hbz

Wissen. Information. Innovation.

„Linked Applications“ am Beispiel von Repository-Software

Felix Ostrowski

Hochschulbibliothekszentrum des Landes Nordrhein-Westfalen

25.11.2009, SWIB09

„Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.“

Eric S. Raymond

http://www.faqs.org/docs/artu/ch01s06.html

hbz

„Another way to make things more fun is to solve the meta-problem.“

Aaron Swartz

http://www.aaronsw.com/weblog/productivity

DRY!

Gedankenanstoß: OPUS (Neu-)Entwicklung

- Hochflexibles Datenschema
- Entsprechende Eingabemasken
- Einfache Erweiterbarkeit
- Vernetzbarkeit

hbz

Klassischer Ansatz

- Grundlage: Relationale Datenbank
- Verwendung von Frameworks auf dieser Basis (Django, RoR, Zend...)
- Mapping zwischen eigenem Schema und RDF (triplify.org)
- Aber: RDF ist kein (Austausch-)Format!

hbz

„Deep Integration“

- Begriff von Denny Vrandecic
  http://www.semanticscripting.org/SFSW2005/papers/Vrandecic-Deep_Integration.pdf
  http://www.semanticscripting.org/SFSW2006/Paper1.pdf
- OWL und OOP: Klassen + Properties
- Mapping von Ontologien direkt in die verwendete Programmiersprache
- Erzwingt eine Formalisierung der Anforderungen (Description Logics)
- „Rails“ für das Semantic Web

hbz

```
<owl:Class rdf:about="#Document"/>

<owl:Class rdf:about="#Person"/>

<owl:DatatypeProperty rdf:about="#name">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#title">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="#creator">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
```

hbz

```
class Person(object):
    name = Property(str)

class Document(object):
    title = Property(str)
    creator = Property(Person)
```

hbz

```
author = Person()
author.name = 'Max Mustermann'

document = Document()
document.title = 'Ein Dokument'
document.creator = author
```

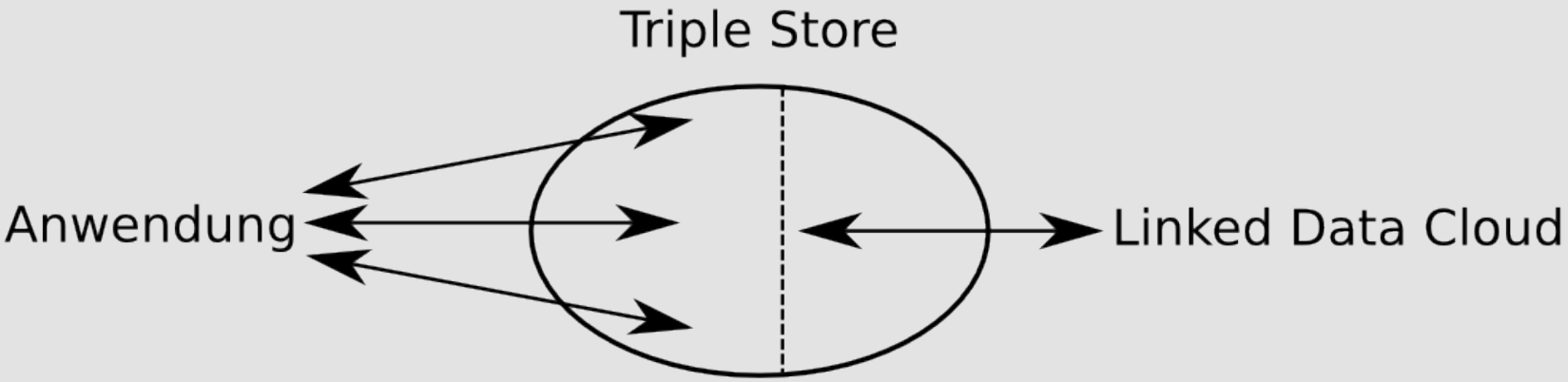
```
<rdf:Description rdf:about="http://example.org/auth/1">
    <rdf:type rdf:resource="#Person"/>
    <name>Max Mustermann</name>
</rdf:Description>

<rdf:Description rdf:about="http://example.org/doc/1">
    <rdf:type rdf:resource="#Document"/>
    <creator rdf:resource="http://example.org/auth/1"/>
    <title>Ein Dokument</title>
</rdf:Description>
```

hbz

Triple Store Interface

- Mapping der Objekte in die DB trivial
- Schema-Änderungen auf Datenbankebene unnötig
- Ggf. Trennung in einen öffentlichen und einen privaten Store (Subgraphen?)

hbz



- Generisches add / remove / query interface
- Grenze der „Closed World Assumption“

  http://bel-epa.com/rdflibdocs/univrdfstore.html
- "link globally, cache locally."

  http://twitter.com/anarchivist/status/3844049229

hbz

Vorteile

- Externe Definition des Datenmodells
- Einfache Integration externer RDF-Daten (unterscheiden sich prinzipiell nicht von den eigenen Daten)
- Einfach zu erweitern (insbesondere bei Verwendung eines Triple Stores)
- SPARQL API

hbz

Probleme

- Abbildung von Namensräumen
- Definition typisierter Literale in OWL
- Kardinalitäten
- Erweiterungen einfach, aber Einschränkungen evtl. schwierig
- Programmiersprachen unterschiedlich gut geeignet

hbz

Anwendung in Repositorien

- Definition von Dokumenttypen als OWL-Ontologie
- Generell kann jede Sicht auf die Daten als Transformation einer RDF-Serialisierung implementiert werden (RDF/XML + XSLT)
- Einbindung beliebiger kontrollierter Vokabulare (SKOS)
- Erweiterung der Beschreibung bestehender Ressourcen

hbz

Vielen Dank!

Fragen? Jetzt oder gerne auch an

ostrowski@hbz-nrw.de