Chapter 19
On Entities in the Web of Data

Michael Hausenblas

Abstract This chapter aims to explore what "entities" in the Web of Data are. As a point of departure we examine a number of widely used RESTful Web APIs in terms of URI space design and hyperlinking support in the offered resource representations. Based on the insights gained from the API review, we motivate the concept of an entity as well as its boundaries. Eventually we discuss the relevance of the entity concept for publishers and consumers of Web data, as well as the impact on Web data design issues.

The Web of Data

In this chapter, we will certainly not be able to resolve a decade-old, well-known issue from the programming domain (Post 1982):

> Nicklaus Wirth, the designer of PASCAL, was once asked, "How do you pronounce your name?". He replied "You can either call me by name, pronouncing it *Veert*, or call me by value, *Worth*." One can tell immediately from this comment that Nicklaus Wirth is a Quiche Eater.

However, as this chapter unfolds, we will discover that a somewhat similar problem exists concerning Web data. When we say *Web data* in the context of this chapter, we mean data as found in Web APIs and dataset collections, such as governmental statistics, eCommerce product data, or media content (such as found in Flickr), with a particular interest in widely deployed formats, including Atom, JSON, but also serialisations of the Resource Description Framework (RDF) (Klyne et al. 2004). Eventually, we are particularly interested in a characteristic of Web data that tells it apart from centralised data sources (such as relational databases): *links between data items*.

M. Hausenblas (✉)
DERI, National University of Ireland Galway, IDA Business Park, Galway, Ireland
e-mail: michael.hausenblas@deri.org

In contrast to Chap. 5, “Hypermedia Types” (by Mike Amundsen) we do not focus on the internals of the used media types or the hyperlink semantics. Also, we do not aim to address the metadata issues as discussed in Chap. 23. “Metadata Architecture in RESTful Design” (by Antonio Garrote and Maria N. Moreno Garcia), however both chapters can be seen as complimentary to the one at hand.

We will start off this chapter by reviewing existing RESTful APIs in “Reviewing RESTful APIs”, and then discuss design considerations regarding the *URI space* (URI Space Design), *representations* (On Representations and Entity Boundaries) and *hyperlinking* support (Hyperlinking) concerning entities. “Limitations and Future Work” reports on the limitations of the work presented and eventually, in “Conclusion” we conclude this chapter.

Reviewing RESTful APIs

Methodology

In the following, we will have a closer look at widely used and popular services that claim to offer RESTful APIs. The goal is to gain a deeper insight into the actual deployment status of resource granularity and the degree of hyperlinking support. In order to assess the before-mentioned aspects, we will examine the following characteristics of each API:

1. *URI space design*. In this respect, we are interested in how the API’s URI space is partitioned: we analyse if all important resources have URIs, how the URI space is organised and how cool the URIs are.[1] For example, they URI space may be organised in a flat manner or hierarchically.
2. *Representations*. Concerning the resource representations offered by the API, we ask if registered media types, such as Atom (Nottingham and Sayre 2005) are used vs. custom formats. Further, we investigate if alternative representations are offered via content negotiation or comparable mechanisms (Raman 2006).
3. *Hyperlinking*. Regarding this aspect we examine if and how hyperlinks are used. Based on the findings in the previous category we analyse the utilisation of hyperlinks in the representations served by the API. We ask especially to which extend they support the discovery of related data items within the site and potentially outside the API.

The selection of the APIs in the following is based on popularity[2] and experiences the author has gained in projects as well as from the interaction with the REST

[1] http://http://www.w3.org/Provider/Style/URI.

[2] http://http://www.programmableweb.com/apis/directory/1?protocol=REST.

community. We are well aware of the fact that the review is neither exhaustive nor definitive, nonetheless it offers an representative insight what is currently available in terms of RESTful APIs on the Web.

Each of the following sections[3] starts out with a table summarising the API characteristics regarding the above aspects and further lists more detailed observations per API. For the summary table we will use *fine-grained* if the URI space exposes all relevant resources and *coarse-grained* if only few resources are exposed. Note that our usage of fine-grained vs. coarse-grained differs from the usage typically found in the literature, where fine-grained refers to object-oriented style and coarse-grained means document-style interactions with fewer, but more structured documents.

Basecamp

Basecamp is a popular Web-based project management tool, mainly dealing with people, notes, to-do items, shared documents, milestones and time spent on activities. According to the documentation, the API[4] is "vanilla XML over HTTP using all four verbs (GET/POST/PUT/DELETE)".

URI space design	Representations	Hyperlinking
Fine-grained	Custom XML, HTML	No

1. *URI space design*. The API exposes the relevant resources of the domain in a fine-granular, hackable manner. For example, each to-do list in a project has a URI.[5]
2. *Representations*. A proprietary XML format is used as the main representation; content negotiation is supported.
3. *Hyperlinking*. There seems to be no explicit usage (or support) of hyperlinking in the XML representations.

Delicious

Delicious is a social bookmarking site, mainly dealing with people, bookmarks, and tags, with a documented API.[6]

[3]The API reviews are presented in alphabetical order.

[4]http://http://developer.37signals.com/basecamp/.

[5]http://https://lidrc.basecamphq.com/projects/4284964/todo_lists.

[6]http://http://www.delicious.com/help/api.

URI space design	Representations	Hyperlinking
Coarse-grained	Custom XML	No

1. *URI space design*. Only few of the main resources the API deals with are in fact exposed, such as posts.[7]
2. *Representations*. The API offers a custom XML without content negotiation. Interestingly, JSON format is offered for certain types of information via a separate, so called "Feed API".[8]
3. *Hyperlinking*. Although the representations contain hyperlinks (for example, in the shape of `<post href='http://example.org'>...</post>`), the data items such as people and their bookmarks are not linked.

Facebook

Facebook is a social network platform, mainly dealing with people, groups, events, messages, applications, and shared media content (images, videos, etc.) with a JSON-centric API.[9]

URI space design	Representations	Hyperlinking
Fine-grained	JSON	No

1. *URI space design*. All major resources are exposed via human-readable URIs, like people[10] or events.[11]
2. *Representations*. The main representation used in the API is JSON (no content negotiation offered) with an extension mechanism via the OpenGraph protocol[12] for integrating external content into the Facebook platform.
3. *Hyperlinking*. Hyperlinks are used in the representations, however, only for stating values such as `"link": "http://www.facebook.com/mhausenblas"` and not to relate the data items within the platform.

[7]http://https://user:passwd@api.del.icio.us/v1/posts/get.

[8]http://http://www.delicious.com/help/feeds.

[9]http://http://developers.facebook.com/docs/api.

[10]http://http://graph.facebook.com/mhausenblas/

[11]http://http://graph.facebook.com/331218348435/.

[12]http://http://developers.facebook.com/docs/opengraph

Flickr

Flickr is an image and video hosting website, mainly dealing with shared media content, people, groups, and tags with a so-called "REST API".[13]

URI space design	Representations	Hyperlinking
Fine-grained	custom XML, JSON	No

1. *URI space design*. The API exposes the main resources, such as photos[14] via distinct URIs, however the method names are explicitly encoded as URI parameters (like `?method=flickr.photos.getInfo`), which is considered a bad practice in the REST community.
2. *Representations*. Both proprietary XML and JSON are offered, albeit not via content negotiation but via a parameter (`format=json`).
3. *Hyperlinking*. One can find usages of typed hyperlinks in the served representations (such as `<url type="photopage">...</url>`, linking a photo to its page), however, in the general case the data items are connected via literal values (for example `<owner nsid="7278720@N02"... />`).

FriendFeed

FriendFeed is an aggregator service, consolidating updates from social (media) platforms, blogs, as well as news feeds with an API[15] that defaults to JSON. It mainly deals with feeds, comments, people, groups and notifications.

URI space design	Representations	Hyperlinking
Fine-grained	JSON, custom XML	No

1. *URI space design*. All important resources in the API have URIs (for example, a person's feed[16]) and follow a logical structure.
2. *Representations*. The default representation the API offers is JSON, and custom XML can be obtained (`format=json`), however not via content negotiation.
3. *Hyperlinking*. Hyperlinks are used in the provided representations, for example to represent the provenance of an entry (a link to a microblog post: `"url": "..."`), however, not to provide navigation between data items in the representations.

[13]http://http://www.flickr.com/services/api/request.rest.html.

[14]http://http://api.flickr.com/services/rest/?method=flickr.photos.getInfo&photo_id=4745449672.

[15]http://http://friendfeed.com/api/documentation.

[16]http://http://friendfeed-api.com/v2/feed/mhausenblas.

GeoNames

GeoNames is a geographical database accessible through numerous APIs,[17] mainly dealing with places, regions, weather, addresses, and geo-coordinates.

URI space design	Representations	Hyperlinking
Fine-grained	custom XML, JSON, others	No

1. *URI space design*. All main resource have dedicated URIs, such as a certain region.[18]
2. *Representations*. The API offers a wide range of representations, including the two most widely supported XML and JSON (via dedicated URIs, no content negotiation), but also other formats, such as CSV, RDF/XML, KML and RSS.
3. *Hyperlinking*. In few places one is able to spot (potentially) typed links, such as `"wikipedia":"de.wikipedia.org/wiki/Mexiko-Stadt"`, however, the relations between the resources is mainly established via literal values one can look-up accross the offered APIs.

Google Maps

The Google Maps API is really a family of APIs providing geographic data for maps applications. In the following we will focus on the Google Geocoding API[19] that converts addresses into geographic coordinates and mainly deals with addresses and geo-locations.

URI space design	Representations	Hyperlinking
Fine-grained	JSON, custom XML	No

1. *URI space design*. Each resource of interest exposed via the API has its own URI (for example, when looking up the address for a certain building[20]).
2. *Representations*. Both JSON and proprietary XML is served from respective URIs (no content negotiation).
3. *Hyperlinking*. There is no indication for hyperlinking usage in the representations.

[17]http://http://www.geonames.org/export/web-services.html.

[18]http://http://ws.geonames.org/findNearbyPlaceName?lat=53.27&lng=-9.04.

[19]http://http://code.google.com/apis/maps/documentation/geocoding/.

[20]http://http://maps.googleapis.com/maps/api/geocode/json?address=IDA+Business+Park+Galway.

Netflix

Netflix is a company that offers DVD rental and on-demand video streaming. The API deals with movies, actors, awards and the like. The Netflix OData API[21] is a representative example for Microsoft's Open Data Protocol (OData) protocol,[22] a query and access protocol building upon Atom and AtomPub.

URI space design	Representations	Hyperlinking
Fine-grained	Atom/XML, JSON	Yes

1. *URI space design*. The API exposes the main resources via distinct URIs, a certain actor,[23] for example.
2. *Representations*. Per default, the API serves Atom (Nottingham and Sayre 2005), however also offers content negotiation. For example, `curl -H "Accept: application/json"` http://odata.netflix.com/v1/Catalog/ yields the JSON representation of the catalog.
3. *Hyperlinking*. The representations partially contain typed hyperlinks as specified by the Atom standard, for example relating an actor to an award as in: `<link rel="..." href="People(189)/Awards" />`. The relations themselves[24] are not dereferencable.

Twitter

Twitter is a microblogging service allowing users to send and read other users' messages (up to 140 characters) dealing mainly with said messages, people, and geo-coordinates. The API[25] comes in a REST flavour[26] and in a so-called stream version.

URI space design	Representations	Hyperlinking
Fine-grained	custom XML, JSON, RSS, Atom	No

[21]http://http://odata.netflix.com/v1/Catalog/.

[22]http://http://www.odata.org/.

[23]http://http://odata.netflix.com/v1/Catalog/People(189).

[24]http://http://schemas.microsoft.com/ado/2007/08/dataservices/related/Awards.

[25]http://http://dev.twitter.com/doc.

[26]http://http://api.twitter.com/1/.

1. *URI space design*. All main resources are exposed via URIs (such as a certain user profile [27]) organised in a flat space.
2. *Representations*. The XML and JSON representations dominate the API and are served via dedicated URIs (`/show.json?`, for example), where for some resources (like the public timeline) also RSS and Atom formats are provided.
3. *Hyperlinking*. Although one can find hyperlinks in the representations, there is no evidence for utilising typed links to relate resources within the API.

Discussion

In terms of *URI space design*, the majority (88%) of the reviewed APIs do a great job in exposing the respective main resources in a fine-granular manner. Very often the URIs can considered to be hackable, which means a developer can easily follow a pattern in constructing them. We note that for our discussion it is of no importance if we consider the URIs opaque[28] or not; in fact hackable URIs often lead to strong coupling as the URI patterns are hard-coded for convenience reasons.

The outcome regarding the *representations* is somewhat inconclusive: some 77% serve proprietary XML, only two support established standards such as Atom, however most offer developer-friendly JSON (which seems to be sufficient for the key-value structure of most responses).

Only a single API out of nine in fact supports true *hyperlinking* in its representations. Although the URIs for the resources have typically been made available (see above), the majority of the APIs seem to ignore the potential benefits in referencing them.

URI Space Design

One of the most important – though often underrated – aspects of RESTful design is how to name the things one wishes to talk about. In more technical terms one may think of URI space design. Richardson and Ruby (2007) have documented valuable good practices regarding the URI space design, but one can also obtain helpful hints from the REST Wiki,[29] where this topic is maintained under the “Noun” label.

Whereas RESTful design in general requires to identify those things we would like to interact with, the following discussion operates under the presumption that we want to establish a fine-grained access to data items in the Web of Data.

[27]http://http://api.twitter.com/1/users/show.json?user_id=817540.

[28]http://http://rest.blueoxen.net/cgi-bin/wiki.pl?OpacityMythsDebunked.

[29]http://http://rest.blueoxen.net/cgi-bin/wiki.pl?FrontPage#nid5TL.

Naming Things

Why is it essential to name things, that is, to assign URIs to all important resources one exposes on the Web? Using a small motivation example may help shed some light on this matter. Imagine a fictitious company that wants to inform about their projects and how people are involved in it. One would expect to find, for example, the following resources: *people*, *projects*, *technologies*, *products* and respective URIs, such as:[30]

- For *collection resources*, such as all projects the company maintains, the URI might be http://company.example.com/project
- *Item resources*, for example a particular project of the company, might be identified by http://company.example.com/project/pr1 and a certain person by http://company.example.com/people/roy.est.

Now, assuming one has the URI handy, one can use the URI in an application (to obtain data from it) or link to the URI from another Web site. Obviously, if such URIs are not available, one can not achieve the above things directly, and even more seriously: the network effect is crippled.

Naming things one is dealing with on the Web, that is, minting URIs for all main resources one exposes is essential for RESTful design. Non-observance cripples the network effect.

URI Fragments for Sub-resources

A special sub-topic of URI space design worthy attention is how to deal with URI fragments[31] to identify sub-resources. Using URI fragments allows to link to things, but they do not allow for interaction with said things through the uniform interface.

Concerning the URI fragment identifiers semantics, we need to consult the *Uniform Resource Identifier (URI): Generic Syntax* (RFC3986) (Berners-Lee et al. 2005):

> The semantics of a fragment identifier are defined by the set of representations that might result from a retrieval action on the primary resource. The fragment's format and resolution is therefore dependent on the media type of a potentially retrieved representation,

[30]Note that we use the terminology for the types of resources (collection and item resources) suggested by Glenn Block in a recent blog post available via http://bit.ly/rest-resource-types.

[31]http://http://www.w3.org/DesignIssues/Fragment.html.

> even though such a retrieval is only performed if the URI is dereferenced. If no such representation exists, then the semantics of the fragment are considered unknown and are effectively unconstrained. Fragment identifier semantics are independent of the URI scheme and thus cannot be redefined by scheme specifications.

We note that most media types[32] do not to specify URI fragments,[33] and where this is the case, the *Architecture of the World Wide Web, Volume One* (AWWW) (Jacobs and Walsh 2004) gives us some guidance; cf. Sects. 3.2.1 (Representation types and fragment identifier semantics) and 3.2.2 (Fragment identifiers and content negotiation) of the AWWW, especially concerning the conflict resolution mechanism for content negotiation.

Summarising, URI fragments allow to identify sub-resources in a straight-forward way. One should ensure that the fragments are made identifiable, for example in the case of HTML this would require support by document authors.[34]

However, there is a number of unresolved issues around their usage, subject to research – for example, concerning the interactions with these sub-resources[35] – and standardisation, for example regarding HTTP redirects.[36]

On Representations and Entity Boundaries

Literal-style vs. Reference-style

To understand entities in the context of the Web of Data, we will first discuss how entities can be represented, following the AWWW (Jacobs and Walsh 2004). Note that in the following, we will deliberately use the terms *representation* (Fielding et al. 1999) and *document* synonymously; later on we will go into detail regarding the notion of a document.

A fundamental characteristic of Web data is the ability to utilise "hyperlinks", essentially a URI reference between resources (Jacobs and Walsh 2004) that typically comes with some link semantics attached. To assess the type and extent of the supported link semantics concerning media types, we refer the reader to Amundsen's work on *Hypermedia Types* (Amundsen 2010).

When consuming Web data one typically has to deal with the extraction of the data structure and its values from a representation. Furthermore, data values might be provided as literal values or, through a hyperlink, as a reference to another resource. In the following we will discuss these two options (literal-style vs. reference-style) in greater detail; note, however, that this does not mean that there can not or may not exist other design options at all.

[32]http://http://www.iana.org/assignments/media-types/.

[33]With a few exceptions, such as HTML and RDF/XML.

[34]Essentially meaning that relevant elements in HTML need to be supplied with and `id` attribute.

[35]http://http://oreillynet.com/xml/blog/2008/02/addressing_fragments_in_rest_1.html.

[36]http://http://lists.w3.org/Archives/Public/www-tag/2010Oct/0003.html.

```
<div>
  <div>Name: Michael Hausenblas </div>
  <div>Residence: 32 Bushypark Lawn, Galway, Irland </div>
</div>
```

Fig. 19.1 Entity represented in HTML

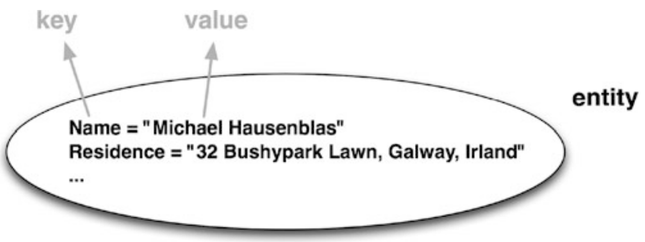
key
value
entity
Name = "Michael Hausenblas"
Residence = "32 Bushypark Lawn, Galway, Irland"
...

Fig. 19.2 Entity's key-value structure

```
<div>
 <div>Name: Michael Hausenblas </div>
 <div>Residence: <a href=”address.html”>my address </a></div>
</div>
```

Fig. 19.3 Entity represented in HTML

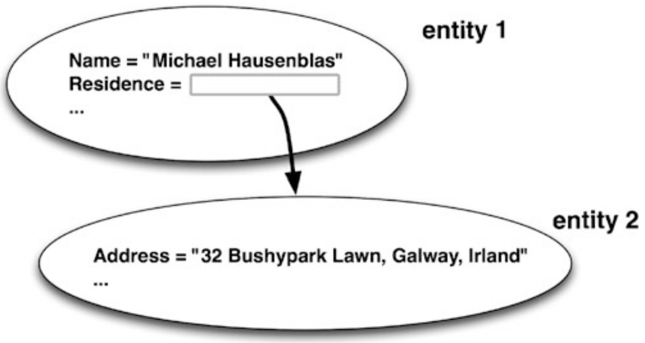
entity 1
Name = "Michael Hausenblas"
Residence =
...
entity 2
Address = "32 Bushypark Lawn, Galway, Irland"
...

Fig. 19.4 Entity's key-value structure

Providing data values via references allows for a greater flexibility compared to literal values and enables the reuse of data within a site and across the Web. However, it also comes with its costs: each reference needs to be resolved and the referred document parsed, yielding additional costs in the data processing.

Consider the following two cases: Figs. 19.1 and 19.2 show an entity with pure literal-style values, whereas Figs. 19.3 and 19.4 depict the case where partial reference-style values are used.

Although one is, in terms of hyperlinking capabilities, restricted by the choice of the representation, we note that using literal-style vs. reference-style is first and foremost a design decision of the resource owner (Jacobs and Walsh 2004). At the end of the day, one has to deal with the trade-off between processing speed (literal-style) vs. flexibility and reusability (reference-style).

Literal-style vs. reference-style data values are design decision of the resource owner, rather than characteristics of representations.

Entity Boundaries

The notion of a document has been the topic of recent discussions (cf. for example Wilde 2009). In the following, we will explore if the notion of a document is helpful in the context of the Web of Data and how this relates to entities.

To approach the issue of a "document notion", let us first step back a bit and discuss what are *directly observable things* on the Web. With directly observable we mean that something can be measured, processed, stored, etc. in the widest sense.

In a first step we want to determine what directly observable things on the Web are. As a starting point, we will use the Web's Retrieval Algorithm as described in Mendelsohn (2009): dereferencing a URI yields a representation of the resource identified by the URI. We note that *URIs* and *representations* are directly observable things, while *resources* are not directly observable. For example, a URI can be bookmarked or the representation of a resource can be stored in a file. Resources, on the other hand are purely conceptional and only are observable indirectly through URIs and the resource representation at a given point in time.

Further, we acknowledge the fact that the hyperlink structure of the Web is crucial for content discovery (Raman 2006). We note that although the discovery is enabled by hyperlinks between resources, the actual communication necessarily needs to be carried out using representations. In this context a special subset of resources is of interest: *information resource*. We will use the definition of information resource as of Jacobs and Walsh (2004), repeated here for convenience in Definition 1.

Definition 1. If all of the essential characteristics of a resource can be conveyed in a message, the resource is an information resource.

For certain applications and use cases, the concept of an information resources is essential, especially when dealing with metadata. Take for example the resource "the current temperature in Galway, Ireland". This resource is identified by the URI:

```
http://example.com/galway/temperature
```

```
<forecast city="Galway">
 <temperature>15</temperature>
</forecast>
```

Fig. 19.5 XML representation from `http://example.com/galway/temperature.xml`

```
@prefix m: <http://purl.org/ns/meteo#> .
@prefix d: <http://dbpedia.org/resource/> .
@prefix : <> .

d:Galway m:forecast :tempForecast .
:tempForecast m:temperature :tempVal .
:tempVal m:celsius "15" .
```

Fig. 19.6 RDF representation from `http://example.com/galway/temperature.ttl`

Further, assume there are two accompanying information resources:

```
http://example.com/galway/temperature.xml
http://example.com/galway/temperature.ttl
```

Having these two information resources available, one is able to state things like "the current temperature is provided to you by company X", where it is clear that not the temperature itself, but the measurement, the data point has been made available by a certain company.

Coming back to the "notion of a document", we now have a look at the representations (Figs. 19.5 and 19.6) retrieved from the two information resources. We understand that both convey the same information. Further, once processed by a consumer (in-memory, loaded into a relational database, etc.) one is unable to tell from which representation it originated. We, hence, claim that the notion of a document – as perceived in the XML representation – in fact has no impact on the consumer.

In fact, if one treats the resource URI and the representation together as a unit, the notion of a document as such is not helpful regarding the Web of Data. We acknowledge the fact that the above example can not be applied in a straight-forward manner to the case where the interaction with the information resource goes beyond a read-only operation (a HTTP `POST` or `PUT`, for example).

For the concept of an entity these observations are insofar essential, as the entity boundaries should not be understood in terms of document boundaries, but in terms of URIs, which potentially occur in resource representations.

Hyperlinking

Equipped with the reference-style design pattern and the idea of treating a resource URI together with its representation as a unit for processing data, we are now ready to approach the definition of an entity. While the term "entity" itself has already

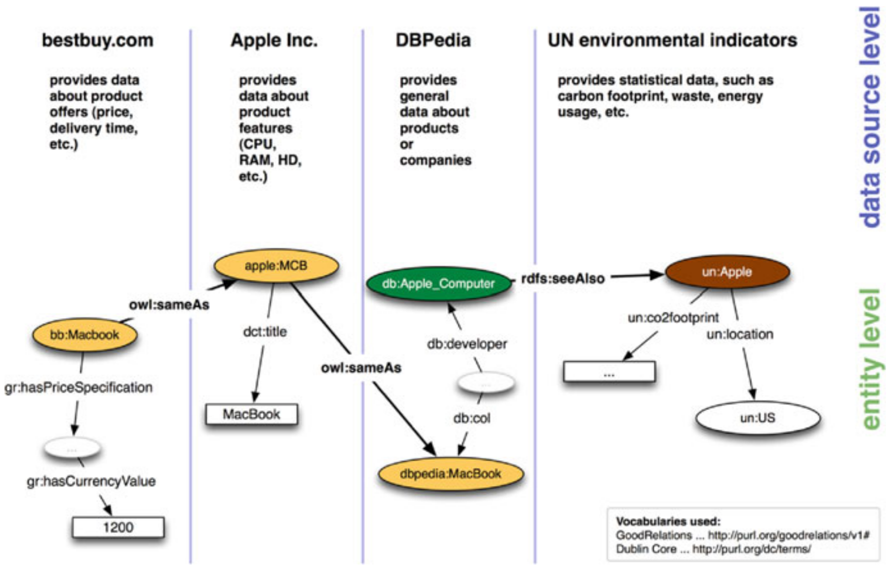
bestbuy.com
provides data about product offers (price, delivery time, etc.)
Apple Inc.
provides data about product features (CPU, RAM, HD, etc.)
DBPedia
provides general data about products or companies
UN environmental indicators
provides statistical data, such as carbon footprint, waste, energy usage, etc.
data source level
entity level
bb:Macbook
gr:hasPriceSpecification
gr:hasCurrencyValue
1200
owl:sameAs
apple:MCB
dct:title
MacBook
owl:sameAs
db:Apple_Computer
db:developer
db:col
dbpedia:MacBook
rdfs:seeAlso
un:Apple
un:co2footprint
un:location
...
un:US
Vocabularies used:
GoodRelations ... http://purl.org/goodrelations/v1#
Dublin Core ... http://purl.org/dc/terms/

Fig. 19.7 Entity example from the Linked Open Data realm

been in use for a while (Bouquet et al. 2008; Umbrich et al. 2010), to the best of our knowledge no agreed definition is available. We hence attempt to define an entity in the following (cf. Definition 2), and note the usage of "connector" as of Fielding and Taylor (2002).

Definition 2. An entity is a thematic view on resources across connectors, materialised through hyperlinks.

The two important bits in Definition 2 are "across connectors" and "hyperlinks"; the former acknowledging the fact that the actual data belonging to an entity is potentially distributed over several data sources and the latter that, if the data items are not explicitly connected, it is hard to impossible to construct an entity. One can even go a step further and assert that entities as such make only sense when the reference-style design is employed, as otherwise out-of-bound information is necessary to consume related information in the Web. Note also that we propose not to restrict what "thematic" might mean, as this is very likely depending on the application that processes an entity.

Take, for example, Fig. 19.7: different Linked Open Data sources[37] – homogenous linked data as of Wilde (2010) – may expose different aspects, such as price, technical features, carbon footprint, etc. regarding products through respective resources. Assume now, one is interested to buy a certain laptop with a particular price limit and carbon footprint. In this application, the entity of interest is "a laptop", and taking the interlinked data items from the four data sources together, one is able to answer the query.

[37]http://http://lod-cloud.net/.

While it seems that from the perspective of a consumer (who has to typically deal with several data sources), the concept of an entity is pretty straight-forward, in case of a data publisher it may not be so obvious. For the Web data publisher, resources and resource identifiers are the primary design elements. Not only are they (along with the choice of appropriate representations) the main building blocks, but are typically considered sufficient in terms of organisation. Regarding the data publisher side of Definition 2, with *connector* we mean in special a *server*, which is assumed to be authoritative for a resource. However, one can also understand the data publisher playing the role of a consumer regarding other data sources (Volz et al. 2009).

Limitations and Future Work

The main limitation of the research presented in this paper lies in the fact that it focuses on the read-only case (HTTP `GET`). A consistent and comprehensive discussion of the "transactional view" is subject of future research, taking into account if and how *update*, *add*, or *remove* would work without having a point of reference, that is, a container, such as a document provides.

Further, we note that it has yet to be discussed how the concept of an entity fits into proposed extensions of the REST style, such as *Computational REST* (CREST) (Erenkrantz 2009), where computational exchange is the primary exchange mechanism between peers, hence relaxing the client-server distinction found in REST.

Conclusion

The transition from document-centric processing to entity-centric processing in the Web of Data is taking place. In this chapter we have first reviewed deployed RESTful APIs in terms of their *URI space design*, concerning the served *representations* and the support of *hyperlinking* in the representations (or the lack thereof).

Based on the insights gained in the API review we have discussed challenges and pitfalls concerning the design of RESTful APIs from an entity-centric point of view, which finally leads us to the importance of the concept of an entity in the context of the Web of Data. The main idea of an entity is that it takes into account the hyperlinking aspect between Web data items and hence provides a model that goes beyond a (single) resource (from a single datasource).

Acknowledgments The author would like to thank Richard Cyganiak for the conversations regarding the notion of document, Mike Amundsen for deeper insights into hypermedia linking aspects, as well as Jonathan Rees and David Booth – in the context of the W3C task force

"Architecture of the World Wide Semantic Web" – for ongoing discussions around HTTP semantics. Last but not least, the author wants to express his gratitude to Erik Wilde for his feedback and his continuing support to ensure that the chapter focuses on under-represented topics in the REST research.

References

M. Amundsen. Hypermedia Types, 2010.

T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. Request for Comments: 3986, January 2005, IETF Network Working Group, 2005.

P. Bouquet, H. Stoermer, D. Cordioli, and G. Tummarello. An Entity Name System for Linking Semantic Web Data. In *WWW 2008 Workshop: Linked Data on the Web (LDOW2008)*, Beijing, China, 2008.

J. Erenkrantz. *Computational REST: A New Model for Decentralized, Internet-Scale Applications*. PhD thesis, University of California, Irvine, 2009.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Request for Comments: 2616, June 1999, IETF Network Working Group, 1999.

R. Fielding and R. Taylor. Principled design of the modern Web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002.

I. Jacobs and N. Walsh. Architecture of the World Wide Web, Volume One. W3C Recommendation 15 Dec 2004, Technical Architecture Group, 2004.

G. Klyne, J. J. Carroll, and B. McBride. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004, RDF Core Working Group, 2004.

N. Mendelsohn. The Self-Describing Web. W3C TAG Finding 7 Feb 2009, Technical Architecture Group, 2009.

M. Nottingham and R. Sayre. The Atom Syndication Format. Request for Comments: 4287, December 2005, IETF Network Working Group, 2005.

E. Post. Real Programmers Don't Use PASCAL, 1982.

T. V. Raman. On Linking Alternative Representations To Enable Discovery And Publishing. W3C TAG Finding 1 November 2006, Technical Architecture Group, 2006.

L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly Media, Inc., 2007.

J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres, and S. Decker. Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources. In *WWW 2010 Workshop: Linked Data on the Web (LDOW2010)*, Raleigh, USA, 2010.

J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and Maintaining Links on the Web of Data. In *ISWC '09: Proceedings of the 8th International Semantic Web Conference*, pages 650–665, Berlin, Heidelberg, 2009. Springer-Verlag.

E. Wilde. REST and RDF Granularity, 2009.

E. Wilde. Linked Data and Service Orientation. In M. Weske, J. Yang, P. Maglio, and M. Fantinato, editors, *8th International Conference on Service Oriented Computing (ICSOC 2010)*, Lecture Notes in Computer Science, page NN, San Francisco, California, 2010. Springer-Verlag.