Universität Leipzig
Softwaretechnik-Praktikum

Arbeitsplan

DTP-16: LD Hypstrr - Datadriven Template Publication

1. Februar 2016

Inhaltsverzeichnis

1 Projektvision

Unser Produkt ist eine Erweiterung des in Ruby geschriebenen Static Site Generation Tools Jekyll zum Erstellen von statischen HTML-Websites basierend auf semantischen RDF-Datensätzen. Dies soll in Form eines Jekyll-Plugins geschehen, sodass der eigentliche Jekyll-Quellcode nicht berührt und so eine leichte Aktualisierungsmöglichkeit gewährleistet wird. Die Abfragen auf den RDF-Daten werden mit SPARQL realisiert, jedoch soll es auch die Möglichkeit geben, ohne SPARQL-Abfragen alle Informationen zu rendern. Weiterhin stellt unser Produkt nur Anfragen an die RDF-Daten und soll diese in keinster Weise verändern. Ziel des Projektes ist, dass es auf Linux Systemen ausführbar ist. Die Nutzung auf Windows- oder Mac-Systemen spielt für uns nur eine untergeordnete Rolle. Eine wichtige Anforderung des Kunden ist, dass das Produkt leicht zu warten und zu erweitern ist, um einen langen Nutzungszeitraum zu gewährleisten. Außerdem wird gefordert, dass die erstellten Websites in einer sinnvollen Ordnerstruktur abgelegt werden, um eine gewisse Übersichtlichkeit zu gewährleisten. Weiterhin soll unser Plugin nach unserem Projekt weiterentwickelt und genutzt werden, weswegen beschlossen wurde, eine Test-Coverage von 100% anzustreben. Als Abnahmekriterium wird eine Test-Coverage von mindestens 90% festgelegt.

2 Voraussetzungen

2.1 An das Team

Als zentrale Bedingung für das erfolgreiche Beenden unseres Projekts sehen wir als Team eine gute gruppeninterne Kommunikation, die es ermöglichen soll, Aufgaben zu verteilen und Probleme bzw. aktuelle Themen zu diskutieren. Hierzu haben wir uns auf Slack meherere Chat-Kanäle eingerichtet, um jederzeit mit der ganzen Gruppe Informationen austauschen können. Weiterhin nutzen wir das Projektmanagement-Tool Jira, das uns helfen soll die Aufgaben aufzuteilen und den Arbeitsaufwand zu protokollieren.

2.2 An die Nutzer

Der Nutzer sollte ein grundlegendes Verständis des Semantic Web und von RDF-Dateien besitzen, da unser Produkt darauf basiert und es für die erfolgreiche Benutzung nicht wegzudenken ist. Weiterhin sollte der User SPARQL -Abfragen beherrschen, da diese in unseren Templates eine elementare Rolle spielen. Weiterhin ist auch etwas Erfahrung in Sachen Templating für den User von Vorteil, um das Produkt in seinem vollen Umfang zu nutzen. Selbstverständlich muss ein Verständnis des Static Site Generation Tools Jekyll vorhanden sein. Außerdem muss der Kunde eigene RDF-Daten mitbringen und die Website schreiben, da unser Projekt lediglich die Möglichkeiten bietet statische HTML-Websites zu generieren, wir jedoch keinesfalls für die Daten des Users und dem Layout der generierten Seiten zuständig sind. Weiterhin muss der Nutzer unseres Produktes seinen eigenen Webserver besitzen, um die erstellte Website auch praktisch einsetzen

zu können. Da mit unserem Programm statische Webseiten erstellt werden muss mehr Festplattenspeicher eingeplant werden als für dynamische Websites (dafür wird weniger Arbeitsspeicher (von Seiten des Servers) benötigt).

3 Designübersicht und Funktionalität

3.1 Design

Die Anwendung basiert auf Jekyll, d.h. der Anwender ruft bestehende Ruby-Commands in der Kommandozeile auf. Sie soll als Jekyll-Plugin (`http://jekyllrb.com/docs/plugins/`) ausgeliefert werden. Das Plugin wird im Optimalfall (Kann-Kriterium) als Ruby gem ausgeliefert. Die Installation bei vorhandener Ruby-Installation erfolgt dann so:

```
gem install jekyll-rdf
```

Die Jekyll-gem wird dann aufgrund definierter Abhängigkeit gleich mitinstalliert. Nachfolgend eine beispielhafte Verwendung der Anwendung:

```
jekyll new mystaticrdf
cd mystaticrdf
jekyll build -rdf /path/to/rdf/file.ttl
```

Das heißt, wir erweitern den jekyll build Command um die RDF-Option oder - falls die RDF-Daten in einer Konfigurationsdatei abgelegt wurden - greift der build-Prozess auf die RDF automatisch zu. Anstatt jekyll mit neuen Befehlen zu erweitern, sollten existierende Befehle wie "build"überschrieben werden, damit bestehende Hooks, wie z.B. für Github Pages weiter funktionieren.

3.2 Funktionalität

FN01-01	Übergabe eines Pfades zum RDF-File an den `jekyll build` Command (siehe Design) und/oder Suche in einem Default-Verzeichnis	Muss
FN01-02	Unterstützter Dateityp: Turtle	Muss
FN01-03	Unterstützter Dateityp: N-Triples	Kann
FN01-04	Unterstützter Dateityp: N-Quads	Kann
FN02-01	Im `jekyll build`-Prozess werden die RDF-Daten über die rdf-Gem eingelesen und in einem "Data-Store" zwischengespeichert.	Muss
FN02-02	Die zu speichernden Daten können durch eine SPARQL-Query eingeschränkt werden. Diese kann in der Config-Datei hinterlegt werden. Wird keine Query hinterlegt, werden alle URIs im "Data-Store" abgelegt.	Muss
FN02-03	Es gibt eine Auswahl an vordefinierten Einschränkungen (z.B nur Subjekte, nur Prädikate, usw.)	Kann

FN03	Im `jekyll build`-Prozess wird ein neuer RDF-Generator aufgerufen, der für jede RDF-Ressource im "Data-Store" eine neue Page-Instanz anlegt, also die Vorbereitungen dafür trifft, dass später für jede RDF-Ressource ein HTML-Dokument erzeugt wird.	Muss
FN04-01	Im `jekyll build`-Prozess wird jede zu einer RDF-Ressource gehörende Page gerendert. Dazu wird entweder das klassen-/ressourcenspezifische Template oder ein allgemeines Default-Template herangezogen.	Muss
FN04-02	Das klassen- und ressourcenspezifische Mapping auf Templates wird auf Basis einer Mapping-Konfigurationsdatei durchgeführt. Gibt es für die Klasse einer Ressource kein direktes Mapping werden sukzessive die Oberklassen überprüft. Wenn auch dann kein Mapping gefunden werden kann, wird das Default-Template herangezogen.	Muss
FN04-03	Ist nicht eindeutig klar, welches Template für eine Ressource verwendet werden soll, z.B. weil mehrere Oberklassen existieren, so wird eine Warnung ausgegeben.	Muss
FN04-04	Das ausgelieferte Default-Template sorgt dafür, dass alle Ressourcen untereinander verlinkt werden bzw. ggf. ein externer Link gesetzt wird.	Muss
FN04-05	Der Anwender kann das Default-Template nach Belieben anpassen. Die aktuelle URI wird zur Vefügung gestellt und kann im Template verwendet werden.	Muss
FN04-06	Der Anwender kann klassen- und ressourcenspezifische Templates erstellen und nach Belieben anpassen. Die aktuelle URI wird zur Vefügung gestellt und kann im Template verwendet werden.	Muss
FN05-01	Es wird ein neuer Liquid-Tag eingeführt, über den auf ein mit der aktuellen Ressource als Subjekt in Verbindung stehendes Objekt durch Angabe der URI des Prädikats zugegriffen werden kann. Je nach Datentyp wird ein entsprechendes HTML-Konstrukt gerendert. QNames müssen in der Konfigurationsdatei definiert werden.	Muss
FN05-02	FN05-01 invers (d.h. das Subjekt wird mithilfe des Objekts gefunden)	Kann

FN05-03	FN05-01 für Listen - Es wird ein neuer Liquid-Tag eingeführt, über den auf mehrere mit der aktuellen Ressource als Subjekt in Verbindung stehende Objekte durch Angabe der URI des Prädikats zugegriffen werden kann. QNames müssen in der Konfigurationsdatei definiert werden. Die Ergebnismenge wird als HTML-Liste gerendert, d.h. der Benutzer hat keine Möglichkeit, selber über die Ergebnismenge zu iterieren. Für diesen Fall müsste er sich FN05-05 bedienen.	Muss
FN05-04	FN05-03 invers (d.h. die Subjekte werden über das Objekt gefunden)	Kann
FN05-05	Es wird ein neuer Liquid-Tag eingeführt, der es ermöglicht, über die Ergebnismenge einer SELECT SPARQL-Query zu iterieren. In der SPARQL-Query kann ein Platzhalter für die aktuelle URI verwendet werden.	Muss
FN05-06	Es wird ein neuer Liquid-Tag eingeführt, der es ermöglicht, über die Ergebnismenge einer CONSTRUCT SPARQL-Query zu iterieren oder diese in weiteren Abfragen gemäß FN05-05 zu verwenden. In der SPARQL-Query kann ein Platzhalter für die aktuelle URI verwendet werden.	Kann
FN05-07	Die Sprachvariante (z.B. @en oder @de) eines Literals kann am Tag spezifiziert werden.	Muss
FN05-08	Sind für ein Literal mehrere Sprachvarianten gegeben und ist die Sprache nicht durch den Tag spezifiziert, so fallen wir auf einen Defaultwert zurück.	Kann
FN06	Die übrige Funktionalität von Jekyll bleibt erhalten - Ausnahmen werden dokumentiert.	Muss
FN07	Der `jekyll serve` Command kann sinnvoll verwendet werden, d.h. die Seite wird zur Entwicklungszeit dynamisch ausgeliefert.	Kann
FN08-1	Das jekyll-Plugin wird als gem ausgeliefert.	Kann
FN08-2	Der `jekyll new` Command wird so erweitert, dass eine Beispielanwendung inkl. Datensatz erstellt wird.	Kann
FN08-3	Es wird ein einfach gehaltenes Default-Template als Fallback ausgeliefert.	Muss

4 Arbeitspakete

4.1 Muss-Ziele

1. Vorprojekt [FN02-01, FN03, FN04-01]
   - Zeitlicher Aufwand: 15%
2. Handling RDF-Daten [FN01-01, FN01-02, FN02-02]

- Zeitlicher Aufwand: 10%

3. Verarbeitung der RDF-Daten in Templates und Konfigurationen ermöglichen [FN04-02, FN04-03, FN04-04, FN04-05, FN04-06]
   - Zeitlicher Aufwand: 30%
4. Anpassung der benutzten Templating Sprache Liquid durch neu einzuführende Tags [FN05-01, FN05-03, FN05-05, FN05-07]
   - Zeitlicher Aufwand: 20%
5. Dokumentation, Handbuch, Default-Template [FN06]
   - Zeitlicher Aufwand: 25%

4.2 Kann-Ziele

Die prozentualen Angaben beziehen sich auf den zeitlichen Aufwand der MUSS-Ziele, d.h. ein Arbeitspaket aus den MUSS-Zielen mit 10% nimmt in etwa den gleichen absoluten zeitlichen Aufwand wie ein Arbeitspaket aus den KANN-Zielen mit 10% ein.

- Unterstützung weiterer Datentypen [FN01-03, FN01-04],
  Auslieferung als Ruby gem [FN08-01]
  - Zeitlicher Aufwand: 10% (inkl. 2% Zusatzdokumentation)
- Filter-Helper für Abfragen (Einschränkung ohne aufwändige SPARQL-Anfragen) [FN02-03]
  - Zeitlicher Aufwand: 20% (inkl. 5% Zusatzdokumentation)
- Erweiterung des 4. Pakets um inverse Suchen [FN05-02, FN05-04],
  Iteration über neue Ergebnismengen durch Construct [FN05-06],
  Default-Wert für Sprache [FN05-08]
  - Zeitlicher Aufwand: 20% (inkl. 2% Zusatzdokumentation)
- Erweiterung von jekyll-Befehlen: `jekyll serve` [FN07] und `jekyll new` [FN08-02]
  - Zeitlicher Aufwand: 20% (inkl. 5% Zusatzdokumentation)

5 Vorprojekt

Im Vorprojekt wird prototypisch die Funktionalität von Hypstrr demonstriert. Es stellt also grundlegend bereits die spätere Hauptfunktion zur Verfügung. Um dies angemessen zu repräsentieren werden folgende Punkte umgesetzt:

- Auslesen von Daten aus einer RDF-Datenquelle [FN02-01]
- Verarbeitung der ausgelesenen Daten [FN03]
- Generierung von statischen HTML-Seiten unter Verwendung der vorbereiteten Daten und eines einfachen Templates [FN04-01]

Was wird benötigt?

- Eine lauffähige Jekyll-Installation
- Ein RDF-Datenset

Was muss umgesetzt werden? Beim Ausführen des Befehls "jekyll build"muss zunächst eine RDF-Datenabfrage mittels RDF.rb ausgeführt. Die empfangenen Daten werden im Datenspeicher von Jekyll abgelegt und von dort aus im normalen Arbeitsfluss weiterverarbeitet. Dazu ist es notwendig, dass eine Möglichkeit gegeben ist im Template auf die einzelnen RDF-Daten (Im konkreten Fall Name der URI) zugreifen zu können.

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.

7 Quellen

- http://www.w3c.de/about/
- http://www.w3.org/RDF/
- http://stackoverflow.com/questions/9755113/how-are-rdf-and-rdfs-related
- https://de.wikipedia.org/wiki/Uniform_Resource_Locator
- https://de.wikipedia.org/wiki/Metadaten
- https://de.wikipedia.org/wiki/RDF-Schema
- https://de.wikipedia.org/wiki/SPARQL
- https://en.wikipedia.org/wiki/OntoWiki
- http://aksw.org/Projects/OntoWiki.html
- https://web.archive.org/web/20090208105225/
- http://www.digitaldivide.net/articles/view.php?ArticleID=20
- http://libreas.eu/ausgabe15/texte/001.htm