Integrating Image Resources Into Virtual Research Environments For The Humanities – a Simple Image Presentation Interface (SIPI) based on IIIF

Lukas Rosenthaler
lukas.rosenthaler@unibas.ch
University of Basel, Switzerland

Andrea Bianco
andrea.bianco@unibas.ch
University of Basel, Switzerland

Peter Fornaro
peter.fornaro@unibas.ch
University of Basel, Switzerland

Introduction

Presenting images on the web has a long tradition. The <img>-tag was proposed February 1993 by Marc Andreessen, at this time employed by the National Center for Supercomputing Applications of the University of Illinois at Urbana-Champaign, a pioneering institution for the development of the World Wide Web. Marc Andreessen later founded Netscape, the browser that helped to make the web popular. Its most simple form, the image tag contained an src-attribute containing the URL of the image, such as **<img src="http://dhlab.unibas.ch/imgs/sample.jpg">.** Basically, the mechanism to include images into a website has since remained the same. The dynamic manipulation of the DOM with JavaScript (also a invention introduced by Marc Andreessen and Netscape) allowed users to dynamically add, delete or exchange images within a website. Thus complex web-applications that dealt dynamically with images became possible.

However, while on the side of the browser some scaling is possible, the URL had to address an image file with fixed properties (file format, image size etc.). As the computing power of servers grew, it became possible to render the image on the fly with properties that are passed using the HTTP request (e.g. embedded in the URL or as URL-parameters). Recently, the International Image Interoperability Framework (IIIF) emerged as a standard syntax to pass parameters such as image size in pixels, cropping region, rotation angle and file format using a well-defined URL syntax. The IIIF-standard allows to share image resources between web-applications such as Virtual Research Environments (VRE) in a standardized way. There are quite a few IIIF-compliant image servers around, some using a native implementation, some wrapping image transformation programs using a script language such as python.

By a mandate of the State Secretariat for Education, Research and Innovation (SERI), the Swiss Academy of Humanities and Social Sciences (SAHSS) has created a new institution for the preservation and long-term curation of research data in the humanities (Data and Service Center for the Humanities, DaSCH). It ensures permanent access to research data in order to make it available for further research. A pilot program started in 2013 and has been successfully finished. The DaSCH will be permanently installed on January 1st 2017, to guarantee seamless services. The Digital Humanities Lab (DHLab) of the University of Basel has been mandated with the operation of the new institution. For this purposed, DHLab has developed a flexible platform based on semantic web technologies (RDF, RDFS, OWL). Besides text sources, about 500,000 high-resolution images have been ingested to the system during the pilot phase. We decided to use IIIF for presenting the images in order to maximize the interoperability with external systems. Furthermore, we need to preserve only one image file, since IIIF allows using the archiving master also for dissemination and presentation. However, none of the existing IIIF-compliant servers satisfied our needs.

Design Requirements

Therefore, we decided to design and implement our own IIIF server guided by the following requirements:

- Interoperability with internal databases (e.g. RDF-triplestores, RDBMS etc.) containing annotations, metadata etc. as well as access permissions.
- Preservation of all embedded metadata (e.g. EXIF, XMP, TIFF etc.) during all format conversions
- ICC color profile conversions where necessary

- User authentication compatible with the proposed draft of the IIIF for authentication
- High-performance transformation of images including rotation, format conversions (e.g. 16 bit to 8 bit depth) etc.
- Support of Secure Socket Layer (SSL/https)
- A configurable image cache in order to reduce the computational load on the server
- Support of cross origin resource sharing (CORS)
- Import and transformation of images. The server must be able to import images and convert them to the desired master file format (in our case JPEG2000).
- Features beyond the scope of the IIIF-standard such as adding watermarks, size restrictions etc.
- Integrated simple webserver functionality
- Modular extensibility, e.g. integrating support for RTI imaging (both initial transformation and serving a web-based RTI-viewer – Fornaro et al, 2016)

Implementation

In order to fulfill these requirements, we decided to use C++11 to develop a native, modular, high performance IIIF-compliant image server. A decisive feature was to make the server scriptable with a script language without compromising on performance. We decided to use Lua, an extremely fast, performing, and extensible script language with a small footprint that has only small overhead. It is widely in use for computer games such as World of Warcraft or Minecraft. In order to support different image file formats, we use open source libraries such as libtiff, libjpeg etc. and a modular, extensible architecture. In order to support the JPEG2000 image format, we rely on the kakadu-library. Unfortunately kakadu is not open source, but it is one of the most performant JPEG2000-libraries available. In addition, when acquiring a license, the full source code is provided. Depending on the licensing model, the free distribution of binary packages is included.

In order to reduce the computational load of the server, an efficient caching service has been implemented. The canonical IIIF-URL is used as key for caching since it is unique for each parameter set of the IIIF request.

The Lua-interpreter has been extended with SIPI-specific functionality. Using configurable routes, a fully IIIF-compliant image server has been implemented with the following features:

- Full support of SSL (https) using the OpenSSL library.
- Preservation of metadata. We use the open source exiv2 library and own parsing/generating routines to bridge the differences between the different image formats.
- ICC profiles are interpreted and converted using the open source "little cms" library.
- Before serving an image, a configurable pre-flight lua-script may be executed. Within this script different tasks can be performed, e.g. querying a database for access rights, adding watermarks, ICC profile conversions etc.
- Native support of JSON Web Tokens. JWT's may be analyzed using simple Lua functions (e.g. in the pre-flight script) for authentication and access control.
- Querying other databases using RESTful services. These RESTful query functions are also exposed to Lua.
- Image upload using HTTP multipart/form-data headers. The upload process and file conversions can be controlled with simple Lua scripts.
- Cache control with a simple web-based administration interface.
- Native support for sqlite3 databases from the embedded Lua.

SIPI is open source and can be found on Github. In order to use the JPEG2000 format, a licensed copy of the kakadu library has to be provided by the user. We will provide an extensive manual and binary downloads (including JPEG2000 support) for all major Linux distributions, OS X and a docker image on http://sipi.io

Conclusion

SIPI is a fully IIIF-compliant native image server which integrates extremely well into existing platforms. The flexibility provided by the embedded scripting language, as well as the features going beyond the IIIF specification, allow the integration of IIIIF-based interoperability into existing imaging platforms and image repositories. The support of the secure socket layer (https) access control is a necessity in the environment of digital humanities.

SIPI can easily be customized and extended for special purposes. Elaborated imaging methods (e.g. support for multi-spectral images, image processing functions etc.) could be implemented using the existing SIPI server as base. However, such enhancements will

require extensions to the IIIF-syntax. Further development at the DHLab will include the preprocessing of RTI-images as well support for other media types such as PDF and moving image. We are also working on the implementation of the Amazon S3 client-API in order for SIPI to directly serve images that are stored in a S3 compatible cloud storage.

Bibliography

Fornaro, P., Bianco, A., Rosenthaler, L., (2016) Digital Materiality with Enhanced Reflectance Transformation Imaging. Archiving Conference. 19. April 2016 Vol., no. 1, p. 11-14