Lightroom Nested Keywords  
(Keyword Trees for Lightroom)  
(README.md)

Lightroom Nested Keywords is a website to display Lightroom images that were tagged with hierarchical keywords. This is written in HTML5, CSS3, PHP and is based on the Bootstrap framework.

For a working demo:

* <http://nestedkeywords.com> for a demo website
* <http://photos.shorelinehistoricalmuseum.org> for a working instance in production
* <http://photos.normanrhansen.com> for the author’s website of family photos

# Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

## Prerequisites

What things you need to install the software and how to install them

Give examples

### Installing

A step by step series of examples that tell you how to get a development env running

Say what the step will be

Give the example

And repeat

until finished

End with an example of getting some data out of the system or using it for a little demo

## Running the tests

Explain how to run the automated tests for this system

### Break down into end to end tests

Explain what these tests test and why

Give an example

### And coding style tests

Explain what these tests test and why

Give an example

## Deployment

Add additional notes about how to deploy this on a live system

## Built With

* Bootstrap - The web framework used
* PHP – Server-side scripting
* jQuery – Client-side visual effects
* JavaScript – Auto-complete fields
* XML – Server-side database. (We do not use MySQL or other database.)

## Contributing

Please read [CONTRIBUTING.md](https://gist.github.com/PurpleBooth/b24679402957c63ec426) for details on our code of conduct, and the process for submitting pull requests to us.

## Versioning

We use [SemVer](http://semver.org/) for versioning. For the versions available, see the [tags on this repository](https://github.com/your/project/tags).

## Authors

* **Barry Hansen** - Initial work - [Milliwatts](https://github.com/milliwatts)

See also the list of [contributors](https://github.com/milliwatts?tab=contributors) who participated in this project.

## License

This project is licensed under the MIT License - see the [LICENSE.md](https://gist.github.com/milliwatts/LICENSE.md) file for details

## Acknowledgments

* Hat tip to anyone whose code was used
* Inspired by Simplicity for Lightroom by Andrew Briggs, and ation
* etc

# General Goals

The web user interface should be:

* Easy to navigate (target audience is casual visitors)
* Responsive web design (layout supports both workstations and mobile devices)
* Only digital photos (not for museum artifacts, objects, or multimedia)
* All online photos are low resolution (to protect property rights)
* File names are user friendly for download (not random 16-character GUIDs)
* Show useful metadata about each photo (description, keywords, filename, collection, etc.)
* Include copyright and how to contact SHM

# User Interface

The main methods of navigating the images should include:

## Top Level Landing Page

The landing page introduces the topic and links to the primary search methods. This page includes the museum’s copyright and how visitors may request higher-quality images.

Primary Searches:

1. Open a Collection (i.e. Shoreline images, Playland images, Norm Hansen images, etc.)
2. Randomized images (no options)
3. Photo Finder (text search)
4. Keyword Glossary (descriptions and shortcuts to search by keyword)

## 1. Randomized Images

This top-level page offers spontaneous discovery of images.

This page shows a long array of thumbnail images from all collections in random order. Refreshing the web page delivers a new random set.

## 2. Photo Finder

This top-level page provides a flexible and friendly search by text and keywords.

The visitor can type keywords and/or text to search for.

* *Keywords*: Our goal is an easy way to find keywords by using an autocomplete text field.
* *Text*: Our goal is an easy way to find people, since proper names are not in our keyword set.

In our implementation, we don’t need additional features to search dates or locations. These are both stored in hierarchical keywords.

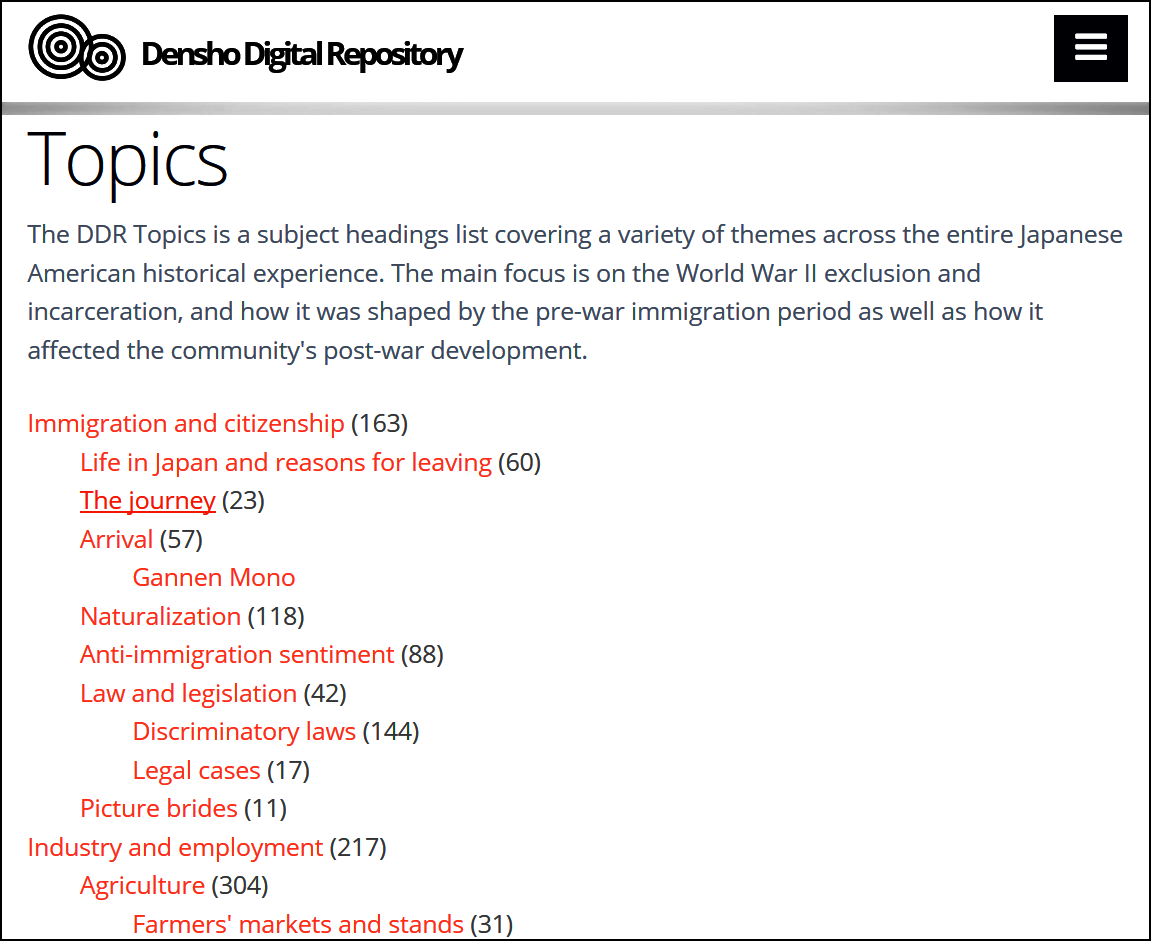
Text: \_\_\_\_\_\_\_\_  
Keyword 1: \_\_\_\_\_\_\_\_ [ keyword explorer ]  
Keyword 2: \_\_\_\_\_\_\_\_ [ keyword explorer ]  
Keyword 3: \_\_\_\_\_\_\_\_ [ keyword explorer ]  
  
 Search Reset

## 3. Keyword Index

This top-level page describes the structured keyword hierarchy.

The goal is to help visitors understand the SHM keyword tags, which has been designed for our specific images. A link for each keyword is a shortcut to show a list of tagged images.

Example of keyword hierarchy navigation from Densho:



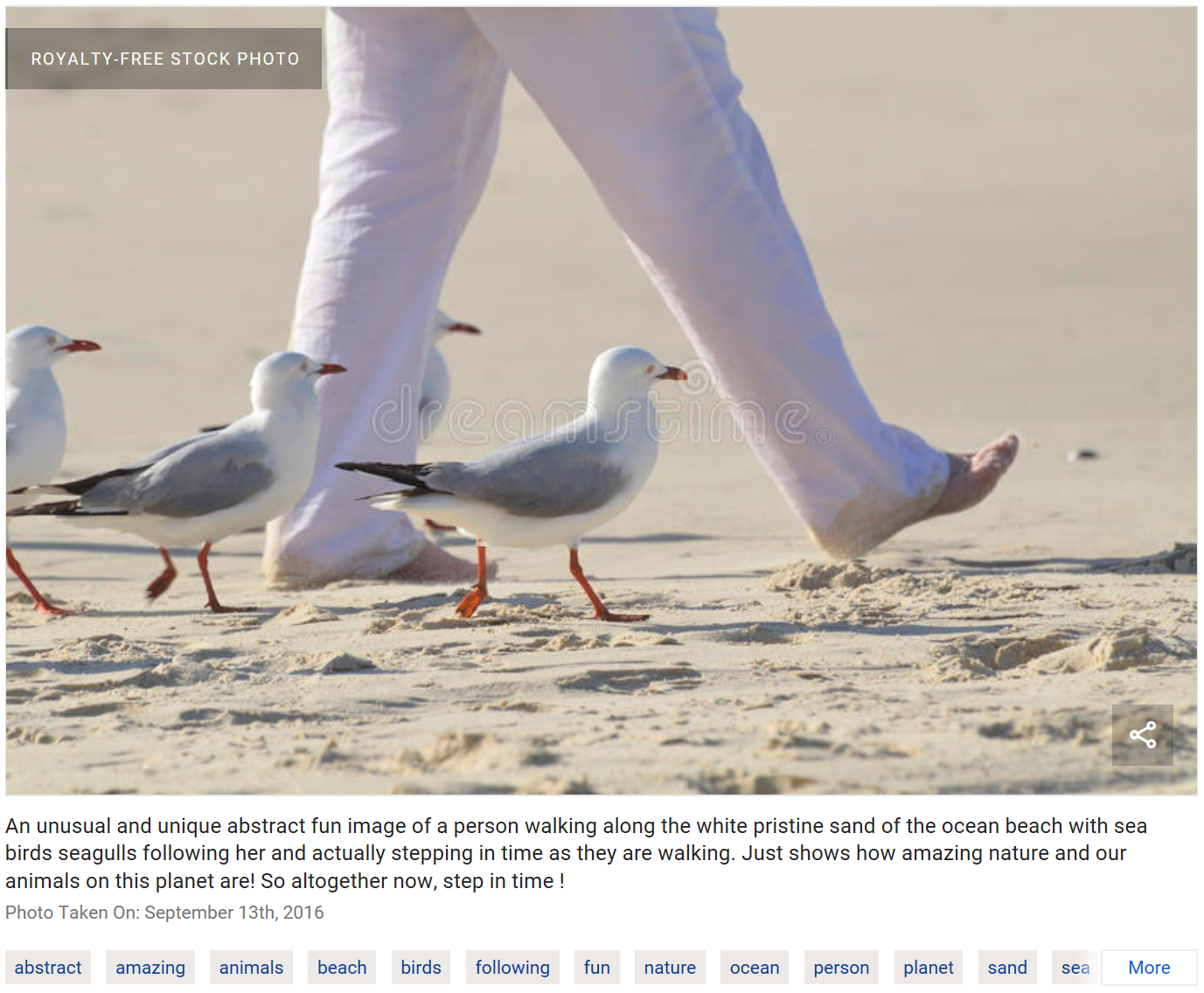
## Image Viewer

This low-level page provides details about a selected image.

Clicking on a thumbnail will open the Image Viewer page. This will show:

* Image: A regular-sized view of the image previously selected
* Description: Text from our “headline” plus “caption” fields
* Attributes: Filename, dimensions, catalog number, …
* Keywords: Clickable keywords that are assigned to this image. Clicking a keyword will show a new list of thumbnails with all photos that share this keyword.

Example from iStockPhoto:



# User Instructions

## For the Archivist:

These tips will help the person managing photos in Lightroom.

### Avoid Blanks in Filenames

We recommend using hyphens to replace spaces in naming files. Avoid blanks and underscores.

Although some people substitute underscores “\_” for spaces, this is not recommended. The underscore will visually disappear where the word is underlined, such as in HTML links.

Although this server programming supports blanks in filenames, the spaces are less user-friendly. Functionally, everything works fine. For example, you can name an image “SHM 0123.jpg” and it will be displayed as expected with all other images. However, it will confuse the user. Spaces are a special character in web addresses and they are encoded as ‘%20’ in the address bar. For example, “SHM 0123.jpg” becomes “SHM%200123.jpg”.

### Duplicate File Names Not Allowed

Every filename must be unique, regardless of the extension. For example, you cannot use both “boat.tif” and “boat.jpg”.

Although Lightroom and other file systems allow it, the publish-to-web process resolves everything into “.jpg” files. Therefore, publishing a source image file with one extension will overwrite any image file with the same base name.

### How to Choose Titles and Captions

For best results, choose a short title and a long caption. The title is displayed under each image thumbnail, so, if the title were very long, then it ruins the even flow of thumbnails on the page.  
Bad title: “1913 (Cir.) Ronald School and Bell Tower. Teacher is Agnes Coffield. Oversize drawer.”  
Good title: “1913 ca Ronald School”

Captions are only shown on the detail page. The detail page has lots of screen space available and so the caption is a good place to provide lists of names and other specifics.

Indicate dates consistently. We believe the date is so important that we show the date, if known, in the first part of the title. Whatever you do, be consistent.

Indicate “circa” dates consistently. We prefer showing “circa” as “(ca)” after the year.  
Example: “1913 ca Ronald School”

### How to Choose Keywords

For best results and usability, you should adopt a controlled vocabulary and apply it consistently.

Keyword hierarchy should be no more than 5 levels deep to remain understandable and usable. If you have more levels, it becomes more difficult to follow. The web server code supports unlimited nesting but has only been tested to six levels deep.

Keywords are case-sensitive, especially during searches. Always choose keywords with the first letter in upper case, because the search function requires it.

Keywords should be in plural form, ie “Dresses” not “Dress”. This generally allows people to search for either singular or plural form and successfully find the terms.

Keywords can contain spaces but be sure to use only a single space between words, e.g “Schoolᴗphotos” and not “Schoolᴗᴗphotos”. Otherwise the search results may not work as expected. To help identify the number of spaces as you traverse the phot collection, watch the URL for multiple plus marks: results.html?key1=School++photos

### How to Apply Keywords

You will need:

* [Adobe Lightroom Classic CC](https://www.adobe.com/products/photoshop-lightroom-classic.html) v8 or above
* Optionally [Photo Mechanic](http://www.camerabits.com/) v5 or above

Keywording images can be a time-consuming process. Our vocabulary has 500 keywords, and we found that one archivist could tag about 30 images in a four-hour workday.

There are no restrictions on how you organize your Lightroom *source* files into folders. The ‘export’ process will flatten the folder structure when it publishes your images.

You can hide selected keywords from the website. To do so, in Lightroom, edit the keyword and remove the checkmark from “Include on Export”. For example, we tag our photos with the media type, such as “Attributes | Photo properties | Original media | N=Negative”. If you have keywords that are of no value to your web visitors, then simply don’t export the keyword.

Every image should be assigned at least one keyword. This program is strongly oriented toward handling keywords; if an image doesn’t have a keyword then it’s practically unreachable and undisplayable.

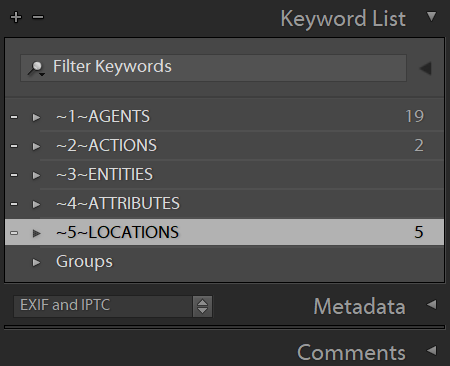
### How to Tag Pictures to Publish

“~6~ Attributes > Publish on Internet

### How to Change the Displayed Order of Topics

Suppose you want the “Keyword Index” page to show its columns in a specific order.

Ordinarily, everything is shown in alphabetical order, starting with the Lightroom’s keyword list. Choose the order of topics by picking top-level keywords in alphabetical order. You can preface the group names with numbers to force the order:



However, these are not friendly names for web pages, so you can use a feature to substitute keywords:

* adobe-xmp-for-php.php, around line 67
* $substitute\_from=array( '~1~AGENTS', '~2~ACTIONS', '~3~ENTITIES', …
* $substitute\_to =array( 'Agents', 'Actions', 'Entities', …

## For the Release Manager:

These tips will help the person transferring photos from Lightroom into the web server.

### You will need

* [Adobe Lightroom Classic CC](https://www.adobe.com/products/photoshop-lightroom-classic.html) (to organize and tag your image library)
* File Transfer Program, FTP (such as free [FileZilla Client](https://filezilla-project.org/) or [Beyond Compare](http://www.scootersoftware.com/)).

### How to Update the Website’s Images

Method:

1. Lightroom Classic CC:
   1. Select the images to export
   2. Export selected images to local folder “photos/large” at medium resolution.  
      We recommend 640-px size maximum height.   
      Be sure to select option to include keyword tags.
   3. Export selected images to local folder “photos/small” at thumbnail resolution.  
      We recommend 133-px maximum height.
2. FTP:
   1. Delete any existing files inside the target system folder of photos
   2. Copy all photos to the web server, i.e. “photos/large” and “photos/small”
3. Web browser:
   1. Open web page <http://www.mysite.com/build-keyword-database.html>   
      This will refresh the internal database when it displays this page.
   2. Look for errors reported in the web page and resolve them.

## For the Web Master:

These tips will help the person starting their own photo library using this framework for the first time.

### How it Works

You will need:

* Familiarity with HTML5, PHP, CSS3, [Bootstrap](https://getbootstrap.com/docs/3.3/) 3.3, [jQuery](https://jquery.com/) 3.3
* Your favorite web coding workbench (such as free [Visual Studio Code](https://code.visualstudio.com/))
* No database (instead of MySQL we use XML files and built-in PHP support)

Image files are stored locally on the web server. You can change this in PHP settings and store the source images almost anywhere, as long as the PHP file functions can read them.

Be sure to store images on a high-speed file system. If you use a remote the remote storage can reduce performance, since this program will frequently read file information.

### php/custom.php

The file “php/custom.php” must be updated for each project:

* Your own domain name:  
  $mySubdomain = 'test.';  
  $myDomain = 'shorelinehistoricalmuseum';  
  $myTLD = '.org';
* Contact information:  
  $myOwnerName = 'Barry Hansen';  
  $myWebmasterEmail = 'barry@electromagneticsoftware.com';
* Your HTML file names:  
  $searchPage = 'search.html'; // URL to search for photos  
  $resultPage = 'search.html'; // URL to show search results  
  $detailPage = 'detail.html'; // URL to show one photo's details
* Your page footer:  
  <address>  
   <b>Shoreline Museum</b> &bull; PO Box 55594, Shoreline, WA  
   <br/><a href="mailto:shm@shm.org">shm@shm.org</a>  
   <br/>206-542-7111  
  </address>
* Your database file names:  
  $imageDatabaseBasename = 'xml/image-db';  
  $keywordTreeBasename = 'xml/keyword-tree-db';  
  $keywordFlatBasename = 'xml/keyword-flat-db';  
  $autocompleteFilename = 'js/autocomplete-keyword.js';
* Your folder names where images are stored:  
  $myLargeImageFolder = 'photos/large';  
  $mySmallImageFolder = 'photos/small';

### Linux File Permissions

If your web server is Linux-based, then you need to make a few selected files writeable by the Apache process. These files are overwritten by PHP programming and save the cached version of critical internal data structures.

* js/autocomplete-keyword.js
* xml/image-db.xml and .php
* xml/image-db.xml and .php
* xml/keyword-flat-db.xml and .php
* xml/keyword-tree-db.xml and .php
* /image-list.csv

On some Linux systems, the Apache process runs in the ‘www-data’ user profile.

It’s probably simplest to use “chmod 777 filename” to individually allow it to be rewritten.

You can verify proper operation by visiting <http://photos.normanrhansen.com/build-keyword-database.html> and look for links to the updated XML files in a table at the top.

### File Storage

Our implementation is to store images on a local drive to the web server. This gives good performance and it is a low-cost system in common usage in 2019.

Files are uploaded via FTP or your other favorite method of transferring files to the web server.

Alternative: If additional space or CDN (content delivery system) is needed, a knowledgeable webmaster can modify the source PHP code to use cloud storage services. Note that several online file storage services are available at low-to-zero cost, such as AWS, Google Drive, Microsoft OneDrive, Apple iCloud, and more.

### Bookmarkable Links

The design is for every page to have a unique URL that can be bookmarked, shared and indexable by search engines. So, the Search strings are encoded in the URL and preserved across pages.

For example, suppose user “Alice” finds an interesting group of photos after her search, and wants to share it with user “Bob.” She can copy/paste her address bar and send it to Bob. Furthermore, if Alice opens a photo after her search, she can copy/paste the URL and it will include the search criterion in the link so that Bob and return to Alice’s original search page and modify it further.

Encoding:

* Specific image is selected by its name on disk, and not by Image ID since file extensions vary.  
  File name can contain blanks and other special characters  
  ?file= *urlencoded* basename.ext
* Text = *free form text*?text=*urlencoded string*
* Keywords = *choice from keyword index*?key1=*urlencoded string* (same as “key”)&key2=*urlencoded string*&key3=*urlencoded string*
* Some pages display multiple images  
  ?count=*number*

### Bootstrap Framework

This design uses the Bootstrap v3 framework (<https://getbootstrap.com/>), an open source toolkit for web user interface technology. It provides a big boost for writing new websites from scratch with a responsive, modern look and feel.

Since Bootstrap has nothing in common with WordPress, we run this website in a subdomain on separate web server hardware.

### Web Server

The current implementation uses a subdomain, e.g. “**photos**.shorelinehistoricalmuseum.org” instead of “**www**.shorelinehistoricalmuseum.org”.

This design means that the photo collection runs completely independently of the organization’s main website. There are no dependent files or frameworks between the two. Indeed, the photo subdomain can run on different hardware or different web hosting plan than the main “www” website.

This was a deliberate decision to make photos fully independent. It simplifies software development to avoid building dependencies. For the Shoreline Historical Museum, the main website runs on an obsolete WordPress version using an old theme which was poorly suited to responsive web pages.

It is the webmaster’s responsibility to write their own HTML to integrate the main website’s look-and-feel, color schemes or visual elements as much or as little as desired.

### Search Internals

Our text search will examine title, caption and filename for:

* Find a few letters or one word
* Find a few words in any order implicitly joined by AND
* Find an exact phrase enclosed in double quotes

Visitors can achieve more advanced search queries with Google’s index of the site. For example:  
 <https://www.google.com/search?q=ronald+site%3Ashorelinehistoricalmuseum.org>

The “text search” is kept simple due to limited programming resource. For example, we don’t support similar-sounding words ([Soundex](https://en.wikipedia.org/wiki/Soundex)), [stop words](https://en.wikipedia.org/wiki/Stop_words), [rank sorting](https://en.wikipedia.org/wiki/Ranking_(information_retrieval)), [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), or other complexity. For example, if you don’t find “Jensen” then you’ll have to guess-n-check on your own for alternate spellings like “Jenson” and “Johnson”.

Some elements of implementation:

* Ignores leading/trailing blanks
* Multiple words are treated as a list of independent words, searched in any order
* Multiple words imply AND
* Allows search for special characters (comma, ampersand, semicolon, etc)

Implementation is actually three different searches:

* Single word
* Multiple words
* Words enclosed in double-quotes

One-word test cases:

* Yard - 19 results, SHM 259, 457, 494, 495, 704, …
* Ronald - 61 results: SHM 009, 011, 014, 015, 016, …
* 457 - 4 results: SHM 457, 1457 front, 1457 back, SHM 1559

Two-word test cases

* SHM 457 - 1 result: SHM 457
* Ronald school - 36 results: SHM 014, 019, 064, 071, 093, …
* School Ronald - “
* “School Ronald” - 0 results
* Ronald school 1938 - 1 result: SHM 198
* & - 376 results

### Collections

A brief discussion of “collections of photos” is worthwhile, because this software does not have any explicit support for “Collections.” This is by design.

Our museum’s word “collection” refers to a body of photos or artifacts that were donated together as a group. We have several groups of images that were donated at different times by different people. For *physical* images we store the original photos in groups for various topics such as:

1. SHM Historical Collection
2. Norman R. Hansen Collection
3. Playland Collection
4. Aurora Avenue Collection
5. School Class Photos Collection

However, once an image is scanned, indexed and keyworded, our museum has no particular need for distinguishing between the original collections. Our web visitors generally don’t care or need to know how the photo arrived. If, for some reason, the visitor needs to contact the museum for inquiries about a single photo, the image ID (file name) is all we need to easily find the original print or negative among all of our collections.

If someday, the “Collections” concept becomes important, our easy solution is to apply a keyword to each image to identify the collection. The new keywords (collection name) should be nested under an umbrella keyword such as “Collections.” The robust keyword features of this web implementation will automatically make them easily findable and searchable, using the same approach as other existing keywords.

### Maintainability

Our goal is to implement the project in ways that can be maintained by subsequent web developers.

* The implementation uses public, well-known and well documented web technology.
* Software is built on the current stable version of each dependency (HTML5, CSS3, PHP 7, etc.)
* If possible, the amount of custom programming should be kept to a minimum.
* Programming should adhere to modern best practices including documentation, comments and source code formatting.
* There are both unit tests and functional test cases.

Custom programming uses version control of source files to ensure long-term storage and availability. A public archive such as GitHub will serves these needs nicely and make it Open Source in the hope of contributions from the programming community.

### Keyword Internals

When Lightroom saves hierarchical keywording to an image, it writes a list of all the words in the hierarchy to the **IPTC** keywords field. It also writes the hierarchy to "Hierarchical Subject" in XMP-lr. So, keywords "animal>mammal>fox" in our LR keyword list would, in a JPG file, be IPTC keywords: "animal, fox, mammal" and in XMP-lr Hierarchical Subject: "animal|mammal|fox." ([source](https://www.lightroomqueen.com/community/threads/hierarchical-keywording.22661/))

[ExifTool](https://www.sno.phy.queensu.ca/~phil/exiftool/) can display all possible information from metadata. So, one possible implementation is to spawn the ExifTool to read the desired metadata, and then process its output into the desired formats.

[Adobe XML for WP](https://wordpress.org/plugins/adobe-xmp-for-wp/) WordPress plug-in can retrieve Lightroom metadata, including hierarchical keywords. This is a good source of PHP coding examples.

### Keyword Hierarchy Example

Here is the keyword hierarchy of the Shoreline Historical Museum. For brevity, only the top few levels are shown.

This example is documented here for convenience. You can visit our production website at photos.shorelinehistoricalmuseum.org/keywords.html to view the ‘live’ keyword tree.

The actual keywords are arbitrary. The coding has no hard-coded keywords; the software will read all the keywords from image metadata and build its internal database from what is found.

The prefix “~1~” determines the sort order of top-level groups. For our purposes, we want dates and locations to have prominence.

* ~1~DATES
  1. 1800-1899
     + 1880s, 1890s
  2. 1900-1999
     + 1900s, 1910s, 1920s, …
  3. 2000-2099
     + 2000s, 2010s, 2020s, …
  4. No Date
* ~2~LOCATIONS
  1. Cities
     + Ballard, Bellingham, Bothell, …
  2. Landmarks and historical markers
  3. Named Places
     + Aurora Village, Briarcrest, Brookside, …
  4. Named Streets
     + 4d Ave NW, 5th Ave NW, 8th NW, …
  5. Unknown location
* ~3~ACTIONS
  1. Non-moving
     + Conversing, Eating, Observing, Posing, …
  2. Performing
     + Playing, Baseball, Basketball, Bicycling, …
  3. Traveling
     + Driving, Parading, Riding, Running, Sailing, …
  4. Undefined actions
  5. Working
* ~4~AGENTS
  1. Livestock
     + Chickens, Cows, Goats, Horses, Mules
  2. Occupations
     + Communications, Construction, Entertainment, …
  3. People
     + Adults, Children, Elderly, Infants
  4. Pets
     + Cats, Dogs
  5. Wildlife
     + Birds, Fish, Mammals
* ~5~ENTITIES
  1. Buildings
     + Banks, Churches, Club / Civic Buildings, Commercial Buildings, …
  2. Civic organizations
     + Boy Scouts, Coast Guard, Elk's Association, Shoreline Historical Society
  3. Clothing
     + Casual wear, Costumes, Dress clothes, Eyewear, Footwear, Hats, …
  4. Flags
  5. Food
  6. Furniture
  7. Infrastructure
     + Boardwalks, Breakwaters, Bridges, Dams, Docks & Pilings, …
  8. Natural features
     + Beaches, Canals, Cleared areas, Fields and Farms, Forests, …
  9. Signage
  10. Teams
      + Dance Teams, Lake Forest Park Baseball Team, …
  11. Technologies
      + Cameras and photographs, Electric & electronic devices, …
* ~6~ATTRIBUTES
  1. Emotions
     + Bemused, Determined, Happy, Neutral, Proud, Sad, Serious, Worried
  2. Events depicted
     + Civic Events, Fires, Holidays, Natural events, Social events, World's Fair
  3. Photo properties
     + Genre, Non-SHMs, Photos of, Print properties, …
  4. Viewer impressions
     + Activism, Anticipation, Barren, Beautiful, Careworn, Casual, …

# Appendix A: Programming Techniques

## Lazy Loading Techniques for Images

* <https://css-tricks.com/the-complete-guide-to-lazy-loading-images/>

# Appendix B: Other Websites

Search words: controlled vocabulary, hierarchical keywords, lightroom

## GitHub LightroomKeywordHierarchy

<https://github.com/ericvaandering/LightroomKeywordHierarchy>

Readme: A controlled vocabulary for Lightroom tagging based on IPTC and other sources of keyword hierarchies. While IPTC is used as a starting point, it's more useful for news stories, so some of the cateories don't map well to photographs.

Additional sources of keywords which have been incorporated into this scheme:

* William Beem's list: <http://williambeem.com/tagging-photos-with-keywords/>
* Controlled Vocabularly - Information visibly on the description here: <http://www.controlledvocabulary.com/index.html>

Barry: This project has no code. It contains one file for one particular controlled vocabulary.

## GitHub Simplicity for Lightroom

Source: <https://github.com/brggs/Simplicity-for-Lightroom>   
Demo: <http://www.andrewbriggsphotography.co.uk/simplicity/>

Plug-in for Lightroom to publish to the web. Generates code using Bootstrap, jQuery and [FancyBox3](http://fancyapps.com/fancybox/3/docs/).

Barry:

* FancyBox supports 0/1/2 captions each photo.
* Search is implemented locally using JavaScript based on id filtering by <div class=”id”>, implemented with Isotope (<https://isotope.metafizzy.co/filtering.html>).

Forked the GitHub project into:

* Documents/GitHub/simplicity-gallery-v1.2 – for editing and synch with GitHub
* C:\Program Files\Adobe\Adobe Lightroom Classic CC\Shared\webengines\simplicity-gallery-v1.2.lrwebengine – for running in Lightroom

## GitHub Best Practices

* <https://10up.github.io/Engineering-Best-Practices/>

## ContentDM

This digital collection software is most-used by libraries. Frightfully expensive.

Umbrella organization is OCLC (Online Computer Library Center) at [www.oclc.org](http://www.oclc.org)

Intended for storing master files for long-term preservation in the OCLC data center.

Features:

* Researchers can combine images with other collections via supporting the International Image Interoperability Framework (IIIF) Image API.
* Responsive websites for smartphones, tablets and workstations.
* Web Content Accessibility Guidelines (WCAG) compliance for, um, something.

“Featured websites” are listed by ContentDM <https://www.oclc.org/en/contentdm/collections.html>

They provide useful examples of layouts with detailed information.

## Mirador Viewer Compatibility

Mirador is a standalone image viewer application for research. It supports comparing content from web repositories that support the International Image Interoperability Framework (IIIF) APIs.

For example, a researcher could compare different versions of the same scanned artwork from different image repositories.

<http://projectmirador.org/docs/>

If supporting IIIF is important to Shoreline Historical Museum, then a future project could add web server programming to provide this API.

## Historic Saranac Lake Wiki

It is possible to offer a crowd-sourced local history site. An engaging example is:  
Historic Saranac Lake, New York <https://localwiki.org/hsl/>

## Issaquah History Museum

* Main website [www.issaquahhistory.org](http://www.issaquahhistory.org)
* Digital collections <http://issaquah.pastperfectonline.com/>

## Southern Oregon Digital Archives

Very well designed “Search” page is a good example of page elements and powerful search capability. The user options are encoded in the URL.

* [digital.sou.edu/digital/search](https://digital.sou.edu/digital/search/)

## iStockPhoto

* [www.istockphoto.com](http://www.istockphoto.com)

## Dreams Time

* [www.dreamstime.com](http://www.dreamstime.com)

## ARTstor

<http://www.artstor.org/>

## Densho – Japanese Internment Stories

* Main website [densho.org](https://densho.org/)
* Digital repository <http://ddr.densho.org/>
* Example keyword hierarchy <http://ddr.densho.org/browse/topics/>

## Images in Museum Collections: A Study of Eleven Museums

* <http://msc.mellon.org/msc-files/Open%20Access%20Report%2004%2025%2013-Final.pdf>
* Andrew W. Mellon Foundation studied issues of online open access to image collections.
* Almost all museums in this study do have a relationship with:
  + [Bridgeman Art Library](http://www.bridgemanimages.com/en-US/about-bridgeman/news/bridgeman-the-world-s-art-collections-online),
  + [Scala Archives](http://www.scalarchives.com/web/)
  + [Art Resource](http://www.artres.com/)
  + and seven of the eleven are part of the [Google Art Project](https://artsandculture.google.com/).

## Guidelines for the Construction, Format and Management of Controlled Vocabularies

* PDF <https://groups.niso.org/apps/group_public/download.php/6484>
* Highlights the importance of delivering
  1. Synonyms
  2. Disambiguation e.g. Richmond Beach (city) Richmond Beach (beach)
  3. Hierarchy e.g. Broader term, Narrower term

## cPanel Documentation (Atlassian)

* Example of effective presentation of tagged content, including “see related tags.”  
  <https://documentation.cpanel.net/label/68Docs/domains>