HARMOSEARCH
the future of information services

FP7-SME-1
Project no. 262289

HARMOSEARCH

Harmonised Semantic Meta-Search in
Distributed Heterogeneous Databases

SEVENTH FRAMEWORK PROGRAMME

D8.2 DOCUMENTATION FROM TRAINING AND KNOW-HOW TRANSFER

Due date of deliverable: 2013-02-28
Actual submission date: 2013-03-21

Start date of project: 2010-12-01 Duration: 27 month

Project funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	

PROJECT ACRONYM: **HARMOSEARCH**

Project Title: Harmonised Semantic Meta-Search in Distributed Heterogeneous Databases

Grant Agreement: 262289

Starting date: December 2010 **Ending date:** February 2013

Deliverable Number: D8.2, final version

Title of the Deliverable: Documentation from training and know-how transfer

Lead Beneficiary: EC3 Networks

Task / WP related to the Deliverable: WP 8, Task 8.3

Type (Internal or Restricted or Public): Public

Author(s): Manfred Hackl

Partner(s) Contributing: [X+O], EC3 Networks, Promoter

Contractual Date of Delivery to the CEC: February 28th 2013

Actual Date of Delivery to the CEC: March 21st 2013

PROJECT CO-ORDINATOR

Company name: [X+O]

Name of representative: Manfred Hackl

Address: Siebensterngasse 4/22, 1070 Vienna, Austria

Phone number: +43-676-842755-100

Fax number: +43-676-842755-599

E-mail: manfred.hackl@xpluso.com

Project WEB site address: www.harmosearch.org

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

A condition for the successful exploitation of the project results is that not only the software tools are passed on from the RTD-partners to the SME partners, but also the knowledge how they are built and used. This is an crucial issue since the SMEs will more or less depend on their own in using the results after the project ending date.

Training activities and know-how transfer have therefore had an important role in the project setup and in the project execution and where kind of omnipresent in all activities. This deliverable summarises the activities and results of this work, which was mostly carried out by the RTD-partners. Where necessary it refers or links to other sources of information.

1.2 RELATIONSHIP WITH OTHER DOCUMENTS

Much of the know-how is covered by other deliverables, which are listed below in chapters “Technical documentation” as well as “User documentation”. This document is only seen as a meta-document summarizing and listing the documentation covered by other deliverables.

1.3 STRUCTURE OF THE DOCUMENT

The document contains of three main chapters.

The first one, “Type of ”, describes the different means and forms how the training and know-how transfer was done.

The second one, “Technical documentation”, lists all the technical documentation available also after project end, excluding only inline documentation integrated into the source code.

The third one, “User documentation”, lists all the user documentation available also after project end that can be used by the SMEs as well as by users of the system.

Thus the two later chapters do not give a full listing of documentation, since they do not cover meetings during the project duration. They only reflect what is documented for later.

2 TYPE OF KNOW-HOW TRANSFER

2.1 PHYISICAL AND VIRTUAL MEETINGS

The consortium held 7 physical meetings, in which most of the time was dedicated to know-how transfer. System components were presented by the RTD-partners, explained and discussed and the usage was trained to the SMEs and also other partners.

This started already in the very first meeting in Vienna, where the technical basis, pre-existing material and the ontology where explained.

This is the full list of physical meetings in which know-how transfer tool place:

1. 06.-7.12.2010, Kick-off Meeting, Vienna, Austria
2. 31.03.-01.04.2011 2nd Consortium Meeting, Pisa, Italy
3. 27.-29.07.2011 3rd Consortium Meeting, Berlin, Germany
4. 02.-04.11.2011 4th Consortium Meeting, Paris, France
5. 15.-17.02.2012 5th Consortium Meetina, Vienna, Austria
6. 19.-22.06.2012 6th Consortium Meeting, Cavalese, Italy
7. 06.-08.02.2013 7th Consortium Meeting, Vienna, Austria

During these meetings all components produced or improved since the last meeting were trained in its construction (architecture, components, structure – thus the technical issues) and in its usage. This was done not only for software components, but also e.g. the query mapping. This interactive know-how transfer was very effective and was the main pillar in learning to understand and use the system.

In addition, two special online training sessions were organized for the use of the mapping tool internally. And the mapping tool was also instructed to museum representatives (non-project members) in following meetings, even the tool was not fully ready during some of the events:

1. 27.05.2011 Adriamuse (museum cooperation project in the Adriatic area) kick-off in Rimini, Italy
2. 30.10.2011 Adriamuse kick-off in Croatia
3. 16.10.2012 "Conference of working group documentation" of German Museums Association, Berlin, Germany

2.2 DELIVERABLES

Most of the deliverables cover technical knowledge of the project results and are therefore the main documentation for the time after the end of the project. One deliverable is also a dedicated user manual (D6.4 Manual for the mapping tool).

2.3 ONLINE SOURCES

The online sources available are the user-help inside the HarmoSearch portal and the consortium-internal online knowledge base, built with a wiki system.

The user-help on the portal, called “Support Center” shall guide the user in using the system in the different use cases or actions available.

The consortium internal wiki is a collection of technical and user documentation and was partly also the starting point for some of the deliverables (like D6.4 Manual for the mapping tool) and also for the Support Center (also covered in D7.2 Running prototype).

2.4 VIDEOS

Eventually two small videos have been produced to help users with typical use cases when using the mapping tool. They are not a full documentation, but rather between some promotion to evaluate the potential use of the system and some general guidance when using it.

3 TECHNICAL DOCUMENTATION

As said before, the technical documentation is available in two sources: First some of deliverables and second on the project-internal wiki.

The deliverables with relevance for the technical documentation are listed in the following paragraph, but without deeper discussion. The names are self-explaining and each of them has been submitted to the commission:

1. D2.2 Architectural design
2. D3.1 Ontology for the query model
3. D3.2 Ontology for the registry model
4. D3.3 Extension to the Harmonise Ontology for metadata representation
5. D4.1 Semantic query – Query language specification
6. D4.2 Implementation of query processor as contribution to mapping application
7. D4.3 Metasearch engine (covering also the manual for the webservices)
8. D5.2 Registry component
9. D6.2 Final mapping tool
10. D7.2 Running prototype

The following technical documentation available on the wiki has been added as annex to this document (in order not to disturb an overview of the documentation when reading this document):

1. Core Metasearch Engine Installation and Configuration
2. Semantic Registry Installation and Configuration
3. Crawler Installation and Configuration
4. Mapping Tool Installation and Configuration

4 USER DOCUMENTATION

The two main sources for user documentation are the Support Center on the portal, covered also by "D7.2 Running prototype" and the mapping tool tutorial, covered fully by "D6.4 Manual for the mapping tool". In addition some internal user documentation is available on the project wiki as well as the two videos.

Below is a screenshot of the Support center's home page and a detailed page.

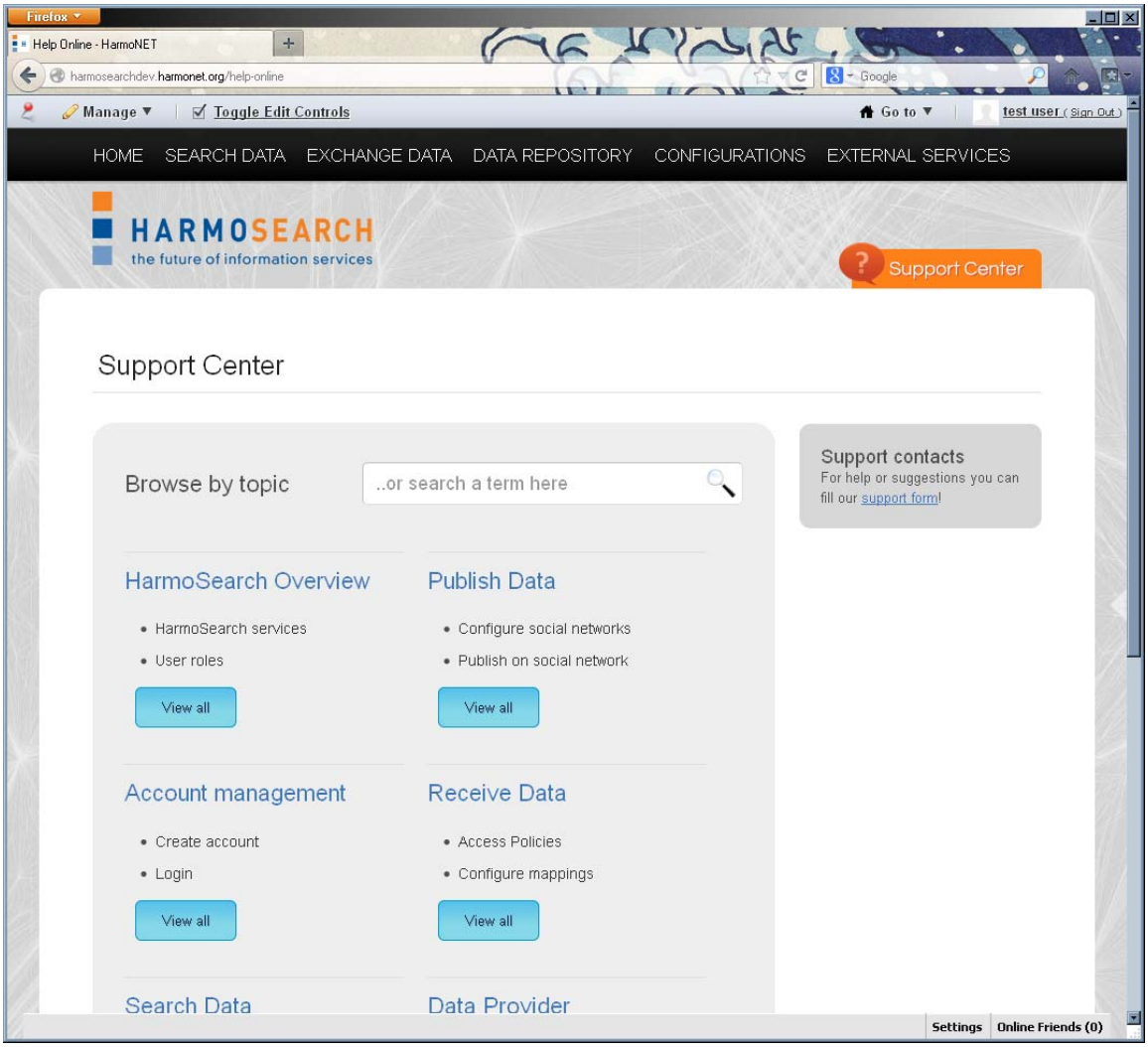


Figure 1: Screenshot of Support Center homepage

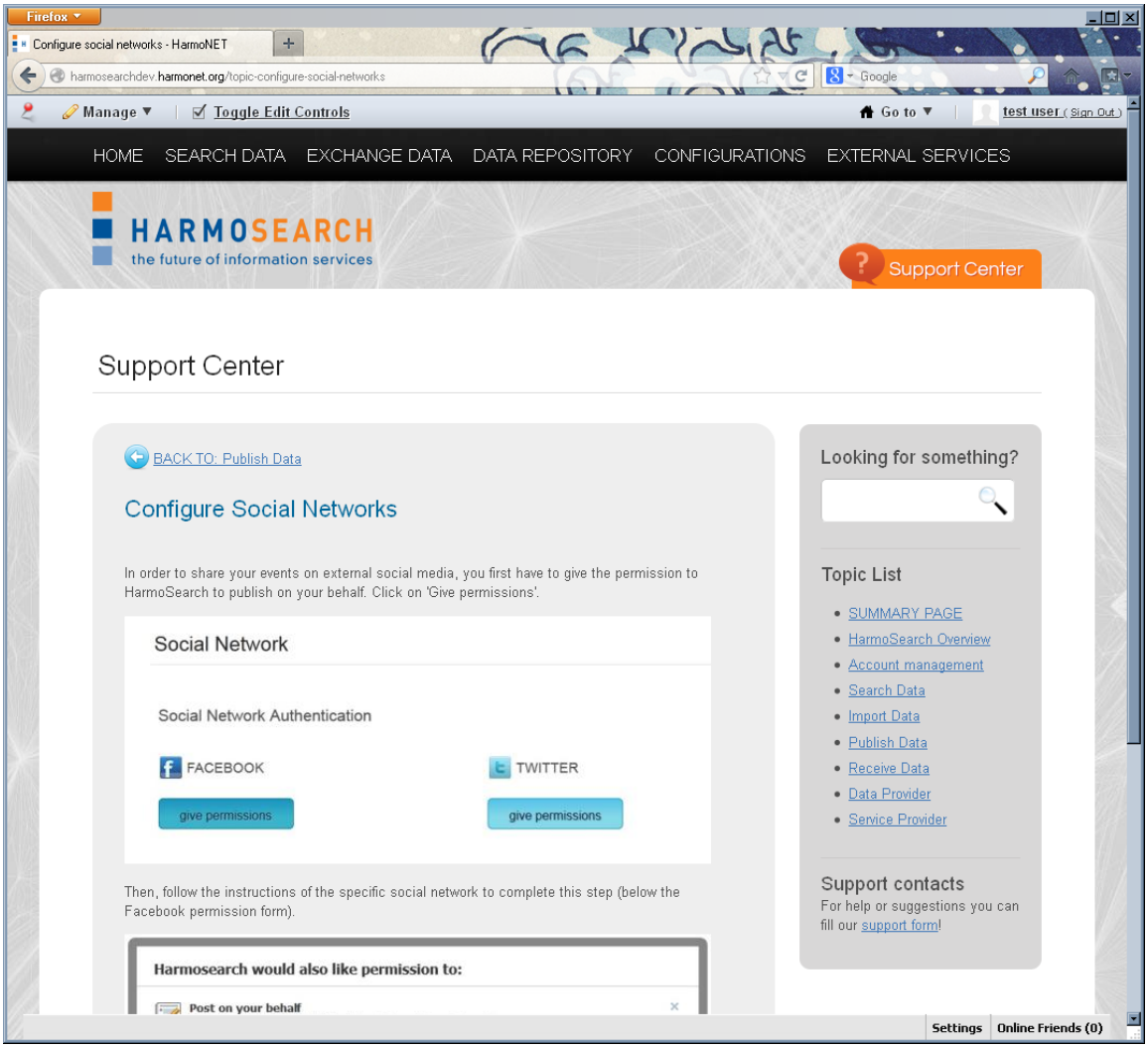


Figure 2: Screenshot of Support Center detailed page

The deliverable "D6.4 Manual for the mapping tool" serves as reference and is written for domain experts that have little or no experience in creating a mapping. The tools main functionality is thereby explained by means of a concrete business scenario which will be used throughout the manual.

Besides these two documentation sources the project wiki covers also the following tutorials, which explain how to configure and use the respective components. Both can be found in the annex as well.

1. Semantic Registry Tutorial
2. Crawler Tutorial

Eventually the project consortium produced two videos, which are available on the project website (http://www.harmosearch.com/index.php?id=69) as well as on youtube (links are on the website as well).

This videos are:

1. XML Data integration with HarmoSearch
   This video tutorial focuses on XML Data integration using the HarmoSearch mapping tool. The example shows the simple mapping from one xml schema to another xml schema.
2. Exporting a mapping project with HarmoSearch
   This video tutorial illustrates how existing mapping projects can be exported to an archive file using the HarmoSearch mapping tool. The resulting archive file can be shared among other users.

5 ANNEX

5.1 CORE METASEARCH ENGINE INSTALLATION AND CONFIGURATION

5.1.1 Runtime Installation

- install jdk 1.6.x on your environment
- download http://sourceforge.net/projects/lportal/files/Liferay%20Portal/6.0.6/liferay-portal-tomcat-6.0.6-20110225.zip/download in harmosearch/Code_Repository/server and unzip there
- delete from harmosearch/Code_Repository/server/liferay-portal-6.0.6/tomcat-6.0.29/webapps the following applications:
    - sevencogs-hook
    - sevencogs-mobile-theme
    - sevencogs-theme
- install postgres 9.0.x
- download the postgres jdbc driver version 9: http://jdbc.postgresql.org/download.html, JDBC4 Postgres Driver and replace server/liferay-portal-6.0.6/tomcat-6.0.29/lib/ext/postgresql.jar with the downloaded one

5.1.2 Configuration

Configure the liferay installation path

- copy harmosearch/Code_Repository/server/configuration/portal/lib/portal-ext.properties into harmosearch/Code_Repository/server/liferay-portal-6.0.6/tomcat-6.0.29/lib/
- open the copied file and change the path to reflect the path to your liferay installation (liferay.dir)

Postgres database

- create a user harmosearchdev with password !!harmosearchdev
- create a database harmosearchdev assigned to this user
- import harmosearch/Code_Repository/server/data/db/harmosearchdev.backup into this database
- create a user lpharmosearchdev with password !!lpharmosearchdev
- create a database lpharmosearchdev assigned to this user
- import harmosearch/Code_Repository/server/data/db/lpharmosearchdev.backup into this database
- create a user hyperjaxb3 with password hyperjaxb3
- create a database hyperjaxb3 assigned to this user
- import harmosearch/Code_Repository/server/data/db/ hyperjaxb3.backup into this database

Datastore

- create the directory harmosearch/Code_Repository/server/liferay-portal-6.0.6/datastore
- copy harmosearch/Code_Respository/server/data/QueryLanguage.xsd and harmosearch/Code_Respository/server/data/city_country.xml into harmosearch/Code_Repository/server/liferay-portal-6.0.6/datastore

Hosts file

- Modify the hosts file in C:/Windows/System32/drivers/etc/hosts by adding a new host: 127.0.0.1 euromuse.ectrldev.org

Application context settings

- open harmosearch/Code_Repository/dev/liferay-plugin-sdk-6.0.6/portlets/WebAccessPortal-portlet/WEB-INF/context/applicationContext.xml
- set the property testMode to TRUE
- set the property testEmail to your email address
- set the property baseDir to the location of your server/liferay-portal-6.0.6/datastore
- set the property portalURL to the url of the liferay portal on your machine, which is http://euromuse.ectrldev.org:8080

Build configuration

- Copy harmosearch/Code_Respository/dev/liferay-plugin-sdk-6.0.6/build.yourloginname.properties by replacing yourloginname by your actual login name in the operating system
- open the new file and change the path to reflect the path to your server
- To compile you can use the Ant command 'ant all' from within \dev\liferay-plugin-sdk-6.0.6\portlets\WebAccessPortal-portlet. You must also compile the CustomCreateHook from within \dev\liferay-plugin-sdk-6.0.6\hooks\CustomCreateUser-hook.

5.1.3 Start the system

Deploy

- copy into harmosearch/Code_Repository/server/liferay-portal-6.0.6/tomcat-6.0.29/lib/ext/:
  - harmosearch/Code_Respository/registry/RegistryServiceNew/dist/RegistryService.jar
  - /dev/liferay-plugin-sdk-6.0.6/portlets/WebAccessPortal-portlet/WEB-INF/SocialNetworksApiComplete.jar
  - harmosearch/Code_Respository/registry/RegistryCoreNew/lib/commons-logging-1.11.jar
- copy into harmosearch/Code_Repository/server/liferay-portal-6.0.6/deploy/:
  - harmosearch/Code_Respository/liferay-plugin-sdk-6.0.6/dist/WebAccessPortal-portlet.war
  - harmosearch/Code_Respository/liferay-plugin-sdk-6.0.6/dist/CustomCreateUser-hook.war

- harmosearch/Code_Respository/registry/RegistryCoreNew/dist/RegistryCore.war
- harmosearch/Code_Respository/registry/RegistryPortlet/dist/RegistryPortlet.war

Launch the application

- Run harmosearch/Code_Repository/registry/Fuseki-Server/fuseki-init-snapshot.bat.
- Run harmosearch/Code_Repository/registry/Fuseki-Server/fuseki-start.bat and keep it running.
- Start Tomcat
- Access http://euromuse.ectrldev.org:8080
- Administrator is 'HarmoNET' (credentials in harmosearch/Code_Repository/server/install/readme.txt)

5.2 SEMANTIC REGISTRY INSTALLATION AND CONFIGURATION

5.2.1 Semantic Registry Components

This section gives an overview of the different components of the HarmoSearch semantic registry, their purpose and where to find them on the HarmoSearch SVN.

Registry Service (jar file)

The registry service is a common component that enables the sharing of functionalities between registry portlets. It is a JAR file that offers a Java interface for accessing the registry's functionalities. Furthermore, it offers the possibility for any servlet or portlet to register as the implementation provider for these interfaces.

The registry service jar file has to be deployed in the container's (e.g.,. tomcat's) common libraries path (e.g., tomcat/lib). This makes sure the same classes are accessed by all deployed servlets (and portlets). Note that the registry service library is also required by any portlet making use of the registry access jar convenience library!

The registry service component can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/RegistryService
```

Find a detailed description of the registry service here.

Registry Core (servlet)

The registry core contains the main business logic of the semantic registry. It is the component that actually handles requests, e.g., to create a new data provider or to check which ones are relevant for a given HarmoSearch query. On the data layer, the registry's triple store is queried using SPARQL over HTTP queries. The HTTP endpoint at which the triple store can be reached is also configured in this component.

Upon being deployed on a servlet container (e.g., tomcat), the registry core servlet tries to register itself as the implementation provider for the registry service interfaces.

The registry core component can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/RegistryCore
```

Find a detailed description of the registry core component here.

Triple Store

The triple store serves as the semantic database for the HarmoSearch registry. Its task is to store all the information required by the registry core and to apply a semantic reasoner on these data in order to draw the correct conclusions. The triple store is required to implement the SPARQL over HTTP specification, since it is queried by the registry core in this way. Which HTTP endpoint to use is configured in the core component.

We have two triple stores available, Fuseki and Sesame. For a productive environment currently only Sesame can be used. The following sections give a brief overview of each.

Fuseki Server (server)

Fuseki is a SPARQL over HTTP server based on the Jena system. In the provided configuration it uses the included TDB database for persistence and the added Pellet reasoner for complete OWL-DL reasoning. OWL-DL is required for enabling complete reasoning with respect to the compatibilities of different individually described sub domains in the HarmoSearch semantic registry. See deliverable D3.2 for details on this topic.

With the provided scripts, Fuseki is fast, easy to use and easy to handle. The drawback is that due to some not fully understood problems between TDB and Pellet, the data becomes corrupted when the server is stopped. For this reason and until there may be an update fixing this issue, Fuseki is recommended only for development purposes. There, its light weight and ease of use are invaluable, especially if experiments are prone to corrupt the database, requiring a reset anyway.

The Fuseki server can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/Fuseki-Server
```

Find a detailed description of the Fuseki server here.

Sesame Server (servlet)

The Sesame server is another triple store that can be used as the semantic database for the HarmoSearch registry. It is deployed as two servlets, one for the server itself and one for a "workbench" which allows for easy manipulation and setup of the server. Sesame implements the SPARQL over HTTP protocol. In the provided edition it is bundles with OWLIM Lite, a semantic reasoner supporting the OWL-RL profile.

This is not as powerful as OWL-DL and does not fully support the sub domain reasoning as explained in deliverable D3.2. However, for practical purposes the explicit hierarchy of the sub domains should be sufficient. At least until a scenario is encountered that is dependent on such individual sub domain description. At which time a newer version of Fuseki and Pellet may overcome their problems.

Sesame is a very stable productive environment, but slightly more heavy weight than Fuseki. See the detailed description for how to set up Sesame correctly in order to work with the HarmoSearch semantic registry.

The Sesame triple store can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/Sesame-Owlim-Server
```

Find a detailed description of the sesame triple store here.

Registry Webservice (servlet)

The web service access for the semantic registry is implemented in a standalone servlet. It needs to be deployed in the same servlet container (e.g., tomcat) as the registry core and the registry service must be available on this container. If these prerequisites are fulfilled, then the registry web service servlet will make a web service facade available for the services exposed by the registry core. The communication with the registry core servlet happens through the API interface proxy realised in the registry service.

The web service servlet offers web services in both REST and SOAP style. A small test web application is also available to test some of the provided functions.

The registry web service component can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/RegistryWebservice
```

Find a detailed description of the registry web service component here.

Registry Portlet (portlet)

The registry portlet is a portlet application which actually provides two portlets: one for testing the metasearch functions of the semantic registry and one for managing the data stored in the data registry part of the semantic registry. For a detailed description of the functionalities of these portlets see the Semantic Registry Tutorial. For a more detailed description on the data stored in the semantic registry see deliverable D3.2.

The registry portlet has to be deployed on a portlet container. In the HarmoSearch production environment this will always be the Liferay portal, whereas for testing and development purposes also the Pluto portal can be used. The registry portlet accessed the semantic registries functionalities through the convenience classes provided by the Registry Access JAR.

Note that the portlet for managing the registry's data is currently only a proof-of-concept prototype and needs to be improved with respect to completeness and usability.

The registry portlets can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/RegistryPortlet
```

Find a detailed technical description of the registry portlet here.

Details on how to use the portlets can be found in the Semantic Registry Tutorial.

5.2.2 Eclipse Projects

All components of the semantic registry except for the triple stores and servers are distributed as Eclipse projects. The following section gives details about the project structure and how to build the components. Basically the projects are standard Eclipse dynamic web projects and all make use of the Spring framework including Spring MVC and Spring Portlet MVC. All required jar files should be packaged with the projects. Opening them in Eclipse should not cause any problems.

Registry Access JAR (jar file)

The registry access project produces the *RegistryAccess.jar* file which is a support library to access registry functionality. It contains classes for easy access of registry functionality and a sample config file. A Java application using the registry access jar should provide an instance of

such a config file somewhere in its classpath. This config file manages the way to access the registry - by API or by web service.

The registry access jar file provides access to the registry functionalities through classes implementing static wrapper methods for all functionalities provided by the registry core. It handles the decision whether to access these functionalities directly or via web service request and make the appropriate calls.

Note that any application using the registry access convenience classes also has to have access to the registry service library!

The *registry access jar* project can be retrieved from SVN at:

```
https://62.149.192.167/repos/harmosearch/Code_Repository/registry/RegistryAccessJar
```

ANT Buildfiles

The output file (jar or war) of every project can either be built using the appropriate eclipse functionalities or, more conveniently, by using the provided ANT script files. The build.xml file is located in the root folder of every Eclipse project and is compatible with the way Eclipse builds the binaries. The ANT build files can also be used from within Eclipse for quick testing. For this purpose *deploy* targets have been created which deploy the war and jar files to local server installations. The respective paths to the deployment directories have to be changed if these deploy targets are to be used on different installations.

The deployable files (WAR or JAR) created by the ANT build scripts are always placed in the "dist" sub folder of the Eclipse project.

Build Libraries

One specific thing to note about the project setup is the *lib* directory in the root directory of the Eclipse projects. This *lib* directory contains libraries which are required for building the relevant files (war, jar) but which are not compiled into the output. As an important example, this is the place where the *RegistryService.jar* file has to be placed in order to compile any project that is dependent on it.

At build time the compiler can access the RegistryService.jar file from this location in order to compile the project. However, the *RegistryService.jar* is **never** packaged into the respective output files. Instead it has to be set up in the common library folder of the container (e.g., tomcat).

5.2.3 Installation and Configuration

This section gives an overview of how to deploy the HarmoSearch semantic registry as a whole. For detailed information about how to set up each of the components, refer to the detailed descriptions.

Basically all components except for the Fuseki server are deployed on a servlet container (e.g., tomcat).

And the triple store's endpoint needs to be configured.

Installation Overview

Basically, the different components have a very loose coupling between them. The only two components which are required to work together in the same container (e.g., tomcat) are the

registry core and the registry web service. In order for them to work together, the container has to be fitted with the registry service jar file.

Otherwise the components can be deployed in various ways, two of which are relevant for real use and are explained below. The interaction between components is either through API calls or through web service (WS) calls. In order to enable API calls, the communicating components must be deployed in the same container and that container has to have the registry service jar file deployed. In order to enable WS calls, the registry core and registry web service components must be deployed together as described above.

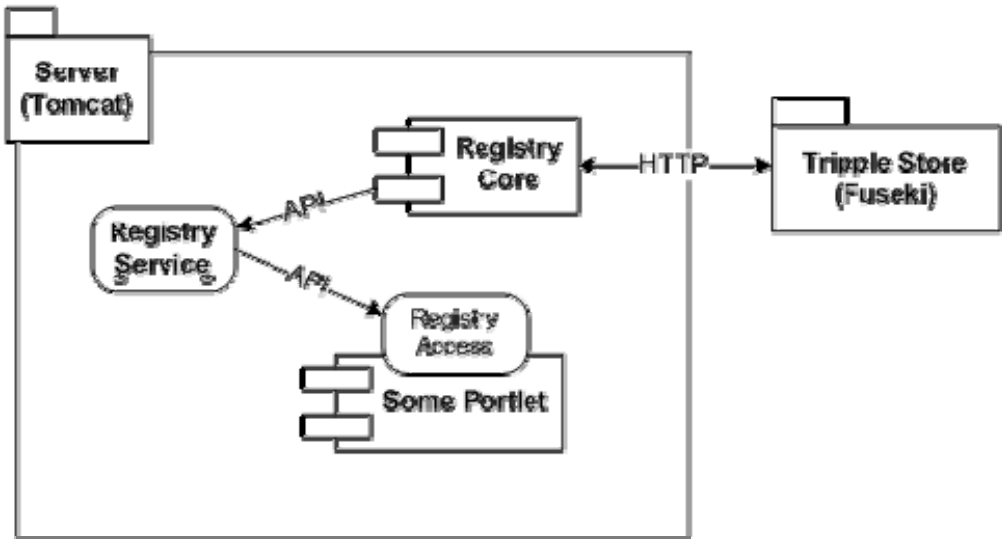
The only configuration that is required in order to fit the system together is to set the correct HTTP endpoint for the triple store in the registry core (see Registry Core Component) and possibly the configuration for the registry access classes provided by the registry service jar file. The latter is used by the portlets or servlets making use of registry functionalities (where each can have its own configuration).

Core and portlet in the same container

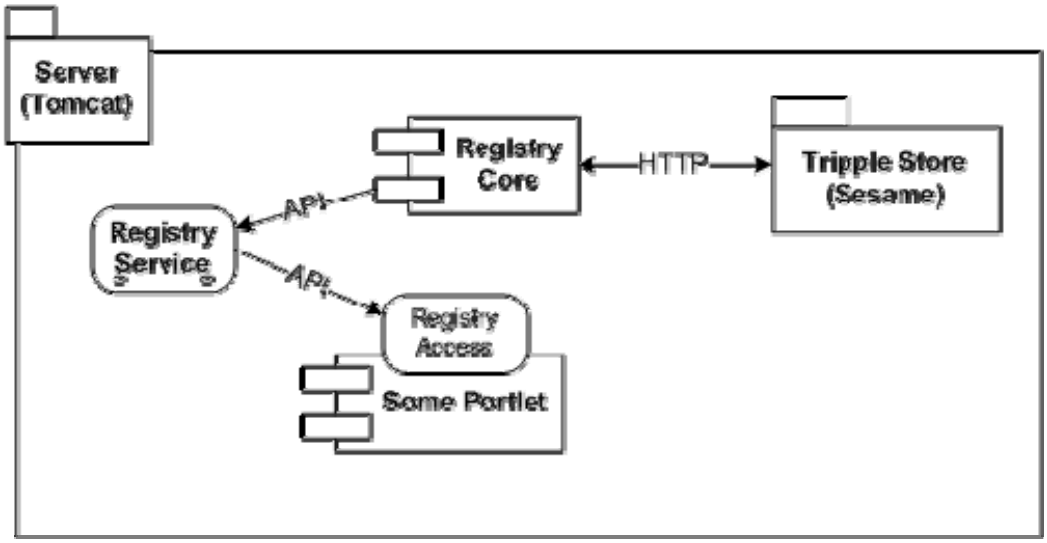
In terms of easy maintenance and convenient configuration it appears desirable to deploy the registry components all in one container (e.g., tomcat). In order for the registry portlets to work, this container also has to have a portal server deployed (e.g., Liferay). Furthermore, the registry service jar file has to be deployed in the common lib directory of the container (e.g., *tomcat/lib*). Note that any other portlet that wishes to make use of registry functionalities can do so in the very same way the registry portlets do, by making use of the registry access jar file.

Deploying the web service component with the other components is possible but not required. Unless the configuration for the registry access jar file allows for API access, all communication between the registry portlets and the registry core does not require web services to be available. In order to expose the web service access to the outside world, the web service component can be deployed too.

Finally, there are two possibilities regarding the triple store. Either Fuseki can be used, in which case the triple store is deployed as a standalone server. The setup then looks like this:



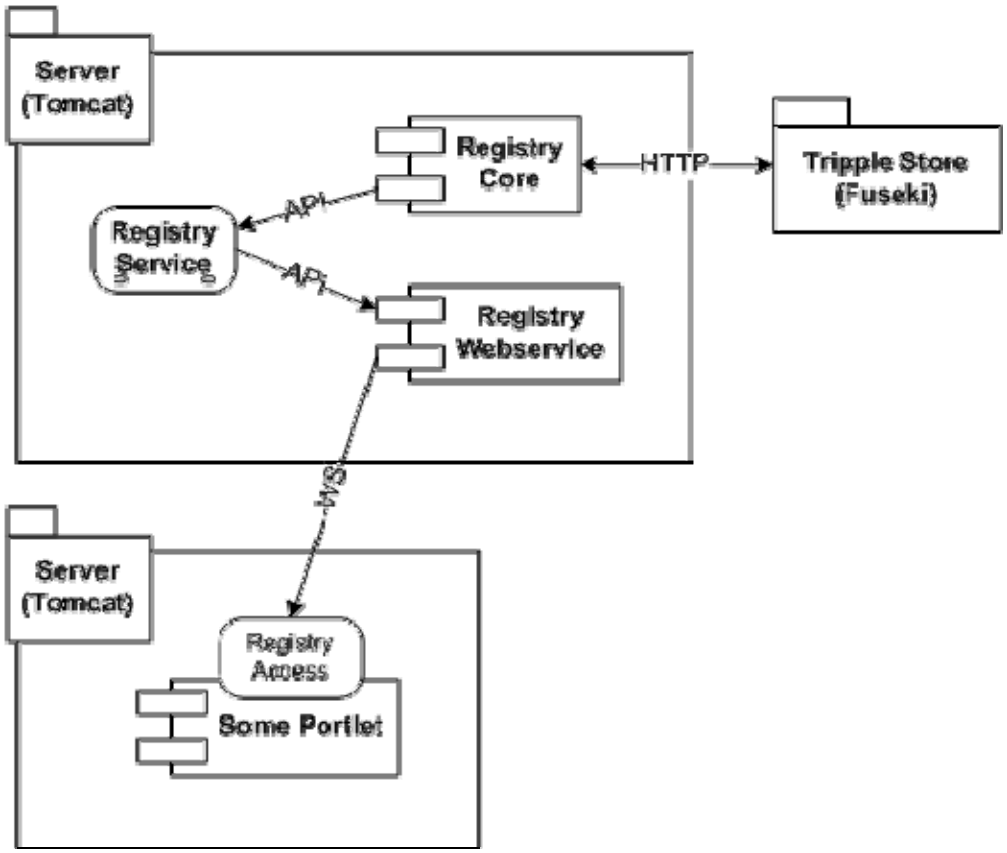
Or the Sesame triple store can be used which is deployed as another portlet. In this case it makes sense to also deploy Sesame in the same container (e.g., tomcat) as the other components. The setup then looks like this:



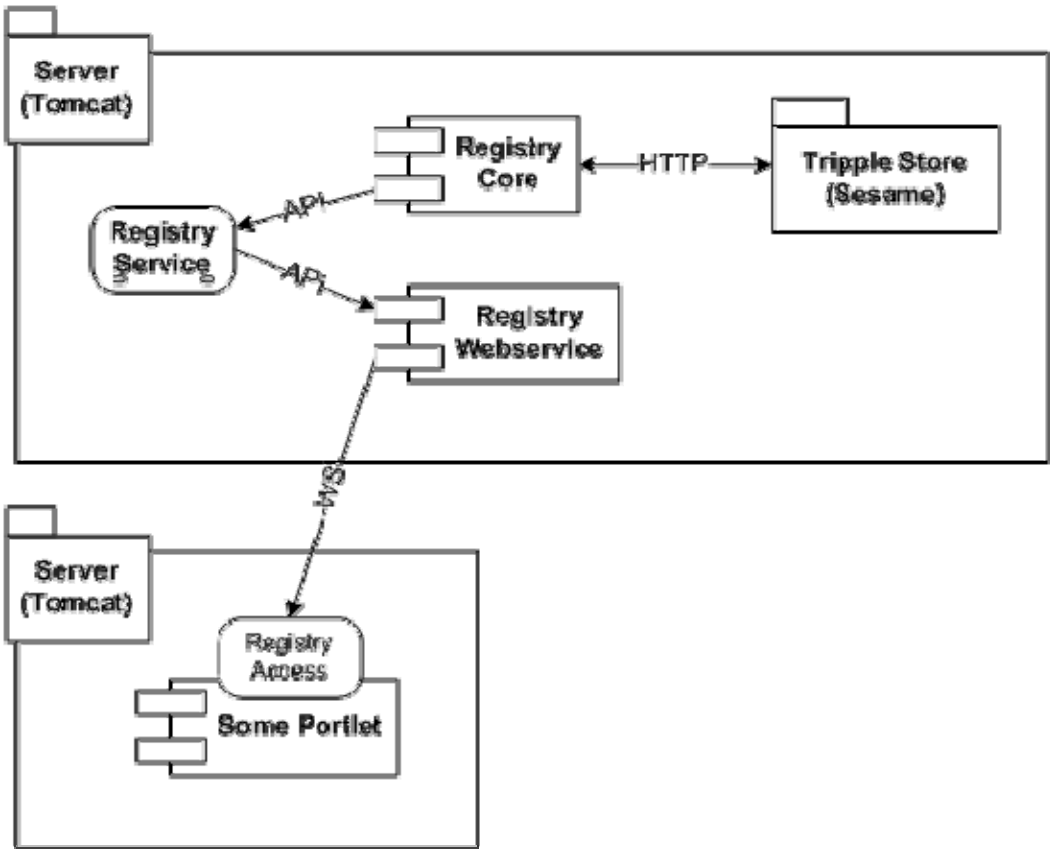
Core and portlets in different containers

As a second important setup scenario, it is also possible to have the registry core and web service components running in a dedicated container (e.g., tomcat) and the portlets or other servlets making use of registry functionalities in an other container (e.g., tomcat with Liferay portal deployed). In this scenario the container running the registry core has to have the registry service jar file deployed in its common lib directory. The portlets or other applications on the second container have to access the registry via web service calls. When making use of the registry access jar, this is wrapped transparently (though some configuration might be required, see Registry Access JAR).

Here again either the Fuseki server can be used in which case the setup looks like this:



Or the Sesame server can be deployed, where again it makes sense to deploy it on the same container (e.g., tomcat) as the registry core. In this case the setup looks like this:



Installation / Deployment Order

It is important to deploy the registry service jar file to the servlet container which will hold the registry core first.

After that step, the registry core, registry web service and registry portlets can be installed and started in arbitrary order. However, the triple store (Fuseki or Sesame) must be installed and running before the other components can actually be used.

5.3 CRAWLER INSTALLATION AND CONFIGURATION

5.3.1 Installing Nutch and Solr to crawl and index web pages

Requirements and general settings

The following steps have to be performed on a system where Nutch and Solr should be set up: These are some prerequisites and general system settings.

- Download and install JDK (in this guide JDK1.6.0.23 is used).
- Set up the variables *JAVA_HOME*, *NUTCH_JAVA_HOME* and *CLASSPATH/. In Windows this should look like:*
  - `CLASSPATH = .;C:\Programme\Java\jdk1.6.0_23\bin`
  - `JAVA_HOME = C:\Programme\Java\jdk1.6.0_23`
  - `NUTCH_JAVA_HOME = C:\Programme\Java\jdk1.6.0_23`
- In Windows, add these paths to the *PATH* variable:
  - `PATH = .;C:\Sun\AppServer\bin;%JAVA_HOME%\bin;%ANT_HOME%\bin`

If additional plugins should be (written and) compiled, also the following steps are required:

- Download and install Apache Ant (in this guide Apache Ant 1.8.2 is used).
- Set up the *ANT_HOME* variable:
  - `ANT_HOME = C:\apache-ant-1.8.2`

In Windows, as Apache Nutch is Unix-based, all commands have to be executed in a *Cygwin* terminal. Therefore, Windows-users have to consider this:

- Download and install *Cygwin*
- *Cygwin* emulates a Unix-like environment of a Windows system. Therefore, all major commands of Unix are supported. The usual Windows file system is emulated, and can be found using e.g. `cd /cygdrive/c/` to access the drive `C:\` of Windows.
- For more information, go to http://cygwin.com/cygwin-ug-net.html.

Installing and configuring Nutch and Solr

These steps have already been done if you download the Nutch and Solr directories from the SVN, but in order to have a complete guide, the steps that should be executed when someone wants to install both - Nutch and Solr - again (or to another place):

- Download and unzip Apache Nutch (http://www.apache.org/dyn/closer.cgi/nutch/). When using *Cygwin*, be sure to not hae any spaces in the path (e.g. *Program Files*), which causes problems.
- Download and upzip Apache Solr (http://www.apache.org/dyn/closer.cgi/lucene/solr/). Again, *Cygwin* users should consider to have no spaces in the path.

As the current versions have some problems and conflicts caused by incompatible libraries, some libraries have to be exchanged:

- Exchange the *Solr* libraries:
  - Delete `NUTCH_ROOT/lib/apache-solr-core-1.4.0.jar`.
  - Delete `NUTCH_ROOT/lib/apache-solr-solrj-1.4.0.jar`.
  - Copy `SOLR_ROOT/dist/apache-solr-core-3.1.0.jar` to `NUTCH_ROOT/lib/`.
  - Copy `SOLR_ROOT/dist/apache-solr-solrj-3.1.0.jar` to `NUTCH_ROOT/lib/`.
- Update the *Hadoop* libraries:
  - Delete `NUTCH_ROOT/lib/hadoop-0.20.2-core.jar`.
  - Delete `NUTCH_ROOT/lib/hadoop-0.20.2-tools.jar`.
  - Download and unpack the new version from http://apache.fastbull.org/hadoop/core/hadoop-0.21.0/.
  - Copy the new version `hadoop-common-0.21.0.jar` to `NUTCH_ROOT/lib/`.
  - Copy the new version `hadoop-mapred-0.21.0.jar` to `NUTCH_ROOT/lib/`.
- Add the *Jackson* libraries:

    - Download http://www.jarvana.com/jarvana/archieve-details/org/codehaus/jackson/jackson-mapper-asl/1.0.1/jackson-mapper-asl-1.0.1.jar.
    - Copy `jackson-mapper-asl-1.0.1.jar` to `NUTCH_ROOT/lib/`.
    - Download http://www.jarvana.com/jarvana/archieve-details/org/codehaus/jackson/jackson-core-asl/1.0.1/jackson-core-asl-1.0.1.jar.
    - Copy `jackson-core-asl-1.0.1.jar` to `NUTCH_ROOT/lib/`.
- Add the *Avro* library:
    - Download http://repo1.maven.org/maven2/org/apache/avro/avro/1.5.0/avro-1.5.0.jar.
    - Copy `avro-1.5.0.jar` to `NUTCH_ROOT/lib/`.

After all these libraries have been updated, call `ant job` in the same directory.

5.3.2 Organizing Solr and Nutch

The following steps explain, how Nutch and Solr are organized in the harmonise SVN (in the directory `Code_Repository/`):

- Copy the following files and directories from `NUTCH_ROOT/` to `server/nutch/`:
    - `bin/`
    - `build/`
    - `conf/`
    - `lib/`
    - `plugins/`
    - `urls/` (create this directory)
    - `webapps/`
    - `.project`
- Copy the following files and directories from `NUTCH_ROOT/` to `dev/nutch/`:
    - `bin/`
    - `lib/`
    - `src/`
    - `build.xml`
    - `default.properties`
- Copy `SOLR_ROOT/*` to `server/solr/`.

5.3.3 Configuring Nutch

These configurations are already set up in the harmonise SVN version (but have to be made if Nutch is installed). In this case, edit the following files:

- `server/nutch/bin/nutch`: Add the following
    - Line 134: `for f in $NUTCH_HOME/plugins/**/*.jar; do`
    - `CLASSPATH=${CLASSPATH}:$f;`
    - `done`
- `server/nutch/conf/nutch-default.xml`:
    - Line 28: Remove the limit of file contents `file.content.limit` by setting it to `-1`.
    - Line 62: Change the value of *http.agent.name* to the name of the crawler.
    - Line 158: Remove the limit of the *http* content `http.content.limit` by setting it to `-1`.

    - Line 268: Remove the limit of the *ftp* content `ftp.content.limit` by setting it to `-1`.
- `dev/nutch/src/plugin/build-plugin.xml`: Modify the following lines:
    - Line 28: `<property name="nutch.root" location="${root}/../../../"/>`
    - Line 35: `<property name="conf.dir" location="${nutch.root}/../../server/nutch/conf"/>`
    - Line 37: `<property name="build.dir" location="${nutch.root}/../../server/nutch/build/${name}"/>`
    - Line 41: `<property name="deploy.dir" location="${nutch.root}/../../server/nutch/build/plugins/${name}"/>`
    - Line 54: `<pathelement location="${nutch.root}/../../server/nutch/build/classes"/>`
- `dev/nutch/build.xml`:
    - Line 548: `<copy todir="${dist.dir}/plugins"> <fileset dir="${build.plugins}"/> </copy>`
    - Copy the plugin folders containing the *jar* file and the `plugin.xml` also to `NUTCH_ROOT/plugins/`
- `dev/nutch/default.properties`: Modify the following lines:
    - Line 10: `conf.dir = ./../../server/nutch/conf`
    - Line 12: `docs.dir = ./../../server/nutch/docs`
    - Line 16: `build.dir = ./../../server/nutch/build`

The following configurations are site-specific. They have to be updated when pages should be crawled, defining these web pages:

- `server/nutch/conf/crawl-urlfilter.txt`
    - Line 40: replace `MY.DOMAIN.COM` with first urls to crawl.
    - Add one line (with same formatting) for each url.
- `server/nutch/conf/regex-urlfilter.xml`:
    - Modify regex expressions to not exclude queries (and facebook urls) from being crawled.
- `server/nutch/urls/nutch`:
    - Add urls to be crawled, one per line.

5.3.4 Configuring Solr

Again, these configurations have already been done in the harmonise SVN version, but are listed here for completeness:

- Copy all files from `server/nutch/conf/*` to `server/solr/conf/`, overwriting any existing files.
- Edit the file `server/solr/example/solr/conf/schema.xml`: In line 71, in the tag `<field name="content"...>`, the `stored` attribute should be changed from `false` to `true`.
- Edit the file `server/solr/example/solr/conf/solrconfig.xml`: In line 702, above the first `requesthandler` tag, add the following:

```
01 <requestHandler name="/nutch" class="solr.SearchHandler" >
02   <lst name="defaults">
 03     <str name="defType">dismax</str>
04     <str name="echoParams">explicit</str>
```

```
05     <float name="tie">0.01</float>
06     <str name="qf">
07       content^0.5 anchor^1.0 title^1.2
 08     </str>
09     <str name="pf">
 10       content^0.5 anchor^1.5 title^1.2 site^1.5
 11     </str>
12     <str name="fl">
13       url
 14     </str>
15     <str name="mm">
 16       2&lt;-1 5&lt;-2 6&lt;90%
17     </str>
18     <int name="ps">100</int>
 19     <str name="q.alt">*:*</str>
 20     <str name="hl.fl">title url content</str>
 21     <str name="f.title.hl.fragsize">0</str>
22     <str name="f.title.hl.alternateField">title</str>
23     <str name="f.url.hl.fragsize">0</str>
24     <str name="f.url.hl.alternateField">url</str>
25     <str name="f.content.hl.fragmenter">regex</str>
26   </lst>
 27 </requestHandler>
```

Adding a new plugin

In the following, the new plugin is always referred to as *newPluginName*.

- Copy or write the Java source files (HTML parser extension, indexer extension, query filter) to `dev/nutch/src/plugin/newPluginName/src/java/`.
- In `dev/nutch/src/plugin/newPluginName/` create the files:
  - `build.xml`, similar to the one of the other plugins, changing in line 5: `file="../build-plugin.xml"`.
  - `plugin.xml`, adapted from the one of the other plugins to the proper one.
- Create the following folders:
  - `server/nutch/build/newPluginName/`
  - `server/nutch/build/plugins/newPluginName/`
  - `server/nutch/plugins/newPluginName/`

Modify the following files:

- `dev/nutch/src/plugin/build.xml`:
  - Add `deploy` and `clean` statements for the new plugin.
- `dev/nutch/src/plugin/build-plugin.xml`:
  - Line 208: add a clean target for the new plugin.
- `dev/nutch/build.xml`:
  - Line 339: add path to new plugin's sources: `<packageset dir="${plugins.dir}/microformats/src/java"/>`.
  - Line 631: add a clean targets for the new plugin: `"${build.plugins}/microformats"` and `${build.plugins.dest}/microformats"`.
- `server/nutch/conf/nutch-default.xml`:

    - Line 984: Modify regex to match new plugin, by adding to the *value* tag: `newPluginName|`.
- `server/nutch/conf/nutch-site.xml`:
    - Line 10: Modify regex to match new plugin, by adding to the *value* tag: `newPluginName|`.
- `server/nutch/conf/schema.xml`:
    - Line 57: Add field types, e.g. for dates, if required: `<fieldType name="date" class="solr.DateField" sortMissingLast="true" omitNorms="true" />`
    - Line 112: Add all fields required for the new plugin which are then mapped in the following to *Solr* fields.
- `server/nutch/conf/solrindex-mapping.xml`:
    - Line 51: Add field pairs for the new plugin to map the crawled fields to their destination in the *Solr* index: `dest` and `source`.

For Solr:

- Again, copy `server/nutch/conf/solrindex-mapping.xml` to `server/solr/example/solr/conf/`.
- Again, copy `server/nutch/conf/schema.xml` to `server/solr/example/solr/conf/`.
- Again, edit the file `server/solr/example/solr/conf/schema.xml`: In line 71, in the tag `<field name="content"...>`, change the `stored` attribute from `false` to `true`.
- In `server/solr/example/solr/conf/custom-fields.xml`: Add field with name and properties, if required.

Compile the plugin:

- Inside `dev/nutch/src/plugin/newPluginName`, call `ant` or `ant jar`.

Starting Solr

Solr can be started from any terminal, independent on the operating system used, by calling inside the folder `server/solr/example/`:

```
java -jar start.jar
```

. During indexing, *Solr* has to be running.

Starting Nutch: Crawling and indexing

Nutch can already be starten from a Unix shell, or by using *Cygwin* in Windows:

- Move to the folder `server/nutch/`.
- Start the crawling process: `bin/nutch crawl urls -dir crawl -depth 2`
    - `-dir` defines the directory to put the crawled data.
    - `-depth` defines the depth of links to follow, starting from the root page.
    - `-threads` defines the number of threads fetched in parallel (optional).
    - `-topN` defines the maximal number of retrieved pages at each level of depth (optional).
- Index the crawled pages (now *Solr* has to be running): `bin/nutch solrindex [http://HOST_ADDRESS:8983/solr/] crawl/crawldb crawl/linkdb crawl/segments/*`

- Exchange `HOST_ADDRESS` by the one which is used, e.g. `localhost`.
- Set the port number correctly, if another one is used (can be looked up in the *Solr* terminal).

In oder to search the index, there are these possibilities:

- The *Solr* admin panel including a request form is located at `[http://HOST_ADDRESS:8983/solr/admin]` .
- Search the XML directly, which is located at the URL one can see in the address bar when getting the results or a request from the admin panel.

Full-text-search In order to search the full text of an event, i.e. not a specified field, like the starting date, or the location, it is necessary to specify the field `event_content`, which causes the search to be restricted to the tag containing the `vevent` specification. The build-in full-text search of *Solr* does also work, but it uses the entire web page which contains the event. As there might be multiple events on a page, and a page could also contain additional information which is not related with the event, this would result in wrong or unwanted search results. Therefore, if e.g. all events related with `meeting` are searched, not specifying whether this term should be contained in the description, the summary or the title, one should use the following query: `event_content:meeting`

Re-crawling It is possible to re-crawl manually all the pages already before the defined timespan has passe, e.g. if it is known that some page has been changed. To do so, the following commands have to be exectued. Note that the parameter value of `adddays` has to be larger than the specified time for re-crawling, where the predefined value is 30 days. Note also that the `depth` should be set to 1 in this case, as the links have already been followed in the previous crawling process and are therefore re-crawled:

```
bin/nutch inject crawl/crawldb urls
bin/nutch generate crawl/crawldb crawl/segments -adddays 31
s1=`ls -d crawl/segments/2* | tail -1`
bin/nutch fetch $s1 -depth 1
bin/nutch updatedb crawl/crawldb $s1
bin/nutch invertlinks crawl/linkdb -dir crawl/segments
```

After having crawled the pages, the indexer has to be called again, using:

```
bin/nutch solrindex [http://localhost:8983/solr/] crawl/crawldb
crawl/linkdb crawl/segments/*
```

Deploy on the server

- go to harmosearch\Code_Repository\server and create a zip file containin the folder nutch and solr
- transfer the file into the server via winscp
- create a tmp directory, move the zip file here, unzip it then run:

```
rm -rf `find . -type d -name .svn`
```

- go to harmosearch\Code_Repository\server\nutch\bin and run:

}

5.4 MAPPING TOOL INSTALLATION AND CONFIGURATION

5.4.1 Quick Details

- Version: 0.4
- Language: English
- File Name (x86): MappingToolSetup_x86_v0.4.exe - Download (restricted access see Installation Instructions)
- File Name (x64): MappingToolSetup_x64_v0.4.exe - Download (restricted access see Installation Instructions)

5.4.2 System Requirements

Supported operation systems: Windows 7, Windows Vista, Windows XP

- One of the following operating systems are necessary:
  - Windows 7 (x86 or x64)
  - Windows Vista (x86 or x64)
  - Windows XP (x86 or x64)
- Java Runtime Kit version 1.6.x or higher - Download

5.4.3 Installation Instructions

1. Download the mapping tool using the link above.
   1. **Note:** The download location is restricted. Accordingly a valid username and password has to be provided.
   2. **Username:** harmosearch **Password:** harmosearch123
2. Install the HarmoSearch mapping tool using the Windows installer.
   1. ***Note:*** *It is important that the installation directory does not contain any spaces, otherwise the tool will not start!*
   2. The default installation folder is **C:\HarmoSearch**
3. In case you want to uninstall the HarmoSearch mapping tool execute *uninstall.exe*

5.4.4 Uninstall Instructions

1. Go the installation location of the HarmoSearch Mapping Tool and open the root folder. Per default the tool is installed to **C:\HarmoSearch**.
2. Execute **uninstall.exe**
3. Follow the instructions of the uninstaller.
4. **WARNING:** Uninstalling the mapping tool will erase all other files in the corresponding installation folder!

5.5 SEMANTIC REGISTRY TUTORIAL

5.5.1 Overview

The semantic registry portlet application is deployed as WAR file on a portal application (e.g., Liferay). In order to be usable, the corresponding pages have to be created on the portal and the portlet applications have to be deployed. See the portal's manual for information on how to do this.

Actually the current implementation of the registry portlet provides two portlets. One is a test application for checking the results of a metasearch request to the registry. The other one is for actually managing the data for Harmonise data providers that is stored in the semantic registry. Both portlets are described in detail in the following sections.

5.5.2 Data Management Portlet

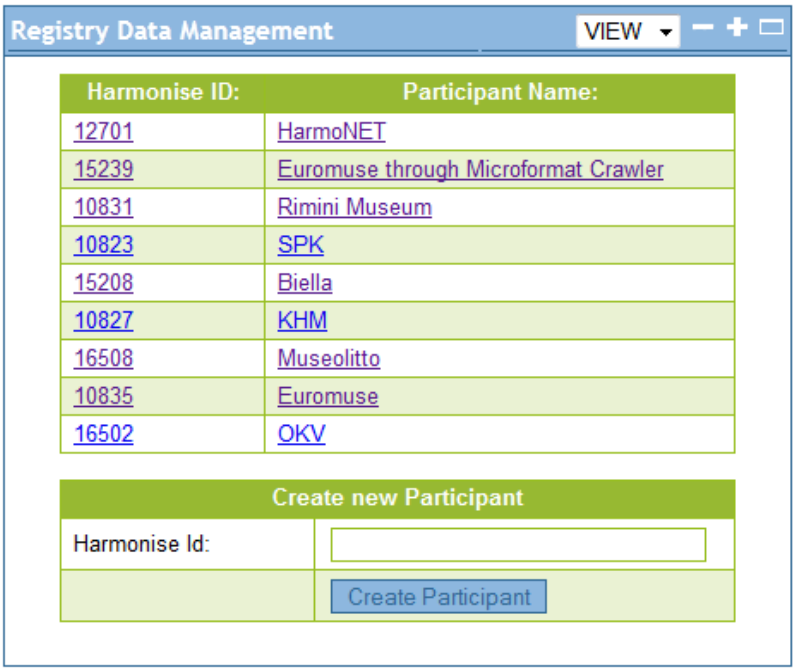
This portlet offers basic functionalities for managing the data Harmonise participants and associated data providers in the semantic registry. Note that these interfaces are subject to major rework and extension and the following description might be outdated at the time reading. Also note that not all of the data items that can be stored in the semantic registry are reflected in this user interface. Its main purpose at the moment is to allow to manage the data required for metasearch processes. This will be extended during the reminder of the project.

For clarification, the following terms are used thoughout the portlet description:

- **Harmonise Participant** - a Harmonise participant is an entity in the semantic registry that corresponds to the organisation a portal user belongs to.
- **(Harmonise) Data Provider** - this describes a queryable data source which is attached to (operated by) a Harmonise participant. A Harmonise participant can have many data providers attached. An example would be a data provider for events and a data provider for attractions, both operated by the Biella participant.

For a detailed description of the data that can be stored in the semantic registry, see deliverable *D3.2 Registry Ontology Model*.

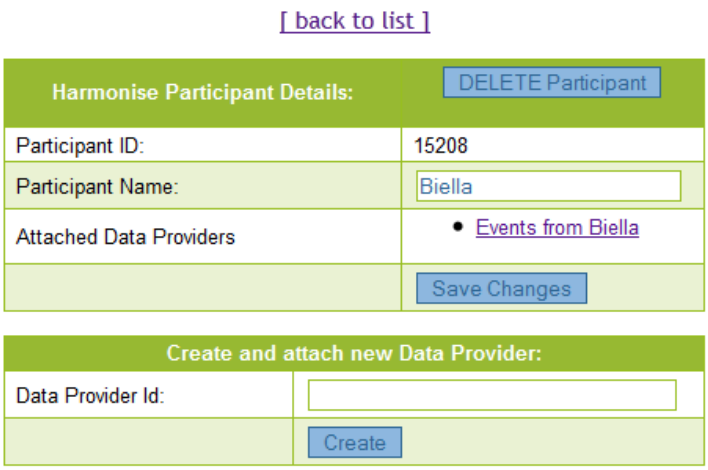
Manage Harmonise Participants



In this view, which should only be available for portal users with administrator privileges, all Harmonise participants currently stored in the registry are shown.

- By clicking on one of the participant names you can go into a more detailed view, allowing to change participant details or delete the participant.
- In the area below the participants list, you can create a new Harmonise participant by entering its Harmonise ID. This should normally be the Harmonise ID of the corresponding portal user (or rather organisation).

When you created a new Harmonise participant or selected an existing one, you will be taken to the following view. This should also be the normal entry view for all portal users without administrator privileges. There, only the information concerning the own organisation should be available.



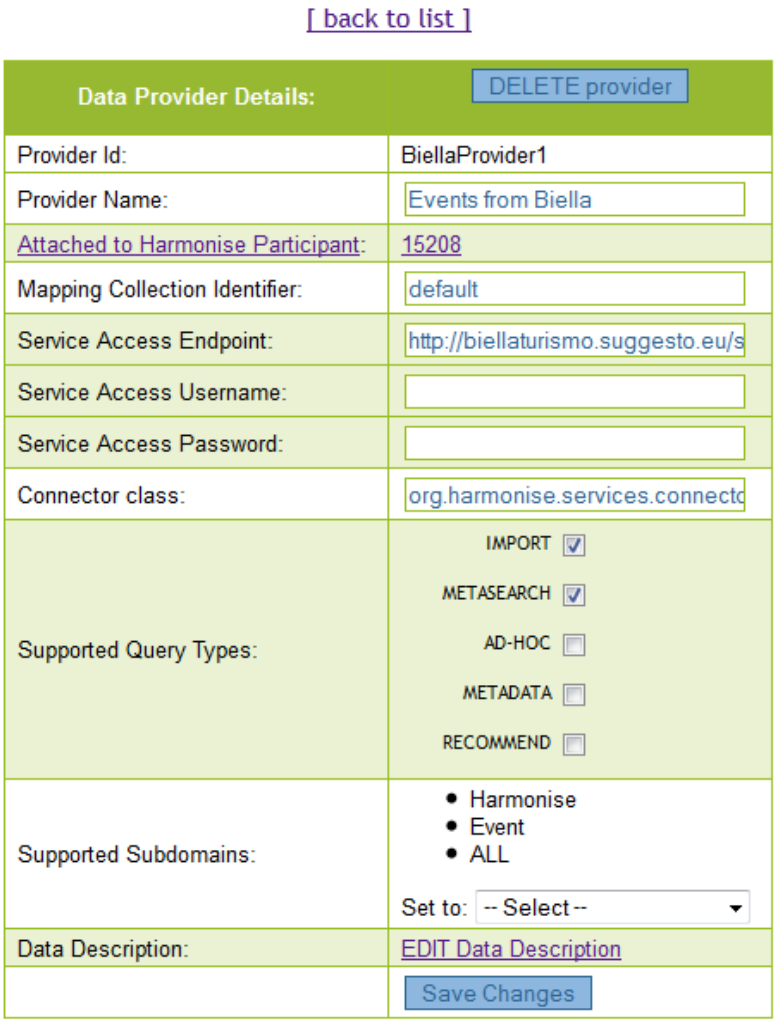
The *back to list* link at the top takes you back to the list of Harmonise participants. It should not be available for users without administrator privileges. In this view you can:

- Delete the Harmonise participant (but not the attached data providers) by clicking the *DELETE Participant* button
- Change the data of the Harmonise participant by changing the corresponding text fields and clicking on the "Save Changes" button. There is no direct feedback that the changes were accepted, but the view always represents the state of the registry. So if after clicking the button the changes values are still shown, then everything should be fine.
- Select an attached data provider for showing the data provider's details
- Create a new data provider in the separate area at the bottom of the portlet. The data provider ID has to be set manually (at the moment) and must be unique. It is advised to combine it with the Harmonise ID in some way. Click on the *Create* button to create the data provider after you entered the ID.

When you select a data provider or create a new one, you are taken to the detailed view for data providers.

Manage Harmonise Data Providers

In order to create a new Harmonise data provider and attach it to a Harmonise participant, you have to follow the steps described above. In order to edit or delete a Harmonise data provider, you have to enter the data provider detail view as described above. This brings you to the following view:



The *back to list* link takes you back to the list of Harmonise participants. It should not be available for users without administrator privileges. In this view you can:

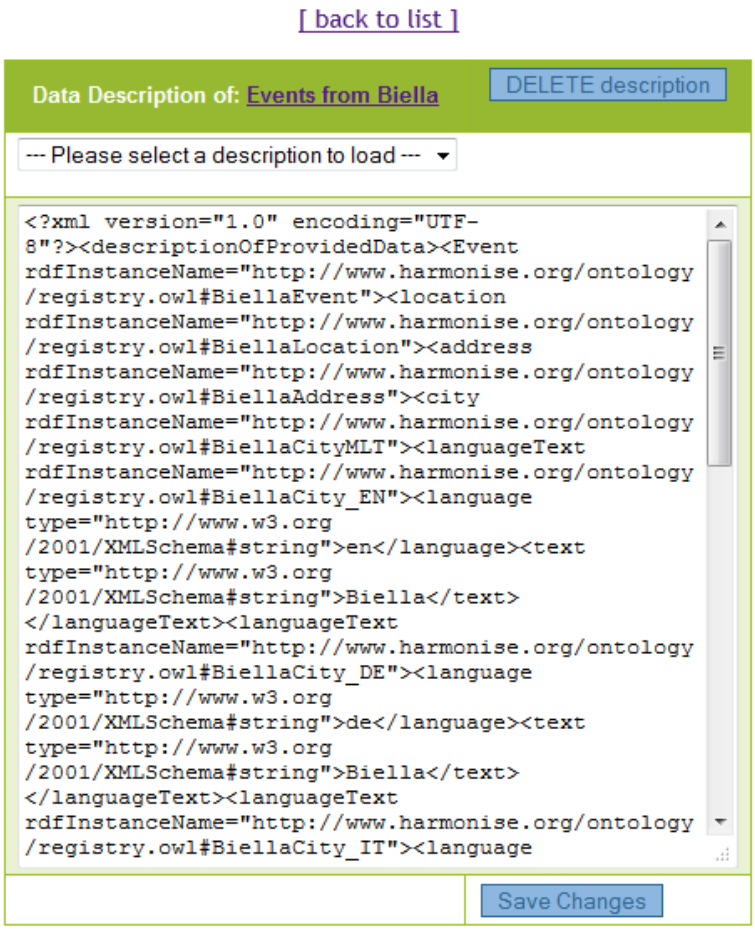
- Delete the Harmonise data provider by clicking the *DELETE Participant* button. Note that this does not delete an attached data data description or the Harmonise participant that the data provider it is attached to.
- Change the data of the Harmonise data provider by changing the corresponding text fields and clicking on the "Save Changes" button. There is no direct feedback that the changes were accepted, but the view always represents the state of the registry. So if after clicking the button the changes values are still shown, then everything should be fine.

- The *Supported Subdomains" field is kind of special. It shows all the subdomains that the reasoner of the semantic registry deducted the actually set subdomain to be compatible with. Ignore the "Harmonise" value here - it is not for actual use.*
- In order to change the supported subdomains, select *one* entry from the select box, e.g., *Event*. Upon saving the changes, this subdomain will be set as the primary one and the reasoner will automatically deduct the compatible ones.

- Select *EDIT Data Description* in order to view and edit the actual description of the provided data.
  - Note that if you created a new Harmonise data provider, then a data description is not automatically generated. You *must* click on the edit link and provide at least a minimal data description for the newly created data provider to be valid.

Managing a Data Provider's Data Description

If you click on *EDIT Data Description*, in the Harmonise data provider view described above, then this will take you to the following view:



This is now the actual description of the provided data in terms of the Harmonise ontology. A previously filled data description will look pretty scary, since a lot of

optional meta data is contained. This is for instance the data type for every data item (*Literal* in RDF) and the RDF node name for every part of the path.

The structure of the XML description of the provided data is this:

- The starting element is *descriptionOfProvidedData*
    - The only child element is one of *Event*, *Accommodation*, *Attraction* and *Gastro*
    - Following the child elements are the elements of a Harmonise data item. The description should only capture those facts which are common to all data items offered by the data provider. Several of the same data elements can be used to express an *or* relationship between the elements on the same level.

For details about the Harmonise XML representation see the XML Schema representation of the Harmonise ontology.

The minimal description only contains the starting element of the respective domain and no further details. For example the minimal description for events looks like this:

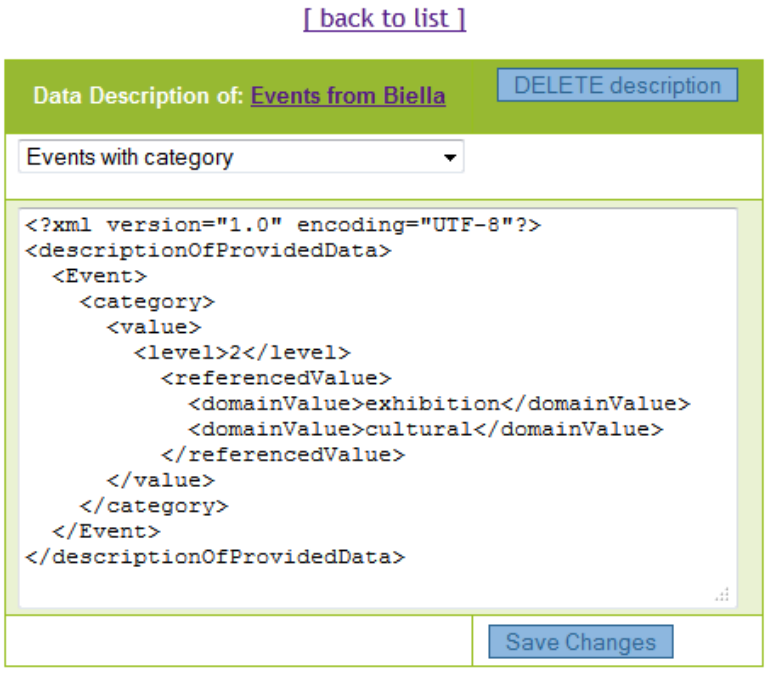
```
<?xml version="1.0" encoding="UTF-8"?>
<descriptionOfProvidedData>
    <Event/>
</descriptionOfProvidedData>
```

Note that there is a special logic to the data description of a Harmonise data provider, which is explained below.

Especially when creating a new data description it is useful to start from a template. There are a handful of different data description templates which can be loaded for the data description by selecting them from the drop down box.

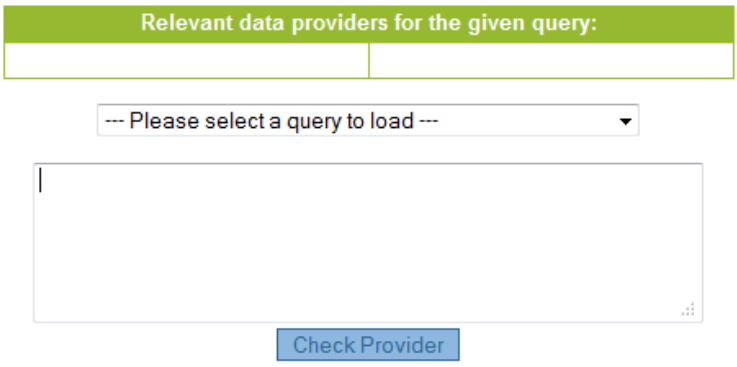
The *back to list* link takes you back to the list of Harmonise participants. It should not be available for users without administrator privileges. The option in this view are:

- Delete the data description by clicking the *DELETE Description* button
- Edit the data description or load a new one by clicking on the select box
- Save changes to the data description by clicking the *Save Changes* button. Again there is no direct feedback that the changes were accepted, but the view always represents the state of the registry. So if after clicking the button the changed description is still shown, then everything should be fine.



Query Test Portlet

This second portlet, named "Query to Provider Test" portlet in the portlet container (e.g., Liferay). It is a simple test application for checking a HarmoSearch (metasearch) query against the data stored in the semantic registry. This is exactly what happens when a metasearch is conducted. The query is analysed and checked against the description of data providers (i.e., data sources) stored in the registry. Those, which are relevant candidates for the given query are returned in order to be queried. Being a candidate for querying means that these data providers (data sources) *possibly* contain data relevant for the query. The matching is done based on the general description of the provider's data. It is entirely possible that queried providers do not actually offer data content corresponding to the query.



Incomplete Knowledge

Note that there is a specific logic behind the reasoning on the data descriptions. The semantic registry is not intended to be an index of data providers' data items,

therefore and for practical reasons we cannot expect all possible data elements to be listed in the data description. Indeed we expect this description to be as short and concise as possible. For example in some specific case it might be perfectly acceptable to have a data provider described only as offering accommodation information in Paris.

The implication of this, however, is that we have to deal with incomplete knowledge in the data description. When dealing with incomplete knowledge, normally one of two logical models is employed.

The first one is the open world assumption, stating that all facts which are not explicitly stated might be true and therefore have to be treated as if they were present. The open world assumption is used in OWL itself. For our data description problem, however, it is not very usable since with the open world assumption we would not actually be able to limit the data. When, as in the previous example, accommodation data is described as being about Paris, then we do not want this description to match a query for accommodation in Pisa. With the open world assumption, this would be the case.

The closed world assumption on the other hand treats all missing knowledge as negative knowledge. In the previous example, there would not be a match for a query for Pisa. The drawback is that the closed world assumption is actually too demanding on the available data for our purpose. For example data could be described to contain information about events in Austria, but not in a specific city since this information changes too often. Then, according to the closed world assumption, all matches with a query asking for events in a specific Austrian city would fail.

For this purpose we implemented a mixture between the open and the closed world assumption for the data description in the HarmoSearch semantic registry. The idea is to treat all data elements where no information is provided in the manner of the open world assumption. In the example with Austrian events without a specified city, the assumption of our logic is that the events can take place in any city in Austria. Therefore queries for events in a given Austrian city would be correctly matched and the query would be sent to the data provider to check whether there really are any events matching the specific query.

On the other hand for all data elements where some information is provided, we treat that information as complete, applying the closed world assumption on this specific information item. In the example above, when the data is described as containing Events in Austria, then we assume that this information is complete and that no events from France, Germany or Italy are available.

This mixture between open and closed world assumption allows us to overcome the problems both singular approaches would pose for our purpose.

Using the query test portlet

Actually using the query test portlet is straightforward. You simply post a HarmoSearch query in the query field at the bottom of the portlet or load one of the predefined queries from the select box. Then click the *Check Provider* button and you will be shown a list of those data providers (i.e., data sources) which are

possibly relevant for the given query. Also some of the access information for these providers is shown.

For information on the HarmoSearch query language see deliverable *D4.1 - Query Language Specification*.

Relevant data providers for the given query:	
Harmonise ID	10835
Full Name	**Euromuse**
Endpoint	http://www.euromuse.net/harmonise/query/query.php
Username	karin
Password	karin
Harmonise ID	15239
Full Name	**Euromuse through Microformat Crawler**
Endpoint	http://localhost:8983/solr
Harmonise ID	15208
Full Name	**Biella**
Endpoint	http://biellaturismo.suggesto.eu/swe/xml/eventi/-/query/list

--- Please select a query to load ---

```
<?xml version="1.0" encoding="UTF-8"?>
<QueryRequest>
    <Priority>normal</Priority>
    <SubDomain>Event</SubDomain>
    <QueryType>METASEARCH</QueryType>
        <Query logicalOperator="AND"></Query>
    <Context>
        <Sender>sender_id</Sender>
    </Context>
</QueryRequest>
```

Check Provider

5.6 CRAWLER TUTORIAL

5.6.1 Crawler

Setting sites to crawl

The websites that should be crawled have to be specified such that the crawler knows from which "root" sites to start, and which (external) links from these pages to follow. Therefore, the following files have to be modified and adjusted if new sites should be crawled or if the crawling of some sites should be avoided in the future:

- server/nutch/conf/crawl-urlfilter.txt
    - Line 40: replace MY.DOMAIN.COM with first urls to crawl.
    - Add one line (with same formatting) for each url.
- server/nutch/urls/nutch
    - Add urls to be crawled, one per line.
- server/nutch/conf/regex-urlfilter.xml

- Modify regular expressions to not exclude your sites (e.g. queries, subsites whose urls might contain a certain number of slashes, ...), if needed.

Starting the crawling, parsing and indexing processes

Nutch can only be startet from a Unix shell, or by using Cygwin in Windows. In such a terminal, the following steps are required for the crawling process:

- Move to the folder server/nutch.
- Start crawling: bin/nutch crawl urls -dir crawl -depth 2
  - -dir defines the directory to put the crawled data.
  - -depth defines the depth of links to follow, starting from the root page.
  - -threads defines the number of threads fetched in parallel (optional).
  - -topN defines the maximal number of retrieved pages at each level of depth (optional).

Before indexing, SOLR has to be started: This can be done from any terminal, independent on the operating system used, by calling inside the folder server/solr/example: java -jar start.jar. During indexing, SOLR has to be running.

Index the crawled pages (now SOLR has to be running):

- Move to the folder server/nutch.
- Start indexing: bin/nutch solrindex [http://HOST_ADDRESS:8983/solr/] crawl/crawldb crawl/linkdb crawl/segments/*
  - Exchange HOST_ADDRESS by the one which is used, e.g. localhost, or solr.harmonet.org.
  - Set the port number correctly, if another one is used (can be looked up in the SOLR terminal).

Deleting crawled data and indices

To delete the crawled data from Nutch, the content of the folder server/nutch/crawl has to be deleted. Alternatively, the whole directory can be deleted. In both of these cases, crawling can be done from scratch, which might be of interest during testing.

When indexing should be redone, then also the current indices have to be deleted. This is done by deleting the *directory* server/solr/example/solr/data/index, not only its content.

Note that only if both, the crawled data and the folder containing the current SOLR database, are deleted, the crawling-parsing-indexing process is repeated entirely.

Test cases

Crawling and parsing tagged events from web pages

Add a web site containing microformat vevent tags, e.g. [http://sabineschneider.it/harmosearch/,] to server/nutch/urls/nutch and (the same web page) as regular expression to server/nutch/conf/crawl-urlfilter.txt. Then start

the crawling process using a Unix terminal (i.e. Cygwin for Windows users, or any desired console for Linux or other Unix users), starting from the root directory of Nutch, i.e. server/nutch, by calling bin/nutch crawl urls -dir crawl -depth 2.

Expected result: The events marked as vevents on the specified pages and all pages which are linked to from the given root site up to the chosen depth (in the above example 2) are parsed and listed in a proper format in the directory server/nutch/crawl.

Indexing parsed events and populating SOLR database

Having crawled some events, start SOLR by moving to the directory server/solr/example and by then calling (from any terminal) java -jar start.jar. As soon as the database is running, call the indexer from server/nutch by typing again in a Unix terminal bin/nutch solrindex [http://solr.harmonet.org:8983/solr/] crawl/crawldb crawl/linkdb crawl/segments/*.

Expected result: The indexed events and all the tagged and parsed information fields are contained in the SOLR database as distinct items, from which they can be read out using queries.

5.6.2 Search Engine

Access the crawled data

(Browser access to SOLR)

In oder to search the index, there are various possibilities:

- The SOLR admin panel including a request form is located at [http://HOST_ADDRESS:8983/solr/admin.]
    - The HOST_ADDRESS has to be adjusted, e.g. to localhost or solr.harmonet.org.
    - Also the port number should be corrected, if SOLR is running on another port.
    - Type the query, i.e. FIELD_NAME:FIELD_VALUE in the field "Query String" of the admin panel, where the field value is not case sensitive, and a wild card is denoted by a *. Thus, if all entries should be returned, the query should look like this: *:*.
- Search the XML directly, which is located at the URL one can see in the address bar when getting the results or a request from the admin panel: [http://HOST_ADDRESS:8983/solr/select/?q=FIELD_NAME%3AFIELD_VALUE]
    - The HOST_ADDRESS has to be adjusted, e.g. to localhost or solr.harmonet.org.
    - Also the port number should be corrected, if SOLR is running on another port.
    - FIELD_NAME is the name of the field whose value is specified, or * to indicate all fields.

- FIELD_VALUE is the value that should be contained in the specified field, in order to fulfill the query. Note that the field value is not case-sensitive, and again a * denotes a wild card.

Test cases

Querying populated SOLR database to obtain event information

After having crawled and indexed some events (see above test cases), you can start querying them.You can choose between two possibilities:

- Go to the SOLR admin panel, located e.g. at [http://solr.harmonet.org:8983/solr/admin/,] and type the query in the second text field (Query String):
  - *:* to obtain *all* entries of the SOLR database.
  - summary:meeting to obtain all entries whose summary field contains the word meeting, where the query is not case-sensitive.
- As an alternative to the SOLR admin panel, the query can directly be sent as a url, using:
  - [http://solr.harmonet.org:8983/solr/select/?q=*%3A*] to get as result *all* database entries.
  - [http://solr.harmonet.org:8983/solr/select/?q=summary%3Ameeting] to get all entries as result, where the (not case-sensitive) word meeting is contained in the summary field.

Expected result: An XML formatted document is returned, containing as response to the given query various result documents, one for each event. Each such document contains tags for all event fields that were tagged, parsed and indexed. In the first case, all events contained in the database will be listed, in the second sample case only the events whose summary field contains meeting will be given as results.