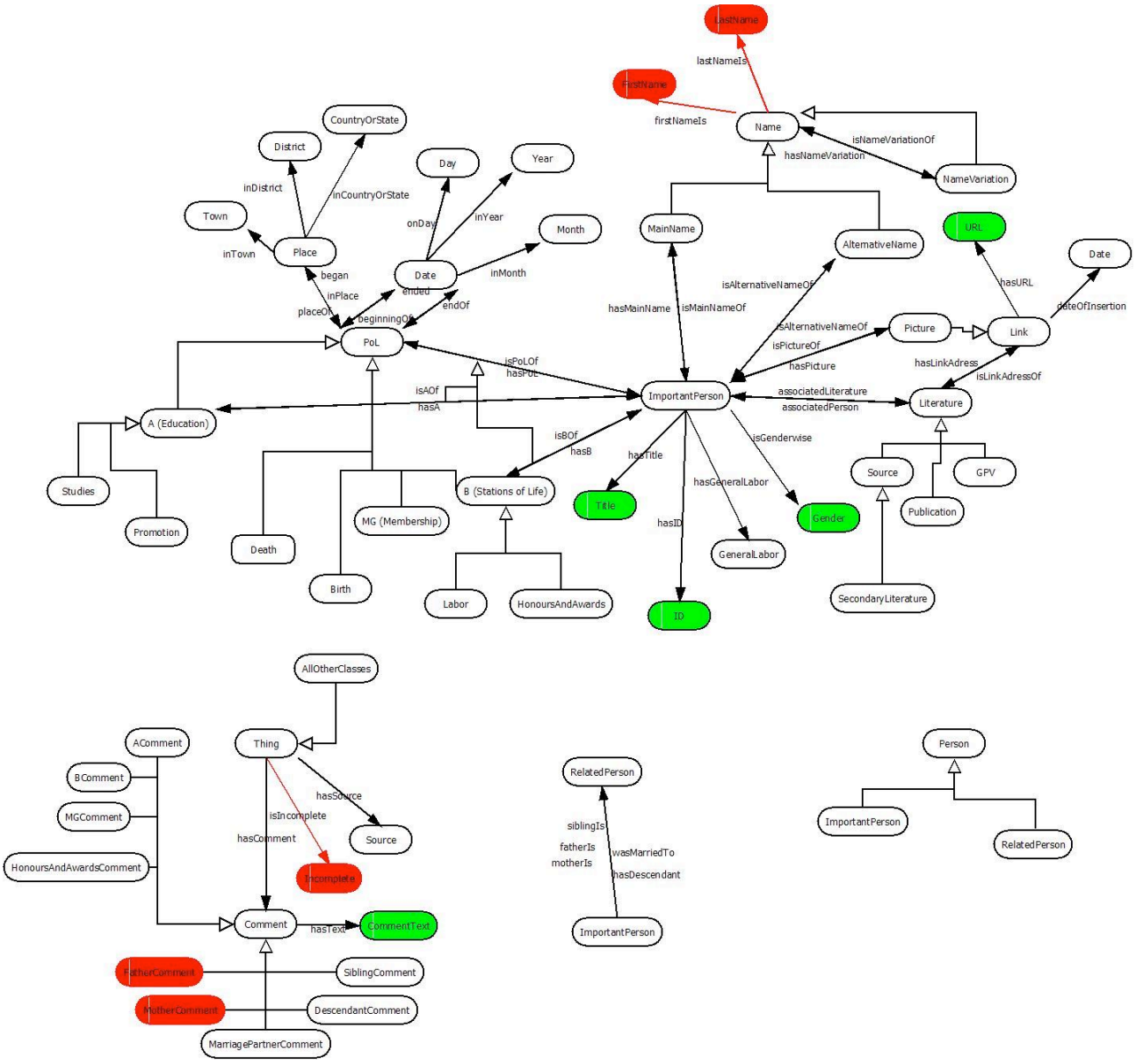
- Es gibt noch Probleme bei einzelnen Unterüberschriften, die noch nicht gelöst wurden. Das Programm wirft einen Fehler, wenn der Fall auftritt.
- Es ist noch nicht Möglich Fußnoten und Interne Verknüpfungen richtig zu verarbeiten. Diese sind in der Grundlage eindeutigen Schlüsselbegriffen zugeordnet, so kann im nächsten Release die Funktion eingefügt werden.

Die Programmcode Dokumentation ist als HTML Version einsehbar.

Vokabularerstellung

Enwurfsbeschreibung:

Eine Komplettübersicht über unser aktuelles Arbeitsvokabular sieht wie folgt aus:



Legende:

Weisse Objekte = Klassen

Grüne Objekte = Daten (als Objekte in Datatype Properties)

Rote Objekte = aktuell nicht verwendete, aber möglicherweise noch benötigte Klassen

Erklärung zu einzelnen Konstruktionen:

Namen: Um die Vielzahl an unterschiedlichen Namen, Namensvariationen und Alternativnamen einer Person darstellen zu können benötigen wir 3 von der Klasse Name abgeleitete Klassen: Eine Klasse Hauptname, die den Namen beschreibt, mit dem die Person identifiziert wird, und der in den html-Seiten sowie im späteren Ansichts-Plugin zuoberst erscheint, eine Klasse Alternativname, die alle deutlich unterschiedlichen Namen einer Person umfasst, sowie eine Klasse Namensvariation, die kleinere Rechtschreib- und Transliterationsvariationen von bereits bestehenden Namen enthält.

ImportantPerson und RelatedPerson: Eine Konstruktion, die nötig wurde, um zwischen einer Vielzahl von verwandten und assoziierten Personen und den eigentlichen Kernpersonen des Datenbestandes zu unterscheiden. Beide sind von Person abgeleitet und können dementsprechend nach Belieben ergänzt werden, sind aber in Suche etc. klar abgetrennt.

Personen ID´s : Eine aus den html-Seiten übernommene Identifikationsstruktur, die wir auch zur Benennung der Ressourcen verwenden (.../ImportantPerson/31004 oder Ähnliches). Durch diese Entscheidung wird die interne Verbindung der Personen im Ansichts-Plugin erheblich vereinfacht.

PoL Unterklassen: Um eine klare Unterteilung der PoL´s wie in den html-Seiten zu erreichen, war es nötig, die dort verwendete Organisationsstruktur (A - Ausbildung, B - andere Lebensstationen, MG - Mitgliedschaften, ...) zu übernehmen. Dazu wurden Oberklassen und Subproperties von hasPoL deklariert, die der Bedeutung entsprechen.

Kommentare: Vielfach gibt es innerhalb der Daten Kommentare, die nicht einer speziellen Information, sondern einer ganzen Gruppe davon zugeordnet sind. Um die Darstellung davon zu erleichtern wurden Unterklassen von Comment erzeugt, die sich nicht auf einzelne Informationen, sondern auf allgemeine Informationen zur Ausbildung, Ehepartnern, Nachfahren, Ehrungen usw. beziehen.

Fehlende Datatype-Properties: Der überwiegende Teil der Daten wird in den Labels der einer Person zugeordneten Objekte gespeichert. Um die Übersichtlichkeit zu bewahren wurden diese in der obigen Grafik nicht eingetragen.