FOAF Vocabulary Specification 0.96

Namespace Document 15 December 2009 - *Amsterdam Edition*

This version:
http://xmlns.com/foaf/spec/20091215.html (rdf)
Latest version:
http://xmlns.com/foaf/spec/ (rdf)
Previous version:
http://xmlns.com/foaf/spec/20071002.html (rdf)
Authors:
Dan Brickley, Libby Miller
Contributors:
Members of the FOAF mailing list (foaf-dev@lists.foaf-project.org) and the wider RDF and SemWeb developer community. See acknowledgements.

Abstract

This specification describes the FOAF language, defined as a dictionary of named properties and classes using W3C's RDF technology.

Status of This Document

FOAF has been evolving gradually since its creation in mid-2000. There is now a stable core of classes and properties that will not be changed, beyond modest adjustments to their documentation to track implementation feedback and emerging best practices. New terms may be added at any time (as with a natural-language dictionary), and consequently this specification is an evolving work. The FOAF RDF namespace, by contrast, is fixed and its identifier is not expected to change. Furthermore, efforts are underway to ensure the long-term preservation of the FOAF namespace, its xmlns.com domain name and associated documentation.

This document is created by combining the RDFS/OWL machine-readable FOAF ontology with a set of per-term documents. Future versions may incorporate multilingual translations of the term definitions. The RDF version of the specification is also embedded in the HTML of this document, or available directly from the namespace URI by content negotiation.

The FOAF specification is produced as part of the FOAF project, to provide authoritative documentation of the contents, status and purpose of the RDF/XML vocabulary and document formats known informally as 'FOAF'.

The authors welcome comments on this document, preferably via the public FOAF developers list foaf-dev@lists.foaf-project.org; public archives are available. A historical backlog of known technical issues is acknowledged, and available for discussion in the FOAF wiki. Proposals for resolving these issues are welcomed, either on foaf-dev or via the wiki. Further work is also needed on the explanatory text in this specification and on the FOAF website; progress towards this will be measured in the version number of future revisions to the FOAF specification.

This revision of the specification includes the following changes:

- `foaf:givenName` and `foaf:familyName` have been changed from `foaf:givenname` and `foaf:family_name` to make FOAF in line with usage of these terms in the Portable Contacts forma. The previous versions remain in the document, marked as 'archaic'.
- Two little-used and poorly defined properties have been marked as 'archaic': `foaf:fundedBy` and `foaf:theme`
- `foaf:holdsAccount` has been marked as 'archaic' in favour of `foaf:account`

In addition, this revision of the specification validates according to the RDFa DTD, as the RDF/XML has been removed from the HTML following discussion on the mailing slist.

As usual, see the changes section for details of the changes in this version of the specification.

Table of Contents

FOAF at a glance

An a-z index of FOAF terms, by class (categories or types) and by property.

Classes: | Agent | Document | Group | Image | OnlineAccount | OnlineChatAccount | OnlineEcommerceAccount | OnlineGamingAccount | Organization | Person | PersonalProfileDocument | Project |

Properties: | account | accountName | accountServiceHomepage | age | aimChatID | based_near | birthday | currentProject | depiction | depicts | dnaChecksum | familyName | family_name | firstName | fundedBy | geekcode | gender | givenName | givenname | holdsAccount | homepage | icqChatID | img | interest | isPrimaryTopicOf | jabberID | knows | logo | made | maker | mbox | mbox_sha1sum | member | membershipClass | msnChatID | myersBriggs | name | nick | openid | page | pastProject | phone | plan | primaryTopic | publications | schoolHomepage | sha1 | surname | theme | thumbnail | tipjar | title | topic | topic_interest | weblog | workInfoHomepage | workplaceHomepage | yahooChatID |

FOAF terms, grouped in broad categories.

FOAF Basics

- Agent
- Person
- name
- nick
- title
- homepage
- mbox
- mbox_sha1sum
- img
- depiction (depicts)
- surname
- familyName
- givenName
- firstName

Personal Info

- weblog
- knows
- interest
- currentProject
- pastProject
- plan
- based_near
- workplaceHomepage
- workInfoHomepage
- schoolHomepage
- topic_interest
- publications
- geekcode
- myersBriggs
- dnaChecksum

Online Accounts / IM

- OnlineAccount
- OnlineChatAccount
- OnlineEcommerceAccount
- OnlineGamingAccount
- account
- accountServiceHomepage
- accountName
- icqChatID
- msnChatID
- aimChatID
- jabberID
- yahooChatID

Projects and Groups

- Project
- Organization
- Group
- member
- membershipClass

Documents and Images

- Document
- Image
- PersonalProfileDocument
- topic (page)
- primaryTopic (primaryTopicOf)
- tipjar
- sha1
- made (maker)
- thumbnail
- logo

Example

Here is a very basic document describing a person:

```
<foaf:Person rdf:about="#me" xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:name>Dan Brickley</foaf:name>
  <foaf:mbox_sha1sum>241021fb0e6289f92815fc210f9e9137262c252e</foaf:mbox_sha1sum>
  <foaf:homepage rdf:resource="http://danbri.org/" />
  <foaf:img rdf:resource="/images/me.jpg" />
</foaf:Person>
```

This brief example introduces the basics of FOAF. It basically says, "there is a foaf:Person with a foaf:name property of 'Dan Brickley' and a foaf:mbox_sha1sum property of 241021fb0e6289f92815fc210f9e9137262c252e; this person stands in a foaf:homepage relationship to a thing called http://danbri.org/ and a foaf:img relationship to a thing referenced by a relative URI of /images/me.jpg

1 Introduction: FOAF Basics

The Semantic Web

> *To a computer, the Web is a flat, boring world, devoid of meaning. This is a pity, as in fact documents on the Web describe real objects and imaginary concepts, and give particular relationships between them. For example, a document might describe a person. The title document to a house describes a house and also the ownership relation with a person. Adding semantics to the Web involves two things: allowing documents which have information in machine-readable forms, and allowing links to be created with relationship values. Only when we have this extra level of semantics will we be able to use computer power to help us exploit the information to a greater extent than our own reading.* - Tim Berners-Lee "W3 future directions" keynote, 1st World Wide Web Conference Geneva, May 1994

FOAF and the Semantic Web

FOAF, like the Web itself, is a linked information system. It is built using decentralised Semantic Web technology, and has been designed to allow for integration of data across a variety of applications, Web sites and services, and software systems. To achieve this, FOAF takes a liberal approach to data exchange. It does not require you to say anything at all about yourself or others, nor does it place any limits on the things you can say or the variety of Semantic Web vocabularies you may use in doing so. This current specification provides a basic "dictionary" of terms for talking about people and the things they make and do.

FOAF was designed to be used alongside other such dictionaries ("schemas" or "ontologies"), and to be usable with the wide variety of generic tools and services that have been created for the Semantic Web. For example, the W3C work on SPARQL provides us with a rich query language for consulting databases of FOAF data, while the SKOS initiative explores in more detail than FOAF the problem of describing topics, categories, "folksonomies" and subject hierarchies. Meanwhile, other W3C groups are working on improved mechanisms for encoding all kinds of RDF data (including but not limited to FOAF) within Web pages: see the work of the GRDDL and RDFa efforts for more detail. The Semantic Web provides us with an *architecture for collaboration*, allowing complex technical challenges to be shared by a loosely-coordinated community of developers.

The FOAF project is based around the use of *machine readable* Web homepages for people, groups, companies and other kinds of thing. To achieve this we use the "FOAF vocabulary" to provide a collection of basic terms that can be used in these Web pages. At the heart of the FOAF project is a set of definitions designed to serve as a dictionary of terms that can be used to express claims about the world. The initial focus of FOAF has been on the description of people, since people are the things that link together most of the other kinds of things we describe in the Web: they make documents, attend meetings, are depicted in photos, and so on.

The FOAF Vocabulary definitions presented here are written using a computer language (RDF/OWL) that makes it easy for software to process some basic facts about the terms in the FOAF vocabulary, and consequently about the things described in FOAF documents. A FOAF document, unlike a traditional Web page, can be combined with other FOAF documents to create a unified database of information. FOAF is a Linked Data system, in that it based around the idea of linking together a Web of decentralised descriptions.

The Basic Idea

The basic idea is pretty simple. If people publish information in the FOAF document format, machines will be able to make use of that information. If those files contain "see also" references to other such documents in the Web, we will have a machine-friendly version of today's hypertext Web. Computer programs will be able to scutter around a Web of documents designed for machines rather than humans, storing the information they find, keeping a list of "see also" pointers to other documents, checking digital signatures (for the security minded) and building Web pages and question-answering services based on the harvested documents.

So, what is the 'FOAF document format'? FOAF files are just text documents (well, Unicode documents). They adopt the conventions of the Resource Description Framework (RDF), and may be written in XML syntax or any other of the syntaxes of RDF such as RDFa or N3. In addition, the FOAF vocabulary defines some useful constructs that can appear in FOAF files, alongside other RDF vocabularies defined elsewhere. For example, FOAF defines categories ('classes') such as `foaf:Person`, `foaf:Document`, `foaf:Image`, alongside some handy properties of those things, such as `foaf:name`, `foaf:mbox` (ie. an internet mailbox), `foaf:homepage` etc., as well as some useful kinds of relationship that hold between members of these categories. For example, one interesting relationship type is `foaf:depiction`. This relates something (eg. a

`foaf:Person`) to a `foaf:Image`. The FOAF demos that feature photos and listings of 'who is in which picture' are based on software tools that parse RDF documents and make use of these properties.

The specific contents of the FOAF vocabulary are detailed in this FOAF namespace document. In addition to the FOAF vocabulary, one of the most interesting features of a FOAF file is that it can contain "see Also" pointers to other FOAF files. This provides a basis for automatic harvesting tools to traverse a Web of interlinked files, and learn about new people, documents, services, data...

The remainder of this specification describes how to publish and interpret descriptions such as these on the Web, using RDF/XML for syntax (file format) and terms from FOAF. It introduces a number of categories (RDF classes such as 'Person') and properties (relationship and attribute types such as 'mbox' or 'workplaceHomepage'). Each term definition is provided in both human and machine-readable form, hyperlinked for quick reference.

What's FOAF for?

For a good general introduction to FOAF, see Edd Dumbill's article, XML Watch: Finding friends with XML and RDF (June 2002, IBM developerWorks). Information about the use of FOAF with image metadata is also available.

The co-depiction experiment shows a fun use of the vocabulary. Jim Ley's SVG image annotation tool show the use of FOAF with detailed image metadata, and provide tools for labelling image regions within a Web browser. To create a FOAF document, you can use Leigh Dodd's FOAF-a-matic javascript tool. For more information on FOAF and related projects, see the FOAF project home page.

Background

FOAF is a collaborative effort amongst Semantic Web developers on the FOAF (foaf-dev@lists.foaf-project.org) mailing list. The name 'FOAF' is derived from traditional internet usage, an acronym for 'Friend of a Friend'.

The name was chosen to reflect our concern with social networks and the Web, urban myths, trust and connections. Other uses of the name continue, notably in the documentation and investigation of Urban Legends (eg. see the alt.folklore.urban archive or snopes.com), and other FOAF stories. Our use of the name 'FOAF' for a Web vocabulary and document format is intended to complement, rather than replace, these prior uses. FOAF documents describe the characteristics and relationships amongst friends of friends, and their friends, and the stories they tell.

FOAF and Standards

It is important to understand that the FOAF *vocabulary* as specified in this document is not a standard in the sense of ISO Standardisation, or that associated with W3C Process.

FOAF depends heavily on W3C's standards work, specifically on XML, XML Namespaces, RDF, and OWL. All FOAF *documents* must be well-formed RDF documents. The FOAF vocabulary, by contrast, is managed more in the style of an Open Source or Free Software project than as an industry standardarisation effort (eg. see Jabber JEPs).

This specification contributes a vocabulary, "FOAF", to the Semantic Web, specifying it using W3C's Resource Description Framework (RDF). As such, FOAF adopts by reference both syntaxes (using XML, N3, or RDFa) a data model (RDF graphs) and a mathematically grounded definition for the rules that underpin the FOAF design.

The FOAF Vocabulary Description

This specification serves as the FOAF "namespace document". As such it describes the FOAF vocabulary and the terms (RDF classes and properties) that constitute it, so that Semantic Web applications can use those terms in a variety of RDF-compatible document formats and applications.

This document presents FOAF as a Semantic Web vocabulary or *Ontology*. The FOAF vocabulary is pretty simple, pragmatic and designed to allow simultaneous deployment and extension. FOAF is intended for widescale use, but its authors make no commitments regarding its suitability for any particular purpose.

Evolution and Extension of FOAF

The FOAF vocabulary is identified by the namespace URI '`http://xmlns.com/foaf/0.1/`'. Revisions and extensions of FOAF are conducted through edits to this document, which by convention is accessible in the Web via the namespace URI. For practical and deployment reasons, note that **we do not update the namespace URI as the vocabulary matures**.

The core of FOAF now is considered stable, and the version number of *this specification* reflects this stability. However, it long ago became impractical to update the namespace URI without causing huge disruption to both producers and consumers of FOAF data. We are therefore left with the digits "0.1" in our URI. This stands as a warning to all those who might embed metadata in their vocabulary identifiers.

The evolution of FOAF is best considered in terms of the stability of individual vocabulary terms, rather than the specification as a whole. As terms stabilise in usage and documentation, they progress through the categories '**unstable**', '**testing**' and '**stable**'. Older terms are marked '**archaic**' which allows the possibility of older forms to become modern again.

The properties and types defined here provide some basic useful concepts for use in FOAF descriptions. Other vocabulary (eg. the Dublin Core metadata elements for simple bibliographic description), RSS 1.0 etc can also be mixed in with FOAF terms, as can local extensions. FOAF is designed to be extended. The FoafVocab page in the FOAF wiki lists a number of extension vocabularies that are particularly applicable to use with FOAF.

FOAF Auto-Discovery: Publishing and Linking FOAF files

If you publish a FOAF self-description (eg. using foaf-a-matic) you can make it easier for tools to find your FOAF by putting markup in the `head` of your HTML homepage. It doesn't really matter what filename you choose for your FOAF document, although `foaf.rdf` is a common choice. The linking markup is as follows:

```
<link rel="meta" type="application/rdf+xml" title="FOAF"
            href="http://example.com/~you/foaf.rdf"/>
```

...although of course change the *URL* to point to your own FOAF document. See also: more on FOAF autodiscovery and services that make use of it.

FOAF and RDF

Why does FOAF use RDF?

FOAF is an application of the Resource Description Framework (RDF) because the subject area we're describing -- people -- has so many competing requirements that a standalone format could not do them all justice. By using RDF, FOAF gains a powerful extensibility mechanism, allowing FOAF-based descriptions can be mixed with claims made in *any other RDF vocabulary*

People are the things that link together most of the other kinds of things we describe in the Web: they make documents, attend meetings, are depicted in photos, and so on. Consequently, there are many many things that we might want to say about people, not to mention these related objects (ie. documents, photos, meetings etc).

FOAF as a vocabulary cannot incorporate everything we might want to talk about that is related to people, or it would be as large as a full dictionary. Instead of covering all topics within FOAF itself, we buy into a larger framework - RDF - that allows us to take advantage of work elsewhere on more specific description vocabularies (eg. for geographical / mapping data).

RDF provides FOAF with a way to mix together different descriptive vocabularies in a consistent way. Vocabularies can be created by different communites and groups as appropriate and mixed together as required, without needing any centralised agreement on how terms from different vocabularies can be written down in XML.

This mixing happens in two ways: firstly, RDF provides an underlying model of (typed) objects and their attributes or relationships. `foaf:Person` is an example of a type of object (a "*class*"), while `foaf:knows` and `foaf:name` are examples of a relationship and an attribute of an `foaf:Person`; in RDF we call these "*properties*". Any vocabulary described in RDF shares this basic model, which is discernable in the syntax for RDF, and which removes one level of confusion in *understanding* a given vocabulary, making it simpler to comprehend and therefore reuse a vocabulary that you have not written yourself. This is the minimal *self-documentation* that RDF gives you.

Secondly, there are mechanisms for saying which RDF properties are connected to which classes, and how different classes are related to each other, using RDF Syntax and OWL. These can be quite general (all RDF properties by default come from an `rdf:Resource` for example) or very specific and precise (for example by using OWL constructs, as in the `foaf:Group` example below). This is another form of self-documentation, which allows you to connect different vocabularies together as you please. An example of this is given below where the `foaf:based_near` property has a domain and range (types of class at each end of the property) from a different namespace altogether.

In summary then, RDF is self-documenting in ways which enable the creation and combination of vocabularies in a devolved manner. This is particularly important for a vocabulary which describes people, since people connect to many other domains of interest, which it would be impossible (as well as suboptimal) for a single group to describe adequately in non-geological

time.

RDF is often written using XML syntax, but when written down in XML, it behaves in rather different ways to 'vanilla' XML: the same RDF can be written in many different ways in XML. This means that SAX and DOM XML parsers are not adequate to deal with RDF/XML. If you want to process the data, you will need to use one of the many RDF toolkits available, such as Jena (Java) or Redland (C). Semantic Web Interest Group members can help with issues which may arise; there is also the foaf-dev@lists.foaf-project.org mailing list which is the main list for FOAF, and two active and friendly IRC channels: #swig and #foaf on freenode.

FOAF cross-reference: Listing FOAF Classes and Properties

FOAF introduces the following classes and properties. View this document's source markup to see the RDF/XML version.

Classes: | Agent | Document | Group | Image | OnlineAccount | OnlineChatAccount | OnlineEcommerceAccount | OnlineGamingAccount | Organization | Person | PersonalProfileDocument | Project |

Properties: | account | accountName | accountServiceHomepage | age | aimChatID | based_near | birthday | currentProject | depiction | depicts | dnaChecksum | familyName | family_name | firstName | fundedBy | geekcode | gender | givenName | givenname | holdsAccount | homepage | icqChatID | img | interest | isPrimaryTopicOf | jabberID | knows | logo | made | maker | mbox | mbox_sha1sum | member | membershipClass | msnChatID | myersBriggs | name | nick | openid | page | pastProject | phone | plan | primaryTopic | publications | schoolHomepage | sha1 | surname | theme | thumbnail | tipjar | title | topic | topic_interest | weblog | workInfoHomepage | workplaceHomepage | yahooChatID |

Classes and Properties (full detail)

Classes

Class: foaf:Agent

Agent - An agent (eg. person, group, software or physical artifact).

Status:	stable
May be the object of:	MSN chat ID weblog Yahoo chat ID made jabber ID sha1sum of a personal mailbox URI name personal mailbox openid AIM chat ID gender holds account holds account tipjar age ICQ chat ID birthday
May have properties:	maker member
has subclass	Organization Person Group
Disjoint With:	Document

The `Agent` class is the class of agents; things that do stuff. A well known sub-class is `Person`, representing people. Other kinds of agents include `Organization` and `Group`.

The `Agent` class is useful in a few places in FOAF where `Person` would have been overly specific. For example, the IM chat ID properties such as `jabberID` are typically associated with people, but sometimes belong to software bots.

[#] [back to top]

Class: foaf:Group

Group - A class of Agents.

Status:	stable
May be the object of:	member
subClassOf	Agent

The `Group` class represents a collection of individual agents (and may itself play the role of a `Agent`, ie. something that can perform actions).

This concept is intentionally quite broad, covering informal and ad-hoc groups, long-lived communities, organizational groups within a workplace, etc. Some such groups may have associated characteristics which could be captured in RDF (perhaps a `homepage`, `name`,

mailing list etc.).

While a Group has the characteristics of a Agent, it is also associated with a number of other Agents (typically people) who constitute the Group. FOAF provides a mechanism, the membershipClass property, which relates a Group to a sub-class of the class Agent who are members of the group. This is a little complicated, but allows us to make group membership rules explicit.

The markup (shown below) for defining a group is both complex and powerful. It allows group membership rules to match against any RDF-describable characteristics of the potential group members. As FOAF and similar vocabularies become more expressive in their ability to describe individuals, the Group mechanism for categorising them into groups also becomes more powerful.

While the formal description of membership criteria for a Group may be complex, the basic mechanism for saying that someone is in a Group is very simple. We simply use a member property of the Group to indicate the agents that are members of the group. For example:

```
<foaf:Group>
 <foaf:name>ILRT staff</foaf:name>
 <foaf:member>
  <foaf:Person>
   <foaf:name>Martin Poulter</foaf:name>
   <foaf:homepage rdf:resource="http://www.ilrt.bris.ac.uk/aboutus/staff/staffprofile/?search=plmlp"/>
   <foaf:workplaceHomepage rdf:resource="http://www.ilrt.bris.ac.uk/"/>
  </foaf:Person>
 </foaf:member>
</foaf:Group>
```

Behind the scenes, further RDF statements can be used to express the rules for being a member of this group. End-users of FOAF need not pay attention to these details.

Here is an example. We define a Group representing those people who are ILRT staff members (ILRT is a department at the University of Bristol). The membershipClass property connects the group (conceived of as a social entity and agent in its own right) with the class definition for those people who constitute it. In this case, the rule is that all group members are in the ILRTStaffPerson class, which is in turn populated by all those things that are a Person and which have a workplaceHomepage of http://www.ilrt.bris.ac.uk/. This is typical: FOAF groups are created by specifying a sub-class of Agent (in fact usually this will be a sub-class of Person), and giving criteria for which things fall in or out of the sub-class. For this, we use the owl:onProperty and owl:hasValue properties, indicating the property/value pairs which must be true of matching agents.

```
<!-- here we see a FOAF group described.
     each foaf group may be associated with an OWL definition
     specifying the class of agents that constitute the group's membership -->
<foaf:Group>
 <foaf:name>ILRT staff</foaf:name>
 <foaf:membershipClass>
    <owl:Class rdf:about="http://ilrt.example.com/groups#ILRTStaffPerson">
     <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="http://xmlns.com/foaf/0.1/workplaceHomepage"/>
         <owl:hasValue rdf:resource="http://www.ilrt.bris.ac.uk/"/>
       </owl:Restriction>
     </rdfs:subClassOf>
   </owl:Class>
 </foaf:membershipClass>
</foaf:Group>
```

Note that while these example OWL rules for being in the eg:ILRTStaffPerson class are based on a Person having a particular workplaceHomepage, this places no obligations on the authors of actual FOAF documents to include this information. If the information *is* included, then generic OWL tools may infer that some person is an eg:ILRTStaffPerson. To go the extra step and infer that some eg:ILRTStaffPerson is a member of the group whose name is "ILRT staff", tools will need some knowledge of the way FOAF deals with groups. In other words, generic OWL technology gets us most of the way, but the full Group machinery requires extra work for implimentors.

The current design names the relationship as pointing *from* the group, to the member. This is convenient when writing XML/RDF that encloses the members within markup that describes the group. Alternate representations of the same content are allowed in RDF, so you can write claims about the Person and the Group without having to nest either description inside the other. For (brief) example:

```
<foaf:Group>
 <foaf:member rdf:nodeID="martin"/>
 <!-- more about the group here -->
```

```
</foaf:Group>
<foaf:Person rdf:nodeID="martin">
  <!-- more about martin here -->
</foaf:Person>
```

There is a FOAF issue tracker associated with this FOAF term. A design goal is to make the most of W3C's OWL language for representing group-membership criteria, while also making it easy to leverage existing groups and datasets available online (eg. buddylists, mailing list membership lists etc). Feedback on the current design is solicited! Should we consider using SPARQL queries instead, for example?

[#] [back to top]

Class: foaf:Organization

Organization - An organization.

Status:	stable
subClassOf	Agent
Disjoint With:	Document Person

The `Organization` class represents a kind of `Agent` corresponding to social instititutions such as companies, societies etc.

This is a more 'solid' class than `Group`, which allows for more ad-hoc collections of individuals. These terms, like the corresponding natural language concepts, have some overlap, but different emphasis.

[#] [back to top]

Class: foaf:Person

Person - A person.

Status:	stable
May be the object of:	geekcode interest familyName work info homepage publications Surname past project family name plan schoolHomepage current project myersBriggs firstName interest topic workplace homepage image knows
May have properties:	knows
subClassOf	Person Agent Spatial Thing
Disjoint With:	Project Document Organization

The `Person` class represents people. Something is a `Person` if it is a person. We don't nitpic about whether they're alive, dead, real, or imaginary. The `Person` class is a sub-class of the `Agent` class, since all people are considered 'agents' in FOAF.

[#] [back to top]

Class: foaf:Document

Document - A document.

Status:	testing
May be the object of:	primary topic topic sha1sum (hex)
May have properties:	weblog is primary topic of interest openid work info homepage publications homepage page schoolHomepage tipjar workplace homepage account service homepage
has subclass	PersonalProfileDocument
Disjoint With:	Person Project Organization

The `Document` class represents those things which are, broadly conceived, 'documents'.

The `Image` class is a sub-class of `Document`, since all images are documents.

We do not (currently) distinguish precisely between physical and electronic documents, or between copies of a work and the abstraction those copies embody. The relationship between documents and their byte-stream representation needs clarification (see `sha1` for related issues).

[#] [back to top]

Class: foaf:Image

Image - An image.

Status:	testing
May be the object of:	thumbnail depicts
May have properties:	thumbnail depiction image

The class `Image` is a sub-class of `Document` corresponding to those documents which are images.

Digital images (such as JPEG, PNG, GIF bitmaps, SVG diagrams etc.) are examples of `Image`.

[#] [back to top]

Class: foaf:PersonalProfileDocument

PersonalProfileDocument - A personal profile RDF document.

Status:	testing
subClassOf	Document

The `PersonalProfileDocument` class represents those things that are a `Document`, and that use RDF to describe properties of the person who is the `maker` of the document. There is just one `Person` described in the document, ie. the person who `made` it and who will be its `primaryTopic`.

The `PersonalProfileDocument` class, and FOAF's associated conventions for describing it, captures an important deployment pattern for the FOAF vocabulary. FOAF is very often used in public RDF documents made available through the Web. There is a colloquial notion that these "FOAF files" are often *somebody's* FOAF file. Through `PersonalProfileDocument` we provide a machine-readable expression of this concept, providing a basis for FOAF documents to make claims about their maker and topic.

When describing a `PersonalProfileDocument` it is typical (and useful) to describe its associated `Person` using the `maker` property. Anything that is a `Person` and that is the `maker` of some `Document` will be the `primaryTopic` of that `Document`. Although this can be inferred, it is helpful to include this information explicitly within the `PersonalProfileDocument`.

For example, here is a fragment of a personal profile document which describes its author explicitly:

```
<foaf:Person rdf:nodeID="p1">
 <foaf:name>Dan Brickley</foaf:name>
 <foaf:homepage rdf:resource="http://danbri.org/"/>
 <!-- etc... -->
</foaf:Person>
```

```
<foaf:PersonalProfileDocument rdf:about="">
   <foaf:maker rdf:nodeID="p1"/>
   <foaf:primaryTopic rdf:nodeID="p1"/>
</foaf:PersonalProfileDocument>
```

Note that a PersonalProfileDocument will have some representation as RDF. Typically this will be in W3C's RDF/XML syntax, however we leave open the possibility for the use of other notations, or representational conventions including automated transformations from HTML (GRDDL spec for one such technique).

[#] [back to top]

Class: foaf:OnlineAccount

Online Account - An online account.

Status:	unstable
May be the object of:	account service homepage account name
May have properties:	holds account holds account
subClassOf	A thing
has subclass	Online Gaming Account Online E-commerce Account Online Chat Account

A OnlineAccount represents the provision of some form of online service, by some party (indicated indirectly via a accountServiceHomepage) to some Agent. The account property of the agent is used to indicate accounts that are associated with the agent.

See OnlineChatAccount for an example. Other sub-classes include OnlineEcommerceAccount and OnlineGamingAccount.

[#] [back to top]

Class: foaf:OnlineChatAccount

Online Chat Account - An online chat account.

Status: unstable

subClassOf Online Account

A OnlineChatAccount is a OnlineAccount devoted to chat / instant messaging.

This is a generalization of the FOAF Chat ID properties, jabberID, aimChatID, msnChatID, icqChatID and yahooChatID.

Unlike those simple properties, OnlineAccount and associated FOAF terms allows us to describe a great variety of online accounts, without having to anticipate them in the FOAF vocabulary.

For example, here is a description of an IRC chat account, specific to the Freenode IRC network:

```
<foaf:Person>
  <foaf:name>Dan Brickley</foaf:name>
  <foaf:account>
    <foaf:OnlineAccount>
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/OnlineChatAccount"/>
      <foaf:accountServiceHomepage rdf:resource="http://www.freenode.net/irc_servers.shtml"/>
      <foaf:accountName>danbri</foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:account>
</foaf:Person>
```

Note that it may be impolite to carelessly reveal someone else's chat identifier (which might also serve as an indicate of email address) As with email, there are privacy and anti-SPAM considerations. FOAF does not currently provide a way to represent an obfuscated chat ID (ie. there is no parallel to the mbox / mbox_sha1sum mapping).

In addition to the generic OnlineAccount and OnlineChatAccount mechanisms, FOAF also provides several convenience chat ID properties (jabberID, aimChatID, icqChatID, msnChatID,yahooChatID). These serve as as a shorthand for some common cases; their use may not always be appropriate.

We should specify some mappings between the abbreviated and full representations of Jabber, AIM, MSN, ICQ, Yahoo! and MSN chat accounts. This requires us to identify an appropriate `accountServiceHomepage` for each. If we wanted to make the `OnlineAccount` mechanism even more generic, we could invent a relationship that holds between a `OnlineAccount` instance and a convenience property. To continue the example above, we could describe how Freenode could define a property 'fn:freenodeChatID' corresponding to Freenode online accounts.

[#] [back to top]

Class: foaf:OnlineEcommerceAccount

Online E-commerce Account - An online e-commerce account.
Status: unstable
subClassOf Online Account

A `OnlineEcommerceAccount` is a `OnlineAccount` devoted to buying and/or selling of goods, services etc. Examples include Amazon, eBay, PayPal, thinkgeek, etc.

[#] [back to top]

Class: foaf:OnlineGamingAccount

Online Gaming Account - An online gaming account.
Status: unstable
subClassOf Online Account

A `OnlineGamingAccount` is a `OnlineAccount` devoted to online gaming.

Examples might include EverQuest, Xbox live, Neverwinter Nights, etc., as well as older text-based systems (MOOs, MUDs and suchlike).

[#] [back to top]

Class: foaf:Project

Project - A project (a collective endeavour of some kind).
Status: unstable
Disjoint With: Document Person

The `Project` class represents the class of things that are 'projects'. These may be formal or informal, collective or individual. It is often useful to indicate the `homepage` of a `Project`.

Further work is needed to specify the connections between this class and the FOAF properties `currentProject` and `pastProject`.

[#] [back to top]

Properties

Property: foaf:homepage

homepage - A homepage for some thing.
Status: stable
Domain: A thing
Range: Document
Inverse Functional Property

The `homepage` property relates something to a homepage about it.

Many kinds of things have homepages. FOAF allows a thing to have multiple homepages, but constrains `homepage` so that there can be only one thing that has any particular homepage.

A 'homepage' in this sense is a public Web document, typically but not necessarily available in HTML format. The page has as a topic the thing whose homepage it is. The homepage is usually controlled, edited or published by the thing whose homepage it is; as such one might look to a homepage for information on its owner from its owner. This works for people, companies, organisations etc.

The homepage property is a sub-property of the more general page property for relating a thing to a page about that thing. See also topic, the inverse of the page property.

[#] [back to top]

Property: foaf:made

made - Something that was made by this agent.

Status: stable
Domain: Agent
Range: A thing

The made property relates a Agent to something made by it. As such it is an inverse of the maker property, which relates a thing to something that made it. See made for more details on the relationship between these FOAF terms and related Dublin Core vocabulary.

[#] [back to top]

Property: foaf:maker

maker - An agent that made this thing.

Status: stable
Domain: A thing
Range: Agent

The maker property relates something to a Agent that made it. As such it is an inverse of the made property.

The name (or other rdfs:label) of the maker of something can be described as the dc:creator of that thing.

For example, if the thing named by the URI http://danbri.org/ has a maker that is a Person whose name is 'Dan Brickley', we can conclude that http://danbri.org/ has a dc:creator of 'Dan Brickley'.

FOAF descriptions are encouraged to use dc:creator only for simple textual names, and to use maker to indicate creators, rather than risk confusing creators with their names. This follows most Dublin Core usage. See UsingDublinCoreCreator for details.

[#] [back to top]

Property: foaf:mbox

personal mailbox - A personal mailbox, ie. an Internet mailbox associated with exactly one owner, the first owner of this mailbox. This is a 'static inverse functional property', in that there is (across time and change) at most one individual that ever has any particular value for foaf:mbox.

Status: stable
Domain: Agent
Range: A thing

Inverse Functional Property

The mbox property is a relationship between the owner of a mailbox and a mailbox. These are typically identified using the mailto: URI scheme (see RFC 2368).

Note that there are many mailboxes (eg. shared ones) which are not the mbox of anyone. Furthermore, a person can have multiple mbox properties.

In FOAF, we often see mbox used as an indirect way of identifying its owner. This works even if the mailbox is itself out of service (eg. 10 years old), since the property is defined in terms of its primary owner, and doesn't require the mailbox to actually be being used for anything.

Many people are wary of sharing information about their mailbox addresses in public. To

address such concerns whilst continuing the FOAF convention of indirectly identifying people by referring to widely known properties, FOAF also provides the `mbox_sha1sum` mechanism, which is a relationship between a person and the value you get from passing a mailbox URI to the SHA1 mathematical function.

[#] [back to top]

Property: foaf:member

member - Indicates a member of a Group
Status: stable
Domain: Group
Range: Agent

The `member` property relates a `Group` to a `Agent` that is a member of that group.

See `Group` for details and examples.

[#] [back to top]

Property: foaf:aimChatID

AIM chat ID - An AIM chat ID
Status: testing
Domain: Agent
Inverse Functional Property

The `aimChatID` property relates a `Agent` to a textual identifier ('screenname') assigned to them in the AOL Instant Messanger (AIM) system. See AOL's AIM site for more details of AIM and AIM screennames. The iChat tools from Apple also make use of AIM identifiers.

See `OnlineChatAccount` (and `OnlineAccount`) for a more general (and verbose) mechanism for describing IM and chat accounts.

[#] [back to top]

Property: foaf:currentProject

current project - A current project this person works on.
Status: testing
Domain: Person
Range: A thing

A `currentProject` relates a `Person` to a `Document` indicating some collaborative or individual undertaking. This relationship indicates that the `Person` has some active role in the project, such as development, coordination, or support.

When a `Person` is no longer involved with a project, or perhaps is inactive for some time, the relationship becomes a `pastProject`.

If the `Person` has stopped working on a project because it has been completed (successfully or otherwise), `pastProject` is applicable. In general, `currentProject` is used to indicate someone's current efforts (and implied interests, concerns etc.), while `pastProject` describes what they've previously been doing.

Note that this property requires further work. There has been confusion about whether it points to a thing (eg. something you've made; a homepage for a project, ie. a `Document` or to instances of the class `Project`, which might themselves have a `homepage`. In practice, it seems to have been used in a similar way to `interest`, referencing homepages of ongoing projects.

[#] [back to top]

Property: foaf:depiction

depiction - A depiction of some thing.

Status: testing
Domain: A thing
Range: Image

The depiction property is a relationship between a thing and an Image that depicts it. As such it is an inverse of the depicts relationship.

A common use of depiction (and depicts) is to indicate the contents of a digital image, for example the people or objects represented in an online photo gallery.

Extensions to this basic idea include 'Co-Depiction' (social networks as evidenced in photos), as well as richer photo metadata through the mechanism of using SVG paths to indicate the *regions* of an image which depict some particular thing. See 'Annotating Images With SVG' for tools and details.

The basic notion of 'depiction' could also be extended to deal with multimedia content (video clips, audio), or refined to deal with corner cases, such as pictures of pictures etc.

The depiction property is a super-property of the more specific property img, which is used more sparingly. You stand in a depiction relation to *any* Image that depicts you, whereas img is typically used to indicate a few images that are particularly representative.

[#] [back to top]

Property: foaf:depicts

depicts - A thing depicted in this representation.
Status: testing
Domain: Image
Range: A thing

The depicts property is a relationship between a Image and something that the image depicts. As such it is an inverse of the depiction relationship. See depiction for further notes.

[#] [back to top]

Property: foaf:familyName

familyName - The family name of some person.
Status: testing
Domain: Person

A number of naming constructs are under development to provide naming substructure; draft properties include firstName, givenName, and surname. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple name property.

[#] [back to top]

Property: foaf:firstName

firstName - The first name of a person.
Status: testing
Domain: Person

A number of naming constructs are under development to provide naming substructure; draft properties include firstName, givenName, and surname. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple name property.

[#] [back to top]

Property: foaf:geekcode

geekcode - A textual geekcode for this person, see http://www.geekcode.com/geek.html

Status: testing

Domain: Person

The geekcode property is used to represent a 'Geek Code' for some Person.

See the Code of the Geeks specification for details of the code, which provides a somewhat frivolous and willfully obscure mechanism for characterising technical expertise, interests and habits. The geekcode property is not bound to any particular version of the code. The last published version of the code was v3.12 in March 1996.

As the Geek Code website notes, the code played a small (but amusing) part in the history of the Internet. The geekcode property exists in acknowledgement of this history. It'll never be 1996 again.

Note that the Geek Code is a densely packed collections of claims about the person it applies to; to express these claims explicitly in RDF/XML would be incredibly verbose. The syntax of the Geek Code allows for '<' and '>' characters, which have special meaning in RDF/XML. Consequently these should be carefully escaped in markup.

An example Geek Code:

```
GED/J d-- s:++>: a-- C++(++++) ULU++ P+ L++ E---- W+(-) N+++ o+ K+++ w--- O- M+ V-- PS++>$ PE++>$ Y++ PGP++ t- 5+++ X++ R+++>$ tv+ b+ DI+++ D+++ G++++ e++ h r-- y++**
```

...would be written in FOAF RDF/XML as follows:

```
<foaf:geekcode> GED/J d-- s:++&gt;: a-- C++(++++) ULU++ P+ L++ E---- W+(-) N+++ o+ K+++ w--- O- M+ V-- PS++&gt;$ PE++&gt;$ Y++ PGP++ t- 5+++ X++ R+++&gt;$ tv+ b+ DI+++ D+++ G++++ e++ h r-- y++** </foaf:geekcode>
```

See also the geek code entry in everything2, which tells us that *the geek code originated in 1993; it was inspired (according to the inventor) by previous "bear", "smurf" and "twink" style-and-sexual-preference codes from lesbian and gay newsgroups*. There is also a Geek Code Decoder Page and a form-based generator.

[#] [back to top]

Property: foaf:gender

gender - The gender of this Agent (typically but not necessarily 'male' or 'female').

Status: testing

Domain: Agent

Functional Property

The gender property relates a Agent (typically a Person) to a string representing its gender. In most cases the value will be the string 'female' or 'male' (in lowercase without surrounding quotes or spaces). Like all FOAF properties, there is in general no requirement to use gender in any particular document or description. Values other than 'male' and 'female' may be used, but are not enumerated here. The gender mechanism is not intended to capture the full variety of biological, social and sexual concepts associated with the word 'gender'.

Anything that has a gender property will be some kind of Agent. However there are kinds of Agent to which the concept of gender isn't applicable (eg. a Group). FOAF does not currently include a class corresponding directly to "the type of thing that has a gender". At any point in time, a Agent has at most one value for gender. FOAF does not treat gender as a *static* property; the same individual may have different values for this property at different times.

Note that FOAF's notion of gender isn't defined biologically or anatomically - this would be tricky since we have a broad notion that applies to all Agents (including robots - eg. Bender from Futurama is 'male'). As stressed above, FOAF's notion of gender doesn't attempt to encompass the full range of concepts associated with human gender, biology and sexuality. As such it is a (perhaps awkward) compromise between the clinical and the social/psychological. In general, a person will be the best authority on their gender. Feedback on this design is particularly welcome (via the FOAF mailing list, foaf-dev). We have tried to be respectful of diversity without attempting to catalogue or enumerate that diversity.

This may also be a good point for a periodic reminder: as with all FOAF properties, documents that use 'gender' will on occassion be innacurate, misleading or outright false.

FOAF, like all open means of communication, supports *lying*. Application authors using FOAF data should always be cautious in their presentation of unverified information, but be particularly sensitive to issues and risks surrounding sex and gender (including privacy and personal safety concerns). Designers of FOAF-based user interfaces should be careful to allow users to omit `gender` when describing themselves and others, and to allow at least for values other than 'male' and 'female' as options. Users of information conveyed via FOAF (as via information conveyed through mobile phone text messages, email, Internet chat, HTML pages etc.) should be skeptical of unverified information.

[#] [back to top]

Property: foaf:givenName

Given name - The given name of some person.
Status: testing

A number of naming constructs are under development to provide naming substructure; draft properties include `firstName`, `givenName`, and `surname`. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple `name` property.

[#] [back to top]

Property: foaf:icqChatID

ICQ chat ID - An ICQ chat ID
Status: testing
Domain: Agent
Inverse Functional Property

The `icqChatID` property relates a `Agent` to a textual identifier assigned to them in the ICQ Chat system. See the icq chat site for more details of the 'icq' service. Their "What is ICQ?" document provides a basic overview, while their "About Us page notes that ICQ has been acquired by AOL. Despite the relationship with AOL, ICQ is at the time of writing maintained as a separate identity from the AIM brand (see `aimChatID`).

See `OnlineChatAccount` (and `OnlineAccount`) for a more general (and verbose) mechanism for describing IM and chat accounts.

[#] [back to top]

Property: foaf:img

image - An image that can be used to represent some thing (ie. those depictions which are particularly representative of something, eg. one's photo on a homepage).
Status: testing
Domain: Person
Range: Image

The `img` property relates a `Person` to a `Image` that represents them. Unlike its super-property `depiction`, we only use `img` when an image is particularly representative of some person. The analogy is with the image(s) that might appear on someone's homepage, rather than happen to appear somewhere in their photo album.

Unlike the more general `depiction` property (and its inverse, `depicts`), the `img` property is only used with representations of people (ie. instances of `Person`). So you can't use it to find pictures of cats, dogs etc. The basic idea is to have a term whose use is more restricted than `depiction` so we can have a useful way of picking out a reasonable image to represent someone. FOAF defines `img` as a sub-property of `depiction`, which means that the latter relationship is implied whenever two things are related by the former.

Note that `img` does not have any restrictions on the dimensions, colour depth, format etc of the `Image` it references.

Terminology: note that `img` is a property (ie. relationship), and that `code:Image` is a similarly named class (ie. category, a type of thing). It might have been more helpful to call `img` 'mugshot' or similar; instead it is named by analogy to the HTML IMG element.

[#] [back to top]

Property: foaf:interest

interest - A page about a topic of interest to this person.
Status: testing
Domain: Person
Range: Document

The `interest` property represents an interest of a `Agent`, through indicating a `Document` whose `topic`(s) broadly characterises that interest.

For example, we might claim that a person or group has an interest in RDF by saying they stand in a `interest` relationship to the RDF home page. Loosly, such RDF would be saying *"this agent is interested in the topic of this page"*.

Uses of `interest` include a variety of filtering and resource discovery applications. It could be used, for example, to help find answers to questions such as "Find me members of this organisation with an interest in XML who have also contributed to CPAN)".

This approach to characterising interests is intended to compliment other mechanisms (such as the use of controlled vocabulary). It allows us to use a widely known set of unique identifiers (Web page URIs) with minimal pre-coordination. Since URIs have a controlled syntax, this makes data merging much easier than the use of free-text characterisations of interest.

Note that interest does not imply expertise, and that this FOAF term provides no support for characterising levels of interest: passing fads and lifelong quests are both examples of someone's `interest`. Describing interests in full is a complex undertaking; `interest` provides one basic component of FOAF's approach to these problems.

[#] [back to top]

Property: foaf:isPrimaryTopicOf

is primary topic of - A document that this thing is the primary topic of.
Status: testing
Domain: A thing
Range: Document
Inverse Functional Property

The `isPrimaryTopicOf` property relates something to a document that is mainly about it.

The `isPrimaryTopicOf` property is *inverse functional*: for any document that is the value of this property, there is at most one thing in the world that is the primary topic of that document. This is useful, as it allows for data merging, as described in the documentation for its inverse, `primaryTopic`.

`page` is a super-property of `isPrimaryTopicOf`. The change of terminology between the two property names reflects the utility of 'primaryTopic' and its inverse when identifying things. Anything that has an `isPrimaryTopicOf` relation to some document X, also has a `page` relationship to it.

Note that `homepage`, is a sub-property of both `page` and `isPrimarySubjectOf`. The awkwardly named `isPrimarySubjectOf` is less specific, and can be used with any document that is primarily about the thing of interest (ie. not just on homepages).

[#] [back to top]

Property: foaf:jabberID

jabber ID - A jabber ID for something.
Status: testing
Domain: Agent
Inverse Functional Property

The `jabberID` property relates a `Agent` to a textual identifier assigned to them in the Jabber messaging system. See the Jabber site for more information about the Jabber protocols and

tools.

Jabber, unlike several other online messaging systems, is based on an open, publically documented protocol specification, and has a variety of open source implementations. Jabber IDs can be assigned to a variety of kinds of thing, including software 'bots', chat rooms etc. For the purposes of FOAF, these are all considered to be kinds of `Agent` (ie. things that *do* stuff). The uses of Jabber go beyond simple IM chat applications. The `jabberID` property is provided as a basic hook to help support RDF description of Jabber users and services.

See `OnlineChatAccount` (and `OnlineAccount`) for a more general (and verbose) mechanism for describing IM and chat accounts.

[#] [back to top]

Property: foaf:knows

knows - A person known by this person (indicating some level of reciprocated interaction between the parties).

Status: testing
Domain: Person
Range: Person

The `knows` property relates a `Person` to another `Person` that he or she knows.

We take a broad view of 'knows', but do require some form of reciprocated interaction (ie. stalkers need not apply). Since social attitudes and conventions on this topic vary greatly between communities, counties and cultures, it is not appropriate for FOAF to be overly-specific here.

If someone `knows` a person, it would be usual for the relation to be reciprocated. However this doesn't mean that there is any obligation for either party to publish FOAF describing this relationship. A `knows` relationship does not imply friendship, endorsement, or that a face-to-face meeting has taken place: phone, fax, email, and smoke signals are all perfectly acceptable ways of communicating with people you know.

You probably know hundreds of people, yet might only list a few in your public FOAF file. That's OK. Or you might list them all. It is perfectly fine to have a FOAF file and not list anyone else in it at all. This illustrates the Semantic Web principle of partial description: RDF documents rarely describe the entire picture. There is always more to be said, more information living elsewhere in the Web (or in our heads...).

Since `knows` is vague by design, it may be suprising that it has uses. Typically these involve combining other RDF properties. For example, an application might look at properties of each `weblog` that was `made` by someone you "`knows`". Or check the newsfeed of the online photo archive for each of these people, to show you recent photos taken by people you know.

To provide additional levels of representation beyond mere 'knows', FOAF applications can do several things.

They can use more precise relationships than `knows` to relate people to people. The original FOAF design included two of these ('knowsWell','friend') which we removed because they were somewhat *awkward* to actually use, bringing an inappopriate air of precision to an intrinsically vague concept. Other extensions have been proposed, including Eric Vitiello's Relationship module for FOAF.

In addition to using more specialised inter-personal relationship types (eg rel:acquaintanceOf etc) it is often just as good to use RDF descriptions of the states of affairs which imply particular kinds of relationship. So for example, two people who have the same value for their `workplaceHomepage` property are typically colleagues. We don't (currently) clutter FOAF up with these extra relationships, but the facts can be written in FOAF nevertheless. Similarly, if there exists a `Document` that has two people listed as its `maker`s, then they are probably collaborators of some kind. Or if two people appear in 100s of digital photos together, there's a good chance they're friends and/or colleagues.

So FOAF is quite pluralistic in its approach to representing relationships between people. FOAF is built on top of a general purpose machine language for representing relationships (ie. RDF), so is quite capable of representing any kinds of relationship we care to add. The problems are generally social rather than technical; deciding on appropriate ways of describing these interconnections is a subtle art.

Perhaps the most important use of `knows` is, alongside the `rdfs:seeAlso` property, to connect FOAF files together. Taken alone, a FOAF file is somewhat dull. But linked in with 1000s of other FOAF files it becomes more interesting, with each FOAF file saying a little more about people, places, documents, things... By mentioning other people (via `knows` or other

relationships), and by providing an `rdfs:seeAlso` link to their FOAF file, you can make it easy for FOAF indexing tools ('scutters') to find your FOAF and the FOAF of the people you've mentioned. And the FOAF of the people they mention, and so on. This makes it possible to build FOAF aggregators without the need for a centrally managed directory of FOAF files...

[#] [back to top]

Property: foaf:logo

logo - A logo representing some thing.
Status: testing
Domain: A thing
Range: A thing

The `logo` property is used to indicate a graphical logo of some kind. *It is probably underspecified...*

[#] [back to top]

Property: foaf:mbox_sha1sum

sha1sum of a personal mailbox URI name - The sha1sum of the URI of an Internet mailbox associated with exactly one owner, the first owner of the mailbox.
Status: testing
Domain: Agent
Inverse Functional Property

A `mbox_sha1sum` of a `Person` is a textual representation of the result of applying the SHA1 mathematical functional to a 'mailto:' identifier (URI) for an Internet mailbox that they stand in a `mbox` relationship to.

In other words, if you have a mailbox (`mbox`) but don't want to reveal its address, you can take that address and generate a `mbox_sha1sum` representation of it. Just as a `mbox` can be used as an indirect identifier for its owner, we can do the same with `mbox_sha1sum` since there is only one `Person` with any particular value for that property.

Many FOAF tools use `mbox_sha1sum` in preference to exposing mailbox information. This is usually for privacy and SPAM-avoidance reasons. Other relevant techniques include the use of PGP encryption (see Edd Dumbill's documentation) and the use of FOAF-based whitelists for mail filtering.

Code examples for SHA1 in C#, Java, PHP, Perl and Python can be found in Sam Ruby's weblog entry. Remember to include the 'mailto:' prefix, but no trailing whitespace, when computing a `mbox_sha1sum` property.

[#] [back to top]

Property: foaf:msnChatID

MSN chat ID - An MSN chat ID
Status: testing
Domain: Agent
Inverse Functional Property

The `msnChatID` property relates a `Agent` to a textual identifier assigned to them in the MSN online Chat system. See Microsoft's the MSN chat site for more details (or for a message saying *"MSN Chat is not currently compatible with your Internet browser and/or computer operating system"* if your computing platform is deemed unsuitable).

It is not currently clear how MSN chat IDs relate to the more general Microsoft Passport identifiers.

See `OnlineChatAccount` (and `OnlineAccount`) for a more general (and verbose) mechanism for describing IM and chat accounts.

[#] [back to top]

Property: foaf:myersBriggs

myersBriggs - A Myers Briggs (MBTI) personality classification.
Status: testing
Domain: Person

The myersBriggs property represents the Myers Briggs (MBTI) approach to personality taxonomy. It is included in FOAF as an example of a property that takes certain constrained values, and to give some additional detail to the FOAF files of those who choose to include it. The myersBriggs property applies only to the Person class; wherever you see it, you can infer it is being applied to a person.

The myersBriggs property is interesting in that it illustrates how FOAF can serve as a carrier for various kinds of information, without necessarily being commited to any associated worldview. Not everyone will find myersBriggs (or star signs, or blood types, or the four humours) a useful perspective on human behaviour and personality. The inclusion of a Myers Briggs property doesn't indicate that FOAF endorses the underlying theory, any more than the existence of weblog is an endorsement of soapboxes.

The values for myersBriggs are the following 16 4-letter textual codes: ESTJ, INFP, ESFP, INTJ, ESFJ, INTP, ENFP, ISTJ, ESTP, INFJ, ENFJ, ISTP, ENTJ, ISFP, ENTP, ISFJ. If multiple of these properties are applicable, they are represented by applying multiple properties to a person.

For further reading on MBTI, see various online sources (eg. this article). There are various online sites which offer quiz-based tools for determining a person's MBTI classification. The owners of the MBTI trademark have probably not approved of these.

This FOAF property suggests some interesting uses, some of which could perhaps be used to test the claims made by proponents of the MBTI (eg. an analysis of weblog postings filtered by MBTI type). However it should be noted that MBTI FOAF descriptions are self-selecting; MBTI categories may not be uniformly appealing to the people they describe. Further, there is probably a degree of cultural specificity implicit in the assumptions made by many questionaire-based MBTI tools; the MBTI system may not make sense in cultural settings beyond those it was created for.

See also Cory Caplinger's summary table or the RDFWeb article, FOAF Myers Briggs addition for further background and examples.

Note: Myers Briggs Type Indicator and MBTI are registered trademarks of Consulting Psychologists Press Inc. Oxford Psycholgists Press Ltd has exclusive rights to the trademark in the UK.

[#] [back to top]

Property: foaf:name

name - A name for some thing.
Status: testing
Domain: A thing

The name of something is a simple textual string.

XML language tagging may be used to indicate the language of the name. For example:

```
<foaf:name xml:lang="en">Dan Brickley</foaf:name>
```

FOAF provides some other naming constructs. While foaf:name does not explicitly represent name substructure (family vs given etc.) it does provide a basic level of interoperability. See the issue tracker for status of work on this issue.

The name property, like all RDF properties with a range of rdfs:Literal, may be used with XMLLiteral datatyped values (multiple names are acceptable whether they are in the same langauge or not). XMLLiteral usage is not yet widely adopted. Feedback on this aspect of the FOAF design is particularly welcomed.

[#] [back to top]

Property: foaf:nick

nickname - A short informal nickname characterising an agent (includes login identifiers, IRC

and other chat nicknames).
Status: testing

The nick property relates a Person to a short (often abbreviated) nickname, such as those use in IRC chat, online accounts, and computer logins.

This property is necessarily vague, because it does not indicate any particular naming control authority, and so cannot distinguish a person's login from their (possibly various) IRC nicknames or other similar identifiers. However it has some utility, since many people use the same string (or slight variants) across a variety of such environments.

For specific controlled sets of names (relating primarily to Instant Messanger accounts), FOAF provides some convenience properties: jabberID, aimChatID, msnChatID and icqChatID. Beyond this, the problem of representing such accounts is not peculiar to Instant Messaging, and it is not scaleable to attempt to enumerate each naming database as a distinct FOAF property. The OnlineAccount term (and supporting vocabulary) are provided as a more verbose and more expressive generalisation of these properties.

[#] [back to top]

Property: foaf:page

page - A page or document about this thing.
Status: testing
Domain: A thing
Range: Document

The page property relates a thing to a document about that thing.

As such it is an inverse of the topic property, which relates a document to a thing that the document is about.

[#] [back to top]

Property: foaf:pastProject

past project - A project this person has previously worked on.
Status: testing
Domain: Person
Range: A thing

After a Person is no longer involved with a currentProject, or has been inactive for some time, a pastProject relationship can be used. This indicates that the Person was involved with the described project at one point.

If the Person has stopped working on a project because it has been completed (successfully or otherwise), pastProject is applicable. In general, currentProject is used to indicate someone's current efforts (and implied interests, concerns etc.), while pastProject describes what they've previously been doing.

[#] [back to top]

Property: foaf:phone

phone - A phone, specified using fully qualified tel: URI scheme (refs: http://www.w3.org/Addressing/schemes.html#tel).
Status: testing

The phone of something is a phone, typically identified using the tel: URI scheme.

[#] [back to top]

Property: foaf:plan

plan - A .plan comment, in the tradition of finger and '.plan' files.
Status: testing
Domain: Person

The plan property provides a space for a Person to hold some arbitrary content that would appear in a traditional '.plan' file. The plan file was stored in a user's home directory on a UNIX machine, and displayed to people when the user was queried with the finger utility.

A plan file could contain anything. Typical uses included brief comments, thoughts, or remarks on what a person had been doing lately. Plan files were also prone to being witty or simply osbscure. Others may be more creative, writing any number of seemingly random compositions in their plan file for people to stumble upon.

See History of the Finger Protocol by Rajiv Shah for more on this piece of Internet history. The geekcode property may also be of interest.

[#] [back to top]

Property: foaf:primaryTopic

primary topic - The primary topic of some page or document.

Status: testing
Domain: Document
Range: A thing
Functional Property

The primaryTopic property relates a document to the main thing that the document is about.

The primaryTopic property is *functional*: for any document it applies to, it can have at most one value. This is useful, as it allows for data merging. In many cases it may be difficult for third parties to determine the primary topic of a document, but in a useful number of cases (eg. descriptions of movies, restaurants, politicians, ...) it should be reasonably obvious. Documents are very often the most authoritative source of information about their own primary topics, although this cannot be guaranteed since documents cannot be assumed to be accurate, honest etc.

It is an inverse of the isPrimaryTopicOf property, which relates a thing to a document *primarily* about that thing. The choice between these two properties is purely pragmatic. When describing documents, we use primaryTopic former to point to the things they're about. When describing things (people etc.), it is useful to be able to directly cite documents which have those things as their main topic - so we use isPrimaryTopicOf. In this way, Web sites such as Wikipedia or NNDB can provide indirect identification for the things they have descriptions of.

[#] [back to top]

Property: foaf:publications

publications - A link to the publications of this person.

Status: testing
Domain: Person
Range: Document

The publications property indicates a Document listing (primarily in human-readable form) some publications associated with the Person. Such documents are typically published alongside one's homepage.

[#] [back to top]

Property: foaf:schoolHomepage

schoolHomepage - A homepage of a school attended by the person.

Status: testing
Domain: Person
Range: Document

The schoolHomepage property relates a Person to a Document that is the homepage of a School that the person attended.

FOAF does not (currently) define a class for 'School' (if it did, it would probably be as a sub-class of Organization). The original application area for schoolHomepage was for 'schools' in the British-English sense; however American-English usage has dominated, and it is now perfectly reasonable to describe Universities, Colleges and post-graduate study using

schoolHomepage.

This very basic facility provides a basis for a low-cost, decentralised approach to classmate-reunion and suchlike. Instead of requiring a central database, we can use FOAF to express claims such as 'I studied *here*' simply by mentioning a school's homepage within FOAF files. Given the homepage of a school, it is easy for FOAF aggregators to lookup this property in search of people who attended that school.

[#] [back to top]

Property: foaf:surname

Surname - The surname of some person.
Status: testing
Domain: Person

A number of naming constructs are under development to provide naming substructure; draft properties include firstName, givenName, and surname. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple name property.

[#] [back to top]

Property: foaf:thumbnail

thumbnail - A derived thumbnail image.
Status: testing
Domain: Image
Range: Image

The thumbnail property is a relationship between a full-size Image and a smaller, representative Image that has been derrived from it.

It is typical in FOAF to express img and depiction relationships in terms of the larger, 'main' (in some sense) image, rather than its thumbnail(s). A thumbnail might be clipped or otherwise reduced such that it does not depict everything that the full image depicts. Therefore FOAF does not specify that a thumbnail depicts everything that the image it is derrived from depicts. However, FOAF does expect that anything depicted in the thumbnail will also be depicted in the source image.

A thumbnail is typically small enough that it can be loaded and viewed quickly before a viewer decides to download the larger version. They are often used in online photo gallery applications.

[#] [back to top]

Property: foaf:tipjar

tipjar - A tipjar document for this agent, describing means for payment and reward.
Status: testing
Domain: Agent
Range: Document

The tipjar property relates an Agent to a Document that describes some mechanisms for paying or otherwise rewarding that agent.

The tipjar property was created following discussions about simple, lightweight mechanisms that could be used to encourage rewards and payment for content exchanged online. An agent's tipjar page(s) could describe informal ("Send me a postcard!", "here's my book, music and movie wishlist") or formal (machine-readable micropayment information) information about how that agent can be paid or rewarded. The reward is not associated with any particular action or content from the agent concerned. A link to a service such as PayPal is the sort of thing we might expect to find in a tipjar document.

Note that the value of a tipjar property is just a document (which can include anchors into HTML pages). We expect, but do not currently specify, that this will evolve into a hook for finding more machine-readable information to support payments, rewards. The OnlineAccount machinery is also relevant, although the information requirements for

automating payments are not currently clear.

[#] [back to top]

Property: foaf:topic

topic - A topic of some page or document.
Status: testing
Domain: Document
Range: A thing

The `topic` property relates a document to a thing that the document is about.

As such it is an inverse of the `page` property, which relates a thing to a document about that thing.

[#] [back to top]

Property: foaf:topic_interest

interest_topic - A thing of interest to this person.
Status: testing
Domain: Person
Range: A thing

The `topic_interest` property is generally found to be confusing and ill-defined and is a candidate for removal. The goal was to be link a person to some thing that is a topic of their interests (rather than, per `interest` to a page that is about such a topic).

[#] [back to top]

Property: foaf:weblog

weblog - A weblog of some thing (whether person, group, company etc.).
Status: testing
Domain: Agent
Range: Document
Inverse Functional Property

The `weblog` property relates a `Agent` to a weblog of that agent.

[#] [back to top]

Property: foaf:workInfoHomepage

work info homepage - A work info homepage of some person; a page about their work for some organization.
Status: testing
Domain: Person
Range: Document

The `workInfoHomepage` of a `Person` is a `Document` that describes their work. It is generally (but not necessarily) a different document from their `homepage`, and from any `workplaceHomepage`(s) they may have.

The purpose of this property is to distinguish those pages you often see, which describe someone's professional role within an organisation or project. These aren't really homepages, although they share some characterstics.

[#] [back to top]

Property: foaf:workplaceHomepage

workplace homepage - A workplace homepage of some person; the homepage of an

organization they work for.

Status: testing
Domain: Person
Range: Document

The workplaceHomepage of a Person is a Document that is the homepage of a Organization that they work for.

By directly relating people to the homepages of their workplace, we have a simple convention that takes advantage of a set of widely known identifiers, while taking care not to confuse the things those identifiers identify (ie. organizational homepages) with the actual organizations those homepages describe.

For example, Dan Brickley works at W3C. Dan is a Person with a homepage of http://danbri.org/; W3C is a Organization with a homepage of http://www.w3.org/. This allows us to say that Dan has a workplaceHomepage of http://www.w3.org/.

```
<foaf:Person>
 <foaf:name>Dan Brickley</foaf:name>
 <foaf:workplaceHomepage rdf:resource="http://www.w3.org/"/>
</foaf:Person>
```

Note that several other FOAF properties work this way; schoolHomepage is the most similar. In general, FOAF often indirectly identifies things via Web page identifiers where possible, since these identifiers are widely used and known. FOAF does not currently have a term for the name of the relation (eg. "workplace") that holds between a Person and an Organization that they work for.

[#] [back to top]

Property: foaf:yahooChatID

Yahoo chat ID - A Yahoo chat ID
Status: testing
Domain: Agent
Inverse Functional Property

The yahooChatID property relates a Agent to a textual identifier assigned to them in the Yahoo online Chat system. See Yahoo's the Yahoo! Chat site for more details of their service. Yahoo chat IDs are also used across several other Yahoo services, including email and Yahoo! Groups.

See OnlineChatAccount (and OnlineAccount) for a more general (and verbose) mechanism for describing IM and chat accounts.

[#] [back to top]

Property: foaf:account

holds account - Indicates an account held by this agent.
Status: unstable
Domain: Agent
Range: Online Account

The account property relates a Agent to an OnlineAccount for which they are the sole account holder. See OnlineAccount for usage details.

[#] [back to top]

Property: foaf:accountName

account name - Indicates the name (identifier) associated with this online account.
Status: unstable
Domain: Online Account

The accountName property of a OnlineAccount is a textual representation of the account name (unique ID) associated with that account.

[#] [back to top]

Property: foaf:accountServiceHomepage

account service homepage - Indicates a homepage of the service provide for this online account.

Status: unstable
Domain: Online Account
Range: Document

The `accountServiceHomepage` property indicates a relationship between a `OnlineAccount` and the homepage of the supporting service provider.

[#] [back to top]

Property: foaf:age

age - The age in years of some agent.

Status: unstable
Domain: Agent
Functional Property

The `age` property is a relationship between a `Agent` and a string representing the year in which they were born (Gregorian calendar). See also `birthday`.

[#] [back to top]

Property: foaf:based_near

based near - A location that something is based near, for some broadly human notion of near.

Status: unstable
Domain: Spatial Thing
Range: Spatial Thing

The `based_near` relationship relates two "spatial things" (anything that can *be somewhere*), the latter typically described using the geo:lat / geo:long geo-positioning vocabulary (See GeoInfo in the W3C semweb wiki for details). This allows us to say describe the typical latitute and longitude of, say, a Person (people are spatial things - they can be places) without implying that a precise location has been given.

We do not say much about what 'near' means in this context; it is a 'rough and ready' concept. For a more precise treatment, see GeoOnion vocab design discussions, which are aiming to produce a more sophisticated vocabulary for such purposes.

FOAF files often make use of the `contact:nearestAirport` property. This illustrates the distinction between FOAF documents (which may make claims using *any* RDF vocabulary) and the core FOAF vocabulary defined by this specification. For further reading on the use of `nearestAirport` see UsingContactNearestAirport in the FOAF wiki.

[#] [back to top]

Property: foaf:birthday

birthday - The birthday of this Agent, represented in mm-dd string form, eg. '12-31'.

Status: unstable
Domain: Agent
Functional Property

The `birthday` property is a relationship between a `Agent` and a string representing the month and day in which they were born (Gregorian calendar). See BirthdayIssue for details of related properties that can be used to describe such things in more flexible ways.

[#] [back to top]

Property: foaf:dnaChecksum

DNA checksum - A checksum for the DNA of some thing. Joke.
Status: unstable

The dnaChecksum property is mostly a joke, but also a reminder that there will be lots of different identifying properties for people, some of which we might find disturbing.

[#] [back to top]

Property: foaf:membershipClass

membershipClass - Indicates the class of individuals that are a member of a Group
Status: unstable

The membershipClass property relates a Group to an RDF class representing a sub-class of Agent whose instances are all the agents that are a member of the Group.

See Group for details and examples.

[#] [back to top]

Property: foaf:openid

openid - An OpenID for an Agent.
Status: unstable
Domain: Agent
Range: Document
Inverse Functional Property

A openid is a property of a Agent that associates it with a document that can be used as an indirect identifier in the manner of the OpenID "Identity URL". As the OpenID 1.1 specification notes, OpenID itself"*does not provide any mechanism to exchange profile information, though Consumers of an Identity can learn more about an End User from any public, semantically interesting documents linked thereunder (FOAF, RSS, Atom, vCARD, etc.)*". In this way, FOAF and OpenID complement each other; neither provides a stand-alone approach to online "trust", but combined they can address interesting parts of this larger problem space.

The openid property is "inverse functional", meaning that anything that is the foaf:openid of something, is the openid of no more than one thing. FOAF is agnostic as to whether there are (according to the relevant OpenID specifications) OpenID URIs that are equally associated with multiple Agents. FOAF offers sub-classes of Agent, ie. Organization and Group, that allow for such scenarios to be consistent with the notion that any foaf:openid is the foaf:openid of just one Agent.

FOAF does not mandate any particular URI scheme for use as openid values. The OpenID 1.1 specification includes a delegation model that is often used to allow a weblog or homepage document to also serve in OpenID authentication via "link rel" HTML markup. This deployment model provides a convenient connection to FOAF, since a similar technique is used for FOAF autodiscovery in HTML. A single document can, for example, serve both as a homepage and an OpenID identity URL.

[#] [back to top]

Property: foaf:sha1

sha1sum (hex) - A sha1sum hash, in hex.
Status: unstable
Domain: Document

The sha1 property relates a Document to the textual form of a SHA1 hash of (some representation of) its contents.

The design for this property is neither complete nor coherent. The Document class is currently used in a way that allows multiple instances at different URIs to have the 'same' contents (and hence hash). If sha1 is an owl:InverseFunctionalProperty, we could deduce

that several such documents were the self-same thing. A more careful design is needed, which distinguishes documents in a broad sense from byte sequences.

[#] [back to top]

Property: foaf:title

title - Title (Mr, Mrs, Ms, Dr. etc)
Status: unstable

The property is a candidate for replacement by a new term, honorificPrefix to follow Portable Contacts usage. If this happens, foaf:title may be marked as archaic.

The approriate values for `title` are not formally constrained, and will vary across community and context. Values such as 'Mr', 'Mrs', 'Ms', 'Dr' etc. are expected.

[#] [back to top]

Property: foaf:family_name

family_name - The family name of some person.
Status: archaic
Domain: Person

This property has been deprecated and is now *archaic usage*. Do not use; use `familyName` instead.

A number of naming constructs are under development to provide naming substructure; draft properties include `firstName`, `givenName`, and `surname`. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple `name` property.

[#] [back to top]

Property: foaf:fundedBy

funded by - An organization funding a project or person.
Status: archaic
Domain: A thing
Range: A thing

This property has been deprecated and is now *archaic usage*. Do not use.

The `fundedBy` property relates something to something else that has provided funding for it.

This property is under-specified, experimental, and should be considered liable to change.

[#] [back to top]

Property: foaf:givenname

Given name - The given name of some person.
Status: archaic

A number of naming constructs are under development to provide naming substructure; draft properties include `firstName`, `givenName`, and `surname`. These are not currently stable or consistent; see the issue tracker for design discussions, status and ongoing work on rationalising the FOAF naming machinery.

There is also a simple `name` property.

[#] [back to top]

Property: foaf:holdsAccount

holds account - Indicates an account held by this agent.
Status: archaic
Domain: Agent
Range: Online Account

This property has been deprecated and is now *archaic usage*. Do not use. Use `account` instead.

The `holdsAccount` property relates a `Agent` to an `OnlineAccount` for which they are the sole account holder. See `OnlineAccount` for usage details.

This property is equivalent to the `account` property, which was introduced primarily to provide simpler naming for the same idea.

[#] [back to top]

Property: foaf:theme

theme - A theme.
Status: archaic
Domain: A thing
Range: A thing

This property has been deprecated and is now *archaic usage*. Do not use.

The `theme` property is rarely used and under-specified. The intention was to use it to characterise interest / themes associated with projects and groups. Further work is needed to meet these goals.

[#] [back to top]

External Vocabulary References

The description of the terms in the FOAF 'dictionary' often make reference to classes and properties elsewhere. This section of the FOAF specification provides a placeholder reference for any FOAF mention of externally defined terms. For example, sometimes we might say that FOAF property has a domain or range of an externally defined class, or that a FOAF class is a sub-class of an external class, or 'disjoint with' such a class (ie. has no common members). Such claims help fix the intended meaning of FOAF terms in relationship to other 'peer' vocabularies.

Status Vocabulary

Each term in FOAF is annotated with properties from the SemWeb Vocab Status Ontology

This was created as an experiment in documenting FOAF's term-centric versioning model, in which a common fixed namespace URI is used, while term definitions slowly and independently evolve through different stability levels. This contrasts with other approaches to versioning which attach versioning information to larger sets of terms.

Note that this mechanism is itself experimental and, in theory at least, 'unstable'. The definitions of 'stable', 'unstable' and 'testing' cannot be defined as global absolutes, but only in relationship to the practices, expectations and social structures around some vocabulary. For their use in FOAF, future versions of this specification could usefully offer more detail about what to expect from a term labelled 'stable'.

vs:term_status
The vs:term_status property indicates the status of a vocabulary term, one of 'stable','unstable','testing'.

W3C Basic Geo (WGS84 lat/long) Vocabulary

Members of the FOAF and W3C Semantic Web Interest Group communities collaborated in 2003 to create a very simple vocabulary that described points in geographic space. This is the W3CBasic Geo Vocabulary. It assumes use of the WGS84 reference system and defines properties geo:lat, geo:long and geo:alt in terms of a class geo:SpatialThing.

The foaf:based_near property relates a spatial thing (typically a foaf:Agent of some kind) to

another spatial thing, which can be described using geo:lat, geo:long etc.

RDF Vocabulary Description - core concepts

The FOAF dictionary of terms is defined using a family of W3C standards: RDF, RDF Schema and OWL. These share a data model and general approach, and provide for increasing levels of expressivity. Here we introduce the core OWL and RDF/S terms used directly in the machine-readable description of FOAF. See W3C's site for the latest and most authoritative OWL and RDF specifications.

FOAF is based on the exchange of free-form descriptions that are structured in terms of things having properties, where the value of each property is expressed as either textually (eg. a name or number), or by reference to another thing. FOAF (as an application of RDF) uses URI identifiers wherever possible to talk about things of interest, whether they are Web pages, classes of thing, properties of things, or even people. See the W3C Web Architecture specification for more background on URIs.

From core RDF, FOAF takes the notion that we are talking about things, and they fall into categories; we call these 'classes'. The core machinery we use from the RDF Schema and OWL technologies simply give us some built-in terminology for talking about things, classes and properties. Here we introduce some of these and discuss briefly how they relate to FOAF's approach to describing things.

owl:Thing
: OWL introduces the class 'Thing' as a name for the universal class of all things. This is sometimes useful when we want to express universality of property use, eg. that anything can be the value of foaf:depicts.

rdf:Property
: RDF has a built-in class called rdf:Property. This is the class of all things like foaf:homepage or dc:creator which define named kinds of relationship between pairs of things, or between things and textually-expressed information.

owl:DatatypeProperty
: The OWL specifications give a name for those properties whose values are textually-expressed: "DataTypeProperty". RDF allows these to be either plain literal values (these can also carry an indicator of their language, via xml:lang), or else "data-typed", which means they are marked with a URI indicating their type (but no language tagging).

owl:ObjectProperty
: ObjectProperty is OWL's name for those properties which are not textually-expressed; instead, they are used when mentioning or referring to some other thing. OWL encourages vocabularies to avoid using a single named property in both 'ObjectProperty' and 'DataTypeProperty' styles. However earlier usage, notably in the Dublin Core community, does just this. Each FOAF property is either an Object Property or DataType Property.

rdf:type
: One of the most commonly used built-in relationships in RDF is 'type'. RDF type relates something to a class that it is in.

rdfs:subClassOf
: RDFS gives a name for the relationship between some specific class and its more general superclass: 'subClassOf'. It would have perhaps been simpler if this was called 'superProperty'. So for example we say that foaf:Person has a subPropertyOf property whose value is foaf:Agent.

rdfs:Class
: RDFS gives the name 'Class' to those things that represent classes of thing, ie. which are values of rdf:type for their members. The OWL language also (for technical reasons) defines owl:Class for essentially the same notion. OWL also includes powerful machinery for defining the membership rules for classes. This is not heavily used in FOAF, beyond the experimental mechanisms associated with foaf:Group.

rdfs:subClassOf
: RDFS gives a name for the relationship between some specific class and its more general superclass: 'subClassOf'. It would have perhaps been simpler if this was called 'superProperty'. So for example we say that foaf:Person has a subPropertyOf property whose value is foaf:Agent.

rdfs:subPropertyOf
: Similar to subClassOf but for hierarchies of properties, we can use rdfs:subPropertyOf to point to a more general super-property, for example we say foaf:aimChatID rdfs:subPropertyOf foaf:nick.

rdfs:domain
: The RDFS specification introduced the notion of a property's domain. This is a way of saying, for some property, something about the kind of classes it is used with. If you know the domain or domains for some property, you know that whenever you see that property applied to something, then that thing ought to be a member of those classes. Note that this does not mean that every description using the property is compelled to mention all those classes, just that the meaning of the property implies also the type of the thing the property is applied to.

rdfs:range
: RDFS also defines a property of properties called 'range'; this works just like rdfs:domain, except for the values of a property. If you know the range of some property, you know what kinds of thing are reasonable values for it.

owl:FunctionalProperty
: OWL provides even more useful information about properties, such as the ability to say that a property is 'functional'. This means simply that for any particular thing, you can expect at most one value for that property. It is simplest to think of this as contextualised to any given time; although OWL doesn't talk about time explicitly. So we might say that 'age' is functional, even though a series of FOAF documents might be published, each truthfully giving different values for my 'age' which made sense in their original context. At the time of writing, the only W3C technology that can take a larger perspective on such different perspectives / views (or 'graphs') is SPARQL. If you have two different values for a given functional property, you know you have a problem; perhaps one is out-of-date, for example. Or perhaps they only differ in trivial detail (eg. date syntax, whitespace).

owl:InverseFunctionalProperty
: OWL also gives a name for properties where common values tell us something about the identity of the thing having the property. On the Web this can be very useful. OWL tells us that two descriptions are of the same thing, if they include truthful mention of some 'inverse functional property' that has the same value. The classic FOAF example could be two mentions of a person having some particular foaf:homepage. This OWL construct is very useful for reasoning about identity and merging scattered and partial descriptions.

owl:inverseOf
: OWL provides a property 'inverseOf' that holds between inverse properties; for example, any two things related by foaf:maker are related in the reverse direction by foaf:made.

owl:disjointWith
: OWL also lets us indicate that two classes have no common members. This can be useful for clarifying modelling assumptions in a language-neutral manner; eg. we might ask whether anything can be both a foaf:Group and a foaf:Organization simultaneously.

Dublin Core terms

The Dublin Core specification provides term definitions that focus on issues of resource discovery, document description and related concepts useful for cultural heritage and digital library applications. FOAF can be used alongside any variants of Dublin Core, but works most effectively with the most modern Dublin Core terms namespace. Note that here we use the prefix 'dct:' to stand for the DC Terms namespace; however it is not unusual to see 'dc' also used.

dct:Agent
: Dublin Core's notion of Agent is much like FOAF's; Dublin Core says "A resource that acts or has the power to act.", we say "things that do stuff". As nobody has provided a counter-example of something fitting one definition but not the other, we say here that foaf:Agent stands in an 'equivalent class' relationship to dct:Agent (and vice-versa).

dct:creator
: The notion of 'creator' in the latest versions of Dublin Core matches FOAF's notion of 'maker'; based on their definitions, every pair of things that are related by one of those properties are also related by the other. We express this by saying that these properties stand in an 'equivalent property' relationship' to one another.

Wordnet terms

Earlier versions of this specification used an experimental companion namespace produced from the lexical database Wordnet (v1.6). This is currently offline, and corresponding sub-class relationships have been ommited from the FOAF documentation. More recent RDF representations of Wordnet now exist, however they don't map Wordnet synsets to classes, so can't be directly used here. Future versions of this specification might restore links to some version of Wordnet in RDF.

SIOC terms

Many terms in the SIOC vocabulary are defined with reference to FOAF. See the SIOC project for details. Future versions of this specification may provide more information here.

Acknowledgments

There are far too many people who have contributed to the FOAF project to name everyone in this early-release of the new improved spec. FOAF wouldn't be such a fun project or be as widely known as it is today without the efforts, enthusiasm and intelligence of the folks who have contributed via the rdfweb-dev list, #foaf IRC channel, and FoafProject wiki site.

That said, a few milestones in FOAF's history should be mentioned. We owe particular thanks to Edd Dumbill for his IBM developerWorks articles (which attracted the affections of the Weblogging crowd) and for his Foafbot application whose evolution those articles have tracked. Also Morten Frederiksen's FoafExplorer, Daniel Krech's Web View aggregator, Jim Ley and Liz Turner's work on FOAFNaut, which alongside FOAFbot, ^have been instrumental in showing how FOAF data can be collected and used. Meanwhile Leigh Dodd's foaf-a-matic has been the data creation tool that has been most people's gateway to FOAFdom. FOAF also owes a lot to the folks at Ecademy, TypePad and elsewhere for showing how end users can share FOAF self-descriptions on the Web without ever seeing a line of XML syntax. Jo Walsh has enthused many about hooking FOAF up to Geo and mapping data, as has Matt Biddulph by explaining the

workings of his FOAF harvesting and image metadata tools. FOAF has also benefited greatly from documentation contributed in non-English languages, many thanks to all contributors of translations (foaf-a-matic and other docs). FOAF is now arguably better documented in Japanese and Spanish than in English, thanks to Masahide Kanzaki and Leandro Mariano Lopez (inkel) respectively. Thanks also to Chris Schmidt for fixing up the spec generation tool (now a Python/Redland script), as well as for contributing numerous cool hacks to the FOAF community. To Richard Cyganiak and others in IRC for (amongst much else) help debugging Apache configurations. To Ian Davis for his wonderful FOAF Logo. And last but not least, Marc Canter is in a class of his own. Thanks all, and to those who aren't listed here yet, but who made a difference...

This brief survey only scratches the surface of a growing body of work. Sincere thanks to all who have contributed tools, documentation, brain cells and enthusiasm to this project. We should also mention that FOAF would not be possible without the collaborative and opensource efforts of the RDF developer community, both in terms of idea sharing (#swig etc) and freely available tools (Jena, Redland, RDFlib, Cwm, Sesame, 3store etc).

Recent Changes

2009-12-15

- terms: foaf:Agent owl:equivalentClass with Dublin Core terms namespace 'Agent'; similarly foaf:maker and 'creator' from DC. These use the most modern http://purl.org/dc/terms/ namespace
- editorial: added note about the use of 'archaic'
- editorial: added a paragraph on changes in this version
- terms: family_name marked as archaic; replaced with familyName
- terms: marked fundedBy as archaic
- terms: marked givenname as archaic; replaced with givenName
- terms: marked holdAccount as archaic; replaced with account
- terms: changed myersBriggs from incorrect objectProperty to datatypeProperty
- terms: added openId into the rdf/xml version (it was present in the dated version but not in index.rdf)
- terms: marked theme as archaic
- editorial: fixed outdated references to RDF/XML
- editorial: updated acknowledgements
- terms: added text about terms used in foaf not in the foaf namespace
- editorial: fixed links and stray references to rdfweb-dev and #rdfig
- editorial: fixed rdfweb.org blog links (now pointing at archive.org where possible)
- editorial: fixed links to the wiki (stray /w/)
- editorial: fixed internal link to sec-glance
- editorial: fixed copyright date to 2009

2007-11-02

- terms: added foaf:openid
- editorial: updated links to the FOAF wiki (which now supports OpenID for logins)
- editorial: spec now uses RDFa DTD declaration.

2007-05-24

- terms: Organization, Group, member, made, maker are now stable.
- editorial: Table of contents represents actual rather than desired structure. links fixed.
- process: Clarified that the specification version and URL are not 0.1; publishing at /foaf/spec
- editorial: Noted that the namespace URI will not change. Given the spec a version (0.9)
- editorial: CSS tweaks (grey boxes for terms), margins, justification
- editorial: Removed various TODOs and editorial comments
- editorial: Updated text for Organization to indicate relationship with Group
- web: Web server configuration now serves 303 redirects for terms, and RDF at the namespace URI if application/rdf+xml requested
- editorial: Updated mentions of rdfig to be swig, rdfweb-dev to be foaf-dev; fixed links accordingly
- editorial: Removed link to validator, since we have RDF inside and the dtd-based validator doesn't like mixed-namespace documents.
- web: created /foaf/spec for specification documents (and archive)
- editorial: Re-organized boilerplate section slightly (spec has url for current version).
- editorial: Updated per-term documentation to link to wiki issue tracker instead of the old bugzilla installation.
- editorial: Updated foaf:Group documentation to mention possiblility of SPARQL
- editorial: Updated foaf:name documentation to mention possibility of XMLLiteral datatyped values
- editorial: Fixed or removed various bad links (ichat, python, internal links)
- editorial: Reworded copyright statement to clarify intent; we used to say "and does not apply to FOAF data formats, vocabulary terms, or technology.", which gave impression we were

holding stuff back. Rather, this is just our understanding of copyright. The underlying technology part is completely supplied by W3C's specifications.
- web: added "rel='alternate'" to document header to aid RDF discovery

Previous Changes

- 2004-09-01: dropped the claim that foaf:homepage has an rdfs:domain of foaf:Agent, to allow other kinds of things to be said to have homepages (eg. Events)