Universität Leipzig
Softwaretechnik-Praktikum
Sommersemester 2014

Entwurfsbeschreibung Gesamtprojekt

Projekt	Graphical SPARQL Builder
Gruppe	s14.swp.gsb
Verantwortlich	Siegfried Zötzsche
Erstellt am	12. April 2014

Inhaltsverzeichnis

1. Allgemeines

Um Anfragen mit SPARQL an RDF-basierte Datenbanken formulieren zu können muss zunächst eine gewisse Einstiegshürde überwunden werden. Selbst für weniger komplexe Anfragen müssen grundlegende Syntax und Vokabular der SPARQL vertraut sein.

Mit dem Graphical SPARQL Builder (GSB) soll diese Einstiegshürde gesenkt werden. Dazu setzt der GSB auf eine graphische Repräsentation der Anfrage gegenüber einer rein textbasierten Anfrage in SPARQL.

Zum einen soll eine Anfrage möglichst intuitiv mithilfe einer graphischen Benutzeroberfläche erstellt werden können, zum anderen soll die Anfrage auch mit minimaler Vorkenntnis von RDF und SPARQL lesbar sein.

2. Produktübersicht

Abbildung 1 gibt eine Übersicht über die Eingliederung und Nutzung des Graphical SPARQL Builder beim Einsatz des Tools im Zusammenspiel mit beteiligten Personen und externen Softwarekomponenten:

Im typischen Anwendungsfall möchte ein User (ohne SPARQL Kenntnisse) eine Anfrage an eine RDF-basierte Datenbank stellen. Dazu stellt er sich die Anfrage grafisch im GSB zusammen und sendet sie ab. Der GSB übersetzt die grafische Repräsentation der

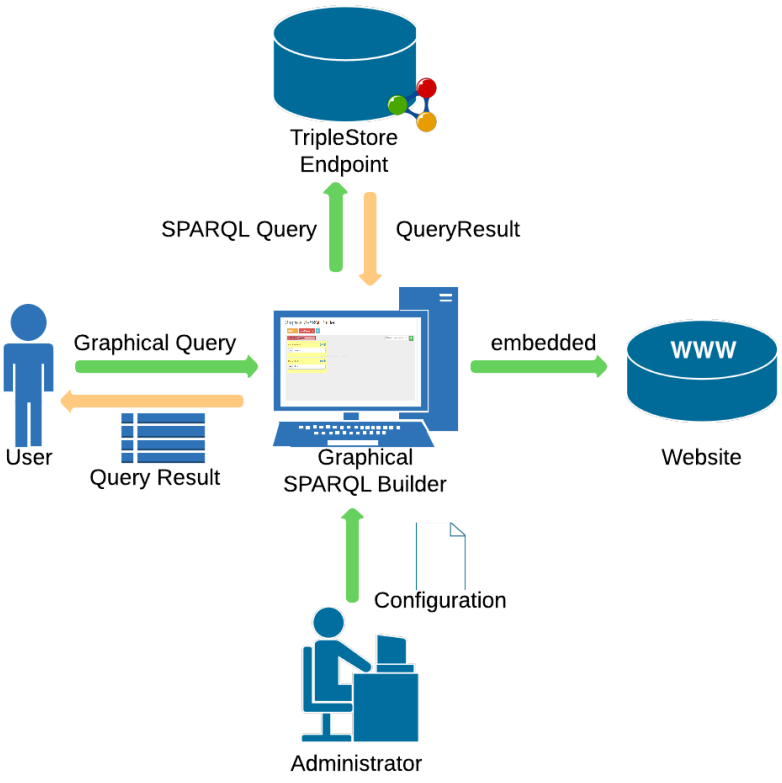

Abbildung 1: Beziehung des GSB zu Anwender, Datenbank und Administration.

Anfrage (zunächst in ein intern vordefiniertes JSON-Format und dann) in eine semantisch und syntaktisch korrekte SPARQL-Anfrage. Das Tool sendet diese dann an einen vom Administrator eingestellten SPARQL-Endpunkt eines TripleStores, von welchem das Anfrageergebnis zurückgegeben wird. Der GSB leitet das Ergebnis an den User weiter.

Der Administrator hat die Möglichkeit über Konfigurationsdateien diverse Einstellungen, wie Sprachwahl, Einschränkungen der bereitgestellten Datenbank, Expertenansicht und zahlreiche GUI-Anpassungen vorzunehmen. Als Single-Page-Anwendung kann der GSB in bestehende Websites eingebettet werden.

3. Grundsätzliche Struktur- und Entwurfsprinzipien

Die Grundsätzliche Struktur des GSB ist im Komponentendiagramm (Abb. 2) schematisch dargestellt.

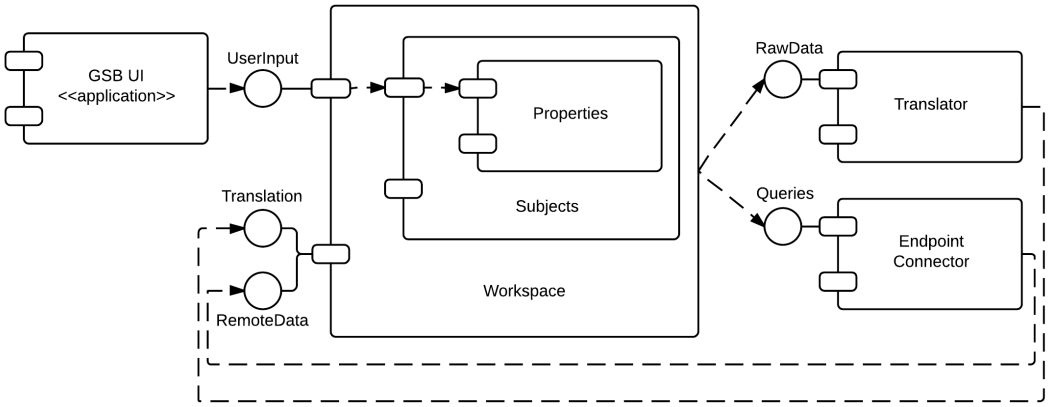

Abbildung 2: Komponentendiagramm des GSB.

Es gibt die Komponente des User Interface (GSB UI), in welcher alle Eingaben des Benutzers des Tools stattfinden. Diese werden in die Komponente Workspace weitergeleitet und dort gegebenenfalls an die Komponenten Subjects und deren Unterkomponente Properties weitergeleitet. Um dies zu verdeutlichen, hier drei Beispiele:

1. Der Nutzer setzt eine Übersetzung eines GSBL-Wortes in Gang, dies betrifft die Komponente “Workspace”

2. Der Nutzer ändert den Alias eines Subjekts, dies würde über die Workspace-Komponente zu der Subjects-Komponente weitergereicht werden.

3. Der Nutzer ändert die Sichtbarkeit einer Eigenschaft, dies würde über die Workspace-Komponente zu der Subjects-Komponente zum Ziel, der Properties-Komponente, weitergereicht werden.

Die Workspace-Komponente steht in Verbindung mit der Translator-Komponente, welche für die Übersetzung der erstellten Rohdaten zu JSON bzw. SPARQL zuständig ist und die Ergebnisse zurück zur Workspace-Komponente überträgt. Ebenso gibt es eine Verbindung zur Endpoint-Connector-Komponente, die die Kommunikation mit einem Endpoint übernimmt. Es können Anfragen gesendet und Ergebnisse empfangen werden.

4. Struktur- und Entwurfsprinzipien einzelner Pakete

User Interface

Bei der Hauptversion des Projekts besteht das Userinterface aus dem Workspace, der die graphisch konstruierte Abfrage beinhaltet, dem Feld in dem nach dem Drücken des "Build QueryButtons die Übersetzung der Anfrage in JSON und in SPARQL zu sehen ist. Im Workspace befindet sich der Hinzufüge-Button für Subjekte sowie ein LIST ALL Objekt. Ein weiterer Button ermöglicht das Speichern der Anfrage im JSON-Format.

Das Laden einer gespeicherten Anfrage wird dem Nutzer durch Drag & Drop ermöglicht. Standardmäßig werden zu Beginn zwei Subjekte (Mensch und Stadt) in den Workspace geladen. Subjekte und Properties sind durch Icons entfernbar sowie in der Abfrage anzeigbar. Ein Link über dem SPARQL-Result-Feld ermöglicht zusätzlich das Anzeigen des Ergebnisses der übersetzten Anfrage, welches zurückgegeben wird nachdem die Anfrage an einen Endpunkt geschickt wurde.

Workspace

Der Workspace umfasst die Sammlung von Subjekten, deren Properties und den Startpunkt sowie die Verbindungen (Linien) zwischen Subjekten und dem Startpunkt und einem Subjekt.

Subjects

Subjects können durch die Icons entfernt (Papierkorb), in der Anfrage aus-/eingeblendet (Auge) und hinzugefügt (Plus-Symbol) werden. Die Properties eines Subjects sind ausblendbar.

Subjects sind Instanzen die mit den jeweiligen ëxternProperties Verbindungen untereinander aufbauen und Subject-interne Properties sammeln. Miteinander verknüpfte Subjects sind durch Linien verbunden. Subjects können durch Drag & Drop auf dem Workspace verschoben werden. Sie bilden den grundsätzlichen Anhaltspunkt für die Übersetzung. (siehe **Translator**)

Properties

Properties können durch die Icons entfernt (Papierkorb), in der Anfrage aus-/eingeblendet (Auge) und hinzugefügt (Add-Button beim Subject-Mouseover) werden. Sie stellen über eine Auswahl im Drop-Down Menü eines Subjects Verbindungen zwischen Subjects her. Properties sind im Vergleich zum Vorprojekt um arithmetische Operatoren (`+`, `-`, `*`, `/`, `COUNT`, `MAX`, `MIN`, `SUM`) erweitert.

Translator

Die Translator-Funktion ist auch in der Hauptversion des Projekts zweigeteilt. Eine Übersetzung liefert auf Basis der im Workspace zusammengestellten graphischen Anfrage (in GSBL) eine Abfrage im JSON-Format und eine weitere Übersetzung von JSON in SPARQL wird danach durchgeführt.

Die Translation wird durch Klick auf den Button "Build Query"gestartet und das Ergebnis wird in den unteren Feldern (JSON- und SPARQL-Resultat) angezeigt. Algorithmisch betrachtet beginnt die Übersetzung bei dem mit dem Startpunkt verknüpften Subject und übersetzt dann rekursiv alle damit verknüpften Subjects und deren Properties, sodass nicht verknüpfte Subjects ignoriert werden.

Endpunkt-Anbindung

Anstatt wie im Vorprojekt lokal gespeicherte MockupDaten zu nutzen, wird im Hauptprojekt auf einen SPARQL-Endpunkt zugegriffen. Die Auswahl gegebener Subjekte beim Hinzufügen, deren mögliche Eigenschaften sowie die LinkedSubjects werden am Endpunkt abgefragt und dann im UI zur Verfügung gestellt. Je nach Vorankommen der Projektarbeiten wird ein mehr oder weniger aufwendiger Cache realisiert um starke Redundanz bei den Anfragen an den Endpunkt zu vermeiden.

5. Datenmodell

Die Grundbausteine des RDF-Datenmodells sind Ressourcen, Eigenschaften und Aussagen. Durch diese ist es möglich RDF Ausdrücke syntaxfrei darzustellen.

Ressource

Eine Ressource wird mit RDF Ausdrücken beschrieben. Folglich muss diese durch eine URI (Universal Resource Identifier) referenzierbar sein und eindeutig von dieser identifiziert werden können.

Eigenschaft

Eigenschaften sind die Attribute von Ressourcen. Im Datenmodell legen sie die erlaubten Werte, den Typ und die Relation der Ressource zu anderen Ressourcen fest.

Aussage

Eine Ressource, kombiniert mit ihrer Eigenschaft und den Wert, den die Eigenschaft dieser Ressource hat, nennt man Äussage". D.h. eine Aussage besteht aus Subjekt-Prädikat-Objekt (Ressource-Eigenschaft-Wert).

Beispiel

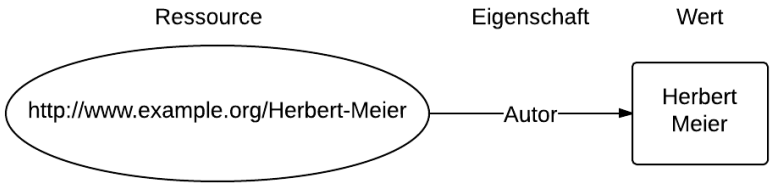

Abbildung 3: Beispiel eines RDF Tripels.

Aussage	Der Autor der Ressource ist Herbert Meier
Subjekt	`http://www.example.org/Herbert-Meier`
Prädikat	Autor
Objekt	"Herbert Meier"

Tabelle 1: Beispiel eines RDF Tripels.

Repräsentation in JSON

Im GSB werden die Informationen der graphischen, zu einer Anfrage zusammengestellten Elemente zunächst innerhalb eines JSON-Objekts gespeichert. Dies soll einen Import/Export der gebauten Anfragen ermöglichen, und dient gleichzeitig als Eingabeparameter für den Übersetzer, welcher das JSON-Objekt schließlich in eine gültige SPARQL-Anfrage wandelt. Die Erstellung dieses Objekts wird dabei erheblich durch die objektorientierte Repräsentation der grafischen Elemente erleichtert.

Im JSON-Objekt werden alle wichtigen Parameter einer Anfrage festgehalten:

- In einem START-Objekt werden zunächst Typ und Linkpartner des Startpunktes gespeichert.
- Danach folgen in einem SUBJECTS-Array alle vorkommenden Subjekte, mit wichtigen Variablen wie URI, Alias, Typ, etc.
- Innerhalb eines Subjekts befindet sich schließlich das properties-Array, in dem alle für das Subjekt gewählten Eigenschaften, inklusive deren eigene Parameter (Alias, URI etc.), zu finden sind.

Die Vorlage für ein solches JSON-Objekt befindet sich in Anhang B.

6. Testkonzept

Für die Qualitätssicherung dieses Projekt werden zunächst einzelne Komponenten auf ihre 'interne Qualität' hin getestet, und schließlich die Zusammenführung dieser. Für Komponenten- bzw Integrationstests bietet das für den GSB genutzte Framework AngularJS dabei bereits eigene, potente Werkzeuge an [1]. Diese erlauben losgelößte Tests einzelner Funktionen, ohne etwa von der Antwort eines XMLHttpRequest anhängig sein

und diesen stattdessen zu simulieren. Für Systemtests dagegen bietet Protractor ein Testframework für AngularJS-Applikationen [2]. Dabei kann die Anwendung direkt im Browser getestet und mit ihr aus der Sicht eines Users interagiert werden.

Besonders in den abschließenden Systemtest-Läufen sind identifizierte Fehler zu dokumentieren, sowie, wenn möglich, deren Ursachen und Bereinigung. Dadurch kann die Vermeidung ähnlicher Probleme erleichtert werden, und Anhaltspunkte für die Suche mögliche Fehlerquellen bereitgestellt werden.

Es soll außerdem ein besonderes Augenmerk auf Nutzungstests durch Projekt-fremde User gelegt werden. Dadurch kann Feedback zu den für den GSB essentiellen Aspekten Benutzbarkeit und Intuitivität gewonnen werden, von Laien, die ihrerseits keine weitreichenden Erfahrungen mit SPARQL haben und sich dadurch mit der Zielgruppe des Projekts decken. Auch das Aufdecken möglicher Ausfälle bei falscher Benutzung sind denkbar. Im Besonderen betrifft dies die Mitarbeiter der Bibliotheca Albertina, welche als mögliche Abnehmer des GSB bereits feststehen. Deren Rückmeldungen sollen insofern stark gewichtet werden, da sie besonders “praxisnahe” Probleme aufzeigen und Verbesserungsvorschläge liefern könnten. Zu diesem Zweck ist auch ein regelmäßiges Treffen geplant.

Literatur

[1] http://docs.angularjs.org/guide/unit-testing

[2] https://github.com/angular/protractor

[3] http://de.wikipedia.org/wiki/DBpedia

[4] http://de.dbpedia.org/

[5] http://wiki.dbpedia.org/Datasets

[6] http://dbpedia.org/sparql

[7] http://pcai042.informatik.uni-leipzig.de/~swp14-gsb/

[8] http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[9] http://de.wikipedia.org/wiki/Ontologie

[10] http://www.iais.fraunhofer.de/fileadmin/user_upload/Abteilungen/OK/PDFs/Alieva_Magisterarbeit.pdf

[11] http://www.w3.org/TR/rdf-primer/

[12] http://de.wikipedia.org/wiki/Resource_Description_Framework

[13] https://en.wikipedia.org/wiki/RDF/XML

[14] http://www.w3.org/TR/rdf-sparql-query/

[15] http://en.wikipedia.org/wiki/SPARQL

A. Glossar

API Application Programming Interface – Programmierschnittstelle. Eine API beschreibt, wie Software-Komponenten bzw. Programme miteinander interagieren können/-sollten. Anders ausgedrückt: eine API ist der Teil eines Softwaresystems, der anderen Programmen zur Verfügung gestellt wird um mit dem Softwaresystem zu interagieren.

DBpedia DBpedia ist eine Datensammlung im RDF Format, deren Datensätze aus der Wikipedia extrahiert wurden. Ziel ist es strukturierte Daten für Webanwendungen zur Verfügung zu stellen. [3, 4, 5]

Endpoint Ein Endpoint ist eine Schnittstelle zwischen der Datensammlung und der Abfragesprache. Nachdem eine Anfrage an den Endpoint gesendet wurde (query) sendet selbiger die Ergebnisse zurück. Ein Beispiel für einen SPARQL-Endpoint ist der "Virtuoso SPARQL Query Editor". [6]

GSB Graphical SPARQL Builder, der Name des Projekts. [7]

i18n internationalization and localization – Anpassung der Software an andere Sprachen und Kulturen ohne Quelltext zu ändern. Sprach- und Kulturspezifika werden über Konfigurationsdateien angepasst.

Ontologie Ontologien sind formalisierte Vokabulare von Begriffen. Diese Vokabulare beziehen sich meist auf eine bestimmte Domäne (Gegenstandsbereich) oder Nutzergruppe. Sie liegen in einer sprachlichen Form vor und umfassen die Begriffe einer Domäne sowie Beziehungen zwischen den Begriffen. [8, 9, 10]

OWL Die Web Ontology Language (OWL, aktuelle Version OWL2) ist eine Beschreibungssprache um Ontologien für das semantische Web zu erstellen und zu publizieren. OWL2-Ontologien können für Informationen verwendet werden, die in RDF geschrieben sind und werden hauptsächlich in Form von RDF-Dokumenten ausgetauscht. [8]

RDF "Resource Description Framework" kurz RDF ist eine Strukturierung von Daten nach dem Muster Subjekt-Prädikat-Objekt. Alle RDF-Daten werden in diesem Tripel-Format gespeichert. RDF gilt als eines der Basis-Elemente des semantischen Webs. Repräsentationen, also syntaktische Standards, des RDF-Prinzips sind N3 (Notation3), Turtle (Terse RDF Triple Language) sowie RDF/XML. Turtle und N3 gelten im Vergleich zu RDF/XML als benutzerfreundlicher. [11, 12, 13]

SPARQL "SPARQL Protocol And RDF Query Language" kurz SPARQL ist eine Abfragesprache für das Datenformat RDF. SPARQL ist graphbasiert und gilt nach dem W3C als Standard für RDF-Abfragen. [14, 15]

Triplestore Ein Triplestore ist ein System zur Speicherung, Verwaltung und Bearbeitung einer Sammlung von RDF-Tripeln. Ein Triplestore bietet gewöhnlich APIs, Reasoning-Verfahren sowie Abfragemöglichkeiten. [10]

B. JSON-Template

```
{
  "START" : {
    "type" : START_TYPE,
    "link" : {
      "direction"   : TO,
      "linkPartner" : HAUPTKLASSEN_ALIAS
    }
  },

  "SUBJECTS": [
    {
      "alias"                : ALIAS,
      "label"                : LABEL,
      "uri"                  : URI,
      "comment"              : COMMENT,
      "view"                 : BOOLEAN,
      "showAdditionalFields" : BOOLEAN,
      "properties"           : [
        { "alias"         : ALIAS,
          "uri"           : URI_PROPERTY,
          "type"          : PROPERTY_TYPE,
          "propertyRange" : PROPERTY_RANGE,
          "view"          : BOOLEAN,
          "operator"      : OPERATOR,
          "link"          : {
            "direction"     : TO_OR_FROM,
            "linkPartner"   : KLASSEN_ALIAS
          },
          "arithmetic"    : {
            "operator"      : OPERATOR,
            "amount"        : VALUE
          },
          "compare"       : {
            "operator"      : OPERATOR,
            "amount"        : VALUE
          }
        },
        { "alias"         : ALIAS,
          "uri"           : URI_PROPERTY,
          "type"          : PROPERTY_TYPE,
          ...
        },
          ...
      ]
    },
    {
      "alias"                : ALIAS,
      "label"                : LABEL,
      "uri"                  : URI,
      "comment"              : COMMENT,
      "view"                 : BOOLEAN,
      "showAdditionalFields" : BOOLEAN,
      "properties"           : [ ... ]
    },
      ...
  ]
}
```