Deliverable

Project Acronym: LoCloud

Grant Agreement number: 325099

Project Title: Local content in a Europeana cloud

D2.6 Crawler ready tagging tools

Revision: Final

Authors:

Asplan Viak Internet AS (AVINET)

Stein Runar Bergheim
Siri Slettvåg

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	**Public**	**x**
C	**Confidential, only for members of the consortium and the Commission Services**	

Revision History

Revision	Date	Author	Organisation	Description
1.0	26.11.2014	S.R. Bergheim S. Slettvåg	AVINET AVINET	Draft for internal review
1.1	1.12.2014	S.R. Bergheim	AVINET	Draft for partner review
1.2	7.1.2015	S.R. Bergheim	AVINET	Incorporating comments from K. Fernie, D. Gavrilis
1.3	3.2.2015	S.R. Bergheim	AVINET	Incorporating results from test cases after installation on server
1.4	12.2.2015	S.R.Bergheim	AVINET	Incorporated comments from D. Gavrilis
1.5	19.2.2015	S.R. Bergheim	AVINET	Incorporated comments from K. Fernie

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Contents

Executive Summary

The LoCloud Crawler Ready Tagging Tools (henceforth CRTT) are a set of experimental tools for automatically **extracting structured metadata** from HTML mark-up loaded from web documents. The objective of the CRTT is to verify if the crawling/indexing method applied by the mainstream search engines could be a viable, **simplified supplement** to the comprehensive Europeana ingestion process. To this end, the CRTT is validated using **small institutions** as a test case.

The basic requirements for providing content to Europeana are that the content is available online, that each digital object can be identified by a **unique URL** and that the digital object is described by a minimal set of required metadata elements.

CRTT assumes as a starting point that content provider institutions wishes to optimize its content for search and retrieval through the major search engines AND Europeana. CRTT demonstrates how **search engine optimization** through unobtrusive **embedded semantics** can achieve this end, as an alternative, an intermediate measure or a supplement to the establishment of a single-purpose data extraction and metadata ingestion mechanism for Europeana that is parallel to the web page publishing workflow.

The CRTT consists of three different components:

1. A **web application** for submitting base-URLs for crawling and/or lists of URLs pointing to deep-content - as well as defining site-specific metadata extraction rules. This interface is available for testing at **http://crtt.avinet.no**, passwords will be supplied on demand.
2. A scheduled **crawler** task that loops through base URLs and crawls them for URLs to landing pages for digital objects, optionally applying filters for which URLs to follow and which URLs to add to the collections.
3. A "spider" or scheduled **indexer** task that loops through batches of URLs and applies the extraction rules to them, populating an XML object according to a simple EDM metadata profile.

Using the CRTT, the effort required by the content-provider institution is limited to submitting a single URL or a list of URLs to deep content. **Generic extraction rules** for metadata from HTML pages are preconfigured. **Custom rules per collection** can optionally be added by the user, whereby improving the quality of extracted metadata.

Testing has demonstrated that the CRTT is a viable way of aggregating an index for **free-text searches**. For generic HTML pages without embedded semantics and without site-specific extraction rules are only suitable for searching and display in result lists, but are **not as accurate** or as complete as a manually created rich **EDM object**.

With the creation of custom extraction rules and/or enhancement of the HTML mark-up however, the resulting metadata quality is sufficient to create **qualified EDM objects**. Such adjustments will, as an added benefit, lead to **search engine optimization**.

This deliverable describes the rationale, technology, testing and next steps for the LoCloud CRTT.

1 Introduction

The LoCloud crawler-ready tagging tool (CRTT) is an experimental prototype that seeks to demonstrate how a metadata aggregation and ingestion process, as the one employed by Europeana, may be simplified. The test case that is used to validate the approach is content from small cultural heritage institutions that do not currently have an established mechanism for providing content to Europeana.

The rationale for the experiment is that one of the underlying objectives of the establishment of Europeana was to enable multi-lingual search and exploration of content from European heritage institutions that would do for the heritage sector what Google has done for the public Internet, but exclude non-quality-assured, non-heritage and commercial/advertising content.

While the mainstream search engines have a scope that is limited to “all public information on the Internet provided by anyone, anywhere”, the implementation of Europeana took as a starting point a controlled number of known content providers, mainly national libraries at the outset.

The number of content providers was too great to solve the search and exploration challenge through federated searches, yet too small, to limit the ambition of Europeana to simple full-text indexing of web pages. Instead, the chosen content ingestion process relies on comprehensive metadata being made available in addition to the digital objects published on web pages. These metadata are prepared by content providers for the single purpose of providing content for Europeana and the complexity of the metadata profiles require personal support and contact from Europeana to contributing parties.

With a potential infinite scope and institutional assignment, but a limited funding, Europeana had to limit their direct support activities to aggregators and large volume content providers out of the simple consideration of capacity. “Aggregators” act as buffers between “local” and “regional” content provider institutions and Europeana. Some of these aggregators are run by permanently funded national or regional entities; others are funded on a project-by-project basis.

This approach permits dialogue on the establishment of highly granular metadata profiles that can be used to represent and transport content from local heritage institutions to Europeana via aggregators. Aggregators both channel the content into fewer and more manageable streams for Europeana to handle and provides down-stream support to “their” institutions on how to interpret the metadata profiles as well as how to populate and shape the metadata in order for Europeana search results to become predictably relevant.

With an even wider scope, not limited to heritage, the mainstream search engines on the Internet today, Google, Yahoo and Bing, have chosen a fundamentally different approach to building their indexes than Europeana. In the place of carefully crafted data complying with comprehensive data models, e.g. the Europeana Data Model, the search engines build their indexes based on reading the complete text of web pages and trying to interpret the meaning of the various pieces of text based on the structure of the mark-up and any embedded semantics in the pages.

The only thing contributed by the content providers is a list of URIs (a site map) – or even a single URI that can be used as a starting point for crawling if the site does not contain "deep content".

The quality of each metadata record produced through the Europeana workflow will always be greater than an entry in the index of one of the mainstream search providers. The efficiency with which content is being ingested as well as the scalability of the search engine approach is infinitely better.

When comparing the performance of an auto-indexed URI towards a manually crafted metadata object in terms of a white-box text search, the results may be quite similar. In terms of more sophisticated search types where the detailed information contained in a metadata profile like EDM is used and combined, a search engine would not be able to compete on precision.

The CRTT is an experimental approach whereby the search-engine ingestion process is applied to heritage content with the aim of benchmarking this approach towards manually crafted metadata in terms of white-box text searches.

Using this approach, the content providers' responsibilities will be reduced to the following three elements, of which two are optional:

1. Provide a list of URLs according to the Google site-map protocol, a simple XML file.
2. Optionally determine "scraping" rules that can be used to extract text content from a page and map it to corresponding metadata elements in a simple metadata profile.
3. Optionally modify/enhance the structure of their HTML mark-up with embedded semantics - with the joint benefit of search engine optimization as well as improved search results on Europeana.

To aid content providers in this process, it is necessary to:

1. Provide a tool that can be used to author and test "scraping" rules, i.e. the LoCloud crawler ready tagging tools.
2. Loop through the list of URLs and process each URL using the rules defined in step one.
3. Feed the output XML/JSON into the upstream ingestion process of Europeana, either directly or via an aggregator.

The receiving-end must implement an interface where the URL of the XML-files can be uploaded and where data that are common to all resources can be specified such as source organization, country as well as content language, licensing and similar.

The LoCloud Crawler-Ready Tagging tools are an experimental implementation of the above workflow that seeks to test if this workflow could be a compliment to the comprehensive Europeana content ingestion workflow that would allow scalability to a very large volume of small institutions that today may not find a way for their content into Europeana.

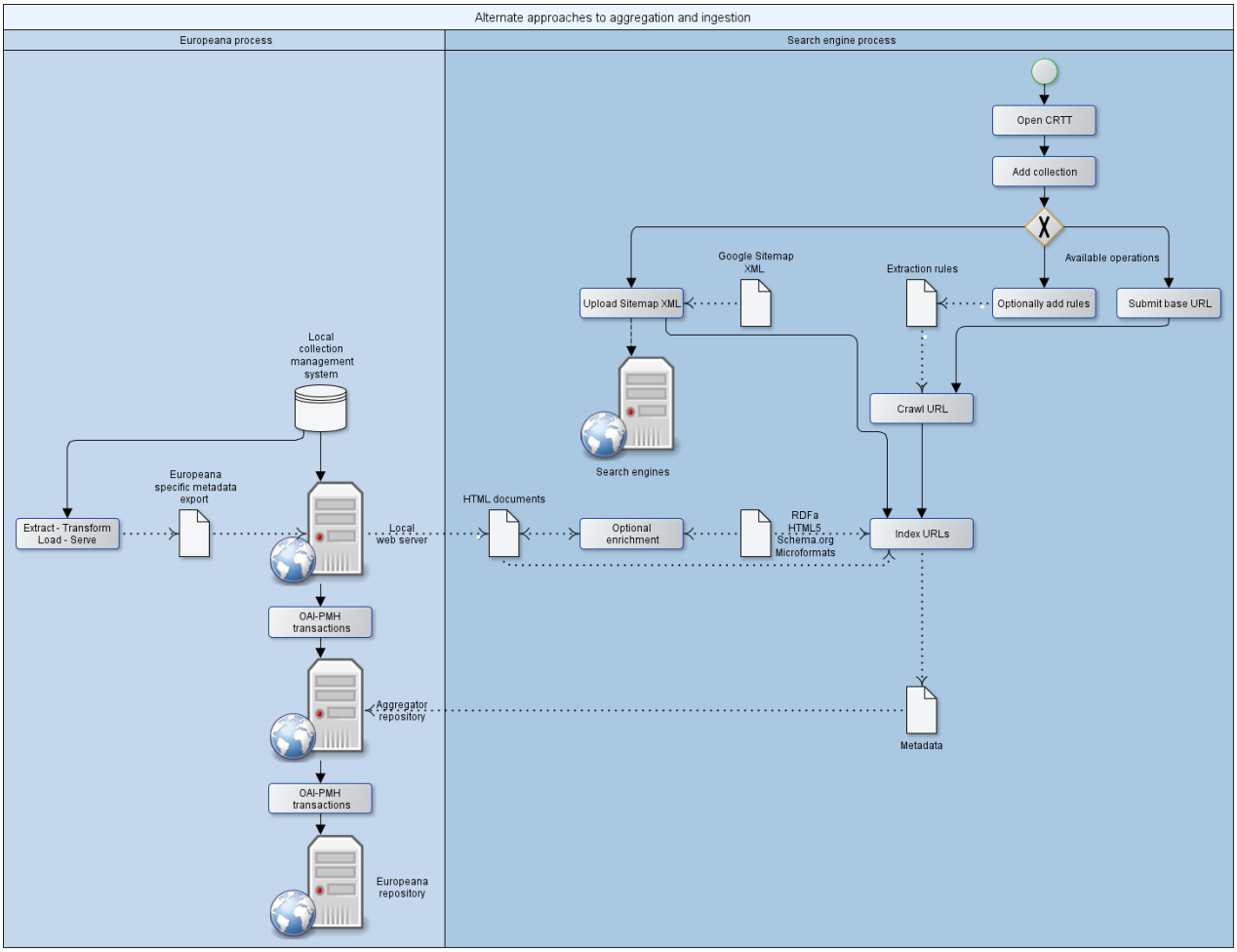


Figure 1: How CRTT fits in with the current Europeana content ingestion workflow

2 Functional requirements

This chapter lists the functional requirements of the CRTT. In this work, we followed the definition of "functional requirements" within the discipline of requirements engineering, i.e. they specify the desired outcome of the system as seen from the perspective of a user. The functional requirements drive the development of the application architecture of the CRTT.

The functional requirements uniquely identified below were gathered during the LoCloud plenary workshop in London in December 2013. In preparation of this workshop, AVINET prepared a presentation that gave an overview of relevant technologies and preliminary experiments. Requirements that were proposed by AVINET and accepted by the partners, or that were proposed by partners during the discussions following the presentation are recorded below.

Table 1: CRTT functional requirements

#	Requirement	Description
1	HTML scraping rules editor	It must be possible to author rules that define how text elements from a HTML document shall be mapped to corresponding elements from the EDM data model.
2	Preview of rule impact	It must be possible to preview how the rule works for a given sample URL without leaving authoring-view. This is necessary so that end-users can efficiently tune their rules to optimize the output.
3	Upload sitemap XML	It must be possible to upload a Google Sitemap XML-file with many URLs to the system.
5	Run tag extraction and indexing	It must be possible to loop through the URLs contained in the provided sitemap and
6	Output XML-file	It must be possible to output an XML-file with the mapped metadata so that these may be ingested into Europeana – or harvested by other EDM-aware environments.
7	Preconfigured tag extraction rules	The application must come with pre-configured rules for extraction of common embedded semantics formats such as Schema.org, Microformats and common HTML5 document structure semantics.

3 Non-functional requirements

This chapter specifies the non-functional requirements of the LoCloud CRTT. These follow the definitions of the requirements engineering discipline where non-functional requirements are defined as requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours.

Whereas the functional requirements define what the CRTT is supposed to do, the non-functional requirements define how the CRTT will achieve the functional requirements described in chapter 2. The non-functional requirements are defined by AVINET based on a study of good practice and state-of-the-art within available free and Open Source technologies.

Table 2: CRTT non-functional requirements

#	Requirement	Description
1	HTTP data reader	It must be possible to read remote data over the Http protocol and load them into a text-string or access them as a stream from within the tagging tool. The encoding should be detected based on the HTML meta tags or HTTP header information. Default encoding if not specified will be interpreted as ISO-8859-1 for the experiement. Internal processing will be done in UTF8
2	DOM parser	It must be possible to load HTML text into a document object model (DOM). The parsing of the HTML must provide elasticity for non-well-formed DOM structures to compensate for common coding practices that do not cause rendering errors.
3	Nested DOM element/attribute queries	It must be possible to access content from the DOM using nested tag and property expressions. This is the mechanism by which content in HTML pages can be selected and mapped to corresponding EDM properties.
4	Handling of microformats embedded semantics	The solution must be able to handle common microformats such as geo, rel-license, rel-tag and hMedia and map them to target EDM elements.
5	Handling of RDFa embedded semantics	The solution must be able to handle RDFa embedded tags and map them to their corresponding EDM elements.

6	Handling of Schema.org embedded semantics	The solution must be able to handle a subset of common Schema.org tags and map them to their appropriate counterparts in EDM
7	Reusable library	The solution should be implemented as a re-usable library so that it can be used in different contexts; embedded into other applications or as an online indexer.
8	Free, open source license	The system must be available via an Open Source license that permits re-use by third-parties for non-commercial purposes. The foreground IPR of the solution belongs to AVINET who is free to issue the tool under additional licenses in the interest of future sustainability.

4 System architecture

The system architecture of the LoCloud Crawler-Ready Tagging Tools was determined based on the functional and non-functional requirements as well as the following practical assumptions:

1. Everyone contributing content to Europeana must have already made this content available on the web.
2. Local institutions get their web traffic from a variety of sources – not only through Europeana.
3. The majority of the web traffic is likely to come through search engines or directly, if the institution has a well-defined and local target group.
4. The efforts made by local institutions to create a metadata export infrastructure benefits mainly Europeana and the aggregators.
5. The institutions are also interested in improving their ranking in the public search engines.
6. There are efficiency savings if the efforts made to contribute content to Europeana also improve the SEO of the institutions content and vica-versa.

Taking these elements as a starting point, the baseline system architecture of the CRTT is shown in Figure 2. An explanation of the architecture is provided below.

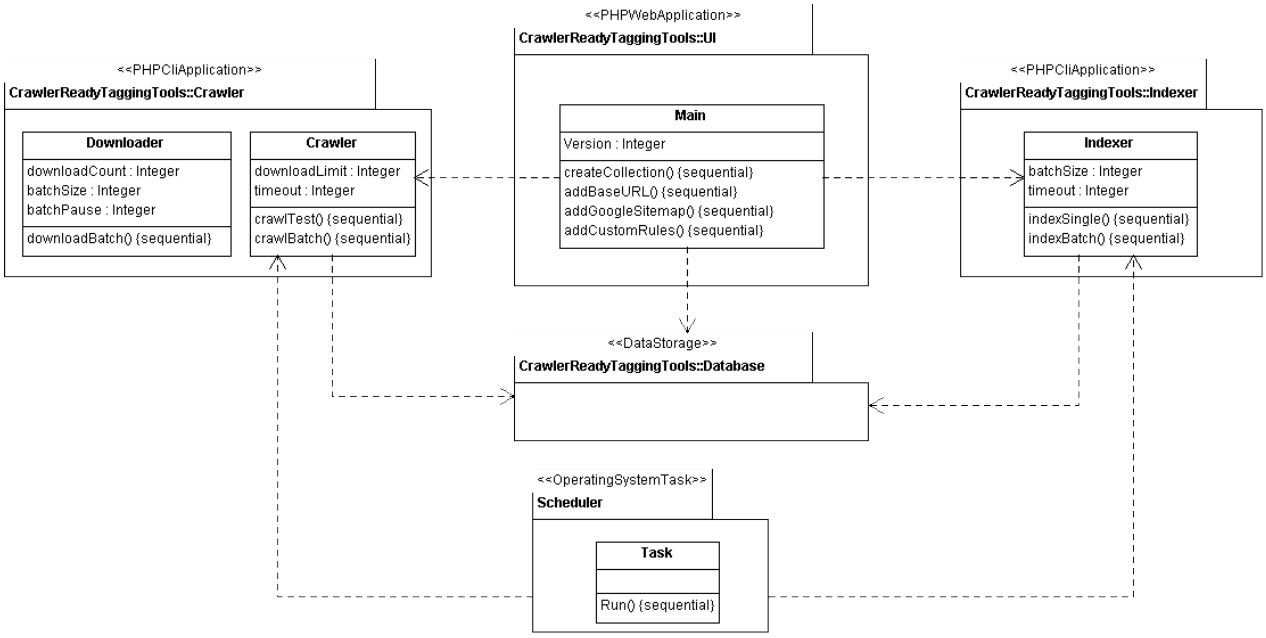


Figure 2: Baseline system architecture

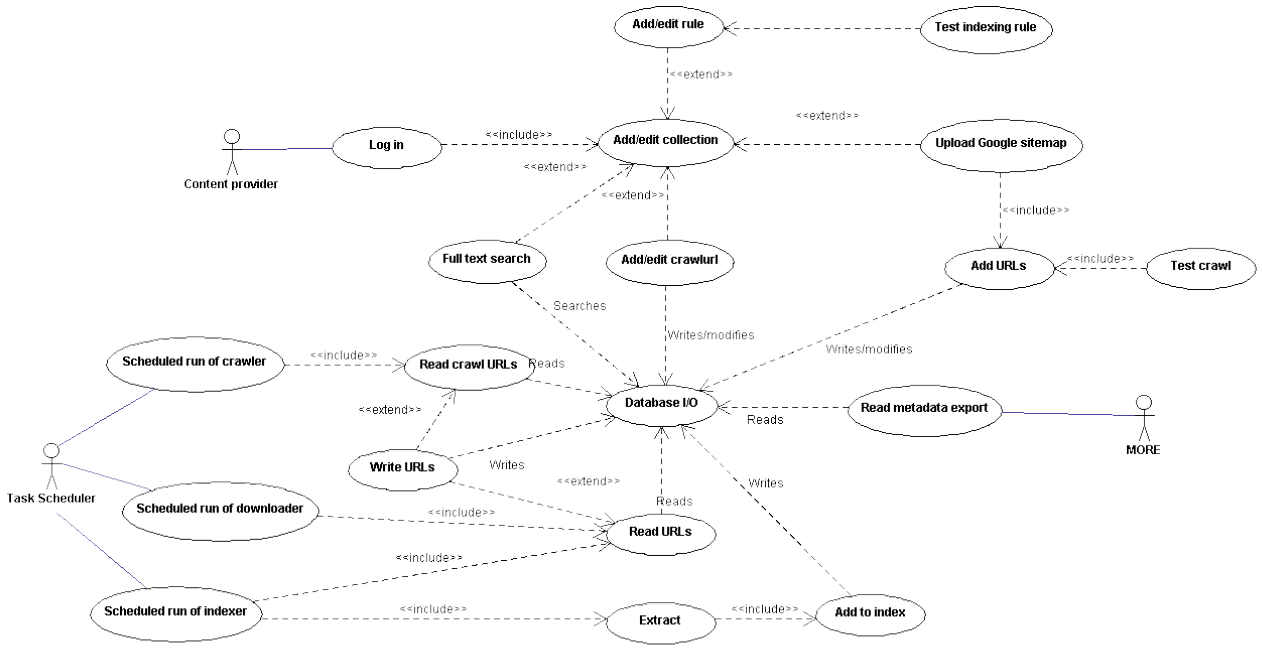


Figure 3: Use Case diagram for the CRTT

The basic premise on which the CRTT is based is that every institution that wishes to contribute data to Europeana must have an existing web site where all their information is published.

On this page, all the metadata are commonly shown as text or other visualizations. Everything that you see on a web page in a web browser is a **HTML document**, unless some special plugin is used such as Acrobat, DejaVu, Adobe Flash or similar.

Mostly, HTML is used as a layout engine without regard for the semantic meaning of its tags and elements. However, non-structured HTML can offer hints that permit us to extract or guess information about a resource. It is very common to assign styles to content elements based on their importance and function – for this reason we can use the formatting specifications to determine which headlines are important – and which are not.

However, HTML also offers the possibility to **embed semantics** within the page in a variety of ways. HTML itself offers tags that have specific meaning; it is also possible to use attributes and meta-elements, in accordance open standards, to express exact metadata are both (1) shown on the page and (2) exposes its semantic meaning to user agents such as search engine crawlers/indexers etc.

The most relevant standards to be used for this purpose include the following: RDFa, Schema.org and Microformats. These are described in more detail in chapter 7 below.

However, even without employing any of these standards it is possible to extract useful “what” metadata from many HTML-pages. For this reason, enhancement of the HTML mark-up of the content is optional. Content providers do not have to alter the structure of their HTML pages – however, if they do, they benefit not only through a simpler way to ingest content into Europeana –

but also through improved SEO and better ranking and visibility of content in Google, Bing and Yahoo! – the major sources of traffic for most web sites.

Mostly, content is kept in databases with a single server side script being responsible for reading and presenting the content on the screen. In many cases, these detailed pages are only available to users after they have performed a search and selected a result from a result list.

It is impossible for a non-user assisted search engine to perform the same operations and provide dynamic input in order to access this type of content, also known as **deep content**. In order to overcome this issue, it is necessary to provide the indexers with a list of all the valid URIs of all detailed landing pages. This is done in accordance with the **Googel Sitemap** format, widely accepted and used by mainstream search providers.

The preparation of a site-map has multiple benefits by improving the visibility and ranking in the major search engines as well as improving the metadata extraction performance of the CRTT application.

For sites with deep content, a sitemap must be submitted to the application through an **index submission form** so that it can be used as a basis for parsing and indexing (and re-indexing) of HTML documents.

For web sites that do not have deep content, it is possible to submit a base URL from which other URLs can be derived through **crawling** and **downloading** of HTML documents. This process will identify a large number of URLs that must be put through the same indexing process as deep links taken from sitemaps.

At this stage, we know that the data are located in HTML documents and we have a list of the URLs of these documents in an XML-file. We now need a number of simple mechanisms to permit us to read the data and **extract information** from the relevant tags in the mark-up.

By default, the extraction of the information will be done in accordance with a set of **pre-defined extraction rules** that are targeted at generic HTML documents and a limited number of embedded semantics types.

Since many sites do not use the HTML elements that have semantic meaning in a correct way, better results may be achieved by making custom extraction rules for each HTML presentation template. For this purpose, it is possible to specify **custom extraction rules**. In order to issue a large number of consecutive queries towards the content structure of a HTML document it is useful to load it into memory and to access it according to the XML Document Object Model (DOM) and its query language XPath.

Using these technologies, any HTML document can be queried by its content structure and any embedded semantics that may be available.

Finally, using the rules and the list of URLs, the **Indexer** loops through each URL, loads the HTML document and applies all relevant rules to it. The result is an index of documents consisting of auto-extracted metadata.

These documents are stored locally in database table and can be exported and brought forward into Europeana or one of its aggregators as qualified or non-qualified EDM objects.

5 System Implementation

5.1 Development methodology

The solution was implemented following the Scrum methodology, as is the practice for all system development activities executed by AVINET.

The methodology is simple and agile; the work is broken down into manageable tasks and added to an overall specification document called a backlog.

The requirements of the backlog are grouped into logical packages and assigned to iterative sprints.

The work is carried out in the form of daily work and interaction within the development team, so called daily Scrum.

At the completion of each sprint, the backlog and the experiences from the iterative work are reviewed. The completed work and the experience are fed into the next sprint so that the development process becomes a continuous improvement cycle.

5.2 PHP web application & programming library

In order to comply with the LoCloud objectives of all software products being free and open source, the application is implemented using the programming language PHP. All functionality is implemented as a re-usable library. The prototype implements a possible user interface, but in principle, any number of applications may be built on top of the library, allowing for maximum flexibility.

5.3 End-user interface

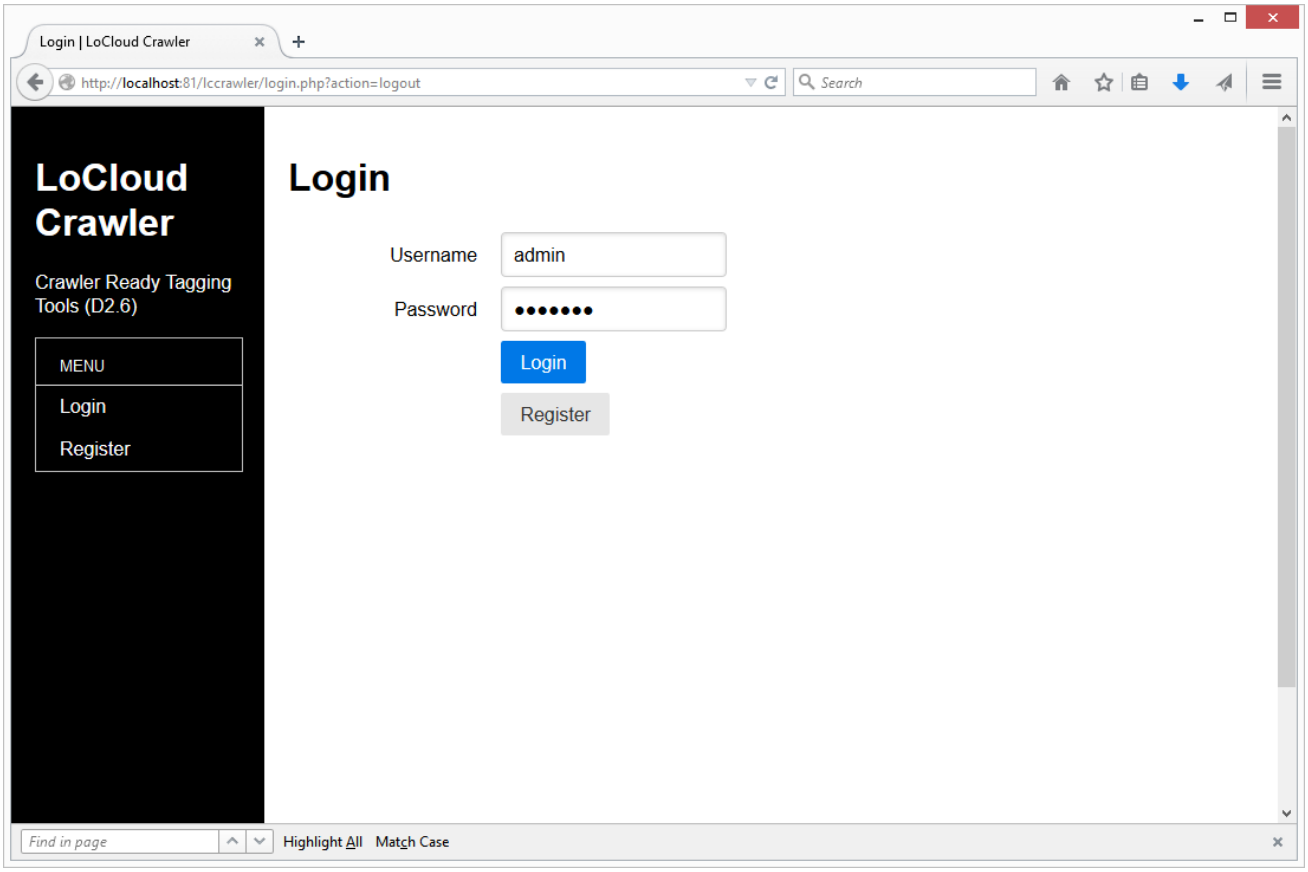
In order to demonstrate the simplified process from the perspective of an end-user, a simple end-user interface has been prepared. Screenshots of the interface are available in the sub-sections below. To try out the crawler, a test-installation is available at the URL http://crtt.avinet.no.

You will need a user name and password to login to the users control panel. The prototype is installed on a shared server and is not prepared for massive traffic. If many users are invoking the application simultaneously, the following performance degrades may be experienced:

- The end-user interface can become less responsive (slower)
- The indexing queue can grow long
- The processing time for crawling and indexing may increase

5.3.1 Login screen (mandatory)

In order to access the CRTT it is necessary to login. In principle, users can register themselves for new accounts. Each user only gets access to his/her own data.



5.3.2 Collection management screen (mandatory)

Before a user can add a URL for crawling or a Google Sitemap XML-file for immediate indexing, he will have to create a collection.

The purpose of creating a collection is that custom metadata extraction rules can be defined for subsets of HTML pages.

There may also be certain metadata that are implicit from the context of the collection but that is not present in the HTML mark-up. Such metadata can be added for all items in a collection at the time of indexing.

In the case of CRTT, this includes geography (a geographical name that will be associated with all items), a default type for digital objects in the collection according to the Europeana code-list and a default license for the content in the collection.

It is important to note that in many cases, licensing may vary from object to object within the same collection. The default licence registered for the collection is the most restrictive license for an individual object in the collection; which is particularly relevant if information is not available about the license of all the objects in that collection.

This might result in “free” works appearing as if they are restricted in the index, on the other hand it will guarantee that “restricted” works are not listed as free. As is evident, this mechanism has drawbacks though it is necessary to implement it in order to provide qualified, albeit inaccurate, EDM records.

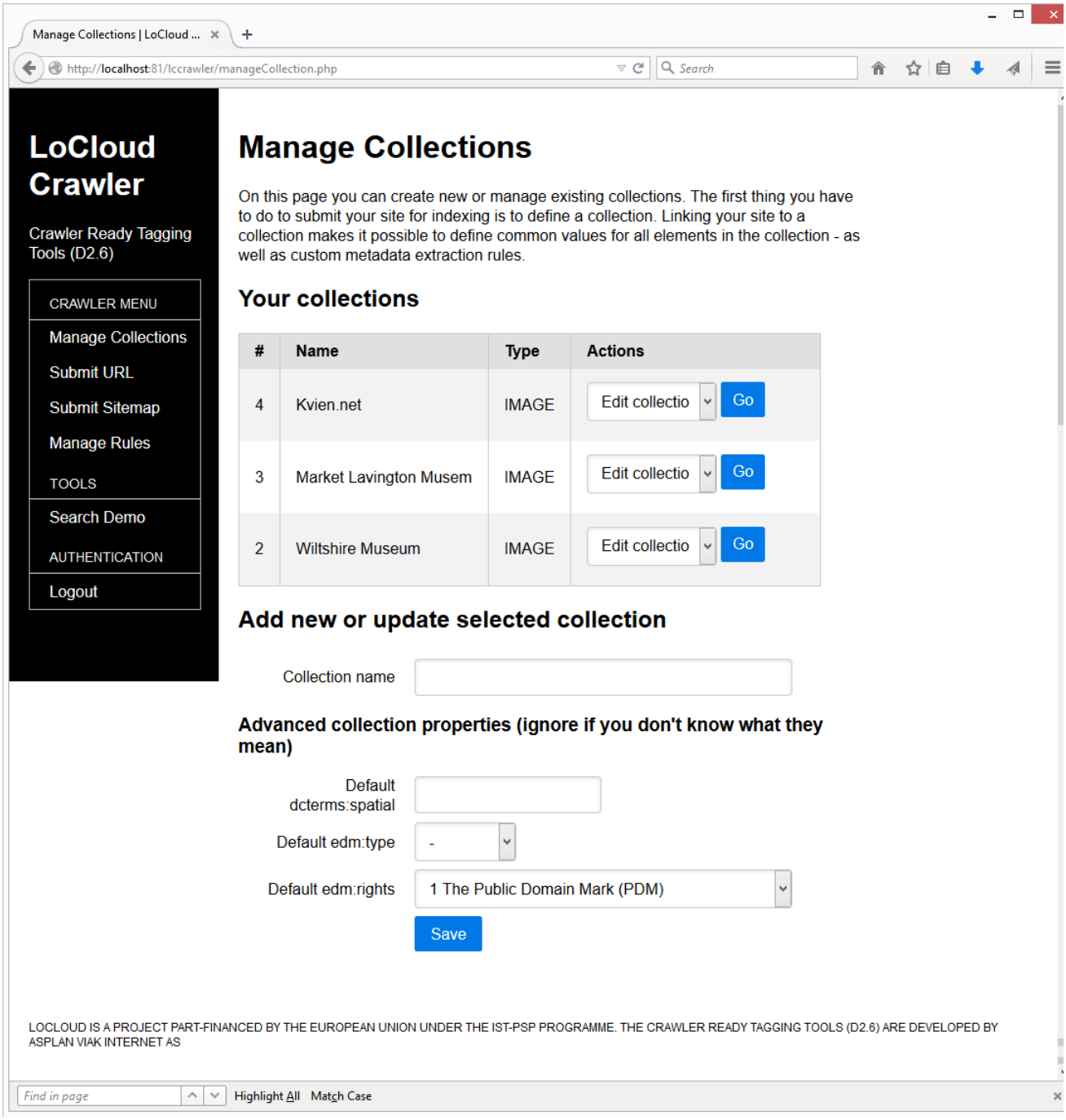


Figure 4: Screenshot of collection management screen

5.3.3 URL submission screen (mandatory)

Having added a collection to the CRTT, the user can then either (1) upload a Google Sitemap XML file containing URLs to deep content or (2) submit a single URL for automatically crawling and extracting relevant URLs from a “front page”.

When submitting an URL, the user can optionally specify two additional properties that will be used during the crawling.

The first is a filter that determines whether the crawler shall follow a URL; the second is a filter that determines whether a URL will be added to the indexing queue once found.

These filters are very simple and merely checks if the filter expression occurs in the URL itself, comparing strings.

The purpose of these filters is to permit the crawler to skip URLs that are not relevant landing pages for digital objects, as web sites will typically include a mix of editorial/news content and actual digital object descriptions.

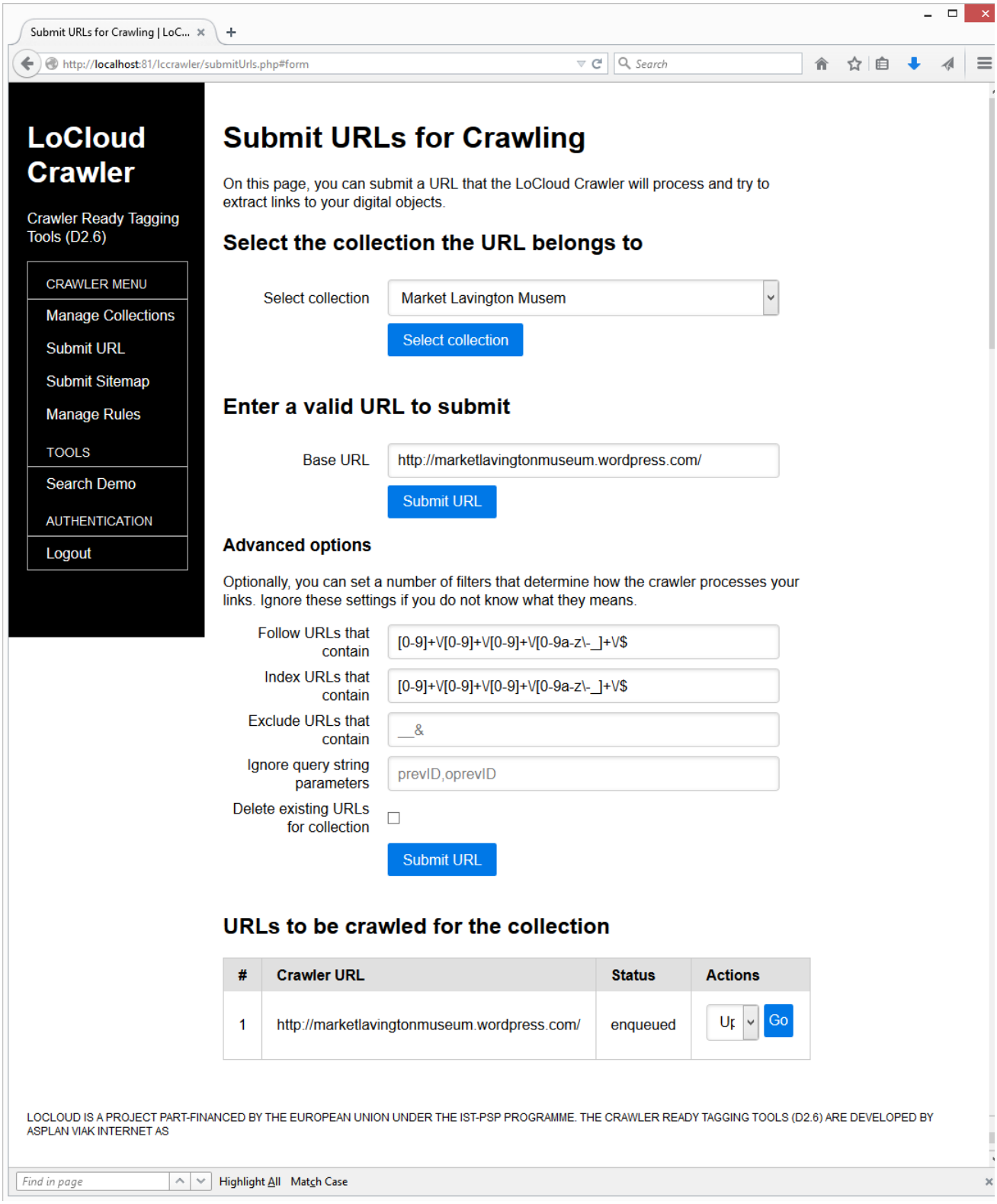


Figure 5: Screenshot of submission screen for URLs to be crawled

5.3.4 Custom rule management screen (optional step)

If the HTML mark-up of the pages the user is trying to index is not semantically strong, it is necessary to define custom metadata extraction rules that can identify information based on its DOM location rather than from its preferred semantic HTML elements.

The form requires the user to select a collection that the rule is valid for, specify which metadata element the value should be mapped to and finally specify the expression to be used to extract info from the DOM.

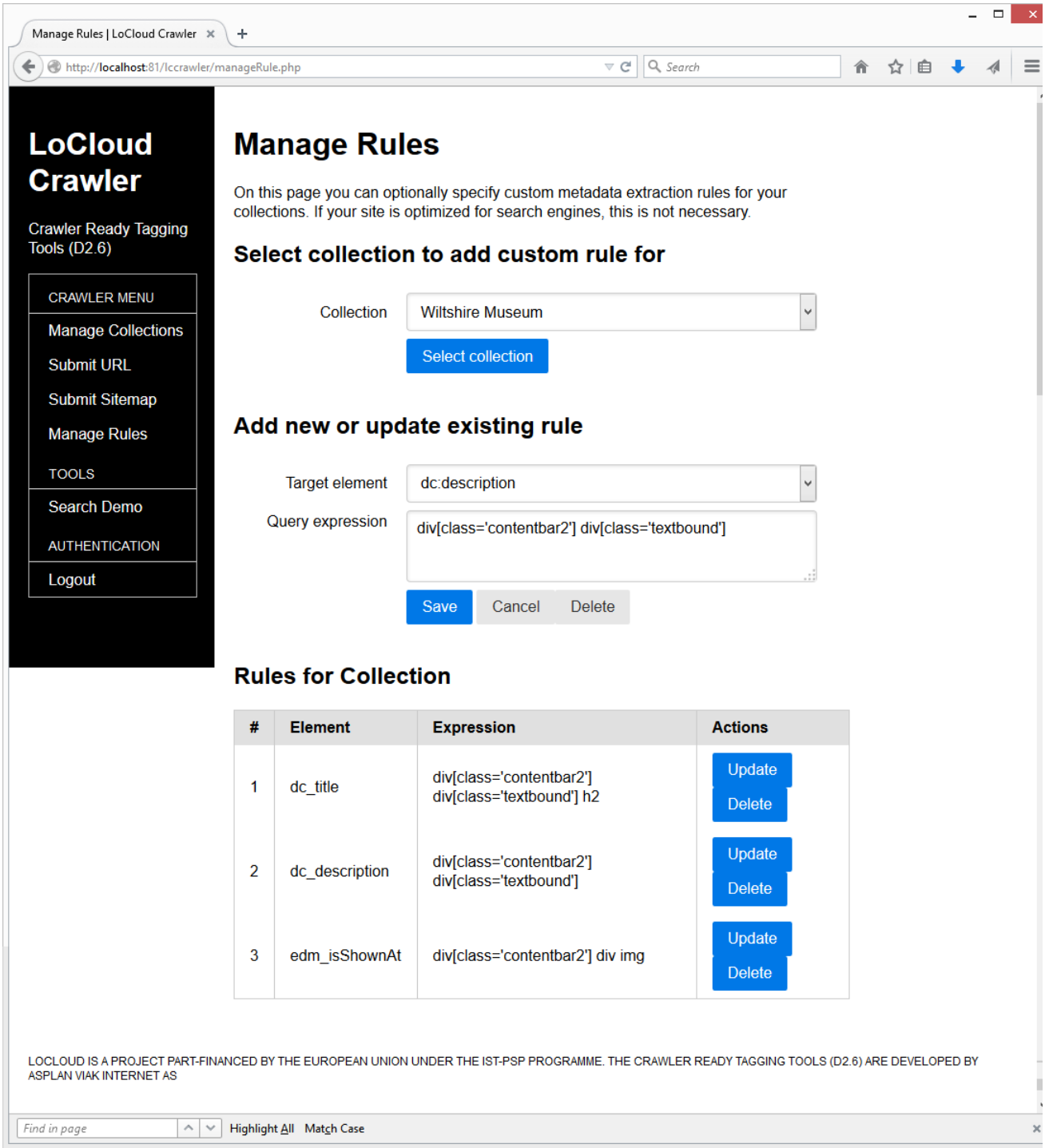


Figure 6: Screen shot of screen to add custom metadata extraction rules for collections

5.3.5 Full-text search screen (optional tool for verifying results)

In order to verify the result of the indexing, the application includes a simple search interface that permits users to search the auto-aggregated digital objects to see how they appear in search results.

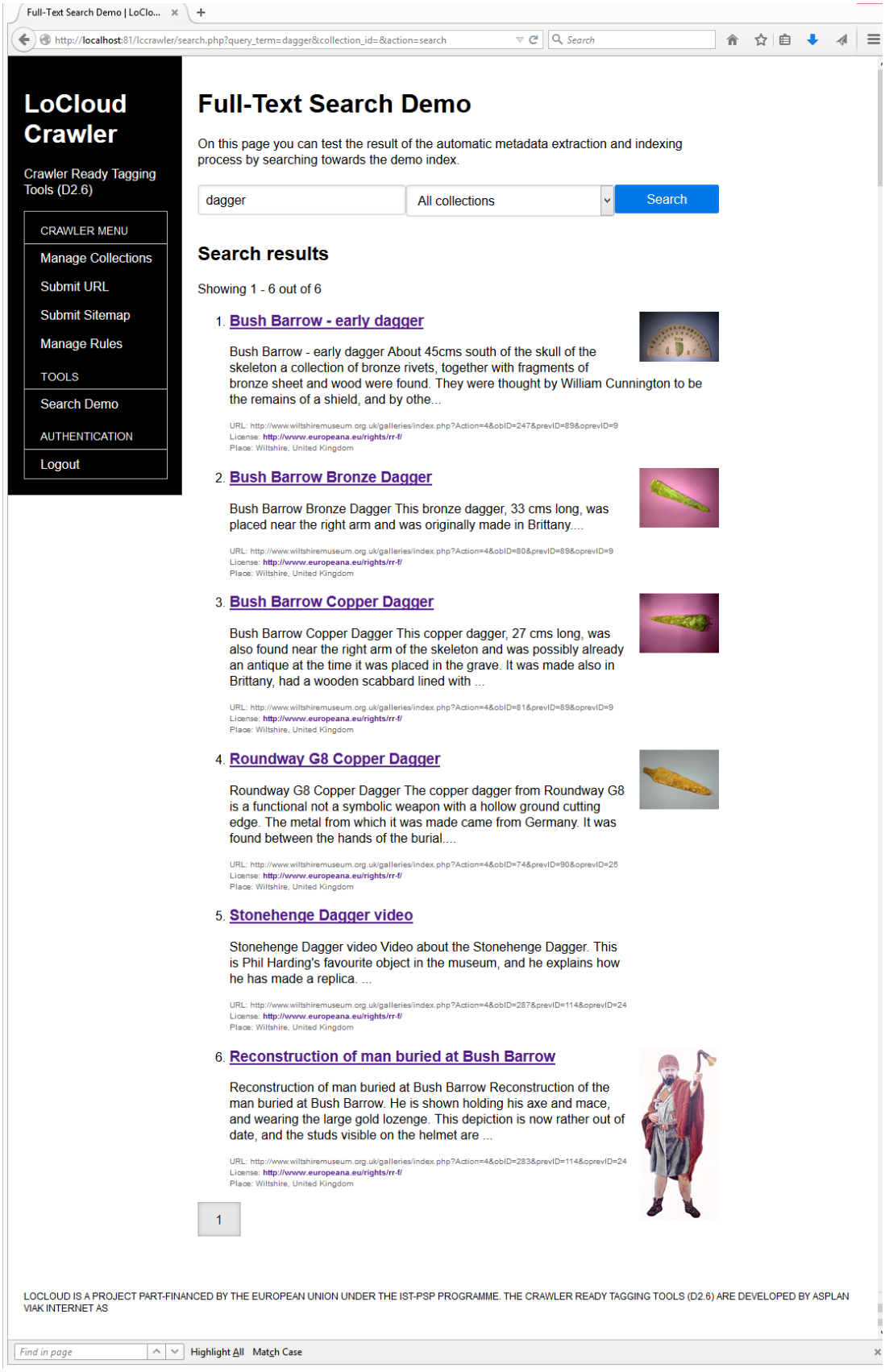


Figure 7: Screenshot of screen to test full-text searching towards the auto-aggregated index.

5.4 Scheduled crawling and downloading task

While the end-users do all their work in the end-user interface, the crawling and indexing action is asynchronous. The crawling of a single base URL can take a long time, hours if a site is large; every HTML document that is linked will have to be downloaded in its entirety, parsed and examined.

It is highly impractical for such a task to be executed synchronously, and so all the URLs submitted by the user to the CRTT are added to a queue from where a background task picks them up and processes them.

The **crawler** is implemented in PHP and is run as a scheduled task using the `PHPCli` interface for execution. The task is scheduled to run every 30 minutes. The maximum execution time for the script is set to 15 minutes and the application will terminate upon reaching this time.

Furthermore, the application is set to terminate whenever a 5MB download limit is reached. It will then resume where it left off the next time it runs.

The crawler conducts the following tasks:

1. It reads a URLs
2. It downloads the document referenced by the URL in its entirety
3. It extracts all the URLs on the page
4. It evaluates the URL against the add to index filter
5. It optionally adds the URL to the indexing queue
6. It evaluates the URLs against the follow filter
7. It optionally loops back to step two for each URL to be followed

Sometimes the process fails in step 2, i.e. it is not possible to download the content of the HTML document referenced by the URL. In such cases, the URL is added to the queue – but with a status of “enqueued”.

Later, it may be picked up by the **downloader** task that loops through all enqueued, non-downloaded URLs and tries to download them. If HTTP errors occur, the status of the URL is updated to reflect the issue at hand.

5.5 Scheduled indexing task

Similar as for the crawler, the **indexer** is implemented using PHP and is scheduled to run every 30 minutes via the PHPCli interface for execution of PHP applications on the command line.

The resource limits enforced for the indexer task is a maximum execution time of 15 minutes after which it will terminate. It will also process a maximum of 100 URLs per run.

The indexer conducts the following tasks

1. It selects the first collection that has non-indexed URLs in the queue

2. It loads custom extraction rules for the collection to be employed in addition to the generic ones that are preconfigured.
3. It selects up to 100 non-indexer URLs from the indexing queue for the collection.
4. For each URL it loads the HTML into a DOM object and queries it using all predefined and custom extraction rules.
5. The results are written into an index table that stores metadata elements as separate fields as well as all text contained within the HTML mark-up as a separate full-text index field.
6. It updates the status of the URL in the indexing queue to be indexed.

In addition to asynchronous batch indexing, the indexer can also handle synchronous indexing of a single URL. The purpose of this function is to be able to re-index single documents efficiently to verify the impact of altered custom metadata extraction rules.

5.6 Concept for integration with MORE

Within the scope of LoCloud, the CRTT will be integrated with MORe for testing purposes. This integration will use the existing APIs exposed by MORe to submit data from CRTT.

Likely, the integration will be based on a user manually “pushing” data from CRTT to MORe on a per collection basis by pressing a button in the user interface.

A key issue that must be addressed when moving metadata from CRTT to MORe is how to identify and approve content providers across the systems. It is likely that CRTT will adapt to MORe as the latter is a bigger, more mature product with many sophisticated solutions already.

6 Target metadata profile

The experiment of the CRTT sought to test whether the search-engine approach to index aggregation could offer any lessons learned to the Europeana community. As such, the obvious choice of target metadata profile was the current version of the Europeana Data Model (EDM) that is currently the preferred ingestion format for the Europeana index.

The Europeana Data Model as a whole is a very comprehensive specification, but reducing it to its simplest valid form, it is in fact quite pragmatic.

The UML class diagram shown in Figure 8 shows the smallest permissible subset of information that must be included in order to represent a digital object as valid EDM. It consists of the three core classes ore:Aggregation, edm:ProvidedCHO and edm:WebResource. Each class has a very limited set of mandatory properties – and a number of conditionally mandatory properties – all of which have been included in the diagram.

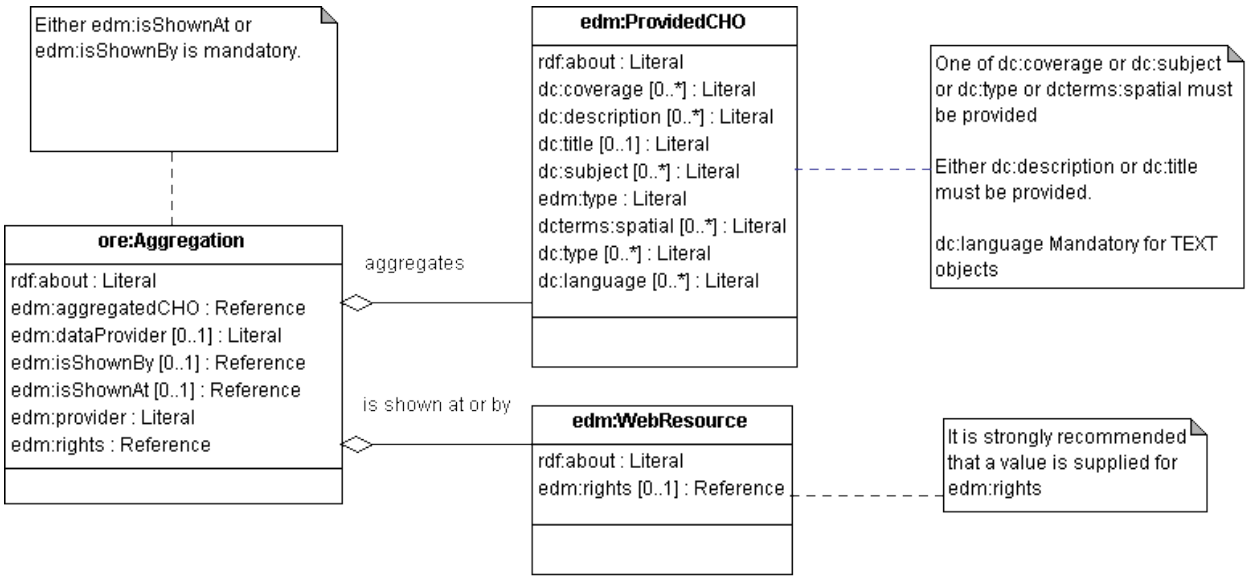


Figure 8: The smallest subset of elements a digital object can be described by and still conform to the EDM

Some of these properties can be populated by extraction of information from HTML mark-up, some of them can be determined by context and given as default values for all objects in a collection.

Since we are interested in providing a simple demo search mechanism, it is helpful for us to represent this as simple relational database table. This allows the search to be implemented using MySQLs full text indexing mechanism to query the auto-extracted content.

The simplest possible subset of data that would have to be stored in order to represent a complete EDM object, expressed as a flat table is as shown in Table 3.

Table 3: Mapping of smallest valid subset of EDM to a relational table for easy indexing in an RDBMS

#	Source EDM elements	Target table field	Comment
1	ore:Aggregation/rdf:about edm:ProvidedCHO/rdf:about	url	
2	dc:subject	dc_subject	
3	edm:ProvidedCHO > dc:language	dc_language	
4	edm:isShownBy edm:WebResource/rdf:about	edm_isshownby	Assuming one web view, e.g. one photo, per indexed resource.
5	edm:provider edm:dataProvider	edm_provider	
6	edm:rights	edm_rights	
7	dc:coverage dc:subject dcterms:spatial dc:type	dc_subject	Only one is mandatory, dc:subject is the easiest to extract automatically
8	dc:title	dc_title	
9	dc:description	dc_description	Not mandatory once dc_title is used – but useful for presentation and relatively easy to extract
10	edm:type	edm_type	Impossible to guess exactly, requires default value for collection as supplement

This has been the data structure that has been observed during the implementation and testing of the CRTT. It does not seek to exploit the full potential of the EDM – but it reflects the completeness of many resources that currently exist in the Europeana index. For this reason, we have assumed it suitable as a basis for validation of the CRTT approach.

7 HTML, embedded semantics and tag extraction

This chapter describes how to author tag extraction rules using the CRTT tool. These are the key element of the CRTT and are required to translate the text elements of a HTML document into their corresponding EDM elements.

7.1 The structure of a HTML document

Everything you see in a web browser, unless you are using some specialized plugin, is the result of rendering of a mark-up language called HTML – Hyper-Text Mark-up Language. HTML can be a complex thing to understand but for using the CRTT tool, it is sufficient to know the following:

A HTML document consists of elements and attributes.

```
<element attribute="value">Inner text</element>
```

An element is enclosed in angular brackets:

```
<element/>
```

An element can be specified with a starting and an ending tag, note that the ending tag starts with a slash after the opening angular-bracket.

```
<element></element>
```

An element can also be specified as a single tag, note that the tag ends with a slash before the closing angular bracket. This is used when the element carries an attribute but no value.

```
<element/>
```

An attribute is a key-value pair that is specified within the brackets of an element; note that the value is enclosed in double quotes.

```
<element attribute="value"/>
```

The names of elements have special meanings that may or may not be observed by the HTML page that is being processed. Tags named “h1”, “h2”, “h3” signifies headlines of different levels. The tag named “p” signifies paragraph text. The tag named “img” signifies an image media object. The tag named “a” signify an HTML anchor, a link to a related or otherwise relevant web resource. The tag “title” signifies the name of the page, as it will show up in search engines and in the title of bookmarks and browser windows.

These semantics are sometimes abused, sometimes observed correctly. Rules will have to be authored to suit individual sources in order to achieve a high quality match towards the corresponding EDM elements.

7.2 Semantic elements in HTML

HTML is a very flexible mark-up language that is very forgiving in terms of syntactical errors and oversights on the part of the document structure. At the same time, the language defines elements that carry specific semantic meaning.

Table 4: Common semantic HTML elements and their meanings

Element	Meaning
<title/>	The title of the HTML page – will be used as title by search engines
<h1/>…<h^n/>	Headlines (title, important subjects etc.)
<p/>	Paragraph text
<meta name="Description"/>	Description of site (not rendered, present in mark-up)
<meta name="Keywords"/>	Keywords describing site (not rendered, present in mark-up)
<section/>, <article/>	HTML 5 element to enable interpretation of what HTML mark-up represents – e.g. a content section or an article (multiple per page)
<header/>, <footer/>, <aside/>	HTML 5 element to enable interpretation of what HTML mark-up represents – e.g. a logo and navigation structure that can be ignored when extracting info about digital objects.

7.3 Search Engine Optimization - enhancing HTML mark-up

Enhancements to the HTML that will improve the quality of auto-extracted metadata from web documents are beneficial both to the experimental LoCloud CRTT tools and for improved discoverability through the major search provider's indices, i.e. Google, Bing, Yahoo! and Yandex etc.

Search engine optimization is a wide professional area in itself and is documented through countless books, reports and other publications[1]. The important thing to be aware of in the context of the CRTT is that the better a site is optimized for search engines, the better it will be suited for auto-extraction of qualified EDM metadata.

7.4 Examples of enhanced mark-up and extraction rule syntax

Rules for extracting metadata from HTML can be expressed using a simple query language towards a HTML document represented as a Document Object Model (DOM). The language inherits from CSS selectors as well as XPath and is a comprehensive and powerful language. It is however not necessary to understand all of it to simply use the CRTT tool. The examples in this section provide a starting point for users who would like to experiment with developing their own custom rules.

[1] An introduction to SEO for beginners by Google: http://googlewebmastercentral.blogspot.be/2008/11/googles-seo-starter-guide.html

7.4.1 Generic tag extraction rule example

A valid rule to capture level one headings looks like this:

```
h1
```

If only level one headings with the class attribute set to “heading” should be extracted, the following expression can be used

```
h1[class='heading']
```

If only level one headings with the id attribute set to “h1_1” should be extracted, the following expression is valid

```
h1[id='h1_1']
```

7.4.2 RDFa tag extraction rule example

RDFa allows the encoding of complete metadata tags directly into a HTML page by adding relevant attributes to the HTML elements where the data are shown.

```
<a name="myobject" />
   <div about="#myobject">
      <table>
         <tr>
         <td>Title: </td><td><span property=”dc:title”>My Object</span></td>
         <td>Creator: </td><td><span property=”dc:creator”>AVINET</span></td>
         <td>Description: </td><td><span property=”dc:description”>An
            object of great significance</span></td>
         </tr>
      </table>
   </div>
</div>
```

Figure 9: RDFa embedded semantics fragment

To extract the element `dc:title` from the above RDFa fragment, it is possible to use the following tag extraction expression.

```
span[property='dc:title']
```

7.4.3 Schema.org rule extraction rule example

The majority of cultural heritage sites are generated from structured data from a database. When this data is formatted into HTML, it may be difficult to recover the original structured data. Hence, Europeana’s approach of ingesting constructed metadata rather than auto-indexing of remote

resources. However, out of concern to scalability and speed, Search Engines cannot afford this complexity of process and so they must bridge the gap in a different manner.

A group of the major players in search have come together and agreed on a set of schemas that can be used to mark-up the content of web pages in such a manner as to permit the search engines to interpret the semantic meaning of the structured data. According to Schema.org: "*On-page mark-up enables search engines to understand the information on web pages and provide richer search results in order to make it easier for users to find relevant information on the web. Mark-up can also enable new tools and applications that make use of the structure*". This is the extension point to which we attach the CRTT tool.

```
<div itemscope itemtype ="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director" itemscope itemtype="http://schema.org/Person">
  Director:    <span    itemprop="name">James    Cameron</span>    (born    <span
itemprop="birthDate">August 16, 1954)</span>
  </div>
  <span itemprop="genre">Science fiction</span>
  <a                              href="../movies/avatar-theatrical-trailer.html"
itemprop="trailer">Trailer</a>
</div>
```

Figure 10: Schema.org embedded semantics fragment

To extract the name of the director, the following metadata extraction rule would suffice:

```
div[itemprop='director'] span[itemprop='name']
```

7.4.4 Microformats extraction rule example

Microformats are a way of using HTML to mark up people, organizations, events, locations, blog posts, products, reviews, resumes, recipes etc. Sites use microformats to publish a standard API that is consumed and used by search engines, browsers, and other sites. The CRTT is capable of extracting content from micro formats by using the rules described below.

Rel-license microformat

This micro format may be used to express the license restrictions of the content, including those specified by Europeana.

```
<a href="http://creativecommons.org/licenses/by/2.0/" rel="license">cc by 2.0</a>
<a href="http://www.apache.org/licenses/LICENSE-2.0" rel="license">Apache 2.0</a>
```

Figure 11: rel-license microformat fragment

To extract the license information from a HTML document, the following tag extraction expression may be used:

```
a[rel='license']
```

This will select the element and it is then possible to extract both the attribute and the inner text of the element.

Rel-tag microformat

This micro format may be used to express relationships to controlled vocabularies, in this case the ontology of Wikipedia.

```
<a href=" http://en.wikipedia.org/wiki/Technology" rel="tag">tech</a>
```

Figure 12: rel-tag microformat fragment

The rel-tag microformat can be extracted using the following tag-extraction expressions:

```
a[rel='tag']
```

This will select the element and it is then possible to extract both the attribute and the inner text of the element.

Geo microformat

This micro format may be used to express coordinates

```
<div class="geo">GEO:
 <span class="latitude">37.386013</span>,
 <span class="longitude">-122.082932</span>
</div>
```

Figure 13: geo microformat fragment

Ethe geo microformat can be extracted using the following tag extraction expression:

```
div[class='geo'] span[class='longitude']
```

At the moment, as mentioned under limitations, the rules are only capable of mapping one target element to one source element – as such coordinates can be extracted to two X/longitude, Y/latitude fields – but cannot at present be combined into a geo URI or other fused representation.

7.5 Predefined extraction rules

A HTML document allows content to be represented in any number of ways. HTML does not enforce the use of its built-in semantics and many web designers exercise the poor practice of using HTML and CSS exclusively as a visual formatting engine – disregarding its semantic structure.

Therefore, a generic set of metadata extraction rules will not yield consistently good results for any web site. However, it is necessary to provide a baseline that exempts non-technical content providers from having to learn XPath and DOM to submit their URLs for crawling and indexing.

To this end, we must consider elements that are implicit - or that we can 'guesstimate' the meaning of by analysing the HTML elements.
The system is preconfigured with a set of generic extraction rules that tries to map HTML elements with a semantic meaning to corresponding elements in the target metadata profile.

Table 5: Predefined generic extraction rules

Target metadata element	Source DOM identifier	Comment
`dc:title`	html head title h1 property=”dc:title”	HTML specifies several elements that can be used to determine or guess the title of a digital object. The most important being the <title/> tag that determines the text used in the title bar of a web browser or as the name when you add a bookmark. In the absence of these elements, the title may be guessed from level one headings <h1/> in the HTML. If semantics in the form of RDFa is applied, titles may also be extracted from HTML elements where the property attribute is set to dc:title. In the latter case, the interpretation of the element is unambiguous, but puts greater demands on the source mark-up.
`dc:description`	meta[name=’description’] *[property=’dc:description’] h2 h3 p	Descriptions do not have a dedicated HTML element as title does. HTML does however provide a <meta name=”description”/> tag for providing a summary of the content on the web page. This tag is targeted at spiders and is not visually displayed when the HTML page is rendered. Second, the indexer will look for RDFa elements with the property attribute is set to dc:description.

		In the absence of these tags, a description may be guessed from the text contained on the page including level two and three headings as well as paragraph text. The N first words will be extracted with an upper limit of 250 characters.
`dc:subject`	meta[name='keywords']	
`edm:rights`	Defaults to value specified for collection	A drop-down with the available license types that are permitted by Europeana is available in the form where the collection is registered.
`edm:type`	Defaults to value specified for collection	

8 Validation and testing

The CRTT has been validated against three real-world cultural heritage sites that all publish HTML documents to the World Wide Web.

The three sites that have been used for testing are:

- Wiltshire Museum gallery site (United Kingdom)
- Market Lavington Museum (United Kingdom)
- Kvien.net, private collection (Norway)

The selection of sites does not carry any specific significance beyond that they should be typical web sites with typical strengths and weaknesses – they should not be exceptionally well-structured or "rich" resources. Additionally, for the sake of the experiment, the sites should not be based on content that is already in Europeana.

8.1 Detailed walk-through of test cases

All three test cases followed the same approach and in order to help the reader follow the auto-extraction process step-by-step, the first test case, Wiltshire Museum's gallery site, is described from start to finish. The approach is identical for the two other test-sites and a summary, presenting key indicators for all three, is included in section 8.1.1 below.

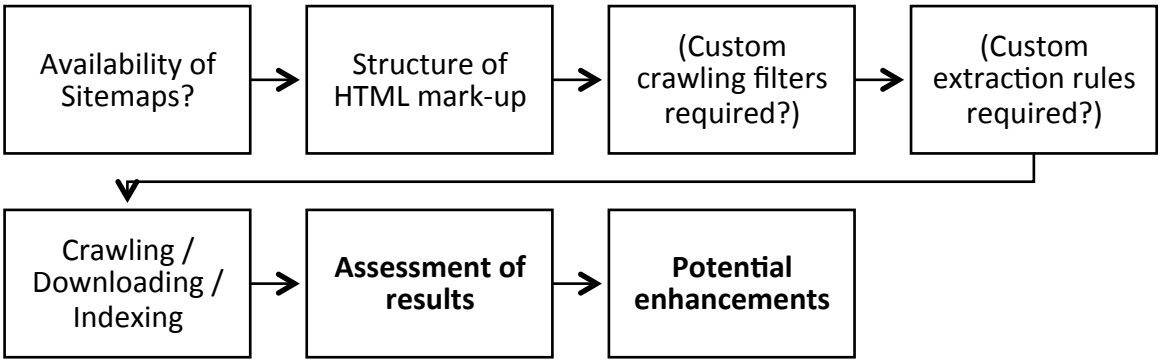


Figure 14: The activities of the test cases, steps that are optional are shown in parenthesis

8.1.1 Availability of URLs

None of the three sites had URLs available as a Google Sitemap file. It would be easy to prepare such a file – but it was not available for use in the test case.

Table 6: Available URLs for the three test cases

Website	URL	Extraction method
Wiltshire Museum	http://www.wiltshiremuseum.org.uk/galleries/	Crawling
Market Lavington Museum	http://marketlavingtonmuseum.wordpress.com/	Crawling
Kvien.net	http://www.kvien.net/Gamle/album/index.html	Crawling

While it would have been convenient to have a Sitemap, not having one posed an interesting challenge in trying to determine which crawled URLs points to landing pages for digital objects – and which points to irrelevant contextual material such as news items, contact pages etc.

8.1.2 Structure of website HTML mark-up

The **Wiltshire Museum galleries** site contains, as the name aptly suggests, a set of galleries. The galleries are categorized by topic and are linked from the front page as thumbnails.

One click further, all digital objects in the galleries are shown as thumbnails, and on the third level, we find the landing pages of the actual digital objects. The site contains different types of content, but for the purpose of the test, we have targeted extraction of digital image objects with associated metadata.

On each page, there is a lot of contextual information such as header, footers, design elements and navigation structures. These must be filtered by providing the crawler with a number of filters. A common presentation template is used for all pages of the same type.

The semantic structure of the site is not exploited to the full potential. For an example, all digital objects have an identical HTML title tag – something that will result in extremely poor performance in searches towards the major search engines.

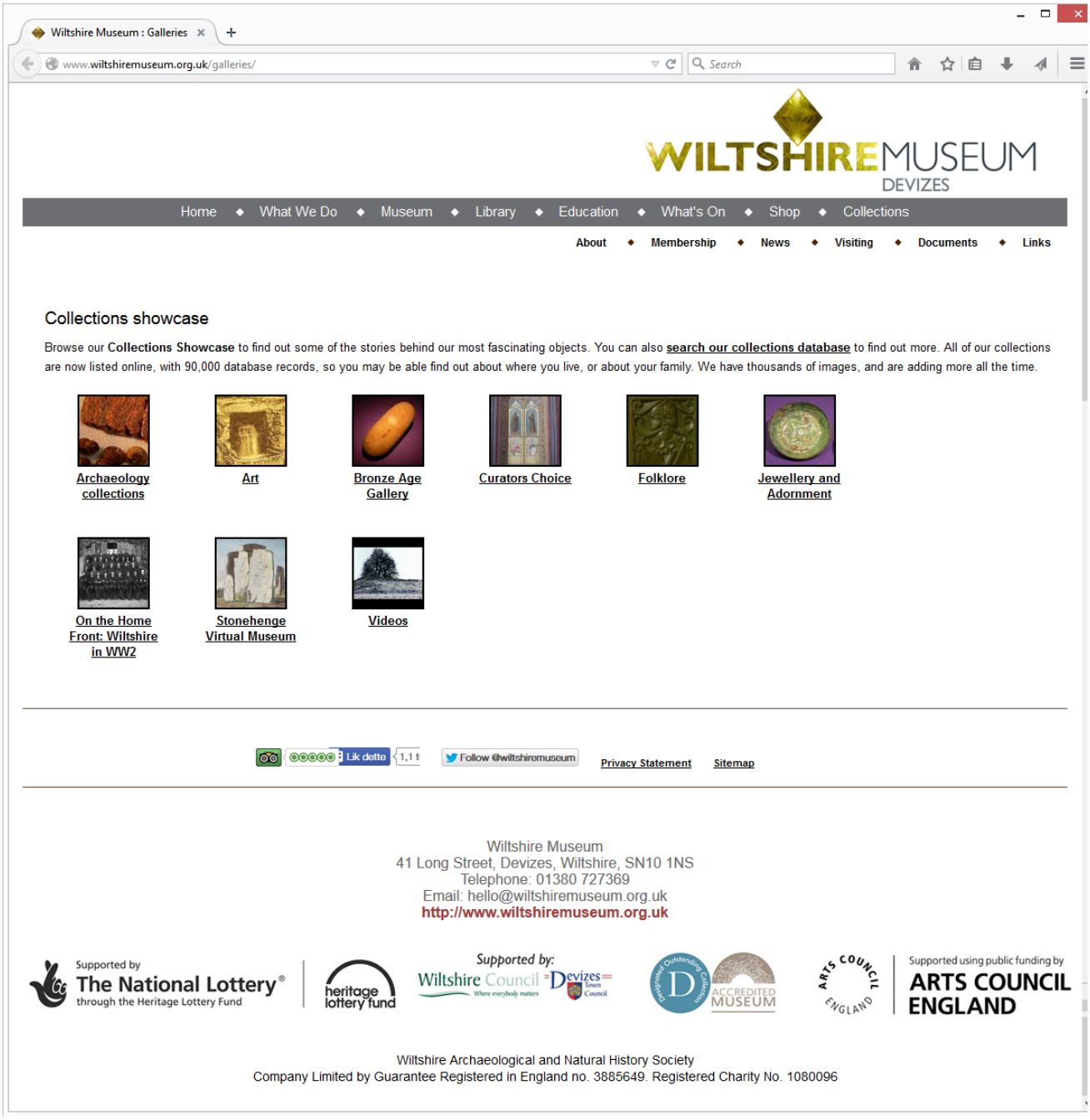


Figure 15: Wiltshire Museum galleries front page

Market Lavington Museum museum blog site is a WordPress blog that contains a mix of contextual information and digital objects. WordPress has ample amounts of plugins that could generate a sitemap in a matter of seconds, but since we have no access to the administrator interface of the site and only the same access as any regular individual browsing the web, crawling is necessary.

No menu permits users to browse content by topic. There are however, other ways to identify links, such as previous/next article links, monthly content archives, last ten articles lists etc. This is deemed sufficient to be able to identify all URLs off the site.

The semantic structure of the HTML mark-up of Market Lavington's site is stronger than that of the other two sites. Meta tags for descriptions and keywords are present on all pages – and the same presentation template is used for all articles, making it relatively easy to extract consistent metadata from the mark-up.

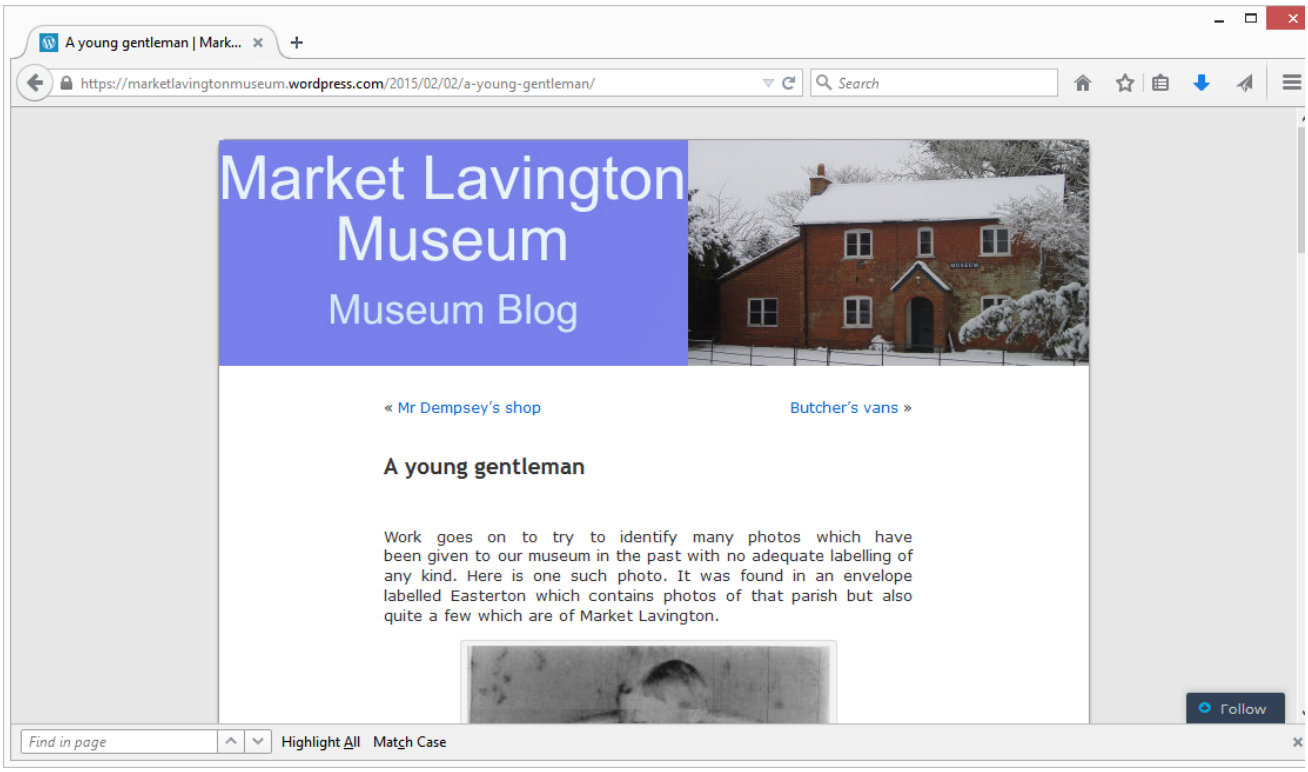


Figure 16: Market Lavington Museum's museum blog web site

The final test case, **Kvien.net – old photos**, is a private photo collection of old post cards that has been scanned and equipped with minimal metadata before it has been published to the Internet using a gallery generator. The collection is shown as a paged gallery with thumbnails linking to presentation pages for individual digital objects.

The semantic structure of the HTML mark-up is poorer than that of Market Lavington – and about on par with that of Wiltshire Museum. The title tag is for an example used for the unique content title – but the value used is not very helpful.

Meta tags exist with descriptions and keywords – but these are not exposed on the page – rendering more information for the search engines than for the end-user. This is however possible to exploit during metadata extraction.

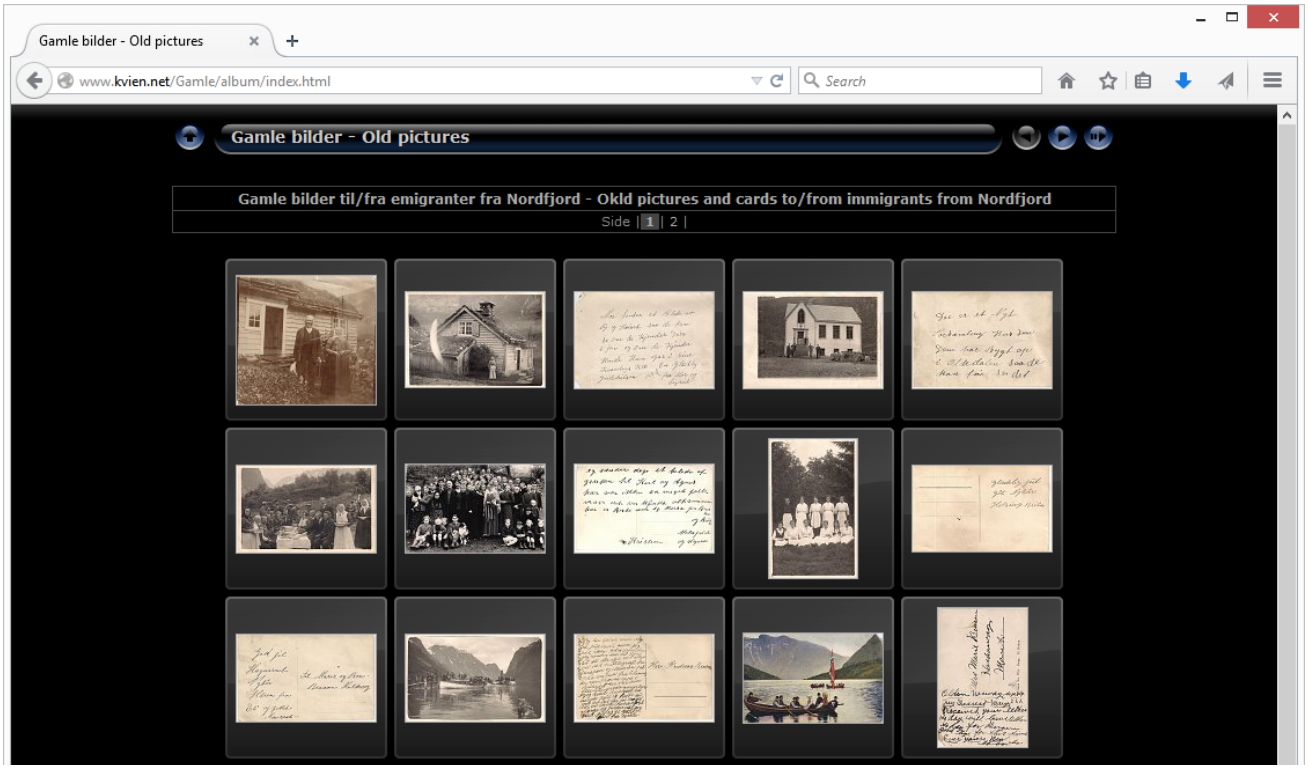


Figure 17: Kvien.net - Old Pictures web site

None of the three sites contains **deep content**. All content on the sites can be found by following from one link to another – no searches are required to reveal content. This is an absolute prerequisite for crawling to be a viable alternative.

8.1.3 Custom crawling filters

In addition to digital object pages, all three sites also include links to pages with contextual information that is not relevant for us. It is therefore necessary to instruct the crawler on how to process the documents as it moves through the site.

The following instructions can been be passed to the crawler:

- A filter for which URLs to follow (`url_follow_filter`)
- A filter for which URLs to index (`url_index_filter`)
- A filter for which URLs NOT to index (`url_exclude_filter`)
- A filter that permits removal of irrelevant query string parameters (`url_remove_params`)

The table below explains which filters have been set for each of the test cases and why.

Table 7: Crawler settings used for the three test cases

Site	Filter	Expression
Wiltshire Museum	The site is a classical database driven structure that relies on query string parameters to load a specific page. That means that the URL, up to and	

<table>
<tr><td></td><td colspan="2">including the path is common to all items and that they must be distinguished by the query string parameters.

All pages that contain links to digital objects had the text string “Action=” in the URL – so we set the crawler to follow all such links that include this term. All digital objects (those that we were after, images) has “Action=4” in the URL – so we instructed the crawler to add any URL that includes this term to the indexing queue.

There are two complicating factors relating to the way links are used on the site. Several unique URLs point to the same page, if not handled, this would result in duplicate digital objects.

This case occurred (two URLs for object 160)

…index.php?Action=4__&obID=160&prevID=86&oprevID=25
…index.php?Action=4&obID=160&prevID=86&oprevID=25

In order to mitigate this, we instructed the crawler to exclude URLs with the expression __& in them.

Second, this case occurred (two URLs for object 160):

…index.php?Action=4&obID=160&prevID=85&oprevID=24
…index.php?Action=4&obID=160&prevID=86&oprevID=25

In order to remove this, we specified that the two query-string parameters prevID and oprevID should be removed prior to determining if the URL is unique.</td></tr>
<tr><td></td><td>url_follow_filter</td><td>Action=</td></tr>
<tr><td></td><td>url_index_filter</td><td>Action=4</td></tr>
<tr><td></td><td>url_exclude_filter</td><td>__&</td></tr>
<tr><td></td><td>url_remove_params</td><td>prevID,oprevID</td></tr>
<tr><td>Market Lavington</td><td colspan="2">This web site used one of the standard URL schemes of WordPress to identify articles. A typical URL looks like the one shown below:

https://[...]/2010/04/22/market-gardening-at-fiddington/

However, also here, each URL occurred with a multitude of different parameters, rendering several unique URLs for the same digital objects.

We therefore set a filter that only URLs that have the following structure /yyyy/mm/dd/name-of-article/ should be followed – and indexed. This is specified using a Regular Expression (RegEx) pattern as shown below.</td></tr>
</table>

	Once this had been set, there was no need for further filtering.	
	url_follow_filter	[0-9]+\/[0-9]+\/[0-9]+\/[0-9a-z\-_]+\/$
	url_index_filter	[0-9]+\/[0-9]+\/[0-9]+\/[0-9a-z\-_]+\/$
	url_exclude_filter	
	url_remove_params	
Kvien.net	The URL for a typical object page looks like this `http://www.kvien.net/Gamle/album/slides/scan0014a.html` We therefore instructed the crawler to add all URLs that contain the expression “slides” to the indexing queue. Upon crawling, a number of duplicate URLs were returned suffixing the term “show_param” to the URL. We therefore instructed the crawler to exclude URLs that included this term.	
	url_follow_filter	
	url_index_filter	slides
	url_exclude_filter	show_param
	url_remove_params	

8.1.4 Custom extraction rules

None of the sites got very good indexing results using only the basic set of built-in rules. Instead, it was necessary to build a set of simple expressions that mapped site-specific mark-up to target metadata elements as shown in Table 8.

At **Wiltshire Museum** galleries site, the web designer has not exploited the semantics of the HTML elements. HTML in conjunction with CSS has been used primarily as a visual layout engine.

The <title/> tag has been used for a static string that is the same for all objects on the site, namely “Wiltshire Museum – galleries”. In a search engine, all digital objects would therefore appear with this title. Due to the multiple occurrences of objects with the same title, results will be given lower priority in search results of major search engines.

The title of digital objects is however described in a <h2> heading element that is located at a fixed spot in the HTML mark-up across all digital object presentation pages. In order to extract dc:description metadata, it was necessary to use a similar rule as for headings, as the text was contained within the same parent HTML element as the title of the digital object.

At **Market Lavington Museum**, both dc:title and dc:description were filled well automatically, all what was needed was to add a rule to extract images.

For **Kvien.net**, it was necessary to add custom rules for extraction of the title as well as the image. Descriptions and subjects were extracted from meta-tags. The main problem with this site is that the content, to begin with, is not very rich – but typical for local cultural heritage sites from small institutions/private collections/individuals.

Table 8: Custom metadata extraction rules

Test case	Target metadata element	Custom extraction rule
Wiltshire Museum	`dc:title`	div[class='contentbar2'] div[class='textbound'] h2
	`dc:description`	div[class='contentbar2'] div[class='textbound']
	`edm:isshownby`	div[class='contentbar2'] div img
Market Lavington Museum	`edm:isshownby`	div[class='entry'] img[class='size-full']
Kvien.net	`dc:title`	span[class='comment']
	`edm:isshownby`	img[class='slideImage']

8.1.5 Crawling / downloading / indexing

These processes are transparent to the end-user but are included here in order to describe how the process was improved throughout the testing.

The crawling exploits the crawling filters described in section 8.1.3 above. It attempts to read the base URL that is supplied for crawling, extract all links inside that page and recursively follow them until there are no more links.

The crawler has been instructed to stay on-site. I.e. it will not stray to other domain names or subdomains.

The crawler reads the URL, downloads the content and stores it in a queue table. Sometimes, the URL cannot be downloaded – in these cases; the URL gets a specific status code that is picked up by the scheduled downloader task.

The downloader checks for crawled, not-downloaded links and downloads them. If they are unreachable or have errors – their status is updated.

An important issue during the download is that content can originate from a multitude of different source encodings. CRTT downloads the content and tries to convert it to UTF-8 before storing it in the database cache.

The last part of this process is the indexer. This task picks up URLs with the status “downloaded” and loops through them, applying all the rules defined for the collection they belong to.

8.2 Assessments of results

The results from the indexing of the test case web sites show that it, with moderate effort, is possible to extract structured metadata from heterogeneous HTML documents.

8.2.1 Completeness

To make qualified EDM is a more complex matter than populating some free-text fields in a permissive metadata profile. In terms of the minimum permissible combination of `dc:title`, `dc:description` etc., it is easy to populate an EDM metadata object with meaningful values. However, some Europeana specific tags are impossible to determine automatically as they require qualitative assessments on the part of the contributor. This goes for the two mandatory elements: `edm:type` and `edm:rights`.

The `edm:type` element determines wheter a digital object is an image, a text, a sound, a video or a 3d object. This can only be correctly interpreted by the context. Luckily, many collections are focussed on single content types. Because of this, we have specified a default value for all objects belonging to the same collection for the test cases. All objects generated for the collection will get this as its default value.

If the `edm:type` for a collection is set to “TEXT” but a custom metadata extraction rule for `edm:isShownBy` points to an HTML <img/> tag, the `edm:type` will automatically be changed to “IMAGE”.

The `edm:rights` element is not present in the HTML markup of any of the test case web sites. The most common expression of licensing is a string containing the copyright sign (C) or © but without specifying detailed terms and conditions. This cannot automatically be translated to one of the 13 permitted licenses for content that is going to be ingested into Europeana. It could however be put in the non-mandatory `dc:rights` EDM-element.

With regard to licensing, we assumed, perhaps too liberally, that a default value can be assigned to all objects belonging to the same collection. As such, users are able to specify a default license at the time of creating or editing a collection.

The same approach as for `edm:type` and `edm:rights` has also been used in a third context; namely to assign a default `dcterms:spatial` value to all objects in a collection. This value will only be filled in of no custom rule is specified that populates it using other data. It is possible that this value should be left blank as the geographic accuracy in many cases would be too coarse to give any meaning, perhaps except when visualizing on a map of Europe.

Table 9 shows the completeness of the auto-extracted metadata as a counting of how many of the mandatory EDM elements are present for the objects from each of the three test case web sites.

Table 10 shows how the digital objects are distributed across `edm:type` values. As is evident, the target of the test cases has been to extract objects that has both text AND images in order to enable attractive search interfaces.

Table 9: Test case metadata completeness, where default values for collection are used it is shown in red

Metadata element	Wiltshire Museum – galleries	Market Lavington Museum - museum blog	Kvien.net - private collection of old photos	Sum
dc:coverage	0	0	0	0
dc:description	158	100	23	281
dc:subject	0	0	32	32
dc:title	158	100	32	290
dc:type	0	0	0	0
dcterms:spatial	158	100	32	290
edm:type	158	100	32	290
edm:isShownAt	158	100	32	290
edm:isShownBy	158	94	32	284
edm:rights	158	100	32	290
edm:aggregatedCHO	-	-	-	-
edm:dataProvider	158	100	32	290
edm:provider	-	-	-	-
Total objects	158	100	32	290

Table 10: Test case metadata distribution across types of EDM objects

Object type	Wiltshire Museum – Galleries	Market Lavington Museum - museum blog	Kvien.net - private collection of old photos	Sum
IMAGE	158	94	32	284
TEXT	0	6	0	6
SOUND	0	0	0	0
VIDEO	0	0	0	0
Total	158	100	32	290

8.2.2 Coverage

For a site that does not provide a pre-emptive listing of all its content, it is impossible to determine of the crawler has picked up all the links to relevant objects – or if it has reached a dead end out of configuration or network errors.

As such, it is impossible to say whether the coverage of the auto-extracted metadata is good. It is notable that in the gallery provided by Kvien.net there are 36 thumbnails – and yet only 32 objects result from the crawling. This could be due to too restrictive filters – or down to simple Http communication errors at the time of crawling.

8.2.3 Accuracy

While the extracted metadata objects satisfies the minimum criteria for an EDM object and thus are permissible for ingestion into Europeana. The accuracy of the data in the test cases cannot reach the potential accuracy level of a manually, purpose-made EDM object exported from a collection management system according to a professional mapping.

It is however important to note that the purpose of this experiment is to validate the suitability of this alternate approach to aggregating a search index – not to produce polished EDM.

8.3 Potential enhancements

This section identifies a number of potential enhancements to the CRTT tool that may be applied after its initial release.

8.3.1 Revise the user interface through Interaction Design

The user interface is functional – and the minimum set of information that is required from an end-user to start using the CRTT is very limited.

Yet, the look and feel aged and boring – and the advanced options are obtrusive to users who may not understand what they are for and who should be allowed to ignore them.

An interaction designer should, taking this report as a starting point, go through the user interface and realign the elements with the end-user workflow in mind at all times.

8.3.2 Video, Sound and 3D object recognition and extraction

Presently, the CRTT only extracts the src property values from HTML <img/> tags for populating the edm:isShownBy elements. This practically excludes the other edm:object types who will default to TEXT (or whatever the collection is set to) but will never have anything but the edm:isShownAt element set.

8.3.3 Extended default extraction rules to handle “well-known” semantics

While none of the test case sites had embedded semantics beyond HTML <meta/> tags, there is a great number of common semantic structures that should be supported by default by the CRTT. These include but are not limited to:

- Addresses
- Geographical locations
- Creative Commons license statements
- Sound files
- Video files

8.3.4 Enable combining expressions

Presently, each extraction rule must match exactly one element in the HTML mark-up and map that to exactly one element in the target metadata profile.

It would be interesting to be able to create expressions that selects multiple elements from the HTML mark-up – and concatenates these into a single target value in the metadata profile.

The present approach is technologically a bit naïve, but as demonstrated by the test cases is effective to some degree.

8.3.5 Deploy in dedicated environment

The CRTT is deployed in a shared hosting environment. This is suitable for demonstrating the concept and arguing the case for the approach. For a large-scale production environment it is however unsuitable and it would be necessary to install the software on dedicated hardware (or dedicated virtual machines)

8.3.6 Support for detecting character sets

When loading HTML web documents from enqueued URLs, it is possible to encounter sites that offer content in different source encodings.

Presently, we are using a built-in PHP multi-byte encoding function to guess the source encoding of the document – and transform it to UTF-8 before saving it to the URL queue cache.

9 Operational concept

The CRTT experiment will be described to all LoCloud partners and other actors in the Europeana eco-system during project events and professional conferences. The emphasis will be on the lessons learned and the methodological approach, using the prototype to prove the concept and permit for experimentation.

The tool itself will also be made available for widespread testing by interested parties both internal and external to the project.

Finally, the dissemination of the CRTT will seek to identify the possible place of such a technology and approach in the wider Europeana infrastructure.

Since the technology is capable of producing qualified EDM metadata objects (and potentially other data models), the CRTT could be connected to the content ingestion workflow of aggregators and search providers. This would enable two parallel service concepts.

9.1 Self-service SaaS application

Similar to the way a user can add a new URL to Google's indexing queue, users can register for an account with the CRTT and submit a single URL, or upload a Google Sitemap XML, and then simply sit back, wait and see the results.

If capable, the user can implement custom extraction rules, re-index documents to verify the impact of the rules - and repeat until satisfied.

Aggregators who would like to build a comprehensive, searchable index of digital cultural heritage objects quickly could offer this service concept. It could also be offered by MORe content providers in order to auto-aggregate EDM data for contributing to Europeana.

In the latter case, the accuracy of the extraction rules must be greater than if the purpose merely is to build a full-text searchable index.

9.2 Supporting services

If the CRTT is offered as a channel for publishing content by aggregation bodies, but the complexity of authoring custom extraction rules and enhancing the HTML mark-up for enhanced SEO is perceived to be too great, that could constitute a potential market for providing supporting services.

Private companies could provide such services on a consultancy basis - but also by public authorities whose organizational mandate includes supporting 'underlying' or 'associated' institutions.

10 Future outlook and next steps

At the time of writing, the CRTT is an **experimental prototype** that was developed to **validate a concept** rather than a production-ready software. This is in line with the objective of the activity.

The CRTT was designed with **functionality** in mind, rather than capacity. The tool requires additional development effort to **improve scalability** for use in live production environments with larger volumes of data.

The validation testing demonstrates that the CRTT approach is **viable for aggregating indices for full-text searching**. It would be worth exploring with Europeana whether, such an index, might co-exist with the EDM based faceted index of Europeana, though this was not envisaged to be a result of the experiment.

Additionally, testing has validated that the CRTT approach is **viable for auto-extracting qualified EDM objects** from HTML pages if the structure of the underlying mark-up permits **custom extraction rules** to identify semantic elements.

It is also possible to envisage that **predefined custom rules templates** could be made for popular and **widely used CMS applications** such as e.g. Joomla or WordPress whereby the threshold for using the system by an end-user is further limited.

It is an undisputable fact that the improvements needed to improve the quality of metadata extracted by CRTT would also lead to **greatly improved SEO** and performance in the indices of major search providers like Google, Bing, Yahoo! and Yandex.

In future LoCloud and Europeana might promote good practices in **HTML information design** and SEO through mark-up embedded semantics as a means of improving discoverability through mainstream search engines - as well as a means of enabling automatic crawling/indexing of data for the Europeana portal.

Further use of CRTT within LoCloud includes custom metadata extraction from new partners sites and **integration with MORe** through preparation of a SIP package suitable for merging with the upstream ingestion workflow, from LoCloud to Europeana.

Furthermore, the CRTT source code will be published as a **public repository** on the social development platform **GitHub**, where it may be 'forked' by any interested party and reused according to a permissive **open source** license.

This deliverable documents the **lessons learned** throughout the CRTT prototyping and testing and feeds into the overall Europeana and LoCloud **self-assessment** and **improvement cycle**.

Glossary

Term	Description
DOM	Document Object Model - a cross-platform convention for interacting with HTML, XHTML and XML documents as objects.
HTML	Hyper-Text Mark-up Language
Microformat	A set of schemas for embedding semantic structures within HTML pages.
RDF	Resource Description Framework an abstract data model for expressing objects as subject-predicate-object triples.
RDFa	Resource Description Framework in Attributes – a way of embedding RDF in HTML tags
Schema.org	A joint set of schemas agreed among all the major search providers that allows embedding of semantic structure within HTML documents.
XML	eXtensible Mark-up Language
XPath	XML Path Language is a query language for selecting nodes from an XML-document. Standard defined by the W3C