Universität Leipzig
Softwaretechnik-Praktikum

Entwurfsbeschreibung

DTP-16: LD Hypstrr - Datadriven Template Publication

11. April 2016

Inhaltsverzeichnis

1 Allgemeines

1.1 Installation

Die Installation unseres Plugins mitsamt Jekyll und allen weiteren benötigten Komponenten erfolgt durch eine Ruby Gem-Installation.

```
gem install jekyll-rdf
```

Dies setzt eine vorhandene Ruby-Installation voraus. Aktiv unterstützt werden die Versionen 2.2.4 und 2.3.0.
Danach legt man eine neue Seite an und passt diese nach den eigenen Wünschen an:

```
jekyll new <<Seitenname>>
```

Es wird ein neues Verzeichnis angelegt, welches den angegebenen Seitennamen trägt. Hier findet man schließlich die Unterordner css, _includes, _layouts, _posts und _sass sowie die Dateien about.md, feed.xml, index.html und die wichtige _config.yml, mit der wir uns in der weiteren Konfiguration näher befassen.

1.2 Konfiguration

Die Konfiguration des Plugins erfolgt durch die Anpassung der _config.yml Datei im Stammordner. Hier legt man den Pfad zur RDF-Datei sowie - optional - Einschränkungen über eine SPARQL-Select Query fest. Außerdem muss man die Abhängigkeit zum jekyll-rdf Gem deklarieren. Dies geschieht per folgenden Einträgen in der _config.yml:

```
gems: [jekyll-rdf]
jekyll_rdf:
  path: "pfad/zur/rdf.ttl"
  query: "<<SPARQL-SELECT Query>>"
```

1.3 Benutzung

Um unser Plugin zu benutzen kann man in Markdown-Files oder Layouts (im Unterverzeichnis _layouts) unsere definierten Liquid-Tags benutzen. So verwendet man für den Zugriff auf ein mit der aktuellen Ressource als Subject in Verbindung stehendes Object durch Angabe eines Prädikates folgendes Tag:

```
{% rdfobject <<Praedikat-URI>> %}
```

Für Listen kann man folgenden Tag verwenden:

```
{% rdflist <<Praedikat-URI>> <<Opening-Tag>> <<Closing-Tag>> %}
```

Als Opening- und Closing-Tags kommen beispielsweise <li> und infrage. Natürlich auch weitere HTML-Tags oder Strings, die vor bzw. nach jedem Object vorkommen sollen.

Um die RDF-Daten durch eine weitere Query einzuschränken und sich das Ergebnis davon ausgeben zu lassen benutzt man folgendes Tag:

```
{% rdfquery <<SPARQL-Query>> %}
```

Weiterhin kann man bereits im Liquid Tag die gewünschte Sprache durch den Parameter **lang** definieren. So greift man beispielsweise bei einem einzelnen Object auf eine bestimmte Sprachvariante zu:

```
{% rdfobject <<Praedikat-URI>> lang=de %}
```

2 Produktübersicht

Unser Produkt ist ein Plugin für das Static Site Generation Tool Jekyll, womit man in der Lage ist, aus RDF-Daten und Templates statische HTML-Websites zu rendern. Der User legt in einer config-Datei fest, welche RDF-Datei, bzw. durch das Hinterlegen einer SPARQL-Query, welcher Teilgraph der RDF-Datei gerendert werden soll. Standardmäßig werden nur Subjekte mit Hilfe eines Default-Templates verarbeitet. Möchte der Nutzer jedoch weitere, bzw. anders aufgebaute Seiten generieren, so kann er klassenspezifische Templates erstellen. In diesen kann der Anwender unter Nutzung von mitgelieferten Tags die HTML-Seiten nach seinen Bedürfnissen anpassen. Es ist zum Beispiel möglich von einem Subjekt ausgehend über dessen Prädikat-URI für alle Objekte eine eigene HTML-Seite zu rendern. Die gerenderten HTML-Ressourcen werden in einer übersichtlichen Ordnerstruktur abgelegt. Unser Produkt wird für die Nutzung auf Linux-basierten Systemen entwickelt, die Ausführbarkeit auf Windows- und Mac-Systemen spielt bei der Entwicklung keine Rolle.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

3.1 Jekyll-Plugins

Unser Produkt besteht aus 7 Klassen: **MainGenerator, RdfPageData, RdfSiteData, TemplateMapper, PropertyTag, PropertyListTag** und **PropertyQueryTag**. Der MainGenerator bildet das Grundgerüst und erbt von der Jekyll-Klasse Generator. Er liest die Config ein und erhält dadurch den Pfad zur RDF-Datei, sowie (falls vorhanden) SPARQL-Query Anfragen. Dieser Graph wird dann in RdfSiteData zwischengespeichert. Aus dem, mit dieser Query Anfragen eingeschränkten, Graphen wird dann für jedes Subjekt mithilfe von RdfPageData eine HTML-Seite generiert. Die Klasse RdfPageData, welche von Jekyll Page erbt, erstellt die HTML-Seite nach den Vorschriften des Templates, welches durch TemplateMapper ausgesucht wird. In den Templates werden die drei Liquid-Tags benutzt, die alle von der Liquid Klasse *Tag* erben. PropertyTag ermöglicht den Zugriff auf ein Objekt, ausgehend vom Subjekt für das aktuell die Seite generiert wird und der URI des Prädikates. Sollten mehrere Objekte in der gleichen Beziehung zum Subjekt stehen, so wird das erste gefundene Objekt genutzt. Um alle Objekte aufzulisten hilft PropertyListTag, mit dem man alle in Verbindung stehenden Objekte auflisten

kann. PropertyQueryTag ermöglicht es den, vom MainGenerator eingelesenen, Graphen weiter für den Renderprozess der Seite einzuschränken, um so dem Nutzer weitere Spezifikationsmöglichkeiten zu gewähren.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 MainGenerator

MainGenerator ist eine Subklasse von Jekyll::Generator und besitzt nur die Methode *generate(site)*, wobei *site* eine Instanz der Klasse Jekyll::Site ist. In unserem Fall ist site speziell die Seite, die mit dem Plugin gerendert werden soll. In generate(site) wird zuerst eine Konfigurationsdatei (_config.yml) gelesen, in welcher der Pfad zu der zu verarbeitenden RDF-Datei angegeben ist. Ist zusätzlich eine SPARQL-Query in der Konfigurationsdatei hinterlegt, so wird diese auf den RDF-Graphen angewandt. Mit dem Ergebnis dieser Query, bzw. dem ganzen Graphen als Parameter, wird eine Instanz von RdfSiteData initialisiert, in der diese Daten zwischengespeichert werden. Basierend der Instanz von RdfSiteData wird danach für jedes Subjekt ein Objekt der Klasse RdfPageData initialisiert und zur zu erstellenden Site hinzugefügt.

4.2 RdfPageData

RdfPageData ist eine Subklasse von Jekyll::Page und besitzt die zwei Methoden *determineDir(subject)* und *initialize(site, base, subject, graph)*. determineDir ist eine Hilfsmethode, die in der initialize-Methode aufgerufen wird, um einen geeigneten Ablageordner zu finden bzw. zu erstellen, damit die erstellten Pages in einer sinnvollen Ordnerstruktur abgelegt werden. Der passende Ordner wird anhand des Parameters subject bestimmt, welcher lediglich das Subjekt angibt, für welches gerade eine Instanz von Page initialisiert wird.
initialize(site, base, subject, graph) ist die Hauptmethode dieser Klasse und erstellt neue Pages. *site* gibt die aktuelle Website an, die aktuell gerendert wird und zu der die Page gehören soll, *base* gibt das Unterverzeichnis unserer Site an. *subject* ist das Element des RDF-Graphen, für das eine Page initialisiert werden soll und *graph* ist der dazugehörige RDF-Graph. *initialize(site, base, subject, graph)* legt den Namen und (mithilfe von determineDir) Ordner der Page fest. Weiterhin wird durch den TemplateMapper das zu subject passende Template bestimmt, auf dessen Basis die Page dann gerendert wird.

4.3 RdfSiteData

RdfSiteData ist eine Subklasse von Jekyll::Site und verfügt nur über die Methode *initialize(graph)*. Diese Methode initialisiert eine Instanz von Site und speichert den mit graph angegebenen RDF-Graphen als siteData ab. Schließlich greifen die einzelnen Pages auf dieses Objekt zu, sodass ein mehrfaches Abfragen oder Übergeben des kompletten Graphen entfällt.

4.4 TemplateMapper

Die Klasse TemplateMapper implementiert nur die Methode *map(subject)*. Subject ist dabei das Subjekt bzw. Objekt, für das gerade eine Seite gerendert wird. Für subject durchsucht map(subject) dann die Template-Mapper Konfigurationsdatei, ob passende Templates hinterlegt wurden, bzw. ob Templates für die Oberklassen von subject existieren. Werden an einer Stelle mehrere Templates gleichzeitig gefunden, so wird eines ausgesucht und gleichzeitig eine Warnung ausgegeben. Das passende Template wird als Rückgabewert übergeben.

4.5 Liquid Tags

Alle LiquidTags sind Subklassen von Liquid::Tag.

4.5.1 PropertyTag

Dieser Tag ermöglicht es von einem Subjekt ausgehend, für das gerade eine Page gerendert wird, auf ein mit dem Subjekt in Verbindung stehendes Objekt durch Angabe des Prädikates zuzugreifen. Dazu besitzt er die Methode *render(context)*, wobei context die Prädikat-URI angibt, über die das Objekt zu finden ist. Existieren mehrere Objekt, die man über das Subjekt und die URI des Prädikates findet, so wird nur das erste gefundene Objekt ausgegeben.

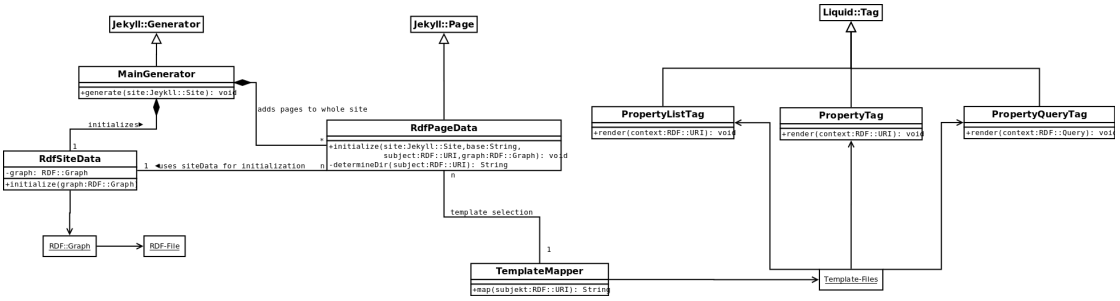
4.5.2 PropertyListTag

Dieser Tag ermöglicht es von einem Subjekt ausgehend, für das gerade eine Page gerendert wird, auf die mit dem Subjekt in Verbidung stehenden Objekte durch Angabe des Prädikates zuzugreifen. Der Tag besitzt deswegen nur die *render(context)* Methode, wobei context die URI das Prädikates angibt, über die die Objekte aufzufinden sind.

4.5.3 PropertyQueryTag

Dieser Tag ermöglicht es von einer SPARQL-Query ausgehend die Ergebnisse der Query zu rendern. Dazu wird die Methode *render(context)* implementiert. Context bezeichnet die Query, dessen Ergebnisse zu Pages verarbeitet werden sollen.

5 Datenmodell



6 Glossar

6.1 FOAF

FOAF steht für "friend of a friend" und ist ein Projekt zur maschinenlesbaren Modellierung sozialer Netzwerke durch ein RDF-Schema, welches Klassen und Eigenschaften definiert. FOAF ist eines der ersten Anwendungen von Semantic-Web-Technologien.

6.2 Graph

Ein Graph ist eine abstrakte Struktur, die eine Menge von Objekten (Knoten) und deren Verbindungen (Kanten) darstellt. Eine Kante existiert immer nur zwischen genau zwei Knoten, ist also eine paarweise Verbindung, und kann gerichtet oder ungerichtet sein.

6.3 HTML

(kurz für: Hypertext Markup Language) ist eine vom W3C entwickelte Markup Sprache, die Texten und Daten Darstellungsformen und Eigenschaften zuweist. HTML-Daten setzen sich aus verschachtelten Tags zusammen und sind Multimedia-kompatibel.

6.4 Linked Data

Linked Data ist eine Methode zum Veröffentlichen strukturierter Daten. Damit werden sie miteinander verknüpft und werden durch semantische Anfragen noch hilfreicher.

6.5 Literal

Ein Literal ist im informationstechnischen Sinne eine Folge von Zeichen, die zur direkten Darstellung der Werte von Datentypen zulässig ist. Dies gilt auch im RDF-Modell, wo eine Ressource nur durch eine andere Ressource oder ein Literal beschrieben werden kann.

6.6 Metadaten

Metadaten sind Daten, die Informationen über Eigenschaften anderer Dateien enthalten, jedoch nicht die Datei selbst. So gehört zum Beispiel zu den Metadaten eines Kinofilms der Name des Regisseurs, des Produktionsstudios, etc.

6.7 Namespaces bzw. Namensräume

Namespaces ermöglichen es, Elemente mit gleichem Namen eindeutig voneinander zu unterscheiden, indem die Elemente und Attribute URIs zugewiesen werden.

6.8 Notation3 bzw. N3

Notation3 ist eine formale Sprache zur Beschreibung von semantischen Daten. Das bekannte Turtle bildet eine Untermenge.

6.9 Ontologie

Eine Ontologie (auch: Vokabular) ist eine sprachlich strukturierte Darstellung von Wissen. Es wird genutzt, um Beziehungen zwischen einzelnen Begriffen darzustellen. Dies ermöglicht mit einem standardisierten Vokabular auch eine Form der digitalen Verwendbarkeit in verschiedenen Systemen und erleichtert das logische Verstehen der Daten.

6.10 OntoWiki

OntoWiki ist eine Wiki-artige Software, die das Bearbeiten und Anzeigen von semantischen Daten zulässt. Die Besonderheit besteht darin, Komplexität der Datenspeicherung zu abstrahieren. Die Bedienung erfolgt Wiki-typisch über eine Weboberfläche.

6.11 SPARQL

SPARQL ist eine Abfragesprache für RDF-Daten. Daten können von verschiedenen RDF-Datenquellen in einer Anfrage abgefragt werden. Um die Lesbarkeit zu erhöhen ist es möglich Präfixe zu definieren. Als Ergebnis wird ein RDF-Graph oder ein Ergebnisset geliefert.

6.12 RDF

RDF (kurz für: Resource Description Framework) ist ein System zur Beschreibung von Ressourcen. Es ist eine technische Herangehensweise im Internet zur Formulierung logischer Aussagen über beliebige Ressourcen. Im RDF Modell besteht jede Aussage aus 3 Einheiten: Subjekt, Prädikat und Objekt.

6.13 RDFS

RDFS (kurz für: Resource Description Framework Schema) ist eine von der W3C entwickelte Ontologie-Beschreibungssprache für RDF-Daten. Mit Hilfe von RDFS lassen sich RDF-Ressourcen semantisch durch Eigenschaften und Relationen untereinander beschreiben, wodurch man Ontologien erzeugen kann. RDFS basiert auf der Idee eines mengentheoretischen Klassenmodells. Besitzt man zum Beispiel eine RDF Instanz von Fahrzeug und eine von Auto, so kann man mit RDFS festlegen, dass Fahrzeug eine Klasse und Auto eine Subklasse von Fahrzeug ist.

6.14 URI

URI (kurz für: Uniform Resource Identifier) ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient. URIs werden zur Bezeichnung im Internet und dort besonders im World Wide Web genutzt. URIs können als Zeichenfolge in digitalen Dokumente eingebunden werden ("Hyperlink"). Aufbau: scheme, authority, path, query, fragment.

6.15 Turtle

Turtle steht für "Terse RDF Triple Language" und ist eine Serialisierung für RDF-Graphen. Turtle-Dateien enden auf .ttl.

6.16 Triple-Store

Ein Triple-Store (auch: RDF-Store) ist eine Datenbank zum Speichern und Verwenden von Tripeln durch semantische Anfragen. Triple-Stores speichern wie relationale Datenbanken Informationen in Tripeln, jedoch wurden Triple-Stores dafür optimiert. Tripel können durch RDF oder andere Formate importiert und exportiert werden. Wenn man dem Tripel einen Namen gibt, hat man einen Quad-Store oder einen sogenannten "named graph".

6.17 W3C

W3C (kurz für: World Wide Web Consortium) ist ein internationales Gremium, dessen Ziel es ist, durch zukunftsorientierte Web-Standards in Form von Protokollen und Richtlinien die gesamten Kapazitäten des World Wide Webs zu erschließen und dessen langfristiges Wachstum zu gewährleisten. Es verfolgt genauer die Vorteile des Internets für jedermann zugänglich zu machen, unabhängig von Soft-, Hardware, Sprache, Kultur, etc., sowie den Aufbau eines semantischen Webs.