1-2013 | Frühjahr | www.ijug.eu

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

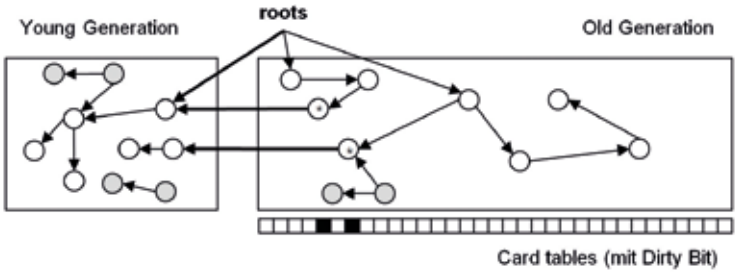
Java aktuell

Java – Programmiersprache mit Tiefgang

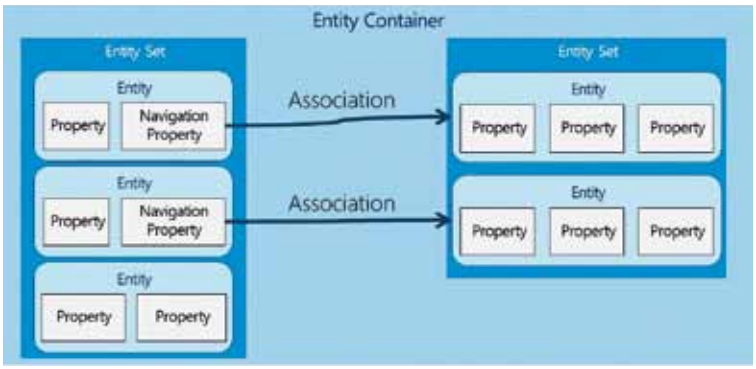
D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

iJUG Verbund

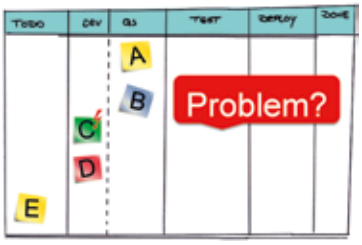
Sonderdruck



Intergenerational References bei der Garbage Collection, Seite 36



Das Entity Data Model im Open Data Protocol, Seite 30



Agiles Entwickeln, Seite 61

Das neue Framework, Seite 18

Linked Data – ein Web aus Daten

Angelo Veltens, http://datenwissen.de

Wer heute im Web Daten abrufen möchte, hat es nicht leicht. Obwohl hinter vielen Diensten große Datenbanken stehen, beschränkt sich das Surfen im Web meist noch auf menschenlesbare Webseiten. Entwickler können bestenfalls den mühsamen Umweg über proprietäre APIs gehen. Mit Linked Data werden die Daten Teil des Webs und das dokumentenbasierte World Wide Web erweitert sich um ein Web aus Daten.

Trotz aller modernen Web-Anwendungen und -Services ist das World Wide Web (WWW) heute immer noch ein Web aus Dokumenten. Diese mögen dynamisch vom Server erzeugt sein und Megabytes an JavaScript enthalten, sie mögen XML-Daten oder JSON-Objekte beinhalten – dennoch sind es Dokumente. Dabei ist das WWW voller Daten. Hinter praktisch jeder Webseite verbirgt sich heute eine Datenbank, sei es das private Wordpress-Blog oder ein großer Online-Shop. Wir sitzen auf wahren Datenschätzen. Doch weil wir sie hinter menschenlesbaren Web-Seiten und heterogenen Web-Services verstecken, bleibt ihr Nutzen verhältnismäßig gering. Sie sind gefangen in Daten-Silos und nur über die vom Betreiber festgelegten Möglichkeiten abrufbar. Mithilfe von Linked Data können wir sie aus ihren Silos befreien und ein globales Web aus Daten schaffen, das einen grundlegend neuen Umgang mit Daten ermöglicht.

Von Daten im Web …

HTML-Dokumente liegen nicht einfach „im Web" – sie sind das Web. Sie bilden es dadurch, dass wir sie miteinander über ihre URLs verlinken. Wir könnten stattdessen Textdateien in ZIP-Archive verpacken und auf einem FTP-Server zum Download anbieten. Das wäre jedoch kein Web. Kurioserweise tun wir aber genau das mit unseren Daten. Wir verpacken sie fein säuberlich und bieten sie zum Download an oder verstecken sie in HTML-Dokumenten. Der Weg über einen Web-Service ist der erste Schritt in die richtige Richtung, stellt uns allerdings dennoch vor einige Probleme.

Angenommen, jemand arbeitet an einer Anwendung, die auf mehrere verschiedene Web-Services zugreifen soll. Nehmen wir als Beispiel einen Dienst, der Nutzern die Bewertung von Hotels ermöglichen soll. Die Anmeldung soll über einen Facebook-Account möglich sein. Welche Hotels es gibt, ermitteln wir über einen Web-Service („Hotel-Service") und Informationen über den Urlaubsort, an dem das Hotel steht, fragen wir über einen weiteren Web-Service ab („Geo-Service").

Um diese Anwendung zu realisieren, müssen wir gegen drei Web-Services programmieren, also gegen drei verschiedene APIs mit drei verschiedenen Datenformaten. Wir müssen drei verschiedene API-Dokumentationen verinnerlichen und unsere Anwendung auf diese APIs zuschneiden. Hinzu kommen die Probleme der Verlinkung: Angenommen, der Hotel-Service liefert uns einen JSON-Datensatz zu einem Hotel (siehe Listing 1).

```
{
  id: 42,
  name: 'Hotel Fiktiva'
  city: 'Berlin'
}
```

Listing 1

Wie gelangen wir, ausgehend von diesem Datensatz, zu den im Geo-Service verfügbaren Informationen über den Urlaubsort? Es gibt keinen Link. Der Datensatz enthält lediglich die Zeichenkette „Berlin". Die passende Anfrage an den Geo-Service müssen wir als Entwickler mithilfe unseres menschlichen Hintergrundwissens und des Studiums der API-Dokumentation selbst herstellen. Eine automatische Verknüpfung der Daten ist nicht möglich.

Die Daten liegen zwar im Web, sind jedoch nicht Teil des Webs. Sie sind abrufbar, aber nicht miteinander verlinkt. Ihr Kontext geht nicht aus ihnen hervor und ihre Bedeutung erschließt sich nur über die jeweilige API-Dokumentation. Ein Entwickler muss sich jedem der drei Dienste separat widmen und die Verknüpfungen anschließend selbst herstellen. Und warum sollte er sich überhaupt auf drei Dienste beschränken? Warum kann er nicht den gesamten öffentlich im WWW verfügbaren Datenbestand für seine Anwendungen nutzen?

… zu einem Web aus Daten

Dieser Schritt zu einem Web aus Daten, durch das wir maschinell ähnlich browsen können, wie wir als menschlicher Nutzer heute das Web aus Dokumenten durchstöbern, wird durch Linked Data möglich. Das Konzept, Daten miteinander zu verlinken, stammt von Tim Berners-Lee, dem Erfinder des WWW. Er hat schon 2006 vier Grundprinzipien von Linked Data aufgestellt [1]. Diese lauten:

1. Nutze Uniform Resource Identifier (URI) als Name für Dinge
2. Nutze HTTP URIs, sodass man diese Namen nachschlagen kann
3. Wenn jemand einen URI abruft, stelle nützliche Informationen mittels Standards (RDF, SPARQL) bereit
4. Füge Links zu anderen URIs ein, sodass man weitere Dinge entdecken kann

Berücksichtigen wir diese Regeln, werden unsere Daten Teil des Webs. Sie sind einfach und verlangen dennoch ein grundlegendes Umdenken im Umgang mit Daten und den

Dingen im Web. Werfen wir nun also einen näheren Blick auf diese Prinzipien.

Alles bekommt einen URI

Ein URI ist eine globale ID. Während im heutigen Web vornehmlich Dokumente – seien es Web-Seiten oder XML-Dokumente – über sie identifiziert und auffindbar gemacht werden, verlangt Linked Data, dass wir alle möglichen Dinge über URIs identifizieren. Dieser Begriff des „Dings" ist essenziell für das Verständnis von Linked Data. Es ist nicht möglich, Produkte, Städte, Personen, Unternehmen und andere „Dinge" im Web zu veröffentlichen. Es ist jedoch möglich, diesen Dingen einen Namen in Form eines URI zu geben.

Wollten wir obiges Beispiel mithilfe von Linked Data realisieren, wären sowohl die Hotels als auch die Urlaubsorte durch einen URI identifizierbar. Wichtig ist, dass diese URIs tatsächlich die Hotels und Orte selbst identifizieren und nicht etwa die Website eines Hotels oder Urlaubsorts. Im „Web of Data" gibt es nicht mehr nur Dokumente, sondern auch „Dinge". Jedes dieser Dinge bekommt einen eigenen URI. Ein Hotel und seine Website sind zwei verschiedene Dinge und somit wird beides durch unterschiedliche URIs identifiziert. Statt von Dokumenten spricht man auch von Informations-Ressourcen, während Dinge, die keine Dokumente sind, als Nicht-Informations-Ressourcen bezeichnet werden.

Das zweite Prinzip verlangt die Verwendung von HTTP-URIs. Dies hat den pragmatischen Hintergrund, dass HTTP-URIs über das Domain-Name-System auflösbar und somit abrufbar sind. Dies gilt beispielsweise nicht für ISBN-URIs, die möglicherweise zur Identifikation von Büchern in Betracht gezogen werden könnten. Da diese URIs jedoch nicht abrufbar sind, ist es nicht möglich, das dritte Prinzip von Linked Data zu erfüllen, nützliche Informationen bei Abruf des URI bereitzustellen.

Dinge und Dokumente

Um überhaupt Informationen bereitstellen zu können, benötigen wir ein Dokument, das Daten enthält. Dieses müssen wir bei Abruf des URI einer Nicht-Informations-Ressource bereitstellen, da letztere sich nicht selbst im Web befinden kann. Nehmen wir an, unter „http://hotels.example/fiktiva" sind Informationen über das Hotel „Fiktiva" abrufbar. Das Hotel selbst muss, da es ein vom Dokument zu unterscheidendes „Ding" ist, über einen eigenen URI identifiziert werden. Um dennoch auf die im Dokument bereitgestellten Informationen zugreifen zu können, empfiehlt sich die Verwendung sogenannter „Hash-URIs" zur Identifikation von Nicht-Informations-Ressourcen. Dabei wird der Fragment-Identifier eines URI genutzt, um die im Dokument beschriebenen Dinge zu identifizieren.

Unser Hotel „Fiktiva" bekäme bei dieser Herangehensweise etwa den URI „http://hotels.example/fiktiva#it". Durch den Fragment-Identifier „#it", der Teil des URI ist, haben wir eine Unterscheidung zur Informations-Ressource mit dem Namen „http://hotels.example/fiktiva" geschaffen. Alternativ könnten wir das Hotel durch „http://hotels.example/hotel/fiktiva" oder einen beliebigen anderen URI identifizieren und mittels HTTP-Status-Code „303" (See Other) eine Weiterleitung zur Informations-Ressource bewirken. Die Verwendung von Hash-URIs hat jedoch den Vorteil, dass sie ohne diese Weiterleitung auskommt, da der Client automatisch den Fragment-Identifier abstreift, bevor er die Anfrage an den Server sendet, und somit selbstständig die Informations-Ressource anfordert.

Aber was sind nützliche Informationen? Nützlichkeit liegt im Auge des Betrachters. Während ein Mensch großen Nutzen an aufbereiteten, interaktiven HTML-Seiten hat, benötigen Dienste, die automatisiert auf ein Angebot zugreifen, maschinenlesbare Daten. Diesem Umstand können wir durch Content-Negotiation gerecht werden. Mithilfe dieser Technik lässt sich abhängig von der Anforderung des Clients eine andere Repräsentation einer Ressource ausliefern. Bleiben wir beim Hotel „Fiktiva" mit dem URI „http://hotels.example/fiktiva#it": Beim Abruf streift der Client den Fragment-Identifier ab und fordert das Dokument „http://hotels.example/fiktiva" an. Weiterhin setzt der Client den HTTP-Accept-Header auf den von ihm bevorzugten MIME-Type, um auszudrücken, welche Art der Repräsentation für ihn nützlich ist. Ein Browser wird beispielsweise „text/html" anfordern, während eine andere Anwendung möglicherweise „text/json" oder „application/rdf+xml" bevorzugt. Der Server versucht den Wünschen des Clients entsprechend, die passende Repräsentation des Dokuments auszuliefern (siehe Abbildung 1).

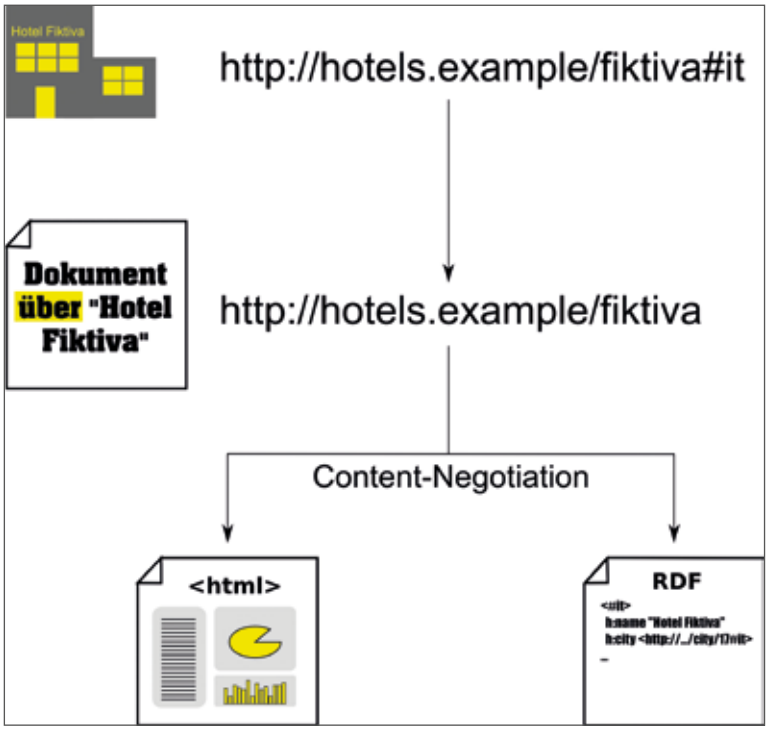


Abbildung 1: Abruf einer Nicht-Informations-Ressource und Content-Negotiation

RDF – die Sprache des „Web of Data“

Mittels Content-Negotiation ist sichergestellt, dass sowohl menschen- als auch maschinenlesbare Informationen bereitgestellt werden können, wenn man den URI einer Nicht-Informations-Ressource abruft. Wir werfen nun einen näheren Blick auf die nützlichen, maschinenlesbaren Daten im Sinne der Linked-Data-Prinzipien. In deren Zentrum steht das Resource Description Framework (RDF). Wie der Name schon sagt, wird es verwendet, um Ressourcen zu beschreiben. Diese können sowohl Informations- als auch Nicht-Informations-Ressourcen sein.

RDF organisiert die Informationen in sogenannten „Tripeln“, bestehend aus Subjekt, Prädikat und Objekt, ähnlich wie in einem einfachen natürlich-sprachlichen Satz. Subjekt ist die Ressource, die beschrieben wird, das Prädikat drückt aus, welche Aussage über diese Ressource getroffen wird, und das Objekt ist das Datum, das durch das Prädikat dem Subjekt zugesprochen wird. Wollen wir zum Beispiel ausdrücken, dass das Hotel „Fiktiva“ 100 Zimmer hat, könnte ein Tripel in der RDF-Turtle-Syntax wie in Listing 2 aussehen.

Der URI des Hotels ist in diesem Beispiel das Subjekt der Aussage, „hotel:anzahlZimmer“ ist das Prädikat, das die Anzahl der Zimmer ausdrückt, und „100“ der Wert der Aussage, also die tatsächliche Anzahl an Zimmern, die wir mit diesem Tripel mitteilen möchten. Ebenso leicht können wir mehrere Aussagen zu diesem Hotel treffen (siehe Listing 3).

Die Aussagen zu einem Subjekt werden durch Semikolon getrennt, die letzte Aussage durch einen Punkt abgeschlossen. Durch Kommas getrennt, können sogar mehrere Objekte einem Prädikat zugeordnet werden. Durch die Unterscheidung zwischen Ding und Dokument ist es sehr leicht, auch Metadaten zu hinterlegen, indem wir Aussagen über den URI des Dokuments (ohne „#it“) treffen (siehe Listing 4).

Dieses Tripel sagt aus, wann das Dokument zuletzt geändert wurde, und nicht etwa, wann das Hotel baulich verändert wurde. Durch die Unterscheidung zwischen Ding und Dokument wird also eine leichte Trennung von Daten und Metadaten möglich.

Zu betonen ist, dass RDF kein Datenformat, sondern tatsächlich ein Modell ist, das keine Syntax vorschreibt. RDF-Tripel können durch unterschiedliche Syntaxen dargestellt werden. Neben der für Menschen leicht zu lesenden, hier verwendeten Turtle-Syntax, gibt es zum Beispiel auch eine XML-Notation. Da es sich bei RDF-Daten letztlich um einen gerichteten Graphen handelt, lassen sie sich auch als solcher darstellen (siehe Abbildung 2). Man beachte, dass die Aussage „hotel:ort“ nicht auf ein Objekt-Literal, sondern auf einen anderen URI verweist. Wir setzen dadurch einen Link zu einer anderen Ressource und verlinken Daten.

```
<http://hotels.example/hotel/fiktiva#it>
  hotel:anzahlZimmer “100“.
```

Listing 2

```
<http://hotels.example/hotel/fiktiva#it>
  hotel:name “Hotel Fiktiva“;
  hotel:anzahlZimmer “100“;
  hotel:ort <http://sws.geonames.
org/2945024/>.
```

Listing 3

```
<http://hotels.example/hotel/fiktiva>
  dc:modified “2012-09-30“.
```

Listing 4

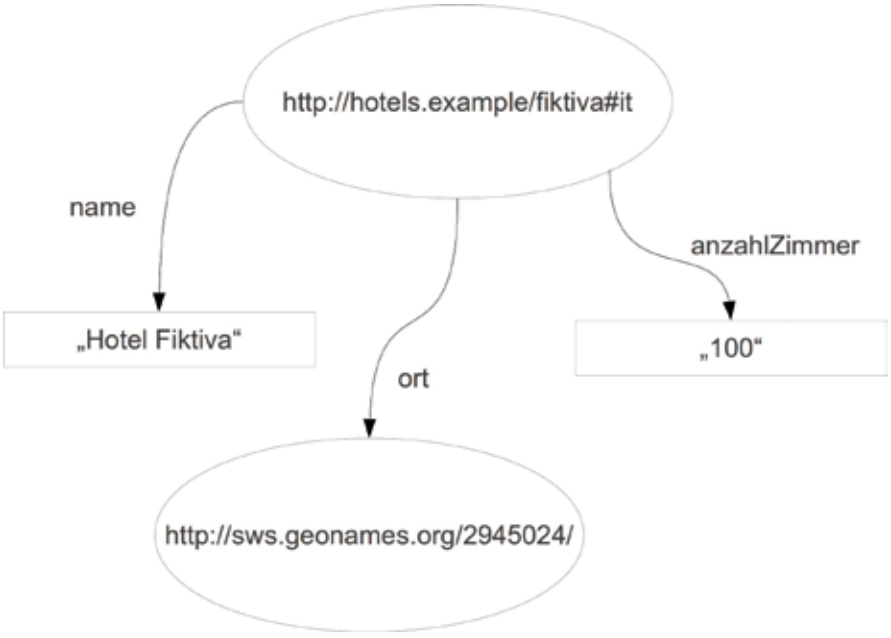


Abbildung 2: RDF-Graph

Daten verlinken

Die genannten Links sind der entscheidende Schritt zu einem Web aus Daten. Dadurch, dass wir Dinge über URIs identifizieren, ist es egal, wo wir die Informationen über diese Dinge hosten. Mithilfe dieser globalen ID sind sie auffindbar, wie jede Seite im WWW auch. Im obigen Beispiel verweist „hotel:ort“ zum Beispiel auf den URI „http://sws.geonames.org/2945024/“. Er ist die ID der Stadt Braunschweig beim Dienst „geonames.org“. Ruft man den URI ab, erhält man weitere Informationen über die Stadt, die in unserem ursprünglichen Datensatz gar nicht vorhanden sind, zum Beispiel die Einwohnerzahl. Dieser Datensatz verlinkt wiederum auf andere Dinge irgendwo im Web. Wir können den Links folgen, um weitere Daten und Zusammenhänge zu entdecken. Aber nicht nur die Subjekte und Objekte können über URIs identifiziert werden: Auch die verwendeten Prädikate selbst sind URIs. Sie sind Teil einer oder mehrerer Ontologien, die in der Turtle-Syntax vorab durch @prefix eingebunden werden (siehe Listing 5).

Hinter dem Präfix „hotel:“ verbirgt sich der URI „http://hotels.example/ontologie/“

```
@prefix hotel: <http://hotels.example/onto-
logie/>
<http://hotels.example/hotel/fiktiva#it>
  hotel:anzahlZimmer "100".
```

Listing 5

```
PREFIX hotel: <http://hotels.example/onto-
logie/>
SELECT ?name ?anzahl
WHERE {
  ?hotel hotel:name ?name;
        hotel:anzahlZimmer ?anzahl;
        hotel:ort <http://sws.geonames.
org/2945024/>.
  FILTER (?anzahlZimmer > 100)
}
```

Listing 6

und das Prädikat „hotel:anzahlZimmer" lautet vollständig „http://hotels.example/ontologie/anzahlZimmer". Ruft man den URI des Prädikats ab, erhält man wiederum RDF-Daten, die die Bedeutung des Prädikats beschreiben. Die Dokumentation der verwendeten Ontologien ist somit selbst Bestandteil des Webs aus Daten und maschinell auswertbar.

Das Web als globale Datenbank

Linked Data bildet ein Web aus maschinell auswertbaren Daten und macht das WWW so zu einer globalen Datenbank. Diese kann mit einer speziellen Abfragesprache namens SPARQL sogar abgefragt werden. Folgende Beispiel-Abfrage könnte alle Hotels in Braunschweig mit mehr als 100 Zimmern finden (siehe Listing 6).

Variablen wird ein Fragezeichen vorangestellt. Die SELECT-Clause beschreibt, welche dieser Variablen von Interesse sind. In der WHERE-Clause werden RDF-Tripel beschrieben, mit denen der Datenbestand abgeglichen wird. Das Ergebnis kann über Filter weiter eingeschränkt werden.

Während Suchmaschinen wie Google heute oft nur Fragen, aber keine Antworten finden, können semantische Suchen auf maschinell auswertbaren Daten operieren und exakte Ergebnisse finden. Eine solche Suchmaschine ist zum Beispiel „sindice.com". Die SPARQL-Schnittstelle [2] ist zwar noch in der Beta-Phase, lädt aber dennoch bereits zu Experimenten ein.

Linked Data in der Praxis

Linked Data kommt heute bereits an zahlreichen Stellen zum Einsatz. Eines der größten Projekte ist „Dbpedia" [3], das Informationen aus Wikipedia extrahiert und als Linked Data bereitstellt. Mit Wikidata [4] arbeitet die Wikimedia Foundation derzeit selbst daran, Daten strukturiert und sogar als Linked Data anzubieten. Die dezentrale Microblogging-Plattform „identi.ca" veröffentlicht Nutzerprofile als Linked Data und Facebook experimentiert mit Open Graph ebenfalls in diese Richtung [5].

Ein großer Vorreiter in Sachen Linked Open Data ist die New York Times, die seit dem Jahr 2009 Daten über rund 10.000 Subjekte veröffentlicht und unter einer Creative-Commons-Lizenz bereitstellt [6]. Die Bibliotheksverbunde Bayern und Berlin-Brandenburg haben gemeinsam ihre riesigen Bestandskataloge als Linked Data verfügbar gemacht und damit Daten von über 23 Millionen Medien online gestellt [7]. Das Web of Data wächst, und je größer es wird, desto mehr nutzt es uns allen. Wie wir selbst Linked Data veröffentlichen und verarbeiten können, werden wir in der nächsten Ausgabe betrachten.

Chancen und Ausblick

Mit Linked Data werden Daten ein natürlicher Bestandteil des Webs. Datenbanken werden Teil des Webs und das Web wird eine riesige, offene Datenbank. Statt unterschiedliche Datenformate über proprietäre APIs abzufragen und miteinander zu vereinbaren, können wir URIs abrufen, um strukturierte Informationen zu den Dingen zu erhalten, die sie identifizieren. Wir können den dort enthaltenen Links folgen, um weitere Informationen zu erhalten, ohne dass wir ein zusätzliches Web-Service-API erlernen und implementieren müssen.

Linked Data bietet darüber hinaus große Chancen auf vielen Gebieten: Indem wir unsere Daten global miteinander verlinken, können wir Zusammenhänge entdecken, die zuvor verborgen blieben. Forschung und Wissenschaft sind nicht mehr nur in der Lage, Forschungsdokumente zu veröffentlichen, sondern auch ihre Rohdaten untereinander zu verknüpfen und so ungeahnte Beziehungen zu entdecken. Die Verbraucher profitieren von einer höheren Markt-Transparenz, wenn als Linked Data veröffentlichte Produkt-Merkmale leicht miteinander verglichen werden können. Unternehmen wird die Marktforschung erleichtert, wenn das Feedback der Verbraucher mit den Produkten und Dienstleistungen, auf die es sich bezieht, verlinkt ist.

Auf Grundlage von Linked Data kann ein „Social Semantic Web" entstehen. Ein Klick auf einen Like-Button erzeugt de facto ein RDF-Tripel mit mir als Subjekt, dem Prädikat „like" und der Sache, die ich mag, als Objekt. Letztlich könnten alle Handlungen, die wir im Web tätigen, als Linked Data abgebildet werden, sei es die Veröffentlichung eines Blog-Artikels, das Hinterlassen eines Kommentars, das Taggen eines Bilds oder das Bewerten eines Produkts. Allein durch die Nutzung des Webs können wir ein Web aus Daten erzeugen.

Dazu ist es notwendig, dass wir Linked Data in unseren Web-Projekten unterstützen. Wie das geht, werden wir in der nächsten Ausgabe zeigen. Wer sich bis dahin näher mit Linked Data befassen möchte, dem steht der Autor mit seinem Blog unter „http://datenwissen.de" zur Verfügung.

Referenzen

[1] http://www.w3.org/DesignIssues/LinkedData.html
[2] http://sparql.sindice.com/
[3] http://dbpedia.org/
[4] http://meta.wikimedia.org/wiki/Wikidata/de
[5] http://developers.facebook.com/docs/opengraph/
[6] http://data.nytimes.com/
[7] http://lod.b3kat.de/doc

Angelo Veltens
angelo.veltens@online.de
http://datenwissen.de

Angelo Veltens studierte Angewandte Informatik an der Dualen Hochschule Baden-Württemberg Karlsruhe und befasste sich in Studienarbeiten und Abschlussarbeit mit Linked Data und semantischem Wissensmanagement. Heute ist er als Software-Entwickler in Braunschweig tätig, arbeitet in seiner Freizeit am „Social Web of Data" und referiert auf Konferenzen zu diesem Thema.

www.ijug.eu

JETZT ABO BESTELLEN

Sichern Sie sich 4 Ausgaben für 18 EUR

Für Oracle-Anwender und Interessierte gibt es das Java aktuell Abonnement auch mit zusätzlich sechs Ausgaben im Jahr der Fachzeitschrift DOAG *News* und vier Ausgaben im Jahr Business News zusammen für 70 EUR. Weitere Informationen unter www.doag.org/shop/

FAXEN SIE DAS AUSGEFÜLLTE FORMULAR AN

0700 11 36 24 39

ODER BESTELLEN SIE ONLINE

go.ijug.eu/go/abo

Interessenverbund der Java User Groups e.V.
Tempelhofer Weg 64
12347 Berlin

Java aktuell

+++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN

Ja, ich bestelle das Abo Java aktuell – das IJUG-Magazin: 4 Ausgaben zu 18 EUR/Jahr

Ja, ich bestelle den kostenfreien Newsletter: Java aktuell – der iJUG-Newsletter

ANSCHRIFT

Name, Vorname

Firma

Abteilung

Straße, Hausnummer

PLZ, Ort

GGF. RECHNUNGSANSCHRIFT

Straße, Hausnummer

PLZ, Ort

E-Mail

Telefonnummer

X Die allgemeinen Geschäftsbedingungen* erkenne ich an, Datum, Unterschrift

*Allgemeine Geschäftsbedingungen:

Zum Preis von 18 Euro (inkl. MwSt.) pro Kalenderjahr erhalten Sie vier Ausgaben der Zeitschrift "Java aktuell - das iJUG-Magazin" direkt nach Erscheinen per Post zugeschickt. Die Abonnementgebühr wird jeweils im Januar für ein Jahr fällig. Sie erhalten eine entsprechende Rechung. Abonnementverträge, die während eines Jahres beginnen, werden mit 4,90 Euro (inkl. MwSt.) je volles Quartal berechnet. Das Abonnement verlängert sich automatisch um ein weiteres Jahr, wenn es nicht bis zum 31. Oktober eines Jahres schriftlich gekündigt wird. Die Wiederrufsfrist beträgt 14 Tage ab Vertragserklärung in Textform ohne Angabe von Gründen.