Thomas Koch
thomas@koch.ro
matriculation number 7250371

December 8, 2011
Fernuniversität Hagen
Faculty of mathematics and computer science

Contents

1. Motivation

Why vCard/CardDav: many clients

Why OpenSocial / Portable Contacts:

- used by Google, LinkedIn,
- used in Enterprise applications like Attlassian tools (Jira, Confluence, ...), Nuxeo CMS, ...
- OpenSocial can be used to implement inhouse portals and populate it with data from the companies GroupWare

2. Advantages of REST for Groupware data

The REST style is motivated by the argument that its application would help to provide certain attributes for an architecture.[Fie00, sec 5.1]

Maybe not so important for a companies internal Groupware: Scalability (in terms of users), Network performance, Efficiency,

- Cachability can keep the data available also in offline mode
- Simplicity helps to develop glue code between different systems
- Modifiability allows to adapt the Groupware to changes in the organization
- Reliability
- Administrativ scalability

3. Evaluation of APIs

3.0.1. Algermissen's Classification of HTTP-based APIs

Jan Algermissen, proposes a "Classification of HTTP-based APIs" in February 2010.[1] He identifies and names a five level order of HTTP based APIs. Each level adds one constraint to be obeyed. The last level obeys all five constraints and is RESTful. (Table 1)

level name	constraint violated	violation description
WS-*	Identification of Resources	Only service endpoint is identified by URI. No resources exposed.
RPC URI-Tunneling	Manipulation of Resources through Representations	SOAP body contains operation name, message not transferred to manipulate resource state.
HTTP-based Type I	Self-Descriptive Messages	Message semantics depend on action specified in message body.
HTTP-based Type II	Hypermedia as the Engine of Application State	Application state machine known at design time.

Table 1: Classification of HTTP-based APIs after Algermissen

[1] `http://nordsc.com/ext/classification_of_http_based_apis.html` (2011-12-08)

In addition to that, Algermissen also provides a list of "Effect on System Properties and Cost" of the different API styles and acknowledges that the initial costs of developing a REST API may be higher compared to the other styles but lower in the long run.

4. Existing Groupware data APIs

4.1. IMAP (Kolab)

Kolab uses an IMAP server as the data store and synchronization protocol for calendar and contact informations. I want to compare this approach to a restful one.

Advantages of IMAP:

- already there, since Mail uses it
- can store blobs/files so no need to map the iCal/vCard files to a relational scheme
- out of the box support for offline work and later synchronization (How does it solve editing conflicts?)

Disadvantages of IMAP:

- Complicate, 38 RFCs according to `http://de.wikipedia.org/wiki/Internet_Message_Access_Protocol` see also: `http://www.apps.ietf.org/rfc/ipoplist.html`
- All clients directly access the iCal/vCard files with no moderation layer in between. This means that no validation or normalization can be done. Schema updates can only be done if all clients cooperate.
- IMAP imposes a folder structure. Google's gmail is an example for another, tag based approach. Messages could have several tags. It is therefor hard to access Gmail via IMAP.
- Sam Varshavchik, the author of the courier Mail Transfer Agent explains the history of IMAP and claims that the IMAP standard is broken: http://www.courier-mta.org/fud/
- IMAP is so complicate that the IMAP wiki holds 10 pages of advises for IMAP client authors: http://www.imapwiki.org/ClientImplementation RFC 2683 "IMAP4 Implementation Recommendations" is a 23 pages document (cut 5 pages for verbosity) explaining how to implement another RFC standard. Is there any widely used standard that needs another RFC explaining how to implement it?
- `http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol#Disadvantages`
- Some attempts to create a simpler alternative to IMAP:
  - http://en.wikipedia.org/wiki/POP4
  - `http://en.wikipedia.org/wiki/Simple_Mail_Access_Protocol` also here http://www.co mta.org/cone/smap1.html
  - `http://en.wikipedia.org/wiki/Internet_Mail_2000`
  - HTTP restful: http://tools.ietf.org/id/draft-dusseault-httpmail-00.txt mailing list: https://www.ietf.org/mailman/listinfo/httpmail
  - BikINI is not IMAP http://bikini.caterva.org
  - Outlook uses HTTP to communicate with Hotmail

  - another rest mail proposal: http://www.prescod.net/rest/restmail/

- more rants: http://blog.gaborcselle.com/2010/02/how-to-replace-imap.html
- IMAP issues found by the chandler project http://chandlerproject.org/bin/view/Jungle/IntrinsicI

4.2. WebDAV, CalDAV, CardDAV

4.2.1. Is WebDAV restful?

Roy Fielding says no: `http://tech.groups.yahoo.com/group/rest-discuss/message/5874`

> PROP* methods conflict with REST because they prevent important resources from having URIs and effectively double the number of methods for no good reason. Both Henrik and I argued against those methods at the time. It really doesn't matter how uniform they are because they break other aspects of the overall model, leading to further complications in versioning (WebDAV versioning is hopelessly complicated), access control (WebDAV ACLs are completely wrong for HTTP), and just about every other extension to WebDAV that has been proposed.
>
> [...]
>
> The problem with MOVE is that it is actually an operation on two independent namespaces (the source collection and destination collection). The user must have permission to remove from the source collection and add to the destination collection, which can be a bit of a problem if they are in different authentication realms. COPY has a similar problem, but at least in that case only one namespace is modified. I don't think either of them map very well to HTTP.

The discussion also continued on the microformats mailing list `http://microformats.org/discuss/mail/microformats-rest/2006-April/thread.html#217`.

see [Amu10] for a restful approach to properties.

Is ATOM an alternative to WebDAV?

> AtomPub is different from DAV in two key respects:
>
> - The client doesn't control where things go, the server does
> - It is allowed and expected that an AtomPub server will look at the incoming information and change it (generate ID, timestamps, sanitize HTML, etc)

Tim Bray, http://www.imc.org/atom-protocol/mail-archive/msg11271.html

4.3. PortableContacts aka OpenSocial

4.3.1. Is OpenSocial restful?

Roy Fielding wrote a blog post about the "SocialSite REST API", stating that it isn't restful at all but clearly an RPC style API.[2] Fielding was referring to SocialSite, which is however an implementation of the OpenSocial specification. Dave Johnson, a contributor to SocialSite, reacted on this critique by opening a discussion on an OpenSocial mailing list:[3]

[2] `http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hyperhtext-driven` (2011-12-06)

[3] `http://groups.google.com/group/opensocial-and-gadgets-spec/browse_thread/thread/aff4ba7373e21284/201a413efa67c26e` (2011-12-06)

> I must admit, it is not clear to me how OpenSocial REST API violates the six rules that Roy stated.

The above quote warrants a short comment. I also thought before, that REST would be so simple that there wouldn't be much need for further studying. Every web developer has some understanding of URIs, HTTP and a bit less of Hypermedia. So it is easy to fall into the trap that everything build on top of HTTP would be restful. Now however, after some more reading about REST, I can easily find violations of the REST constraints in the OpenSocial specification.

Restful APIs are modeled around resources, their representations and links between them. The authors of the OpenSocial API however seem to have modeled their API around the concept of services:[Ope11, Social API Server, sec 2, Services]

> OpenSocial defines several services for providing access to a container's data.

Fielding's critique Fielding listed some rules that a restful API must obey, but did not give explicit examples how OpenSocial violates this rules. The following section will provide such examples.

> A REST API should not be dependent on any single communication protocol, [...] any protocol element that uses a URI for identification must allow any URI scheme to be used for the sake of that identification. *[Failure here implies that identification is not separated from interaction.]*

OpenSocial defines a construct called "REST-URI-Fragment" which is a clear violation of the above rule. This URI fragment is simply an encoding of procedural parameters as elements of an HTTP URI:[Ope11, Core API Server, sec 2.1.1.2.2, REST-URI-Fragment]

> Each service type defines an associated partial URI format. The base URI for each service is found in the URI element associated with the service in the discovery document. Each service type accepts parameters via the URL path. Definitions are of the form:
>
> `{a}/{b}/{c}`

URIs can contain a query component that would be more appropriate to contain parameters. This would also have made it clearer to see that the specification actually defines services instead of resources. One test showing the misfit is to ask how dot-segments ('.' and '..') inside the URI fragment are interpreted and whether this conforms with the letter and spirit of the URI standard.[BLFM05, sec 3.3] Another misfit can be seen in the URI fragment to retrieve one or multiple albums. In this case the 'c' part in the quoted definition above is actually a list of albums to retrieve separated by a slash.

Fielding's second bullet point most likely refers to the `X-HTTP-Method-Override` header. This header is a widely used[4] workaround to allow the use of other HTTP methods than GET and POST from HTML forms or through firewalls.

The next two points again refer to a more serious issue:

> A REST API should spend almost all of its descriptive effort in defining the media type(s) used for representing resources and driving application state[...]. *[Failure here implies that out-of-band information is driving interaction instead of hypertext.]* A REST API must not define fixed resource names or hierarchies[...] *[Failure here implies that clients are assuming a resource structure due to out-of band information[...]].*

[4] `http://www.subbu.org/blog/2008/07/another-rest-anti-pattern` (2011-12-06)

The OpenSocial specification contains a lot of out-of-band information describing how to form URIs to access information or which methods to use on which URIs for different actions. This means that the OpenSocial API is not simple or intuitive to use but requires a client developer to read a lot of specification, thus violating the simplicity property of a restful architecture. Since the URIs are fixed in the specification and necessarily also in clients, the modifiability property is also violated.[Fie00, sec 2.3]

The following tables give some examples of the specified URIs:

URI fragment	Method	Description
`/people/{User-Id}/@self`	GET	profile for User-Id
`/people/{User-Id}/@self`	DELETE	remove User
`/people/{User-Id}/{Group-Id}`	GET	full profiles of group members
	POST	Create relationship, target specified by `<entry><id>` in body
	POST	Update Person
`/people/{Initial-User-Id}/{Group-Id}/{Related-User-Id}`	GET	???
`/people/@supportedFields`	GET	list of supported person profile fields
`/groups/{User-Id}[/{Group-Id}]`	GET	one or all groups of a user
	PUT	update group
	DELETE	delete group
`/groups/{User-Id}`	POST	create group

Table 2: URI fragments for peoples and groups in the OpenSocial REST API

URI fragment	Method	Description
`/albums/{User-Id}/@self`	POST	create album
`/albums/{User-Id}/{Group-Id}[/Album-Id]*`	GET	one or multiple albums
`/mediaItems/{User-Id}/{Group-Id}/{Album-Id}/{MediaItem-Id}`	GET	one mediaitem
`/mediaItem/{User-Id}/@self/{Album-Id}` (sic!)	POST	create mediaitem

Table 3: URI fragments for albums and mediaitems in the OpenSocial REST API

The last URI in Table 3 is obviously missing an "s" behind `mediaItem`. This typo is present and unfixed in the OpenSocial spec since Version 1.0, released in march 2010. This is of course not a big issue in itself, but rather a sign that the specification is too verbose and does over-specify things that should rather be auto-discovered through hyperlinks.

Fielding mentions in a comment to the same blog post that the OpenSocial API "could be made so [restful] with some relatively small changes" but does not specify these changes. However some issues can easily be identified.

First the data structures defined in OpenSocial do not use URIs to refer to other resources. Instead they use Object-Ids that must then be inserted in the appropriate URI templates. Examples are the `recipients`, `senderId`, `collectionIds` of messages and the `ownerId` of albums. The person structure does not contain fields referencing other resources. Thus it does not obviously violate REST like the albums and messages. However it does so even worse since there are hidden references only defined out-of-band in the specification. One can retrieve the albums, relations or messages of a user by filling in the `userId` in one of the specified URI templates. If Users would just contain references to other resources related to a user, the specification could already be shortened a lot.

Another missed opportunity for a much more intuitive API is the relation of media items and albums. This seems to be poster child example for a collection (album) to collection-element (media item) relation which could have made use of the hierarchical character of URI paths. OpenSocial however requires the client developer to use two different URI templates. (Table 3)

A not so small change to OpenSocial would be to either use already standardized and registered media types where possible or to register new types where necessary. It seems that there are some already existing media types that could be a good fit for OpenSocial but only miss a canonical json representation for easy consumption by javascript applications. These are vCard for persons,[5] ATOM entries[NS05] for messages, activities and media items and ATOM categories, collections or workspaces[Gh07] for albums and groups. It would probably be necessary to add extensions to the mentioned media types but vCard and ATOM both already anticipated this need and provided mechanisms to do so.

The use of the ATOM format could promote the adoption of OpenSocial because developers could either reuse existing knowledge about ATOM or would be more motivated to learn about a system that is based on an already widespread format. In fact OpenSocial already mentions ATOM as a way to wrap OpenSocial data. However this wrapping does not build extend and reuse ATOM semantics as proposed above but just puts the OpenSocial data structures inside the entry/content element of ATOM. This kind of misuse of ATOM does of course not deliver any advantage on top of the existing plain JSON or XML representations.[6] Consequently the newest OpenSocial specification deprecates any reference to the ATOM format.

In Algermissen's classification (3.0.1), the OpenSocial REST API would actually be "HTTP-based Type I" due to the lack of media types and direct hyper links between related resources. Algermissen writes that this level has the lowest possible initial cost of all HTTP APIs. Or in other words: The OpenSocial specification authors did not have to invest a lot to come up with this API specification but maintenance and evolution cost may be medium or high.

5. Persistency for Groupware Data

Relational Databases vs. NoSQL databases vs. plain files

Relational databases are not practical for contacts, events or todos. Common patterns in systems that use relational DBs for that purpose:

- artificial limits of entries, e.g. only 3 email addresses per contact, because there are only three columns email1, email2 and email3.
- Fields for custom data like custom1 to customX
- EAV pattern: tables like: id, foreign_id, type, value

6. Synchronizing a large collection

How to efficiently synchronize a large collection of contacts with the server without checking each contact for changes?

Portable Contacts has a filter "updatedSince".

How is synchronization done in CardDAV?

[5] OpenSocial persons are based on portable contacts which in turn borrowed field names from vCards.

[6] compare Bill de Hora, Extensions v Envelopes. 11/2009
`http://www.dehora.net/journal/2009/11/28/extensions-v-envelopes` (2011-21-07)

7. Mediatypes

7.1. Updates with alternative Mediatypes

How to handle updates, if the mediatypes are not isomorph?
How does Google handle PATCH in the calendar API?

7.2. Mediatype conversion

Which fields of portable contacts are derived from vCard: `http://wiki.portablecontacts.net/w/page/17776141/schema`

> No single data representation is ideal for every client. This protocol defines representations for each resource in three widely supported formats, JSON [RFC4627], XML, and Atom [RFC4287] / AtomPub [RFC5023], using a set of generic mapping rules. The mapping rules allow a server to write to a single interface rather than implementing the protocol three times.

[Ope11, Core API Server]

8. Hypermedia in RESTful applications

> The model application is therefore an engine that moves from one state to the next by examining and choosing from among the alternative state transitions in the current set of representations.

[Fie00, sec. 5.3, p.103]

8.1. Hypermedia in OpenSocial

Webfinger, e.g. get a profile picture from an email address
Danger: One can trigger na http request by sending an email.

9. Selection of components

Apache Shindig for Open Social, includes client tests
http://code.google.com/p/kolab-android/
https://evolvis.org/projects/kolab-ws/
http://packages.ubuntu.com/source/maverick/dovecot-metadata-plugin https://launchpad.net/ubu
metadata-plugin/8-0ubuntu1

9.1. REST framework

Jersey recommended by [Kai11] above Restfulie and RESTeasy because of maturity and flexibility.

10. Testing

How to test the ReST/CardDAV interface?
Portable Contacts test client at plaxo `http://www.plaxo.com/pdata/testClient`
`http://code.google.com/p/rest-assured/` `http://restfuse.com/`

A. Standards

A.1. Contacts / Persons

RFC 6450 vCard Format Specification

This document defines the vCard data format for representing and exchanging a variety of information about individuals and other entities (e.g., formatted and structured name and delivery addresses, email address, multiple telephone numbers, photograph, logo, audio clips, etc.). This is the new version and obsoletes RFCs 2425, 2426, and 4770, and updates RFC 2739.

RFC 6351 xCard: vCard XML Representation

This document defines the XML schema of the vCard data format.

Portable Contacts, OpenSocial

Portable Contacts defines contact data structures and a ReST API. It has been integrated in the OpenSocial standard.

Nepomuk Semantic Desktop Contact Ontology

Friend of a friend (FOAF)

FOAF is a

hCard

A.2. Calendaring

RFC 5545 Internet Calendaring and Scheduling Core Object Specification

iCalendar is the core data schema for calendaring information. This is the new version and obsoletes RFC2445

RFC 6321 xCal: The XML format for iCalendar

This specification defines a format for representing iCalendar data in XML. More specifically, is to define an XML format that allows iCalendar data to be converted to XML, and then back to iCalendar, without losing any semantic meaning in the data. Anyone creating XML calendar data according to this specification will know that their data can be converted to a valid iCalendar representation as well.

CalWS RESTful Web Services Protocol for Calendaring

This document, developed by the XML Technical Committee, specifies a RESTful web services Protocol for calendaring operations. This protocol has been contributed to OASIS WS-CALENDAR as a component of the WS-CALENDAR Specification under development by OASIS.

Google Calendar API V3

While not being a standard, the Google Calendar API is RESTful and will surely be implemented by many client applications. It's remarkable that the API supports partial GETs returning only specified fields and the HTTP PATCH verb to update only specified fields.

Open Services for Lifecycle Collaboration (OSLC)

uses FOAF person `http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA?sortcol=table;up=#foaf_Person_Resource`

provides change management, some overlapping to iCal TODOs `http://open-services.net/bin/view/Main/CmSpecificationV2`

reference implementation: `http://eclipse.org/lyo`

A.3. Scheduling

RFC 5546 iCalendar Transport-Independent Interoperability Protocol (iTIP)

Scheduling Events, BusyTime, To-dos and Journal Entries; Specifies the mechanisms for calendaring event interchange between calendar servers. This is the new version and obsoletes RFC2446

RFC 6047 iCalendar Message-Based Interoperability Protocol (iMIP)

Specifies how to exchange calendaring data via e-mail. This is the new version and obsoletes RFC2447.

A.4. Relations and Links

Xhtml Friends Network (XFN)

One of the relations returned by Google's webfinger.

Webfinger

Webfinger in Firefox Contacts Add-On `http://mozillalabs.com/blog/2010/03/contacts-in-the-`

RFC 6415 Web Host Metadata

Extensible Resource Descriptor (XRD)

A.5. out of scope

OMA Converged Address Book V1.0

Standard by the Open Mobile Alliance defining data structures and synchronization of contact data. It references vCard.

W3C Contacts API

A standard on how address books cold be accessed on devices or from JavaScript inside a Web Browser. The standard references vCard, OMA Converged Address Book and Portable Contacts.

W3C vCard ontology

W3C PIM ontology

HR XML

The HR-XML Consortium is the only independent, non-profit, volunteer-led organization dedicated to the development and promotion of a standard suite of XML specifications to enable e-business and the automation of human resources-related data exchanges.

B. People, Groups and Organizations

People

Eran Hammer-Lahav
`http://hueniverse.com` Yahoo!, OAuth

Eliot Lear ¡lear@cisco.com¿
IETF Calsify WG chair

Lisa Dusseault

Lisa Dusseault is a development manager and standards architect at the Open Source Applications Foundation, where she's involved in the Chandler, Cosmo and Scooby projects. Previously, Lisa came from Xythos, an Internet startup where she was development manager for four years. She has also been an IETF contributor on various Internet applications protocols for eight years now, and continues to do this kind of work at OSAF. She co-chairs the IETF IMAP extensions and CALSIFY (Calendaring and Scheduling Standards Simplification) Working Groups. She is also the author of a book on WebDAV and co-author of CalDAV, an open and interoperable protocol for calendar access and sharing.

Peter Saint-Andre ¡stpeter@stpeter.im¿

IETF Calsify WG area director

Joseph Smarr

former Plaxo now Google presentation about portable contacts at vcarddav wg http://tools.ietf.org/age
2.pdf http://josephsmarr.com http://anyasq.com/79-im-a-technical-lead-on-the-google+-team

Mike Conley

`http://mikeconley.ca/blog/` working on a new address book for Thunderbird: `https://wiki.mozilla.org/Thunderbird/tb-planning`

C. Implementations

C.1. Servers

Cyn.in
Python, Open Core

DAViCal

PHP, SQL storage, CalDAV, CardDav

eGroupWare

eXo Platform
Open Core, Java, AGPL, participates in OpenSocial?

Group-Office
PHP, AGPL

Horde

OBM Groupware
PHP, GPL

owncloud

ownCloud supports syncing of calendar and contacts information via the CalDAV and CardDAV protocols.

Scalix
Open Core Scalix Public License (SPL) based on MPL, requires to show the Scalix Logo

Simple Groupware
PHP, GPL, SQL

SOGo
CalDAV and CardDAV, written in Objective-C

Tiki Wiki
PHP, SQL Contacts `http://doc.tiki.org/Contacts`, Calendar `http://doc.tiki.org/Calendar` iCal export apparently no CardDAV/CalDAV many many features!

Tine 2.0
Tine is not eGroupWare

Zarafa

Zimbra
Open Core, Own license (Zimbra Public License), RFP since 2008 open: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=498316

C.2. Clients

Spicebird
built on top of Thunderbird with Calendar

Thunderbird

CardDAV via SoCO connector http://www.sogo.nu/fr/downloads/frontends.html

Evolution, Evolution Data Server

KDE Kontact, Akonadi

more CardDAV
http://wiki.davical.org/w/CardDAV/Clients http://en.wikipedia.org/wiki/CardDAV#Implementations

more CalDAV
http://wiki.davical.org/w/CalDAV_Clients http://en.wikipedia.org/wiki/CalDAV#Implementations

C.3. Web Services

C.4. Portable Contacts

D. Links

- http://thesocialweb.tv
- http://www.vogella.de/articles/REST/article.html REST with Java (JAX-RS) using Jersey - Tutorial
- https://addons.mozilla.org/de/firefox/addon/restclient/
- http://dataportability.org/ still active?

IANA link relations registry http://www.iana.org/assignments/link-relations/link-relatic
xml
ATOM landscape overview http://dret.typepad.com/dretblog/atom-landscape.html

D.1. Apache Shindig

RPC vs. REST API for Shindig/OpenSocial: http://groups.google.com/group/opensocial-and-ga
browse_thread/thread/a4ddf7cd09f90237/5cfa1658e1c1d698?lnk=gst&q=rest#5cfa1658e1c1d698
http://groups.google.com/group/opensocial-and-gadgets-spec/browse_thread/thread/d1a5627fb6e686ce/d27d47dee92a87b2 One argument was support for batching. A restful batching proposal didn't get consensus: https://docs.google.com/View?docid=dc43mmng_23fdbpp7hd&pli=1
Flow of REST requests in Shindig https://sites.google.com/site/opensocialarticles/Home/shindig-rest-java
Google+ is likely to become OpenSocial enabled: http://groups.google.com/group/opensocial-and-gadgets-spec/browse_thread/thread/1187241df6759a9a

Shindig issues to implement OpenSocial 2.0 https://docs.google.com/spreadsheet/ccc?key=0AihdZBncP3KzdGN3dVl3MFpIUlk2TXIyR3hfUDhHZUE&hl=en_US#gid=0

How Shindig supports extensions: https://cwiki.apache.org/confluence/display/SHINDIG/Arbitrary+Extensions+to+Apache+Shindig%27s+Data+Model

Videos about some 2.0 OS features http://groups.google.com/group/opensocial-and-gadgets-browse_thread/thread/7b911edfb1bb3b4d

OS and RDF http://groups.google.com/group/opensocial-and-gadgets-spec/browse_thread/thread/20f62d627003509b

OpenSocial Development Environment (OSDE, Eclipse Plugin) https://sites.google.com/site/opensocialdevenv

D.2. Socialsite

Oracle's (former Sun's) extension to Apache Shindig. Blog http://blogs.oracle.com/socialsite

E. TODO

- Does funambol.org has interesting implementations?

support Plain Text Format (text/plain), RFC5147 URI fragment identifier for plain text?

> Sowohl Atom als auch AtomPub definieren XML-Vokabulare, die eine Erweiterung mit zwei Mechanismen unterstützen. Zum einen ist im Standard definiert, dass neue Elemente in diesen Vokaularen selbst von standardkonformen Prozessoren ignoeriert werden müssen. [...] Gleichzeitig ist es überall dort, wo es nicht explizit verboten ist, möglich, Elemente ausanderen XML-Namespaces einzubetten.

[Til11, p. 102]

References

[Amu10] Amundsen, Mike: *Fielding Property Maps*. http://amundsen.com/examples/fielding-props/, 3 2010

[BLFM05] Berners-Lee, T. ; Fielding, R. ; Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax / RFC Editor. RFC Editor, January 2005 (3986). – RFC

[Fie00] Fielding, Roy T.: *REST: Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, Doctoral dissertation, 2000

[Gh07] Gregorio, J. ; hOra, B. de: The Atom Publishing Protocol / RFC Editor. RFC Editor, October 2007 (5023). – RFC

[Kai11] Kaiser, Guido: *Modellierung und Vergleich von Implementierungen einer REST-Anwendung in verschiedenen Sprachen*, Fernuniversität Hagen, Master thesis, 1 2011. http://www.fernuni-hagen.de/dvt/studium/masterarbeiten_9.shtml

[NS05] Nottingham, M. ; Sayre, R.: The Atom Syndication Format / RFC Editor. RFC Editor, December 2005 (4287). – RFC

[Ope11] OpenSocial and Gadgets Specification Group: *OpenSocial Specification Version 2.0.1*. `http://docs.opensocial.org/display/OSD/Specs`. Version: 11 2011

[Til11] Tilkov, Stefan: *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien*. 2. DPUNKT VERLAG, 2011