Universität Leipzig
Softwaretechnik-Praktikum

Handbuch

DTP-16: LD Hypstrr - Datadriven Template Publication

6. Juni 2016

Inhaltsverzeichnis

1 Projektübersicht

Unser Plugin Jekyll-RDF erweitert das Static Site Generation Tool Jekyll um die Funktionalität, für jedes Subjekt eines Tripels eines RDF-Graphen eine eigene HTML-Ressource zu rendern. Auf jeder einzelnen Seite pro Subjekt werden schließlich standardmäßig alle mit diesem Subjekt in Verbindung stehenden Prädikate und Objekte tabellarisch aufgelistet. Darüber hinaus ist es möglich, über Template-Mappings in der Konfigurationsdatei spezielle, nur für diese URIs bestimmte Layouts anzugeben und diese zu rendern. Es ist möglich die RDF-Dateitypen ttl ("turtle"), nt ("n-triples"), RDF-XML (rdf bzw. xml), n3 und trig fehlerfrei zu verarbeiten und eine Einschränkung des RDF-Graphen durch eine (in der "_config.yml" hinterlegten) SPARQL-SELECT Query vorzunehmen.

2 Systemvoraussetzungen

Um LD Hypstrr benutzen zu können muss eine Linux-Installation vorhanden sein. Zusätzlich muss Ruby in der Version 2.2.4 oder 2.3.0 bereitstehen. Ruby kann entweder über das Paketverwaltungssystem bezogen werden oder muss individuell kompiliert werden.

3 Installation des Tools

3.1 Installation als Ruby Gem

Die einfachste und schnellste Art unser Programm zu installieren stellt die Installation als Ruby Gem dar. Folgender Befehl installiert LD Hypstrr sowie alle zusätzlich benötigten Komponenten, wie beispielsweise Jekyll und die RDF-Library:

```
gem install jekyll-rdf
```

3.2 Installation mithilfe des Git-Repositories

Um LD Hypstrr mithilfe des öffentlichen Git-Repositories zu installieren muss zunächst git installiert werden. Dies geschieht in den meisten Fällen über die Paketverwaltungssoftware. Danach muss das git-Repository gecloned werden:

```
git clone git@github.com:DTP16/jekyll-rdf.git
```

Es wird automatisch ein Ordner namens jekyll-rdf angelegt. In diesen muss nun gewechselt weden um das Ruby gem zu kompilieren:

```
cd jekyll-rdf
gem build jekyll-rdf.gemspec
gem install jekyll-rdf-*.gem
```

4 Nutzung

4.1 Integration in Jekyll

Als ersten Schritt muss man eine Jekyll-Seite erstellen. Dies geschieht über folgende Standardbefehle:

```
jekyll new my_page
cd my_page
```

Danach fügt man unsere ”jekyll-rdf” Gem zu ”_config.yml” hinzu:

```
gems: - jekyll-rdf
```

Abschließend muss noch der Pfad zur RDF-Datei in ”_config.yml” hinterlegt werden:

```
jekyll_rdf:
  path: "simpsons.ttl"
```

4.2 RDF-Daten verarbeiten

Als nächstes kann man ein oder mehere Dateien im ”_layouts”-Ordner anlegen, um die Layouts der RDF-Seiten zu überarbeiten (z.B.”rdf_index.html” oder ”person.html"). Für jede Resource wird eine Seite gerendert. Die Layouts könnten wie folgt aussehen:

```
---
layout: default
---
<div class="home">
  <h1 class="page-heading"><b>{{ page.rdf.name }}</b></h1>
  <p>
    <h3>Statements in which {{ page.rdf.name }} occurs as subject:</h3>
    {% include statements_table.html
            collection=page.rdf.statements_as_subject %}
  </p>
  <p>
    <h3>Statements in which {{ page.rdf.name }} occurs as predicate:</h3>
    {% include statements_table.html
            collection=page.rdf.statements_as_predicate %}
  </p>
  <p>
    <h3>Statements in which {{ page.rdf.name }} occurs as object:</h3>
    {% include statements_table.html
            collection=page.rdf.statements_as_object %}
  </p>
</div>
```

Zum Zugriff auf Objekte, welche über ein Prädikat mit dem aktuellen Subjekt verbunden sind, stehen unsere selbsterstellten Liquid-Filter zur Verfügung. Für nur ein einzelnes Objekt existiert der "rdf_property"-Filter, für eine Liste von Objekten der "rdf_property_list"-Filter, bei welchem dynamisch innerhalb des Templates iteriert wird und somit sehr einfach Beginn- und End-HTML-Tags angegeben werden können. Hier ein Beispiel:

```
<table>
  <tbody>
    <tr>
      <td>Age</td>
      <td>{% page.rdf | rdf_property:
                          'http://xmlns.com/foaf/0.1/age' %}</td>
    </tr>
    <tr>
       <td>Sisters</td>
       <td>
       {% assign resultset = page.rdf | rdf_property_list:
                'http://www.ifi.uio.no/INF3580/family#hasSister' %}
       <ul>
       {% for result in resultset %}
          <li>{{ result }}</li>
       {% endfor %}
       </ul>
       </td>
    </tr>
  </tbody
</table>
```

Außerdem ist es möglich die bevorzugte Sprache anzugeben:

```
{% page.rdf | rdf_property:
                          'http://xmlns.com/foaf/0.1/job','de' %}

{% assign resultset = page.rdf | rdf_property_list: 'http://www.ifi.uio.no/INF
<ul>
{% for result in resultset %}
  <li>{{ result }}</li>
{% endfor %}
</ul>
```

Weiterhin haben wir einen Liquid-Filter implementiert, um eigene SPARQL-Queries ausführen zu können. Jedes Vorkommen von "?resourceUri" wird ersetzt durch die aktuelle URI. Hinweis: Query und Ergebnis müssen separiert werden aufgrund der Liquid-Vorgaben. In der Praxis könnte das dann so aussehen:

```
{% assign query =
```

```
            'SELECT ?sub ?pre WHERE { ?sub ?pre ?resourceUri }' %}
{% assign resultset = page.rdf | sparql_query: query %}
<table>
{% for result in resultset %}
  <tr>
    <td>{{ result.sub }}</td>
    <td>{{ result.pre }}</td>
  </tr>
{% endfor %}
</table>
```

4.3 Einstellen des Default-Templates und Mapping eigener Templates

Es ist möglich das Default-Template zu ändern, sowie für bestimmte Resourcen, Typen oder Klassen Templates bzw. Layouts zu mappen. Dazu muss nur "_config.yml" wie folgt angepasst werden:

```
jekyll_rdf:
  default_template: "rdf_index.html"
  template_mappings:
    "http://xmlns.com/foaf/0.1/Person": "person.html"
    "http://www.ifi.uio.no/INF3580/simpsons#Abraham": "abraham.html"
}
```

Für alle Objekte der Klasse foaf:Person wird nun die Datei person.html zum Rendern benutzt. Für die spezielle Ressource "Abraham" wird dagegen die Datei abraham.html benutzt.
Falls einer Ressource mehrere Templates zugeordnet werden könnten (da die Ressource beispielsweise von mehreren Klassen erbt), so wird eine Warnung in der Kommandozeile ausgegeben. In diesem Fall wird das zuerst gefundene Template benutzt.

4.4 Resourcen-Selektion einschränken

Es besteht weitenhin die Möglichkeit die initialen Resourcenauswahl zu beschränken. Dies geschieht durch das Hinzufügen einer SPARQL-Query in "_config.yml" als Parameter "restriction":

```
jekyll_rdf:
  restriction: "SELECT ?resourceUri WHERE { ?resourceUri
      <http://www.ifi.uio.no/INF3580/family#hasFather>
          <http://www.ifi.uio.no/INF3580/simpsons#Homer> }"
```

Dabei sind, neben einer SPARQL-SELECT Query, drei vordefinierte Schlüsselwörter für die Einschränkungen implementiert:

- "subjects" lädt alle Subjekt-URIs

- ”predicates” lädt alle Prädikat-URIs
- ”objects” lädt alle Objekt-URIs

Außerdem kann man mithilfe der Konfigurationsdatei entscheiden, ob Blank Nodes ebenso gerendert werden sollen oder nicht. Das geschieht durch de Option ”include_blank”:

```
jekyll_rdf:
  include_blank: true
```

Weiterhin kann man durch die Option ”lang” die bevorzugte Sprache der RDF-Literale spezifizieren:

```
jekyll_rdf:
  lang: "de"
```

4.5 Test-Konfiguration

Zusammenfassend könnte eine Konfiguration unseres Plugins wie folgt aussehen:

```
jekyll_rdf:
  path: "rdf-data/simpsons.ttl"
  language: "de"
  include_blank: true
  restriction: "SELECT ?s WHERE { ?s ?p ?o }"
  default_template: "rdf_index.html"
  template_mappings:
    "http://xmlns.com/foaf/0.1/Person": "person.html"
    "http://www.ifi.uio.no/INF3580/simpsons#Abraham": "abraham.html"
```

5 Weiterführende Entwicklung

5.1 Tests ausführen

Wenn man unser Projekt erweitern will, sollte man dazu in der Lage sein Tests durchzuführen. Dies geschieht über den folgenden Befehl:

```
bundle exec rake test
```

Da wir das Ruby-Gem ”pry” benutzen ist es möglich innerhalb des Quellcodes ”binding.pry” zu schreiben, sodass Ruby bei dieser Stelle stehen bleibt sobald man ”rake test” ausführt um dann von dort an weiter zu programmieren bzw. Variablen auszulesen.

5.2 Die generierte Seite testen

Jedes Mal, wenn die Tests ausgeführt werden, wird automatisch die Jekyll-Seite unter ”test/source” berechnet. Man kann einen kleinen Web-Server starten um die Ergebnisse im eigenen Browser einzusehen, z.B. mit Pythons ”SimpleHTTPServer” unter Python 2 oder ”http.server” unter Python 3:

```
cd test/source/_site
python -m SimpleHTTPServer 8000
```

5.3 Klassendokumentation

Die Klassendokumentation ist für jedes gem-Release unter `https://rubygems.org/gems/jekyll-rdf` (Unterpunkt rechts unter "Links:¨ –> "Dokumentation") verfügbar.