Social bookmarking of semantic web services

Bachelor Thesis

Kammerlander Patrick (0215857)
Patrick.Kammerlander@student.uibk.ac.at

Dalle Pezze Marco (0215858)
Marco.Dalle-Pezze@student.uibk.ac.at

23th September, 2008

Supervisor: MSc Ioan Toma
Abstract

The main goal of this thesis is the realization of a website which enables the user to bookmark semantic webservices and rate them. The covered topics include the main technical meaning of webservices with explanation of the terms WSDL, UDDI and SOAP and a section treating the hype Web2.0. Further a description of the created website with screenshots and exact explanations are given. The website is functional and gives the user some valuable feedback about the rated webservices. Some other functionality is realized like a Login-system, a tag cloud, a search-functionality and others. At the end of this documentation a description about the implementation languages used and the resulting issues to realize the site should help the reader to get a general idea about the amount of work put into this thesis without knowing these languages before."
Contents

1 Introduction 1
  1.1 Web services ............................................. 1
  1.2 Social web .............................................. 1

2 Web 2.0 2
  2.1 The beginning: WEB1.0 ................................. 3
  2.2 Definition of Blog ...................................... 4
  2.3 Definition of Tag ...................................... 4
  2.4 Definition of Social Bookmarks ....................... 5
  2.5 Technical meaning of Web2.0 .......................... 5

3 Web services 7
  3.1 Definition of Web services ............................. 7
  3.2 Technical details of Web services ..................... 8
  3.3 SOAP .................................................. 9
  3.4 UDDI .................................................. 11
  3.5 WSDL .................................................. 12
  3.6 Semantic Web Services ................................. 14
    3.6.1 OWL-S ............................................ 14
    3.6.2 WSMO ............................................. 15
    3.6.3 Concluding on semantic Web services ............ 16
  3.7 Useful Web service (Google Friend Connect) .......... 16

4 Contents and Functionality 18
  4.1 Header ................................................. 20
  4.2 Login .................................................. 20
  4.3 Search ................................................. 21
  4.4 Little Tagcloud ........................................ 22
  4.5 Main frame ............................................ 23
    4.5.1 Home .............................................. 23
    4.5.2 Web services ...................................... 24
    4.5.3 Feedback ......................................... 25
    4.5.4 Tags ................................................ 26
    4.5.5 Selected Web service .............................. 27
    4.5.6 Rate ............................................... 28
    4.5.7 Posted and bookmarked Web services ............ 29

5 Implementation 31
  5.1 Programming languages .................................. 31
    5.1.1 PHP ............................................... 31
5.1.2 JavaScript ................................................. 31
5.1.3 MySQL .................................................. 31
5.1.4 HTML ...................................................... 32
5.1.5 CSS ...................................................... 32
5.2 Development environment .................................. 32
5.3 Structure ..................................................... 32
5.4 DB schema ................................................... 33
5.5 Structure of the web page ................................... 34
5.6 Login/Register ............................................... 36
  5.6.1 Sessions ................................................ 36
  5.6.2 IP check ............................................... 36
  5.6.3 Quotes handling ....................................... 36
  5.6.4 Login .................................................. 37
  5.6.5 Logged on ............................................. 38
  5.6.6 Register ............................................... 38
5.7 Search ....................................................... 40
5.8 Tagcloud ................................................... 41
5.9 Algorithm of popular WS's ................................ 42
5.10 Listing of WS's in an abstract form ....................... 42
5.11 Feedback .................................................. 44
  5.11.1 chart ............................................... 44
  5.11.2 script.aculo.us ..................................... 44

6 Conclusion .................................................. 46
6.1 Future Work ............................................... 46
1 Introduction

1.1 Web services

A number of different factors are driving the growth of the Web services (Ws in short) market. A major one is that it represents a standards-based approach to integration and interoperability. Web services are a way of designing and building software so that it is easier to link incompatible systems. Because of the everyday growing amount of Ws’s it is difficult to find the needed one. It is not easy to find a Ws using google, yahoo or other well known search sites. There are a few portals which try to improve the transparency and accessibility of the Ws market. Two online sites which try to match these characteristics are www.xmethods.net and seekda.com. The first one limits its Ws search function to a complete listing of all services and some short information about every service. Seekda’s mission is to “facilitate on-demand use of services over the Web”. It offers a search engine providing access to publicly available Ws. It gives good overall information about nearly every service also because of the possibility to directly use the Ws with the Web Service Tester. But a missing point is the social interoperability of this site. We tried to fulfill this need and realized a website which is part of the social Web.

1.2 Social web

Social web is characterized by knowledge collection, knowledge discovery, knowledge building and knowledge sharing. By providing a system where every user can rate and post Ws’s and see the feedback, these conditions are fulfilled and the web service market becomes clearer. Because of the need of such a system and the challenge of learning new languages such as PHP and CSS and then create the website with database, we decided to do this bachelor thesis.
2 Web 2.0

A short aperture of a typical everyday example using Web2.0 (for example mine):

Switching on the computer and the start page shows the site www.google.com/ig

![Google iGoogle](image)

Figure 2.1: The www.google.com/ig site from Marco Dalle Pezze

This site is the embodiment of Web2.0 because it contains a lot of applications which are attributed to the hype WEB2.0:

- RSS-feeds which display the current news directly from the interested papers (sport, policy, local news, ...)
- A weather screen which displays the weather of a certain city or country
- The newest videos on YouTube
- The last emails out of my email account
- A daily new joke to begin the day with a smile
- A daily new dish to cook for lunch or dinner
- The Wikipedia search function
- etc
The whole site is expanded with every kind of interesting information which can individually be brought into the site by simply choosing it from a large list or even create it by yourself.

Starting up skype or msn messenger (two famous chat programs) shows who of your friends is online and then chat about what you have just read. A look at google mail and the connected calendar will show up what the present duties are. The Google calendar can be linked to other peoples’ google calendars and so it gives a full overview of a working group’s to do list. So take a look at it and know not only what you have to do today but also what your working group does.

For a personal research I need information about “PHP”. So the new search sites are not only google or yahoo, but also del.icio.us or Wikipedia, two other big and powerful new Web2.0 services.

During the research a site like onlineradio with a huge database of every kind of music creates an automatic playlist with the chosen genres. During the first break a short look at the private blog of a few friends gives an update on what happened the day or the weekend before. Obviously a reply (comment) on the nice events posted by a friend must be written.

The description before shows that the common way to use internet today is very interactive and this is exactly what Web2.0 wants to represent.

Web 2.0 is linking people.
..people thinking, sharing and collaborating

The new internet will give the user the possibility to do everything online and of course do it faster and more comfortable than previously. The internet user will have all his software online without the need to install it on the PC. And the best thing is that the most WS’s are for free. But how was it before?

2.1 The beginning: WEB1.0

In order to speak about Web2.0, we need to focus on Web1.0, the predecessor of Web2.0.

Not long ago, the focus was on netscape, alta vista and slow internet connections. Writing an email was not very common and not many people used internet. The digital information had an enormous boom with the growing number of personal computers which were connected to the internet. The broadband enabled every user to read the information in the world wide web every day on his PC, without limitations in time or velocity.

At the beginning the information was unilinear, only with plain text and some pictures at the maximum. The author wrote down what he had to say and any internet user could read it. The published information on the internet has grown and with it also the need for interaction between user and author.

The user will not only read information but also answer and comment on it; even write his own personal information. The user simply wants to interact with the web.

In the next table Tim O'Reilly tried to make a comparison of older sites without interaction of the user and new Web 2.0 sites with interaction of the user.
The last three comparisons are also very important for this thesis because the created website is basically a blog and supports tagging.

2.2 Definition of Blog

A “web log” is a form of web representation or publishing characterized by an indefinite number of entries or posts, which are usually presented as a list in reverse chronological order. Every post can be viewed individually by a permalink URL assigned to each post.

Why this website is basically a blog:
This website displays all WS’s added, in a reverse chronological order. The user is able to add some comments to each WS and every WS (every post) can be viewed individually by a permalink URL. These are the main properties of a blog.

On the other side we have a personal website used as an information site for everyone. An everyday upgrade with information would be in a chronological order and probably subject to some author invented taxonomies. Even the possibility to add some comments or ratings would not be given. But this kind of Website is not up to date and the amount of blogs grows daily. By the second quarter of 2006, 50 million blogs were created - new ones were added at a rate of two per second.

2.3 Definition of Tag

A tag is typically one or two words used to describe some arbitrary item. Because a tag is usually very concise, it is used as a way to categorize said items. One item can have multiple tags, and if used properly, one tag will belong to multiple items.

Another goal of this thesis is to add this interesting feature to the created Homepage. As we can deduce from the definition of a tag, tagging is the possibility to add one or more tags to an item and in our case to a WS. Common tags for new WS’s would be “Web2.0”, “community” or “social software”. If a WS
is tagged with one of these words, it is easier for a user to find it and also find some other possibly interesting WS related to the same tag.

In this sense a tag is like an index in a library:
If a person needs a book which contains for example some information about “PHP” he searches for it with the search computer of the library and then has to go to the rack and look for books with the given index.
It works in the same way for a user who is interested in some WS’s correlated to e.g. “PHP”. He searches for this tag (PHP) and gets a list of all WS’s entered in the database and connected with this tag.

In summary the library is an example of a taxonomy. In this case a few people decide on the indexes and how to organize the cataloguing of all subjects. Because of that the cataloguing is not social.
The created website on the other hand is an example for a folksonomie. Here the user decides the cataloguing of the inserted items by creating his own indexes (tags). The best example here is the tag cloud of the created website shown in Fig. 4.14 The tags with the most WS’s connected are the biggest ones in the cloud.

2.4 Definition of Social Bookmarks

Social Bookmarks are links used to save and share sites on the internet.
Bookmarks are a set of saved links to webpages that want to be remembered.
They become Social Bookmarks if they are public.

2.5 Technical meaning of Web2.0

The concept of Web 2.0 began with a conference brainstorming session between O’Reilly and MediaLive International after the bursting of the dot-com bubble in the fall of 2001. Web2.0 is a hype which is often used but not really well defined because it is not a new technology. It is an umbrella term for all the new applications made with already existing technology to help the user to interact with the internet. This means not only that the user is now able to write, insert, post information on lot of sites without much difficulty but also that he can choose which information he wants to get and filter only the interesting one. Basically we can underline the following features of Web2.0 compared to Web1.0:

1. The Web is now a platform for software. Software will no longer be installed on the operation system but directly on the web, where everybody can use it. The best example here is Wikipedia or Google.

2. The focus is also on the many “small” users and not only on the big companies. This phenomenon is called “long tail”.

3. The knowledge of everyone is used in search engines, file-sharing tools or other networks. The community increases the quality of the application by using it.
4. Re-integration of tools in other services by open programming interfaces. With open source a dynamic innovation is possible.

5. Hyperlinks become more interesting because they are extended with RSS-feeds and in coherence with blogs build an important feature for Web2.0.

6. The most services are cost-free to attract a huge number of users.

7. New services in Web2.0 are continuously changing and developing. This new services often has the ending beta. A Beta version of a software means that it is tested and works but is not yet finished. But a Web2.0 application will never be totally finished on the web. There is always some functionality to be added by the user.

8. The control over the data has changed in a way that the focus is on bringing together the data by the users (syndication) instead of the coordination by the author for who gets what.

9. “User generated Content”: more and more the user builds the content of the page and not the owner of the webpage (for example file sharing tools or blogs). User can create their content. User becomes a Cresumer (Creator and Consumer in one person).

10. And one of the most important things is the growing interactivity between the user and the owner of theWs.

While none of these features or aspects of development are new, the Web 2.0 term is used to describe the current generation of web sites that make good use of HTML and CSS while perhaps improving their interface with PHP or Ajax and social-networking tools.
3 Web services

Basically Ws’s are small units of code which can handle a limited set of tasks. Ws’s generally get called by other programs or applications. Doing this an application or a program which has a task to solve can solve it just in time by using some provided Ws’s of other sites. Through them the program can get the data from other databases which are not directly accessible. Otherwise it would have to make a request and wait for needed data to continue the solving.

3.1 Definition of Web services

The first definition of WS is from Tim Ewald (Microsoft Corporation):

The basic idea behind Web services is to adapt the loosely coupled Web programming model for use in applications that are not browser-based. The goal is to provide a platform for building distributed applications using software running on different operating systems and devices, written using different programming languages and tools from multiple vendors, all potentially developed and deployed independently.

Even IBM has defined Web Services:

Web services are self-describing, self-contained, modular applications that can be mixed and matched with other Web services to create innovative products, processes, and value chains. Web services are Internet applications that fulfill a specific task or a set of tasks that work with many other web services in a manner to carry out their part of a complex work flow or a business transaction. In essence, they enable just-in-time application integration and these web applications can be dynamically changed by the creation of new web services.

Ws’s are becoming more and more important in the World Wide Web. Most popular examples for Ws’s are on travel agencies. These agencies have a program which uses the Ws’s provided by the single airlines to find and eventually book a flight. In this way the agencies don’t have always to contact the companies to ask for the flights. At the same time the problem with overbooking a flight in the case that more agencies ask for the same flight is resolved. Another example of an often used Ws is the validation of credit cards. The site which enables the client to pay with credit card uses the application provided by the company behind this credit card to validate it. In the practical part of this thesis there is the possibility to add a useful Ws which makes the site more social.
interactive. This Ws is called Google Friend Connect and will be explained in Section..

The basic properties of Ws’s are:

- to connect people, systems and devices
- to communicate using open protocols (HTTP, SOAP)
- are self-contained and self-describing (WSDL)
- can be discovered using UDDI
- can be used by other applications
- are independent of operating systems
- are independent of programming languages

3.2 Technical details of Web services

The picture in Fig. 3.1 shows the architecture of a Ws:

![Figure 3.1: Architecture of a Web service](image)

The three actors in this architecture are the Service Broker (SB), the Service Requester (SR) and the Service Provider (SP). The SB is a set of servers which contain the description of Ws’s and also the companies behind them. This description is made with UDDI (Section 3.4). Ones the SR finds the Ws of interest he requests the documentation of the Ws. This documentation is made with WSDL (Section 3.5) and it offers an overall explanation what the Ws does and how to call it (parameters to give and get). Finally the access from the SR to the SP works with SOAP (Section 3.3).

How this works in detail:
The Web services platform is XML and HTTP:

- The HyperText Transfer Protocol is the most used internet protocol to transfer information and other data through the internet and from the WWW to any browser of a client
- The eXtensible Markup Language is a language which can be used to communicate between different platforms and programming languages and still express complex messages and functions

The Web services platform elements are:

- SOAP (Simple Object Access Protocol or Service Oriented Architecture Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

The following description of the three platform elements should help to understand the functionality of WS’s without entering too much into technical details.

3.3 SOAP

SOAP is an XML-based communication protocol for distributed applications to exchange structured and typical data. It uses HTTP, SMTP or FTP to transport information. The area of operability goes from simply sending a message to RPC (Remote Procedure Call). Simply said: SOAP is a protocol for accessing a WS. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and different programming languages without being blocked by firewalls or proxy servers. The following code shows the skeleton of a SOAP element:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

Figure 3.2: Skeleton of a SOAP element

A SOAP message is an ordinary XML document, as clarified in the first row of the code in Fig. 3.2, with an envelope namespace which identifies the document as a SOAP message. The envelope element is the root element of a SOAP
message and defines how the Ws is structured. This is usually:

\[ \text{xmlns:env}="http://www.w3.org/2003/05/soap-envelope.} \]

The Header and the Fault are optional elements. The Header declares routing details and the Fault declares information about errors occurred while processing the message and how to manage them.

The Body is necessary and contains the call and response information. A simple example of a price request call would be:

```
<soap:Body>
  <GetPriceRequest xmlns="http://www.vegetables.com/prices">
    <m:Item>Bananas</m:Item>
  </GetPriceRequest>
</soap:Body>
```

Figure 3.3: example of a SOAP request

The method shown in Fig. 3.3 is called GetPriceRequest and has one parameter which is the item. In this case the request is for the price of bananas.

The response of this message could be:

```
<soap:Body>
  <GetPriceResponse xmlns="http://www.vegetables.com/prices">
    <m:Price>1.90</m:Price>
  </GetPriceResponse>
</soap:Body>
```

Figure 3.4: example of a SOAP response

The price of bananas is 1.90.

**SOAP and HTTP**

HTTP communicates over TCP/IP. An HTTP client connects to an HTTP server using TCP. After establishing a connection, the client can send an HTTP request message to the server:

```
POST /item HTTP/1.1
Host: 89.105.64.39
Content-Type: text/plain
Content-Length: 200
```

The server processes the request and sends back a HTTP response to the client. If the request was successful (200 is the standard success status code in HTTP) the response could be:

```
200 OK
Content-Type: text/plain
Content-Length: 250
```

If the request wasn’t successful the response would be:

```
400 Bad Request
Content-Length: 0
```

A SOAP method is an HTTP request/response that complies with the SOAP encoding rules.

**HTTP + XML = SOAP**
3.4 UDDI

The Universal Description, Discovery and Integration (UDDI) protocol defines a method for publishing and discovering Ws’s. The functional purpose of a UDDI registry is the representation of data and metadata about Ws’s. A registry can be used by the public network or within an organization’s internal infrastructure and offers a standards based mechanism to classify, catalogue and manage Ws’s. By doing this, the Ws’s can be discovered and consumed by other applications. According to this, the standard provides a means to locate a software service, to invoke that service, and to manage metadata about that service.

The UDDI registry consists of three components:

**White Pages:** give information about the company which supplied a Ws. This could be contact information like address or phone number and some known identifiers. There is one white page for every company.

**Yellow Pages:** Provide an industrial categorization based on standard taxonomies. It is comparable with the real yellow pages where the companies are indexed by their business.

**Green Pages:** Offer technical details about a Ws. How to access it and its service bindings. Also the parameter and the references to specifications of interfaces are described. Green Pages are needed to know how a Ws should be used to get the right answer. If a Ws has more possible ways of use, it must have more green pages related to it: a green page for every usage.

The UDDI Data Model for these three components includes the following information:

- A description of a service’s business function (called the businessService)
- Information about the organization that published the service (businessEntity)
- The service’s technical details (bindingTemplate), including a reference to the service’s programmatic interface or API, and various other attributes or metadata such as taxonomy, transports, digital signatures, etc. (tModels)
- Relationships among entities in the registry (publisherAssertion)
- Standing requests to track changes to a list of entities (subscription)
3.5 WSDL

WSDL is an XML document used to describe WSs and to locate them. It specifies also which methods and operations the WSs exposes. The following code shows the skeleton of a WSDL document:

```
<definitions>
  <types>
    <definition of types>........
  </types>
  <message>
    <definition of a message>....
  </message>
  <portType>
    <definition of a port>........
  </portType>
  <binding>
    <definition of a binding>....
  </binding>
</definitions>
```

Figure 3.5: Skeleton of a WSDL document

A WSDL document can contain also other elements, like extension elements or service elements to group the definitions of more WSs's to one single WSDL document. But these are the most used and needed elements of WSDL documents. The most important element of a WSDL document is the \(<\text{portType}\)>.

It could be compared to a function library or a class in java because it describes a WSs, the operations that can be performed with it and the messages that are involved. For example a part of the WSDL document of the WSs used in the SOAP example (Fig. 3.3 and 3.4) is

```
<message name="GetPriceRequest">
  <part name="Item" type="string"/>
</message>

<message name="GetPriceResponse">
  <part name="Price" type="double"/>
</message>

<portType name="priceRequest">
  <operation name="getPrice">
    <input message="GetPriceRequest"/>
    <output message="GetPriceResponse"/>
  </operation>
</portType>
```

Figure 3.6: Example of a WSDL document

In this example the port “priceRequest” defines a request-response operation called “getPrice”. The “getPrice” operation requires an input message called “getPriceRequest” with a parameter called “Item”, and will return an output message called “getPriceResponse” with a parameter called “Price”.

The \(<\text{message}\)> element defines the data elements of an operation. Further, each message can consist of more parts. A part is comparable to a parameter of a function in a traditional programming language. In our example the part of the price request function is the name of the item of which we want to know the price.
3.5 WSDL

The \( \langle \text{types} \rangle \) defines the data types used by the Ws. The \( \langle \text{binding} \rangle \) element defines the message format and protocol details for each port.

**There are three most common styles of using a Ws:**

- RPC
- SOA
- REST

**Remote Procedure Call:** Remote Procedure Call was the first style of use for Ws’s and because of this very common, widely deployed and supported. It should be very familiar to many developers because it is used also in the most programming languages. As the name implies it is a call by the user situated on a certain position, for a procedure situated in a remote position. This style is sometimes criticized for not being loosely coupled because many times it is implemented by mapping the Ws directly to language-specific method calls.

**Service Oriented Architecture:** A software architecture model for building loosely coupled distributed systems. This systems are groups of Ws’s which work together. This group work involves either simple data passing or it could involve two or more services coordinating some activity.

**Representational State Transfer:** It is characterized by a pull-based client-server design, stateless operation, resources named via clean URLs and the core HTTP methods of GET, POST, PUT, and DELETE are used to model the essential REST operations (interfaces).

Finally there is to say that the Web Services platform is a simple, interoperable, messaging framework. It still misses many important features like security and routing. But, these pieces will come once SOAP becomes more advanced.
3.6 Semantic Web Services

With the realization of semantic Web services the automatic identification and composition of Ws’s is possible. This opens new options for the E-Commerce, E-Government and Enterprise Application Integration (EAI): Through the intelligent combination of already existing distributed Ws’s, new better and more individualized can be developed.

This section provides an overview of the service description frameworks used to semantically describe Web Services in order to enable the automation of Web Service discovery, composition, interoperation and invocation. Two major ontologies used in semantic web services are

- OWL-S
- WSMO

These two enhance web services with semantics. The following two sections describe the two ontologies.

3.6.1 OWL-S

OWL-S has been the first approach for an overall framework for describing Semantic Web Services. It started in 2001 and have as a predecessor DAML-S. OWL-S defines an ontology system for describing Web Services, using OWL as the description language. OWL-S defines a set of classes and attributes to describe Ws’s:

- **Service** is the organizational point of reference for the declaration of a Ws
- **Service Profile** is used to discover Ws’s. Holds the name, its provider and a description of the Ws
- **Service Model** is used to describe the functionality of the service and its composition
- **Service Grounding** is used to hold the technical details of a service. It describes how to access the service and a mapping from an abstract to a concrete specification for service usage

OWL-S classifies Ws processes into three groups:

- **Atomic Process** is a process which can be direct called and solves one task in one step
- **Simple Process** is an abstract view of a process, e.g. how to make a set of Atomic Processes. A simple Process can not be called
- **Composite Process** is a process which consists of several other processes. This were assembled by constructs like sequence, split, join, any order or choice
3.6 Semantic Web Services

3.6.2 WSMO

The Web Service Modeling Ontology (WSMO) is also a framework for Semantic Web Services supporting automated Web Service discovery, composition and execution. It is based on the Web Service Modeling Framework (WSMF) which applies two major principles:

1. maximal de-coupling of the different components

2. mediation service, based on Semantic Web Technology, to connect different components

As shown in its logo (FIG. 3.8) WSMO consists of four notions:

**Ontologies:** offer the terminology used by the Web Services

**Goals:** represent user desires

**Web Services:** descriptions, which define different aspects of the Web Service

**Mediators:** realise the principles of strong de-coupling and mediation

A characteristic of the WSMO architecture is the linking of Goals, Web Services and Ontologies by four types of Mediators:
3.6.3 Concluding on semantic Web services

Both of these description frameworks see the essential presence of ontologies to enable an automatic interaction with and between Ws’s. OWL-S focuses on the specialisation and generalisation of the modeling. It wants independent, atomic processes which can be grouped if necessary. WSMO on the other site focuses on equitable group solutions. They can communicate together through scalable mediators which define a different strong binding.

3.7 Useful Web service (Google Friend Connect)

Google Friend Connect (GFC) is a gadget pool to make any website more social. Social gadgets would be a friend list, a photo upload with the possibility to rate and share them, comment and posting-functionality,.. and all this without any programming knowledge.

This works quiet easy: First the user has to choose the functionality out of a given feature list. Then he changes the look and feel of the chosen gadgets to match them with his site. A click on a generate code button and the code is ready for copy-paste on his site.

The most important gadget offered by Google Friend Connect is certainly the Members gadget. With this gadget it is possible to right start a community and so get the main social features:

- Login with an existing Google, Yahoo, AIM or OpenID account
- Invite and change friends out of social networks like Facebook, Google Talk,…
- Research of member-profiles on different portals
- Become new Friends with users on the own site
3.7 Useful Web service (Google Friend Connect)

This means that a user who logs in on a site with GFC are immediately connected with the other GFC members on this site and have the possibility to invite some more friends out of other social networks listed before and can even be seen on other networks. A comment on this site can be seen on other networks which use GFC. The plan is to get an overall connection in every part of interest an internet user could have.
4 Contents and Functionality

The created website gives every user the possibility to rate, bookmark, post and comment Ws’s. How to do these and all other functionality is described in this chapter. The Fig. 4.1 shows the graphical sitemap of the website. The sites are represented as squares and the contents as ellipses. It is clear shown where the single connections from one site to another are and also the amount of information given by the site.

![Figure 4.1: Sitemap of the website](image-url)
After the sitemap the Fig. 4.2 shows the website as it is displayed on a browser. In this case the Firefox 3.0.1 was used. Basically the site is divided into three parts:

- Header
- Side frames
- Main frame

The header and the side frames remain always the same with exception of the login frame which changes when a user signs on or registers at the site. The main frame has various looks as described in Section 4.5.

Figure 4.2: The website shown with Firefox 3.0.1
4 Contents and Functionality

4.1 Header

We designed the header on our own. We wanted to create a nice logo for “social bookmarking of semantic web services” and gained our thoughts as shown in Fig. 4.3.

Figure 4.3: Header

4.2 Login

The first side frame on the upper left side is named “LOGIN”. Here a user has always the possibility to make the login or if he is not registered to do this by clicking on the register button shown in Fig 4.4a. If the user logs in regularly, the frame will change, welcome the new user and give him the possibility to view all his inserted or bookmarked Ws’s in the main frame and to logout as shown on Fig 4.4b. If the login was not successful the user will be informed as shown in Fig 4.4c.

Figure 4.4: The three possible login sequences
If the user is not registered by clicking the register button in the main frame a form appears which asks for the username, email, name and password of the new user (Fig. 4.5). Of course the name and the password will be checked as described in Section 5.6.6. For a valid registration the user gets a corresponding feedback otherwise he will be informed about the wrong or missing inserts.

![Form to register a new user](image1)

4.3 Search

The content of the second side frame on the left side is the search functionality. This function is needed to find a specific Ws as well as to find some services which relate to a specific term. The exact approach is described in Section 5.7. If a term is found the main frame shows the results by listing all the matches and highlighting the founded terms (Fig. 4.7). The user can simply click on the title of the WS of interest and gets to the specific page. Additionally the search functionality enables the search of users. The result of a found user is the link to the site which looks like the site described in Section 4.5.7. The difference
4 Contents and Functionality

between this site and the one referred to is that here are displayed the WS’s of the specified user instead of the logged in user.

Figure 4.7: Result of search function with the term “fast”

4.4 Little Tagcloud

The last side frame on the left side of our website shows a small tag cloud. This cloud shows the most important tags of this site. The tags differ from each other by the size. A more important tag is bigger than a less important one. The result by clicking on a tag is a listing in the main frame of the ten most voted Ws’s which are related to this tag. This Ws’s are arranged with the most voted at the top and going down to the less voted. If the user clicks the “more...” link in the lower left corner of this frame the main frame jumps to the tag-tab described in Section 4.5.4.

Figure 4.8: Tagcloud of the side frame
4.5 Main frame

This is the most important frame of the website because in this central frame all the functionality is performed. The main frame is clearly laid out because it is arranged with tabs (Fig. 4.9). This offers a user-friendly navigation through the website. Every tab has a different content and is clearly signed if clicked by the user. If none of the tabs is relevant (e.g. search or the register function), no tab is selected but the sub-line indicates where the user actually is.

![Figure 4.9: Tabs and subline of the main frame](image)

The main frame consist of four tabs:

- Home
- Web services
- Feedback
- Tags

Four more sites can be displayed in the main frame which are not directly accessible by clicking the tabs. “Selected Web service” and “Rate” are two sites of them which can be selected through the content of other sites as described in the corresponding sections. The other two, “My posted Web services” and “My bookmarks” will be displayed if the equal named links in the side frame “Login” were clicked.

4.5.1 Home

This tab is active on entering the website and as highlighted in the sub-line the user can see a list of the five most voted Web services. This list is arranged like a blog where every entry is a Ws. This tab is shown in the first screenshot of this chapter in Fig. 4.2. Each of these listed Ws’s on this site consists of the following information:

**Listed Web service**

![Figure 4.10: a listed Web service](image)

Every listed Ws, compared with a blog entry, gives a short overview of what the service is about. On the upper left corner the title of the Ws is used to catch the attention of the user for this subject and as a link to further information
4 Contents and Functionality

about this selected Ws (Section 4.5.5).
The upper right corner offers a link to the Homepage of the described Ws.
Under the title a short extract of the description of this Ws gives an overview
about the topic of this service. This text ends with three points which can
redirect the user to a site with further information about this Ws like the title.
In the lower part of the listed Ws the user gets to know the first five tags which
were added to it by its author. Every tag can be clicked on like in the tag cloud
to know other Ws’s related to this matter. The number next to the tags is the
amount of ratings which were given to the Ws. The last number in the lower
right corner is the average rating of the specific Ws.

4.5.2 Web services

Figure 4.11: Main frame with active tab “Web services”

The second tab of the mainframe is named “Web services”. As the name
implies here the user gets an overall listing of all Ws’s added to this site in a
reverse chronological order so that the last entered Ws can be seen first. The
listed Ws’s are represented as described in Section 4.5.1. The user has also
the possibility to display the next or the previous five entries by simply clicking on
the arrows beside the page number at the end of the site. In the sub-line of
this tab the user has the possibility to choose either to have a look at the list
of Ws’s or to add a new service. This could easily be made by clicking on the
corresponding link in the sub-line (Fig. 4.11). This link opens the form for the
insert of a new Ws if the user is logged in or it writes an error message when
he’s not. If the user is logged in correctly the main frame offers the form shown
in Fig. 4.12 to add a new Ws:
4.5 Main frame

Figure 4.12: Form to add a web service

The first field is for the name or title of the WS. The second field should be a working URI where the user can access to the WS of interest. In the third field the author can give a description of his new WS which should be an easy to understand text that inspires the users to have a look at this new site and rate it.

The last field is very important because with it the author can add the tags that are relevant for the new WS. Every tag must be separated by a whitespace. If the author writes an already existing tag the corresponding function identifies it and builds the right connections otherwise it adds the new tag to the database and consequently makes it visible also in the tag cloud.

At the end of the form a submit button allows the insert of the new WS in the database and gives a positive notice in the main frame instead of the form. If the author wants to clear all the fields an accordingly named button allows this action.

4.5.3 Feedback

Figure 4.13: Main frame with active tab “Feedback”

The third tab contains the feedback of WS’s. The user gets information about which are the most popular, the best rated, the worst rated or the most rated WS’s by simply clicking on the specific link in the sub-line of the tab. Every list has a different algorithm to calculate the ranking which will be described
in Section 5.9. The single Ws’s are arranged with the respectively best ranked entry on top of the list.

4.5.4 Tags

![Tag cloud in alphabetical order](image)

Figure 4.14: Tag cloud in alphabetically order

The content of this last tab is a huge tag cloud complete with all tags which are saved in the database and therefore connected to some Ws’s. As alike the little tag cloud in the side-frame this tag cloud shows tags with different sizes. The size of a tag depends on how often it is used to describe a Ws and consequently linked to it. The exact algorithm which results the size of each tag is explained in Section 5.8. In the sub line of this tab the user can choose either if he wants to arrange the tags in the tag cloud alphabetically by clicking on the corresponding link or arrange them by popularity with the other link. The tag cloud shown in Fig. 4.14 is arranged in alphabetically order.
4.5.5 Selected Web service

This site in the main frame will be displayed if the user clicks on a listed Ws. It shows all the information which is available for the chosen service and therefore the tab Web services is active. The composition of this site is arranged as follows:

In the upper part the user can again see the title and the date of adding on the left and the URI on the right site of the main frame. Among the title the user can read the whole description of this Ws. Among the description on the right side the user gets the information about who the author is. The authors username is a link to a site which looks like the site described in Section 4.5.7. The difference between this site and the one refered to is that here are displayed the WS’s of the specified user instead of the logged in user. Following the authors link there is shown the complete list of the tags related to this Ws. If a user is logged in among the tags a link gives the user the possibility to bookmark this Ws. If the user is not logged in the link is not visible.

Now separated with two accordingly named headlines the user can interact with the system by leaving a comment for this Ws or rate it. The comment can be filled out with a simple form below the last entered comment in the list. This
form requires at least the name and email of the user who wants to comment. The comments will then be displayed in chronological order without showing the email address of the author because of privacy reasons. The comments will be displayed as shown in Fig. 4.15. Among the headline “RATINGS” the user can clearly see how the overall rating of this WS is. On the left side of the rating square there is a vertical listing of the possible marks from one till ten. Rightward of every mark the user can see the number of ratings for this mark, first shown with white lines and then as exact number. The number below the marks tells the user the overall mean of the marks for this service. As shown in Fig. 4.15 the mean value for this WS is 5.44 for a total of 100 ratings. The total number of ratings is shown in the lower right corner of the rating square. Below the rating square a link offers the possibility to any user to give a rating for a WS. By clicking this link the main frame offers the rate site.

4.5.6 Rate

Figure 4.16: Rating of a selected Web service

This site is displayed with the active tab “FEEDBACK”. It gives the possibility to any user even if not registered to rate a chosen WS. If displayed, the site looks like Fig. 4.16. In the sub-line of the tab Feedback the user can see which WS he is about to rate. If the user wants again to return to the description of this WS he only need to click on the title of the WS. On top of the main frame the user can see the first rating square extended which is the overall rating. To give a valid rating the user must at least make an overall rating. To do this the user must give a mark by selecting it from the drop-down Combo box below the rate square. An open Combo box is also shown in Fig. 4.16. If the user clicks the vote button without giving a mark an error advice appears. If the user wants to give a more precise rate for a WS he can extend also the other
rating squares by clicking on the arrows rightward the corresponding name of the rating. The click on the arrow slides down the corresponding rating square. In Fig. 4.17 the overall and the costs rating squares are open.

If a user has already rated a Ws he does not can vote the same Ws again during the same session. This avoids wrong results at the rating statistics because of multiple rating of the same user.

![Figure 4.17: Overall rating and costs rating](image)

### 4.5.7 Posted and bookmarked Web services

These two sites have quite the same look as shown in Fig. 4.18. Both can only be viewed if the user is correctly signed in. One site (Fig. 4.18a) lists all the Ws’s which were inserted by this user with the possibility to remove them. This site gets called through the “My posted Web services” link in the “Login” side frame.

The other site (Fig. 4.18b) lists all the Ws’s which were bookmarked by this user with the possibility to remove the bookmark of the single Ws’s. This site gets called with the “My bookmarks” link in the “Login” side frame.
Figure 4.18: (a) My posted Web services (b) My bookmarked Web services
5 Implementation

After the description of the programming languages used and the development environment, this Chapter explains what the major implementation issues are about developing the site.

5.1 Programming languages

The programming languages used to realize this site are PHP, JavaScript, MySQL, HTML and CSS. In the following subchapters every language is shortly explained. A focus on single features which are part of the web page will resolve possibly doubts of the reader.

5.1.1 PHP

PHP ("Hypertext Preprocessor", originally for "Personal Home Page Tools") is a script language principally for web development. It is a server-side programming language which generally runs on a web server, taking in HTML embedded PHP code to create web pages.

PHP has a lot of advantages: it is free and open source software, it runs on different operating systems such as UNIX and Windows and most web servers provide it. PHP can be interfaced very well with MySQL which is the most popular online database. Its syntax is most similar to C and C++ language syntax, so, programmers find it easy to learn and use. Therefore, we decided to use PHP to realize the web site.

5.1.2 JavaScript

JavaScript is also a script language most often used for client-side web development, embedded in or included from HTML pages and interacting with the Document Object Model (DOM) of the page. Existing JavaScript libraries provide functionalities such as dynamic visual effects and user interface elements.

5.1.3 MySQL

MySQL is a relational database management system (RDBMS). It is an Open Source Software and works on many different system platforms. MySQL is popular for dynamic web applications because it is often used with Apache web server and PHP. Different popular systems are built on these components such as MediaWiki software (on which runs Wikipedia), which is written in PHP and uses MySQL.
5 Implementation

5.1.4 HTML

HyperText Markup Language (HTML) is a markup language to structure contents (text, images and links) in documents. It is principally used in the web.

5.1.5 CSS

Cascading Style Sheets is a stylesheet language for structured documents. It is mainly used to describe the presentation of a document written in a markup language such as HTML and XML. The main advantage of CSS is the separation of document content (structure defined with HTML) from document presentation (layout defined with CSS). This improves the adaptability of the layout on different representation systems. CSS is considered as the standard style sheet language of web pages.

5.2 Development environment

The whole code (PHP, HTML, CSS, MySQL and JavaScript) was developed using Notepad++ which is an Open Source code editor for Windows supporting syntax highlighting for the named languages and other features. The web server package XAMPP was used to test the project. It provides an easy installation and configuration of an Apache web server on a local pc system with a MySQL database and interpreters for code written in PHP and PERL.

The web site was tested on Internet Explorer (version 7.0.5730.11) and Mozilla Firefox (version 3.0.1) on Windows XP operating system.

5.3 Structure

Using the programming languages named in Section 5.1 a model-view-controller design pattern is automatically implemented. This very common pattern separates the presentation of the application from the underlying application logic. The three parts of this pattern are:

**Model:** This part represents the application logic. In our application all the functions which interact with the database (selecting or storing data) belong to this part. The “hard work” will be done in this part.

**View:** The view represents the graphical user interface. In our application this is the CSS-code to visualize the content of the web page.

**Controller:** The controller joins the view part to the model part. In our application this is PHP and HTML code.
5.4 DB schema

The database is quite simple:

The central table is Webservice with the attributes name, uri, entry date, last voting date and description and the primary key id. Every Webservice has a user assigned as its poster and can be bookmarked by other users.

Users are stored with id, name, surname, email address and username and password to log on. To every Webservice tags can be associated and are saved with id and name. Webservice acts as a blogentry, so comments, which are saved with author, email address, text and the entry date, are enabled. Every rating of a Ws is stored in Feedback with numbers for the overall, quality, time and for the costs estimation and the voting date.
5 Implementation

5.5 Structure of the web page

The PHP and CSS code in Fig. 5.3 represent the basic structure of the web site. It is simple structured with div block elements in HTML, which are formatted with CSS and in which PHP-code from other files is included with the `include()` statement. This statement includes and evaluates the specified file. So the different contents are loaded in the different parts of the site. The login-PHP-file - for example - is loaded in the first box on the left.

![Figure 5.3: (a) index.php (b) index.css](image_url)

The mainframe/index.php file loads content depending on the parameter value `$_GET['section']` in the main frame of the site, which is done with a switch-case control flow. In Fig. 5.4 is a simplified code segment of the file.
5.5 Structure of the web page

```php
<?php

if(isset($_GET['section'])) {
    switch($_GET['section']) {
    case "FB":
        include "3_feedback/index.php";
        break;
    case "WS":
        include "2_webservice/index.php";
        break;
    case "TAG":
        include "4_tag/index.php";
        break;
    case "selected_ws":
        include "2_webservice/selected_ws.php";
        break;
    case "usp":
        include "5_usp/index.php";
        break;
    default: // if section invalid
        // show home
        include "1_home/index.php";
        break;
    }
}

// if section not set
// show home
include "1_home/index.php";
```
5 Implementation

5.6 Login/Register

5.6.1 Sessions

PHP provides functionalities to preserve data across a sequence of accesses. When a user accesses the website a session is started with `session_start()` and the generated unique session-id is allocated to the user saving it in a cookie on the user side or passed as parameter in the URL. Through this ID the user is identifiable and his data is retrievable in his own session file (which contains the session information) on the web server.

5.6.2 IP check

The implementation of sessions does not guarantee that user information stored in a session is only accessible by the user who has generated the session. So further security arrangements are required such like IP-checking. When the user accesses the site and a session is started the IP-address of the user is saved in the session.

```
<?php
session_start();
if(!isset($_SESSION['IP'])) {
    $_SESSION['IP'] = $_SERVER['REMOTE_ADDR'];
}
```

Figure 5.5: session

On further accesses the IP of the user gets checked if it agrees with the stored IP. If it does not agree an error message is displayed and `die()` terminates the execution of the script.

```
if($_SESSION['IP'] != $_SERVER['REMOTE_ADDR']) {
    echo "ERROR MESSAGE";
    die();
}
```

Figure 5.6: IP check

5.6.3 Quotes handling

For security reasons PHP provides a process that escapes incoming data. The 'Magic Quotes' process (when on) automatically escapes the single-quote, double-quote, backslash and NULL characters with a backslash. One of the magic quote directives is `magic_quotes_gpc` which affects HTTP Request data (GET, POST, and COOKIE). The Strings have to be adapted again because this setting is ON (as default) and can not be changed at runtime. This is done with the function `stripslashes()`, which removes all backslashes from a String except double backslashes (```) which are made into single backslash (`\`).
This control flow and the IP check are on top of the PHP-site, so that every incoming data is escaped.

### 5.6.4 Login

When the user tries to log in, the function login() (see Fig. 5.8) is called, which returns ‘true’ if the user was identified with his password. Then, if the password corresponds to the user, the user-ID is saved in the session.
5 Implementation

5.6.5 Logged on

When a user logs in successfully, three links appear in the side-frame instead of the login form and register button.

The first link lists in the main frame the Ws’s which have posted the logged in user and the second link lists the bookmarked Ws’s of the user. On clicking the logout link the session is destroyed and the login form and register button appear again.

By doing the login the content of the main frame does not change because the previous displayed content is known by the $_GET[] parameters and is recovered. If the displayed content needs a logged in user and the user has just logged out, an alert or an other (related) content is displayed.

5.6.6 Register

When a user clicks on the register-button in the side-frame, a form appears in the main frame. By submitting the form, the username gets checked if it already exists in the database. If true, an error message appears. If the username is still free, the password is checked if it matches the password confirmation. If they match, the register-function is called, or else an error message is shown.

```php
define = "SELECT COUNT(*) AS count FROM user WHERE username = "\'".$_POST['username']\'\"\"
$result = mysql_query($define) OR die(mysql_error());
true = mysql_fetch_assoc($result);
mysql_free_result($result);
if ($true['count'] > 0)
    echo 'that username is already in use - try another name.';
else if ($_POST['password'] != $_POST['password_confirmation'])
    echo 'confirm password';
else
    register($_POST['username'], $_POST['email'], $_POST['name'], $_POST['surname'], $_POST['password']);
```

Figure 5.10: username and password control

The function register() saves the user information as new dataset in the table ‘user’. The password is saved encrypted by the function MD5(). Message-Digest Algorithm 5 (MD5) is a widely used cryptographic hash function.
function register($username, $email, $fname, $surname, $pass) 
{
    $sql = "INSERT INTO 'user' ( 
        'id', 
        'username', 
        'email', 
        'fname', 
        'surname', 
        'password', 
        'right' 
    ) VALUES ( 
        NULL, addslashes($username), 
        $email, 
        addslashes($fname), 
        addslashes($surname), 
        $pass, 
        '0' 
    );

    mysql_query($sql) OR die(mysql_error());
}

Figure 5.11: MySQL insert to register new user
5.7 Search

MySQL has support for full-text indexing and searching optimized for large datasets. In MySQL full-text indexing is an index of the type FULLTEXT which can be only applied on MyISAM-tables and can be created only for CHAR, VARCHAR or TEXT columns. The definition can be given in the CREATE TABLE statement when a table is created, or added later using ALTER TABLE or CREATE INDEX. Figure 5.12 shows the FULLTEXT index definition over the columns “name” and “description” of the table “Webservice”.

```
$db = mysql_connect('localhost', 'root', '');
mysql_select_db('show');

//WebService
$sql = "CREATE TABLE Webservice |
| id INT(10) NOT NULL PRIMARY KEY UNIQUE auto_increment,
| name VARCHAR(255) NOT NULL,
| url VARCHAR(500) NOT NULL UNIQUE,
| entry_date DATETIME NOT NULL,
| last_voting_date DATETIME NOT NULL,
| description VARCHAR(1000),
| poster INT NOT NULL,
| FULLTEXT (name, description) |
);"

$result = mysql_query($sql) or exit ("Can't create the table 'WebService' in the database.");

$query = "SELECT id, name, description
FROM webservice
WHERE MATCH (name, description) AGAINST ('"$string."');

$result = mysql_query($query);
```

Figure 5.12: FULLTEXT index definition

Full-text searching is performed using MATCH(...) ... AGAINST(...) syntax. The columns in which has to be searched are listed in MATCH(...) and the search term is in AGAINST(...). Fig. 5.12 shows the MySQL query which contains the MATCH(...) ... AGAINST(...) syntax.

```
$query = "SELECT id, name, description
FROM webservice
WHERE MATCH (name, description) AGAINST ('"$string."');

$result = mysql_query($query);
```

Figure 5.13: Search Function
5.8 Tagcloud

The tagclouds in the third side frame and in the specific main frame are generated with the `get_tagcloud()` function. First the function retrieves all tags in order of popularity that is the amount of webservices which are tagged with the specified tag. Then it takes the first retrieved tags (specified by the parameter value `$cloud_size`) and sort them alphabetically if needed (specified by the parameter value `$sort`).

The tagcloud in the side frame shows the 20 most popular Ws’s in alphabetical order. The tagcloud in the main frame shows the 100 most popular webservices in the desired order (alphabetical or in order of popularity).

Then the function computes the interval between the most and least popular tag to class each tag to one of ten CSS-classes where the different font-sizes are defined.

```php
function get_tagcloud($cloud_size, $sort) {
    $query = "SELECT tag.name AS tag, nr_webServices AS quantity
            FROM tag_web
            GROUP BY tag.name
            ORDER BY quantity DESC";
    $result = mysql_query($query);

    //get the specified amount of popular tags
    for ($i=0; $i<$cloud_size; $i++) {
        $tag = mysql_fetch_array($result);
        $tagcloud][] = $tag['tag'] . $tag['quantity'];
    }

    //specify whole order ALPHABETICAL
    if ($sort) {
        ksort($tagcloud);
    }

    //get the longest and shortest array values:
    $max_qry = max(array_values($tagcloud));
    $min_qry = min(array_values($tagcloud));

    //find the range of values
    $range = $max_qry - $min_qry + 1;

    //returns the
    $tagcloud = '[';

    //loop throughout tagarray
    foreach ($tagcloud as $key => $value) {
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
        $tagcloud[] = "$key: "$value .';';
    }

    return $tagcloud;
}
```

Figure 5.14: (a) tagcloud function (b) css for tagcloud
5 Implementation

5.9 Algorithm of popular WS’s

To define WS’s as popular a algorithm is used where three parameters are considered:

- Time
- Number of interests
- Rate

These three parameters can be taken just from the dates of the table feedback, where every rating for any WS is stored with the date (and time). They have to fulfil following conditions:

- **Time**: only ratings are considered whose date is not older than a year.
- **Number of interests**: only WS’s are considered whose amount of the specified ratings is greater then the mean of the specified ratings of all web services.
- **Rate**: only the best voted web services are considered.

This is done with two MySQL-queries. The first query (Fig. 5.15) computes the amount of all specified ratings for each WS which has at least one specified rating. Next the mean is calculated with the query’s results.

```sql
9  query = "SELECT COUNT(*) as amount
10  FROM feedback
11  WHERE DATE_ADD(voting_date, INTERVAL 1 YEAR) > NOW();
12  GROUP BY wsbservice
13  ";
```

Figure 5.15: First MySQL query to retrieve popular WS’s

The second query (Fig. 5.16) then retrieves the WS’s which has at least mean specified ratings in order of the average rating beginning with the highest.

```sql
25  query = "SELECT wsbservice, COUNT(wsbservice) AS amount, AVG(nr_overall) as vote
26  FROM feedback
27  WHERE DATE_ADD(voting_date, INTERVAL 1 YEAR) > NOW();
28  GROUP BY wsbservice
29  HAVING amount > "from$x."
30  ORDER BY vote DESC
31  LIMIT 0,500;"
```

Figure 5.16: Second MySQL query to retrieve popular WS’s

5.10 Listing of WS’s in an abstract form

The listed WS’s which are displayed in an abbreviated form (see Section 4.5.1) are structured with the help of the function get_ws($id).

This function first retrieves from the database all the corresponding data of the WS with the ID committed by the parameter value $id from the database. Then it structures the data for displaying and extracts the first 150 characters of the description with the function substr($description, 0, 150).
Then it searches all the related tags of the Ws in the database and structures them. The mean rate and the amount of all ratings of the given Ws are retrieved with a MySQL-query and also structured.
5 Implementation

5.11 Feedback

5.11.1 chart

To compute and display the charts (shown in Fig. 4.15 and Fig. 4.16) first a MySQL-query gets the amount of each mark of the given Ws and saves them in an array.

With this array the function get_stat($array) generates the chart starting by searching the maximum value of the array. This value is divided by a number which indicates the width of the chart, that is the scale-value. In the first case this number is 190 and in the second 80.

For each mark the width of the bar in the chart is calculated and displayed. Additionally the mean of all ratings of the given Ws is computed with them and then formatted with the PHP-function number_format($mean, 2, '.', NULL):

- $mean is the number to format, in this case the mean of all ratings of a specific WS
- 2 is the number of decimal points
- . is the separator for the decimal point
- NULL is the thousands separator, in this case irrelevant

5.11.2 script.aculo.us

To hide and slide out the charts of the quality, time and costs rating script.aculo.us is used.

script.aculo.us is a JavaScript library which is built on the JavaScript-Framework Prototype. It provides visual effects over the Document Object Model (DOM) and so “easy-to-use, cross-browser user interface JavaScript libraries to make your web sites and web applications fly”.

On the script.aculo.us downloads page the latest versions of the libraries prototype.js, scriptaculous.js, builder.js, effects.js, dragdrop.js, slider.js and controls.js are available in a package. They are extracted in the directory /javascripts. To use the provided functionalities, the files prototype.js and scriptaculous.js have to be loaded in the web page (Fig 5.17).

The scriptaculous.js loader script loads automatically in the other libraries. To call upon the functions, HTML script tags or HTML event handler attributes are used. As shown in Fig 5.18 onClick sets the script to run when the user clicks on the link. The script slides out the block-level element “rating3”.

Figure 5.17: Script.aculo.us
Figure 5.18: effect function call
6 Conclusion

In the end we were both satisfied with our project. We learned how tricky it could be to realize a website maintaining a right, clear programming structure to keep the overview of the whole code (PHP, HTML,...) and to use the cascading style sheets accurately. Because although CSS is a web standard developed by the World Wide Web Consortium, different browsers (or yet two different versions of one browser) do not interpret it the same way, so many tutorials and bulletin boards were browsed to get the final result.

With the excursion in the background of our topic we have learned how a Web service would be created and described. We have learned the main terminologies related to the hype Web2.0 and now recognize them much more during the surfing through the World Wide Web.

6.1 Future Work

There are a few features which could be added to the web page to make it even more social. The possibility of building groups for users could be provided. The insertion of a Ws could somehow be checked to avoid some inserts which are not services.
Bibliography

[18] http://www.idea-lounge.net/pro
[20] http://news.zdnet.co.uk/itmanagement/0,1000000308,2130093,00.htm?r=2
List of Figures

2.1 The www.google.com/ig site from Marco Dalle Pezze . . . . . . . 2
3.1 Architecture of a Web service . . . . . . . . . . . . . . . . . . . 8
3.2 Skeleton of a SOAP element . . . . . . . . . . . . . . . . . . . . 9
3.3 example of a SOAP request . . . . . . . . . . . . . . . . . . . . . 10
3.4 example of a SOAP response . . . . . . . . . . . . . . . . . . . . 10
3.5 Skeleton of a WSDL document . . . . . . . . . . . . . . . . . . . 12
3.6 Example of a WSDL document . . . . . . . . . . . . . . . . . . . 12
3.7 Service ontology . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.8 WSMO elements . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
4.1 Sitemap of the website . . . . . . . . . . . . . . . . . . . . . . . . 18
4.2 The website shown with Firefox 3.0.1 . . . . . . . . . . . . . . . . 19
4.3 Header . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.4 The three possible login sequences . . . . . . . . . . . . . . . . . 20
4.5 Form to register a new user . . . . . . . . . . . . . . . . . . . . . 21
4.6 search side frame . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.7 Result of search function with the term “fast” . . . . . . . . . . 22
4.8 Tagcloud of the side frame . . . . . . . . . . . . . . . . . . . . . . 22
4.9 Tabs and subline of the main frame . . . . . . . . . . . . . . . . . 23
4.10 a listed Web service . . . . . . . . . . . . . . . . . . . . . . . . . . 23
4.11 Main frame with active tab “Web services” . . . . . . . . . . . . 24
4.12 Form to add a web service . . . . . . . . . . . . . . . . . . . . . . 25
4.13 Main frame with active tab “Feedback” . . . . . . . . . . . . . . 25
4.14 Tag cloud in alphabetically order . . . . . . . . . . . . . . . . . . 26
4.15 a selected Web service . . . . . . . . . . . . . . . . . . . . . . . . 27
4.16 Rating of a selected Web service . . . . . . . . . . . . . . . . . . 28
4.17 Overall rating and costs rating . . . . . . . . . . . . . . . . . . . 29
4.18 (a) My posted Web services (b) My bookmarked Web services . . 30
5.1 mvc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
5.2 schema of the database . . . . . . . . . . . . . . . . . . . . . . . . 33
5.3 (a) index.php (b) index.css . . . . . . . . . . . . . . . . . . . . . 34
5.4 mainframe/index.php . . . . . . . . . . . . . . . . . . . . . . . . . 35
5.5 session . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
5.6 IP check . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
5.7 quotes handling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
5.8 MySQL query to retrieve the user . . . . . . . . . . . . . . . . . . 37
5.9 Login . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
5.10 username and password control . . . . . . . . . . . . . . . . . . . 38
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.11</td>
<td>MySQL insert to register new user</td>
<td>39</td>
</tr>
<tr>
<td>5.12</td>
<td>FULLTEXT index definition</td>
<td>40</td>
</tr>
<tr>
<td>5.13</td>
<td>Search Function</td>
<td>40</td>
</tr>
<tr>
<td>5.14</td>
<td>(a) tagcloud function (b) css for tagcloud</td>
<td>41</td>
</tr>
<tr>
<td>5.15</td>
<td>First MySQL query to retrieve popular WS’s</td>
<td>42</td>
</tr>
<tr>
<td>5.16</td>
<td>Second MySQL query to retrieve popular WS’s</td>
<td>42</td>
</tr>
<tr>
<td>5.17</td>
<td>Script.aculo.us</td>
<td>44</td>
</tr>
<tr>
<td>5.18</td>
<td>effect function call</td>
<td>45</td>
</tr>
</tbody>
</table>