

Diploma Thesis

February 9, 2007

PAX

A Semantically Annotated Publication Database
Using Ajax

Michael Hermann

of Wattwil SG, Switzerland (02-713-261)

supervised by

Prof. Dr. Harald Gall

Dr. Gerald Reif



University of Zurich
Department of Informatics



Diploma Thesis

PAX

A Semantically Annotated Publication Database
Using Ajax

Michael Hermann



University of Zurich
Department of Informatics



Diploma Thesis

Author: Michael Hermann, hermann.michael@gmail.com

Project period: 14.08.2006 - 14.02.2007

Software Evolution & Architecture Lab
Department of Informatics, University of Zurich

Abstract

Web applications for managing scientific publications on an institute's, research group's, and researcher's homepage can hardly be found. This thesis presents the implementation of such a Web Application called PAX. It was developed according to the requirements of the "Department of Informatics" at the University of Zurich. PAX makes use of Ajax for providing a user friendly interface, supporting drag and drop as well as dynamic changes of the Web pages. This includes showing and hiding parts of the page depending on user interaction, as well as manipulating the underlying database by adding, sorting and removing parts of the Web page. It handles the import of BibTeX files as well as the export formats BibTeX and CSV. Furthermore, PAX provides a simple user management. The semantic annotation, which is available in RDF and RDFa, follows the SWRC-unizh ontology, which is an extension of the SWRC ontology. An approach for incorporating the generation of semantic data into a Web application is presented. The new Web application aims to enlarge the amount of semantic data available in the Web and allows future research in the area of the Semantic Web.

Zusammenfassung

Es existieren nur wenige Web-Anwendungen um wissenschaftliche Publikationen auf der Homepage eines Institutes, einer Forschungsgruppe oder eines Forschers zu verwalten. Diese Diplomarbeit präsentiert die Implementation einer solchen Web-Applikation namens PAX. Sie wurde gemäss den Anforderungen des "Institut für Informatik der Universität Zürich" entwickelt. PAX benutzt Ajax um eine benutzerfreundliche Oberfläche anzubieten, welche sowohl Drag-and-Drop als auch dynamische Änderungen der Webseite unterstützt. Die schliesst einerseits das Ein- und Ausblenden von Teilen der Seite ein, abhängig von der Interaktion des Benutzers, andererseits die Manipulation der Datenbank durch Hinzufügen, Sortieren und Löschen von Teilen der Webseite. Es werden der Import von BibTex-Dateien und der Export in den Formaten BibTeX und CSV unterstützt. Zudem stellt PAX eine simple Benutzerverwaltung bereit. Die semantische Annotation, welche in RDF und RDFa verfügbar ist, befolgt die SWRC-unizh Ontologie, welche eine Erweiterung der SWRC-Ontologie ist. Es wird ein Ansatz präsentiert die Generierung von semantischen Daten in eine Web-Anwendung einzubauen. Die so entstandene Web-Applikation zielt darauf, die Menge der vorhandenen semantischen Daten im Web zu vergrössern und macht weitere Forschung im Bereich des Semantic Web möglich.

Contents

1	Introduction	1
2	Recent Web Technologies	3
2.1	Web 2.0 and Ajax	3
2.1.1	Web 2.0	3
2.1.2	Ajax - Asynchronous Javascript And XML	4
2.1.3	XMLHttpRequest Object	6
2.1.4	Ajax Frameworks	7
2.2	Semantic Web	9
2.2.1	Shortcomings of the WWW	9
2.2.2	Resource Description Framework - RDF	10
2.2.3	RDF Schema	10
2.2.4	Web Ontology Language - OWL	11
2.2.5	Embedding RDF in XHTML	12
2.2.6	The Current State of the Semantic Web	14
2.2.7	Semantic Web Outlook	14
3	Review of the State of the Art	15
3.1	Basilic - Bibliography Server for Research Laboratories	15
3.2	AIFB SEmantic PortAL	16
3.3	Connotea	16
3.4	BibSonomy	17
3.5	Discussion	18
4	Requirements	19
4.1	General Requirements	19
4.2	Roles	19
4.3	Function Range	20
4.3.1	Publications	20
4.3.2	Projects	22
4.3.3	Publishers	23
4.3.4	Partners/Funding Organizations	23
4.3.5	Research Groups	24
4.3.6	The Relation Between Employees and Authors/Editors	24
4.3.7	Employees	25
4.3.8	Authors/Editors	25
4.4	Output Formats	26
4.5	User Interface	26

4.5.1	Type Ahead	26
4.5.2	Creating, Sorting, and Removing Objects	27
4.6	Summary	27
5	The SWRC-unizh Ontology	29
5.1	SWRC - Semantic Web for Research Communities	29
5.2	The Modified SWRC Ontology	31
6	Implementation	33
6.1	Modules and Views	34
6.2	Filtering	35
6.3	User Management and Credentials	35
6.3.1	Storing Passwords	35
6.3.2	Login and Authentication	35
6.3.3	Access Restriction	35
6.3.4	Logout	36
6.3.5	Changing Passwords	36
6.3.6	Resetting Passwords	36
6.3.7	Mail	36
6.4	Output Formats	36
6.4.1	Bibtex Generation	36
6.4.2	RDF/OWL Generation	37
6.4.3	RDF Inclusion via Link	38
6.4.4	RDFa generation	39
6.4.5	CSV generation	40
6.5	Managing PDF Files	40
6.5.1	PDF Upload and Download	40
6.6	Managing Technical Reports	41
6.6.1	PDF Modification	41
6.7	BibTeX Importer	42
6.8	Ajax User Interface	43
6.8.1	Login/Sign out	43
6.8.2	Mandatory and Optional BibTeX fields	45
6.8.3	Managing Custom BibTeX fields	45
6.8.4	Type Ahead	46
6.8.5	Adding Elements	46
6.8.6	Creating New Elements	47
6.8.7	Removing Elements	47
6.8.8	Sorting Elements	48
6.8.9	Dependent Drop-Down Menu	48
6.9	Handling Import Errors	49
6.10	Data Model	49
7	Evaluation	51
7.1	Import	51
7.2	Modules	52
7.3	Output Formats	57

8	Conclusions	59
8.1	Conclusions	59
8.2	Summary of Contributions	60
8.3	Future Research	60
A	BibTeX Fields	63
B	Umlaut Encoding	65
C	The Underlying Data Model	67

List of Figures

2.1	Zimbra groupware client	4
2.2	The traditional model for Web applications compared to the Ajax model	5
2.3	Graphical representation of a triple	10
3.1	Adding a publication in Basilic	16
3.2	Bibsonomy (user view)	17
4.1	The relation between employees and authors/editors	24
4.2	Required classes for the publication database	27
5.1	The SWRC top level concepts	29
5.2	The SWRC publication classes	30
5.3	The SWRC organization classes	30
5.4	The SWRC person classes	31
5.5	The SWRC project classes	31
5.6	Extension of the SWRC project class	31
6.1	The architecture of PAX	33
6.2	Input mask for articles	44
6.3	Mandatory and optional BibTeX fields for manuals	45
6.4	Adding custom BibTeX fields	45
6.5	Type ahead in the author field	46
6.6	Adding a new author to a publication	46
6.7	Creating an external author	47
6.8	Sorting authors	48
6.9	Dependent drop-down menu	48
6.10	Import error	49
7.1	Options for BibTeX import	51
7.2	The result of an import	52
7.3	Publication list	52
7.4	Filtered Publication list	53
7.5	Show a single publication	53
7.6	Editing a publication	54
7.7	Project list including RDFa	54
7.8	Publisher list	55
7.9	Partner list	55
7.10	Group list	56
7.11	Employee list	56
7.12	Author/Editor list	57
C.1	The data model used in PAX	67

List of Tables

2.1	The most popular Ajax frameworks and libraries	8
3.1	Available solutions for managing publications in a database	15

B.1 Umlauts handled by PAX	65
--------------------------------------	----

List of Listings

6.1 Example of BibTeX output in PAX	36
6.2 The LaTeX template for PDF modification in PAX	42

Chapter 1

Introduction

"Fifteen years ago, one of the hardest things to do was not to develop the initial version of HTTP, or to create a browser that was also an editor, or even to get approval for the purchase of equipment (!). The difficult thing was to convince people that the Web was something they should adopt." Tim Berners-Lee [Upd05]

Publishing scientific publications on an institute's, research group's, and researcher's homepage (on the Web) is a common problem in academic research. The "Department of Informatics" (IFI)¹ at the University of Zurich is organized in groups. Each group has a professor as their head and several senior assistants and assistants as members. For posting their publications on the Web each group has its own solution. Some have just simple Web pages, others have content management systems for entering their publications. Therefore they are lacking of a single solution which meets the actual requirements and is flexible enough to be adapted in future as well. And as it is the Department of *Informatics* their solution could be some kind of 'showpiece' according to the latest standards and recommendations in the further development of the World Wide Web (WWW) which includes the areas of Web 2.0 and the Semantic Web.

In this thesis we developed the publication database PAX that aims for having a user friendly interface (using Ajax) and supporting data exchange by semantic annotations and BibTeX export. The semantic annotation can be used by other applications for doing machine-based processing, e.g. reasoning. The Web 2.0 technologies provide a comfortable user interface (UI) and bring well-known desktop technologies like drag-and-drop to the Web browser. With Ajax, it is also possible to modify single parts of the displayed page without reloading the entire Web page.

The Web application PAX displays scientific publications and projects for the visitors. Employees can login to modify or delete the publications and projects they are allowed to and create publishers, partners/funding organizations and research groups as well as authors/editors. Publications are created according to the widely accepted BibTeX [Kop96] standard. Administrators can create, modify, and delete *all* these objects. Additionally they are responsible for the user management which includes creating, updating and deleting employees.

Standard XHTML [XHT01] Web pages are not the only output of the application, also RDF [RDF04a] and BibTeX data are generated on demand. Furthermore, the XHTML page can be accessed with included RDF data, which is called RDFa [RDF06] and is currently a Working Draft at the World Wide Web Consortium (W3C).²

¹<http://www.ifi.unizh.ch/>

²<http://www.w3.org/>

For a comfortable collection of the existing publications, an import function handling BibTeX files is provided. During the import already existing authors, editors, and publishers are recognized and associated automatically to the publication. A PDF file can be uploaded and attached for each publication. For citing to be possible, technical reports have a unique number according to the IFI standard. Furthermore, two standard cover pages are generated from a LaTeX template and attached to the beginning of the uploaded PDF file.

The application is built upon a PHP framework called symfony [Syma], which itself includes a client-side Ajax framework named script.aculo.us [Scr]. It follows the rules of Object-Oriented Programming (OOP) and Model-View-Controller (MVC).

As some sites in the Web are already annotated with semantic data, PAX enlarges this foundation by a large amount. For Research Communities there even exists a vocabulary in form of the SWRC ontology [SWR], which already went through several iterations and has a wide range. With a wide base of annotated data, further research and development can be done by IFI or others interested in this area.

The remainder of this thesis is organized as follows. Chapter 2 introduces the concepts of Web 2.0, Ajax, and the Semantic Web. Chapter 3 gives a glance at the current solutions for publication databases and shows why another solution is necessary. In Chapter 4 the requirements for the Web application are listed and explained. Chapter 5 describes the extension of the used ontology and Chapter 6 explains the implementation itself. In Chapter 7 the evaluation takes place. The discussion is concluded in Chapter 8 with some final thoughts.

Recent Web Technologies

"The Web is far from 'done', It is in only a jumbled state of construction, and no matter how grand the dream, it has to be engineered piece by piece, with many pieces far from glamorous." Tim Berners- Lee [BL99]

2.1 Web 2.0 and Ajax

The development of the WWW is an ongoing process. The terms Web 2.0 and Ajax came into the focus of the development community recently. Sometimes these terms are misused, so they are briefly introduced in the following.

2.1.1 Web 2.0

The concept of Web 2.0 began with a conference brainstorming session between O'Reilly and MediaLive international. At the first Web 2.0 conference, in October 2004, John Battelle and Tim O'Reilly listed a set of principles and approaches to present their idea. As they list seven different principles, a single Web 2.0 application cannot fulfill all of them. Therefore two aspects are pointed out in the following. [O'R05]

Harnessing Collective Intelligence

In Web 2.0 area network effects from user contributions are the key to market dominance. Sites like del.icio.us¹ and Flickr² have pioneered a concept called "folksonomy". In contrast to taxonomies or ontologies, a vocabulary emerges by usage. Users categorize their sites using freely chosen keywords, referred to as tags; in this way the vocabulary is created.

Other applications that build online communities are blogs, wikis and social networking sites like MySpace³ and Orkut⁴. These days high prices are paid for acquiring such start-ups, but there is criticism too. There are some revolutionary ideas, but they have to prove first being capable of making money in the long-run. [Lor06]

¹<http://www.del.icio.us>

²<http://www.flickr.com>

³<http://www.myspace.com/>

⁴<http://www.orkut.com/>

Rich User Experiences

For this thesis “Rich User Experiences” is the most important principle. Several approaches brought improvement to the user experience in the WWW: Applets, JavaScript, DHTML, Macromedia’s Flash. But the potential to deliver full scale applications did not hit the mainstream until Google introduced GMail⁵ and quickly afterwards Google Maps⁶, Web based applications with rich user interfaces and PC-equivalent interactivity. Another popular example is Zimbra⁷, seen in Figure 2.1, a groupware product including a server and a client software. In these applications, which are running in a Web browser, tool tips, drag-and-drop, right-click menus, and type-ahead pop-ups are common features. These are provided by the use of Ajax. [O’R05]

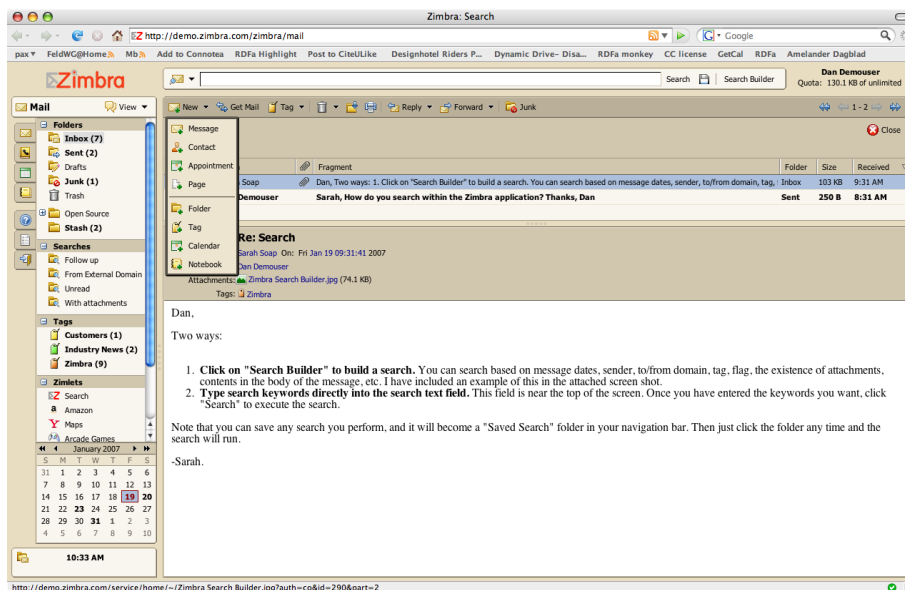


Figure 2.1: Zimbra groupware client

Nowadays a lot more Ajax applications than only E-Mail or groupware clients are revealed: Spreadsheets, Word Processors, Online Maps, Graphics editors and more [Got06a] - and their number is growing day-by-day. Another interesting approach seem to be the Web 2.0 portals, for example seen on Netvibes.⁸

2.1.2 Ajax - Asynchronous Javascript And XML

In classic Web applications most user actions trigger an HTTP request to the server. While the server processes this request and returns an XHTML page to the client, the user is waiting.

An Ajax application eliminates the start-stop-start-stop nature of interaction of Web applications by introducing an intermediary - an Ajax engine - between client and server. This engine, as illustrated in Figure 2.2, allows the user’s interaction with the application to happen asynchronously - independent of communication with the server. Now user actions take the form of

⁵<http://www.gmail.com>

⁶<http://maps.google.com/>

⁷<http://www.zimbra.com/>

⁸<http://www.netvibes.com/>

a JavaScript call to the Ajax engine. Some responses like data validation can be handled by this engine, some need a request to the server, which does not interrupt the user's interaction with the application. This means the user is no longer forced to wait five seconds for a Web page to reload every time he clicks on something. In addition the amount of data transferred is smaller. Instead of resending a whole page with some small changes, only a XML fragment (other content types are allowed too) has to be sent to the client. In this way the application changes in real-time and keeps page content fresh. There is another advantage: as the request is asynchronous, several requests for different parts of the page can happen at the same time. This is a more natural way for user interaction and is very similar to desktop application behavior.

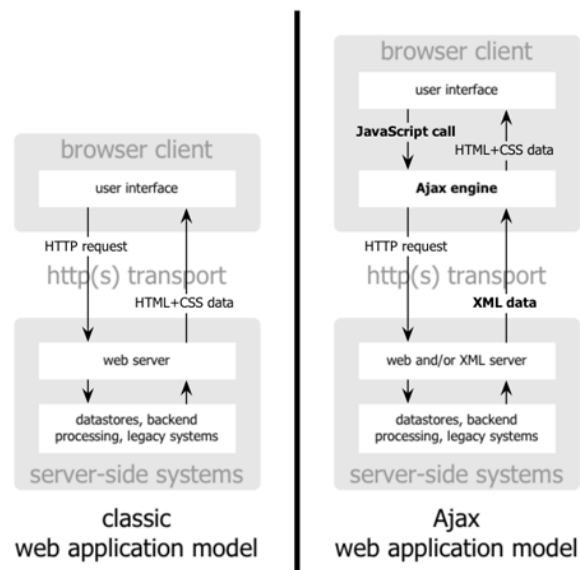


Figure 2.2: The traditional model for Web applications (left) compared to the Ajax model [Gar05]

Ajax is not a plugin or a proprietary technology, it is an architectural style - a high level design pattern - which powerfully puts together the following existing technologies [Gar05, Mah06]:

- Standards-based presentation using Extended Hypertext Markup Language (XHTML) and Cascading Stylesheets (CSS)
- Dynamic display and interaction by manipulating the Document Object Model (DOM)
- Data interchange and manipulation using Extensible Markup Language (XML) and XSL Transformation (XSLT)
- Asynchronous data retrieval using XMLHttpRequest
- JavaScript binding everything together

Ajax Example

For example *without* Ajax searching information retrieval for a car brand could look like this:

1. The user requests the page `http://example.com/cars/`.

2. The server responds with an XHTML page.
3. The user types letter "H" into a search field and presses enter which requests `http:// example.com/cars?filters=H`.
4. The server searches its database and responds with a **complete** XHTML page displaying the filtered car brands.
5. The user clicks on "Honda" which sends a request to `example.com/cars/honda`.
6. The server responds with a **complete** XHTML page.

With Ajax it could look like this:

1. The user requests the page `http://example.com/cars/`.
2. The server responds with an XHTML page.
3. The user types letter "H" into a search field and presses enter which triggers an Ajax request to `http://example.com/cars?filters=H`.
4. The server responds with **four lines** of XHTML:

```
<ul>
  <li>Honda</li>
  <li>Hyundai</li>
</ul>
```

The answer is displayed by the Client as a menu with the two suggestions.

5. The user selects "Honda" which sends a request to `example.com/cars/honda`.
6. The server responds with **some lines** of XHTML about Honda. These lines are inserted into the DOM and displayed to the user.

2.1.3 XMLHttpRequest Object

In the following we briefly introduce the implementation of Ajax. A detailed manual on Ajax Programming can be found at Mozilla Developer Center [MDC07].

Making a HTTP Request

The first step in JavaScript is to create an XMLHttpRequest object for Mozilla, Safari and other browsers or an ActiveX object for Internet Explorer (simplified code version).

```
if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Afterwards, it has to be defined which action will handle the response.

```
http_request.onreadystatechange = function() {  
    // do the thing  
};
```

Now the request has to be implemented. The `open()` method of the HTTP request class defines the method (GET, POST,...), the URL where the request is sent to and if it is asynchronous (`true`) or not:

```
http_request.open('GET', 'http://www.example.org/some.file', true);
```

The parameters of `send()` can be anything you have to send to the server. The data should be in form of a query string, e.g. `'type=car&brand=H&so=on'`.

```
http_request.send('type=car&brand=H&so=on');
```

Handling the Response

Like defined before, `function()` handles the response. This function has to check the state of the request, the value 4 stands for "complete", what means that the data resent form the server was loaded. The function could be called earlier, e.g. after sending the request (state 2).

```
if (http_request.readyState == 4) {  
    // everything is good, the response is received  
} else {  
    // still not ready  
}
```

The next thing to check is the HTTP server response status code, for example 200 can be of interest.

```
if (http_request.status == 200) {  
    // perfect!  
} else {  
    // there was a problem with the request,  
    // for example the response may be a 404 (Not Found)  
    // or 500 (Internal Server Error) response codes  
}
```

Now the response, accessible via `http_request.responseXML`, can be handled and traversed using the JavaScript DOM functions. E.g. it can be attached to another node, which is referenced by an ID:

```
response = http_request.responseXML;  
node = getElementById('car-info');  
node.appendChild(response);
```

2.1.4 Ajax Frameworks

For simplifying the use of Ajax a framework can be used. But as the number of available frameworks is so big, it is not an easy task to find the right one immediately. [Got06b] lists a ranking of the most popular Ajax frameworks and libraries, according to a Burton Group's research, shown in Table 2.1.

Prototype	26.6%
script.aculo.us	19.5%
Direct Web Remoting	14.8%
Dojo	11.1%
Ruby on Rails	10.0%
Rico	6.8%

Table 2.1: The most popular Ajax frameworks and libraries [Got06b]

Prototype is leading with 26.6% and script.aculo.us follows with 19.5%. Notice that Rico and script.aculo.us build on Prototype. Again, Ruby on Rails uses script.aculo.us, so together Prototype share is more than 60%. But not all these frameworks or libraries are exactly for the same purpose, a possible distinction is the one between client and server side, explained subsequently.

Client Side Frameworks and Libraries

On the client side we have general-purpose helper libraries for JavaScript programming. E.g. Prototype can be thought of a library for developers of other libraries. Its emphasis is to extend the JavaScript language itself to support a more object-oriented programming style. The library also provides Ajax support in the form of an Ajax object that can resolve a cross-browser XMLHttpRequest object. Since browsers have different kinds of implementations, hiding the incompatibilities between Web browsers from different vendors is one of the major tasks of a client side Ajax framework.

At a next level we have prebuilt widgets or complete widget sets for Ajax developers. The script.aculo.us libraries are built on top of Prototype and provide two major pieces of functionality. An Effects library defines a range of animated visual effects to change size, position and transparency of DOM elements. Second, the drag-and-drop library provides the `Sortable` class which takes a parent DOM element as an argument and enables drag-and-drop functionality for its child nodes.

Another product at this level is Dojo [Doj]. It provides a lot of Widgets, like floating windows, text editors, or date pickers which can be used instantly. For Dojo there is an Eclipse plug-in [ATF] developed with support of by IBM, RedHat, Oracle, Yahoo and others.

Other characteristics of client side frameworks are the independence of server side technology, no dynamic javascript code generation and the possibility of combining several Ajax frameworks. [CPJ05, LMS06a, Shi06]

Server Side Frameworks

The client side frameworks are executed in the browser and can be served as static JavaScript files from any server. The server side frameworks reside on the server and generate at least some JavaScript code or XHTML markup dynamically. Direct Web Remoting (DWR) is a Java-based framework which exposes its methods directly as Ajax requests. This is a convenient way to access these functions or methods from the Web browser code. [DWR]

Ruby on Rails (RoR) is a Web development framework written in the Ruby programming language. It bundles together solutions for presenting content using templates and mapping

server-side objects to a database. Development happens fast, since for a lot of common code code-generating techniques are used. RoR uses the Prototype framework for Ajax purposes. [RoR]

There exist many other frameworks not listed in Table 2.1. Google Web Toolkit (GWT) is a server side framework too. The Ajax application developed with GWT is built completely in Java and afterwards compiled into a Web application (XHTML and JavaScript). The whole application interface can be developed with Swing or SWT without coming in touch with any JavaScript code. For managing the communication with the server Remote Procedure Calls (RPC) are used, but in a proprietary way. [GWT]

The server side framework that we chose for developing for PAX is called symfony. It is a PHP open source framework, which is very similar to RoR. It makes use of an object-relational mapping (ORM) tool to access the database tables like objects and it is compatible with a lot of different databases. With symfony, you do not need to create code from scratch, it can create templates for Creating, Retrieving, Updating and Deleting (CRUD). For the use of Ajax it is based on the script.aculo.us framework (using Prototype). For creating a backend administration there exists the so called "admin generator", which automates repetitive tasks using a system of custom configuration. If this does not fulfill the requirements, it provides the possibility to plug the own code into the generated PHP templates. Symfony is based fully on the MVC pattern and the OOP paradigm. [CPJ05, LMS06b, Syma]

2.2 Semantic Web

Daily life changed since the public accessibility of the WWW: the way people communicate with each other, how information is disseminated and retrieved, and how business is conducted. The Semantic Web will dramatically improve the WWW and its use. The idea of the Semantic Web is not a new one. Tim Berners-Lee mentioned it publicly first at the World Wide Web Conference in 1994. But most of it is not realized yet. Therefore it is not always an easy task to explain its idea and convince people of the need for it. In this section it is shortly explained why the World Wide Web is not enough. [AvH04, SBLH06]

2.2.1 Shortcomings of the WWW

Even if the WWW is adopted widely and people who use it are quite pleased with it, it has a lot of shortcomings. Information retrieval is not always easy, often awkward. For example someone is looking for sources about the technology Ajax: both in general search engines like Google ⁹ and in specialized book search engines like that of NEBIS ¹⁰ the useful results are blended with those of the greek hero also named Ajax. The applications involved cannot distinct between the two items. This is because of the lack of semantics in a computer readable form - today's Web is mostly defined for humans to read. If the user clicks the delivered hyperlinks, from the description of the books he can conclude if it is about the technology or the greek hero. The current applications cannot.

Here the Semantic Web sets in - its purpose is to bring structure to Web pages and facilitate the task for automated software tools. This happens as an addition to the current Web and not as a separate one. Current Web applications can be enhanced - *annotated* - to participate and deliver

⁹<http://www.google.com/>

¹⁰<http://www.nebis.ch/>

semantics. This happens by encoding additional metadata into the source code. Therefore the W3C defined standards like RDF and OWL. [AvH04, BLHL01]

2.2.2 Resource Description Framework - RDF

RDF is essentially a data model, whose basic building block is an object-attribute-value triple, called a *statement*. An example statement is:

The person identified by "http://example.com/person/id57" has the name "Harald Gall".

The simplest way of describing this statement would be:

http://example.com/person/id57, http://example.com/ontology#name, #Harald Gall

The statement could also be illustrated by a graph, as seen in Figure 2.3, what is useful for human understanding.



Figure 2.3: Graphical representation of a triple

A third representation possibility is based on XML. It is called RDF/XML and imposes a *serialization*. The order of descriptions is not significant, the graph model is the real data model of RDF while the other two possibilities are just possible serial representations of the graph. [AvH04]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:person="http://example.com/ontology#">
  <rdf:Description rdf:about="http://example.com/person/id57">
    <person:name>Harald Gall</person:name>
  </rdf:Description>
</rdf:RDF>
```

2.2.3 RDF Schema

As in RDF no means for defining an application-specific vocabulary is given, RDF Schema (RDFS) provides these facilities needed for describing classes and properties. With RDFS, the *Core Classes* are defined, so that classes, resources, literals, properties and statements can be declared. In addition *Core Properties for defining Relationships* are set: instances, subclasses and subproperties can be used and with the *Core Properties for Restricting Properties* domains and ranges can be marked. [AvH04, RDF04b]

For example a university class can be declared subclass of the organization class:

```
<rdfs:Class rdf:about="http://example.com/ontology#University">
  <rdfs:subClassOf>
    <rdfs:Class rdf:about="http://example.com/ontology#Organization"/>
  </rdfs:subClassOf>
</rdfs:Class>
```

A book class can be labeled by a language specific label, like the german translation "Buch" or with a comment like "A complete book, not formed from separate papers.":

```
<rdfs:Class rdf:about="http://example.com/ontology#Book">
  <rdfs:label xml:lang="de">Buch</rdfs:label>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    A complete book, not formed from separate papers.
  </rdfs:comment>
</rdfs:Class>
```

Further the range of the "isWorkedOnBy" property can be set to "Person" and its domain to "Topic":

```
<rdf:Property rdf:about="http://example.com/ontology#isWorkedOnBy">
  <rdfs:range rdf:resource="http://example.com/ontology#Person"/>
  <rdfs:domain rdf:resource="http://example.com/ontology#Topic"/>
</rdf:Property>
```

2.2.4 Web Ontology Language - OWL

As RDF is limited to binary ground predicates and RDFS to subclass and property hierarchy, with domain and range definition out of these properties, there are use cases that require more expressiveness. Therefore the W3C defined OWL - it builds upon RDF and RDFS and has the same kind of syntax. As it uses formal semantics, it is possible to do reasoning about the knowledge what allows the following:

- Checking the consistency of the ontology and the knowledge
- Checking for unintended relationships between classes
- Automatically classifying instances in classes

Therefore it adds the possibilities to define a local scope for a property, to express the disjointness of classes, the boolean combination of classes, cardinality restrictions, and special characteristics of properties (transitive, unique and inverse).

For example the property “publisher” is restricted to refer only to “Organizations” when it is applied for an “InBook”.

```
<owl:Class rdf:about="http://example.com/ontology#InBook">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="http://example.com/ontology#publisher"/>
      <owl:allValuesFrom>
        <owl:Class rdf:about="http://example.com/ontology#Organization"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL Sublanguages

The richer a language is, the more inefficient the reasoning support becomes. Therefore OWL offers three species, each fulfilling different aspects of this full set of requirements: *OWL Full* is fully upward-compatible with RDF. That means that any legal RDF document is also a legal OWL Full document and any valid RDF/RDFS conclusion is also a valid OWL Full conclusion. The drawback is that with this power the language is undecidable and no complete (or efficient) reasoning is possible. *OWL Description Logic (DL)* restricts how the constructors from RDF and OWL may be used. This ensures that the language corresponds to a well studied description logic but drops full compatibility with RDF. An RDF document will have to be extended in some ways and restricted in others to be a legal OWL DL document. On the contrary every legal OWL DL document is a legal RDF document. *OWL Lite* is further restricted to a subset of the language constructors. Therefore it is easier to grasp and to implement with the trade-off of having restricted expressivity. [AvH04, OWL04b, OWL04a]

OWL uses RDF and RDFS: all OWL species use RDF for their syntax and instances are declared in RDF.

Deciding which OWL sublanguage shall be used depends on the needs. If reasoning is a must, then OWL DL or OWL Lite has to be chosen and if the language shall be easy to understand OWL Lite probably is the right choice. Vice versa if more expressive constructs are needed OWL Full is an obvious choice.

2.2.5 Embedding RDF in XHTML

A widely use of RDF will probably be in the World Wide Web, therefore it is necessary to embed RDF in Web pages. The following possibilities exist:

- Embed annotations in the document
- Link to an external document containing the annotations
- Annotations link to the document - the document is unchanged

The first two out of these possibilities will shortly be examined now.

Embed Annotations in the Document with RDFa

There exist several approaches for embedding RDF metadata in an XHTML document, see [Pal02]. A newer one is RDFa (formerly known as RDF/A). The intended features of RDFa are:

- Independence: pick the vocabulary of metadata terms
- Modularity: reuse other vocabularies
- Evolvability: possibility to change the meaning of the vocabulary over time
- Do not repeat yourself: if you render data in XHTML, why repeat it in the structured form?
- In-context: metadata is right next to the data it describes, if you copy and paste the XHTML, you get the metadata along with it

RDFa does not validate in XHTML 1.0 but it also does not break anything in today's browsers. Not even in XHTML strict rendering mode.

Imagine the following piece of XHTML data:

```
<html xmlns="http://www.w3.org/1999/xhtml">
...
<div>Gerald Reif, Martin Morger, Harald Gall, Semantic Clipboard</div>
...
</html>
```

Using RDFa the XHTML code could look like this:

```
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:example="http://example.com/ontology#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xml:lang="en" lang="en">
...
<div about='http://example.com/publication/id6'>
  <span rel='example:author' type='example:Person' href='author/id2'>
    Gerald Reif
  </span>,
  <span rel='example:author' type='example:Person' href='author/id5'>
    Martin Morger
  </span>,
  <span rel='example:author' type='example:Person' href='author/id3'>
    Harald Gall
  </span>
  <span property="example:title" datatype="xsd:string">
    Semantic Clipboard
  </span>
</div>
....
</html>
```

The `about` attribute determines the subject, which is in this case a publication identified by the URI `'http://example.com/pax/publication/edit/id/6'`

We have two possibilities to define the object: URIs and Literals - here both kinds are used. When the object is a URI, the predicate is designated using `rel` (or `rev` for a reverse relationship.) The `rel` attribute defines the RDF predicate, which is `'example:author'`. (The corresponding ontology is defined in the `head` section.) The predicate points to the object, whose URI is given in `href` (e.g. `href='author/id2'`) and its `type` is `'example:Person'`. When the object is a literal, `property` is used for defining the predicate, `'title'` in this case. The object has the `datatype` `'xsd:string'` and its value is given as a literal: `'Semantic Clipboard'`. [Rei06, Pal02, RDF06, Adi06]

Link to an External Document Containing the Annotations

Another possibility is that a link points to an external meta data description, which also has to be accessible, what should not be a problem in the case of Web pages. Therefore the `link` element in the `head` section is used like this:

```
<link rel="meta" type="application/rdf+xml" title="RDF Version"
href="publications.rdf" />
```

[RDF, RDF00]

2.2.6 The Current State of the Semantic Web

Nowadays semantically annotated Web pages are rarely found. It is just like the chicken-and-egg problem: What emerges first? - semantic annotations (if there are no tools for doing anything with them) or tools which use semantic annotation (if there are no annotations)? But fortunately there are several projects at several places which are doing research on the Semantic Web. At the University of Zurich tools like the *Semantic Clipboard*¹¹ or *Ginseng*¹² handle Semantic data.

Current applications creating and using semantic data are *Friend-of-a-Friend*, *FOAF*¹³ and *Creative Commons*, *CC*¹⁴. Friend-of-a-Friend defines an ontology in the range of social networking. Creative Commons is an RDF-based representation of copyright policies to facilitate and maximize appropriate reuse. [SBLH06]

2.2.7 Semantic Web Outlook

In my opinion it is only a matter of time until the Semantic Web is widely used and Web pages are annotated (where it is useful). The advantages and the potential are immense and often it is just a little step to generate RDF metadata from an existing database. As more ontologies emerge and agreements are found, it will be common to generate these data instantly while generating the Web page.

¹¹<http://seal.ifi.unizh.ch/semclip/>

¹²<http://www.ifi.unizh.ch/ddis/research/semweb/talking-to-the-semantic-web/ginseng/>

¹³<http://www.foaf-project.org/>

¹⁴<http://creativecommons.org/>

Review of the State of the Art

The existing solutions for managing publications are of different kind. There are Web applications which can be installed on any server, there are Web portals which offer their service via Web access. Both types can be closed or open source, leading to four categories as shown in Table 3.1; of each category one solution is introduced in the following.

	open source	closed source
any server	Basilic	AIFB Portal
provider's server	Connotea	Bibsonomy

Table 3.1: Available solutions for managing publications in a database

3.1 Basilic - Bibliography Server for Research Laboratories

Basilic [Bas] is a PHP open source publication database, which is used several times. It requires an Apache Web server with PHP and MySQL installed and properly configured. The installation happens by executing a shell script, thereby a PHP file is generated. If this file is accessed via Web server it shows the remaining errors and warnings. The MySQL tables are created by pasting an already prepared text file. The Web application handles publications and authors as RDF resources, other "objects" (e.g. the publisher or even editors) are handled as strings. In the list view publications can be searched by author, title and year.

For internal use separate pages (only available on the intranet) have to be accessed. In this way publications and authors can be created, modified and deleted. Before creating a new publication a type has to be chosen. Afterwards, entry fields can be filled (see Figure 3.1), authors can be added, sorted and removed via JavaScript menu. Creating a new author requires accessing another page and afterwards returning to the publication page.

Adding Documents to a publication happens by moving them to the right folder on the server. A large range of file formats is supported - from movies to postscript files. After updating the publication, these files are included into the Web page of the publication. For importing new entries BibTeX files can be loaded via a python script which is based on the xdkbibtex [XDK] library. This

library can be downloaded as Linux binary.

As export format PubliCNRS¹ is supported as well as BibTeX. The Look and Feel is customizable via CSS. Possible extensions are the use of multilingual Web sites and managing multiple groups.

Figure 3.1: Adding a publication in Basilic

3.2 AIFB SEmantic PortAL

The AIFB SEmantic PortAL (AIFB SEAL)² uses a closed source software developed by the Institute of Applied Informatics and Formal Description Methods (AIFB). In [SBH⁺05, HS04] some insights are provided. The Portal is built in close relation to the SWRC ontology, developed by the same institute. This means that projects, research groups, authors/editors and even topics are treated as objects and not as literals. Available export formats are BibTeX and RDF compliant to the SWRC ontology.

3.3 Connotea

Connotea³ is a free online reference management system with tagging support and bookmarklets.⁴ As user contribution is a key factor for Connotea, it can be claimed a Web 2.0 application. There are several import formats supported: Firefox Bookmarks, RIS, EndNote, BibTeX (Experimental),

¹<http://publicnrs.inist.fr/>

²<http://www.aifb.uni-karlsruhe.de/about.html>

³<http://www.connotea.org/>

⁴Bookmarklets are added to the bookmarks in the user's Web browser. Adding any Web page to Connotea happens by clicking on the bookmarklet which executes some JavaScript code.

ISI Web of Knowledge (Experimental), and MODS (Experimental).

Export is available in RIS, EndNote, BibTeX, MODS (XML), RSS and RDF. The generated RDF file uses Dublin Core [DCa], Dublin Core Terms [DCb], PRISM Basic namespace [Pri] and the already mentioned FOAF [FOA].

The source code of Connotea is released under the terms of the GNU GPL⁵ and it offers a Web API⁶.

3.4 BibSonomy

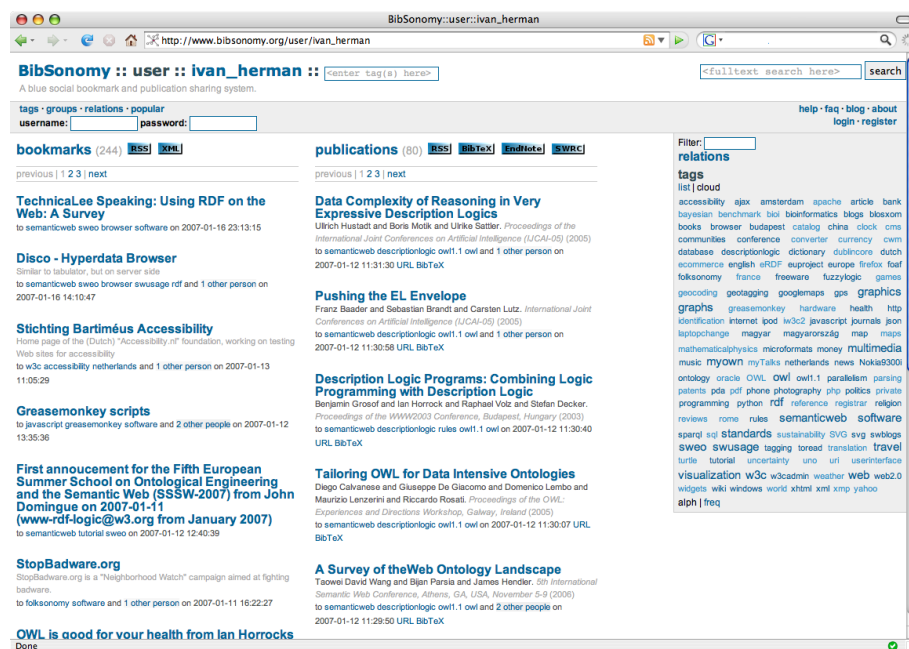


Figure 3.2: Bibsonomy (user view)

BibSonomy⁷ is not only a publication sharing system but also a social bookmark system and is also claimed a Web 2.0 application. On the bookmark side, so called *folksonomies* are used, conceptual structures created by people. In contrast to ontologies, they rely on emergent semantics which result from the converging use of the same vocabulary.

On the publication side their data model is based on BibTeX. Publications can be entered manually, PDF/PS files can be uploaded and extra URLs can be associated with a publication. Additional BibTeX entries can be entered into a *misc* field. BibTeX import is possible by uploading a local file. Publication entries can be public, private or only viewable for friends. Publications without an author, an editor and a year field are ignored and duplicate entries are recognized.

⁵<http://www.gnu.org/copyleft/gpl.html>

⁶<http://www.connotea.org/wiki/WebAPI>

⁷<http://www.bibsonomy.org/>

For each publication tags have to be associated. A third possibility for adding publications is via a JavaScript bookmarklet.

Export is available for RSS, BibTeX, and EndNote. Furthermore, RDF according to the SWRC is supported as an output format. Publishers, authors and editors are treated as literals and therefore do not have a unique identifier. The basic building blocks are an Apache Tomcat servlet Container using Java Server Pages, Java Servlets and a MySQL database as backend. Bibsonomy is shown in Figure 3.2. [HJSS06]

3.5 Discussion

From the beheld applications each one has some very interesting features, but none does meet our requirements as discussed in Chapter 4 or qualifies for an extension or reuse.

In Bibsonomy and Connotea users can upload descriptions of any publication. By doing this a big amount of data is collected, for which the portals provide powerful search tools and a lot of export formats. PAX has to be a tool for managing someone's **own** publications.

Advanced tools for managing an institute's, a research group's or a researcher's publications are AIFB SEAL and Basilic. AIFB SEAL does not make use of Ajax or JavaScript and it is closed source. That means it has to be licensed and cannot be modified arbitrarily to fulfill all the requirements. What can be reused and extended is the underlying ontology called SWRC.

Basilic is an open source Web application but it does not support Ajax and fulfills only a part of the requirements. It handles publications, authors, and research groups. In PAX not only publications, but also projects have to be handled. In the background publishers, partners/funding organizations, research groups, and authors/editors have to be managed. They must be treated as resources with an identifier, otherwise the power of semantic annotation is not exhausted. Basilic supports BibTex import via command line interface for the server administrator. In PAX import has to be available via user interface so that every employee can import and manage his own publications. As Basilic does not build on a Web application framework, it is not considered extensible for our purpose.

For the exposed reasons PAX is developed as a standalone Web application. Semantic annotations build on SWRC, which is extended as presented in Section 5.1.

Chapter 4

Requirements

The requirements for the Web application PAX were elaborated with the SEAL¹ and the DDIS² group at IFI and are presented in the following.

4.1 General Requirements

There are some general requirements and basic conditions to be followed. Like indicated, the aimed software should have a comfortable, desktop-like interface using Ajax and be accessible via Web browser. Its annotation has to follow the RDF recommendation and be compliant with the SWRC ontology. The layout of the page has to be adjustable via cascading style sheets (CSS) [CSS06]. PAX should at least run on an Apache Web server with PHP support and a MySQL database installed.

4.2 Roles

Different users of the application can be distinguished. Most people probably will access the Web page via Internet. Other users are the employees of the university which basically enter new publications and projects. A third kind of users are the administrators, employees which have special access rights. These three roles are examined further now.

A **visitor** is anyone who accesses the Web page, is not logged in as an employee and therefore only can visit the existing entries. This means projects and publications are displayed in list view and by clicking them on an individual page. The available RDF, BibTeX files can be downloaded (see Section 4.4) and of course the PDF file, if one is attached and its download is enabled. Search and filter functions for publications and projects are available.

An **employee** has to authenticate himself by name and password. After the login process, he has the access rights defined in Section 4.3. Notice that employees can be professors, senior assistants or assistants which can be added as authors/editors to publications or as heads/members to projects, or secretaries which must not be added to any publication or any project. For employees it is possible to download a CSV file containing the publications, what is used for the annual

¹<http://seal.ifi.unizh.ch/>

²<http://ifi.unizh.ch/ddis/>

academic report of the institute.

Administrators are allowed to create, modify, delete any resource available. There is a default administrator which can assign administrator rights to other employees. In principle every employee (professors, secretaries, and (senior) assistants) can be nominated administrator depending on the institute's policy.

4.3 Function Range

The following resources need to be covered:

- Publications
- Projects
- Publishers
- Partners/Funding Organizations
- Research groups

User management could be implemented in several ways. For the needs given, the following resources are appropriate:

- Employees
- Authors/Editors

The necessary functions for each out of these resources are listed and described now.

4.3.1 Publications

Publications can be viewed in a list or in detail as individual instances. In list view no RDFa data is included by default, but the same list view can be accessed including the RDFa metadata. On the individual page view RDFa is always included.

Publications are stored according to the BibTeX standard which includes the following publication types: *Article*, *Book*, *Booklet*, *Conference*, *InBook*, *InCollection*, *InProceedings*, *Manual*, *MastersThesis*, *Misc*, *PhDThesis*, *Proceedings*, *TechReport*, *Unpublished*. In addition the following two types have to be available: *DiplomaThesis* (treated like MastersThesis) and *DoctoralThesis* (treated like PhDThesis). Each publication has mandatory and optional fields in the standard (see Appendix A), but in PAX the employee is not forced to fill in mandatory fields. The title field has to be unique in PAX and cannot be left blank. A special field is type: it is allowed for each publication type by the standard, PAX uses this field for marking publications as a demo, a poster or anything else. If it is filled, it is displayed in the list view after the publication type like this: e.g. *InProceedings/demo*. A field called abstract is available for each publication, as it is used very often, and the german counterpart "Zusammenfassung" is available for the four thesis publication types. Moreover the employee must be allowed to create additional fields with any name. These fields do not have to be included into the RDF.

To include also publications which employees did before working at IFI, a label for marking this is available. For each publication data has to be available in RDF and BibTeX. These files have

to be accessible via an explicit URL, so they can be indexed by search engines. For the entries two dates have to be stored: a date when the publication entry was created and a date when it was modified.

PDF Download

For each publication the possibility to attach a PDF file has to be given. Tied to the PDF files there are some special requirements: a date field has to be available for storing a date until when download is disabled. Another label can be chosen if the PDF file has not to be available for download at all. In foresight for future support of the Open Archives Initiative (OAI) [OAI], a label is available to mark if a PDF file is allowed to be published on another server or not.

Associations

By adding a new publication an employee is stored as creator. For a publication any of the existing research groups can be assigned. An arbitrary number of authors can be added to an individual publication, their order must be changeable and therefore user-defined. The same two conditions hold for editors. A publication can be associated to any project that is listed, here the order is not of importance, and it can be allocated to one publisher at maximum. Every publication can be associated to any of the research groups.

Access Rights

Access to publications is as follows: every employee can add new publications. An employee is allowed to modify or delete a publication if he did create it, if he is in a group that is assigned to the publication, or if he is an author or editor of the publication. Administrators can modify and delete any publication, even technical reports.

Technical Reports

Creating technical reports must be considered well. From the time a technical report is available in the Web, it can be cited by anyone and therefore modifications can only be made in exceptional case. That is the reason why technical reports are treated in a special way: they need a not-to-be-overseen warning and they can only be changed the same day they are created, afterwards only an administrator has the rights to do this. For technical reports two additional cover pages are added to the beginning of the PDF and a serial number is generated automatically. Some changes can have effects on the cover pages, therefore on each saving the PDF (without cover pages) must be uploaded again. The cover pages have to be generated from a LaTeX template.

Searching and Filtering

From the query view searching and filtering in PAX is the same: Adding a new criterion creates an additional restriction to the query. Our distinction is the following: In search fields the user can type in what he likes to. In filter fields the user can also type in what he likes to, but while he is typing suggestions are made (see Section 4.5.1), so that he will probably choose and enter an existing value. That is why we call it filtering. Another requirement for searching and filtering is that a match in the middle of a word has also to be taken into account. E.g. searching for "ab" has also to deliver an author/editor called "Patrick Knab".

In the publication list, entries can be searched by title and abstract. In addition filtering by publication type, type, year, authors/editors, groups, and publisher is possible. For year searching a range can be defined, that means a minimum and a maximum year can be entered. Filling authors/editors searches for authors *and* editors. If "only my last import" is set to "yes" in a search, only results from the last import are displayed, on "no" the search is not restricted with this criterion. "Published at IFI" filters publications by the already described label, what excludes former publications of employees, which were written at another university.

Sorting and Categorizing

The list view has to be sortable by year. The list has to display a category for every year, beginning at the highest number: e.g. 2007, 2006, 2005, etc. Optionally sorting by publication type has to be available, which creates categories for every publication type, sorted alphabetically descending: Article, Book, Booklet, etc.

Direct URLs

For the institute's, an employee's, a research group's, or a project's publications an explicit and permanent URL to show only their publications has to be available, so that they can include the link on their Web page and that it can be indexed by search engines.

BibTeX Import

A lot of publications in a short period of time will be added by moving from the old solutions to PAX. All former publications have to be acquired, therefore adding new publications is facilitated by offering BibTeX file import. A BibTeX file can be uploaded from the employee's computer or transferred via URL from anywhere in the Web. Authors and editors have to be recognized and associated automatically, duplicates of publications have to be avoided. Import is not allowed for any not defined publication type, e.g. *jurthesis*, and for technical reports. As described in Section 4.3.1, technical reports must be handled with care. On creating one, automatic changes happen: an auto-generated number is assigned and the uploaded PDF file is modified. Because of the auto-generated numbers only the administrator is allowed to acquire former technical reports.

4.3.2 Projects

Projects can be displayed in list view, where RDFa inclusion is an option, and as individual pages also with RDFa included.

For projects the following attributes have to be available: title, homepage, kind, and description. The title field is the only one that must be filled. A project by default is of the kind "Research Project", this can be changed to "Cooperation". In list view the kind of the project has to be displayed after the title. More fields available are a start date and an end date, a logo and a field for entering keywords. By default a project is labeled as "active" but can be changed to be not active anymore.

Associations

By creating a new project, the employee is assigned as its creator. Any available research group can be associated. Heads and members, which are chosen of the available authors/editors, can be added, but there is no ordering. Partners and funding organizations can also be added but not

ordered. If the kind of the project is "Cooperation", the contact person for a partner or funding organization is displayed by viewing an individual project.

Access Rights

Creating projects is allowed for every employee. Modifying, deleting is possible for creators, members of the associated research groups, heads and members of the project and the administrator. Deleting has to be confirmed additionally.

Searching and Filtering

The project list can be searched by title, description, and keywords. Adding more criteria restricts the search. Filtering by groups is also possible.

Sorting and Categorizing

The project list has to display the active projects first under the title "Active Projects" and afterwards the not active ones under the title "Former Projects".

4.3.3 Publishers

Publisher have a unique name (required) and a homepage.

Associations

Each publisher can be associated to any number of publications.

Access Rights

Creating publishers is allowed for employees. Changes have an impact on every publication that refers to the publisher. Therefore changing a publisher can affect publications of any research group. For this reason deleting and modifying publishers is only allowed to administrators.

4.3.4 Partners/Funding Organizations

Partners/Funding Organizations are resources which can be associated to a project either as partners or as funding organizations. The attributes of partners/funding organizations are the name, which is required, a logo, a homepage, and a contact person.

Associations

As described, all entries can be associated to an undefined number of projects as partners and/or funding organizations.

Access Rights

Partners and funding organizations can only be managed by an administrator. The creation of new partners/funding organization is also restricted to an administrator.

4.3.5 Research Groups

Creating a research group requires filling in name and short name, these fields must be unique. A homepage is optional.

Associations

A research group is associated to an arbitrary number of employees, which can be in any position, and to undefined numbers of publications and projects.

Access Rights

Research Groups will not change very often. Modifications have a big impact on the data, that is why creating, deleting and modifying is only allowed to administrators.

4.3.6 The Relation Between Employees and Authors/Editors

PAX has to offer user management via Web interface. Therefore some requirements are given. Every employee must be able to authenticate. Some employees are authors and editors of publications or heads and members of projects, some are not. Furthermore, if an employee changes its name, for example by marrying, the already assigned publications and projects still have to display the former name while the newly created one has to display the new name. Therefore we decided to treat employees and authors/editors as different objects. This has another advantage: on employing an external author/editor no publications or projects have to be modified. Only a new employee has to be created and the author/editor has to be connected to it.

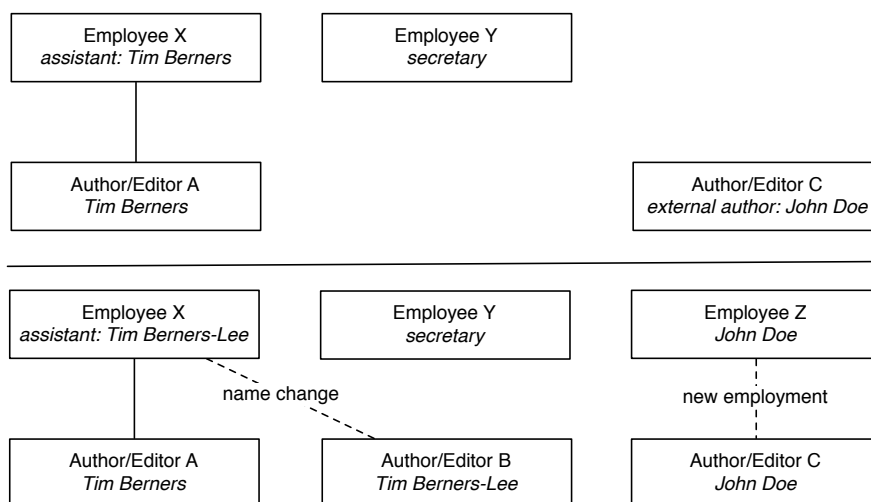


Figure 4.1: The relation between employees and authors/editors

In Figure 4.1 these examples are shown: Employee X is an assistant called "Tim Berners", the corresponding author/editor is Author/Editor A with the same name. Tim marries a wife with the surname "Lee", therefore his surname in the database has to be changed to "Berners-Lee". This creates Author/Editor B and associates it to Employee X. Former publications now reference

still Author/Editor A and new publications can reference Author/Editor B.

Employee Y is a secretary which is not associated to any publication or project, therefore no author/editor is associated to her.

“John Doe” is newly employed but already did publications together with employees, therefore he was created as external Author/Editor C. Now an employee account (Employee Z) can be created, allowing John to authenticate, and this employee can be connected to Author/Editor C. Like this no publication or project has to be touched.

4.3.7 Employees

Like described in the preceding Section 4.3.6, employees are the ones employed by the institute as professors, (senior) assistants or secretaries. The required fields for an employee are first name, surname, login, and email. The last two are necessary for authentication. Optional fields are numerous: position, password, distinction field, homepage, email, phone, and room. The combination of first name, surname, and distinction field has to be unique. Homepage, phone, and room have to be created because of future plans to create the people Web site of the institute with the same data.

Two labels are available. The first one decides if the employee can be associated to publications and projects or not. E.g. a secretary has no need for that, as shown in Figure 4.1 (Employee Y). By default it is selected on creation; saving an employee with this label automatically leads to the creation of an associated author/editor. Another label is available for declaring an employee to be administrator, this is not chosen by default.

Associations

The association of employees to authors/editors happens transparently. By creating a new employee a new author/editor with the same name is created and connected. By renaming an employee, a second author/editor is created according to the new name. Employees can be associated to any number of research groups, but do not have to.

Access Rights

Because of the complexity of user management, employees can only be handled by an administrator.

Passwords

An employee must have the possibility to change its password anytime he wants, and an option to reset a forgotten password has to be offered.

4.3.8 Authors/Editors

An instance of this class can have several associations: it can be author of some publications, it can be editor of some publications, it can be head of a project and it can be member of a project. The given name is a compromise: just calling it Author could be misleading, calling it Author/Editor/Head/Member could be confusing too, so it is called Author/Editor. As described, Authors/Editors are generated transparently when a new employee is created or an employee's

name changes. The first name, the surname, and the distinction field are copied from the employee.

Acquiring an external one means creating a new Author/Editor without associations to any employees.

Associations

The possible but not mandatory associations all were mentioned before: an Author/Editor can be connected to one employee at maximum, to an undefined number of publications (as author or as editor), and to an undefined number of projects (as heads or as members).

Access Rights

When a new publication is entered, the authenticated employee must have the possibility to create new external authors or editors without the need for an administrator's help. The newly created instances have to be accessible to any research group, so deleting and modifying them can only be done by an administrator.

4.4 Output Formats

Via "RDF for the entire database" the whole database is translated into one RDF file. If no filters/searches are applied, "BibTeX for filtered publications" and "CSV for filtered publications" outputs all publications in BibTeX, respectively CSV (comma separated values) file format. If a filter or a search is applied, only the delivered publications are taken into account for BibTeX and CSV output. Every research group can access all IFI publications from the last year in a CSV file via button (according to a given schema from the university of Zurich). This is necessary for uploading an annual academic report into another Web application. For any individual resource or for all resources of one class (e.g. for all publishers) RDF files are created on demand. For individual publications BibTeX output is available.

4.5 User Interface

The application has to have an easy-to-use Web interface. By the possibilities due to the use of Ajax the following User Interface (UI) requirements are given in the following.

4.5.1 Type Ahead

The UI has to be more desktop-like than in common Web applications. Filtering has to be supported by type ahead suggestion (sometimes called auto complete), which means that a list with suggestions is displayed below the entry field. Of course other values are allowed as well. For example if a user only knows the surname of an author, e.g. *Gall*, he can type it in and instantly he can choose entries from a list, with all the entries where Gall is a part of, e.g. *Harald C. Gall* or *Betram Gallus*.

4.5.2 Creating, Sorting, and Removing Objects

Creating external authors, external editors, and publishers for a publication or external heads and members for projects must not require to navigate away from page where the employee is entering or editing a publication or project. The user has to get feedback if a new instance he likes to create already exists or if a required field is not filled.

Adding already existing entries has to be supported by type ahead suggestion. The order of authors and editors has to be sortable by drag and drop. Removing one has to be achieved by a single click. If a mistyped entry cannot be found and added, the user has to be informed. Adding, sorting or removing anything on the page must not force reloading the whole page.

4.6 Summary

The resulting classes and properties, according to the described requirements in the preceding sections, are shown in Figure 4.2.

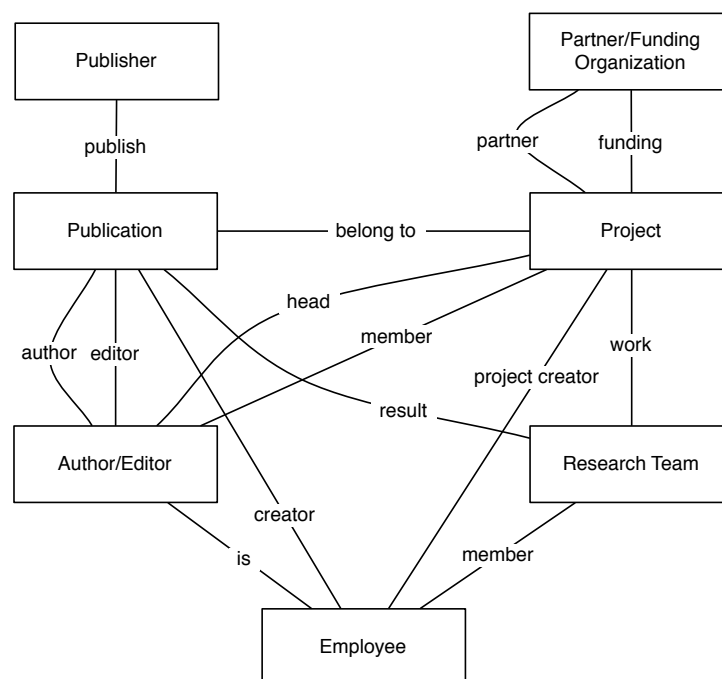


Figure 4.2: Required classes for the publication database

The SWRC-unizh Ontology

As described in the general requirements (Section 4.1), the SWRC ontology had to be used for PAX. The SWRC ontology [SWR] was designed for representing knowledge about researchers and research communities. Its creator is the Institute AIFB at the University of Karlsruhe¹. The version 0.7 of SWRC is published in OWL²; it uses a special subset of OWL called OWL-DLP³, which is close to OWL-DL. As the provided classes and properties were not fully adequate for PAX, an extension of the SWRC was done. We call it SWRC-unizh and use the namespace `swrc-unizh`.

In the following section, a short overview of the relevant classes of the SWRC ontology v0.7 is given. It is shown why the provided classes are not fully sufficient and which extension is done.

5.1 SWRC - Semantic Web for Research Communities

As shown in Figure 5.1, the SWRC contains seven top level concepts. The relevant ones for PAX are *Document*, *Organization*, *Person* and *Project*.

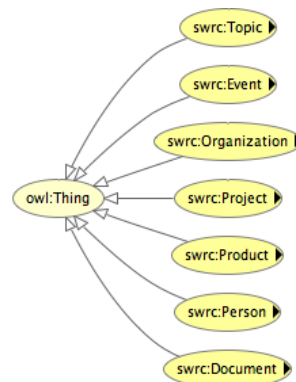


Figure 5.1: The SWRC top level concepts

¹<http://www.aifb.uni-karlsruhe.de/Forschungsgruppen/WBS>

²<http://ontoware.org/frs/download.php/299/swrc.v0.7.owl>

³<http://logic.aifb.uni-karlsruhe.de>

The *Publication*, which is a direct subclass of *Document*, concept (shown in Figure 5.2) is closely tied to the BibTeX standard and therefore contains the subconcepts *Article*, *Book*, *Booklet*, *Conference*, *InBook*, *InCollection*, *InProceedings*, *Manual*, *MasterThesis*, *Misc*, *PhDThesis*, *Proceedings*, *TechnicalReport*, and *Unpublished*, which corresponds with our needs. The remaining classes are not used in PAX.

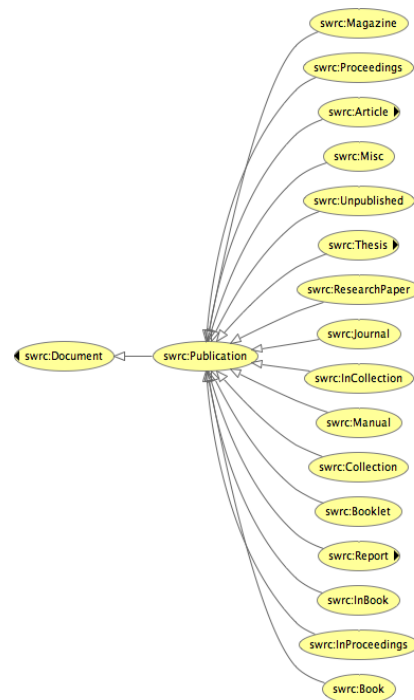


Figure 5.2: The SWRC publication classes

The *Organization* has the subclasses *Association*, *Department*, *Enterprise*, *Institute*, *ResearchGroup*, and *University*. Out of these, *Enterprise* is used for publishers, *ResearchGroup* is used for the research groups and the main class *Organization* is used for partners and funding organizations. The other four classes displayed in Figure 5.3 are not used in PAX.

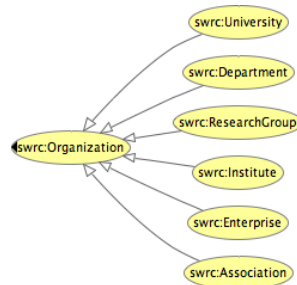


Figure 5.3: The SWRC organization classes

The *Person* concept models humans - it is specialized by the subconcepts *Employee* and *Student*, which contain more subconcepts itself. Since the only class we use is *Employee*, these classes, shown in Figure 5.4, are not examined further.



Figure 5.4: The SWRC person classes

The *Project* concept, shown in Figure 5.5, has two direct subconcepts named *Development Project* and *Research Project*, only the second one is used for annotations in PAX.

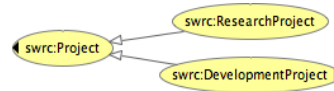


Figure 5.5: The SWRC project classes

For interlinking the presented classes, we are satisfied with the provided object properties. [SBH⁺05]

5.2 The Modified SWRC Ontology

What is not covered in SWRC are cooperations. Projects in PAX can be research projects or cooperations. Therefore the SWRC ontology is extended by a *Cooperation* Class. The resulting project subclasses are shown in Figure 5.6.

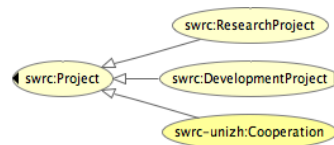


Figure 5.6: Extension of the SWRC project class

In our Web application projects (of any kind) can be associated to partners and funding organizations (which are treated both as *Organization*). SWRC provides an object property called *financedBy* to do the connection to the funding organization, so we add another property called *hasPartner* to connect a project to a partner.

Chapter 6

Implementation

In this chapter we discuss the implementation of the PAX Web application. The acronym PAX stands for "a semantically annotated *Publication* database using *AjaX*". It is built by the use of symfony 1.0.0 beta 1, which integrates scrip.aculo.us 1.7.0 beta 1 and builds itself on PHP5. PAX follows the Model-View-Controller design pattern (described in [GHJV95]). Figure 6.1 shows an overview of the central components of the architecture. On top the different views can be seen. For composing these views, the controller classes are needed. For recurring tasks helper classes are used by the controller. The access to the controller is ruled by the user management, which is part of the controller as well. On the lowest layer the model is responsible for data access of the controller classes. The database used is a MySQL database, but it can be replaced by another database supported by symfony.

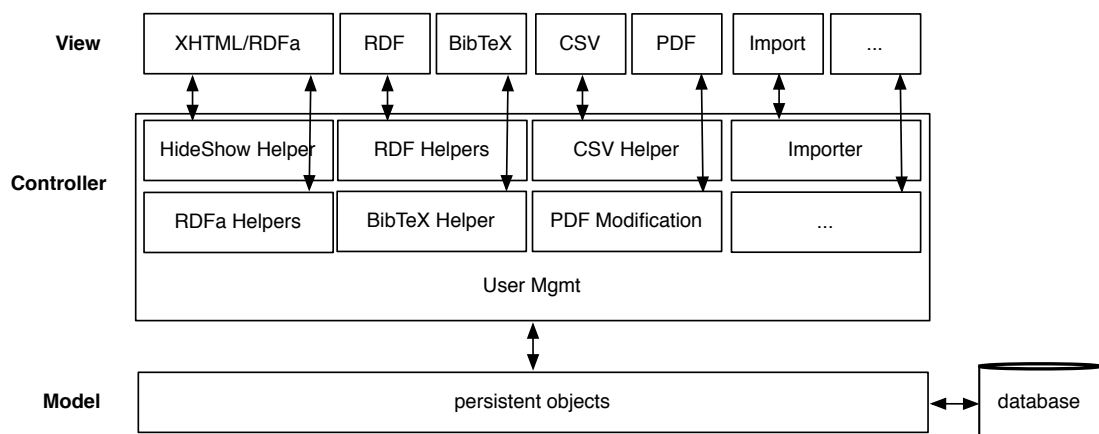


Figure 6.1: The architecture of PAX

The following sections are organized as follows. First an overview of the views for each module is given and the access to the views via URL is explained, then the implementation of filters is explained. After looking at the underlying user management, the generation of the corresponding output formats RDFa, BibTeX, and CSV is presented. Afterwards, the handling and modification of PDF files is examined as well as the special treatment of technical reports. In the proceeding section the BibTeX import function is explained and at the end of this chapter the Web interface

(using Ajax) is presented and explained.

6.1 Modules and Views

The modules in PAX are: author, employee, mail, partner, project, publication, publisher, and group. The `mail` module has no view because it is only used in the background of the application as described in Section 6.3.7. The other modules have the following views:

- All the modules have a `list` view and an `edit` view, requiring an `id` as parameter for their objects. These are the views generated for CRUD mentioned in Section 2.1.4 and their output consists only of XHTML.
- The `project` and the `publication` modules have a `listrdfa` view, which is the same as the `list` view but with RDFa included. Individual objects have a `show` view, which shows an individual publication respectively project with RDFa always included.
- All modules except `employee` have a `rdfall` and a `rdf` view for the individual instances.
- In addition `project` and `publication` have a `bibtexall` (all objects) and a `bibtex` view (individual objects).
- The `publication` module has an `import` view which is described in Section 6.7.
- The `employee` module has a `login` view which is described in Sections 6.3.2 and 6.8.1.

The URLs in PAX are organized as follows: The basic path in all URLs is `/pax/web/index.php/`. It is followed by the module name, the specific view name and, if necessary, by additional parameters. E.g. the `show` view of the project with the `id` 1 is available at `http://example.com/pax/web/index.php/project/show/id/1`.

Redirections

The following views are redirections:

- The `pdf` view of an individual publication is redirected to the attached PDF file.
- The `publications` view of a project is a redirection to the `list` view of the publications, filtered for the concerned project.
- The `publications` view of an employee is a redirection to the `list` view of the publications, filtered for the author/editor whose `id` is passed as parameter.
- The `publications` view of a group is a redirection to the `list` view of the publications, filtered for the concerned group.
- The `csv` view of a group is a redirection to the `csvall` view of the publications, filtered for IFI, the concerned group and the last year.
- The `ifi` view of publications is a redirection to the same view but filtered for the publications published at IFI.

The following URLs are shortcuts:

- `/` points to `publication/list`.

- `/login` points to `employee/login`.
- `/symfony/login` points to `employee/login`.
- `/ifi` points to `publication/ifi`.

RDF/OWL Resource Identifiers

Additionally each resource of each module is not only reachable via `{module}/id/{id}`, but also via `{module}/id/instance{id}`, e.g. `publication/id/instance27`. This is because of the required RDF/OWL output explained in Section 6.4.2: in RDF an instance name cannot begin with a number.

6.2 Filtering

As described in Section 4.3.1, the difference between searching and filtering in PAX is in the type ahead support for the sooner - it can be seen in Section 4.5.1. The filters for *title*, *abstract*, *publication type*, *type*, *projects*, *groups*, *publisher* and *published at ifi* are filters which influence the result in the sense that only the publications which fulfill the criteria are taken into account. The value of *authors/editors* is a criterion which has to be fulfilled by the author, by the editor or by both. Another special filter is *year*. A field *from* and a field *to* can be filled in. The first field restricts the results to a minimum, the second to a maximum year.

6.3 User Management and Credentials

The symfony framework offers simple mechanisms to identify a user, to manage his credentials and to restrict access to certain parts of the application to authenticated users with certain credentials only [Symb]. How these mechanisms are used in PAX is shown in the following sections.

6.3.1 Storing Passwords

For encrypting password storage the *Secure Hash Algorithm* (SHA-1) is used. During the storage of the password, a *salt* is generated and stored. The salt and the chosen password are used to set the SHA1 password and store it into the database.

6.3.2 Login and Authentication

After entering name and password, these values are validated, which means that they must match an entry in the database. If the login exists and the encrypted password matches as well, the user is set to be authenticated. Depending on having administrator rights or not, his credentials are set to *'admin'* or *'employee'*. The user interface during the login process is described in Section 6.8.1.

6.3.3 Access Restriction

With these credentials, for every action in the controller (notice that a view requires at least one dedicated action to be processed) access can be restricted depending on the credentials of the current user. The required restrictions were described in Section 4.3. Each action can provide access to administrators only, to administrators *and* employees or to anybody.

6.3.4 Logout

On logging out, the user is set to be not authenticated anymore and credentials are removed.

6.3.5 Changing Passwords

The authenticated user has the possibility to change his password. Therefore he has to enter his old password, and twice a new one, so that mistyping can be prevented.

6.3.6 Resetting Passwords

For the case an employee forgets his password two possibilities are provided. One is a password reset by the administrator, the second is a password reset by the employee himself. The administrator can open the `edit` view of the employee, check *"send new password on saving"*, and save the employee again. The employee itself can click on *"Forgot your password?"* in the login area. By doing this he is offered the possibility to enter his e-mail address (which therefore has to be unique in the system). In both cases a password is generated randomly and sent to the stored e-mail address of the employee.

6.3.7 Mail

PAX uses `sendmail` [Sen], an open source UNIX application, to send e-mails. The e-mails are sent in XHTML format with an image embedded, a plain text version is added as well. For this reason the application also includes the `mail` module, mentioned in Section 6.1.

6.4 Output Formats

6.4.1 Bibtex Generation

The BibTeX generation is placed in the `publication` module. Its view does omit the basic layout with title, search fields et cetera and its mime type is *text/plain*. The view makes use of the `bibtexHelper` class. The output follows the BibTeX standard, where first an `@` indicates the beginning of a new entry. The cite key is composed by the word *"instance"* and the id of the publication. Each intended field is checked for being filled in and written out if existing. Then all additional BibTeX entries are displayed and in the end the creation and modification dates are added.

```
@inproceedings{instance2,
  address = {Athens, Georgia, USA},
  author = {Gerald Reif and Martin Morger and Harald Gall},
  keywords = {Semantic Desktop},
  month = {November},
  title = {Semantic Clipboard},
  year = {2006}}
date-Added = {2006-09-29 10:13:27 +0200},
date-Modified = {2006-09-29 10:14:27 +0200}}
```

Listing 6.1: Example of BibTeX output in PAX

Displaying BibTeX entries for all filtered entries does an iteration over all the delivered publications and concatenates the results in the view.

6.4.2 RDF/OWL Generation

For each resource in PAX an RDF representation is available, moreover an RDF file is offered for all resources of the same type, e.g. all publications, and also for the entire database. The RDF files are generated on demand.

rdfHelper

A class called `rdfHelper` provides the four most used functions:

- A function `rdf_intro()` which returns the *"intro"* of the RDF file, the first lines which are the same in all RDF files in PAX.
- A function `rdf_outro()` which returns the *"outro"* of the RDF file, the last line which is the same in all RDF files in PAX.
- A function `rdfForString($name, $value)` which returns a string like this:
`<swrc:{name} rdf:datatype='&xsd:string'>{URL + value}</swrc:{name}>`
- A function `rdfForResource($name, $rdf_type, $path)` which returns a string like this:
`<swrc:{name} rdf:type='{rdf_type}' rdf:resource='{path}' />`

More rdfHelpers

For each module in PAX a additional Helper called `rdf{Module}Helper` class exists, which offers the RDF parts for the associated resources.

- For the `Group` module, it is called `rdfGroupHelper` and offers the RDF code for associations to the employees, the publications, and the projects.
- The `rdfPartnerHelper` offers the RDF code for associations to projects. Here it must be distinguished if the role of the association is as partner or as funding organization.
- In the `rdfProjectHelper` associations to members, heads, groups, partners, funding organizations, and publications are processed.
- The `rdfPublisherHelper` outputs RDF data for the associated publications.
- The `rdfPublicationHelper` handles the following associated resources: authors, editors, projects, groups, and publishers.

rdfAuthorEditorHelper

A special helper is the `rdfAuthorEditorHelper`: First, it handles the associations, which are those to publications (as authors and as editors) and to projects (as heads and as members).

In the SWRC the distinction between authors/editors and employees is not made, therefore we do not include the employees into the RDF data. For example, an employee did marry (see Section 4.3.6) and therefore his surname changed, so we mark which authors/editors are the same. This is achieved with the help of OWL and the "sameAs" property, which links an individual to another individual, indicating that two URI references actually refer to the same thing. [OWL04b]

```
<swrc:Person rdf:about='http://example.com/authoreditor/instance38'>
...
<owl:sameAs rdf:resource='http://example.com/authoreditor/instance137' />
...
</swrc:Person>
```

If an employee is connected to the concerned author/editor, then metadata for the phone number, the room and the associated groups is generated and attached to the author/editor. (Notice that this data is connected to the *employee*.)

RDF for an Individual Resource

With these helpers, now the RDF data for an individual resource is created. In the `rdf` view of a module the following parts are inserted:

- The common RDF lines of each RDF file in PAX - via `rdf_intro()`
- An opening tag for the concerned resource
- All datatype properties - via `rdfForString()`
- All used resource properties for the resource - via Helper class for the module.
- A closing tag for the concerned resource
- The closing RDF lines - via `rdf_outro()`

RDF for a Module

The RDF data for a module like `project` is straightforward. The view is called `rdfall` and puts together the RDF pieces of all individual resources, with exception of the RDF intro and outro, which only appear once at the beginning and the end.

RDF for the entire database

On demand a dump of the entire database in RDF is generated. The view named `rdfunizh`, which is accessible in the `publication` module, connects all RDF module files into one RDF file.

6.4.3 RDF Inclusion via Link

Like described in Section 2.2.5, an RDF version of the page (except for employee pages) is referenced in the head section of the XHTML document. For individual entries it points to `http://example.com/{module}/rdf/id/{id}` and for the list view to `http://example.com/{module}/rdfall`.

6.4.4 RDFa generation

Like described in Section 2.2.5, the Web pages can include RDF data in the XHTML code directly. This approach, called RDFa, is used in PAX as well.

RDFaHelper

A Helper class called `RDFaHelper` provides the following methods:

- `rdfaForString($name, $property, $value, $labeled)`
- `rdfaForObjects($property, $object_id_name_array, $type, $rel, $module, $labeled)`

The `rdfaForString()` method accepts four parameters. A `name` parameter is used for the label, `property` is used for writing out the datatype property, e.g. `swrc:title`, `value` is the text visible to the user. If `labeled` is set to false, the `div` and the `label` tags are not written out. Generally output is like this:

```
<div class="form-row">
  <label for="{name}">{name}</label>
  <div class="content">
    <span property="{property}" datatype="xsd:string">
      {value}
    </span>
  </div>
</div>
```

The method `rdfaForObjects()` handles output for several resources at once. The parameter `labeled` has the same use as in `rdfaForString()`. `Property` here is used to label the form-row with the property name. For each object in content the RDF predicate is given in `rel` and the type of the RDF object is given in `type`. The path to the object is generated using `module` and the `id` of an individual object, `value` is the name of the object. The `id` and the `value` are stored in `object.idname_array`.

```
<div class="form-row">
  <label for="{property}">{property}</label>
  <div class="content">
    <span rel='{rel}' type='{type}' href='{path to first object}'>
      {value of first object}
    </span>,
    <span rel='{rel}' type='{type}' href='{path to second object}'>
      {value of second object}
    </span>,
    ...{remaining entries in the array}...
  </div>
</div>
```

More RDFa Helpers

Like in RDF generation, each module has its own Helper for generating RDFa/XHTML code for all datatype and all resource properties. The naming scheme follows the pattern `rdfa{Module}Helper`.

6.4.5 CSV generation

The structure of the “comma separated value” output is like that:

```
{title of the category}
{required fields for the category}
{...entries...}
```

The entries are separated by commas, each entry is enclosed in quotation marks. A `csvHelper` class provides the methods for writing out the fields. For example the category “11.2.11 Beiträage in Kongress-, Tagungs- und Workshopbaenden (Proceedings)” has the following fields:

```
authors,title,editors,booktitle,address,publisher,year,pages,isbn,url,
```

An example entry is the following:

```
"Gerald Reif, Martin Morger, Harald C. Gall","Semantic Clipboard",,
"Semantic Desktop and Social Semantic Collaboration Workshop at the
5th International Semantic Web Conference ISWC06",
"Athens, Georgia, US","","","2006","13",,,
```

6.5 Managing PDF Files

6.5.1 PDF Upload and Download

Attaching a PDF to a publication is possible in the `edit` view of the publication. A basic upload functionality is given by the framework, but no possibility for dynamic upload paths is given. It is a requirement to keep the PDF file name on upload, so we decided to organize the PDF storage with a well defined folder structure. The PDF directory for publication with the id `x` for example is:

```
web/uploads/pdf/publication/{x}/{filename}.pdf
```

With this structure, there is no problem if two PDF files have the same name and it is flexible for an extension to attach other files to a publication. On upload the file name is checked for not allowed characters like “ ”, “.” or “?”, they are replaced with “_” or removed.

Downloadability is managed as follows. Three conditions have to be given for a download icon being visible and making download possible for the user: (1) a PDF file must be attached, (2) the label which marks the general availability for download must be set, and (3) the download date (if filled in) must be reached. With this three conditions fulfilled, the PDF is downloadable.

The proceeding section is about technical reports. As described in Section 6.6.1, for technical reports the PDF file is modified on upload.

6.6 Managing Technical Reports

Technical reports are publications which did not appear in any of the other forms (book, inproceeding etc.), but their content has to be accessible publicly anyhow. In PAX they are treated basically like the other publications, with some additional properties. When *techreport* is chosen in the menu, the following happens:

- A sized warning appears, saying that saving cannot be undone.
- The mandatory and optional fields are shown and hidden as described in Section 6.8.2.
- A “*technical report*” section is shown.

The “*technical report*” section looks different for employees and administrators. For employees a preview of the automatically generated “number” is shown, in administrator layout a “number” can optionally be typed in manually, e.g. for technical reports from previous years. Typing in something into this field suppresses the automatic generation and therefore takes this value. As IFI-`{year.xxxx}` it is not a number, and since the field must be very flexible to acquire any character string possible, it cannot be treated as a number in the database. On the other side, to generate new numbers the highest existing number must be detectable.

In PAX two fields are used for handling this: `techreportnumber` stores only numbers, while `techreportstring` stores only strings. On typing in a self-chosen string, this one is stored in `techreportstring` while `techreportnumber` is left empty. Otherwise a number is requested from the model. If it is the first number requested for the current year, it will have the year first and 0001 at the end (e.g. 2007-0001), otherwise it is the highest assigned number increased by one. The same value is stored into `techreportstring` but with a leading “IFI-” and a point after the fourth cipher, e.g. IFI-2007.0001. The `techreportnumber` is used to determine the highest existing number only.

Employees can edit a technical report the same day only, administrators have no restriction in doing this.

6.6.1 PDF Modification

For technical reports the PDF file is modified. After uploading the PDF file, PAX does the following in the background:

1. Open LaTeX template, replace placeholders with values.
2. Copy PDF file to `tex/` directory.
3. Typeset LaTeX including PDF file.
4. Copy PDF file and log file back to original directory.
5. Remove PDF file and temporary files in `tex` directory.

In the first step a provided LaTeX template is loaded. This template is stored permanently in `web/tex/template.tex` and looks like this:

```

\pdfoutput = 1
\documentclass[formal]{techReport}

\usepackage{ifi}
\usepackage{pdfpages}

\reportType{Technical Report} % headline for all technical reports
\reportNo{TECHREPORTNUMBER} % tech report number generated by PAX
\title{TITLE} % title of the report - mandatory
\dateCreated{CREATION-DATE} % mandatory
\dateModified{MODIFICATION-DATE} % optional (only when report updated)
\author{AUTHORS} % authors of the report separated by '\ '
\abstract{ABSTRACT} % abstract - optional
\group{GROUPS}
% group that published the report. If there is more than one group,
% the groups are separated by '\ ' (group information currently not displayed)

\begin{document}
\maketitle

\includepdf[pages=--]{FILENAME}
% pdf file name of the uploaded file. Will be added to the generated title page.
\end{document}

```

Listing 6.2: The LaTeX template for PDF modification in PAX

The strings written in uppercase letters are the placeholders which are replaced by the values of the current publication, e.g. GROUPS could be replaced by *Software Evolution And Architecture Lab*. The MODIFICATION-DATE is only replaced if the modification date of the publication is different to the creation date. The second step is copying the PDF file from `web/uploads/pdf/publication/{x}/{filename}.pdf` to `web/tex/{filename}.pdf`. This step is necessary because of the following line in the template:

```
\includepdf[pages=--]{FILENAME}
```

This command is part of step 3 - the typesetting - and inserts the uploaded PDF file into the generated PDF file. Now the original PDF file has two cover pages prepended. In the forth and fifth step the modified PDF file overwrites the uploaded one, a file called `{filename}.log` is copied to the upload directory as well and in `tex/` all files beginning with `{filename}` are deleted.

6.7 BibTeX Importer

The import function is located in the `publication` module and has its own view. There the employee has the possibility to choose some general options for the imported publications. It can be chosen if they were published at IFI and which research groups they are associated to. A BibTeX file can be uploaded through a provided function of the framework, or a URL can be entered into an input field.

In the controller the following happens: if a URL was entered, it is tried to fetch the content from that URL and write it into a temporary file. If a file is uploaded the content is stored into an array with each line as an entry.

For parsing the BibTeX entries, the `biblio` module from drupal [Dru] is used. It returns an array in which one entry corresponds to one publication. It is checked if a title exists, if the title does not exist in the database yet and if the publication type is a supported one (all except techreport). Afterwards, the returned entries are checked for umlauts and white spaces at the beginning or end of the string. The white spaces are removed and the umlauts are replaced as shown in Appendix B. The entries *data-added* and *date-modified* are removed, because this is handled by PAX itself. The *title*, *publication type*, *year*, *abstract*, and *type* are stored directly. Storing *authors* requires some more efforts. The existing authors/editors are stored into an array at the beginning of the import action and now for every potential new author/editor it is checked if he corresponds to one in this array. Matching requires an exact match of first name *and* surname. The second possibility is an exact match of the surname while the first name in the import file or the database is abbreviated. This is assumed to be the case if the first name ends with a *."*. *Editors* are treated the same way. *Publishers* match if the name of the new publisher is found at the beginning of an existing one. Therefore *"Press"* and the german counterpart *"Verlag"* are removed for this comparison and then a *"LIKE"*-query is executed. If, for example, the imported publication has a publisher entry with *"Springer Verlag"* as its value, then *"Springer"* is compared to the database entries and therefore matches *"Springer"* or *"Springer Verlag"*, no matter in which form it is stored. Mismatches are stored in designated fields. After processing these entries, the remaining entries are stored as usual BibTeX fields with a name and a corresponding value.

Afterwards, the `import` view is displayed again for new imports, now the uploaded data is displayed and upload errors (like unsupported publication type) are displayed.

Via *"Show my last import in list..."* the employee has the possibility to filter the `list` view of the `publication` module for the last imported entries. This requires an additional label for each publication, which is set to `false` for older publications and `true` for the newly added on import. In `list` view an icon indicates the entries with remaining mismatches, the employee has to add these entries by hand. This is described further in Section 6.9.

6.8 Ajax User Interface

One requirement described in Section 4.5 was to provide an easy-to-use interface comparable to desktop applications. In Figure 6.2 the article input mask is shown and in the following the AJAX parts of the UI are described.

6.8.1 Login/Sign out

For logging in, neither the user does have to request a separate page, nor the login part constantly uses a fixed space of user interface. Instead, if the user clicks on *"login"*, a login area appears. Here the user can enter name and password or click if he forgot his password. Pressing *"cancel"* makes the login area disappear.

After logging in, the *"login"* button is replaced by a *"sign out"* button. Additionally a button named *"change my password"* is placed as well as a line called *"show my publications"*. The later points to `employee/publications/id/{id}`.

Publication Type:	article
Academic Report Type:	do not consider value is chosen automatically when selecting 'publication type' - changes are allowed
Abstract:	
Title:	
type:	
year:	
BibTex	
journal:	
pages:	
note:	
volume:	
number:	
month:	
Add field...	
Relations	
Authors:	Publication has to be saved before authors can be added... save
Editors:	Publication has to be saved before editors can be added...
Projects:	Publication has to be saved before projects can be added...
Research Groups:	☐ Software Engineering and Architecture Lab (SEAL) ☐ Dynamic and Distributed Information Systems (DDIS) ☐ Database Technology (dbtg) ☐ Requirements Engineering (REQ) ☐ Computational Linguistics (cl) ☐ Visualization and MultiMedia Lab (vmml) ☐ Artificial Intelligence (aialab) ☐ Information Management (im) ☐ Communication Systems (csg) ☐ Educational Engineering (ee)
Publisher:	Publication has to be saved before publishers can be added...
More	
PDF Download date:	
PDF:	Browse... optional PDF upload
show PDF:	☒ PDF is displayed to page visitors
publish PDF:	☐ it is allowed to publish the PDF on another page of the university
published at IFI:	☒ published at IFI
Creator	
Creator:	 save

Figure 6.2: Input mask for articles

6.8.2 Mandatory and Optional BibTeX fields

Depending on the selected publication type, BibTeX fields are displayed. For example a manual has `title` as a mandatory field and the following as optional fields: `year`, `author`, `organization`, `note`, `month`, `note`, `address`, and `edition`. `Title` and `year` are always visible in PAX, the first is required and the second is used for categorization, so the employee can always fill it in. The `authors` are handled separately as well as the `organization`, which is the research group in PAX. The remaining fields are `note`, `month`, `address`, and `edition`, as shown in Figure 6.3.

BibTeX	
note:	
month:	
address:	
edition:	
Add field...	

Figure 6.3: Mandatory and optional BibTeX fields for manuals

On changing the publication type, first all not filled in BibTeX fields are hidden and then all mandatory and required fields for the chosen publication type are shown. Since only the not filled BibTeX fields are hidden, all already filled fields and additional fields created by the employee are displayed as well. For each publication type there exists a JavaScript function in the XHTML code. The necessary functions are created by the `HideShowHelper` class.

6.8.3 Managing Custom BibTeX fields

Via an *"add field..."* button additional fields are shown by JavaScript. Their name and value can be chosen, as shown in Figure 6.4. The save action in the controller accesses these fields by a unique name, therefore they are all created on requesting the page, but only made visible via the *"add field..."* button. We set the number of possible additional fields to 15 in the `edit` view. If more fields are needed, they are available after saving the publication with the first 15 fields filled in.

edition:	
institute	IFI
Add field...	

Figure 6.4: Adding custom BibTeX fields

6.8.4 Type Ahead

Another feature is type ahead in input, search and filter fields. E.g. typing in "bern" in the author field, has to show a list with a predefined maximal number of matching authors/editors (set to five in PAX), like shown in Figure 6.5. These entries can be chosen via tabulator. The process in the background is the following, according to Section 2.1.2:

1. A request to `publication/autocompleteauthor?author={authoreditorname}` is sent.
2. On the server the model is accessed to deliver the (alphabetically) first five matching authors.
3. An XML fragment containing the found author/editor names is responded from the server and processed by the Ajax engine, which displays the answers as a drop down menu.

Figure 6.5: Type ahead in the author field

6.8.5 Adding Elements

After choosing an author/editor, pressing the "add" button adds the author/editor to the publication as an author, the new author is displayed immediately in the GUI. This is shown in Figure 6.6.

Figure 6.6: Adding a new author to a publication

Here the process is the following:

1. The employee clicks on the "add" button.
2. A request to `publication/authoraddajax?author={authorname}?id={id}` is sent.
3. On the server it is checked if the author/editor is not added yet as an author. If not, it is connected to the publication as an author.
4. An XML fragment containing an id and a message is responded from the server and processed by the Ajax engine, which inserts the author into the DOM and displays the message.

6.8.6 Creating New Elements

For adding new authors the *"add new external author"* button has to be clicked. Via JavaScript a first name, a surname, and a distinction field are displayed as well as a *"cancel"* and a *"save and add"* button. (See Figure 6.7) If all entries are filled and the author/editor does not exist yet in the database it is added to the database and the DOM as follows.



Figure 6.7: Creating an external author

1. The employee fills in the necessary fields and clicks on *"save and add"*.
2. On the client it is checked if the necessary fields are filled in.
3. A request is sent to
`publication/authorcreate?name={name}?surname={surname}?distinction={distinction}`.
4. On the server it is checked if the author/editor does not exist yet; if not, it is created via model.
5. Still on the server, the author/editor is connected to the publication as an author.
6. An XML fragment is responded from the server and processed by the Ajax engine, which inserts the author into the DOM.

6.8.7 Removing Elements

For removing an author, the employee has to click on the icon left to the author. The author fades away and is removed from the DOM. Removing an author happens as follows:

1. The employee clicks on the icon to remove an author.
2. A request to `publication/authordel?id={id}` is sent.
3. On the server the association between the author and the publication is removed via model.
4. The author/editor's id is responded as confirmation for success in removing the author.

6.8.8 Sorting Elements

As described in Section 2.1.4, the underlying Ajax framework, Prototype, provides a `Sortable` class. This class is used for sorting authors via drag and drop. (See Figure 6.8)

1. The employee drags an author to another position in the author list and drops it.
2. A request is sent to
`publication/authorsort?order[]={first_id}?order[]={second_id}?order[]={third_id}`.
3. On the server a method in the model is invoked to do the sorting in the database.

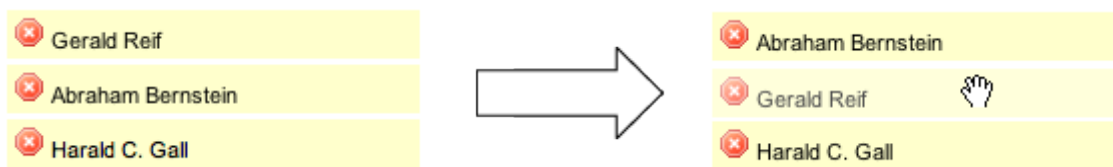


Figure 6.8: Sorting authors

6.8.9 Dependent Drop-Down Menu

The drop-down menu for the *academic type* is dependent on the value chosen in the *publication type* menu. With the use of `HideShowHelper`, a pre-built JavaScript function handles this automatic selection. The selected value can be changed manually.

Publication Type:	phdthesis
Academic Report Type:	11.1.3 Dissertationen

value is chosen automatically when selecting 'publication type' - changes are allowed

Figure 6.9: Dependent drop-down menu

6.9 Handling Import Errors

In Section 6.7 it is described that not recognized authors, editors, and publishers are stored separately and that the employee has to create them by hand. In Figure 6.10 it is shown how this is indicated in the `edit` view of the publication. The exact author entry from the BibTex field is displayed. In the example shown, the author/editor *Christian Kaiser* is not existing in the database. The employee can add it manually via *"create new external author"* and afterwards click on *"clear"*. This sends a request to `publication/clearauthor?id={id}`, where `{id}` is the id of the current publication. On the server this clears the storage for the not recognized authors, and on the client the *"not added"* section is hidden via JavaScript.

The screenshot shows a web interface titled "Relations". It is divided into three main sections:

- Authors:** A list of three authors, each with a red 'x' icon in a yellow box:
 - Abraham Bernstein
 - Esther Kaufmann
 - Christoph Kiefer
- add author:** A text input field followed by a green "add" button with a plus icon. Below the input field is a green button labeled "create new external author".
- not added:** A section with a yellow warning triangle icon and the text: "Some of the authors were not found in the database during the import - please create them by hand". Below this text is a yellow box containing the text: "Bernstein, Abraham and Kaufmann, Esther and Kaiser, Christian and Kiefer, Christoph". At the bottom of this section is a red "clear" button with an 'x' icon.

Figure 6.10: Import error

6.10 Data Model

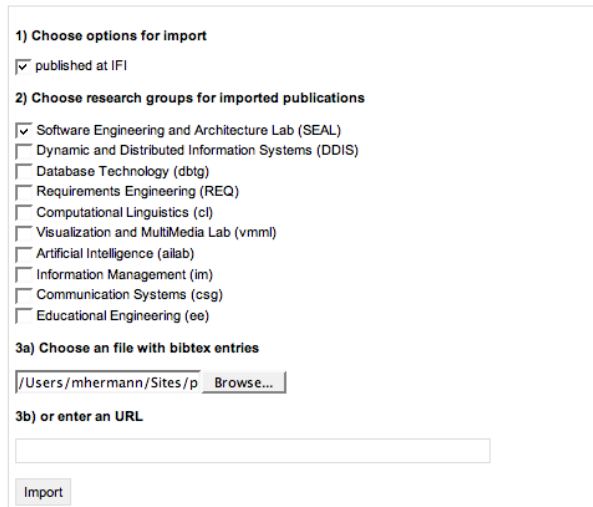
To speed up the application and reduce the number of queries executed, a high amount of redundancy was added to the `publication` and the `project` modules. The resulting data model is attached in Appendix C.

Evaluation

PAX and its user interface were tested by the SEAL group at IFI by acquiring all their publications and projects. The found bugs were fixed.

7.1 Import

In the following the `import` view is shown and explained. The options which can be chosen by the user are shown in Figure 7.1, he can choose if the publications were published at IFI and in which groups. Below a URL can be entered or a file to upload can be chosen.



The screenshot shows a dialog box titled "1) Choose options for import". It contains three sections:

- 1) Choose options for import**
 - ☒ published at IFI
- 2) Choose research groups for imported publications**
 - ☒ Software Engineering and Architecture Lab (SEAL)
 - ☐ Dynamic and Distributed Information Systems (DDIS)
 - ☐ Database Technology (dbtg)
 - ☐ Requirements Engineering (REQ)
 - ☐ Computational Linguistics (cl)
 - ☐ Visualization and MultiMedia Lab (vmml)
 - ☐ Artificial Intelligence (ailab)
 - ☐ Information Management (im)
 - ☐ Communication Systems (csg)
 - ☐ Educational Engineering (ee)
- 3a) Choose an file with bibtex entries**
 - Text input: `/Users/mhermann/Sites/p` with a "Browse..." button.
- 3b) or enter an URL**
 - Empty text input field.

At the bottom is an "Import" button.

Figure 7.1: Options for BibTeX import

The results of the import are the ones shown in Figure 7.2. First it is summarized how many publications were imported and how many were not and then the reasons for failures are listed. A button can be clicked to view all imported publications in `list` view. In the lower part of the window the imported file is listed entirely.

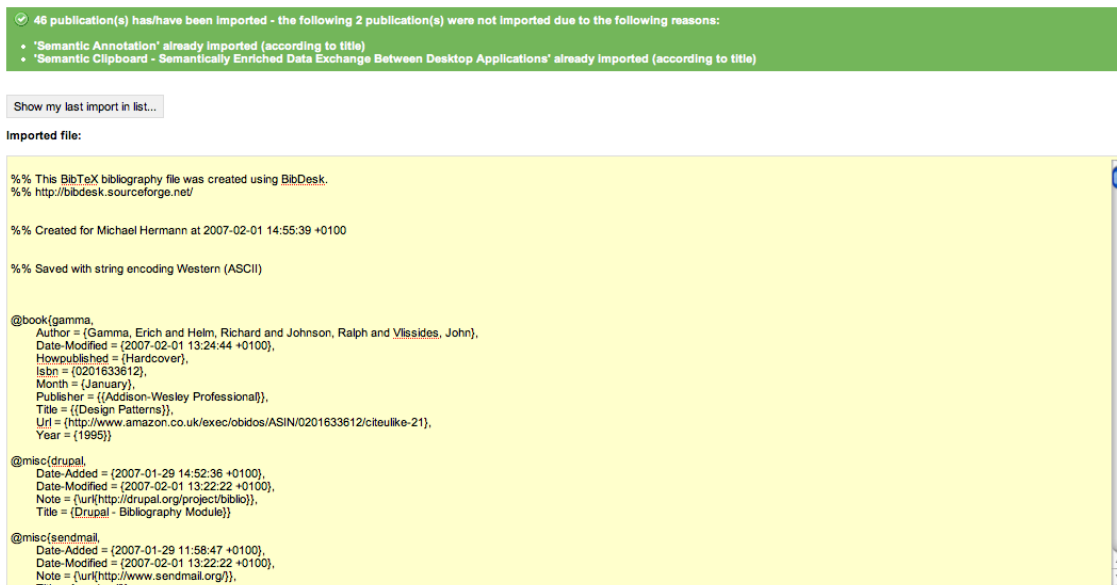


Figure 7.2: The result of an import

7.2 Modules

The usability of the user interface has been tested as well by the SEAL group at IFI. In the following the major parts of the PAX UI are shown and described. The `list` view of publication (Figure 7.3) shows all the publications. On top left the navigation is accessible, on top right there is the "login" button. On the right side the search fields and filters are shown. Each publication has icons assigned for accessing RDF, PDF, and BibTeX output.

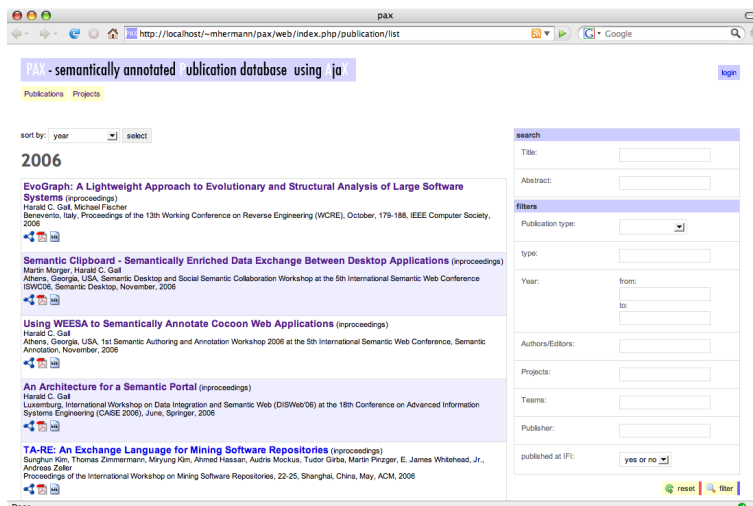


Figure 7.3: Publication list

A filtered list view of publications can be seen in Figure 7.4. Furthermore a employee is logged in, therefore more options are available. He can edit the publications he is allowed to. He can change his password, add a new publication, or import publications, as seen in the upper right corner.

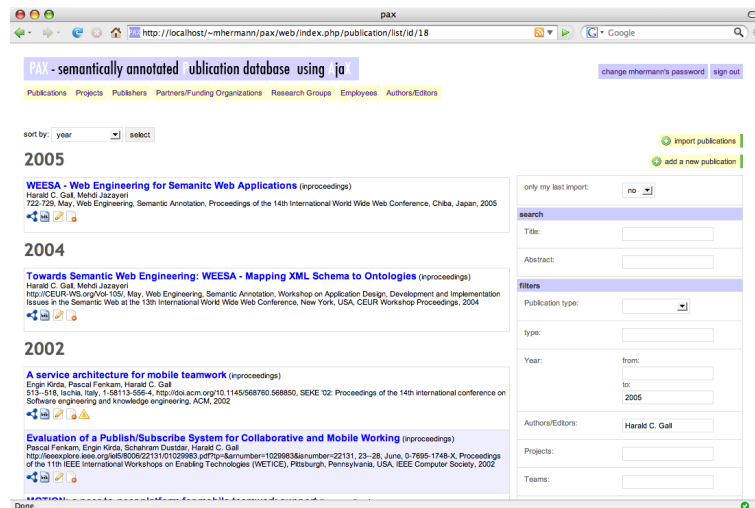


Figure 7.4: Filtered Publication list

The show view of publication (Figure 7.5) displays a single publication with all stored entries. The page is annotated in RDFa in the background.

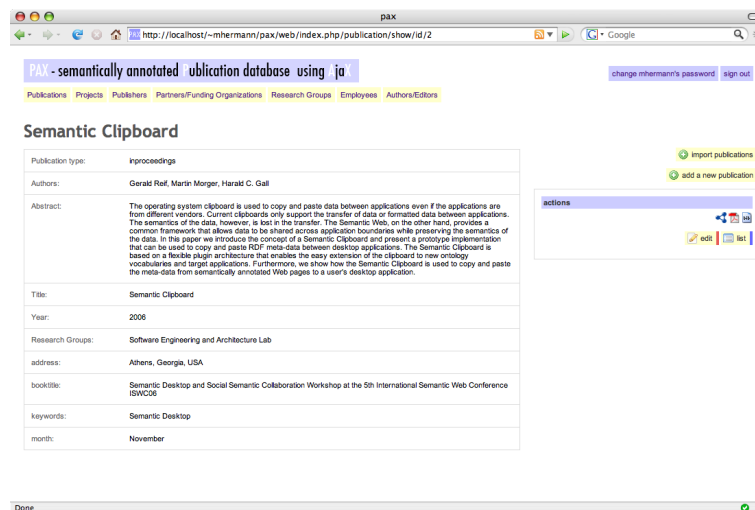


Figure 7.5: Show a single publication

The `edit` view of publication, shown in Figure 7.6, offers the interface for modifying a publication. The lower part of the page, which is hidden in the figure, manages author, editors, projects and more. Its parts were already shown and explained in Section 6.8.

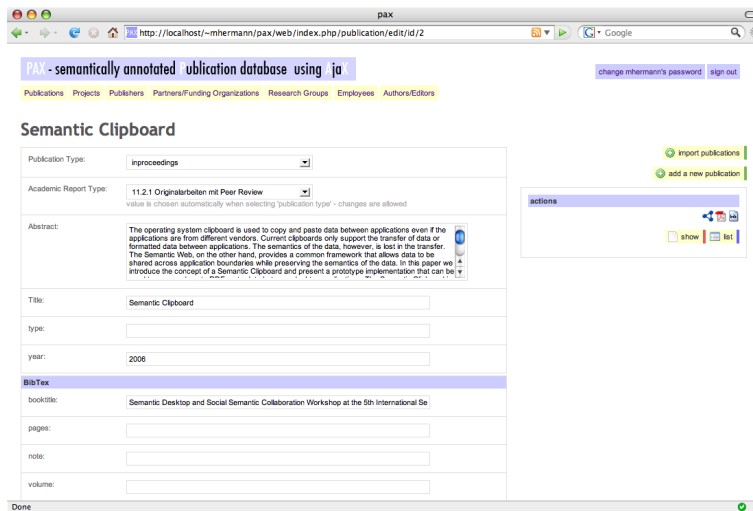


Figure 7.6: Editing a publication

The `rdfalist` view of projects (Figure 7.7) is analogous to the one of publications. Each project has an RDF icon below its description.

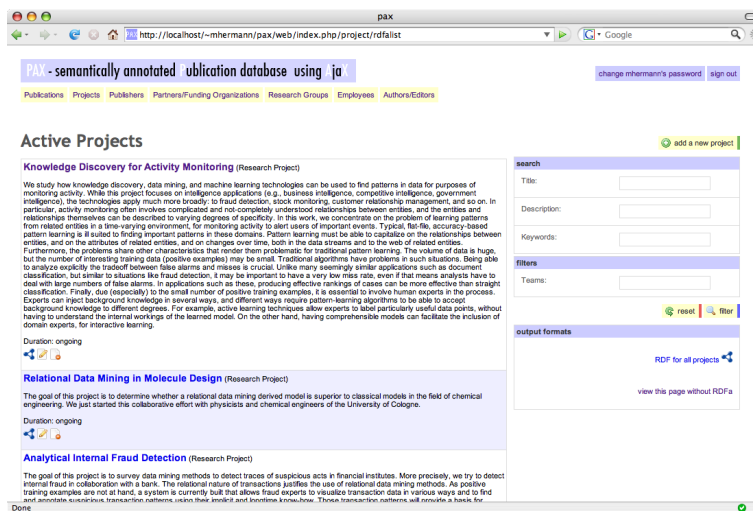


Figure 7.7: Project list including RDFa

The `list` view of publishers, seen in Figure 7.8, is displayed as a table which can be sorted by clicking on row titles.

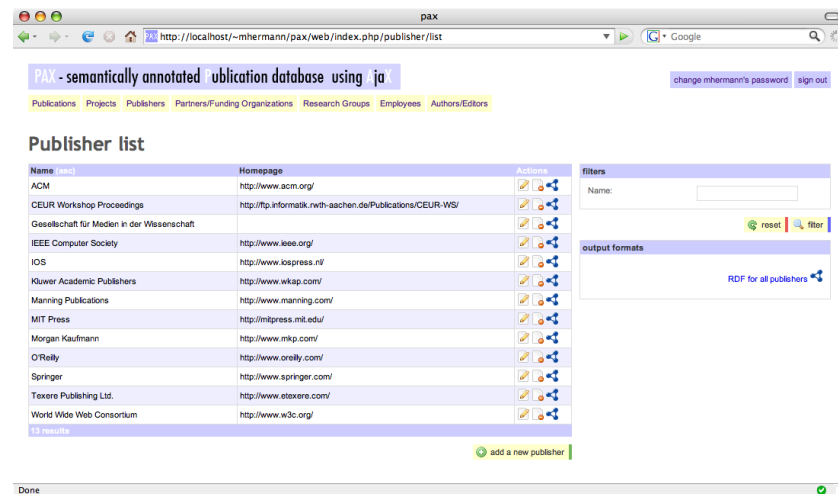


Figure 7.8: Publisher list

The `list` view of partners/funding organizations (Figure 7.9) displays the attached logo of each partner.

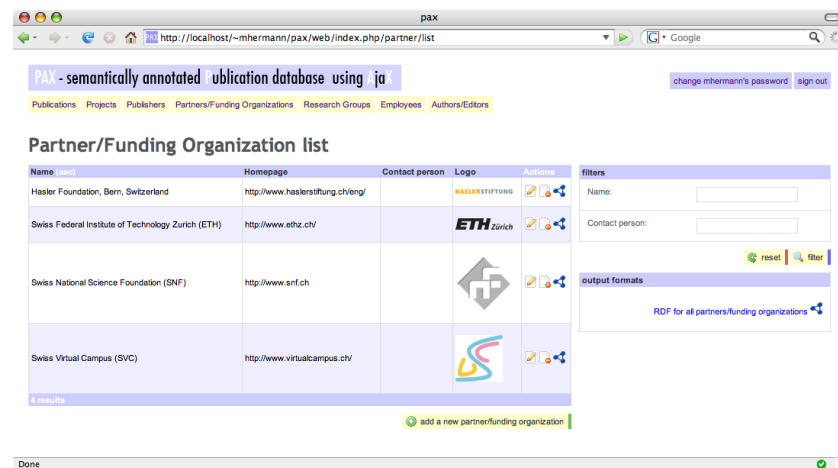


Figure 7.9: Partner list

The `list` view of research groups is shown in Figure 7.10. It offers direct links to the publication list filtered for a single group and as well for CSV output.

Name	Shortname	Direct URL	Homepage	Actions
Artificial Intelligence	alab	alab	http://www.if.unizh.ch/groups/alab/	[Icons]
Communication Systems	csg	csg	http://www.csg.unizh.ch/	[Icons]
Computational Linguistics	cl	cl	http://www.if.unizh.ch/groups/CL/	[Icons]
Database Technology	dtg	dtg	http://www.if.unizh.ch/groups/dtg/index.html	[Icons]
Dynamic and Distributed Information Systems	DDIS	DDIS	http://www.if.unizh.ch/ddis/	[Icons]
Educational Engineering	ee	ee	http://www.if.unizh.ch/ee/index.html	[Icons]
Information Management	im	im	http://www.if.unizh.ch/im/	[Icons]
Requirements Engineering	REQ	REQ	http://www.if.unizh.ch/groups/req	[Icons]
Software Engineering and Architecture Lab	SEAL	SEAL	http://seal.if.unizh.ch	[Icons]
Visualization and Multimedia Lab	vmmf	vmmf	http://www.if.unizh.ch/vmmf/	[Icons]

Figure 7.10: Group list

The `list` view of employees (Figure 7.11) provides direct links to the publication list filtered for the associated authors/editors. The employees which have no direct link, do not have an author/editor associated.

Firstname	Surname	Distinction	Position	Login	direct URL	Author/Editor	Admin	Actions
admin				admin				[Icons]
Abraham	Bernstein		Professor	avi	Abraham Bernstein	✓		[Icons]
Sandro	Boccuzzo		Assistant	boccuzzo	Sandro Boccuzzo	✓		[Icons]
Michael	Daenzer		Assistant	daenzer	Michael Daenzer	✓		[Icons]
Beat	Fluri		Assistant	fluri	Beat Fluri	✓		[Icons]
Norbert E.	Fuchs		Professor	Norbert E. Fuchs		✓		[Icons]
Harald C.	Gall		Professor	gall	Harald C. Gall	✓	✓	[Icons]
Anne	Gähring				Anne Gähring	✓		[Icons]
Emanuel	Giger		Assistant	giger	Emanuel Giger	✓		[Icons]
Stefanie	Hauske		Assistant	hauske	Stefanie Hauske	✓		[Icons]
Michael	Herrmann			mherrmann			✓	[Icons]
Esther	Kaufmann		Assistant	kaufmann	Esther Kaufmann	✓		[Icons]
Christoph	Kiefer		Assistant	kiefer	Christoph Kiefer	✓		[Icons]
Zehra	Kilic		Secretary	kilic			✓	[Icons]
Patrick	Knab		Assistant	knab	Patrick Knab	✓		[Icons]
Jiwon	Li		Assistant	li	Jiwon Li	✓		[Icons]
Samira	Ludi		Secretary	ludi			✓	[Icons]
Jonas	Luell		Assistant	luell	Jonas Luell	✓		[Icons]
Martin	Pötzger		Senior Assistant	pötzger	Martin Pötzger	✓		[Icons]
Gerald	Reif		Senior Assistant	reif	Gerald Reif	✓	✓	[Icons]

Figure 7.11: Employee list

In Figure 7.12 the `list` view of authors/editors is shown. If available, the associated employee is listed.

PAX - semantically annotated publication database using ja

change mhermann's password sign out

Publications Projects Publishers Partners/Funding Organizations Research Groups Employees Authors/Editors

Author/Editor list

Firstname	Surname	Distinction field	Employee	Actions
Ben	Adida			
Grigoris	Antoniou			
Guruduth	Banavar			
Christoph	Burki			
Carlos	Bento			
Tim	Berners-Lee			
Abraham	Bernstein		Abraham Bernstein	
Stephan	Boehdorn			
Andreas	Blumauer			
Sandro	Boccuzzo		Sandro Boccuzzo	
Klemens	Boehm			
Christian	Boehm			
Erik	Brynjolfsson			
Amílcar	Cardoso			
Scott	Clearwater			
Dave	Crane			
Kevin	Crowston			
Monica	Crubezy			
Isabel	Cruz			
Gianpaolo	Cugola			

filters

Firstname:

Surname:

reset filter

output formats

RDF for all authors/employees

Figure 7.12: Author/Editor list

7.3 Output Formats

The annotations have been successfully tested with Protege 3.2, OilEd 3.5.7, and RDF Gravity 1.0. All of them validate correctly against OWL according to the WonderWeb OWL Validator¹. The BibTeX output has been successfully tested with BibDesk 1.3.1.

¹<http://phoebebus.cs.man.ac.uk:9999/OWL/Validator>

Conclusions

Publishing scientific publications on an institute's, research group's, and researcher's homepage (in the Web) was a common problem in academic research. PAX is a Web application built exactly for this purpose. It meets the requirements from IFI and builds up on the symfony PHP framework. Its user interface was built using Ajax and the entire application is annotated semantically with RDF and RDFa.

8.1 Conclusions

After the development of PAX, conclusions can be made in the areas of Web development, Ajax and Semantic Web.

1. Using a Web development framework eases a lot. Common and repetitive tasks are simplified by the framework, errors can be avoided, flexibility is bigger.
2. Using an Ajax framework is inevitable for the Web application to be compatible with different kinds and versions of Web browsers. The usage is facilitated through accessibility to functions in a object-oriented programming style.
3. Frameworks for RDF integration into PHP Web applications do not exist yet. Since RDF is well specified and syntax is straightforward, abstractions can be made, like done in `rdfHelper` (Section 6.4.2). The Helper class abstracts RDF triples and provides methods for generating RDF code easily.
4. The usage of Ajax can reduce data traffic and waiting time for users. Furthermore it makes applications reacting more user-friendly than common Web applications, in a desktop-application-like way.
5. Web user interfaces and frameworks for composing them still can be improved. Well known techniques like keyboard shortcuts, drag-and-drop and dynamic drop-down menus can be integrated better, more easily and by default.

RDF integration directly into XHTML is possible by the use of RDFa which is still a recommendation draft at W3C. During integrating RDFa in PAX the following points stood out:

- While programming according to MVC, different views are allowed. Not each view necessarily displays all available information. E.g. a `list` view of publications cannot display as much information for each publication as a Web page about an individual publication. The underlying RDFa can only annotate the data which is also visible on the Web page.

- As RDFa is for machine use and often for reasoning, the machine needs the complete data, which is distributed over several pages instead of one being available in one file like with RDF.
- For the ease-of-use, a user often is not presented all information available. E.g. a visitor in the WWW does not need a `list` view of the authors/editors. For machines ease-of-use in this case is not a relevant criteria, therefore the authors/editors could be included into the data (even if one is in the database, but has no publication yet). In RDF this is possible, in RDFa it is not.

8.2 Summary of Contributions

In this thesis we described the implementation of the Web application PAX. It is an application for posting publications and projects in the Web. The requirements were numerous: publishers, partners/funding organizations, research groups and authors/editors had to be managed in the background. In addition, user management for employees had to be handled. Different publication types were required to display different fields and react differently on creation, deletion, modification, and import. A BibTeX import function had to be built in as well as PDF Modification for technical reports. For output several file formats were required. Our goal was to implement PAX providing a comfortable, desktop-like user interface by the use of Ajax. Web pages should be annotated with semantic data in RDF format.

The following contributions are made by this thesis:

1. RDF file output was built into a PHP Web application. For each individual resource, for a module, and for the entire database RDF data is generated on demand. Each page contains a link to its RDF representation.
2. RDFa annotation was built into a PHP Web application. The publication list and the project list can be displayed with RDFa included. Instances of publications and projects are always displayed with RDFa included into the XHTML source code.
3. A Web application with a desktop-like user interface was built. Building on symfony and script.aculo.us frameworks, Ajax was used for showing and hiding parts of the Web page, for manipulating the database in the background, for adding, sorting, and removing parts of the Web page and the DOM.

8.3 Future Research

On the metadata generation side there is a need for automated tasks to integrate RDF and RDFa generation into Web development frameworks.

As the Semantic Web is still not in widespread use, there is a huge potential for applications. As ontologies spread more and more, the area for processing the generated metadata is widening.

For example query interfaces like that of Ginseng [BKKK06] could be connected to the PAX database and SWRC-unizh ontology. In this way natural language queries should be possible.

Applications like the Semantic Clipboard [RMG06] could be extended by a plug-in to handle the RDF(a) metadata of PAX. By doing this, for example, employee data could be pasted into an

address book, or a plug-in could be written to transform the publication data to any application format like that of EndNote¹.

With the World Wide Web, the difficult thing was to convince people that it was something they should adopt. As soon as the applications existed, network effects kicked in and spreading went on by itself. With the Semantic Web, early development stages can be compared to those of the WWW, what induces that once a certain threshold is reached, the Semantic Web will be adopted too.

¹<http://www.endnote.com/>

Appendix A

BibTeX Fields

@article	mandatory fields: author, title, journal, year optional fields: volume, number, pages, month, note
@book	mandatory fields: author or editor, title, publisher, year optional fields: volume or number, series, address, edition, month, note
@booklet	mandatory fields: title optional fields: author, howpublished, address, month, year, note
@conference	mandatory fields: author, title, booktitle, year optional fields: editor, volume or number, series, pages, publisher, address, month, note
@inbook	mandatory fields: author or editor, title, chapter and/or pages, publisher, type, year optional fields: volume or number, series, address, edition, month, note
@incollection	mandatory fields: author, title, booktitle, publisher, year optional fields: editor, volume or number, type, series, edition, chapter, pages, address, month, note
@inproceedings	mandatory fields: author, title, booktitle, year optional fields: editor, volume or number, series, pages, publisher, address, month, note
@manual	mandatory fields: title optional fields: author, address, edition, month, year, note
@masterthesis	mandatory fields: author, title, school, year optional fields: address, month, note, type
@misc	mandatory fields: one of the optional fields optional fields: author, title, howpublished, month, year, note

@phdthesis	mandatory fields: author, title, school, year optional fields: address, month, note, type
@proceedings	mandatory fields: title, year optional fields: editor, publisher, volume or number, series, address, month, note
@techreport	mandatory fields: author, title, institution, year optional fields: type, number, address, month, note
@unpublished	mandatory fields: author, title, note optional fields: month, year

All types are allowed a key field, which should be filled in if neither author nor editor is used.

Appendix B

Umlaut Encoding

The following umlauts are handled by PAX in BibTeX in- and output:

umlaut	enconding
ä	{\ "a}
ö	{\ "o}
ü	{\ "u}
Ä	{\ "A}
Ö	{\ "O}
Ü	{\ "U}
é	{\ 'e}
è	{\ 'e}
à	{\ 'a}
ß	{\ ss}

Table B.1: Umlauts handled by PAX

The Underlying Data Model

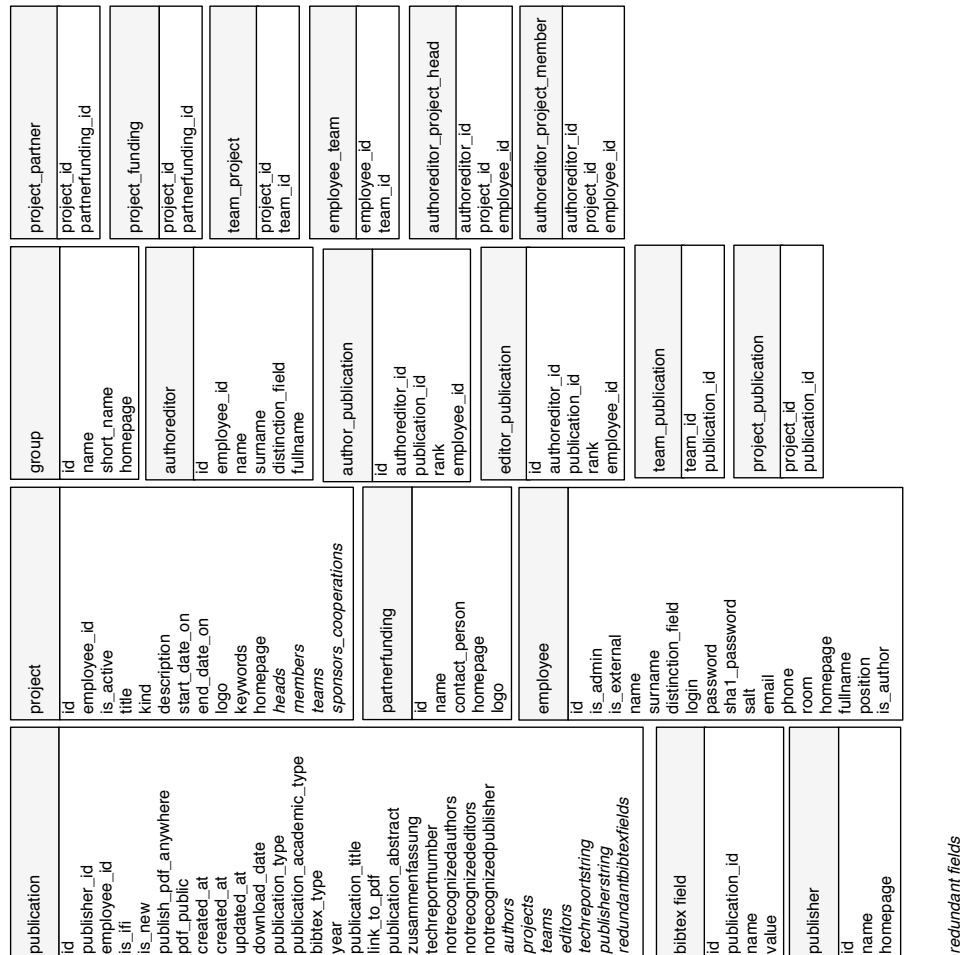


Figure C.1: The data model used in PAX

References

- [Adi06] Ben Adida. People Writing about RDFa, May 2006. <http://rdfa.info/2006/05/29/people-writing-about-rdfa/> (Last visited 09.02.2007).
- [ATF] ATF Project - An Eclipse Web Tools Incubator Project. <http://www.eclipse.org/atf> (Last visited 09.02.2007).
- [AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer (Cooperative Information Systems)*. MIT-Press, April 2004.
- [Bas] Basilic: Automated Bibliography Server. <http://artis.imag.fr/Software/Basilic> (Last visited 09.02.2007).
- [BKKK06] Abraham Bernstein, Esther Kaufmann, Christian Kaiser, and Christoph Kiefer. Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies. In *2006 Jena User Conference*, May 2006.
- [BL99] Tim Berners-Lee. *Weaving the Web*. Texere Publishing Ltd., November 1999.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284:34–43, May 2001.
- [CPJ05] Dave Crane, Eric Pascarello, and Darren James. *Ajax in Action*. Manning Publications, October 2005.
- [CSS06] Cascading style sheets home page, 2006. <http://www.w3.org/Style/CSS/> (Last visited 09.02.2007).
- [DCa] Dublin core element set v1.1 namespace. <http://purl.org/dc/elements/1.1/> (Last visited 09.02.2007).
- [DCb] Dublin Core Terms. <http://purl.org/dc/terms/> (Last visited 09.02.2007).
- [Doj] Dojo, the JavaScript Toolkit. <http://dojotoolkit.org/> (Last visited 09.02.2007).
- [Dru] Drupal - Bibliography Module. <http://drupal.org/project/biblio> (Last visited 09.02.2007).
- [DWR] Direct Web Remoting. <https://dwr.dev.java.net/> (Last visited 09.02.2007).
- [FOA] FOAF Vocabulary Specification. <http://xmlns.com/foaf/0.1/> (Last visited 09.02.2007).

- [Gar05] Jesse James Garrett. Ajax: A New Approach to Web Applications, February 2005. <http://adaptivepath.com/publications/essays/archives/000385.php> (Last visited 09.02.2007).
- [GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley Professional, January 1995.
- [Got06a] Hari K. Gottipati. Google is not the leader in Ajax applications, August 2006. http://www.oreillynet.com/xml/blog/2006/08/who_is_the_leader_in_ajax_appl.html (Last visited 09.02.2007).
- [Got06b] Hari K. Gottipati. Survey says Prototype is the most used Ajax toolkit/framework, July 2006. http://www.oreillynet.com/xml/blog/2006/07/whats_the_best_ajax_toolkitfra.html (Last visited 09.02.2007).
- [GWT] Google Web Toolkit (Beta). <http://code.google.com/webtoolkit/> (Last visited 09.02.2007).
- [HJSS06] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A social bookmark and publication sharing system. In Aldo de Moor, Simon Polovina, and Harry Delugach, editors, *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, Aalborg, Denmark, July 2006. Aalborg University Press.
- [HS04] Jens Hartmann and York Sure. An Infrastructure for Scalable, Reliable Semantic Portals. *IEEE Intelligent Systems*, 19(3):58–65, May 2004.
- [Kop96] Helmut Kopka. *LaTeX, 3 Bde., Bd.1, Einführung*. Addison Wesley Verlag, January 1996.
- [LMS06a] Christoph Leisegang, Stefan Mintert, and Bastian Spanneberg. Äpfel und Birnen. *iX*, 8:54–61, August 2006.
- [LMS06b] Christoph Leisegang, Stefan Mintert, and Bastian Spanneberg. Die Nullnummern. *iX*, 9:66–74, September 2006.
- [Lor06] Laura Lorek. Users, Money Shifting to Participatory Web 2.0, December 2006. <http://www.rismedia.com/wp/2006-12-19/users-money-shifting-to-participatory-web-20/> (Last visited 09.02.2007).
- [Mah06] Michael Mahemoff. *Ajax Design Patterns*. O'Reilly, June 2006.
- [MDC07] AJAX:Getting Started, February 2007. http://developer.mozilla.org/en/docs/AJAX:Getting_Started (Last visited 09.02.2007).
- [OAI] Open Archives Initiative. <http://www.openarchives.org/> (Last visited 09.02.2007).
- [O'R05] Tim O'Reilly. What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software, September 2005. <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html> (Last visited 09.02.2007).
- [OWL04a] OWL Web Ontology Language Overview, February 2004. <http://www.w3.org/TR/owl-features/> (Last visited 09.02.2007).

- [OWL04b] OWL Web Ontology Language Reference, February 2004. <http://www.w3.org/TR/owl-ref/> (Last visited 09.02.2007).
- [Pal02] S. B. Palmer. RDF in HTML: Approaches, June 2002. <http://infomesh.net/2002/rdfinhtml/index.html> (Last visited 09.02.2007).
- [Pri] PRISM - Publishing Requirements for Industry Standard Metadata. <http://prismstandard.org/namespaces/1.2/basic/> (Last visited 09.02.2007).
- [RDF] Frequently Asked Questions about RDF: How do I put some RDF into my HTML pages? <http://www.w3.org/RDF/FAQ#How> (Last visited 09.02.2007).
- [RDF00] Issue faq-html-compliance: The suggested way of including RDF meta data in HTML is not compliant with HTML 4.01 or XHTML, 2000. <http://www.w3.org/2000/03/rdf-tracking/> (Last visited 09.02.2007).
- [RDF04a] Resource Description Framework (RDF), 2004. <http://www.w3.org/RDF/> (Last visited 09.02.2007).
- [RDF04b] RDF Vocabulary Description Language 1.0: RDF Schema, February 2004. <http://www.w3.org/TR/rdf-schema/> (Last visited 09.02.2007).
- [RDF06] RDFa Primer 1.0 - W3C Working Draft, May 2006. <http://www.w3.org/TR/2006/WD-xhtml-rdfa-primer-20060516/> (Last visited 09.02.2007).
- [Rei06] Gerald Reif. Semantic Annotation. In Tassilo Pellegrini and Andreas Blumauer, editors, *Semantic Web - Wege zur vernetzten Wissensgesellschaft*. Springer Verlag, April 2006.
- [RMG06] Gerald Reif, Martin Morger, and Harald Gall. Semantic Clipboard - Semantically Enriched Data Exchange Between Desktop Applications. In *Semantic Desktop and Social Semantic Collaboration Workshop at the 5th International Semantic Web Conference ISWC06*, Athens, Geogria, USA, November 2006.
- [RoR] Ruby on rails. <http://www.rubyonrails.org/> (Last visited 09.02.2007).
- [SBH⁺05] York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC Ontology - Semantic Web for Research Communities. In Carlos Bento, Amilcar Cardoso, and Gael Dias, editors, *Proceedings of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005)*, volume 3803 of LNCS, pages 218 – 231, Covilha, Portugal, December 2005. Springer Verlag.
- [SBLH06] N. Shadbolt, T. Berners-Lee, and W. Hall. The Semantic Web Revisited. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 21(3):96–101, May 2006.
- [Scr] script.aculo.us - web 2.0 javascript. <http://script.aculo.us/> (Last visited 09.02.2007).
- [Sen] sendmail. <http://www.sendmail.org/> (Last visited 09.02.2007).
- [Shi06] Sang Shin. AJAX Programming Online Course, July 2006. <http://www.javapassion.com/ajaxcodecamp/> (Last visited 09.02.2007).
- [SWR] OntoWare: SWRC Ontology Project Info. <http://ontoware.org/projects/swrc/> (Last visited 09.02.2007).

- [Syma] symfony - open-source php web framework. <http://www.symfony-project.com/> (Last visited 09.02.2007).
- [Symb] symfony - how to manage user credentials. <http://www.symfony-project.com/book/trunk/security> (Last visited 09.02.2007).
- [Upd05] Andrew Updegrove. The Semantic Web - An interview with Tim Berners-Lee. *Consortium Standards Bulletin*, June 2005.
- [XDK] The XdkBibTeX Library. <http://artis.imag.fr/Membres/Xavier.Decoret/resources/xdkbibtex/> (Last visited 09.02.2007).
- [XHT01] XHTMLTM 1.1 - Module-based XHTML, 2001. <http://www.w3.org/TR/xhtml11/> (Last visited 09.02.2007).