D5.6.1 – Multimedia Annotation Service – Documentation and Final Prototype

This deliverable is software.

co-funded by the European Union

The project is co-funded by the European Union, through the ***e*Content*plus*** programme

http://ec.europa.eu/econtentplus

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

ECP-2008-DILI-528001

EuropeanaConnect

D 5.6.1 – Multimedia Annotation Service Documentation and Final Prototype

Deliverable number/name	*D 5.6.1 Multimedia Annotation Service – Documentation and Final Prototype*
Dissemination level	*Public*
Delivery date	*18.04.2011*
Status	*v 1.0*
Author(s)	*Rainer Simon*

e*Content*plus

This project is funded under the *e*Content*plus* programme,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

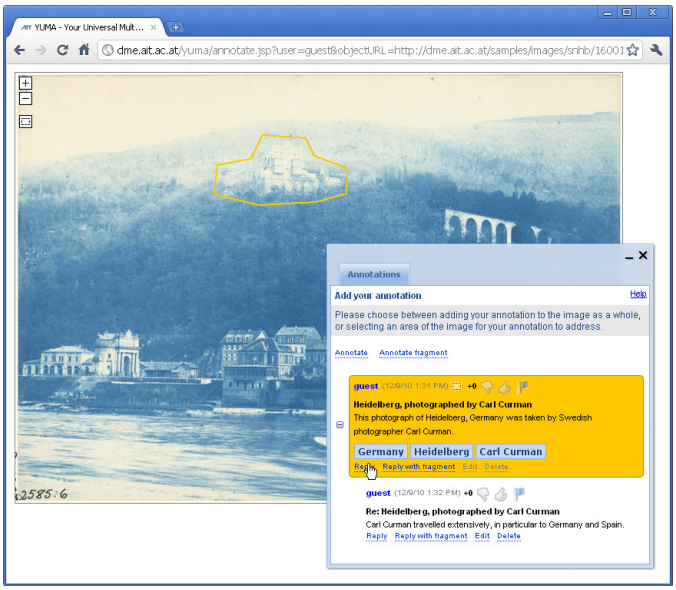
Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

Introduction

Annotations are a fundamental scholarly practice common across disciplines. They enable scholars to organize, share and exchange knowledge, and collaborate in the analysis of source material. At the same time, annotations offer additional context to the original items and their metadata. Annotations provide explanations which may help other users in the understanding of a particular item, or point them to related material which may be useful for its interpretation.

As cultural institutions are making increasing efforts to digitize their holdings and to make them available to the public over the Web, the role of annotations is evolving. Institutions are discovering the added value of user-contributed knowledge: to institutions, annotations can serve as a source of additional metadata which can improve search and retrieval, and thus help users to discover content they wouldn't have found otherwise. To users, adding comments, notes or tags to collection items is a convenient way to organize and personalize the information they find, or to share it with others online.



Until now, however, no widely adopted annotation application that can manage more than one specific media type exists. If annotation functionality is provided at all, it is usually based on an in-house solution, employing proprietary data models which are not interoperable with those of other systems. As a result, annotation data is locked in closed data silos, and usable only within a single institution.

The YUMA Universal Media Annotation Framework

The *YUMA Universal Media Annotator* (**YUMA**) provides a browser-based annotation framework for different types of digital media content. With YUMA, users create “Post-It”-style free-text annotations that can pertain to the digital item as a whole, or to a part of it. Additionally, an innovative semi-automatic *Semantic Tagging* approach supports users in augmenting their annotations with structured context information that can be used to enrich and complement traditional collection metadata. To address the issue of annotation interoperability, YUMA publishes annotations according to the principles of Linked Open Data (LOD). This way, annotations are pulled out of institutional data silos and become part of the Web, where they can be processed, re-used, and linked to by other services and systems.

YUMA is entirely browser based and supports annotation of *images*, *digitised maps* and, through prototype implementations, *audio* and *video*. Basic features include:

- Creating annotations with *public* or *private* visibility
- Replying to annotations

- Shape drawing tools for images, maps and video (see Fig. 1)
- Semantic tagging, supported by the “Context Tag Cloud”
- RSS feeds for following discussion around a particular digital media item or specific annotation; or for following a particular user
- Special support for high-resolution map images:
  - Tile-based rendering for faster delivery
  - Collaborative geo-referencing
  - Semantic tag suggestions based on geographic location
  - Overlay of present-day country borders, coast outlines, etc.

To showcase a potential future integration scenario with Europeana, YUMA is currently available in the Europeana ThoughtLab[1].

Architecture Overview

YUMA is based on a distributed architecture that consists of two core elements, which will be described in detail in the following sections:

- The **YUMA Server**, a common “backend” service component for all content types which provides annotation storage, management and search as well as flexible publishing and export functionality.
- The **YUMA Suite**, a set of browser-based end-user applications for annotating content of the different media types.

As YUMA is designed for integration into a host environment such as the Europeana portal, it lacks typical “portal” functionality such as user management. Instead, APIs and authentication mechanisms are foreseen, which allow the host environment to integrate YUMA as an external, loosely-coupled service.

[1] http://www.europeana.eu/portal/thoughtlab_usergeneratedcontent.html

The YUMA Server

The YUMA Server is the storage and management back-end of the YUMA Framework. It can be deployed with a relational database such as MySQL or PostgreSQL or, alternatively, with a MongoDB NoSQL database. Please note, however, that at the time of writing MongoDB support is experimental and available with limited functionality only.

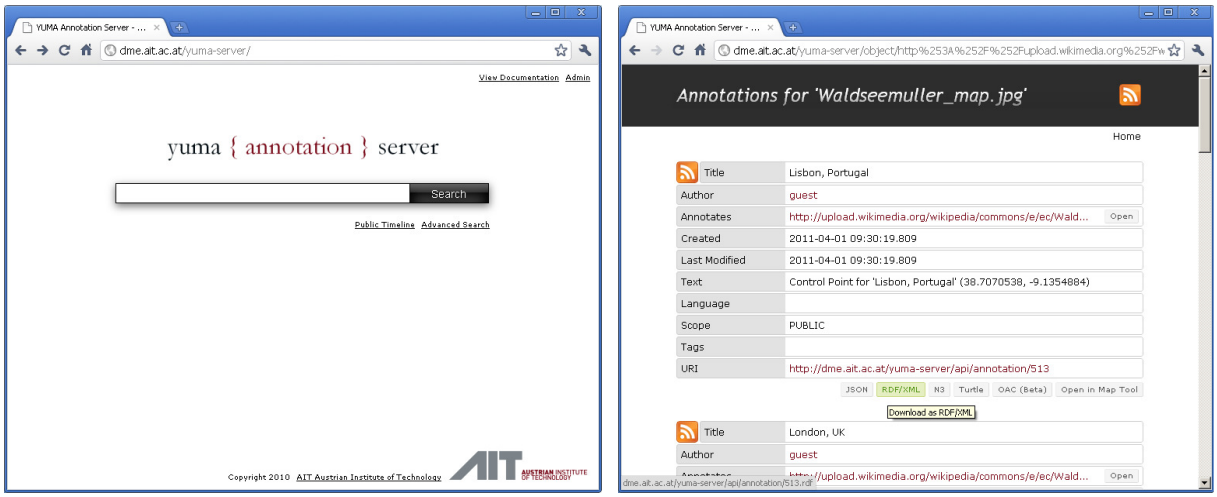


Fig. 2. YUMA Server GUI: search (left), list of annotations for a media object (right)

The YUMA Server features a user interface for search and for basic administration tasks (see Fig. 2). Since YUMA is designed for integration with a host environment, the user interface has actually “maintenance purpose”. As its primary interface to the outside world, the Server provides a range of APIs:

- A **JSON REST API** for creating, retrieving, updating and deleting annotations
- Annotation **search** via the OpenSearch RSS protocol
- A **Linked Data** interface that exposes annotations as RDF, according to the OAC model and the (legacy) LEMO annotation vocabulary. OAC is an emerging ontology for describing scholarly annotations of Web-accessible information resources; and YUMA is among the first annotation solutions to implement it. The interface provides RDF/XML, N3 and TURTLE output, and supports content negotiation.
- **RSS feeds** on objects and users, reply feeds on annotations

Server Requirements and Deployment

The YUMA Server is delivered as a Java Web application archive (.war) file. For local deployment, the YUMA Server requires Java 1.6, and a Java Servlet container that implements the Servlet 2.4 specification such as Tomcat version 5.5 (or higher). Furthermore, the YUMA

Server requires access to a supported relational database base such as MySQL or PostgreSQL[2]. To deploy the YUMA Server, follow these steps:

- Deploy the .war in your Servlet container
- Open the `web.xml` file, located in the /WEB-INF folder
- Adapt the configuration in the `web.xml` file. Modify the following parameters according to your environment:
  - `server.base.url` – the externally visible base URL of your YUMA Server
  - `suite.base.url` – the externally visible URL of a YUMA Suite installation
  - `admin.username` and `admin.password` – username and password for the administration dashboard
  - `configuration` – deployment configuration. Make sure this parameter is set to 'deployment' when in production!
- Ensure that all database configuration parameters are correct. **Important:** you may need to add the appropriate JDBC driver manually. The driver .jar must be placed in the /WEB-INF/lib directory
- Create a database with the same name as configured in `annotation.db.name`
- You may need to restart your Servlet container

API Details

After successful deployment, the API of a YUMA Server installation is available under the following root path:

http://<yourdomain>/yuma-server/api/

The various API endpoints are found at the following URLs:

http://<yourdomain>/yuma-server/api/annotation

HTTP POST to this URL will create a new annotation. Supported payload formats are JSON and (with limited functionality) RDF/XML. In case of success, the server will respond with HTTP 201 (Created) code, the URI of the annotation in the Location header, and the ID that was assigned to the annotation by the system in the response body.

http://<yourdomain>/yuma-server/api/annotation/{id}

HTTP GET to this URL will retrieve the annotation with the specified ID. The server will respond with HTTP 200 (OK) and the serialized annotation in the payload. Supported response formats are JSON and RDF (XML, N3, TURTLE). The format decision is based on the MIME type specified in the HTTP Accept header (content negotiation). Alternatively, the return type can be forced by appending an extension. Supported extensions are .json, .rdf, .n3 and .turtle.

[2] A list of supported databases can be found here: http://community.jboss.org/wiki/SupportedDatabases

HTTP PUT to this URL will update the annotation with the specified ID. If successful, the server will respond with HTTP 200 (OK), the URI of the annotation in the Location header, and the (new) ID assigned to the annotation by the system in the response body.

HTTP DELETE to this URL will delete the annotation with the specified ID. The server will respond with HTTP 204 (No Content) in case of success.

http://<yourdomain>/yuma-server/api/tree/{objectUri}

HTTP GET to this URL will retrieve the entire tree of annotations that exist for an annotated multimedia object, specified by its (URL-encoded) URI. The server will respond with HTTP 200 (OK) and the serialized tree of annotations in the payload. Supported response formats are JSON and RDF (XML, N3, TURTLE). The format decision is based on the MIME type specified in the HTTP Accept header (content negotiation). Alternatively, the return type can be forced by appending an extension. Supported extensions are .json, .rdf, .n3 and .turtle.

http://<yourdomain>/yuma-server/api/search?q=<query>&start=<start>&count=<count>

This URL provides a search API. Responses are returned in OpenSearch RSS format. The `count` and `start` parameters specify the total amount of search results to return, and the offset of the first returned result in the global list of results, respectively. Both parameters are optional.

The YUMA Annotation Model

Internally, YUMA represents annotations as hierarchical data structures, with only a few mandatory properties. Fig. 3 shows a diagram of the YUMA Annotation Class Model. (Green boxes represent actual classes. Blue boxes represent enumerations, i.e. the range of possible values for the Scope and Media Type fields, respectively.) The upper properties - above the dotted line - are mandatory properties. Properties below the dotted line are optional.

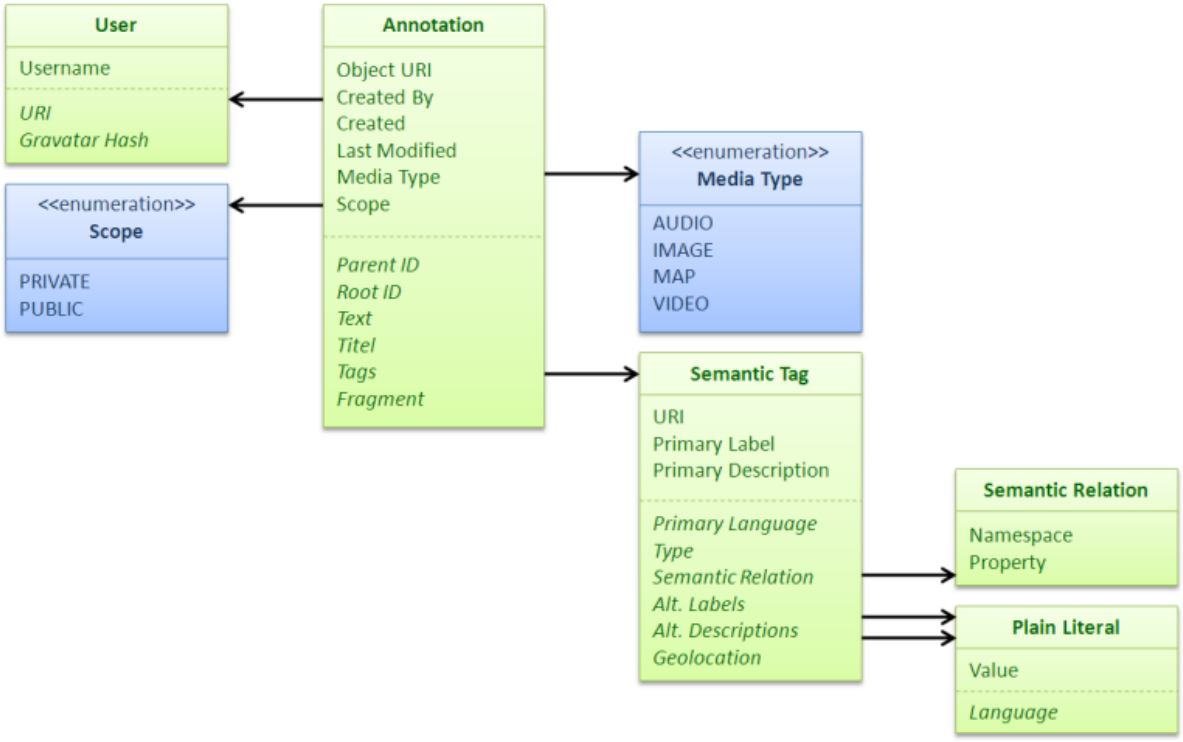


Fig. 3. YUMA annotation class model

Most of the properties (such as `Object URI`, `Created By`, `Created`, `Last Modified`, `Media Type`, `Scope`, `Text` or `Title`) should be self explanatory.

The `Parent ID` and `Root ID` properties are only needed if the annotation is a reply to another annotation. In this case, the `Parent ID` is the ID of the immediate parent annotation, and the `Root ID` is the top-most annotation in the reply thread.

The `Fragment` property contains the description of the media fragment (e.g. image region, audio time interval, etc.) to which the annotation pertains. (If the annotation pertains to the entire media item, the `Fragment` is empty.) In the current implementation, YUMA does not make any assumptions about the syntax of the fragment identifier. The property is simply a String that can take any arbitrary value. It is the responsibility of the client (i.e. map or audio) to make sense of the fragment information.

The `Tags` property contains a list of Semantic Tags which form part of the annotation. A detailed description of Semantic Tagging will follow in the next Section ("The YUMA Suite"). Many of the properties of a Semantic Tag, however, should also be self explanatory (e.g. `URI`, `Primary Label`, `Primary Description` or `Primary Language`).

The `Alternative Labels` and `Alternative Descriptions` properties are lists of Plain Literals: as shown in Fig. 3, a Plain Literal is a String value with an optional language code. Through this mechanism, Semantic Tags may include labels in alternative spellings, or labels/descriptions in languages other than the tag's primary language.

The `Semantic Relation` may define a relation that is part of a semantic vocabulary or ontology, to make the relation between the tag and the annotated media (fragment) semantically more expressive. In case no `Semantic Relation` is specified for the tag, RDF representations will express the relation between tag and annotated media as `rdfs:seeAlso` per default.

The `Type` property is a freely definable field for application-specific use.

The `GeoLocation` field is available for any tags that represent real-world entities. In particular, if the tag represents a reference to a place (such as `dbpedia:Vienna`), `GeoLocation` may be used to include the geographical data (point, polygon, etc.). At present, YUMA does not make any assumptions on either the geographical reference system (WGS84 or other) or expression format (WKT, KML, GML, etc.). The contents of the `GeoLocation` field are simply treated as a String and it is the responsibility of the client to handle the data appropriately.

JSON Annotation Format

The most compact serialization format for the YUMA annotation model is implemented in JSON. Properties in the model are mapped to a JSON structure using pre-defined keys. The JSON snippet below provides an example. (Please note that timestamps are serialized according to Unix Time.)

```
{
  "parent-id" : "" ,
  "root-id" : "" ,
  "title" : "The Pillars of Hercules" ,
  "text" : "In antiquity, the Strait of Gibraltar was also as The Pillars
           of Hercules.",
  "scope" : "PUBLIC" ,
  "last-modified" : 1224043200000 ,
  "created" : 1224043200000 ,
```

```
    "created-by" : "rsimon" ,
    "media-type" : "IMAGE" ,
    "object-uri" :  "http://upload.wikimedia.org/wikipedia/commons/e/ec/
                     Waldseemuller_map.jpg"
  }
```

RDF Annotation Format

As mentioned in the API description, the YUMA Server also offers RDF serialization in different RDF vocabularies (OAC, LEMO) and serialization languages (RDF/XML, N3, Turtle). The snippet below provides an example for RDF/XML serialization according to the OAC annotation vocabulary.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://www.w3.org/2008/content#"
    xmlns:j.1="http://purl.org/dc/terms/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:oac="http://www.openannotation.org/ns/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

  <rdf:Description
     rdf:about="http://dme.ait.ac.at/samples/maps/oenb/AC04248667.tif">

    <rdf:type rdf:resource="http://www.openannotation.org/ns/Target"/>
  </rdf:Description>
  <rdf:Description
     rdf:about="http://dme.ait.ac.at/yuma-server/api/annotation/618">

    <dc:title>Corsica</dc:title>
    <j.1:created>2011-04-11 11:12:34.295</j.1:created>
    <j.1:modified>2011-04-11 11:12:55.747</j.1:modified>
    <dc:creator>rsimon</dc:creator>

    <oac:hasBody rdf:resource="http://dme.ait.ac.at/yuma-
                               server/api/annotation/618#body"/>

    <oac:hasTarget rdf:resource="http://dme.ait.ac.at/yuma-
                                 server/api/annotation/618#target"/>

    <rdf:type rdf:resource="http://www.openannotation.org/ns/Annotation"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://dme.ait.ac.at/yuma-
                              server/api/annotation/618#target">
    <rdf:type
     rdf:resource="http://www.openannotation.org/ns/Target"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://dme.ait.ac.at/yuma-
     server/api/annotation/618#body">

    <rdfs:label>The island of Corsica.</rdfs:label>

    <rdfs:seeAlso rdf:resource="http://pleiades.stoa.org/places/991339"/>
    <rdfs:seeAlso rdf:resource="http://sws.geonames.org/3175395/"/>
    <rdfs:seeAlso rdf:resource="http://sws.geonames.org/3017382/"/>

    <rdf:type rdf:resource="http://www.openannotation.org/ns/Body"/>
  </rdf:Description>
```

```
</rdf:RDF>
```

Developer Information

To conclude the overview of the YUMA Server, this subsection provides a brief introduction into the structure of the YUMA Server code base. Programmers planning to modify or extend the source code should first familiarize themselves with one or more of its library dependencies (library versions reflect the status at the time of writing):

- RESTEasy: JAX-RS implementation used for the API controllers (http://www.jboss.org/resteasy, version 2.0.1.GA)
- Hibernate: persistence framework for relational databases (http://www.hibernate.org/, version 3.6.0.Final)
- Jena Semantic Web Framework: used for RDF (de-) serialization (http://jena.sourceforge.net/, version 2.6.4)
- Rome: RSS/GeoRSS feed generation (http://rome.dev.java.net/, version 1.0)
- Apache Wicket: Web framework used for the GUI (http://wicket.apache.org/, version 1.4.14)

The YUMA Server package structure is roughly organized as follows:

- `at.ait.dme.yuma.server.config` contains classes necessary for runtime access to configuration data.
- `at.ait.dme.yuma.server.controller` contains the abstract base classes for REST controllers and input/output format handling classes, plus various implementations for currently supported formats. Implementations are grouped in dedicated subpackages, currently `json`, `opensearch`, `rdf` and `rss`.
- `at.ait.dme.yuma.server.db` contains an abstract base class for storage backends, plus various implementations. Implementations are grouped into dedicated subpackages, currently `hibernate` (relational backends), `mongodb` (MongoDB NoSQL database) and `europeana` (Europeana Annotation API).
- `at.ait.dme.yuma.server.exception` contains specific exceptions and their REST mappings, where applicable.
- `at.ait.dme.yuma.server.gizmos` is reserved for experimental features. Currently, the only existing gizmo is a language guesser that determines the language of the annotation based on an N-Gram model.
- `at.ait.dme.yuma.server.gui` contains the user interface implementation.
- `at.ait.dme.yuma.server.model` contains the data model classes.

Getting Started

Before getting started, locate the file `web.xml.template` in the src/main/webapp/WEB-INF folder. Create a copy of this file named `web.xml` and make sure the settings (see above) are appropriate to your system setup.

The YUMA Server project comes with an included starter class that will launch the project in an embedded Jetty Web server (version 6.1.26). This way, you can work interactively on the code in

your IDE without the need to constantly re-build and re-deploy. Please ensure that your IDE compiles the application classes to the src/main/webapp/WEB-INF/classes directory. Then locate the class

`at.ait.dme.yuma.server.bootstrap.StartAnnotationServer`

in the /src/main/test folder, and launch it as a Java application. Per default, Jetty will launch on port 8080. The YUMA Server will be available at http://localhost:8080/yuma-server. Modify the settings in the starter class if you want a different port or application root path. When developing, please note that changes in HTML templates and CSS will be effective immediately, while changes in Java classes require a restart of the Jetty before taking effect!

Adding a New Input or Output Data Format

You can add new input and/or output formats to the annotation server by implementing a custom `FormatHandler`, along with your own subclass of the `AbstractAnnotationController` (both found in the `at.ait.dme.yuma.server.controller` package). `FormatHandlers` are responsible for parsing and serializing data formats. `AnnotationControllers` expose these formats through REST endpoints. Refer to the other existing format implementations (JSON, RDF, RSS) for guidance. Be sure to read the JavaDoc in the code (especially in the base classes) for instructions and hints.

Adding Support for a New Storage Backend

You can add support for new types of storage backends by implementing your own custom subclass of `AbstractAnnotationDB` (see `at.ait.dme.yuma.server.db` package). Subclasses need to provide a set of methods for basic CRUD and query operations. Please note that this is normally **not** required if you want to attach a new relational database! If your database is among the Hibernate-supported databases[3], you should only need to add the appropriate JDBC driver to the project, and edit the Hibernate configuration in the web.xml file. (See also the deployment instructions above.)

[3] http://community.jboss.org/wiki/SupportedDatabases

The YUMA Suite

The YUMA Suite is a set of browser-based end-user tools for annotating content of a number of specific media types. At present, the YUMA Suite includes tools for *images* and *digitised maps* as well as prototype implementations for *audio* and *video* annotation. The map tool is similar to the image tool, but features a “Google-Maps-like" interface for high-resolution content, and adds geographical features such as geo-referencing and map-overlay. The audio and video prototypes have been developed in Adobe Flash/Flex technology initially, but are now being re-designed and re-implemented on the basis of HTML 5.

The user interface offers similar functionality across all of the media types. Each tool includes a list of all existing annotations (e.g. in a sidebar or floating window, cf. Fig. 4 and 5) and provides GUI elements for creating, editing, and deleting annotations. Since annotations can pertain either to the digital item as a whole or to a part of it, YUMA provides selection functionality which allows the user to mark regions: Fig. 4, for example, shows range annotations in the audio annotation prototype; Fig. 5 shows how polygon shapes can be drawn in the map annotation tool to indicate a specific area on the map. To facilitate communication and collaboration, it is possible to (a) reply to annotations, and (b) keep track of discussions around a particular media item or annotation via RSS feeds.

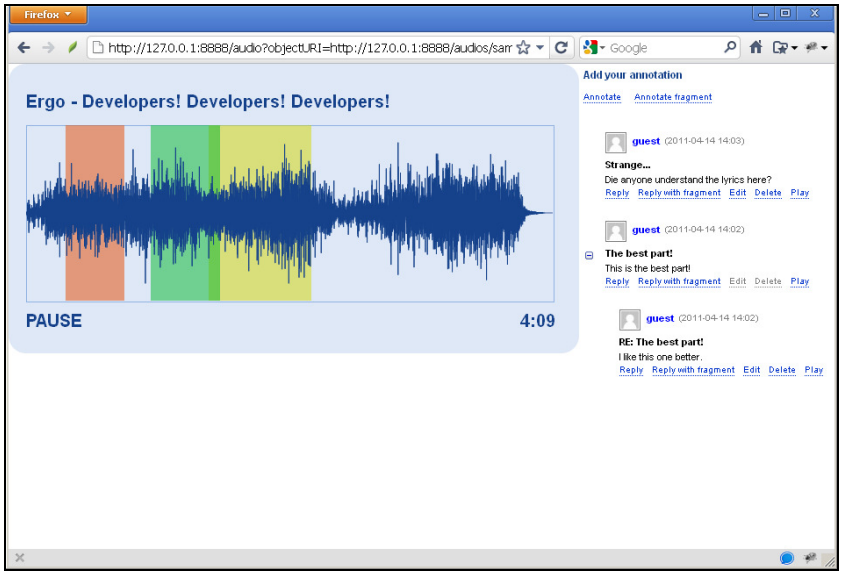


Fig. 4. Audio segment annotation (HTML 5 prototype)

The map annotation tool provides several features not found in the other tools: first, the tool leverages zoomable Web image formats (such as Zoomify[4] or TMS[5]) to display high-resolution map scans in the Web browser. If no zoomable version of the map image is available, the system can perform on-the-fly conversion from a wide range of source image formats (including, among others, JPEG, TIFF or JPEG 2000) to a number of zoomable Web image formats. (The integrated

[4] http://www.zoomify.com/

[5] http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

conversion functionality has additionally been made available as a stand-alone open source project named MagickTiler[6].)

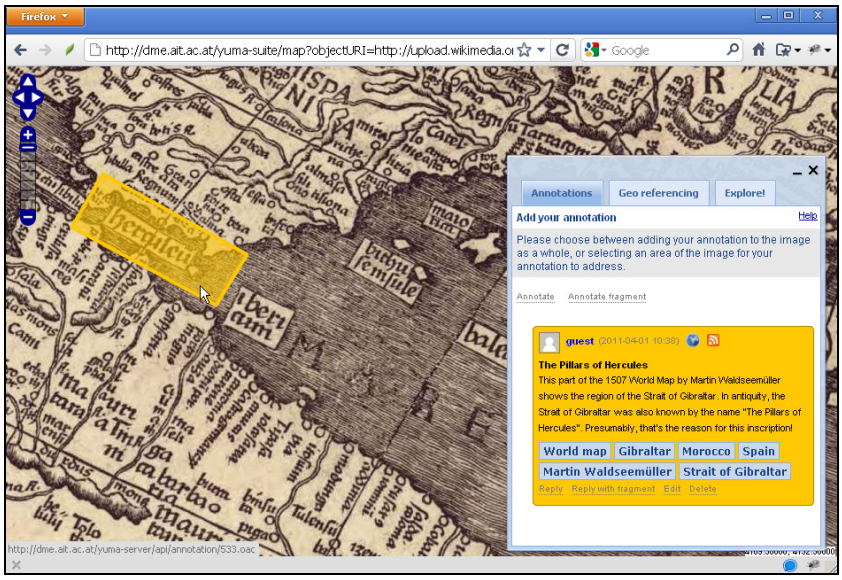


Fig. 5. Map annotation and shape drawing

Second, the tool offers the ability to geo-reference a map by collaboratively adding so-called *control points* – recognizable points on the map of which the geographical coordinates are known: e.g. known landmarks, cities, natural formations, etc. depicted on the map. Estimated geo-coordinates of any point on the map can then be calculated by interpolating between those known control points. Once at least 4 control points have been added to a map, the map annotation tool can offer additional features such as place search or overlay of present-day country borders (see Fig. 6, top left). Furthermore, annotation shapes can be automatically translated to their (approximate) geo-coordinates and then overlaid on a present-day map, shown in a separate floating window (see Fig. 6, top right) or exported to the KML format for viewing in Google Earth or similar KML-compatible virtual globe browsers (see Fig. 6, bottom).

[6] http://code.google.com/p/magicktiler

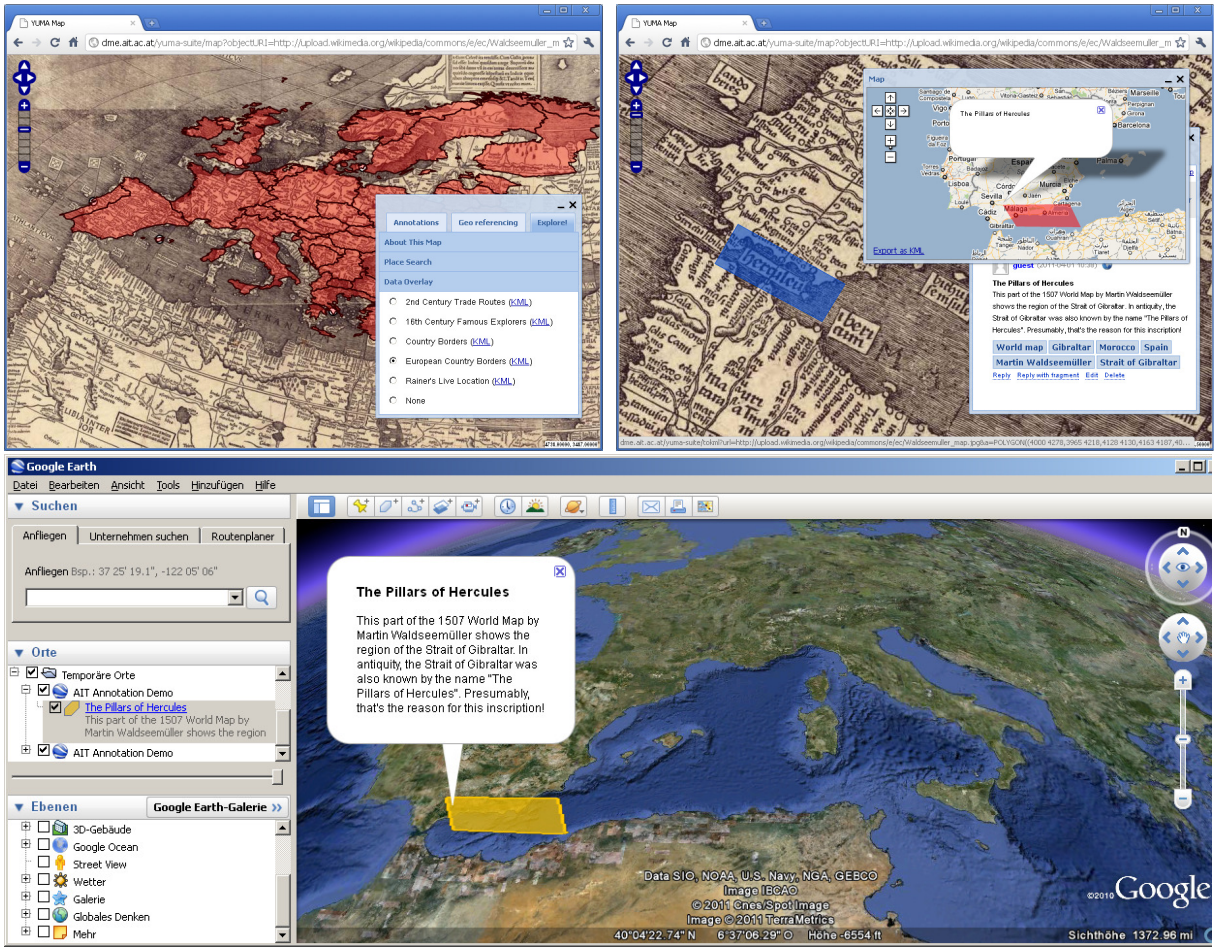


Fig. 6. Data overlay (top, left), export of annotations to geo-coordinates (top, right), export to KML for viewing in Google Earth (bottom)

Semantic Tagging

In general, YUMA enables users to create "Post-It"-style free-text annotations. In addition, however, YUMA also supports *semantic tagging*: when users create or edit annotations, the system supports them in making these semantically more expressive by suggesting links to semantic resources which may be relevant to the context of the annotation. Using Named Entity Recognition (NER), the system attempts to identify names of e.g. places, persons or other identifiable concepts in the annotation text, and will suggest appropriate links.

The map annotation tool additionally suggests links to relevant geographical entities pertaining to the annotated area. For example, if a user annotates the region of the Strait of Gibraltar on a historic map of the world (as shown in Fig. 7), the system will suggest the countries – e.g. Gibraltar and Spain – and cities – Malaga or Torremolinos – in the area. These suggestions are presented in the form of a tag cloud. Hovering over a tag with the mouse pointer will display a short textual description for that tag, allowing the user to judge whether it is truly relevant to the annotation. The user can accept the suggested tag by clicking on it. The tag will then show a tick mark to indicate that it has been selected, and that the underlying link has been added to the annotation.

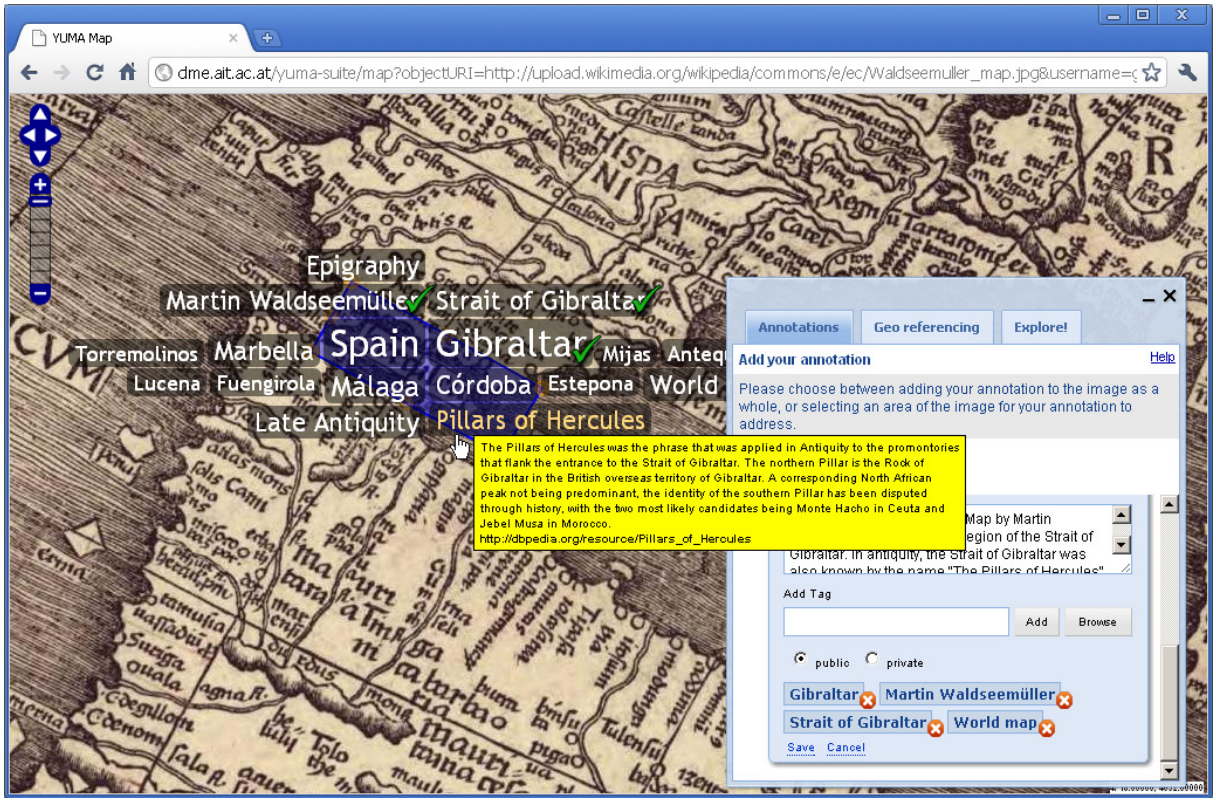


Fig. 7. Semantic tagging with the Context Tag Cloud

While it is possible to configure YUMA to work with an institutional vocabulary, YUMA's primary approach to semantic tagging is based on Linked Data: the current implementation relies on a combination of DBpedia Spotlight[7] and Geonames[8] Web services to produce tag suggestions. However, different NER tools (such as GATE[9]) and data sources (such as the Pleiades[10] Ancient World Gazetteer) have been experimented with as well.

Adding semantic context to user annotations provides a number of benefits: firstly, unlike a free-form tag, a link to a semantic resource is not ambiguous. By clicking the “Gibraltar” tag, the user makes it explicit to the system that the annotation relates to Gibraltar, the British Overseas Territory situated between Spain and Morocco – not Gibraltar in Venezuela or Gibraltar, Wisconsin. Secondly, semantic resources are interlinked with more data on the Linked Data Web. This means that users can easily obtain additional information – e.g. short explanatory text abstracts, related geographic or demographic data, or information about related persons. They can furthermore discover related online content such as relevant Web pages, images, or other items in related Linked Data sets. Leveraging Linked Data, it is also a straightforward process to obtain synonymous name variants, alternative spellings, or names in different languages for a resource. Since such information can be included by the system when storing the annotation, it can be considered during the retrieval process alongside the item’s original metadata. This way,

[7] http://dbpedia.org/spotlight

[8] http://geonames.org

[9] http://gate.ac.uk/

[10] http://pleiades.stoa.org/

multilingual and synonym search become possible without the need for additional manual cataloguing efforts. Moreover, since all links generated with YUMA are user-verified, it can be expected that the quality and correctness of the links is high, in particular if links have been confirmed by multiple users independently. This opens the potential to draw on these links to augment the original metadata of the object.

Online Demonstrations and Additional Information

Further information on tool usage, features and developer information can be found online. The main showcase page is located at

http://dme.ait.ac.at/annotation

The page provides access to a live YUMA sandbox installation that can be used freely, without registration, on sample content kindly provided by the Swedish National Heritage Board (images), the Austrian National Library (maps), the DISMARC project (audio) and the Netherlands Institute for Sound and Vision (video).

Additional screencasts with up-to date information about the map annotation tool are available here:

http://vimeo.com/21798451 (map geo-referencing)

http://vimeo.com/21798530 (map annotation and semantic tagging)

http://vimeo.com/21798618 (enhanced search: multilingual and synonym search)

A comprehensive screencast covering all of the above aspects is located here

http://dme.ait.ac.at/annotation/yuma_map_demo_full.htm

An introduction MagickTiler, the image tiling software that evolved out of YUMA's on-the-fly tiling functionality can be found here:

http://www.slideshare.net/aboutgeo/2010-1216-magicktiler

Client Suite Requirements and Deployment

Just like the YUMA Server, the YUMA Suite is delivered as a Java Web application archive (.war) file. For local deployment, the YUMA Suite requires Java 1.6, and a Java Servlet container that implements the Servlet 2.4 specification such as Tomcat version 5.5 (or higher). To deploy the YUMA Suite, follow these steps:

- Deploy the .war in your Servlet container
- Open the `web.xml` file, located in the /WEB-INF folder
- Adapt the configuration in the `web.xml` file. Modify the following parameters according to your environment:
  - `annotation.server.base.url` – the URL of your YUMA Server installation
  - `tag.server.url` – the URL of a YUMA Tag Server (optional, not covered in this document)
  - `tiles.root.path` – the path to the directory where the map annotation tool should store dynamically generated tilesets for map images in non-zoomable image formats (Note that in the current version, the path must be inside your Web application directory; e.g. if your application is located inside the directory

`/tomcat6/webapps/yuma-suite`, then your tileset root dir should be in `/tomcat6/webapps/yuma-suite/tilesets`)

- `gmaps.api.key` – a valid Google Maps API key for your domain (this is required for Google Maps overlay in the map annotation tool, compare Fig. 6, top right). Please visit http://code.google.com/apis/maps/signup.html to obtain a free API key for your domain!

Developer Information

To conclude the overview of the YUMA Suite, this subsection provides a brief introduction into the structure of the YUMA Suite code base. The YUMA Suite is built with the Google Web Toolkit (GWT, version 2.2 at the time of writing). Those planning to modify or extend the source code should therefore first have a firm understanding of GWT. Furthermore, it makes sense to get familiar with one or more of the YUMA Suite's various (Java and JavaScript) library dependencies (library versions reflect the status at the time of writing):

- gwt-mosaic: GWT GUI widget library (http://code.google.com/p/gwt-mosaic/, version 0.4.0-RC4)
- Apache Wicket: Web framework used for the GWT host pages (http://wicket.apache.org/, version 1.4.14)
- RESTEasy: JAX-RS implementation used for the RESTful communication with the YUMA Server (http://www.jboss.org/resteasy, version 2.0.1.GA)
- OpenLayers: JavaScript Web mapping library used as the basis for the map viewing GUI in the map annotation tool (http://openlayers.org/, version 2.9)
- Raphaël: JavaScript vector graphics library, used for the implementation of the Context Tag Cloud (http://raphaeljs.com/, version 1.5.2)
- MagickTiler: Java library for converting images into formats suitable for publishing as zoomable Web images (http://code.google.com/p/magicktiler/, version 0.9)

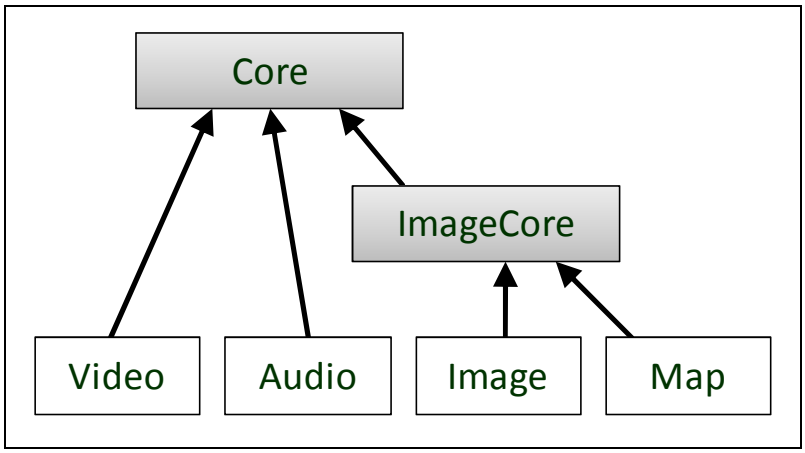


Fig. 8. Suite GWT Project Module Structure

The YUMA Suite is a GWT multi-module project. (Please refer to the documentation on organizing GWT projects[11] for background information about modules.) The modules are structured as shown in Fig. 8. Grey boxes denote modules with core functionality only (i.e. modules that do not declare an application entrypoint); white boxes denote application modules (i.e. modules with an entrypoint).

The **Core** module contains all functionality that is common across all media annotation tools (e.g. the base GUI elements for displaying annotation lists, or the functionality for exchanging data with the YUMA Server). Audio and Video modules inherit the Core module directly.

Since there are a lot of commonalities between the image and the map annotation tool, there is an additional **ImageCore** module, which encapsulates all common features of these two tools (e.g. the image-specific annotation editing form implementation or the tag cloud). Image and Map modules inherit from this module.

In terms of package structure, the YUMA Suite is roughly organized as follows: At the top level

- `at.ait.dme.yuma.suite.apps` contains the actual annotation tool modules
- `at.ait.dme.yuma.suite.framework` contains common server-side components, in particular the host page environment

Inside the `apps` package, the code is organized according to GWT conventions. Five different subpackages exist for the `core`, `image`, `map`, `audio` and `video` modules, respectively. The ImageCore module is located in the `image.core` subpackage.

Getting Started

As explained above, the Suite project consists of four separate executable GWT modules. Please refer to the Google Web Toolkit developer documentation[12] for details on how to launch a module in development mode. (For Eclipse users, the project comes with pre-configured Launch configurations for convenience, named `StartImageTool`, `StartMapTool`, `StartAudioTool`, `StartVideoTool`, respectively.)

Before starting in development mode, locate the file `web.xml` in the src/main/webapp/WEB-INF folder and make sure the settings (see above) are appropriate to your system setup.

Adding a Client Implementation for a New Media Type

Since the YUMA Framework is inherently media-independent, the Suite's base modules (in particular the Core module) can be used to create additional application modules, which implement annotation functionality for new media types. A new application module should be placed in a dedicated sub-package, inside the `at.ait.dme.yuma.suite.apps` package (for example "`at.ait.dme.yuma.suite.apps.pdf`", for a PDF annotation module). Key steps to take include:

- Create a GWT module descriptor file named <ModuleName>.gwt.xml in the module sub-package (e.g. Pdf.gwt.xml).

[11] http://code.google.com/webtoolkit/doc/1.6/DevGuideOrganizingProjects.html

[12] http://code.google.com/webtoolkit/doc/latest/DevGuideCompilingAndDebugging.html

- Inherit the Core module. (Refer to GWT developer documentation or consult the module descriptor files of other YUMA Suite application modules for reference.)
- The `at.ait.dme.yuma.suite.apps.core.client.MediaViewer` class is an abstract base GUI class for displaying annotatable media in the browser. The new module must provide an implementation of this class, according to its specific media type. Add this implementation to your GUI as the main viewing area.
- The `at.ait.dme.yuma.suite.apps.core.client.treeview.AnnotationPanel` class implements the "Annotation Panel" GUI element. The Annotation Panel displays the list of annotations for the current media item (cf. Fig. 4 and Fig. 5). It can be added to your GUI as is. However, your module needs to provide it with media specific implementations of the following two classes upon instantiation:

  1. `at.ait.dme.yuma.suite.apps.core.client.treeview.AnnotationTreeNode`
  2. `at.ait.dme.yuma.suite.apps.core.client.treeview.AnnotationEditForm`

  Class (1) defines how a single annotation is presented in the Annotation Panel list; (2) defines the design of the media-specific editing form which is shown in the panel while editing an annotation.
- Serialization and parsing of annotations (required for communication with the YUMA Server) is handled transparently in the `JSONAnnotationHandler` class in the `at.ait.dme.yuma.suite.apps.core.server.annotation` package. However, unless your application module re-uses an existing fragment vocabulary (e.g. SVG for image and map fragments) you will need to extend the handler with your own parsing- and serialization-logic, according to your media fragment format.
- The new application module will also require a host page in which to run. Follow the examples in `at.ait.dme.yuma.suite.framework.pages` for information on how to create your own host page.

Conclusions and Future Work

In this document we have provided an overview of the YUMA Annotation Framework, which is currently being showcased in the Europeana ThoughtLab.

Reactions to the tool which have been documented as part EuropeanaConnect user testing (Milestones M5.8.1 and M5.8.2), or which have been received during the course of presentations and demonstrations have been predominantly positive. However, a realistic assessment of the toolset's utility and usability to portal visitors – as well as its limitations – will only be possible after a full production-level deployment. For Europeana, such a deployment is currently not planned. Due to IPR uncertainties and strategy changes in its setup phase, Europeana does not include full digital items as originally foreseen, but only provides metadata and thumbnails, which are not sufficient for annotation purposes.

Ongoing and future work is therefore focused on easing the integration of YUMA with other portal environments or Content Management Systems, as well as on identifying specific use cases and scenarios where subsets of YUMA's various functionalities (semantic tagging, map browsing, geo-referencing) can be applied.

A second stream of ongoing work pertains to evaluating the utility of semantically enriched annotations for the purpose of retrieval in large collections. To this end, we are currently conducting a crowdsourcing experiment[13] based on approx. 6.000 digitized historic maps (and their metadata records) provided by the US Library of Congress. It is planned to make the results of this experiment available at appropriate conferences. (Preliminary results will be presented at JCDL 2011.)

[13] http://compass.cs.univie.ac.at

Publications

The following publications were produced during the development of YUMA until the time of writing. Please note that parts of this deliverable are based on these.

Simon, R., Haslhofer, B., Robitza, W. And Momeni, E. (2011) Semantically Augmented Annotations in Digitized Map Collections. Proceedings of the Joint Conference on Digital Libraries (JCDL), Ottawa, Canada, June 13–17, 2011. (Accepted for publication).

Simon, R., Haslhofer, B. and Jung, J. (2011) Annotations, Tags & Linked Data – Metadata Enrichment in Online Map Collections through Volunteer-Contributed Information. 6th International Workshop on Digital Technologies in Cartographic Heritage, April 7–8, 2011, The Hague, Netherlands.

Haslhofer, B. and Simon, R. (2011) Historic Map Annotations with YUMA. *Workshop on 'Using the OAC Model for Annotation Interoperability'*, Chicago, IL, USA.

Simon, R., Korb, J., Sadilek, C., Baldauf, M. (2010) Explorative User Interfaces for Browsing Historical Maps on the Web. *e-Perimetron*, 5 (3): 132–143.

Simon, R., Sadilek, C., Korb, J, Baldauf, M. and Haslhofer, B. (2010) Tag Clouds and Old Maps: Annotations as Linked Spatiotemporal Data in the Cultural Heritage Domain, *Proceedings of the Workshop On Linked Spatiotemporal Data 2010*, held in conjunction with the 6th International Conference on Geographic Information Science (GIScience 2010), September 14–17, 2010, Zurich, Switzerland.

Haslhofer, B., Momeni, E., Gay, M., Simon, R. (2010) Augmenting Europeana Content with Linked Data Resources. *6th International Conference on Semantic Systems*, Graz, Austria. ACM.

Sadilek, C., Simon, R., and Haslhofer, B. (2010) Service Orchestration for Linking Open Data: Applying a SOA principle to the Web of Data, *1st International Workshop on Recent Trends in SOA Based Information Systems (RTSOABIS 2010)*, co-located with the 12th International Conference on Enterprise Information Systems, June 8–12, 2010, Funchal, Madeira, Portugal.

Simon, R., Korb, J., Sadilek, C. and Schmidt, R. (2009) Collaborative Map Annotation in the Context of Historical GIS, *Proceedings of the Workshop on Geospatial Computing for the Arts, Humanities and Cultural Heritage*, Oxford, UK.

Description of Software Developed for Europeana within EuropeanaConnect

The software is a showcase for the Multimedia Annotation Tools (referred to as *YUMA*, short for *YUMA Universal Multimedia Annotator*) developed in Work Package 5. The YUMA suite of annotation tools is being implemented under the leadership of the AIT Austrian Institute of Technology, in cooperation with Vienna University.

Link to software	http://dme.ait.ac.at/annotation
Login information	not required
Development environment	no specific requirements (choose your favourite)
Programming language used	Java 1.6
Application server used	Tomcat 6
Database requirements	Hibernate-supported RDBMS required (see list of supported DBs: http://community.jboss.org/wiki/SupportedDatabases
Operating system requirements	no specific requirements
Port requirements / default ports used	no specific requirements (matter of configuration)
Interface	HTTP
Licensing conditions	EUPL v1.1