A Book, a Web Browser and a Tablet: How Bibliotheca Alexandrina's Book Viewer Framework Makes It Possible

Bibliotheca Alexandrina (http://www.bibalex.org/)

Introduction

A lot of institutions around the world are engaged in multiple digitization projects aiming at preserving the human knowledge present in books and availing them through multiple channels to people around the whole globe. These efforts will sure help close the digital gap particularly with the arrival of affordable e-readers, mobile phones and network coverage. However, the digital reading experience has not yet arrived to its maximum potential. Many readers miss features they like in their good old books and wish to find them in their digital counterpart. In an attempt to create a unique digital reading experience, Bibliotheca Alexandria (BA) created a flexible book viewing framework that is currently used to access its current collection of more than 220,000 digital books in five different languages which includes the largest collection of digitized Arabic books.

Using open source tools, BA used the framework to develop a modular book viewer that can be deployed in different environments and is currently at the heart of various BA projects. The Book viewer provides several features creating a more natural reading experience. As with physical books, the reader can now personalize the books he reads by adding annotations like highlights, underlines and sticky notes to capture his thoughts and ideas in addition to being able to share the book with friends on social networks. The reader can perform a search across the content of the book receiving highlighted search results within the pages of the book. More features can be further added to the book viewer through its plugin architecture.

BA's Book Viewer Framework

BA's book viewer framework provides several services that can be used to build a powerful book viewer experience: morphological search in different languages, image processing, copyright management, server load balancing and scalability, and a personalization engine that includes different types of annotation services.

Figure 1.0 shows the different components of the framework representing the blueprint of a book viewer. A book viewer is divided into two main blocks; the client side component and the server side component. The client side handles the user interface and the rendering of the book on the different client platforms, e.g. mobiles, tablets, web browsers and gaming consoles. It provides features like multiple level zooming, thumbnail view, flip view and scroll view. The current implementation of the book viewer is internationalized into 3 different languages that can be extended.

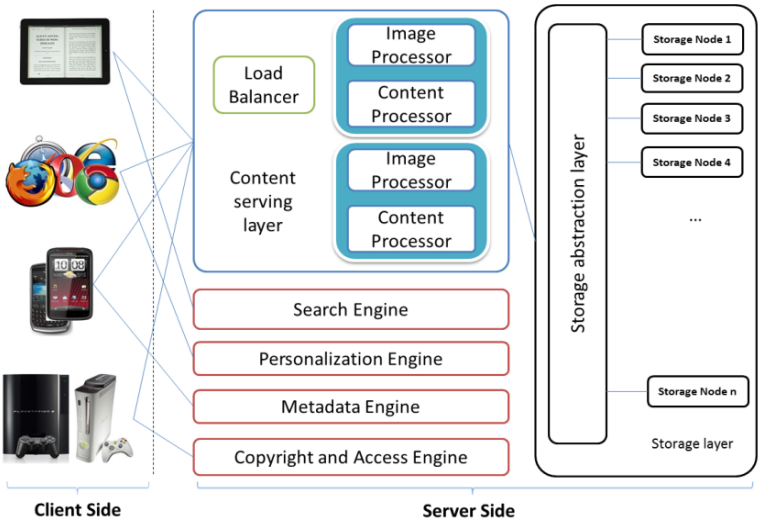

Figure 1: BA Book viewer framework Architecture

The server side holds the core logic of the book viewer. The *Content Serving Layer* streams the book pages to the clients and performs the necessary image processing. It also streams the book text and word coordinates. The *Search Engine* module performs search queries against the full content of the book. A *Copyright Engine* coordinates access to the books by different clients based on a set of rules. A *personalization engine* gives the user the ability to annotate the displayed content and share the book across social networks.

The book viewer framework demonstrates a modular design that provides many benefits: it is possible to disable any function that is not currently present using a simple configuration file, for example, the implementer can decide to disable search within content if the textual content of the book is not available, or he can decide to disable the personalization features if it is not possible to store user generated content. The framework modularity makes it also possible to replace any engine implementation with another one provided that it implements the same engine interface. All the server functionalities are available through RESTful calls to the different clients. This abstract design provides great flexibility making it possible to develop new clients as more devices arise.

The Client Side

A client is responsible for presenting the book to the user. Different clients communicate with servers via a standard set of APIs to retrieve metadata, perform a search query, retrieve search results with highlight information, authenticate and access annotation services. BA has currently different implementations of the client varying from a web implementation supporting major browsers available at http://dar.bibalex.org to an Android mobile and tablet implementation still under development.

BA's web based book viewer (Figure 2) displays related books and provides sharing and embedding information that can be used by users to spread the word about your content. The book viewer supports skins and can easily be embedded in your current project where it blends with your current design. Once the user decides to use the annotation features, he is required to authenticate. You can use any authentication service provided that it implements the necessary interfaces on the server side. Once authenticated, the user can add his annotations. The different kinds of annotations currently implemented are highlights, underlines and sticky notes.

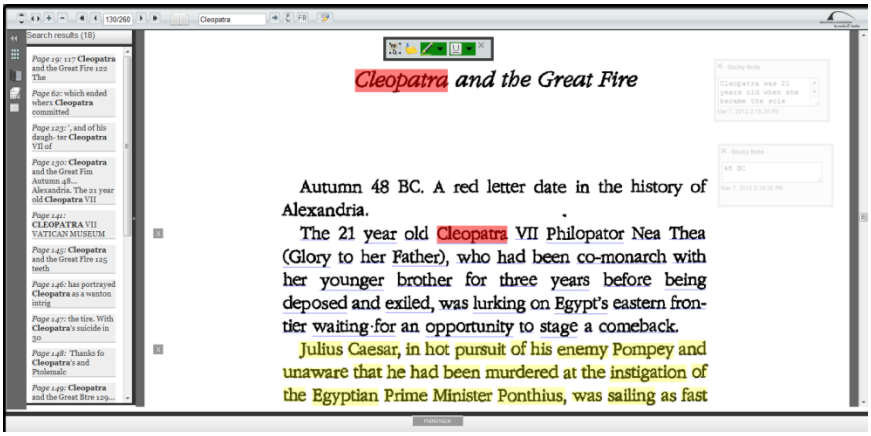

Figure 2: BA Book Viewer

The Book viewer implements several strategies to streamline the user experience. It caches several pages in advance to reduce the time spent by the user to load the next page. The viewer maintains a cache window to enable the reader to move to and fro smoothly. If a reader decides to scroll through the book, the book viewer waits till the cursor stops and starts to load the cache window around this location. The viewer utilizes a simple format for the coordinates information rather than XML to reduce the time required for loading. Whenever a page is loaded, the book viewer loads its word coordinate information as well to be used in defining word selections and search highlights. Whenever the submits a search

query, a request is sent to the *Search Engine* which in turn replies with a list of matching pages and snippets containing the matched terms. The matching terms are used to extract the word coordinates of the matching pages and display the Highlights. The viewer provides multiple levels of zooming. It initially starts using Javascript to enlarge the page to a certain threshold. Once the threshold is reached, another version of the image with a larger resolution is requested from the *Content Serving Layer* which performs the necessary scaling. Until this higher resolution image is loaded, the viewer will display the current version at a larger scale.

The Server Side

The server side of the book viewer framework provides RESTful services over HTTP with data represented in JSON, a flexibility that makes it possible to develop multiple clients. The server side consists of several components; a *Storage Layer*, a *Content Serving layer*, a *Load Balancer*, a *Metadata Engine*, a *Search Engine*, a *Personalization Engine* and a *Copyright Engine*. The *Storage Layer* consists of content storage nodes that stream the raw content of the book to the *Image* and *Content Processors*. BA currently uses storage nodes based on commodity hardware which provide a cost effective way to scale up as content increase. A storage abstraction layer provides a RESTful interface for fetching files from the storage nodes. It isolates the underlying storage implementation and makes it possible to implement different storage policies and several tiers of storage based on the frequency of use and other factors: e.g. frequently used objects can be kept on the fastest storage tier. The Storage layer handles caching and load balancing across storage nodes in addition to ensuring content availability through redundancy. The *Content Serving layer* consists of powerful machines that perform the necessary image conversion and scaling necessary for displaying the books on different devices. It caches the resulting images and streams them to the clients together with page text and word coordinates to support the different functionalities of the client. To improve the performance, the *Load Balancer* dispatches the requests to the least utilized node in the *Content Serving Layer*. The *Metadata Engine* prepares technical (e.g. number of pages) and bibliographic (e.g. title) information about the book in JSON format. The implementer is responsible to translate the metadata standard used at the backend systems to JSON. Metadata can be coupled with search URLs to provide means of performing further queries against the whole collection when they are displayed as hyperlinks on the client. Solr is used at the heart of the *Search Engine*. A different search provider, e.g. a data base or XML, can be used provided that it implements the required RESTful interface. Using Solr, a search query results in a list of page hits with highlight information and snippets, which when coupled with word coordinates makes it possible to mark the search hits on top of the images of the book. The *Personalization Engine* supports different annotation services that can be used by client to provide a richer user experience. The personalization engine can run over traditional SQL databases, Solr Index or more specialized annotation servers e.g. Annotea[1].

The *Copyright and Access Engine* coordinates access to the books based on their copyright information. The copyright information is defined as a set of rules detailing if the object is in copyright, subject to an embargo, the number of licenses you have for the display of simultaneous copies of this book, who have the right to access this book and what actions are permitted (e.g. view, partial view, print, …etc). For example, given several clients accessing your collection of books at the same time, the *Copyright and Access Module* will coordinate with the different clients to make sure that the maximum number of simultaneous copies of the book are not exceeded.

[1] The Annotea Project, http://www.w3.org/2001/Annotea/

For a given book, the Copyright information can be translated into rules along three dimensions; time of access, location of access (e.g. geographic location, IP range) and role of the user who requests access to content. (Figure 3)

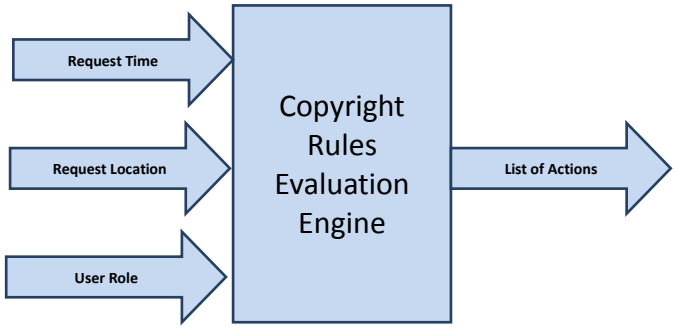

Figure 3: Evaluating Copyright Rules

The Book Viewer Framework supports the organization of the books into groups (sets). A book might belong to different groups. Groups are hierarchical and inherits rules from parent groups. Rules can be defined on the group level or the book (leaf) level. Rules for child groups or leafs take precedence over parent rules. The rules are stored in XML format. The *Copyright and Access Engine* evaluates and outputs a token containing a list of actions to be implemented and understood by the *Content Serving Layer* and the client side implementation. The token expires after a period of inactivity.

Conclusion and Future work

BA Book Viewer Framework presents a modular approach to implementing different types of book viewers targeted towards emerging platforms like mobile phones, tablets and gaming consoles thus increasing the exposure of the content to a variety of users. Its modularity helps the implementer to select the components that are suitable for implementation for a particular environment in addition to the flexibility to change the underlying implementation to integrate with the current systems. BA plans to add the necessary APIs to allow implementers to extend the functionalities of the book viewers through plugins. Development is underway to enhance the Storage Layer at the server side to ensure scalability and performance. Although it is currently used for books, the BA's Book Viewer Framework is currently being extended to include any type of media object.

Based on the framework, BA implemented its book viewer currently used at the heart of several projects, e.g. http://dar.bibalex.org with more features planned for addition, e.g. sharing annotations. Viewers for different types of media objects are also being added to the development pipeline.