Appendix 5.

Annotation tool overview

Trish Rose-Sandler and Marcela Mora

Initially, a characterization of four open source annotation tools (Hypothesis.is, Digilib, Annotorious, RERUM) was made based on the following criteria:

- Purpose
- License
- Current Level of Development
- Hardware requirements
- Software requirements
- Functionality for creating user roles
- Standards compliance
- Storage Options
- Support

As the project was granted a 1-year extension, and the tools scenario was further developed, another four additional open source interactive annotation tools were also considered and mentioned (RECOGITO, Annotator, VIA, Pundit Pro) in the overview of this assessment.

Hypothes.is

https://web.hypothes.is/

Purpose

Hypothesis is a non-profit organization that creates open source software, is involved in standards development and fosters community. Using annotation, we enable sentence-level note taking or critique on top of news, blogs, scientific articles, books, and terms of service, ballot initiatives, legislation, and more.

License

Hypothesis is open source. BSD License

Current Level of Development

Very active – Hypothesis is very embedded in the community in which it serves. That said it is a non-profit and could terminate its services at any time Termination Conditions "We may launch new or terminate existing services at our discretion. If we discontinue our public annotation services, we will give you at least 30 days' notice to download or otherwise preserve your data. It is our objective that even if we no longer exist as an active organization, we will make provisions to keep annotations and other data publicly available and preserved for as long as possible."

Hardware requirements

None since it runs through the user's browser

Software requirements

Download Hypothesis to your browser and can begin annotating webpages, PDF or ePub documents

Functionality for creating User Roles

Users have to login to create annotations. Annotations can be shared with the public, kept private or limited only to a group

Standards compliance

Based on Open Annotation Collaboration

Storage Options

Data is stored in the cloud centrally on Amazon Web Services

Support

Mailing list dev+subscribe@list.hypothes.is

Slack channel http://slack.hypothes.is/

Archive https://groups.google.com/a/list.hypothes.is/forum/#!forum/dev

Digilib

http://digilib.sourceforge.net/

Purpose

Digilib is a web based client/server technology for images. The image content is processed on the fly by a Java Servlet on the server side so that only the visible portion of the image is sent to the web browser on the client side. It supports a wide range of image formats and viewing options on the server side while only requiring an internet browser with JavaScript and a low bandwidth internet connection on the client side. It enables very detailed work on an image as required by scholars with elaborate viewing features like an option to show images on the screen in their original size. It facilitates cooperation of scholars over the internet and novel uses of source material by image annotations and stable references that can be embedded in URLs. It facilitates federation of image servers through a standards compliant IIIF Image API.

License

Open Source Software under the Lesser General Public License version 3.0

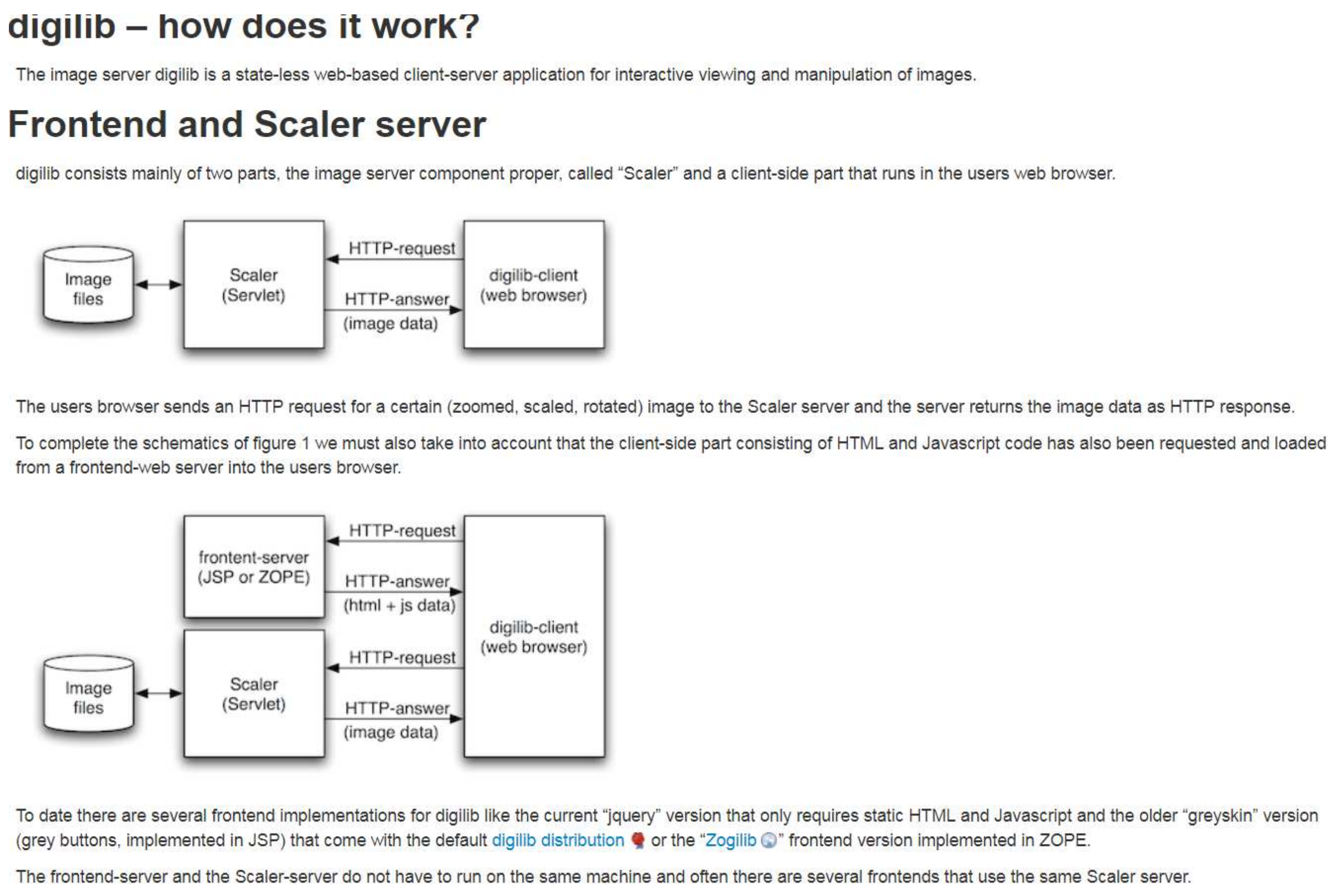
Current Level of Development

Fairly active – latest commit April 7 2018

Latest release March 26 2018

Users have been making issue requests as recent as July 2019. Developers have responded to these requests.

Hardware requirements

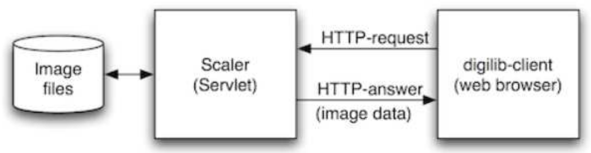


digilib – how does it work?

The image server digilib is a state-less web-based client-server application for interactive viewing and manipulation of images.

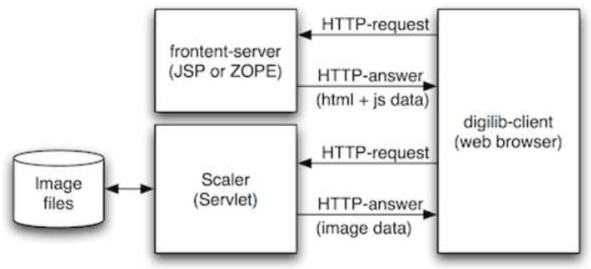
Frontend and Scaler server

digilib consists mainly of two parts, the image server component proper, called "Scaler" and a client-side part that runs in the users web browser.



The users browser sends an HTTP request for a certain (zoomed, scaled, rotated) image to the Scaler server and the server returns the image data as HTTP response.

To complete the schematics of figure 1 we must also take into account that the client-side part consisting of HTML and Javascript code has also been requested and loaded from a frontend-web server into the users browser.


frontent-server (JSP or ZOPE)
HTTP-request
HTTP-answer (html + js data)
digilib-client (web browser)
Image files
Scaler (Servlet)
HTTP-request
HTTP-answer (image data)

To date there are several frontend implementations for digilib like the current "jquery" version that only requires static HTML and Javascript and the older "greyskin" version (grey buttons, implemented in JSP) that come with the default digilib distribution or the "Zogilib" frontend version implemented in ZOPE.

The frontend-server and the Scaler-server do not have to run on the same machine and often there are several frontends that use the same Scaler server.

Software requirements

Download git, java and maven that are all freely available

The best way to get the latest and greatest digilib is using the git version control and the Maven build tool. Git will download the digilib code and Maven will compile, and install the latest digilib version and all required libraries.

Requirements:

- git
- Java JDK version 7 or later
- Maven version 3 or later

Build:

1. Clone the digilib repository

   git clone https://github.com/robcast/digilib.git

2. Change to the repository

   cd digilib

3. build and run the webapp in the embedded Jetty runtime for development

   mvn jetty:run-exploded --projects webapp
   and watch digilib at http://localhost:8080/digilib/digilib.html or follow the build and install instructions on the documentation pages.

Functionality for Creating User Roles

digilib has different mechanisms for the tasks of *authentication* - establishing the identity of the user requesting the image (more accurately the roles associated to this identity) - and *authorization* - establishing the rules for accessing specific images (the roles required to access the image).
The authe**n**tication mechanism is implemented by the digilib.auth.Auth**n**Ops interface implemented through the class configured in the digilib-config parameter authnops-class while the authorization mechanism is implemented by the digilib.auth.AuthzOps interface implemented through the class configured in authzops-class.
All authentication and authorization classes are configured through different elements in the XML configuration file

```
digilib-auth.xml
```

in the WEB-INF directory (the file name can be configured with the digilib-config parameter auth-file).

In short: classes you need to set both authnops-class and authzops-class in digilib-config with two of the described below (or implement your own) and create a digilib-auth.xml file with the configuration for the chosen implementations.

There are 6 methods to choose from when setting up authentication:

1. **IpAuthnOps – assigns roles based on IP address of user**
2. **IpServletAuthnOps - a**ssigns roles based on the IP address of the user requesting the image and uses the function of the Servlet Container if the roles provided by the IP address are not sufficient.
3. **OpenIDAuthnOps -** assigns roles based on an OpenId-Connect token passed with the request. The token can be passed either in the URL parameter id_token or as a cookie with the name id_token
4. **IpOpenIdAuthnOps -** assigns roles based on the IP address of the user requesting the image (see IpAuthnOps above) and uses an OpenId-Connect token passed with the

request (see OpenIdAuthnOps above) if the roles provided by the IP address are not sufficient.

5. **PathAuthzOps -** requests roles based on the directory path of the requested image. All images in the given directory and all its subdirectories can be accessed only if the user can provide one of the requested roles.
6. **MetaAccessAuthzOps -** requests roles using "access" information in the file metadata.

Standards compliance

Not clear what standards their annotation functionality follows

Storage Options

There is not much info about the storage options for digilib annotations except that they can be done on client or server side

- **client-side annotations**: you can put points or rectangular marks on any image as annotations that can be saved and recreated as a URL.
- **server-side annotations**: you can also put points or rectangular marks on an image with some annotation text that is shared through an annotation server.

Support

Project mailing list for developers at http://digilib.sourceforge.net/mail-lists.html
github issue tracking https://github.com/robcast/digilib
List of developers and their emails http://digilib.sourceforge.net/team-list.html

Annotorious

https://annotorious.github.io/demos/hello-world.html

Annotorious is a JavaScript annotation library. Adds annotation functionality to existing Web pages. It is (was) an ongoing open source project. It is on github. It has a LGPL license, which means it is also safe to use for commercial purposes. It does not have many features attached to it, but the focus was to make it simple to include it in a webpage.

1. Link annotorious into your webpage add a stylesheet and a javascript file
2. Mark images as annotable
    - You just use a shortcut and add a custom CSS class and when the webpage loads, nnotorius scans and makes them annotatable. You get a little drawing tool that allows you to make a box on the part of the image you want to annotate and a pop-up box appears so you can start to write a text. You can edit and delete a previous annotation.
      They plan to add a simple polygon drawing tool. You do not need to write your own JavaScript on your own page to do it

- JavaScript API-build your own mashups
- Plug-In framework-extend Annotorious
- Embeddable-integrate into host app
- Modules-additional media types (under development)

Purpose

Annotorious is being developed under the leadership of the Austrian Institute of Technology. The software has its roots in the YUMA Universal Media Annotator prototype, developed as part of the Europeana*Connect* research project. It is designed to add drawing and commenting to images on a webpage (client-side library only)

demos https://annotorious.github.io/demos/hello-world.html (Note: best viewed in Internet Explorer)

License

MIT License
Free for commercial or non-commercial use

Current level of development

The readme.md page on github says "currently unsupported" - its wiki help pages on github have not been updated since 2013. Changes to code mostly over a year ago. A post in the Google group dated Nov.28, 2018 says "annotorious has not been actively maintained for several years (And I won't have the time to change that anytime soon.)". Nevertheless, since the time of this project's evaluation, the tool started to be actively developed again a year later, and new improvements were being added almost weekly by mid-2020.

Hardware requirements

none

Software requirements

None if you just want to use the basic annotation functionality and not store anything. Enable it through linking the Annotorious source files (javascript and css) into the html header

How Do I Add Annotorious to My Web Page?

- Unzip the contents of the package on your server
- Link the Annotorious CSS file into the <head> of your Web pages
- Link the Annotorious JavaScript file into the <head> of your Web pages

Example:

```
<head>
  <link rel="stylesheet" type="text/css" href="css/annotorious.css" />
  <script type="text/javascript" src="annotorious.min.js"></script>
</head>
```

Specify which images should be annotatable

Option 1. Add a CSS class called annotatable to the image. This is the easiest way to add annotation functionality, and I'd always recommend using this approach unless your page loads images dynamically via JavaScript, after the page has loaded.

Example:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="css/annotorious.css" />
    <script type="text/javascript" src="annotorious.min.js"></script>
  </head>

  <body>
    <img src="example.jpg" class="annotatable" />
  </body>
</html>
```

Option 2: Annotation-enable your images via JavaScript, using anno.makeAnnotatable(img); You can use this approach if your page loads images dynamically via JavaScript.
Example:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="css/annotorious.css" />
    <script type="text/javascript" src="annotorious.min.js"></script>
    <script>
      function init() {
        anno.makeAnnotatable(document.getElementById('myImage'));
      }
    </script>
  </head>
  <body onload="init();">
    <img src="example.jpg" id="myImage" />
  </body>
</html>
```

A Note to jQuery Users
Some versions of Annotorious conflict with jQuery. When using Annotorious and jQuery on the same page, you should put jQuery into no-conflict mode, and either assign a new variable name to jQuery's **$**alias or, alternatively, wrap your jQuery code into a function where you can use the **$** alias in the local scope, like so:

```
jQuery.noConflict();



jQuery(document).ready(function($) {

  // You can use the locally-scoped $ in here as an alias to jQuery.

  $('div').hide();

});

```

Note: Instead of hosting the Annotorious source files yourself, you may also hot-link to the latest versions on the Annotorious site:
http://annotorious.github.com/latest/annotorious.min.js
http://annotorious.github.com/latest/annotorious.css

Functionality for creating User Roles

None

Standards compliance

Nothing on website indicates its adherence to standards but found a talk on YUMA which does say it's based on OAC

Storage options

Not built into app but there are storage plugins to connect app with databases and storage backends. For example, Parse.com (cloud storage on Parse platform hosting service https://github.com/dommmel/annotorious-parse-plugin) ElasticSearch Storage (stores annotations on ElasticSearch server).

Support

Google group https://groups.google.com/forum/#!forum/annotorious and Github issue tracker https://github.com/annotorious/annotorious/issues but both are not very active since its not being supported anymore

RERUM

http://centerfordigitalhumanities.github.io/rerum/web/#/welcome

Purpose

Java web service for a RERUM compliant digital object repository. Visit rerum.io for more general information. Want to use the API? Learn how at the API page.
Stores important bits of knowledge in structured JSON-LD objects:

- Web Annotation / Open Annotation objects
- SharedCanvas / International Image Interoperability Framework objects
- FOAF Agents
- *any* valid JSON object, even if there is no type specified!

License

Other than declaring RERUM an "open and free repository" I was unable to find any information on RERUM site about licensing of its data.

Current Level of Development

Very active

Hardware requirements

Not applicable

Software requirements

Steps for connecting your annotation server to RERUM's public annotation store

1. Register annotation server ("we suggest registrant is application developer"). Creates public agent for application, email address for person. Returns refresh token and access token that we have to copy or store
2. Connect to public RERUM server the base URL is http://devstore.rerum.io/v1

Functionality for creating user roles

Not really applicable because RERUM is not an interface for creating annotations but rather storing them, however the annotations they store always "always include asserted ownership and transaction metadata so consumers can evaluate trustworthiness and relevance"

Standards compliance

Requires annotations submitted as JSON Linked Data (or at the very least JSON data). Prefers data follow the W3C Annotation protocol. https://www.w3.org/TR/annotation-protocol/ RERUM follows REST, IIIF and Web Annotation standards to form its responses to users.

Storage Options

Basic Principles

1. **As RESTful as is reasonable**—accept and respond to a broad range of requests without losing the map;
2. **As compliant as is practical**—take advantage of standards and harmonize conflicts;
3. **Save an object, retrieve an object**—store metadata in private (__rerum) property, rather than wrap all data transactions;
4. **Trust the application, not the user**—avoid multiple login and authentication requirements and honor open data attributions;
5. **Open and Free**—expose all contributions immediately without charge to write or read;
6. **Attributed and Versioned**—always include asserted ownership and transaction metadata so consumers can evaluate trustworthiness and relevance.

What we add

You will find a __rerum property on anything you read from this repository. This is written onto all objects by the server and is not editable by the client applications. While applications may assert *anything* within their objects, this property will always tell the Truth. The details are in the documentation, but broadly, you will find:

- created specific creation date for this [version of this] object
- isOverwritten specific date (if any) this version was updated without versioning
- generatedBy the agent for the application that authenticated to create this object
- isReleased a special flag for RERUM, indicating this version is intentionally public and immutable
- releases an object containing the most recent ancestor and descendant releases
- history an object containing the first, previous, and immediate derivative versions of this object

Support

Github issue tracking https://github.com/CenterForDigitalHumanities/rerum/issues?title=Request%3A%20&body=Thanks%20for%20contributing%21

RERUM Blog https://blog.ongcdh.org/tag/rerum/

Contact directly digitalhumanities@slu.edu or 314-977-4248

The following is an overview of four additional open source interactive annotation tools:

RECOGITO

Semantic Annotation without the pointy brackets. Work on texts and images. Identify and mark named entities. Use your data in other tools or connect to other data on the Web. Without the need to learn to code.

Recogito is part of Pelagios. Pelagios is an international initiative concerned with the development of Linked Open Data (LOD) methods, tools and services to better interconnect the vast and ever-growing range of historical resources online. In particular, it associates place references within those resources to online gazetteers that offer URI-based identifiers for such places. Some of its major outputs have been the development of **Recogito**, a tool for semantically annotating place references in images and texts, and Peripleo, a service for visualizing and exploring the graph of data that these annotations form.

It has a 10 min tutorial in six languages: https://recogito.pelagios.org/help/tutorial

What is Recogito?

Recogito is an online platform for collaborative document annotation. It is maintained by Pelagios Commons, a Digital Humanities initiative aiming to foster better linkages between online resources documenting the past.

Recogito provides a personal workspace where you can upload, collect and organize your source materials - texts, images and tabular data - and collaborate in their annotation and interpretation.

Recogito helps you to make your work more visible on the Web more easily, and to expose the results of your research as Open Data.

It allows to:

- Upload a document
- Create annotations
- Identify and map places
- Export your data
- Invite other users

Annotator

by Nick Stenning & Aron Carroll

- http://annotatorjs.org/
- https://github.com/openannotation/annotator

The Annotator is an open-source JavaScript library and tool that can be added to any webpage to make it annotatable. Annotations can have comments, tags, users and more. Moreover, the Annotator is designed for easy extensibility so it's a cinch to add a new feature or behavior.
Annotator is a JavaScript library for building annotation applications in browsers. It provides a set of interoperable tools for annotating content in webpages. For a simple demonstration, visit the Annotator home page or download a tagged release of Annotator from the releases page and open demo.html.
Components within Annotator provide:

- user interface: components to create, edit, and display annotations in a browser.
- persistence: storage components help you save your annotations to a remote server.
- authorization and identity: integrate Annotator with your application's login and permissions systems.
- Components within Annotator provide:
    - - user interface: components to create, edit, and display annotations in a browser.
    - - persistence: storage components help you save your annotations to a remote server.
    - - authorization and identity: integrate Annotator with your application's login and permissions systems.

```
<script src="annotator.min.js"></script>
```

VGG Image Annotator (VIA)

Abhishek Dutta, Ankush Gupta and Andrew Zisserman

Overview
VGG Image Annotator is a simple and standalone manual annotation software for image, audio and video. VIA runs in a web browser and does not require any installation or setup. The complete VIA software fits in a single self-contained HTML page of size less than 400 Kilobyte that runs as an offline application in most modern web browsers.
VIA is an open source project based solely on HTML, JavaScript and CSS (no dependency on external libraries). VIA is developed at the Visual Geometry Group (VGG) and released under the BSD-2 clause license which allows it to be useful for both academic projects and commercial applications.
A more detailed user guide (with screenshots and descriptions) is available here.

1. **Load Images**: The first step is to load all the images that you wish to annotate. There are multiple ways to add images to a VIA project. Choose the method that suits your use case.
    - Method 1: Selecting local files using browser's file selector
        1. Click Project → Add local files
        2. Select desired images and click Open
    - Method 2: Adding files from URL or absolute path
        1. Click Project → Add files from URL
        2. Enter URL and click OK
    - Method 3: Adding files from list of url or absolute path stored in text file
        1. Create a text file containing URL and absolute path (one per line)
        2. Click Project → Add url or path from text file
        3. Select the text file and click Open
2. **Draw Regions**: Select a region shape (rectangle, circle, ellipse, polygon, point, polyline) from the left sidebar and draw regions as follows:
    - Rectangle, Circle and Ellipse

        - Press left mouse button, drag mouse cursor and release mouse button.
        - To define a point inside an existing region, click inside the region to select it (if not already selected), now press left mouse button, drag and release to draw region inside existing region.
        - To select, click inside the region. If the click point contains multiple regions, then clicking multiple times at that location shuffles selection through those regions.
    - Point
        - Click to define points.
        - To draw a region inside existing region, click inside the region to select it (if not already selected), now click again to define the point.
        - To select, click on (or near) the existing point.
    - Polygon and Polyline
        - Click to define vertices.
        - Press **[Enter]** to finish drawing the region or press [Esc] to cancel.
        - If the first vertex needs to be defined inside an existing region, click inside the region to select it (if not already selected), now click again to define the vertex.
        - To select, click inside the region. If the click point contains multiple regions, then clicking multiple times at that location shuffles selection through those regions.

1. **Create Annotations**: For a more detailed description of this step, see Creating Annotations : VIA User Guide. Click the View → Toggle attributes editor to show attributes editor panel in left sidebar and add the desired file or region attributes (e.g. name). Now click View → Toggle annotations editor to show the annotation editor panel in the bottom side. Update the annotations for each region.

1. **Export Annotations**: To export the annotations in json or csv format, click Annotation → Export annotations in top menu bar.

1. **Save Project**: To save the project, click Project → Save in top menubar.

Pundit Annotator Pro

"Pundit 2.0 is a semantic web annotation system that supports users in creating structured data on top of web pages. Annotations in Pundit are RDF triples that users build starting from web page elements, as text or images. Annotations can be made public and developers can access and combine them into RDF knowledge graphs, while authorship of each triple is always retrievable. Pundit enables users to annotate different kind of entities and to contribute to the collaborative creation of a knowledge graph. This, in turn, refines in real-time the exploration functionalities of the library's faceted search, providing an immediate added value out of the annotation effort".

Pundit Annotator is a much simpler annotator (than the above tools) and packs in just the necessary features for a solopreneur to grab ideas from the web and add comments. It provides only a Chrome extension (and not bookmarklet unlike others).

After installing its extension, you would see its button beside address bar. On any page, click it to bring the "Pundit Sidebar". Then after, you can select any text on the page and you would see two options/buttons: "Comment" and "Highlight" (both of these work as expected). For checking out your annotations, you can login to your account and browse "Notebooks".

–Semi-automatic linking of entities in text
–Configurable semantic annotation templates, allowing to create complex an- notations in few steps
–Free composition of triples to link elements in web document (e.g. words, images, images parts) to LOD entities and among each other
– Configurable annotation vocabularies of entities and relations to be used in triples {Delivery of annotations as RDF graphs via SPARQL or via open or authenticated REST API
–Delivery of the annotation environment as-a-service, so that web applications can make their content annotatable by calling a REST API (feed.thepund.it), as well as as a bookmarklet, or simply as a javascript library