D2.4.5 SERVICE DEPLOYMENT AND EXPERIMENTATION IN EUROPEANALABS

Advanced Search Services and Enhanced Technological Solutions for the European Digital Library

Grant Agreement Number: 250527

Funding schema: **Best Practice Network**

Deliverable D2.4.5 WP2.4

<Report>
V.1.0– 10 May 2011
Document. ref.: ASSETS.D2.4.5.CNR.WP2.4.V1.0

Programme Name: ..... ICT PSP
Project Number: 250527
Project Title:..... ASSETS
Partners:..... Coordinator: ENG (IT)
Contractors:
Document Number: D2.4.5
Work-Package:..... WP2.4
Deliverable Type: ..... Report
Contractual Date of Delivery: M12
Actual Date of Delivery: ..... May 10 2011
Title of Document: ..... Service Deployment and Experimentation in Europeanalabs
Author(s): ..... CNR
Approval of this report ..... Approved

Summary of this report: ..... see Executive Summary

History:..... see Change History

Keyword List: ..... ASSETS, Models, Services, Interfaces

Availability ..... This report is:
X public
limited to ASSETS consortium distribution
limited to EU Programme distribution
restricted

Change History

Version	Date	Status	Author	Description
0.5	March 28, 2011	Draft	CNR	Initial version
0.6	April 13, 2011	Draft	CNR	Pre-review version
0.7	April 15, 2011	Draft	CNR	Reviewer version
0.8	April 26,2011	Draft	CNR	Reviwed by ENG
1.0	May 10, 2011	Final	CNR	Final Delivery

Table of Contents

Executive Summary

The ASSETS Project is going to develop a number of services that will be integrated in the Europeana services infrastructure. This deliverable provides a description of the ASSETS server components and a detailed description of the deployment and installation of the ASSETS software in the ASSETS platform.

1. Introduction

The ASSETS Project is going to develop a number of services that will be integrated in two different platforms: the Europeana and the ASSETS platforms. The Europeana platform includes all the services supporting the Europeana portal and API. In comparison to this, the ASSETS platform includes the services developed by ASSETS, and a part of the Europeana modules which are strictly needed for the development of ASSETS services. The ASSETS platform, on the other hand, is expected to advance the Europeana platform with respect to the services offered on multimedia content, which are at the moment outside of the scope of core Europeana services.

At the time of the preparation of the ASSETS Description of Work, integration of the ASSETS services into the Europeana platform was given a high priority. To this end, the following scenario is envisaged in the Description of Work:

1. At the start of the project, a sandbox would be created on the EuropeanaLabs. The sandbox is a virtual machine that replicates the architecture of the Europeana platform at the start of the ASSETS project, including its database and index. This sandbox is referred to in the ASSETS Description of Work as "the ASSETS sandbox".
2. Any new service developed by ASSETS would be deployed as a stand-alone service on the ASSETS sandbox, to be tested against the Europeana database, until the satisfactory quality level would be reached.
3. At this point, the service would be integrated in the Europeana architecture on the ASSETS sandbox.
4. Any time after the successful completion of the previous step, the Europeana architects could deploy the service on the real Europeana platform. This deployment would require a minimum amount of resources, because the technicalities of integration had been already tested (step 3) on a replica of the Europeana platform.

In reality, the envisaged procedure turned out to be difficult to realize for a number of reasons, mostly related to the fact that the Europeana architecture, database and index are an evolving set. In order to keep the Europeana platform aligned with the ASSETS sandbox, a large amount of resources from the Europeana Development Team would have been required, in a continuous manner.

The ASSETS Infrastructure Team, designed then an alternative, but similar approach. This approach gives priority to integration of the services on the ASSETS platform, and is structured in the following way:

1. At the start of the project, the ASSETS platform will be created on the ASSETS servers at ATC. This platform replicates the basics of the Europeana architecture, and includes a snapshot of the Europeana database and index.
2. Any new service developed by ASSETS would be deployed as a stand-alone service on the ASSETS platform, to be tested against the snapshot of the Europeana database, until the satisfactory quality level would be reached.
3. At this point, the service would be exposed on the ThoughtLab of Europeana.
4. Any time after the successful completion of the previous step, the Europeana

architects could deploy the service on the real Europeana architecture.

This procedure moves the effort of the Europeana Development Team to Step 4, which is clearly heavier than the analogous step of the previous procedure. Anyway, this process reduces to minimum, the effort required from the Europeana side during the service development. It is felt that, by following the appropriate guidelines in the development of services, the alternative approach will reduce the dependecies between ASSETS and Europeana development efforts, making this process to be more suitable to the agenda of the Europeana.

The present deliverable was planned to outline the guidelines that the ASSETS software architects followed in executing step 1 of the procedure described above.

2. Service deployment and experimentation in Europeanalabs

The guidelines cover the main aspects of the system deployment process, and include:

- Description of the ASSETS software architecture
- Description of the ASSETS platform features
- Instructions of ASSETS server deployment

2.1 Software architecture

The goal of this section is to describe the system architecture of ASSETS software.

2.1.1 Overview

The proposed ASSETS architecture is sketched in the following figure:

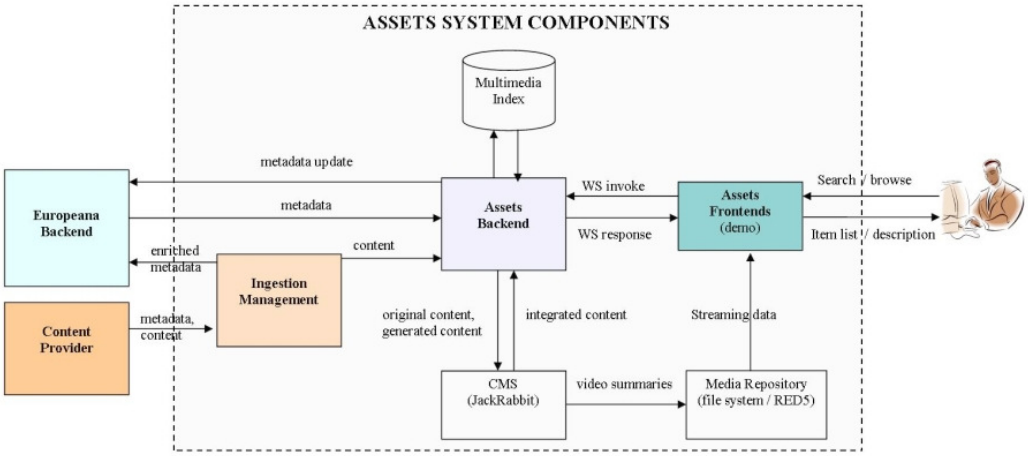

Figure 1 – ASSETS Architecture

The dashed line marks the border of the ASSETS system; the named arrows represent the dataflows exchanged between the ASSETS system components (internally or with the outer world).

The ASSETS internal architecture is composed by three main modules.

- The Ingestion Management module has the role of collecting the metadata information and the content from the content providers and submitting it for storage into the ASSETS&Europeana.
- The ASSETS Frontend implements the Graphical User Interface which offers a rich set of browsing and searching functionality for end users.
- The ASSETS Backend module implements the business functionalities and makes it available on Internet through a Web API.

2.1.2 *ASSETS software architecture*

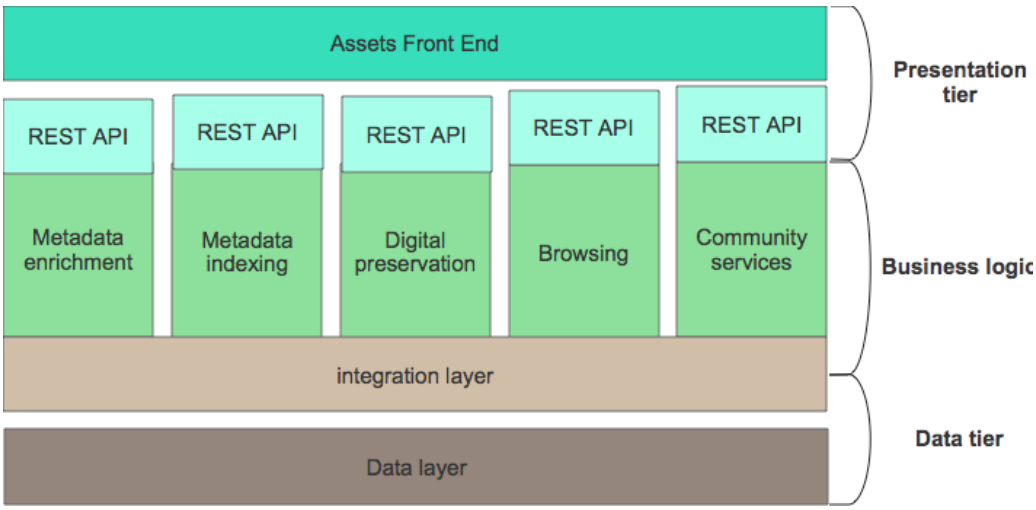

Figure 2 – ASSETS Three-tier Architecture

Considering the "three-tier architecture" paradigm typically adopted to describe information management systems, the ASSETS system can be described as follows:

Presentation tier: the ASSETS system will provide RESTful APIs enabling external applications to use its functionalities, a set of GUIs for managing specific functionalities such as data ingestion and data access and browsing and web GUI (ASSETS portal) for browsing services and functionalities.

Business logic: a set of autonomous software components implementing the following functionalities:

- Metadata Enrichment, Heterogeneity Reduction, Information Extraction and Classification;
- Indexing, Ranking and Retrieval for improved results;
- Digital Preservation for planning and guaranteeing long-term access and usability of content;
- Browsing and Content Characterisation of multimedia objects;
- Community for supporting user generated content, tagging, etc.

It also implements a middleware offering integration functionalities and enabling built-in interaction between components.

Data tier: enables software components to interact in a transparent way with data repositories. The ASSETS system will need to manage different kind of information objects: multimedia objects, structured data, text, indexes, etc. For this reason there will be several different kinds of data sources.

The main goal in adopting the described architecture is to obtain a loosely coupled system: each software component implementing the business logic has no (or little) knowledge of other components and also is not supposed to know which tools are actually used to store or manage data.

From the deployment point of view this means that each components can be packaged including the set of functionalities it needs to interact with the data layer and deployed separately as a standalone component. For instance the component implementing metadata enrichment services, when completed, will be embedded in the Europeana Unified Ingestion Manager (UIM). To do this we have designed a special integration component implementing:

- functionalities that enable ASSETS services to interact each others;
- functionalities implementing interactions with the data layer;
- functionalities shared between service components

More information regarding the design of the individual ASSETS services is presented in the deliverable D2.0.4 The ASSETS APIs.

In the following paragraphs we'll present it outlining the technical requirements for building and deploying the runtime infrastructure.

2.2 The ASSETS experimentation platform

The EuropeanaLabs is a server infrastructure hosted at ISTI providing virtual machines, also called Europeana sandboxes. Each sandbox offers a controlled set of resources and is accessible via the network.

Europeana sandboxes may be used for testing purposes or to host application servers used by the Europeana community (e.g collaborative servers, project management tools etc).

Currently it is not possible to replicate the architecture of the Europeana platform in a sandbox because the Europeana architecture, databases and indexes are an evolving set and in order to keep the Europeana platform aligned in a sandbox, a large amount of resources from the Europeana Development Team would be required, in a continuous manner.

This is a problem for ASSETS, since most of the implemented functionalities are based on Europeana services.

Therefore the ASSETS Infrastructure Team, as described above, adopted an alternative approach where an ASSETS experimentation platform is created on the ASSETS servers at ATC and this platform replicates the core of the Europeana architecture including a snapshot of the Europeana databases and indexes.

2.2.1 The ASSETS platform solution stack

The solution stack of software adopted to implement the ASSETS experimentation platform includes:

- OS: 2.6.32-22-server #36-Ubuntu SMP UTC 2010 x86_64 GNU/Linux
- Postgresql 8.4
- mongoDB 1.8.0
- Apache Jackrabbit 2.2.4
- Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
- Java HotSpot(TM) 64-Bit Server VM (build 16.3-b01, mixed mode)

- apache-tomcat-6.0.18
- apache-solr 1.4

The server running this stack has 4 GB RAM and the disk size is 200 GB.

2.2.2 Configuring the ASSETS experimentation platform

In order to install the ASSETS server in the experimentation platform a number of actions are required to configure the runtime infrastructure.

2.2.3 Download and install Apache Tomcat

The Apache Tomcat [7] is used to deploy and manage the Web Applications implementing the ASSETS services.

It is assumed that Apache Tomcat is installed and located in a folder that will be named "CATALINA_HOME" in the rest of this document. In the experimentation platform we have installed the release 6.0.18 of Apache Tomcat. The increase of the Tomcat JVM memory to 1 GB is recommended.

The following commands will update the initial and maximum heap size, and need to be run before starting your tomcat:

```
export CATALINA_OPTS="-Xms512m -Xmx1024m"
```

or

```
export JAVA_OPTS="-Xms512m -Xmx1024m"
```

They create an environment variable called CATALINA_OPTS or JAVA_OPTS contains the required options to make tomcat start heap size 512M and maximum heap size 1024M.

2.2.4 Download and Install Apache SOLR

The Apache SOLR engine [3] is used to index the content stored in ASSETS.

1. Download the latest release of Apache SOLR Search Engine. Rename the web archive file to "solr.war" and place the war file in the webapps directory:

```
$CATALINA_HOME/webapps
```

2. After the solr.war deployment, change the configuration file (web.xml), by setting the following parameter:

```
$CATALINA_HOME/webapps/solr/WEB-INF/web.xml
<env-entry>
    <env-entry-name>solr/home</env-entry-name>
    <env-entry-value>SOLR_HOME </env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

SOLR_HOME is the directory containing solr configuration files.

3. Modify $SOLR_HOME/conf/solrconfig.xml by setting the following attribute:

```
<!-- Used to specify an alternate directory to hold all index
data other than the default ./data under the Solr home.
If replication is in use, this should match the replication
configuration. -->
<dataDir>INDEX_HOME</dataDir>
```

Where INDEX_HOME is the directory of the ASSETS index, e.g.:

```
<dataDir>/home/solr_dump_full/export/solr/data/</dataDir>
```

2.2.5 Download and Install Jackrabbit

The Apache Jackrabbit [2] is a content repository manager and it is used to store files, mainly media files, in the ASSETS.

Download the latest release of Jackrabbit (standalone). Run the jackrabbit.jar with the following command:

```
java -jar jackrabbit.jar -p 8090
```

If the server starts correctly, you will see the following message:

```
Welcome to Apache Jackrabbit!
-------------------------------
Using repository directory jackrabbit
Writing log messages to jackrabbit/log
Starting the server...
Apache Jackrabbit is now running at http://localhost:8090/
```

2.2.6 Install Database Management Systems

Two DBMSs are required in the Experimentation Platform: mongoDB [6] and PostgreSQL [5]. Install both following the instructions in the documentation.

While the mongoDB simply needs to be started, a database must be manually created in PostgreSQL in order to run the Europeana portal locally, (the following instructions, provided by Europeana, assume that the command line interface is used):

1. create the 'europeana' user:

```
createuser –U postgres –P –D –R europeana
```

2. create 'europeana' database and enter the password

```
createdb –E utf–8 –O europeana –U postgres europeana
```

3. if you get the error: "database creation failed: ERROR: new encoding (UTF8) is incompatible with the encoding of the template database (SQL_ASCII)". In that case add the template parameter:

```
createdb –E utf–8 –O europeana –U postgres europeana –T
template0
```

4. test if you can login

```
psql europeana europeana
```

Note that the database name and the credentials of the user accessing it must be reported in the Europeana property file (see below).

Both DBMS should be up and running before activating the ASSETS server.

2.2.7 Install the ASSETS war files

The ASSETS services are implemented by a set of autonomous components interacting with each other and exchanging information through the common data layer. An integration software component exposes the functionalities of the via REST APIs. In order to install a service the core ASSETS server should be in place.

1. To install the ASSETS core (and a set of services), download the zip archive from the ASSETS developer server [1]. The archive contains the following files:
   I. ASSETS#common.war (ASSETS integration layer)
   II. portal.war (ASSETS front end portal)
   III. api.war (Europeana API)
   IV. ASSETS#ir-image.war
   V. ASSETS#ir-video.war
   VI. ASSETS#ir-text.war
   VII. europeana.properties.template (configuration file)

VIII. README.txt (installation instructions).

2. Rename the europeana.properties.template file as europeana.properties and copy it in a directory which will be reference in the future as PROPERTIES_HOME.

3. Change java opts in Tomcat by setting –Deuropeana.properties as below:

```
JAVA_OPTS="$JAVA_OPTS -Deuropeana.properties=PROPERTIES_HOME/europeana.properties"
```

4. Customise europeana.properties following the instructions in the document.

5. Copy the web archive (war) files in the $CATALINA_HOME/webapps/ directory, and deploy them.

Please note that currently the integration layer war contains also a wrapper of the Europeana backend in order to replicate the basics of Europeana Architecture as explained above.

2.2.8 Move static resources to server

Based on the following attributes set in europeana.properties file, for example:

```
static.page.path = /home/europeana_resources/staticpages
message.resource=file:/home/message_keys/messages
log4j.xml=/home/log4j.xml
```

Store the related bundles inside the server ("staticpages", "messages").

2.2.9 Create 'admin' user

1. Access the europeana database:

```
sudo –u postgres psql europeana
```

2. Insert a new record in the users table:

```
INSERT INTO users(
            id, email, enabled, firstname, languages,
lastlogin, lastname, newsletter, "password", projectid,
```

```
providerid, registrationdate,"role", username)
     VALUES (1, '[EMAIL]', true, null , null , null , null ,
false,'3685e330ec5277a9dd5661c61f2bc55811f5a628'  ,null  ,null,
null , 'ROLE_ADMINISTRATOR' , 'admin' );
```

The [EMAIL] value is an existing email account address accessible by the user.

For security purposes it is necessary to change the password. This is feasible by means of clicking the "I forgot my password" link on the login form. The admin user may modify the user privileges through the "User Administration" page accessible from the footer menu.

2.2.10 Configure Logging

Based on the log4j [4] attribute set in europeana.properties file, for example:

```
log4j.xml=/home/log4j.xml
```

Place and configure the log4j configuration file in the related directory inside the server.

2.2.11 The ASSETS services for the Europeana ingestion

As mentioned above, there are a number of ASSETS services that will be used in the ingestion process of Europeana (mainly metadata enrichment services and digital preservation services). The Europeana ingestion workflow is currently implemented by a set of separate software tools integrated at “data level” by sharing data stores.

To improve efficiency and reliability of the data ingestion phase, the ASSETS, Europeana & Tel is working on reimplementing the ingestion workflow module: Unified Ingestion Manager (UIM). This module is built by using the OSGi Architecture [8] and the Apache Karaf framework implementaion [9]. The ASSETS services which will be integrated into the UIM tool are OSGI-aware, i.e. they implement the required interfaces for being deployed as OSGI Plugins. The first release of the ingestion services will be available in the second year of the project. The installation instructions for the OSGI container will be added to this document at that time.

3. Conclusions

This deliverable describes the technical solutions adopted to implement and run the ASSETS software and the steps needed to install the ASSETS experimentation platform.

The first part of the document lists the planned steps for the integration of ASSETS services in Europeana platform and describes the reasons for this planning. Inparticular it explains the need of having the core Europeana services installed in the ASSETS experimentation platform in order to not be concerned of changes made to Europeana platform.

The sections 2.1.1 and 2.1.2 show that the ASSETS platform has been designed to be a loosely coupled software system where services interact each others and with the data layer using an integration layer and expose their functionalities on the Web using Web services protocols. This description is used to introduce the ASSETS runtime platform wich is composed by one servlet container implementing web services functionalities and two DBMSs and one CMS for the data storage layer; in order to embed also the Europeana core search functionalities also a Search engine tools is included in the ASSETS platform.

The tools and servers of the Exprimentation platforms are described in details in the last part of the document, for every component the installation and configuration instructions are reported.

References

1. ASSETS core server: http://www.europeanalabs.eu/svn/ASSETS/builds/
2. Apache Jackrabbit: http://jackrabbit.apache.org/
3. Apache SOLR: http://lucene.apache.org/solr/
4. Apache Log4j: http://logging.apache.org/log4j/index.html
5. PostgreSQL: http://www.postgresql.org/
6. mongoDB: http://www.mongodb.org/
7. Apache Tomcat: http://tomcat.apache.org/
8. OSGi: http://www.osgi.org/About/Technology
9. Apache Karaf: http://karaf.apache.org/