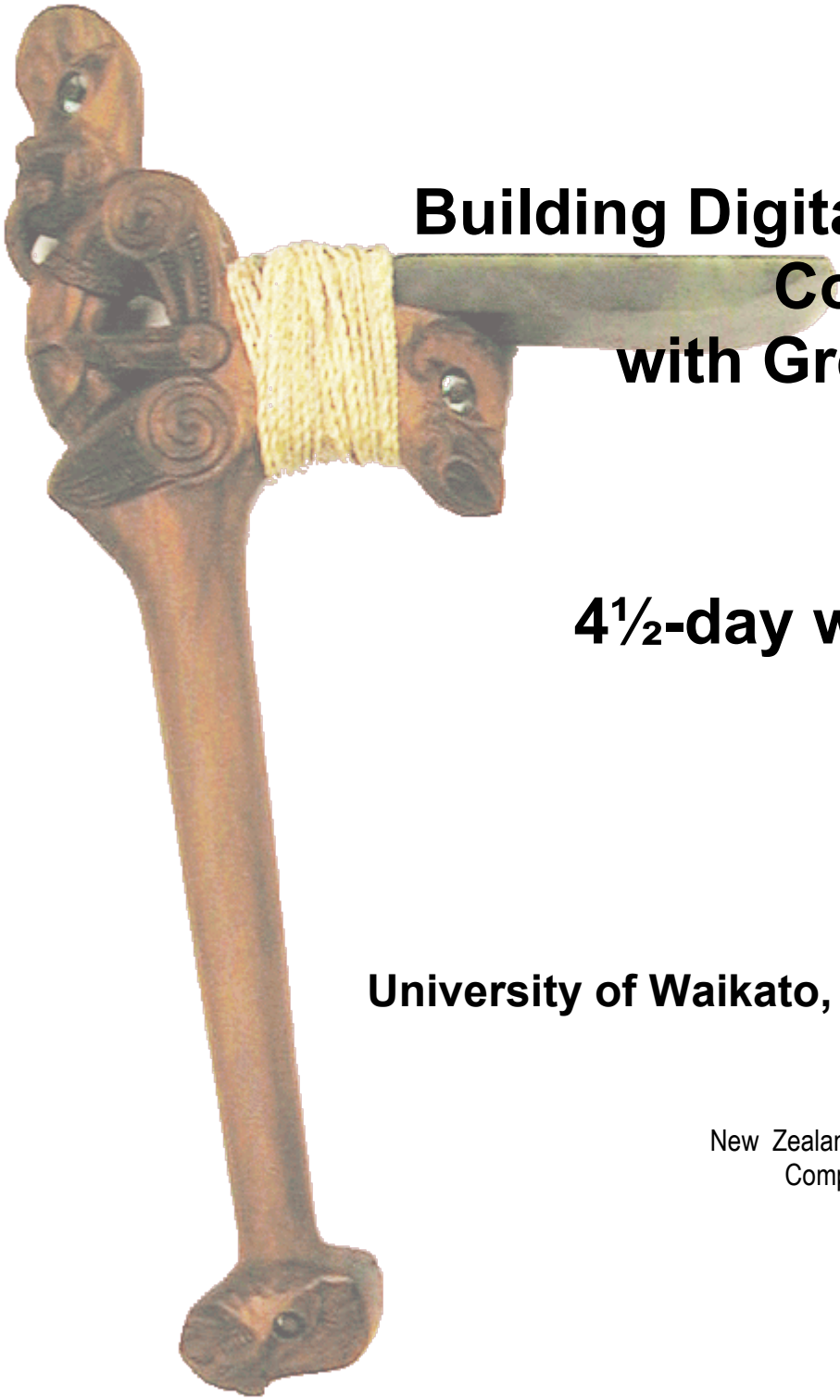


Building Digital Library Collections with Greenstone

4½-day workshop

Ian H. Witten
University of Waikato, New Zealand

Notes prepared by
New Zealand Digital Library Project
Computer Science Department
University of Waikato
New Zealand
nzdl.org
2009

Schedule

Day 1
- Morning: Introduction
  - Introductions: participants introduce themselves and the Pacific DL project
  - Discussion: where will the raw material be coming from?
  - Lecture: Overview of Greenstone, with examples of its use elsewhere
  - Lab 1 Building collections and adding metadata
    - 1.1 Small collection of HTML files
    - 1.2 Simple image collection
    - 1.3 Collection of Word and PDF files
    - 1.4 Exporting a collection to CD-ROM/DVD
- Afternoon: Greenstone collection design
  - Lab 2: Designing collections
    - 2.1 A large collection of HTML files
    - 2.2 Enhanced collection of HTML files
  - Lab 3: Collections of bibliographic material
    - 3.1 Bibliographic collection
    - 3.2 CDS/ISIS collection

Day 2
- Morning: Formatting in Greenstone
- Lecture: Greenstone formatting
  - Lab 4: Greenstone formatting exercises
    - 4.1 Formatting the HTML collection
    - 4.2 Formatting the Word and PDF collection
- Afternoon: Advanced collection configuration
  - Lecture: Advanced collection configuration: section tagging, advanced PDF handling, exploding bibliographic collections
  - Lab 5: Advanced collection configuration
    - 5.1 Pointing to documents on the web
    - 5.2 Enhanced Word document handling
    - 5.3 Enhanced PDF handling
    - 5.4 Section tagging for HTML documents

Day 3
- Morning: Multimedia and scanned images
  - Lecture: Two examples—multimedia and scanned images
  - Lab 6: Multimedia
    - 6.1 Looking at a multimedia collection
    - 6.2 Building the multimedia collection
- Afternoon: Scanned images
  - Lab 7: Multimedia
    - 7.1 Scanned image collection
    - 7.2 Advanced scanned image collection

Day 4
- Morning: Customization
  - Lecture: Customization: macro files and stylesheets
  - Lab 8: Customization
    - 8.1 Customization: Macro files and spreadsheets
- Afternoon: Sharing

Lecture: Sharing collections with OAI-PMH
Lab 9: Sharing collections with OAI-PMH
9.1 Open Archives Initiative (OAI) collection
9.2 Downloading over OAI

Day 5
Morning: What next?
Discussion: moving Greenstone collections around

Materials

1. Workbook containing

Introduction
Greenstone Digital Library Software factsheet
PowerPoint slides for all workshop sessions
Laboratory exercises
How to format the output of your collection

2. Sample files for tutorial exercises

Small set of HTML files (*simple_html*)
Word and PDF documents (*Word_and_PDF*)
Difficult PDF documents (*difficult_documents*)
Image files (*images*)
Large set of HTML files (*tudor*)
MARC records (*marc*)
Multimedia collection (*beatles/advbeat_large*)
Small multimedia collection (*beatles/advbeat_small*)
Scanned image collection (*niupepa*)
Files exported from OAI (*oai*)
Files exported from DSpace (*dspace*)
CDS/ISIS files (*isis*)
Files for interface customization exercise (*custom*)

The Greenstone digital library software

Greenstone is a suite of software for building and distributing digital library collections. It is not a digital library but a tool for building digital libraries. It provides a new way of organizing information and publishing it on the Internet in the form of a fully-searchable, metadata-driven digital library. It has been developed and distributed in cooperation with UNESCO and the Human Info NGO in Belgium. It is open-source, multilingual software, issued under the terms of the GNU General Public License. Its developers received the 2004 IFIP Namur award for "contributions to the awareness of social implications of information technology, and the need for an holistic approach in the use of information technology that takes account of social implications."

http://www.greenstone.org

Technical

Platforms. Greenstone runs on all versions of Windows, and Unix, and Mac OS-X. It is very easy to install. For the default Windows installation absolutely no configuration is necessary, and end users routinely install Greenstone on their personal laptops or workstations. Institutional users run it on their main web server, where it interoperates with standard web server software (e.g. Apache).

Interoperability. Greenstone is highly interoperable using contemporary standards, It incorporates a server that can serve any collection over the Open Archives Protocol for Metadata Harvesting (OAI-PMH), and Greenstone can harvest documents over OAI-PMH and include them in a collection. Any collection can be exported to METS (in the Greenstone METS Profile[1]), and Greenstone can ingest documents in METS form. Any collection can be exported to DSpace ready for DSpace's batch import program, and any DSpace collection can be imported into Greenstone.

Interfaces. Greenstone has two separate interactive interfaces, the Reader interface and the Librarian interface. End users access the digital library through the Reader interface, which operates within a web browser. The Librarian interface is a Java-based graphical user interface that makes it easy to gather material for a collection (downloading it from the web where necessary), enrich it by adding metadata, design the Reader interface, and publish the collection.

Collection building. Collection designers normally interact with the Librarian interface to develop collections that are hosted on the same computer. Collections can easily be copied from one Greenstone installation to another. The Librarian interface also allows authenticated users to manage collections on remote servers, allowing for distributed collection development. An applet version of the Librarian interface is also available.

Metadata formats, Users define metadata interactively within the Librarian interface. These metadata sets are predefined:

Dublin Core (qualified and unqualified)
RFC 1807
NZGLS (New Zealand Government Locator Service)
AGLS (Australian Government Locator Service)

New metadata sets can be defined using Greenstone's Metadata Set Editor. "Plug-ins" are used to ingest externally-prepared metadata in different forms, and plug-ins exist for

XML, MARC, CDS/ISIS, ProCite, BibTex, Refer, OAI, DSpace, METS

Document formats, Plug-ins are also used to ingest documents. For textual documents, there are plug-ins for

PDF, PostScript, Word, RTF, HTML, Plain text, Latex, ZIP archives, Excel, PPT, Email (various formats), source code, MediaWiki wikis

For multimedia documents, there are plug-ins for

Images (any format, including GIF, JIF, JPEG, TIFF), MP3 audio, Ogg Vorbis

[1] published at *http://www.loc.gov/standards/mets/mets-profiles.html*

audio, and a generic plug-in that can be configured for audio formats, MPEG, MIDI, etc.

User base

Distribution. As with all open source projects, the user base for Greenstone is unknown. It is distributed on SourceForge, a leading distribution centre for open source software.

Distributed via SourceForge since:	Nov 2000
Average downloads since then:	4500/month
Currently running at:	4500/month
Proportion of downloads that are documentation:	35%
Proportion of downloads that are software:	65%
Of these,	80% are Windows binaries
	10% are Linux binaries
	10% are Mac binaries
	5% are source

Examples. Examples of public Greenstone collections (see http://www.greenstone.org for URLs) can be found at:

Association of Indian Labour Historians, Delhi
Auburn University, Alabama
California University at Riverside
Charles Darwin University,Australia
Chicago University Library
Detroit Public Library
Gresham College, London
Hawaiian Electronic Library
iArchives, Utah
Illinois Wesleyan University
Ionian University, Greece
Indian Institute of Management
Kyrgyz Republic National Library
LeHigh University, Pennsylvania
Mari El Republic, Russia
National Centre for Science Information, Bangalore, India
Netherlands Institute for Scientific Information Services
New York Botanical Garden
Oxford Digital Library, Oxford University
Peking University Digital Library
Philippine Research Education and Government Information Network
Secretary of Human Rights of Argentina
Slavonski Brod Public Library, Slovenia
State Library of Tasmania
Stuttgart University of Applied Sciences
Texas A&M University Center for the Study of Digital Libraries
Tulane University
University of Illinois
University of Namibia
University of North Carolina ibiblio project
Vietnam National University
Vimercate Public Library, Milan, Italy
Washington Research Library Consortium
Welsh Books Council
Yale University

UN agencies with an interest in Greenstone include

UNESCO, Paris
: Sponsors distribution of the Greenstone software as part of its *Information for All* programme

Food and Agriculture Organization (FAO), Rome
: Their *Information Management Resource Kit* uses Greenstone as the (only)

example of digital library software in the *Digitization and digital libraries* self-instructional module

Institute for Information Technology in Education (IITE), Moscow
Have commissioned an extensive course on *Digital libraries in education* that uses Greenstone for all the practical work

United Nations University (UNU), Japan
Two CD-ROM collections of UNU material have been produced

Humanitarian collections. Greenstone is used by Human Info NGO in Belgium to produced collections of humanitarian information and distribute them on CD-ROM widely throughout the developing world. (For more information, contact Michel Loots <mloots@humaninfo.org>)

Number of humanitarian collections:	approx 35–40
Annual distribution of each one:	approx 5,000 copies

Languages. One of Greenstone's unique strengths is its multilingual nature. The reader's interface is available in the following 50 languages:

Arabic, Armenian, Bengali, Catalan, Chinese (both simplified and traditional), Croatian, Czech, Dari, Dutch, English, Farsi, Finnish, French, Gaelic, Galician, Georgian, German, Greek, Hebrew, Hindi, Hungarian, Indonesian, Italian, Japanese, Kannada, Kazakh, Kyrgyz, Latvian, Malayalam, Maori, Marathi, Mongolian, Polish, Portuguese (BR and PT versions), Pushto, Romanian, Russian, Serbian, Sinhalese, Slovak, Spanish, Tamil, Telugu, Thai, Turkish, Ukrainian, Urdu, Vietnamese

The Librarian interface is in

Arabic, Catalan, Chinese (simplified), English, French, Greek, Latvian, Marathi, Romanian, Spanish, Russian, Vietnamese.

The full Greenstone documentation (which is extensive) is in:

English, French, Spanish, and Russian.

Training is a bottleneck for widespread adoption of any digital library software. Many international training courses have been run.

UNESCO
has sponsored training courses in Bangalore (2002 and 2003), Almaty (2003), Senegal (2004), Suva (2004, 2006, 2007)

Self-study courses
FAO and UNESCO IITE have produced training material on Greenstone

Digital Library conferences
There have been Greenstone tutorials (on several occasions) at all major digital library conferences: JCDL, ECDL, ICADL, ICDL

Librararian conferences
There have been Greenstone workshops and presentations at LITA, DLF, ALA Annual Conference

Payson Institute, Tulane University
has run courses that use Greenstone collections as a resource in locations in Africa (Burkina Faso, Cameroon, Cote d'Ivoire, Democratic Republic of Congo, Ghana, Rwanda, Senegal, Sierra Leone, Togo) and Latin America (Argentina, Bolivia, Colombia, Ecuador, Guatemala)

Others
There have been several Greenstone courses in India (e.g. Khozikode, Poona, New Delhi, Kalikut), Africa (e.g. Namibia, Cape Town, Arusha), some in Canada (Vancouver, Calgary, Edmonton), one in Cuba (Havana), and many in the US.

E-mail support

Number of people on Greenstone email lists: 600

Number of countries represented:	70
Number of messages (excluding spam):	150/month

The *toki* (adze) was a gift from the Māori people in recognition of the New Zealand Digital Library Project's contributions to indigenous language preservation, and resides in the project laboratory at the University of Waikato. In Māori culture there are several kinds of *toki*, with different purposes. This one is a ceremonial adze, *toki pou tangata*, a symbol of chieftainship. The *rau* (blade) is sharp, hard, and made of *pounamu* or greenstone—hence the Greenstone software, at the cutting edge of digital library technology. There are three figures carved into the *toki*. The forward-looking one looks out to where the *rau* is pointing to ensure that the *toki* is appropriately targeted. The backward-looking one at the top is a sentinel that guards where the *rau* can't see. There is a third head at the bottom of the handle which makes sure that the chief's decisions—to which the *toki* lends authority—are properly grounded in reality. The name of this *taonga*, or art-treasure, is *Toki Pou Hinengaro*, which translates roughly as "the adze that shapes the excellence of thought." *Haramai te toki, haumi e, hui e, tāiki e.*

Building Digital Library Collections with Greenstone

Workshop

Ian H. Witten ihw@cs.waikato.ac.nz

http://nzdl.org
http://greenstone.org

Course material prepared by

Greenstone Digital Library Project
University of Waikato, New Zealand

What we wanted

- "Collections" of digital material
- Individualized, depending on metadata etc
- Up to several Gb of text …
- … + associated images, movies, whatever
- Fully searchable
- Served on WWW, or published on removable media
- Run anywhere, on any computer
- Fully internationalized
- Non-exclusive: documents and metadata in any format
- Non-prescriptive: standard and non-standard metadata

What we got: Greenstone

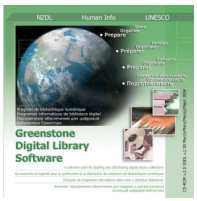
Access
- Accessible via any Web browser
- Server runs on anything (all Windows + Unix+Mac)
- Collections can be published on CD-ROM/DVD
- Trivial to install
- GUI interface for building and publishing collections

Searching browsing
- Collection-specific
- Full-text and fielded search
- Flexible browsing facilities
- Metadata-based (Dublin Core recommended)
- Creates all access structures automatically

Extensible
- Plugins — new document, metadata formats
- Classifiers — new metadata browsers

Multi-*
- Multilingual: Documents *and* interfaces
- Multimedia: image, video, audio collections exist
- Multiformat: Documents *and* metadata

UNESCO: Distributing Greenstone DL software



Sustainable development

"Give a man a fish, feed him for a day
Teach a man to fish, feed him for life"

Greenstone software on CD-ROM

- **GNU licensed**
- **Fully documented … in English/French/Spanish/Russian**
- **Language interfaces … Arabic Chinese Czech … Thai Turkish**
- **Unix/Windows/Mac OS-X**
- **Trivial to install**
- **GUI interface for gathering, enriching, building …**
- **Serve collections on Web or write them to CD-ROM**
- **Document formats: HTML, Word, PDF, PS, plain text, e-mail**
- **Metadata formats: XML, DC, OAI, MARC, …**

download from *http://greenstone.org*

Greenstone facts

Distribution
- Open source: Gnu GPL
- Distributed via SourceForge since: Nov 2000
- Average downloads: 5000/month since then
- Humanitarian CD-ROMs produced: 40-45
- Distribution for each one: 5000/year

International
- Languages for interface: 42
- Languages for full software + manuals: 4
- Countries represented on email lists: 60
- UNESCO training courses in: Bangalore, Almaty, Dakar, Suva, …

UN Agencies
- UNESCO, Paris ("Information for All" programme)
- FAO, Rome (Info Management Resource Kit)
- UNU, Japan (CD-ROM collections of UNU material)

Technical centers
- University of Waikato, New Zealand
- Indian Institute of Sciences, Bangalore
- University College, London
- University of Cape Town, South Africa
- University of Lethbridge, Canada

Standards

Metadata
- Can use any metadata set, Dublin Core supplied
- Plugins for

XML	**Refer**
MARC	**OAI**
CDS/ISIS	**METS (subset)**
ProCite	**DSpace**
BibTex	

- METS can be used as Greenstone's internal representation

Serving
- Web
- Can publish Greenstone collections on CD-ROM
- Can publish Greenstone collections on OAI
- Export collections to METS
- Export collections to DSpace (*ready for DSpace's batch import program*)

Documents
- Plugins for

PDF	**ZIP**	**Images** (any format: GIF, JPEG, TIFF …)
PostScript	**Excel**	**MP3**
Word, RTF	**PPT**	**Ogg Vorbis**
HTML	**Email**	**UnknownPlug** (e.g. for audio, MPEG, Midi)
Plain text	**Source code**	
Latex		

Agenda

Getting started
Install Greenstone
What does it do?
Platforms and versions
Documentation and help
Example collections

Building collections
Small HTML collection
Collection of Word and PDF files
Simple image collection
Dublin Core metadata
Large HTML collection
Bibliographic collection

Customizing
Formatting the HTML collection
Exploding bibliographic databases
Multimedia collection

Advanced stuff
Formatting
Collection-building process
Macro files and stylesheets
Personalizing your home page
Join the Greenstone community

Greenstone: Platforms

- Operating system:
  - Windows (any version)
  - Linux (any version)
  - Unix (most versions, e.g. Solaris)
  - Mac OS X (some problems with GLI interface)
- Restrictions:
  - For Librarian interface (GLI), need Java— which is no longer supported on Windows 95
- Disk space
  - 50 MB for a binary installation
  - 215 MB for the example collections (optional)
  - 5 MB for online documentation
  - 25 MB for "export to CD" function

Local library vs Web library

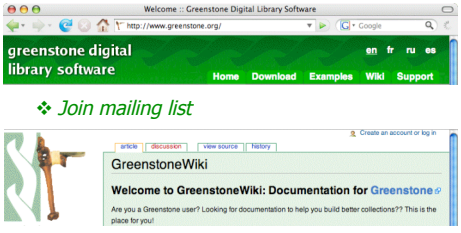
- Local library: stand-alone
  - Serves collections on a standalone PC …
  - And on others on the same network
  - Includes built-in Web server
- Web library: uses external web server
  - Apache, Microsoft PWS/IIS
  - Need to configure web server (geeky job?)
- Windows: Both local library and web library
  - All versions: 95, 98, NT, 2000, ME, XP
  - Binaries supplied
  - Normally use local library (else must set up server)
  - Web library works with Microsoft PWS, IIS
- Unix, Mac OS/10: Web library only
  - Use Apache (or other web server)
  - Linux binaries supplied
  - Tested on SUN Solaris, Mac OS/10
  - Need GDBM (standard on Linux)
- Greenstone developed on Linux

Documentation and Help

Manuals on the CD-ROM (docs)

- *Installer's Guide* (install.pdf, 36pp)
  Versions of Greenstone, installation procedure, Greenstone collections, setting up the web server, configuring your site, personalizing your installation
- *User's Guide* (user.pdf, 90pp)
  Overview of Greenstone, using Greenstone collections, the collector, administration, software features, glossary of terms
- *Developer's Guide* (develop.pdf, 113pp)
  Understanding the collection building process, getting the most out of your collections, the Greenstone runtime systems, configuring your Greenstone site
- *From Paper To Collection* (paper.pdf, 30pp)
  Scanners and scanning, OCR, 3 examples – from 1,000 to 100,000 pages, Creating an electronic collection

Documentation and Help



- *Join mailing list*
- *FAQ*: sections on
  - Obtaining Greenstone
  - Installing Greenstone
  - Running Greenstone
- *Tutorial exercises*
  - Installing Greenstone
  - Updating a Greenstone installation

Example collections

development library subset
The DLS collection has the same structure as the Greenstone demo collection. It's fairly complex, and if you're just starting out you might prefer to look at some other collections first (e.g. msword and pdf demonstration, or the greenstone archives, or the simple image collection).

msword and pdf demonstration
This contains a few documents in PDF, MSWord, RTF, and Postscript formats, demonstrating the ability to build collections from documents in different formats. The collection configuration file is very simple.

greenstone archives
A collection of email messages from the Greenstone mailing list archives, this uses the Email plugin, which parses files in email formats. The collection configuration file is very simple.

bibliography collection
With about 4,000 bibliography entries, this collection incorporates a form-based search interface that allows fielded searching. It is fairly complex.

bibliography supplement
This tiny collection of 10 bibliography entries illustrates the "supercollection" facility which searches several collections together, seamlessly. It operates together with the Bibliography collection, and its configuration file is almost the same.

MARC example
Based on some MARC records from the Library of Congress, this is a simple collection (and does not allow form-based searching).

Example collections

OAI example	Using the Open Archive Protocol and the Import-From feature, this retrieves metadata from an archive and builds a collection from the records. In this case they are images, so both the OAI and Image plugins are used.
simple image collection	This very basic image collection contains no text and no explicit metadata -- which makes it rather unrealistic. The configuration file is about as simple as you can get.
authentication and formatting demo	With the same material as the original Greenstone demo collection, this shows off two independent features: non-standard document formatting, and controlled access to the documents via user authentication.
garish demo	This collection also contains the same material as the Greenstone demo. Its appearance has been altered to show how the pages generated can be set out differently. It relies on a non-standard macro file that is supplied with Greenstone.
CDS/ISIS example	This collection is built from a CDS/ISIS database of about 150 bibliography entries. It uses the ISISPlug plugin, which reads the standard ISIS .mst and .fdt files and converts them to Greenstone metadata.

Agenda

Getting started
- Install Greenstone
- What does it do?
- Platforms and versions
- Documentation and help
- Example collections

Building collections
- Small HTML collection
- Collection of Word and PDF files
- Simple image collection
- Dublin Core metadata
- Large HTML collection
- Bibliographic collection

Customizing
- Formatting the HTML collection
- Exploding bibliographic databases
- Multimedia collection

Advanced stuff
- Formatting
- Collection-building process
- Macro files and stylesheets
- Personalizing your home page
- Join the Greenstone community

Tutorial exercise 1.1
Small collection of HTML files

Invoke GLI: build a small collection of HTML files
- ❖Gather
- ❖Create
- ❖Look at extracted metadata
- ❖Set up shortcut in the Librarian interface

Tutorial exercise 1.2
Simple image collection

Gather
Create
Enrich: Add dc.Title and dc.Description metadata
Create
Design: Change format to display new metadata
Design: Change the size of image thumbnails
Design: Add browser for dc.Description (AZList)
Create
Design: Add searchable index based on dc.Description
Create

Dublin Core Metadata

The basics: 15 elements

Content
Responsibility
Manifestation

Metadata	Definition
Title	The name given to the resource by the creator or publisher
Creator	The person or organization primarily responsible for the intellectual content of the resource
Subject	The topic of the resource
Description	A textual description of the content of the resource
Publisher	The entity responsible for making the resource available
Contributor	A person or organization (other than the Creator) who is responsible for making significant contributions to the intellectual content of the resource
Date	A date associated with the creation or availability of the resource
Type	The nature or genre of the content of the resource
Format	The physical or digital manifestation of the resource
Identifier	An unambiguous reference that uniquely identifies the resource within a given context
Source	A reference to a second resource from which the present resource is derived
Language	The language of the intellectual content of the resource
Relation	A reference to a related resource, and the nature of its relationship
Coverage	Spatial locations and temporal durations characteristic of the content of the resource
Rights	Information about rights held in the resource

Tutorial exercise 1.3
Collection of Word and PDF files

Gather
Create
Enrich: View the extracted metadata

Tutorial exercise 1.4
Exporting a collection to CD-ROM/DVD

Tutorial exercise 2.1
Large HTML collection— Tudor

Gather
Create
Design: Extract more metadata from the HTML
Create
Design: Blocking the stray images
Create
Gather: Looking at different views of the files

Tutorial exercise 2.2
Enhanced HTML collection— Tudor

Enrich: Add hierarchically-structured metadata
Design: Add a Hierarchy classifier
Create
Design: Add a PHIND classifier
Create
Design: Partition the full-text index
Create
Create: Controlling the building process

Tutorial exercise 3.1
Bibliographic collection—Part A

- **Using fielded searching**
- **Not exploding/formatting**

Tutorial exercise 4.1
Formatting the HTML collection— Tudor

Format: Simplify the default format statement
Format: Formatting the Hierarchy classifier
Format: Formatting the bookshelves

Tutorial exercise 4.2
Formatting the Word and PDF collection

Format: Simplify the default format statement
Linking to Greenstone/original version of document
Making bookshelves show how many items
Displaying multivalued metadata
Advanced multivalued metadata

Tutorial exercise 3.1
Bibliographic collection—Part B

- Using fielded searching
- Exploding the database
- Reformatting the collection to use the exploded metadata

Tutorial exercise 3.2
CDS/ISIS collection

- Building a collection from a CDS/ISIS database
- Adding indexes, browsing classifiers
- Reformatting it

Tutorial exercise 6.2
Building a multimedia collection

- Manually correcting metadata
- Browsing by media type
- Suppressing dummy text
- Using AZCompactList rather than AZList
- Making bookshelves show how many items they contain
- Adding a PHIND phrase browser
- Branding the collection with an image
- Using UnknownPlug
- Cleaning up a title browser using regular expressions
- Using non-standard macro files
- Using different icons for different media types
- Changing the collection's background image
- Building a full-size version of the collection
- Adding an image collage browser

Site-wide formatting options

`gsdl/etc/main.cfg`

Enable or disable
- Depositor
- GLI applet
- Usage logging

Macro files to be loaded

Interface languages supported

Default values for cgi arguments

Collection-specific formatting options

`gsdl/collect/<collname>/etc/collect.cfg`
(or using GLI)

- DocumentImages
- DocumentTitles
- DocumentHeading format string
- DocumentContents
- DocumentButtons
- DocumentText format string
- DocumentArrowsTop, DocumentArrowsBottom
- DocumentSearchResultLinks
- DocumentUseHTML [frames for documents]
- NavigationBar
- AllowExtendedOptions

Formatting lists

- VList
- HList
- DateList
- SearchVList
- DocumentVList
- CL1VList
- CL1HList
- CL1DateList

Format strings

- [link][/link]
- [href]
- [srclink][/srclink]
- [icon]
- [srcicon]
- [num]
- [numleafdocs]
- [Text]
- [RelatedDocuments]
- [highlight][/highlight]
- [Summary]
- [DocOID]
- [DocTopOID]
- [DocRank]
- [DocImage]
- [collection]
- [collection:meta-name]
- [metadata-name]

Extended metadata

- [parent:Title]
- [parent(Top):Title]
- [parent(All):Title]
- [parent(All': '):Title]
- [child:Subject]
- [child(All'xxx'):Subject]
- [child(2):Subject]
- [child(last):Subject]
- [sibling:Subject]
- [sibling(All'
'):Subject]
- [sibling(2):Subject]
- [sibling(last):Subject]
- [parent:sibling:Subject]
- [child:sibling:Subject]
- [cgisafe:parent(Top):Title]
- [cgisafe:sibling(All'
'):Subject]
- [format:Date]

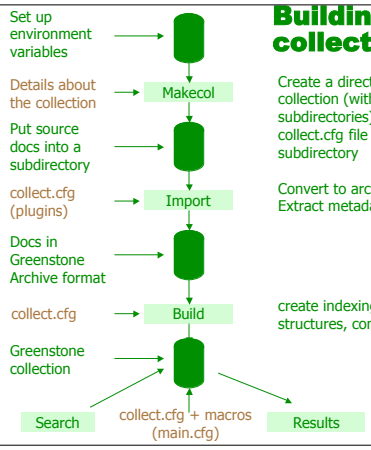
Extended Format String items

- [DocumentButtonDetach]
- [DocumentButtonHighlight]
- [DocumentButtonExpandText]
- [DocumentButtonExpandContents]
- [DocTOC]

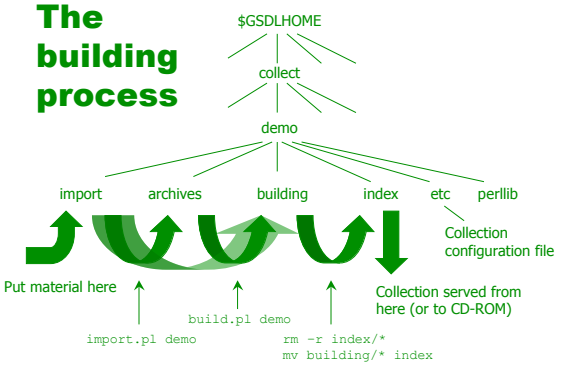
Conditional expressions

- {If}{[metadata], action-if-non-null, action-if-null}
- {If}{[metadata] op value, action-if-true, action-if-not-true
  - eq, ne, gt, ge, lt, le, sw, ew
- {Or}{[metadata1], [metadata2], [metadata3] …}
- Nested If/Or

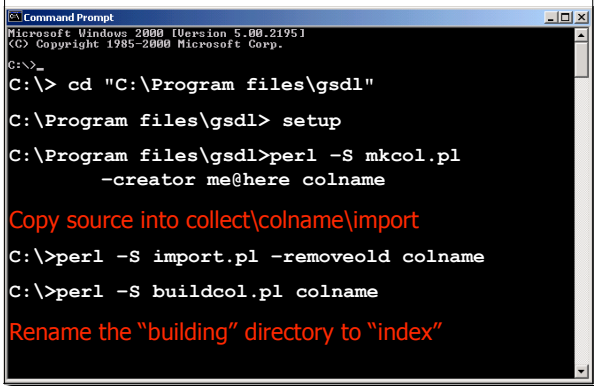
Building a collection

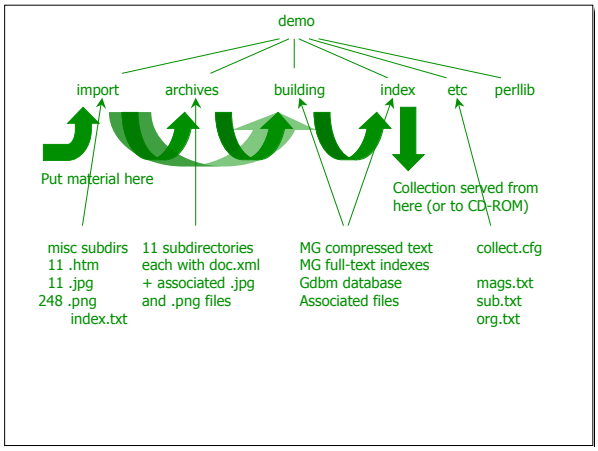


The building process

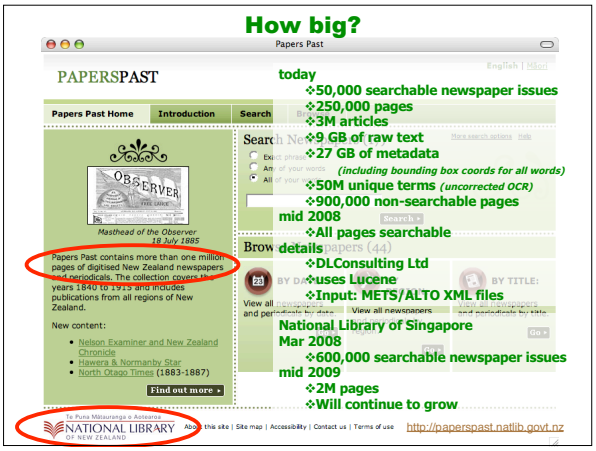


The building process





How big?



today
- 50,000 searchable newspaper issues
- 250,000 pages
- 3M articles
- 9 GB of raw text
- 27 GB of metadata
- *(including bounding box coords for all words)*
- 50M unique terms *(uncorrected OCR)*
- 900,000 non-searchable pages

mid 2008
- All pages searchable

details
- DLConsulting Ltd
- uses Lucene
- Input: METS/ALTO XML files

National Library of Singapore
Mar 2008
- 600,000 searchable newspaper issues

mid 2009
- 2M pages
- Will continue to grow

Collection configuration file

- name, icon, etc
- description
- email of creator
- search indexes
- plugins
- classifiers

how to format
- documents
- query results
- classifiers

```
creator         sjboddie@cs.waikato.ac.nz
maintainer      sjboddie@cs.waikato.ac.nz
public          true
beta            true

indexes         section:text section:Title document:text
defaultindex    section:text

plugin          GAPlug
plugin          ArcPlug
plugin          RecPlug

classify        Hierarchy -hfile sub.txt -metadata Subject -sort Title
classify        HDLList -metadata Title
classify        Hierarchy -hfile org.txt -metadata Organization -sort Title
classify        List -metadata Howto

format          SearchVList "<td valign=top>[link][icon][/link]</td>
                <td>{If}{[parent(All': '):Title],[parent(All': '):Title]: }
                [link][Title][/link]</td>"
format          CL4VList    "<br>[link][Howto][/link]"
format          DocumentImages true
format          DocumentText "<h3>[Title]</h3>\n\n<p>[Text]"

collectionmeta collectionname  "greenstone demo"
collectionmeta collectionextra "This is a demonstration collection for the
                Greenstone digital library software.\nIt contains a small
                subset (11 books) of the Humanity Development Library"
collectionmeta iconcollectionsmall "/gsdl/collect/demo/images/demosm.gif"
collectionmeta iconcollection      "/gsdl/collect/demo/images/demo.gif"
collectionmeta .section:Title      "section titles"
collectionmeta .document:text      "entire books"
collectionmeta .section:text       "chapters"
```

Alter configuration

Add full-text index of titles	indexes document:Title
... or authors	indexes document:Creator
Add alphabetic author browser	classify AZList –metadata Creator
Include Word documents	plugin WordPlug
Include PDF documents	plugin PDFPlug
Separate index for each language	languages en fr es
Extract acronyms and add list	plugin PDFPlug –extract_acronyms
Import OAI metadata	plugin OAIPlug
Extract phrase hierarchy and add browser	classify Phind
Alter the format of any of the above	format ...
Restrict collection's interface langs	format PreferenceLangs en|fr|es
Change default interface language	cgiarg shortname=1 argdefault =fr

Agenda

Getting started
- Install Greenstone
- What does it do?
- Platforms and versions
- Documentation and help
- Example collections

Building collections
- Small HTML collection
- Collection of Word and PDF files
- Simple image collection
- Dublin Core metadata
- Large HTML collection
- Bibliographic collection

Customizing
- Formatting the HTML collection
- Exploding bibliographic databases
- Multimedia collection

Advanced stuff
- Formatting
- Collection-building process
- Macro files and stylesheets
- Personalizing your home page
- Join the Greenstone community

Tutorial exercise 8.1

Customization: macro files and stylesheets

- Changing the background and header images
- Changing the color of the navigation bar, page title and page text
- Adding a footer
- Make your own Greenstone home page
- Collection specific customization
- How to determine which images to replace

Interface in English

Interface in Chinese

Interface language macros

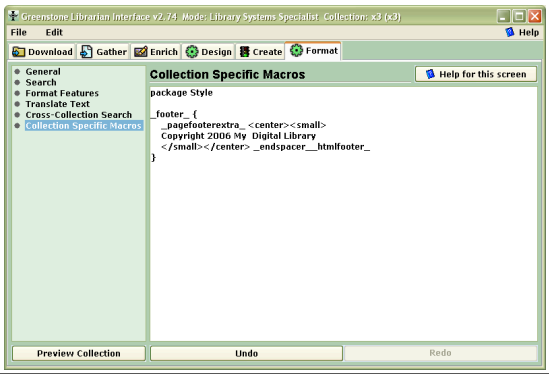
english.dm

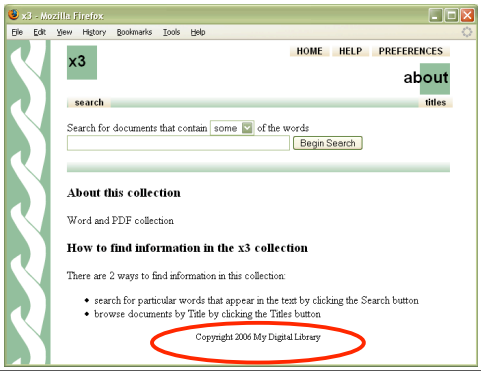
```
#--------------------------------------------------
# text macros
#--------------------------------------------------

_textimageabout_ {About page}
_textimagehelp_ {Help page}
_textimagepref_ {Preferences page}
_textimagegreenstone_ {Greenstone Digital Library Software}

_textimagesearch_ {Search for specific terms}
_textimageCreator_ {Browse alphabetical list of authors}
_textimageTitle_ {Browse alphabetical list of titles}
_textmatches_ {Matches }
_textbeginsearch_ {Begin Search}

_texticonsmalltext_ {View this section of the text}
_texticondetach_ {Open this page in a new window}
_texticonexpandtoc_ {Expand table of contents}
_texticonexpandtext_ {Display all text}
_texticonhighlight_ {Highlight search terms}

_textlanguage_ {Interface language: }
_textsimplemode_ {simple query mode}
_textadvancedmode_ {advanced query mode (allows boolean searching u
```

chinese.dm

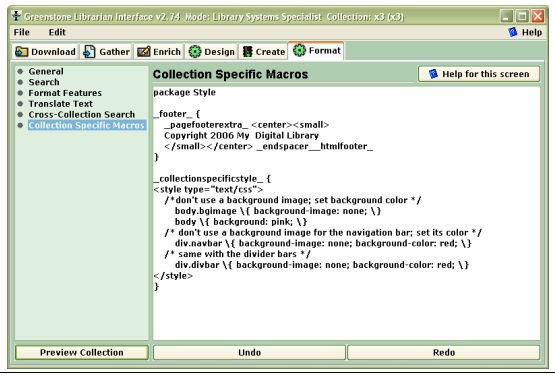
```
#--------------------------------------------------
# text macros
#--------------------------------------------------

_textimageabout_ [l=zh] {主页}
_textimagehelp_ [l=zh] {帮助页}
_textimagepref_ [l=zh] {选项页}
_textimagegreenstone_ [l=zh] {绿石数字化图书馆软件}

_textimagesearch_ [l=zh] {特定范围检索}
_textimageCreator_ [l=zh] {按作者名字的阿拉伯字母顺序浏览 }
_textimageTitle_ [l=zh] {按题名的阿拉伯字母顺序浏览}
_textmatches_ [l=zh] {匹配 }
_textbeginsearch_ [l=zh] {开始查询}

_texticonsmalltext_ [l=zh] {  浏览该部份文本}
_texticondetach_ [l=zh] {  打开一个新的窗口显示本页}
_texticonexpandtoc_ [l=zh] {  扩展书录表}
_texticonexpandtext_ [l=zh] {  显示所有文本}
_texticonhighlight_ [l=zh] {  查询项以粗体显示}

_textlanguage_ [l=zh] {界面语言：}
_textsimplemode_ [l=zh] {简单查询方式}
_textadvancedmode_ [l=zh] {高级查询方式(可以使用 !、&、| 和括号来布尔查询)}
```

Collection-specific macro

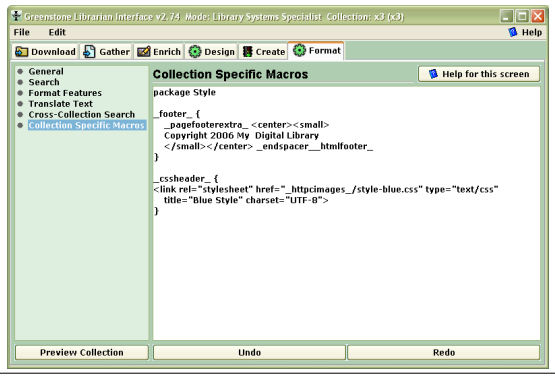


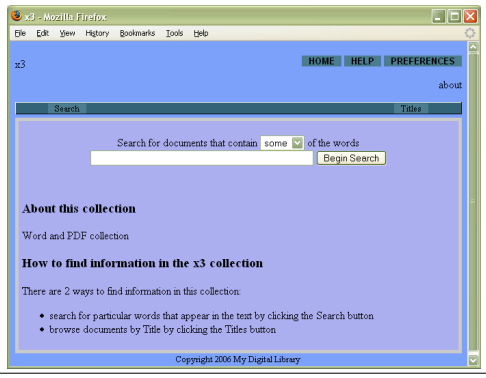


Collection-specific style



Collection-specific stylesheet





style_blue.css

```
body {
 background: #80a0fb; /* light blue */
 color: #000000;
}

a:link { color: #006666; }

a:hover { color: #000066; }

a:visited { color: #666633; }

img.link { border: 0; }

/* for unesco/human info etc logos on home page */
img.logo {
 border: 0;
 float: right;
 margin-left: 10px;
}

div.v_list {
 padding: 10px;
 font-size: smaller;
}
/* specific columns, named in the format string */
div.vlist_entry {
 clear: both; /* comes after any previous floating divs */
 border-bottom: 1px grey solid;
 margin-top: 10px;
 margin-bottom: 5px;
}
```

```
div.vlist_entry div {
 width: 15%;
 padding-left: 1%;
 padding-right: 1%;
 float: left;
}

div.vlist_entry div.descr { width: 45% }
#banner {
 margin-bottom: 5px;
}
/* this div contains bannerlinks and bannerimage paras */
div.pageinfo {
 float: right;
 text-align: right;
}
/* home, help, preferences links */
p.bannerlinks {
 font-family: arial;
 font-size: 10pt;
 font-weight: bold;
}
/* the image based on the page action */
p.bannerimage {
 display: none;
}
/* the image for the collection, shown in the banner */
div.collectimage {
 text-align: left;
 float: left;
 height: 100%;
}
```

Agenda

Getting started
- Install Greenstone
- What does it do?
- Platforms and versions
- Documentation and help
- Example collections

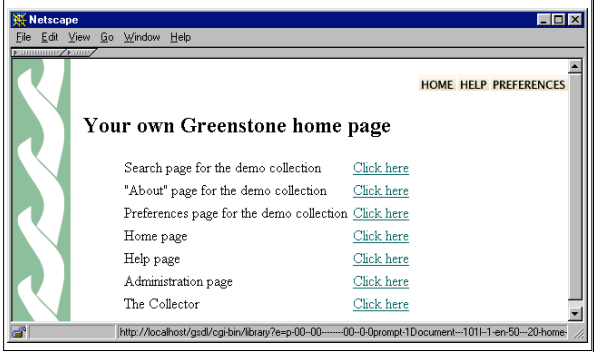
Building collections
- Small HTML collection
- Collection of Word and PDF files
- Simple image collection
- Dublin Core metadata
- Large HTML collection
- Bibliographic collection

Customizing
- Formatting the HTML collection
- Exploding bibliographic databases
- Multimedia collection

Advanced stuff
- Formatting
- Collection-building process
- Macro files and stylesheets
- Personalizing your home page
- Join the Greenstone community

Personalizing your home page

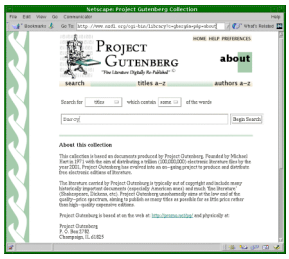
C:\Program Files\gsdl\etc\main.cfg change *home.dm* to *yourhome.dm*



yourhome.dm

```
package home
_content_ {
<h2>Your own Greenstone home page</h2>
<ul>
<table>
<tr valign=top><td>Search page for the demo collection<br></td>
      <td><a href="_httpquery_&c=demo">Click here</a></td></tr>

<tr><td>"About" page for the demo collection</td>
      <td><a href="_httppageabout_&c=demo">Click here</a></td></tr>

<tr><td>Preferences page for the demo collection</td>
      <td><a href="_httppagepref_&c=demo">Click here</a></td></tr>

<tr><td>Home page</td>
      <td><a href="_httppagehome_">Click here</a></td></tr>

<tr><td>Help page</td>
      <td><a href="_httppagehelp_">Click here</a></td></tr>

<tr><td>Administration page</td>
      <td><a href="_httppagestatus_">Click here</a></td></tr>

<tr><td>The Collector</td>
      <td><a href="_httppagecollector_">Click here</a></td></tr>

</table>
</ul>
}
```

about.dm



- can contain conditional statements
  - _If_ ...
- can take arguments
  - a=p, l=de etc.

```
package about

##########################
# about page content
##########################

_pagetitle_ {_collectionname_}

_content_ {
<center>
_navigationbar_
</center>
_query:queryform_
<p>_iconblankbar_
<p>_textabout_
_textsubcollections_
<h3>_help:textsimplehelpheading_</h3>
_help:simplehelp_
}

_textabout_ {
<h3>_textabcol_</h3>
_Global:collectionextra_
}
```

Macro names begin and end with underscore

Content defined in curly brackets

Content is plain text, HTML, other macros, Javascript, applet links

Agenda

Getting started
- Install Greenstone
- What does it do?
- Platforms and versions
- Documentation and help
- Example collections

Building collections
- Small HTML collection
- Collection of Word and PDF files
- Simple image collection
- Dublin Core metadata
- Large HTML collection
- Bibliographic collection

Customizing
- Formatting the HTML collection
- Exploding bibliographic databases
- Multimedia collection

Advanced stuff
- Formatting
- Collection-building process
- Macro files and stylesheets
- Personalizing your home page
- Join the Greenstone community

Join the Greenstone community

- Mailing list
- Wiki
- User-contributed documentation
- Documentation, documentation, documentation
  - new exercises, documented example collections (public-domain material?), fix up exercise text …
- Give workshops!
- Resources:
  - we undertake contracts
  - DLConsulting
- Become a Friend of Greenstone
- Become a Regional Support Center

Greenstone Workshop, 2009

Lab 1 Building collections and adding metadata

1.1. Building a small collection of HTML files

- Running the Greenstone Librarian Interface
- Starting a new collection
- Adding documents to the collection
- Building the collection
- Viewing the extracted metadata
- Viewing the internal links and external links
- Setting up a shortcut in the Librarian interface

1.2. A simple image collection

- Adding Title and Description metadata
- Change Format Features to display new metadata
- Changing the size of image thumbnails
- Adding a browsing classifier based on Description metadata
- Creating a searchable index based on Description metadata

1.3. A collection of Word and PDF files

- Viewing the extracted metadata
- Manually adding metadata to documents in a collection
- Document Plugins
- Search indexes
- Browsing classifiers
- Renaming the search indexes
- Classifying on multiple metadata

1.4. Exporting a collection to CD-ROM/DVD

Lab 2: Designing collections

2.1. A large collection of HTML files—Tudor

- Extracting more metadata from the HTML
- Looking at different views of the files in the **Gather** and **Enrich** panels

2.2. Enhanced collection of HTML files—Tudor

- Adding hierarchically-structured metadata and a **Hierarchy** classifier
- Adding a hierarchical phrase browser (PHIND)
- Partitioning the full-text index based on metadata values
- Controlling the building process

Lab 3: Collections of bibliographic material

3.1. Bibliographic collection

- Using fielded searching
- Exploding the database
- Reformatting the collection to use the exploded metadata

3.2. CDS/ISIS collection

Lab 4: Greenstone formatting exercises

4.1. Formatting the HTML collection—Tudor

4.2. Formatting the Word and PDF collection

Tidying up the default format statement
Linking to Greenstone version or original version of documents
Making bookshelves show how many items they contain
Displaying multi-valued metadata
Advanced multi-valued metadata

Lab 5: Advanced collection configuration

5.1. Pointing to documents on the web

5.2. Enhanced Word document handling

Using Windows native scripting
Modes in the Librarian Interface
Defining styles
Removing pre-defined table of contents
Extracting document properties as metadata

5.3. Enhanced PDF handling

Modes in the Librarian Interface
Splitting PDFs into sections
Using image format
Using **process_exp** to control document processing (advanced)
Opening PDF files with query terms highlighted

5.4. Section tagging for HTML documents

Lab 6: Multimedia

6.1. Looking at a multimedia collection

6.2. Building a multimedia collection

Manually correcting metadata
Browsing by media type
Suppressing dummy text
Using **AZCompactList** rather than **AZList**
Making bookshelves show how many items they contain
Adding a Phind phrase browser
Branding the collection with an image
Using **UnknownPlugin**
Cleaning up a title browser using regular expressions
Using non-standard macro files
Using different icons for different media types
Changing the collection's background image

Building a full-size version of the collection
Adding an image collage browser

Lab 7: Scanned images

7.1. Scanned image collection

Grouping documents by series title and displaying dates within each group
Displaying scanned images and suppressing dummy text
Searching at page level

7.2. Advanced scanned image collection

Adding another newspaper to the collection
XML based item file
Using **process_exp** to control document processing
Switching between images and text

Lab 8: Customization

8.1. Customization: macro files and stylesheets

Collection specific customisation
Changing the colour of the page title and page text
Make your own Greenstone home page
How to determine which images to replace (advanced)

Lab 9: Sharing collections with OAI-PMH

9.1. Open Archives Initiative (OAI) collection

Tweaking the presentation with format statements

9.2. Downloading over OAI

Downloading using the Librarian Interface
Downloading using the command line

Lab 10: Miscellaneous

10.1. Downloading files from the web

10.2. Editing metadata sets

Running GEMS
Creating a new metadata set
Adding a new element to a metadata set

10.3. Building and searching with different indexers

Build with Lucene
Search with Lucene
Build with MGPP
Search with MGPP
Use search mode hotkeys with query term
A quick reference of the search mode hotkeys in MGPP

Lab 1 Building collections and adding metadata

1.1. Building a small collection of HTML files

You will need some HTML files, such as those in the simple_html folder in sample_files.

Running the Greenstone Librarian Interface

1. Start the Greenstone Librarian Interface:

   Start → All Programs → Greenstone-2.81 → Librarian Interface (GLI)

   After a short pause a startup screen appears, and then after a slightly longer pause the main Greenstone Librarian Interface appears. (A command prompt is also opened in the background.)

Starting a new collection

2. Start a new collection within the Librarian Interface:

   File → New...

3. You will create a collection based on a few HTML web pages from the Tudor collection.

   A window pops up. Fill it out with appropriate values—for example,

   Collection title: Small HTML Collection
   Description of content: A small collection of HTML pages.

   Leave the setting for **Base this collection on:**at its default: **-- New Collection --**, and click **<OK>**.

4. Next you must gather together the files that will constitute the collection. A suitable set has been prepared ahead of time in *sample_files → simple_html*. Using the left-hand side of the Librarian Interface's **Gather** panel, interactively navigate to the *sample_files* folder.

Adding documents to the collection

5. Now drag the *simple_html* folder from the left-hand side and drop it on the right. The progress bar at the bottom shows some activity. Gradually, duplicates of all the files will appear in the collection panel.

   You can inspect the files that have been copied by double-clicking on the folder in the right-hand side.

6. Since this is our first collection, we won't complicate matters by manually assigning metadata or altering the collection's design. Instead we rely on default behaviour. So pass directly to the **Create** panel by clicking its tab.

Building the collection

7. To start building the collection, click the **<Build Collection>** button.

8. Once the collection has built successfully, a window pops up to confirm this. Click **<OK>**.

9. Click the **<Preview Collection>** button to look at the end result. This loads the relevant page into your web browser (starting it up if necessary).

Viewing the extracted metadata

10. Back in the Librarian Interface, click the **Enrich** tab to view the metadata associated with the documents in the collection.

11. Presently there is no manually assigned metadata, but the act of building the collection has extracted metadata from the documents. Double click the *simple_html* folder to expand its content. Then single-click *aragon.html* to display all its metadata in the right-hand side of the panel. The initial fields, starting "dc.", are empty. These are Dublin Core metadata fields for manually entered data.

12. Use the scroll bar on the extreme right to view the bottom part of the list. There you will see fields starting "ex." that express the extracted metadata: for example **ex.Title**, based on the text within the HTML Title tags, and **ex.Language**, the document's language (represented using the ISO standard 2-letter mnemonic) which Greenstone determines by analyzing the document's text.

13. Close the collection by clicking **File → Close**. This automatically saves the collection to disk.

Viewing the internal links and external links

14. Hyperlinks in a Greenstone collection work like this. If the link is to a document that is also in the collection, clicking it takes you to that document in the collection. If the link is to a document that is *not* in the collection, clicking it takes you to that document on the web.

    Open *boleyn.html* and look for the link to *Katharine of Aragon* (in the 5th paragraph of the *Biography* section). This links to a document inside the collection--*aragon.html*. View this document by clicking the link. For an external link, click *letters written by Katharine* (in the *Primary Sources* section). This takes you out on to the web. If you want a warning message to be displayed first, you can open *Greenstone → etc → main.cfg* file and uncomment the line `cgiarg shortname=el argdefault=prompt`.

Setting up a shortcut in the Librarian interface

15. To set up a shortcut to the source files, in the **Gather** panel navigate to the folder in your local file space that contains the files you want to use—in our case, the *sample_files* folder. Select this folder and then right-click it, and choose **Create Shortcut** from the menu. In the **Name** field, enter the name you want the shortcut to have, or accept the default *sample_files*. Click **<OK>**. Close all the folders in the file tree in the left-hand pane, and you will see the shortcut to your source files.

1.2. A simple image collection

1. In the Librarian Interface, start a new collection (**File → New...**) called **backdrop**. Fill out the fields with appropriate information. For **Base this collection on:**, select the item **Simple image collection (image-e)** from the pull-down menu.

   When you base a collection on an existing one, it inherits all the settings of the old one, including which metadata sets (if any) the collection uses.

2. Copy the images provided in *sample_files → images* into your newly-formed collection.

3. Change to the **Create** panel and **build** the collection.

4. **Preview** the result.

5. Click on **Browse** in the navigation bar to view a list of the photos ordered by filename and presented as a thumbnail accompanied by some basic data about the image. The structure of this collection is the same as **Simple image collection (image-e)**, but the content is different.

6. Back in the Librarian Interface, change to the **Enrich** panel and view the extracted metadata for *Bear.jpg*.

Adding Title and Description metadata

7. We work with just the first three files (*Bear.jpg*, *Cat.jpg* and *Cheetah.jpg*) to get a flavour of what is possible. First, set each file's **dc.Title** field to be the same as its filename but without the filename extension:

   Click on *Bear.jpg* so its metadata fields are available, then click on its **dc.Title** field on the right-hand side. Type in **Bear**.

   Repeat the process for *Cat.jpg* and *Cheetah.jpg*.

8. Add a description for each image as **dc.Description** metadata.

   What description should you enter? To remind yourself of a file's content, the Librarian Interface lets you open files by double-clicking them. It launches the appropriate application based on the filename extension, Word for .doc files, Acrobat for .pdf files and so on.

   Double-click *Bear.jpg*: on Windows, the image will normally be displayed by Microsoft's Photo Editor (although this depends on how your computer has been set up).

   Back in the **Enrich** pane, make sure that *Bear.jpg* is selected in the collection tree on the left hand side. Enter the text **Bear in the Rocky Mountains** as the value for the **dc.Description** field.

   Repeat this process for *Cat.jpg* and *Cheetah.jpg*, adding a suitable description for each.

9. Go to the **Create** panel and click **<Build Collection>**. Once it has finished building, **preview** the

collection. You will not notice anything new. That's because we haven't changed the design of the collection to take advantage of the new metadata.

Change Format Features to display new metadata

10. Now we customize the collection's appearance. Go to the **Format** panel and select **Format Features** from the left-hand list. Leave the feature selection controls at their default values, so that **All Features** is selected for **Choose Feature**, and **VList** is selected as the **Affected Component**. In the **HTML Format String**, edit the text as follows:

    - Change `_ImageName_:` to `Title:`
    - Change `[Image]` to `[dc.Title]`
    - After `[dc.Title]<br>` add `Description: [dc.Description]<br>`

    Metadata names are case-sensitive in Greenstone: it is important that you capitalize "Title" and "Description" (and don't capitalize "dc").

11. The new format statement is displayed in the list of assigned format statements. The first substitution alters the fragment of text that appears to the right of the thumbnail image, the second alters the item of metadata that follows it. The addition displays the description after the Title.

12. Preview the collection by clicking the **<Preview Collection>** button. When you click on **Browse** in the navigation bar the presentation has changed to "Title: Bear" and so on. Each image's description should appear beside the thumbnail, following the title.

After the first three items, the Title and Description become blank because we have only assigned Dublin Core metadata to these first three. To get a full listing, enter all the metadata.

*Changes in the **Format** panel take place immediately and you can see the result straightaway by clicking the **Preview Collection**. If you modify anything in the **Gather**, **Enrich** or **Design** panels, you will need to rebuild the collection.*

Changing the size of image thumbnails

13. Lets change the size of the thumbnail image and make it smaller. Thumbnail images are created by the **ImagePlugin** plug-in, so we need to access its configuration settings. To do this, switch to the **Design** panel and select **Document Plugins** from the list on the left. Double-click **ImagePlugin** to pop up a window that shows its settings. (Alternatively, select **ImagePlugin** with a single click and then click **<Configure Plugin...>** further down the screen). Currently all options are off, so standard defaults are used. Select **thumbnailsize**, set it to **50**, and click **<OK>**.

14. **Build** and **preview** the collection.

15. Once you have seen the result of the change, return to the **Design** panel, select the configuration options for **ImagePlugin**, and switch the **thumbnailsize** option off so that the thumbnail reverts to its normal size when the collection is re-built.

Adding a browsing classifier based on Description metadata

16. Now we'll add a new browsing option based on the descriptions. In the **Design** panel, select

Browsing Classifiers from the left-hand list. Set the menu item for **Select classifier to add** to **AZList**; then click <**Add Classifier...**>.

17. A window pops up to control the classifier's options. Set the **metadata** option to **dc.Description** and click <**OK**>.

18. **Build** the collection, and **preview** it. Choose the new **Descriptions** link that appears in the navigation bar.

*Only three items are shown, because only items with the relevant metadata (**dc.Description** in this case) appear in the list. The original browse list includes all photos in the collection because it is based on **ex. Image**, extracted metadata that reflects an image's filename, which is set for all images in the collection.*

Creating a searchable index based on Description metadata

19. Now we'll add an index so that the collection can be searched by descriptions. Switch to the **Design** panel and select **Search Indexes** from the left-hand list. Click the <**New Index**> button. Select **dc.Description** from the list of metadata to include in the index, leave **Indexing level:** at its default, "document", and click <**Add Index**>.

20. Switch to the **Create** panel, **build** the collection, then **preview** it. There is now a **Search** button in the navigation bar. As an example, search for the term "bear" in the *document:dc.Description* index (which is the only index at this point).

21. To change the text that is displayed for the index (document:dc.Description), go to the **Format** panel back in the Librarian Interface. Select **Search** from the left-hand list. This panel allows you to change the text that is displayed on the search form. Change the **Display text** for the **document:dc.Description** index to "descriptions" (or other suitable text). Go back to the browser and reload the search page. Your new text will appear in the search form.

1.3. A collection of Word and PDF files

You will need some source files like those in the sample_files → Word_and_PDF folder.

1. Start a new collection called **reports** (**File → New...**) and base it on **-- New Collection --**.

2. Copy all the .doc, .rtf, .pdf and .ps files from *sample_files → Word_and_PDF → Documents* into the collection. There are 9 files in all: you can select multiple files by clicking on the first one and shift-clicking on the last one, and drag them all across together. (This is the normal technique of multiple selection.)

3. Switch to the **Create** panel, and **build** and **preview** the collection.

Viewing the extracted metadata

4. Again, this collection contains no manually assigned metadata. All the information that appears—title and filename—is extracted automatically from the documents themselves. Because of this the quality of some of the title metadata is suspect.

5. Back in the Librarian Interface, click the **Enrich** tab to view the automatically extracted metadata. You will need to scroll down to see the extracted metadata, which begins with "ex.".

6. Check whether the **ex.Title** metadata is correct for some of the documents by opening them. You can open a document from the Librarian Interface by double clicking on it.

7. The extracted Title metadata for some documents is incorrect. For example, the Titles for *pdf01.pdf* and *word03.doc* (the same document in different formats) have missed out the second line. The Title for *pdf03.pdf* has the wrong text altogether.

Manually adding metadata to documents in a collection

8. In the **Enrich** panel, manually add Dublin Core **dc.Title** metadata to those documents which have incorrect **ex.Title** metadata. Select *word03.doc* and double-click to open it. Copy the title of this document ("Greenstone: A comprehensive open-source digital library software system") and return to the Librarian Interface. Scroll up or down in the metadata table until you can see **dc.Title**. Click in the value box and paste in the metadata.

9. Now add **dc.Creator** information for the same document. You can add more than one value for the same field: when you press **Enter** in a metadata value field, a new empty field of the same type will be generated. Add each author separately as **dc.Creator** metadata.

10. Close the document (in Microsoft Word) when you have finished copying metadata from it. External programs opened when viewing documents must be closed before building the collection, otherwise errors can occur.

11. Next add **dc.Title** and **dc.Creator** metadata for a few of the other documents.

12. You will notice as you add more values, they appear in the **Existing values for ...** box below the metadata table. If you are adding the same metadata value to more than one document, you can

select it from this list. For example, *pdf01.pdf* and *word03.doc* share the same Title; and many documents have common authors.

*If you build and preview your collection at this point, you will see that the **Titles** list now shows your new Titles. However, the **dc.Creator** metadata is not displayed. You need to alter the collection design to use this metadata.*

Document Plugins

13. In the Librarian Interface, look at the **Document Plugins** section of the **Design** panel, by clicking on this in the list to the left. Here you can add, configure or remove plugins to be used in the collection. There is no need to remove any plugins, but it will speed up processing a little. In this case we have only Word, PDF, RTF, and PostScript documents, and can remove the **ZIPPlugin**, **TextPlugin**, **HTMLPlugin**, **EmailPlugin**, **ImagePlugin**, **PowerPointPlugin**, **ExcelPlugin**, **ISISPlug** and **NULPlugin** plugins. To delete a plugin, select it and click **<Remove Plugin>**. **GreenstoneXMLPlugin** is required for any type of source collection and should not be removed.

Search indexes

14. The next step in the **Design** panel is **Search Indexes**. These specify what parts of the collection are searchable (e.g. searching by title and author). Delete the **ex.Source** index, which is not particularly useful, by selecting it and clicking **<Remove Index>**.

15. Modify the **ex.Title** index to include **dc.Title** by selecting the index in the **Assigned Indexes** box and clicking **<Edit Index>**. Select **dc.Title** from the list of metadata, and click **<Replace Index>**. Searching this index will search both **dc.Title** and **ex.Title** metadata. If you want to restrict searching to just the manually added **dc.Title** metadata, edit the index again and deselect **ex.Title** from the list of metadata.

16. You can add indexes based on any metadata. Add a new index based on **dc.Creator** by clicking **<New Index>**. Select **dc.Creator** in the list of metadata, and click **<Add Index>**.

Browsing classifiers

17. The **Browsing Classifiers** section adds "classifiers," which provide the collection with browsing functions. Go to this section and observe that Greenstone has provided two classifiers, *AZLists* based on **ex.Title** and **ex.Source** metadata. These correspond to the *Titles* and *Filenames* buttons on the collection's access bar.

    Remove the **ex.Source** classifier by selecting it and clicking **<Remove Classifier>**.

18. Modify the **ex.Title** classifier to use **dc.Title** instead. Select the classifier and click **<Configure Classifier...>**. In the **metadata** box, select **dc.Title** instead of **ex.Title**. Click **<OK>**.

19. Now add an **AZCompactList** classifier for **dc.Creator**. Select **AZCompactList** from the **Select classifier to add** drop-down list and click **<Add Classifier...>**. A popup window **Configuring Arguments** appears. Select **dc.Creator** from the **metadata** drop-down list and click **<OK>**.

    AZCompactList is like **AZList**, except that values that appear multiple times in the hierarchy are automatically grouped together and a new node, shown as a bookshelf icon, is formed.

20. Switch to the **Create** panel, and **build** and **preview** the collection.

21. Check that all the facilities work properly. There should be three full-text indexes, called *text*, *dc.Title* (or *dc.Title,Title* if you didn't deselect **ex.Title** in the search indexes step above), and *dc.Creator*. The *Titles* list should display all the documents to which you have assigned **dc.Title** metadata (and only those documents). The *Creators* list should show one bookshelf for each author you have assigned as **dc.Creator**, and clicking on that bookshelf should take you to all the documents they authored.

Renaming the search indexes

22. The default display text for the indexes in the drop-down list on the search page contains the content of the index. Now we will change this display text to make it nicer. Go to the **Format** panel by clicking its tab. This panel is split into several sections, each controlling some aspect of collection presentation.

23. Select **Search** in the left hand list. This section allows you to modify what text is displayed for the drop-down lists in the search form (indexes, subcollections, levels etc). Set the **Display text** for the **dc.Title** (or **dc.Title,Title** if you didn't deselect **ex.Title** in the search index) index to be "titles", and that for the **dc.Creator** index to be "creators". Preview the collection by clicking the **Preview Collection**. The search form should display the new text.

Classifying on multiple metadata

24. The new *Titles* list shows only those documents which have been assigned **dc.Title** metadata. For many documents, extracted Titles may be fine, and it is impractical to add the same metadata again as **dc.Title**. Fortunately there is a way we can use both metadata types in one classifier: specify a list of metadata names in the classifier.

25. In the **Browsing Classifiers** section of the **Design** panel, select the **AZList** for **dc.Title** in the **Assigned Classifiers** box and click **<Configure Classifier...>**. Note you can achieve the same result by double clicking on the classifier.

26. In the **metadata** field, type ",ex.Title" after the "dc.Title"—i.e. make it read

    ```
    dc.Title,ex.Title
    ```

27. If you have already done the **Enhanced Word document handling** exercise, some of the documents will have extracted ex.Creator metadata, and some will have dc.Creator. To use both of these in the Creators classifier, make a similar change to the **AZCompactList**: make the **metadata** field read `dc.Creator,ex.Creator`.

    Build the collection again and **preview** it. Now all of the documents should appear in the *Titles* list (and extracted Creators should appear in the *Creators* list).

We will play around with the format statements and customize the outlook of this collection in the ***Formatting the Word and PDF collection*** *exercise.*

1.4. Exporting a collection to CD-ROM/DVD

*To publish a collection on CD-ROM or DVD, Greenstone's Export to CD-ROM export module must be installed. This is included with CD-ROM distributions, and all distributions 2.70w and later. It must be installed separately for non-CD-ROM versions of Greenstone, version 2.70 and earlier (see **Installing Greenstone**).*

1. Launch the Greenstone Librarian Interface if it is not already running.

2. Choose **File → Write CD/DVD image...**. In the resulting popup window, select the collection or collections that you wish to export by ticking their check boxes. You can optionally enter a name for the CD-ROM: this is the name that will appear in the menu when the CD-ROM is run. If a name is not entered, the default **Greenstone Collections** will be used. You can also specify whether the resulting CD-ROM will install files onto the host machine when used or not. Click **<Write CD/DVD image>** to start the export process.

   The necessary files for export are written to:

   Greenstone → tmp → exported_xxx

   where xxx will be similar to the name you have entered. If you didn't specify a name for the CD-ROM, then the folder name will be *exported_collections*.

   You need to use your own computer's software to write these on to CD-ROM. On *Windows XP* this ability is built into the operating system: assuming you have a CD-ROM or DVD writer insert a blank disk into the drive and drag the *contents* of *exported_xxx* into the folder that represents the disk.

   The result will be a self-installing Windows Greenstone CD-ROM or DVD, which starts the installation process as soon as it is placed in the drive.

Lab 2: Designing collections

2.1. A large collection of HTML files—Tudor

You will need the files in the sample_files → tudor folder.

1. Invoke the Greenstone Librarian Interface (from the Windows *Start* menu) and start a new collection called **tudor** (use the **File** menu), based on the default **-- New Collection --**.

2. In the **Gather** panel, open the *tudor* folder in *sample_files*.

3. Drag *englishhistory.net* from the left-hand side to the right to include it in your **tudor** collection. (This material is from Marilee Hanson's Tudor England Collection at http://englishhistory.net/tudor.html, distributed with her permission.)

4. Switch to the **Create** panel and click <**Build Collection**>.

5. When building has finished, **preview** the collection.

Extracting more metadata from the HTML

6. The browsing facilities in this collection (*Titles* and *Filenames*) are based entirely on extracted metadata. Return to the **Enrich** panel in the Librarian Interface and examine the metadata that has been extracted for some of the files.

7. Many HTML documents contain metadata in `<meta>` tags in the `<head>` of the page. Open up the *englishhistory.net → tudor → monarchs → boleyn.html* file by navigating to it in the tree on the left hand side, and double clicking it. This will open it in a web browser. View the HTML source of the page (**View → Source** in Internet Explorer, **View → Page Source** in Mozilla). You will notice that this page has *page_topic*, *content* and *author* metadata.

8. By default, **HTMLPlugin** only looks for Title metadata. Configure the plugin so that it looks for the other metadata too. Switch to the **Design** panel and select the **Document Plugins** section. Select the **plugin HTMLPlugin** line and click <**Configure Plugin...**>. A popup window appears. Switch on the **metadata_fields** option, and set the value to

   ```
   Title,Author,Page_topic,Content
   ```

   Click <**OK**>.

9. Switch to the **Create** panel and **rebuild** the collection. Go back to the **Enrich** panel and look at the extracted metadata for some of the HTML files in *englishhistory.net → tudor → monarchs*. The new metadata should now be visible.

Looking at different views of the files in the Gather and Enrich panels

10. Switch to the **Gather** panel and in the right-hand side open *englishhistory.net → tudor*.

11. Change the **Show Files** menu for the right-hand side from **All Files** to **HTM & HTML**. Notice the files displayed above are filtered accordingly, to show only files of this type.

12. Change the **Show Files** menu to **Images**. Again, the files shown above alter.

13. Now return the **Show Files** setting back to **All Files**, otherwise you may get confused later. Remember, if the **Gather** or **Enrich** panels do not seem to be showing all your files, this could be the problem.

2.2. Enhanced collection of HTML files—Tudor

We return to the Tudor collection and add metadata that expresses a subject hierarchy. Then we build a classifier that exploits it by allowing readers to browse the documents about Monarchs, Relatives, Citizens, and Others separately.

Adding hierarchically-structured metadata and a Hierarchy classifier

1. Open up your **tudor** collection (the original version, not the **webtudor** version), switch to the **Enrich** panel and select the *citizens* folder (a subfolder of *englishhistory.net* → *tudor*). Set its **dc.Subject and Keywords** metadata to **Tudor period|Citizens**. The vertical bar ("|") is a hierarchy marker. Selecting a *folder* and adding metadata has the effect of setting this metadata value for all files contained in this folder, its subfolders, and so on. A popup alerts you to this fact. Click **<OK>** to close the popup.

2. Repeat for the *monarchs* and *relative* folders, setting their **dc.Subject and Keywords** metadata to **Tudor period|Monarchs** and **Tudor period|Relatives** respectively. Note that the hierarchy appears in the **Existing values for dc.Subject and Keywords** area.

   If you don't want to see the popup each time you add folder level metadata, tick the **Do not show this warning again** checkbox; it won't be displayed again.

3. Finally, select all remaining files—the ones that are not in the *citizens*, *monarchs*, or *relative* folders—by selecting the first and shift-clicking the last. Set their **dc.Subject and Keywords** metadata to **Tudor period|Others**: this is done in a single operation (there is a short delay before it completes).

   When multiple files are selected in the left hand collection tree, all metadata values for all files are shown on the right hand side. Items that are common to all files are displayed in black—e.g. **dc.Subject and Keywords**—while others that pertain to only one or some of the files are displayed in grey—e.g. any extracted metadata.

   Metadata inherited from a parent folder is indicated by a folder icon to the left of the metadata name. Select one of the files in the *relative* folder to see this.

4. Switch to the **Design** panel and select **Browsing Classifiers** from the left-hand list. Set the menu item for **Select classifier to add** to **Hierarchy**; then click **<Add Classifier...>**.

5. A window pops up to control the classifier's options. Change the **metadata** to **dc.Subject and Keywords** and then click **<OK>**.

6. For tidiness' sake, **remove** the **classifier** for **Source** metadata (included by default) from the list of currently assigned classifiers, because this adds little to the collection.

7. Now switch to the **Create** panel, **build** the collection, and **preview** it. Choose the new **Subjects** link that appears in the navigation bar, and click the bookshelves to navigate around the four-entry hierarchy that you have created.

Adding a hierarchical phrase browser (PHIND)

Next we'll add an interactive hierarchical phrase browsing classifier to this collection.

8. Switch to the **Design** panel and choose the **Browsing Classifiers** item from the left-hand list.

9. Choose **Phind** from the **Select classifier to add** menu. Click <**Add Classifier...**>. A window pops asking for configuration options: leave the values at their preset defaults (this will base the phrase index on the full text) and click <**OK**>.

10. **Build** the collection again, **preview** it, and try out the new **Phrases** option in the navigation bar. An interesting PHIND search term for this collection is "king". Note that even though it is called a phrase browser, only single terms can be used as the starting point for browsing.

Partitioning the full-text index based on metadata values

Next we partition the full-text index into four separate pieces. To do this we first define four subcollections obtained by "filtering" the documents according to a criterion based on their ***dc.Subject and Keywords*** *metadata. Then an index is assigned to each subcollection. This will enable users to restrict a search to a subset of the documents.*

11. Switch to the **Design** panel, and click **Partition Indexes**. This feature is disabled because you are operating in **Librarian** mode (this is indicated in the title bar at the top of the window).

12. Switch to **Library Systems Specialist** mode by going to **Preferences...** (on the **File** menu) and clicking <**Mode**>. Read about the other modes too.

13. Return to the **Partition Indexes** section of the **Design** panel. Ensure that the **Define Filters** tab is selected (the default). Define a subcollection filter with name **monarchs** that matches against **dc. Subject and Keywords**, and type **Monarchs** as the regular expression to match with. Click <**Add Filter**>. This filter includes any file whose **dc.Subject and Keywords** metadata contains the word *Monarchs*.

14. Define another filter, **relatives**, which matches **dc.Subject and Keywords** against the word **Relatives**. Define a third and fourth, **citizens** and **others**, which matches it against the words **Citizens** and **Others** respectively.

15. Having defined the subcollection filters, we partition the index into corresponding parts. Click the **Assign Partitions** tab. Select the citizens subcollection and click <**Add Partition**>. Next select monarchs, and click <**Add Partition**>. Repeat for the other two subcollections, so that you end up with four partitions, one based on each subcollection filter.

    The order they appear in the **Assigned Subcollection Partitions** list is the order they will appear in the drop down menu on the search page. You can change the order by using the <**Move Up**> and <**Move Down**> buttons.

16. **Build** and **preview** the collection.

17. The search page includes a pulldown menu that allows you to select one of these partitions for searching. For example, try searching the *relatives* partition for *mary* and then search the *monarchs* partition for the same thing.

18. To allow users to search the collection as a whole as well as each subcollection individually,

return to the **Partition Indexes** section of the **Design** panel and select the **Assign Partitions** tab. Select all four subcollections by either checking their boxes or press the **Select All** button, and click **<Add Partition>**.

19. To ensure that the combined index appears first in the list on the reader's web page, use the **<Move Up>** button to get it to the top of the list here in the **Design** panel. Then **build** and **preview** the collection.

20. Search for a common term (like *the*) in all five index partitions, and check that the numbers of words (not documents) add up.

21. The text in the drop down box on the search page is based on the filters each partition was built on. To change the text that is displayed, go to the **Search** section of the **Format** panel. The single filter partitions have sensible default text, but the combined partition does not. Set the **Display text** for the combined partition to "all". **Preview** the collection.

22. In the Librarian Interface, return to **Librarian** mode, using **Preferences...** (on the **File** menu).

Controlling the building process

Finally we look at how the building process can be controlled. Developing a new collection usually involves numerous cycles of building, previewing, adjusting some enrich and design features, and so on. While prototyping, it is best to temporarily reduce the number of documents in the collection. This can be accomplished through the ***maxdocs*** *parameter to the building process.*

23. Switch to the **Create** panel and view the options that are displayed in the top portion of the screen. Select **maxdocs** and set its numeric counter to **3**. Now **build**.

24. Preview the newly rebuilt collection's **Titles** page. Previously this listed more than a dozen pages per letter of the alphabet, but now there are just three—the first three files encountered by the building process.

25. Go back to the **Create** panel and turn off the **maxdocs** option. **Rebuild** the collection so that all the documents are included.

Lab 3: Collections of bibliographic material

3.1. Bibliographic collection

This exercise looks at using fielded searching in a collection. Fielded searching is best used for metadata rich collections. Here we use bibliographic data in MARC format.

1. Start a new collection called **Papers Bibliography** which will contain a collection of example MARC records of the working papers published at the Computer Science Department, Waikato University. Enter the requested information and base it on **-- New Collection --**.

2. In the **Gather** panel, open the *sample_files → marc* folder, drag *CMSwp-all.marc* into the right-hand pane and drop it there. A popup window asks whether you want to add **MARCPlugin** to the collection to process this file. Click **<Add Plugin>**, because this plugin will be needed to process the MARC records.

3. Now select **Browsing Classifiers** within the **Design** panel and **remove** the default classifier for **Source** metadata.

4. In the **Search Indexes** section, **remove** the **ex.Source** index. In this collection all records are from the same file, so **ex.Source** metadata, which is set to the filename, is not particularly interesting or useful.

5. Switch to the **Create** panel, **build** the collection, and **preview** it. Browse through the *Titles* and view a record or two. Try searching—for example, find items that include **graphics**.

6. Back in the Librarian Interface, go to the **Browsing Classifiers** section of the **Design** panel. Select **AZCompactList** from the **Select classifier to add** drop down menu, and click **<Add Classifier...>**. In the popup window, select **ex.Subject** as the metadata item. Click **<OK>**.

   ***AZCompactList** is like **AZList**, except that terms that appear multiple times in the hierarchy are automatically grouped together and a new node, shown as a bookshelf icon, is formed.*

7. **Build** the collection and **preview** the result.

Using fielded searching

8. Now let's look at fielded searching. In the browser, go to the *PREFERENCES* page. You will notice that there is a **Query style:** option which enables you to switch between "normal" and "fielded" search. Change to fielded search now, press the **set preferences**, and click on the **Search** button to go back to the Search page. The search form has changed to a fielded form.

9. You can specify which search form types are available for a particular collection, and which one is the default, using the **SearchTypes** format statement. In the **Format** panel select **Format Features** from the left-hand list. Select the **SearchTypes** format statement from the list of assigned formats, and set the contents to **form**. This will make only fielded searching available for this collection.

*Search type options include **form** and **plain**. You can specify one or both separated by a comma. If both are specified, the first one is used as the default: this is the one that the user will see when they first enter the collection.*

10. **Preview** the collection again. Notice that the collection's home page no longer includes a query box. (This is because the search form is too big to fit here nicely.) To search, you have to click **Search** in the navigation bar. Note that the *PREFERENCES* page has changed so that the "normal" query style is no longer offered.

11. Look at the search form in the collection. There are two fields that can be searched: *text* and *titles*. Add some more fields to search on by going back to the Librarian Interface.

12. In the **Design** panel, go to the **Search Indexes** section. Add a new index based on **ex.Subject** by clicking **<New Index>**, selecting **ex.Subject** in the list of metadata, and clicking **<Add Index>**.

13. **Rebuild** the collection and **preview** the results. Notice the extra field in the **... in field** drop-down menus in the search form. You can do quite complicated queries by searching for words in different fields at the same time.

14. To change the text that is displayed in the drop-down menus of the search form, go to the **Search** section of the **Format** panel. Here you can change the display text for the indexes.

Exploding the database

15. Go to the **Enrich** panel and try to see the metadata. It doesn't appear! This is because the metadata is associated with records inside the file, not the file itself.

    Metadata file types, such as MARC, CDS/ISIS, BibTex etc. can be imported into Greenstone but their metadata cannot be viewed in the Librarian Interface. To edit any metadata you need to go back to the program that created the file.

    Greenstone provides a way of *exploding* a metadata database so that each record appears as an individual document, with viewable and editable metadata. This process is irreversible: once this step has been done, the database is deleted and can no longer be used in its original program.

16. In the **Gather** panel, you may notice that the MARC database has a different coloured icon to other files. This green icon indicates that a file is a metadata database that can be exploded. Right-click on the file and choose **Explode Metadata Database** from the menu. A new window opens, containing options for the exploding process. A description of each option can be obtained by hovering the mouse over the option.

    Turn on the **metadata_set** option by checking its box. This option indicates which metadata set to explode the metadata into. The default set is the "Exploded Metadata Set"—a metadata set which initially has no elements in it, but will receive a new element for each metadata field retrieved from the database.

17. Click **<Explode>** to start the exploding process. This may take a short while, depending on the size of the database.

18. Once exploding has finished, the MARC database file will have been deleted, and three folders created in its place. These folders contain an empty file for each record in the original database. The metadata for these records can be viewed and edited by switching to the **Enrich** panel.

19. Because the MARC file is no longer present, and the collection contains empty (.nul) files, we need to change the list of plugins. In the **Document Plugins** section of the **Design** panel, remove **MARCPlugin**.

20. **Rebuild** and **preview** the collection. You will notice that the *Subjects* classifier is empty, searching no longer returns any results, and the document display is useless.

    Although the *Titles* classifier was built on **ex.Title**, it still displays the correct titles, but in the **Enrich** panel you can see the **ex.Title** metadata are actually the filenames rather than titles of the MARC records. This is because the default **VList** format uses the **exp.Title** metadata. In the **Format Features** section of the **Format** panel, select **VList** in the list of assigned format statements. The resulting format statement looks like:

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./
    srclink]</td>
    <td valign="top">[highlight]
    {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
    [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}
    ```

    Since there is no **dc.Title** metadata and **exp.Title** comes before **ex.Title**, the exploded titles will be displayed.

Reformatting the collection to use the exploded metadata

The collection previously used extracted (ex.) metadata, but now it uses exploded (exp.) metadata. The *Subjects* classifier and search indexes were built on ex metadata, which is why they no longer work properly.

There is also no longer any text in the documents. Previously, **MARCPlugin** stored the raw record as the "text" of each record. Now that the metadata is in the Librarian Interface, there is no longer the concept of raw record, and so there is no text.

We need to modify the collection design to take note of these changes.

21. In the **Search Indexes** section, change the Title index to use **exp.Title**: select the Title index in the **Assigned Indexes** list and click <**Edit Index**>. Deselect **ex.Title** in the list of metadata, and select **exp.Title**. Click <**Replace Index**>.

22. Remove the **ex.Subject** index by selecting it in the **Assigned Indexes** list and clicking <**Remove Index**>. Add an index on **exp.Subject**: click <**New Index**>, select **exp.Subject** in the metadata list, and click <**Add Index**>.

23. The text index is no longer any use, so remove that index too.

24. To enable combined searching across all indexes at once, click <**New Index**>, tick the **Add combined searching over all assigned indexes (allfields)** checkbox, and click <**Add Index**>. Move this to the top of the list using the <**Move Up**> button, so that it appears first in the drop down list. Click <**Set Default Index**> on the right so that it becomes the default field for searching.

25. To explicitly use the **exp.Title** metadata, in the **Browsing Classifiers** section, change the **ex. Title AZList** to use **exp.Title** metadata. Double click the **ex.Title AZList** in the **Assigned Classifiers** list, and change the **metadata** option to use **exp.Title**. Click <**OK**>. Do the same thing for the Subject **AZCompactList**, changing **ex.Subject** to **exp.Subject**.

26. **Rebuild** and **preview** the collection. The classifiers should be back to normal and searching should now work.

27. In the **Format Features** section of the **Format** panel, select **VList** in the list of assigned format statements.

    - There is no dc metadata for this collection, so replace `{Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}` with `{Or}{[exp.Title],[ex.Title],Untitled}`.
    - There are no source or thumb icons, so remove the second line: `<td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./srclink]</td>`.
    - The ex.Source metadata is set to the nul filename, so remove that from the display: remove `{If}{[ex.Source],<br><i>([ex.Source])</i>}`

    The resulting format statement looks like:

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">[highlight]
    {Or}{[exp.Title],[ex.Title],Untitled}
    [/highlight]</td>
    ```

28. Clear the **DocumentHeading** format statement by selecting it in the list of assigned format statements and deleting the contents in the **HTML Format String**. The record Title will be displayed as part of the **DocumentText** format, so we don't need it here.

29. Next, edit the **DocumentText** format statement. Delete the contents and replace it with

    ```
    <table>
    <tr><td>Title:</td><td>[exp.Title]</td></tr>
    <tr><td>Subject:</td><td>[exp.Subject]</td></tr>
    <tr><td>Publisher:</td><td>[exp.Publisher]</td></tr>
    </table>
    ```

30. The *DETACH* and *NO HIGHLIGHTING* buttons are not very useful for this collection, so lets get rid of them. Edit the **DocumentButtons** format statement to make it empty.

3.2. CDS/ISIS collection

*This exercise is similar to the **Bibliographic collection** exercise, except that a CDS/ISIS database is used instead of a MARC database, and we do not explode the database.*

1. Start a new collection called **ISIS Collection** (base it on **New Collection**).

2. Drag the files from *sample_files → isis* (excluding the *format_tweaks* folder) into the collection.

3. **Build** and **preview** the collection. The default indexes, classifiers and formats are not very useful for this data. There is no metadata searching, and the *Titles* classifier is completely empty. The filenames classifier is useless because all records come from the same file.

4. In the **Search Indexes** section of the **Design** panel, remove the useless Source and Title indexes, and add new indexes for Photographer^all, Country^all and Notes^all metadata.

   CDS/ISIS metadata has subfields, and these are represented using ^.

5. In the **Browsing Classifiers** section, remove the existing (useless) classifiers for **Title** and **Source**, and add a new **AZList** for **Photographer**.

6. **Rebuild** and **preview** the collection.

7. In the **Format Features** section of the **Format** panel, change the **VList** format statement to display **Photographer** and **Notes** metadata. Change it to look like:

```
<td valign=top>[link][icon][/link]</td>
<td valign=top><b>[ex.Photographer^all]</b><br/>[ex.Notes^all]</
td>
```

8. Make fielded searching the default by changing the **SearchTypes** format statement to **form, plain** (instead of **plain,form**).

ISISPlug stores a nicely formatted version of the record as the document text, and this is what is displayed when we view a record. Lets tidy it up a little more.

9. Remove the *DETACH* and *NO HIGHLIGHTING* buttons by setting the **DocumentButtons** format statement to empty.

10. Remove the "Untitled" at the top of the document by setting the **DocumentHeading** format statement to empty.

11. Finally, lets link to the raw record, which is stored as **ISISRawRecord** metadata. Edit the **DocumentText** format statement to look like the following. (This format can be copied from *sample_files → isis → format_tweaks → document_text.txt*.)

```
<p>[Text]</p>
{If}{_cgiargshowrecord_,
<p><b>CDS Record:</b><br/><tt>[ISISRawRecord]</tt></p>
```

```
<center><a href=\'_gwcgi_?e=_cgiarge_&a=d&d=_cgiargd_\'>Hide CDS
Record</a></center>,
<center><a href=\'_gwcgi_?e=_cgiarge_&a=d&d=_cgiargd_&showrecord=1
\'>Show CDS Record</a></center>
}
```

12. Preview the collection.

Lab 4: Greenstone formatting exercises

4.1. Formatting the HTML collection—Tudor

1. Open up your **tudor** collection, go to the **Format** panel (by clicking on its tab) and select **Format Features** from the left-hand list. Leave the editing controls at their default value, so that **Choose Feature** displays *All Features* and **VList** is selected as the **Affected Component**. The text in the **HTML Format String** box reads as follows:

   ```
   <td valign=top>[link][icon][/link]</td>
   <td valign=top>[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]} [ex./
   srclink]</td>
   <td valign=top>[highlight]
   {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
   [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   This displays something that looks like this:

   
   A discussion of question five from Tudor Quiz: Henry VIII
   (quizstuff.html)

   for a particular document whose *Title*metadata is **A discussion of question five from Tudor Quiz: Henry VIII**and whose *Source*metadata is **quizstuff.html**.

   This format appears in the search results list, in the **Titles**list, and also when you get down to individual documents in the **Subjects**hierarchy. This is Greenstone's default format statement.

Greenstone's default format statement is complex because it is designed to produce something reasonable under almost any conditions, and also because for practical reasons it needs to be backwards compatible with legacy collections.

2. Delete the contents of the **HTML Format String** box and replace it with this simpler version:

   ```
   <td>[link][icon][/link]</td>
   <td>[ex.Title]<br>
       <i>([ex.Source])</i>
   </td>
   ```

   Preview the result (you don't need to build the collection, because changes to format statements take effect immediately). Look at some search results and at the **Titles** list. They are just the same as before! Under most circumstances this far simpler format statement is entirely equivalent to Greenstone's more complex default.

   *But there's a problem. Beside the bookshelves in the **Subjects** browser, beneath the subject appears a mysterious "()". What is printed for these bookshelves is governed by the same format statement, and though bookshelf nodes of the hierarchy have associated* Title *metadata—their title is the name of the metadata value associated with that bookshelf—they do not have **ex.Source** metadata, so it comes out blank.*

3. In the **Format Features** section of the **Format** panel, the **Choose Feature** menu (just above **Affected Component** menu) displays *All Features*. That implies that the same format is used for the search results, titles, and all nodes in the subject hierarchy—including internal nodes (that is, bookshelves). The **Choose Feature** menu can be used to restrict a format statement to a specific one of these lists. We

will override this format statement for the hierarchical *subject* classifier. In the **Choose Feature** menu, scroll down to the item that says

CL2: Hierarchy -metadata dc.Subject and Keywords

and select it. This is the format statement that affects the second classifier (i.e., "CL2"), which is a **Hierarchy**classifier based on **dc.Subject and Keywords**metadata.

Click <**Add Format**>to add this format statement to the collection.

Edit the **HTML Format String**box below to read

```
<td>[link][icon][/link]</td>
<td>[ex.Title]</td>
```

4. **Preview** the **Subjects** list in the collection. First, the offending "()" has disappeared from the bookshelves. Second, when you get down to a list of documents in the subject hierarchy, the filename does not appear beside the title, because **ex.Source** is not specified in the format statement and this format statement applies to all nodes in the *subject* classifier. Note that the search results and titles lists have not changed: they still display the filename underneath the title.

5. Let's change the search results format so that **dc.Subject and Keywords** metadata is displayed here instead of the filename. In the **Choose Feature** menu (under **Format Features** on the **Format** panel), scroll down to the item **Search** and select it. Click <**Add Format**> to add this format statement to the collection. Change the **HTML Format String** box below to read

```
<td>[link][icon][/link]</td>
<td>[ex.Title]<br>
    [dc.Subject]
</td>
```

6. To insert the **[dc.Subject]**, position the cursor at the appropriate point and either type it in, or select it from the **Insert Variable...** drop down menu. This menu shows many of the things that you can put in square brackets in the format statement.

7. **Preview** the collection. Documents in the search results list will be displayed like this:


A discussion of question five from Tudor Quiz: Henry VIII
Tudor period|Others

(The vertical bar appears because this **dc.Subject and Keywords** metadata is hierarchical metadata. Unfortunately there is no way to get at individual components of the hierarchy. For most metadata, such as title and author, this isn't a problem.)

8. Finally, let's return to the *Subjects* hierarchy and learn how to do different things to the bookshelves and to the documents themselves. In the **Choose Feature** menu, re-select the item

CL2: Hierarchy -metadata dc.Subject and Keywords

Edit the **HTML Format String**box below to read

```
<td>[link][icon][/link]</td>
<td>{If}{[numleafdocs],<b>Bookshelf title:</b> [ex.Title],
         <b>Title:</b> [ex.Title]}
</td>
```

Again, you can insert the items in square brackets by selecting them from the **Insert Variable...** drop down box.

*The **If** statement tests the value of the variable **numleafdocs**. This variable is only set for internal nodes of the hierarchy, i.e. bookshelves, and gives the number of documents below that node. If it is set we take the first branch, otherwise we take the second. Commas are used to separate the branches. The curly brackets serve to indicate that the **If** is special—otherwise the word "If" itself would be output.*

9. **Preview** the collection and examine the subject hierarchy again to see the effect of your changes. Bookshelves should say **Bookshelf title:** and then the title, while documents will display **Title:** and the title. Note that the number of documents in the bookshelf is not displayed: we are using `[numleafdocs]` to test what kind of item in the list we are at, but we are not displaying it.

4.2. Formatting the Word and PDF collection

In this exercise, we play around with the format statements in the Word and PDF collection.

1. Open the **reports** collection in the Librarian Interface and go to the **Format Features** section of the **Format** panel.

Tidying up the default format statement

2. In this part of the exercise, we make the format statement simpler without changing the resulting display.

   Greenstone's default format statement is complex because it is designed to produce something reasonable under almost any conditions, and also because for practical reasons it needs to be backwards compatible with legacy collections. For this collection, we don't need all of the complexity.

   Make sure that the **VList** format statement is selected in the list of formats.

   The default **VList** format statement looks like the following:

   ```
   <td valign="top">[link][icon][/link]</td>
   <td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./
   srclink]</td>
   <td valign="top">[highlight]
   {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
   [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   This format statement is the default used for any vertical list, such as search results, classifiers, and document table of contents.

   `{Or}{[ex.thumbicon],[ex.srcicon]}` chooses *ex.thumbicon* metadata if its there, otherwise chooses *ex.srcicon* metadata. If neither are present, nothing is displayed. For this collection there is no *ex.thumbicon* metadata so the choice is not needed.

   Replace `{Or}{[ex.thumbicon],[ex.srcicon]}` (highlighted above) with `[ex.srcicon]`.

   There is no *exp.Title* metadata, so remove that element from `{Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}`.

   The resulting format statement looks like the following:

   ```
   <td valign=top>[link][icon][/link]</td>
   <td valign=top>[ex.srclink][ex.srcicon][ex./srclink]</td>
   <td valign=top>[highlight]
   {Or}{[dc.Title],[ex.Title],Untitled} [/highlight] {If}{[ex.Source],
   <br><i>([ex.Source])</i>}</td>
   ```

   Preview the collection to make sure the display hasn't changed. You shouldn't notice any difference when looking at search results, classifiers etc.

Linking to Greenstone version or original version of documents

3. For collections with documents that undergo a conversion process during importing (e.g. Word, PDF, PowerPoint documents, but not text, HTML documents), the original file is stored in the collection along with the converted version. The default **VList** format statement links to both versions:

   `[ex.link][icon][/ex.link]` links to the Greenstone HTML version, while `[ex.srclink][ex.srcicon][/ex.srclink]` links to the original.

   Choose **SearchVList** in **Format Features** by selecting **Search** from the **Choose Feature** drop down list, and **VList** from the **Affected Component** list. Click **<Add Format>** to add the **SearchVList** format statement into the list of assigned formats. Experiment with removing either of the two links from the format statement.

   To see the results of your changes, preview the collection and do a search. You are making changes to **SearchVList**, which means the changes will only apply to search results.

   Storing and displaying the original allows users to see the correct format, but requires the user to have the relevant program installed. It also increases the size of the collection. The Greenstone version can be viewed in a browser, but may not look as nice.

Making bookshelves show how many items they contain

4. Next, we'll customize the format for the *Creators* list. Classifier bookshelves have only a few pieces of metadata to display: `[ex.Title]` and `[numleafdocs]`. Whatever metadata the classifier has been built on, the bookshelf label is always stored as `[ex.Title]`. This is why a Creator is printed out for each bookshelf even though `[dc.Creator]` is not specified in the format statement. `[numleafdocs]` is only defined for bookshelves, so this metadata can be used in an `{If}` statement to make bookshelves and documents display differently in the list.

   Make each bookshelf in the Creator classifier show how many entries it contains. In the **Format Features** section of the **Format** panel, select the **CL2 AZCompactList** classifier which is based on **dc.Creator** metadata from the **Choose Feature** drop down list, and **VList** from the **Affected Component** list. Click the **<Add Format>** button to add this format into the list of assigned formats. Note that it gets added as **CL2VList** in this list: it is the **VList** format for the second (**CL2**) classifier.

   Append the following text to the bottom of the format statement:

   ```
   {If}{[numleafdocs],<td><i>([numleafdocs])</i></td>}
   ```

   Preview the collection. Click on the *Creators* list and notice that the bookshelves now display how many documents they contain.

   This revised format statement has the effect of specifying in brackets how many items are contained within a bookshelf. Since only bookshelves define `[numleafdocs]`, only they will display this. By modifying **CL2VList** instead of **VList**, the change will only apply to the second classifier (Creators).

Displaying multi-valued metadata

5. Next we modify the document entries in the Creator classifier to display all authors. Back in **Format Features**, select the **CL2VList** format in the list of assigned formats. After `{If}{[ex.Source],<br>` in the format statement, add `[sibling:dc.Creator]`.

   `[ex.Source]` is not defined for bookshelves, so can also be used to differentiate bookshelves and documents.

The resulting format statement looks like:

```
<td valign=top>[link][icon][/link]</td>
<td valign=top>[ex.srclink][ex.srcicon][ex./srclink]</td>
<td valign=top>[highlight]
{Or}{[dc.Title],[ex.Title],Untitled}[/highlight]
{If}{[ex.Source],<br>[sibling:dc.Creator]
<i>([ex.Source])</i>}</td>
{If}{[numleafdocs],<td><i>([numleafdocs])</i></td>}
```

This will display the Greenstone link, the link to the original, then the Title. For bookshelves, it will also display how many documents the bookshelf contains. For documents, it will display all the Authors (Creators), and the source document. `[sibling:dc.Creator]` displays all the Creator metadata for the document, separated by a space (" "), while `[dc.Creator]` displays only the first author. Preview the *Creators* list and make sure that all authors are displayed for documents.

6. You can change the separator between the authors. Modify the format statement, and replace `[sibling:dc.Creator]` with `[sibling(All'<br/>'):dc.Creator]`. This will add a new line after each author (`<br/>` specifies a line break in HTML). Preview the *Creators* list.

   If you have done exercise **Enhanced Word document handling**, the collection will have both dc. Creator and ex.Creator metadata. To display both, you can use

   ```
   [sibling:dc.Creator] [sibling:ex.Creator]
   ```

   To display dc.Creator if it is present, otherwise display ex.Creator, use

   ```
   {Or}{[sibling:dc.Creator],[sibling:ex.Creator]}
   ```

Advanced multi-valued metadata

7. You may notice that the **AZCompactList** classifier has two options after the **metadata** option: **firstvalueonly** and **allvalues**. Manually added metadata can be used to replace or enhance automatically extracted metadata, and these options control exactly which pieces of metadata a document is classified by.

   For example, say we have two documents. Document 1 has four Creators specified (dc.Creator = dcA, dc.Creator = dcB, ex.Creator = exA, ex.Creator = exB), while document 2 has three (ex.Creator = exA, ex.Creator = exB, ex.Creator = exC). The following table shows which metadata values each document is classified by, for the different classifier options:

AZCompactList options	Document 1	Document 2
-metadata dc.Creator,ex.Creator	dcA, dcB	exA, exB, exC
-metadata dc.Creator,ex.Creator -firstvalueonly	dcA	exA
-metadata dc.Creator,ex.Creator -allvalues	dcA, dcB, exA, exB	exA, exB, exC

8. Now we set the **firstvalueonly** option for the *Creators* classifier. Switch to the **Browsing Classifiers** section of the **Design** panel, select the **AZCompactList** for **dc.Creator** metadata in the **Assigned Classifiers** box and click <**Configure Classifier...**>. Select the **firstvalueonly** option.

   Rebuild and **preview** the collection. Now the *Creators* list classifies documents based on the first author appearing in the **dc.Creator** metadata.

   If you set the **metadata** field of **AZCompactList** to `dc.Creator,ex.Creator` in the **A collection of**

Word and PDF files exercise, now the *Creators* list will classify based on the first author appearing in either the **dc.Creator** metadata or the **ex.Creator** metadata.

Lab 5: Advanced collection configuration

5.1. Pointing to documents on the web

1. Open up your **tudor** collection, and in the **Gather** panel inspect the files you dragged into it. The first folder is *englishhistory.net*, which opens up to reveal *tudor*, and so on. The files represent a complete sweep of the pages (and supporting images) that constitute the *Tudor citizens* section of the *englishhistory.net* web site. They were downloaded from the web in a way that preserved the structure of the original site. This allows any page's original URL to be reconstructed from the folder hierarchy.

2. In the **Design** panel, select the **Document Plugins** section, then select the **plugin HTMLPlugin** line and click <**Configure Plugin...**>. A popup window appears. Locate the **file_is_url** option (about halfway down the first block of items) and switch it on. While you are there, switch off the **smart_block** option so that stray images are not processed. Click <**OK**>.

   Setting this option to the **HTMLPlugin** means that Greenstone sets an additional piece of metadata for each document called **URL**, which gives its original URL.

   It is important that the files gathered in the collection start with the web domain name (*englishhistory.net* in this case). The conversion process will not work if you dragged over a subfolder, for example the *tudor* folder, because this will set **URL** metadata to something like

   http://tudor/citizens/...

   rather than

   http://englishhistory.net/tudor/citizens/...

   If you have copied over a subfolder previously, delete it and make a fresh copy. Drag the folder in the right-hand side of the **Gather**panel on to the trash can in the lower right corner. Then obtain a fresh copy of the files by dragging across the *englishhistory.net*folder from the *sample_files* → *tudor*folder (or the **Downloaded Files**folder if you have done exercise **Downloading files from the web**) on the left-hand side.

3. To make use of the new URL metadata, the icon link must be changed to serve up the original URL rather than the copy stored in the digital library. Go to the **Format** panel, select the **Format Features** section and edit the **VList** format statement by replacing

   ```
   [link][icon][/link]
   ```

   with

   ```
   [weblink][webicon][/weblink]
   ```

4. Switch to the **Create** panel and **build** and **preview** the collection. Note that the document icons have changed. The collection behaves exactly as before, except that when you click a document icon your web browser retrieves the original document from the web . If you are working offline you will be unable to retrieve the document.

5.2. Enhanced Word document handling

The standard way Greenstone processes Word documents is to convert them to HTML format using a third-party program, wvWare. This sometimes doesn't do a very good job of conversion. If you are using Windows, and have Microsoft Word installed, you can take advantage of Windows native scripting to do a better job of conversion. If the original document was hierarchically structured using Word styles, these can be used to structure the resulting HTML. Word document properties can also be extracted as metadata.

1. In your digital library, preview the **reports** collection. Look at the HTML versions of the Word documents and notice how they have no structure-they have been converted to flat documents.

Using Windows native scripting

2. In the Librarian Interface, open up the **reports** collection. Switch to the **Design** panel and select the **Document Plugins** section on the left-hand side. Double click the **WordPlugin** plugin and switch on the **windows_scripting** option.

   In the **Search Indexes** section, check the **section** checkbox to build the indexes on section level as well as document level.

3. **Build** the collection. You will notice that the Microsoft Word program is started up for each Word document—the document is saved as HTML from Word itself, to get a better conversion. **Preview** the collection. In the **Titles** list, notice that *word03.doc* and *word06.doc* now have a book icon, rather than a page icon. These now appear with hierarchical structure.

   The default behaviour for **WordPlugin** with **windows_scripting** is to section the document based on "Heading 1", "Heading 2", "Heading 3" styles. If you open up the *word03.doc* or *word06.doc* documents in Word, you will see that the sections use these Heading styles.

   Note, to view style information in Word, you can select **Format → Styles and Formatting** from the menu, and a side bar will appear on the right hand side. Click on a section heading and the formatting information will be displayed in this side bar.

4. Some of the documents do not use styles (e.g. *word01.doc*) and no structure can be extracted from them. Some documents use user-defined styles. **WordPlugin** can be configured to use these styles instead of Heading 1, Heading 2 etc. Next we will configure **WordPlugin** to use the styles found in *word05.doc*.

Modes in the Librarian Interface

5. The Librarian Interface can operate in four modes. Go to **File → Preferences... → Mode** and see the four modes and what functionality they provide access to. **Librarian** is the default mode.

6. Change the mode to **Library Systems Specialist** because you will need to use regular expressions to set up the style options in the next part of the exercise.

Defining styles

7. Open up *word05.doc* in Word (by double-clicking on it in the **Gather** pane), and examine the

title and section heading styles. You will see that various user-defined header styles are set such as:

- *ManualTitle*: Title of the manual
- *ChapterTitle*: Level 1 section heading
- *SectionHeading*: Level 2 section heading
- *SubsectionHeading*: Level 3 section heading
- *AppendixTitle*: Appendix section title

8. In the **Document Plugins** section of the **Design** panel, select **WordPlugin** and click **<Configure Plugin...>**. Four types of header can be set which are:

   - `level1_header (level1Header1|level1Header2|...)`
   - `level2_header (level2Header1|level2Header2|...)`
   - `level3_header (level3Header1|level3Header2|...)`
   - `title_header (titleHeader1|titleHeader2|...)`

   These header options define which styles should be considered as title, level 1, level 2 and level 3 styles.

   Ensure that the **windows_scripting** option is checked, and set the options as follows (spaces in the Word styles are removed when converting to HTML styles, and these options must match the HTML styles):

   ```
   level1_header:(ChapterTitle|AppendixTitle)
   level2_header: SectionHeading
   level3_header: SubsectionHeading
   title_header: ManualTitle
   ```

   *If you can't see these options in the **WordPlugin** configuration pane, check that you are in **Library Systems Specialist** mode as described above.*

   Once these are set, click **<OK>**.

9. Close any documents that are still open in Word, as this can prevent the build process from completing correctly.

10. **Build** the collection and **preview** it. Look in particular at *word05.doc*. You will see that this document is now also hierarchically structured.

    If you have documents with different formatting styles, you can use `(...|...)` to specify all of the different styles.

Removing pre-defined table of contents

11. If you look at the HTML versions of *word05.doc* and *word06.doc*, you will see that it now has two tables of contents. One is generated by Greenstone based on the document's styles, the other was already defined in the Word document. **WordPlugin** can be configured to remove predefined tables of contents and tables of figures. The tables must be defined with Word styles in order for this to work.

12. To remove the tables of contents and figures from *word06.doc* and the table of contents from

word05.doc, switch on the **delete_toc** option in **WordPlugin**. Set the **toc_header** option to `(MsoToc1|MsoToc2|MsoToc3|MsoTof|TOA)`. In this document, the table of contents and list of figures use these four style names. Click <**OK**>.

13. **Build** and **preview** the collection. Both *word05.doc* and *word06.doc* should now have only one table of contents.

14. Switch the Librarian Interface back to **Librarian** mode (**File** → **Preferences...** → **Mode**).

Extracting document properties as metadata

15. When the **windows_scripting** option is set, word document properties can be extracted as metadata. By default, only the Title will be extracted. Other properties can be extracted using the **metadata_fields** option.

16. In the **Enrich** panel, look at the metadata that has been extracted for *word05.doc* and *word06.doc*. Now open the documents in Word and look at what properties have been set (**File** → **Properties**). They have Title, Author, Subject, and Keywords properties. **WordPlugin** can be configured to look for these properties and extract them.

17. In the **Design** panel, under **Document Plugins**, configure **WordPlugin** once again. Switch on the configuration option **metadata_fields**. Set the value to

    ```
    Title,Author<Creator>,Subject,Keywords<Subject>
    ```

    This will make **WordPlugin** try to extract Title, Author, Subject and Keywords metadata. Title and Subject will be saved with the same name, while Author will be saved as Creator metadata, and Keywords as Subject metadata.

18. Make sure you have closed all the documents that were opened, then **rebuild** the collection.

19. Look at the metadata for the two documents again in the **Enrich** panel. You should now see ex.Creator and ex.Subject metadata items. This metadata can now be used in display or browsing classifiers etc.

5.3. Enhanced PDF handling

Greenstone converts PDF files to HTML using third-party software: *pdftohtml.pl*. This lets users view these documents even if they don't have the PDF software installed. Unfortunately, sometimes the formatting of the resulting HTML files is not so good.

This exercise explores some extra options to the PDF plugin which may produce a nicer version for display. Some of these options use the standard pdftohtml program, others use ImageMagick and Ghostscript to convert the file to a series of images. Ghostscript is a program that can convert Postscript and PDF files to other formats. You can download it from http://www.cs.wisc.edu/~ghost/ (follow the link to the current stable release).

1. In the Librarian Interface, start a new collection called "PDF collection" and base it on **-- New Collection --**.

   In the **Gather** panel, drag just the PDF documents from *sample_files → Word_and_PDF → Documents* into the new collection. Also drag in the PDF documents from *sample_files → Word_and_PDF → difficult_pdf*.

   Go to the **Create** panel and build the collection. Examine the output from the build process. You will notice that one of the documents could not be processed. The following messages are shown: "The file pdf05-notext.pdf was recognised but could not be processed by any plugin.", and "3 were processed and included in the collection. 1 was rejected".

2. Preview the collection and view the documents. *pdf05-notext.pdf* does not appear as it could not be processed. *pdf06-weirdchars.pdf* was processed but looks very strange. The other PDF documents appear as one long document, with no sections.

Modes in the Librarian Interface

*The Librarian Interface can operate in different modes. The default mode is **Librarian** mode. We can use **Expert** mode to work out why the pdf file could not be processed.*

3. Use the **Preferences...** item on the **File** menu to switch to **Expert** mode and then build the collection again. The **Create** panel looks different in **Expert** mode because it gives more options: locate the **<Build Collection>** button, near the bottom of the window, and click it. Now a message appears saying that the file could not be processed, and why. Amongst all the output, we get the following message: "Error: PDF contains no extractable text. Could not convert pdf05-notext.pdf to HTML format". pdftohtml.pl cannot convert a PDF file to HTML if the PDF file has no extractable text.

4. We recommend that you switch back to **Librarian** mode for subsequent exercises, to avoid confusion.

Splitting PDFs into sections

5. In the **Document Plugins** section of the **Design** panel, configure **PDFPlugin**. Switch on the **use_sections** option.

   In the **Search Indexes** section, check the **section** checkbox to build the indexes on section level

as well as document level.

Build and **preview** the collection. View the text versions of some of the PDF documents. Note that these are now split into a series of pages, and a "go to page" box is provided. The format is still a bit ugly though, and pdf05-notext.pdf is still not processed.

Using image format

6. If conversion to HTML doesn't produce the result you like, PDF documents can be converted to a series of images, one per page. This requires ImageMagick and Ghostscript to be installed.

7. In the **Document Plugins** section, configure **PDFPlugin**. Set the **convert_to** option to one of the image types, e.g. **pagedimg_jpg**. Switch off the **use_sections** option, as it is not used with image conversion.

8. **Build** the collection and **preview**. All PDF documents (including pdf05-notext.pdf) have been processed and divided into sections, but each section displays "This document has no text.". For the conversion to images for PDF documents, no text is extracted.

9. In order to view the documents properly, you will need to modify the format statement. In the **Format Features** section on the **Format** panel, select the **DocumentText** format statement. Replace

   ```
   [Text]
   ```

   with

   ```
   [screenicon]
   ```

10. Preview the collection. Images from the document are now displayed instead of the extracted text. Both *pdf05-notext.pdf* and *pdf06-weirdchars.pdf* display nicely now.

    In this collection, we only have PDF documents and they have all been converted to images. If we had other document types in the collection, we should use a different format statement, such as:

    ```
    {If}{[parent:FileFormat] eq PDF,[screenicon],[Text]}
    ```

    ***FileFormat** is an extracted metadata item which shows the format of the source document. We can use this to test whether the documents are PDF or not: for PDF documents, display [screenicon], for other documents, display [Text].*

Using process_exp to control document processing (advanced)

11. Processing all of the PDF documents using an image type may not give the best result for your collection. The images will look nice, but as no text is extracted, searching the full text will not be available for these documents. The best solution would be to process most of the PDF files as HTML, and only use the image format where HTML doesn't work.

12. We achieve this by putting the problem files into a separate folder, and adding another **PDFPlugin** plugin with different options.

13. Go to the **Gather** panel. Make a new folder called "notext": right click in the collection panel and select **New folder** from the menu. Change the **Folder Name** to "notext", and click **<OK>**.

    Move the two pdf files that have problems with html (*pdf05-notext.pdf* and *pdf06-weirdchars*.pdf) into this folder by drag and drop. We will set up the plugins so that PDF files in this *notext* folder are processed differently to the other PDF files.

14. Change to **Library Systems Specialist** mode so that you can add two of the same plugin, and use regular expressions in the plugin options (**File → Preferences... → Mode**).

    For version 2.71, you'll need to close GLI now then restart it to get the list of plugins to update properly.

15. Switch to the **Document Plugins** section of the **Design** panel. Add a second PDF plugin by selecting **PDFPlugin** from the **Select plugin to add:** drop-down list, and clicking **<Add Plugin...>**. This plugin will come after the first PDF plugin, so we configure it to process PDF documents as HTML. Set the **convert_to** option to **html**, and switch on the **use_sections** option. Click **<OK>**.

16. Configure the first PDF plugin, and set the **process_exp** option to '**notext.*\.pdf**'.

17. The two PDF plugins should have options like the following:

    ```
    plugin PDFPlugin -convert_to pagedimg_jpg -process_exp "notext.*\.
    pdf"
    plugin PDFPlugin -convert_to html -use_sections
    ```

    The *paged_img* version must come earlier in the list than the *html* version. The **process_exp** for the first **PDFPlugin** will process any PDF files in the *notext* directory. The second **PDFPlugin** will process any PDF files that are not processed by the first one.

    Note that all plugins have the **process_exp** option, and this can be used to customize which documents are processed by which plugin. This option is only visible in **Library Systems Specialist** and **Expert** modes.

    Change back to **Librarian** mode.

18. Edit the **DocumentText** format statement. PDF files processed as HTML will not have images to display, so we need to make sure they get text displayed instead. Change `[screenicon]` to `{If}{[NoText] eq "1",[screenicon],[Text]}`.

19. Build and preview the collection. All PDF documents should look relatively nice. Try searching this collection. You will be able to search for the PDFs that were converted to HTML (try e.g. "bibliography"), but not the ones that were converted to images (try searching for "FAO" or "METS").

Opening PDF files with query terms highlighted

20. Next we'll customize the **SearchVList** format statement to highlight the query terms in a PDF file when it is opened from the search result list. This requires Acrobat Reader 7.0 version or

higher, and currently only works on a Microsoft Windows platform.

21. The search terms are kept in the macro variable **_cgiargq_**, and we append **#search="_cgiargq_"** to the end of a PDF file link to pass the query terms to the PDF file.

    PDFPlugin renames each PDF file as **doc.pdf** and saves it in a unique directory for that document, so we use

    ```
    _httpcollection_/index/assoc/[archivedir]/doc.pdf
    ```

    to refer to the PDF source file. (However, if you used the **-keep_original_filename** option to **PDFPlugin** when building the collection, the original name of the PDF file is kept, and we use

    ```
    _httpcollection_/index/assoc/[archivedir]/[Source]
    ```

    instead to locate the PDF source file.)

22. Select **SearchVList** from the list of assigned formats. We need to test whether the file is a PDF file before linking to doc.pdf, using `{If}{[ex.FileFormat] eq 'PDF',,}`. For PDF files, we use the above format instead of the `[ex.srclink]` and `[ex./srclink]` variables to link to the file.

    The resulting format statement is:

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">{If}{[ex.FileFormat] eq 'PDF', <a href=
    \"_httpcollection_/index/assoc/[archivedir]/doc.pdf#search=&quot;
    _cgiargq_&quot;\">[ex.srcicon]</a>,
    [ex.srclink][ex.srcicon][ex./srclink]}</td>
    <td valign="top">[highlight]
    {Or}{[dc.Title],[ex.Title],Untitled}
    [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
    ```

    When the PDF icons are clicked in the search results, Acrobat will open the file with the search window open, and the query terms highlighted.

5.4. Section tagging for HTML documents

1. In a browser, take a look at the Greenstone demo collection. Browse to one of the documents. This collection is based on HTML files, but they appear structured in the collection. This is because these HTML files were tagged by hand into sections.

2. Using a text editor (e.g. WordPad) open up one of the HTML files from the demo collection: *Greenstone → collect → demo → import → fb33fe →fb33fe.htm*. You will see some HTML comments which contain section information for Greenstone. They look like:

```
<!--
<Section>
  <Description>
    <Metadata name="Title">Farming snails 1: Learning about snails;
    Building a pen; Food and shelter plants</Metadata>
  </Description>
-->

<!--
</Section>
<Section>
  <Description>
    <Metadata name="Title">Dew and rain</Metadata>
  </Description>
-->
```

   When Greenstone encounters a `<Section>` tag in one of these comments, it will start a new subsection of the document. This will be closed when a `</Section>` tag is encountered. Metadata can also be added for each section—in this case, **Title** metadata has been added for each section. In the browser, find the **Farming snails 1** document in the demo collection (through the *Titles* browser). Look at its table of contents and compare it to the `<Section>` tags in the HTML document.

3. Add a new Section into this document. For example, lets add a new subsection into the **Introduction** chapter. In the text editor, add the following just after the Section tag for the **Introduction** section:

```
<!--
<Section>
  <Description>
    <Metadata name="Title">Snails are good to eat.</Metadata>
  </Description>
-->
```

   Then just before the next section tag (**What do you need to start?**), add the following:

```
<!--
</Section>
-->
```

   The effect of these changes is to make a new subsection inside the **Introduction** chapter.

4. Open the Greenstone demo collection in the Librarian Interface. In the **Document Plugins** section of the **Design** panel, note that **HTMLPlugin** has the **description_tags** option set. This

option is needed when `<Section>` tags are used in the source documents.

5. **Build** and **preview** the collection. Look at the **Farming snails 1** document again and check that your new section has been added.

Lab 6: Multimedia

6.1. Looking at a multimedia collection

1. Copy the entire folder

   sample_files → beatles → advbeat_large

   (with all its contents) into your Greenstone *collect* folder. If you have installed Greenstone in the usual place, for Greenstone version 2.81 and above, this is

   My Computer → Local Disk (C:) → Users → <Username> → collect

   where <Username> is the username under with Greenstone is installed. For versions below 2.81, this is

   My Computer → Local Disk (C:) → Program Files → Greenstone → collect

   Put *advbeat_large* in there.

2. On Windows, if the Greenstone Digital Library Local Library Server is already running, re-start it by clicking the CD icon on the task bar and then pressing *Restart Library*. If not, start it up by selecting *Greenstone Digital Library* from the *Start* menu. On Linux and Mac, just do a forced reload/refresh of the web browser.

3. Explore the Beatles collection. Note how the *Browse* button divides the material into seven different types. Within each category, the documents have appropriate icons. Some documents have an audio icon: when you click these you hear the music (assuming your computer is set up with appropriate player software). Others have an image thumbnail: when you click these you see the images.

4. Look at the *Titles* browser. Each title has a bookshelf that may include several related items. For example, *Hey Jude* has a MIDI file, lyrics, and a discography item.

5. Observe the low quality of the metadata. For example, the four items under **A Hard Day's Night** (under "H" in the *Titles* browser) have different variants as their titles. The collection would have been easier to organize had the metadata been cleaned up manually first, but that would be a big job. Only a tiny amount of metadata was added by hand—fewer than ten items. The original metadata was left untouched and Greenstone facilities used to clean it up automatically. (You will find in **Building a multimedia collection** that this is possible but tricky.)

6. In the file browser, take a look at the files that makes up the collection, in the

   sample_files → beatles → advbeat_large → import

   folder. What a mess! There are over 450 files under seven top-level sub-folders. Organization is minimal, reflecting the different times and ways the files were gathered. For example, *html_lyrics* and *discography* are excerpts of web sites, and *images* contains various images in

JPEG format. For each type, drill down through the hierarchy and look at a sample document.

6.2. Building a multimedia collection

We will proceed to reconstruct from scratch the Beatles collection that you have just looked at. We develop the collection using a small subset of the material, purely to speed up the repeated rebuilding that is involved.

1. Start a new collection (**File → New...**) called **small beatles**, basing it on the default **-- New Collection --**. (Basing it on the existing Advanced Beatles collection would make your life far easier, but we want you to learn how to build it from scratch!)

2. Copy the files provided in

   sample_files → beatles → advbeat_small

   into your new collection. Do this by opening up *advbeat_small*, selecting the eight items within it (from *discography* to *beatles_midi.zip*), and dragging them across. Because some of these files are in MP3 and MARC formats you will be asked whether to include **MP3Plugin** and **MARCPlugin** in your collection. Click **<Add Plugin>**.

3. Change to the **Enrich** panel and browse around the files. There is no metadata—yet. Recall that you can double-click files to view them.

   (There are no MIDI files in the collection: these require more advanced customisation because there is no MIDI plugin. We will deal with them later.)

4. Change to the **Create** panel and **build** the collection.

5. **Preview** the result.

Manually correcting metadata

6. You might want to correct some of the metadata—for example, the atrocious misspelling in the titles "MAGICAL MISTERY TOUR." These documents are in the discography section, with filenames that contain the same misspelling. Locate one of them in the **Enrich** panel. Notice that the extracted metadata element **ex.Title** is now filled in, and misspelt. You cannot correct this element, for it is extracted from the file and will be re-extracted every time the collection is re-built.

7. Instead, add **dc.Title** metadata for these two files: "Magical Mystery Tour." Change to the **Enrich** panel, open the discography folder and drill down to the individual files. Set the **dc.Title** value for the two offending items.

*Now there's a twist. The **dc.Title** metadata won't appear in Titles because the classifier has been instructed to use **ex.Title**. But changing the classifier to use **dc.Title** would miss out all the extracted titles! Fortunately, there's a way of dealing with this by specifying a list of metadata names in the classifier.*

8. Change to the **Design** panel and select the **Browsing Classifiers** section. Double-click the **ex. Title** classifier (the first one) to edit its configuration settings.

- Type `dc.Title,` before the `ex.Title` in the metadata box—i.e. make it read

  ```
  dc.Title,ex.Title
  ```

  and click <**OK**>.

Build the collection again, and **preview** it.

Extracted metadata is unreliable. But it is very cheap! On the other hand, manually assigned metadata is reliable, but expensive. The previous section of this exercise has shown how to aim for the best of both worlds by using extracted metadata but correcting it when it is wrong.

Browsing by media type

9. First let's remove the **AZList** classifier for filenames, which isn't very useful, and replace it with a browsing structure that groups documents by category (discography, lyrics, audio etc.). Categories are defined by manually assigned metadata.

   - Change to the **Enrich** panel, select the folder *discography* and set its **dc.Format** metadata value to "Discography". Setting this value at the folder level means that all files within the folder inherit it.
   - Repeat the process. Assign "Lyrics" to the *html_lyrics* folder, "Images" to *images*, "MARC" to *marc*, "Audio" to *mp3*, "Tablature" to *tablature_txt*, and "Supplementary" to *wordpdf*.
   - Switch to the **Design** panel and select the **Browsing Classifiers** section.
   - Delete the **ex.Source** classifier (the second one).
   - Add an **AZCompactList** classifier. Select **dc.Format** as the **metadata** field and specify "browse" as the **buttonname**. Click the **sort** checkbox, and select **ex.Title** in the drop-down list: this will make the classifier display documents in alphabetical order of title.

   Build the collection again and **preview** it.

*Note how we assigned **dc.Format** metadata to all documents in the collection with a minimum of labour. We did this by capitalizing on the folder structure of the original information. Even though we complained earlier about how messy this folder structure is, you can still take advantage of it when assigning metadata.*

Suppressing dummy text

10. Alongside the Audio files there is an MP3 icon, which plays the audio when you click it, and also a text document that contains some dummy text. Image files also have dummy documents. These dummy documents aren't supposed to be seen, but to suppress them you have to fiddle with a format statement.

    - Change to the **Format** panel and select the **Format Features** section.
    - Ensure that **VList** is selected, and make the changes that are highlighted below. You need to insert five lines into the first line, and delete the second line. (Note, the changes are available in a text file, see below.) Change:

      ```
      <td valign=top>[link][icon][/link]</td>
      <td valign=top>[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}
      ```

```
[ex./srclink]</td>
<td valign=top>[highlight]
{Or}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
```

to this:

```
<td valign=top>
{If}{[dc.Format] eq 'Audio',
[srclink][srcicon][/srclink],
{If}{[dc.Format] eq 'Images',
[srclink][thumbicon][/srclink],
{If}{[dc.Format] eq 'Supplementary',
[srclink][srcicon][/srclink] [link][icon][/link], [link]
[icon][/link]}}}</td>
<td valign=top>[highlight]
{Or}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
```

To make this easier for you we have prepared a plain text file that contains the new text. In WordPad open the following file:

sample_files → beatles → format_tweaks → audio_tweak.txt

(Be sure to use WordPad rather than Notepad, because Notepad does not display the line breaks correctly.) Place it in the copy buffer by highlighting the text in WordPad and selecting **Edit → Copy**. Now move back to the Librarian Interface, highlight all the text that makes up the current **VList** format statement, and use **Edit → Paste (ctrl-v)** to transform the old statement to the new one.

Preview the result. You may need to click the browser's <**Reload**> button to force it to re-load the page.

11. While we're at it, let's remove the source filename from where it appears after each document.
    - In the **VList** format feature, delete the text that is highlighted below:

```
<td valign=top>
{If}{[dc.Format] eq 'Audio',
[srclink][srcicon][/srclink],
{If}{[dc.Format] eq 'Images',
[srclink][thumbicon][/srclink],
[link][icon][/link]}}</td>
<td valign=top>[highlight]
{Or}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
```

Preview the result (you don't need to rebuild the collection.)

Using AZCompactList rather than AZList

12. There are sometimes several documents with the same title. For example, *All My Loving* appears both as lyrics and tablature (under *ALL MY LOVING*). The **Titles** browser might be improved by grouping these together under a bookshelf icon. This is a job for an **AZCompactList**.

- Change to the **Design** panel and select the **Browsing Classifiers** section.
- Remove the **ex.Title** classifier (at the top)
- Add an **AZCompactList** classifier, and enter **dc.Title,ex.Title** as its metadata.
- Finish by pressing **<OK>**.
- Move the new classifier to the top of the list (**<Move Up>** button).

Build the collection again and **preview** it. Both items for *All My Loving* now appear under the same bookshelf. However, many entries haven't been amalgamated because of non-uniform titles: for example *A Hard Day's Night* appears as four different variants. We will learn below how to amalgamate these.

Making bookshelves show how many items they contain

13. Make the bookshelves show how many documents they contain by inserting a line in the **VList** format statement in the **Format Features** section of the **Format** panel. The added line is shown highlighted below. The complete format statement can be copied from *sample_files → beatles → format_tweaks → show_num_docs.txt*.

```
<td valign=top>
{If}{[dc.Format] eq 'Audio',
[srclink][srcicon][/srclink],
{If}{[dc.Format] eq 'Images',
[srclink][thumbicon][/srclink],
[link][icon][/link]}}</td>
<td>{If}{[numleafdocs],([numleafdocs])}</td>
<td valign=top>[highlight]
{Or}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]</td>
```

Preview the result (you don't need to build the collection.) Bookshelves in the titles and browse classifiers should show how many documents they contain.

Adding a Phind phrase browser

14. In the **Browsing Classifiers** section on the **Design** panel, add a **Phind** classifier. Leave the settings at their defaults: this generates a phrase browsing classifier that sources its phrases from *Title* and *text*.

Build the collection again and **preview** it. Select the new **Phrases** option from the navigation bar. Enter a single word in the text box, such as **band**. The phrase browser will present you with phrases found in the collection containing the search term. This can provide a useful way of browsing a very large collection. Note that even though it is called a phrase browser, only single terms can be used as the starting point for browsing.

Branding the collection with an image

15. To complete the collection, lets give it a new image for the top left corner of the page. Go to the **General** section of the **Format** panel. Use the browse button of **URL to 'about page' image:** to select the following image:

sample_files → beatles → advbeat_large → images → beatlesmm.png

Preview the collection, and make sure the new image appears.

Using UnknownPlugin

In this section we incorporate the MIDI files. Greenstone has no MIDI plugin (yet). But that doesn't mean you can't use MIDI files!

16. **UnknownPlugin** is a useful generic plugin. It knows nothing about any given format but can be tailored to process particular document types—like MIDI—based on their filename extension, and set basic metadata.

    In the **Document Plugins** section of the **Design** panel:

    - add **UnknownPlugin**;
    - activate its **process_extension** field and set it to "mid" to make it recognize files with extension *.mid*;
    - Set **file_format** to "MIDI" and **mime_type** to "audio/midi".

    In this collection, all MIDI files are contained in the file *beatles_midi.zip*. **ZIPPlugin** (already in the list of default plugins) is used to unpack the files and pass them down the list of plugins until they reach **UnknownPlugin**.

17. **Build** the collection and **preview** it. Unfortunately the MIDI files don't appear as Audio under the *browse* button. That's because they haven't been assigned **dc.Format** metadata.

    - Back in the **Enrich** panel, click on the file *beatles_midi.zip* and assign its **dc.Format** value to "Audio"—do this by clicking on "Audio" in the **Existing values for dc. Format** list. All files extracted from the Zip file inherit its settings.

Cleaning up a title browser using regular expressions

*We now clean up the **Titles** browser.*

*To do this we must put the Librarian Interface into a different mode. The interface supports four levels of user: **Library Assistant**, who can add documents and metadata to collections, and create new ones whose structure mirrors that of existing collections; **Librarian**, who can, in addition, design new collections, but cannot use specialist IT features (e.g. regular expressions); **Library Systems Specialist**, who can use all design features, but cannot perform troubleshooting tasks (e.g. interpreting debugging output from Perl programs); and **Expert**, who can perform all functions.*

*So far you have mostly been operating in **Librarian** mode. We switch to **Library Systems Specialist** mode for the next exercise.*

18. To switch modes, click **File → Preferences... → Mode** and change to **Library Systems Specialist**. Note from the description that appears that you need to be able to formulate regular expressions to use this mode fully. That is what we do below.

19. Next we return to our **Titles** browser and clean it up. The aim is to amalgamate variants of titles by stripping away extraneous text. For example, we would like to treat "ANTHOLOGY 1", "ANTHOLOGY 2" and "ANTHOLOGY 3" the same for grouping purposes. To achieve this:

- Go to the Title **AZCompactList** under **Browsing Classifiers** on the **Design** panel;
- Activate **removesuffix** and set it to:

```
(?i)(\\s+\\d+)|(\\s+[[:punct:]].*)
```

Build the collection and **preview** the result. Observe how many more times similar titles have been amalgamated under the same bookshelf. Test your understanding of regular expressions by trying to rationalize the amalgamations. (Note: *[[:punct:]]* stands for any punctuation character.) The icons beside the Word and PDF documents are not the correct ones, but that will be fixed in the next format statement.

*The previous exercise was done in **Library Systems Specialist** mode because it requires the use of regular expressions, something librarians are not normally trained in.*

*One powerful use of regular expressions in the exercise was to clean up the **Titles** browser. Perhaps the best way of doing this would be to have proper title metadata. The metadata extracted from HTML files is messy and inconsistent, and this was reflected in the original Titles browser. Defining proper title metadata would be simple but rather laborious. Instead, we have opted to use regular expressions in the **AZCompactList** classifier to clean up the title metadata. This is difficult to understand, and a bit fiddly to do, but if you can cope with its idiosyncrasies it provides a quick way to clean up the extracted metadata and avoid having to enter a large amount of metadata.*

Using non-standard macro files

To put finishing touches to our collection, we add some decorative features

20. Close the collection in the Librarian Interface (**File → Close**).

21. Using your Windows file browser outside Greenstone, locate the folder

    sample_files → beatles → advbeat_large

22. Open up another file browser, and locate the small beatles collection in your Greenstone installation:

    Greenstone → collect → smallbea

    smallbea is the folder name generated by Greenstone for this collection. You can determine what the folder name is for a collection by looking at the title bar of the Librarian Interface: the folder name is displayed in brackets after the collection name.

23. Using the file browser, copy the *images* and *macros* folders from the *advbeat_large* folder into the *smallbea* folder. (It's OK to overwrite the existing *images* folder: the image in it is included in the folder being copied.) The *images* folder includes some useful icons, and the *macros* folder defines some macro names that use these images.

    To see the macro definitions, open the collection in the Librarian Interface (**File → Open...**) and view the **Collection Specific Macros** section in the **Format** panel.

Using different icons for different media types

24. Re-edit your **VList** format statement to be the following (in **Format Features** on the **Format** panel). You can copy this text from the file *sample_files → beatles → format_tweaks → multi_icons.txt*.

```
<td valign=top>
  {If}{[numleafdocs],[link][icon][/link]}
  {If}{[dc.Format] eq 'Lyrics',[link]_iconlyrics_[/link]}
  {If}{[dc.Format] eq 'Discography',[link]_icondisc_[/link]}
  {If}{[dc.Format] eq 'Tablature',[link]_icontab_[/link]}
  {If}{[dc.Format] eq 'MARC',[link]_iconmarc_[/link]}
  {If}{[dc.Format] eq 'Images',[srclink][thumbicon][/srclink]}
  {If}{[dc.Format] eq 'Supplementary',[srclink][srcicon][/
srclink]}
  {If}{[dc.Format] eq 'Audio',[srclink]{If}{[FileFormat] eq 'MIDI',
_iconmidi_,_iconmp3_}[/srclink]}
</td>
<td>
{If}{[numleafdocs],([numleafdocs])}
</td>
<td valign=top>
[highlight]
{Or}{[dc.Title],[Title],Untitled}
[/highlight]
</td>
```

25. **Preview** your collection as before. Now different icons are used for discography, lyrics, tablature, and MARC metadata. Even MP3 and MIDI audio file types are distinguished. If you let the mouse hover over one of these images a "tool tip" appears explaining what file type the icon represents in the current interface language (note: *extra.dm* only defines English and French).

Changing the collection's background image

26. Go to the **Collection Specific Macros** section in the **Format** panel.

27. The content is fairly brief, specifying only what needs to be overridden from the default behaviour for this collection. Near the top you should see:

```
_collectionspecificstyle_ {
<style>
body.bgimage \{ background-image: url("_httpcimages_/beat_margin.
gif"); \}
\#page \{ margin-left: 120px; \}
</style>
}
```

    Replace the text **beat_margin.gif** with **tile.jpg**.

    This line relates to the background image used. The new image *tile.jpg* was in the *images* folder that was copied across previously.

28. **Preview** the collection's home page. The page background is now the new graphic.

    Other features can be altered by editing the macros—for example, the headers and footers used on each page, and the highlighting style used for search terms (specify a different colour, use

bold etc.).

Building a full-size version of the collection

29. To finish, let's now build a larger version of the collection. To do this:

    - Close the current collection (**File → Close**).
    - Start a new collection called *large beatles* (**File → New...**).
    - Base this new collection on *small beatles*.
    - Copy the content of *sample_files → beatles → advbeat_large → import* into this newly formed collection. Since there are considerably more files in this set of documents the copy will take longer.
    - **Build** the collection and **preview** the result. (If you want the collection to have an icon, you will have to add it from the **Format** panel.)

Adding an image collage browser

30. Switch to the **Design** panel and select the **Browsing Classifiers** section. Pull down the **Select classifier to add** menu and select **Collage**. Click **<Add Classifier...>**. There is no need to customize the options, so click **<OK>** at the bottom of the resulting popup.

31. Now change to the **Create** panel and **build** and **preview** the collection.

Lab 7: Scanned images

7.1. Scanned image collection

Here we build a small replica of Niupepa, the Maori Newspaper collection, using five newspapers taken from two newspaper series. It allows full text searching and browsing by title and date. When a newspaper is viewed, a preview image and its corresponding plain text are presented side by side, with a "go to page" navigation feature at the top of the page.

*The collection involves a mixture of plugins, classifiers, and format statements. The bulk of the work is done by **PagedImagePlugin**, a plugin designed precisely for the kind of data we have in this example. For each document, an "item" file is prepared that specifies a list of image files that constitute the document, tagged with their page number and (optionally) accompanied by a text file containing the machine-readable version of the image, which is used for full text searching. Three newspapers in our collection (all from the series "Te Whetu o Te Tau") have text representations, and two (from "Te Waka o Te Iwi") have images only. Item files can also specify metadata. In our example the newspaper series is recorded as **ex.Title** and its date of publication as **ex.Date**. Issue **ex.Volume** and **ex.Number** metadata is also recorded, where appropriate. This metadata is extracted as part of the building process.*

1. Start a new collection called **Paged Images** and fill out the fields with appropriate information: it is a collection sourced from an excerpt of Niupepa documents.

2. In the **Gather** panel, open the *sample_files → niupepa → sample_items* folder and drag the two subfolders into your collection on the right-hand side. A popup window asks whether you want to add **PagedImagePlugin** to the collection: click **<Add Plugin>**, because this plugin will be needed to process the item files.

3. Some of the files you have just dragged in are the newspaper images; others are text files that contain the text extracted from these images. We want these to be processed by **PagedImagePlugin**, not **ImagePlugin** or **TextPlugin**. Switch to the **Document Plugins** section of the **Design** panel and *delete **ImagePlugin** and **TextPlugin***.

4. Open up the configuration window for **PagedImagePlugin** by double-clicking on the plugin. Switch on its **create_screenview** configuration option by checking the box. The source images we use were scanned at high resolution and are large files for a browser to download. The **create_screenview** option generates smaller screen-resolution images of each page when the collection is built. Click **<OK>**.

5. Now go to the **Create** panel, **build** the collection and **preview** the result. Search for "waka" and view one of the titles listed (all three appear as *Te Whetu o Te Tau*). Browse by **Titles** and view one of the *Te Waka o Te Iwi* newspapers. Note that only the *Te Whetu o Te Tau* newspapers have text; *Te Waka o Te Iwi* papers don't.

This collection was built with Greenstone's default settings. You can locate items of interest, but the information is less clearly and attractively presented than in the full Niupepa collection.

Grouping documents by series title and displaying dates within each group

*Under **Titles** documents from the same series are repeated without any distinguishing features such as date, volume or number. It would be better to group them by series title and display other information*

*within each group. This can be accomplished using an **AZCompactList** classifier rather than **AZList**, and tuning the classifier's format statement.*

6. In the **Design** panel, under the **Browsing Classifiers** section, delete the **AZList** classifiers for **ex. Source** and **ex.Title**.

7. Now add an **AZCompactList** classifier, setting its **metadata** option to **ex.Title**, and add a **DateList** classifier, setting its **metadata** option to **ex.Date**.

8. **Build** the collection, and **preview** the *Titles* list and the *Dates* list.

9. Now we change the format statement for *Titles* to display more information about the documents. In the **Format Features** section of the **Format** panel, select the **ex.Title** classifier (CL1) in the **Choose Feature** list, and **VList** in the **Affected Component** list. Click **<Add Format>** to add this format statement to your collection. Delete the contents of the **HTML Format String** box, and add the following text. (This format statement can be copied and pasted from the file *sample_files → niupepa → formats → titles_tweak.txt*.)

```
<td valign="top">[link][icon][/link]</td>
<td valign="top">
{If}{[numleafdocs],[ex.Title] ([numleafdocs]),
Volume [ex.Volume] Number [ex.Number] Date [ex.Date]}
</td>
```

10. Refresh in the web browser to view the new *Titles* list.

    As a consequence of using the **AZCompactList** classifier, bookshelf icons appear when titles are browsed. This revised format statement has the effect of specifying in brackets how many items are contained within a bookshelf. It works by exploiting the fact that only bookshelf icons define `[numleafdocs]` metadata. For document nodes, Title is not displayed. Instead, Volume, Number and Date information are displayed.

11. The *Dates* list groups documents by date. A numeric date is displayed at the end of each document title, for example 18580601. This is in the Greenstone internal date format, which is crucial for the **DateList** classifier (CL2) to correctly parse date metadata and generate an ordered date list. However, you can make the date look nice by adding a **[Format:]** macro to date metadata.

12. Now we format the date. In the **Format Features** section of the **Format** panel, select the **DateList** classifier *and set **Affected Component** to **DateList***. Replace the last line

    ```
    <td>{Or}{[dc.Date],[exp.Date],[ex.Date]}</td>
    ```

    with

    ```
    <td>{Or}{[dc.Date],[exp.Date],[format:ex.Date]}</td>
    ```

    Refresh in the web browser to view the new *Dates* list.

Displaying scanned images and suppressing dummy text

*When you reach a newspaper, only its associated text is displayed. When either of the **Te Waka o Te Iwi** newspapers is accessed, the document view presents the message "This document has no text." No scanned image information (screen-view resolution or otherwise) is shown, even though it has been computed and stored with the document. This can be fixed by a format statement that modifies the default behaviour for **DocumentText**.*

13. In the **Format Features** section of the **Format** panel, select the **DocumentText** format statement. The default format string displays the document's plain text, which, if there is none, is set to "This document has no text." Change this to the following text. (This format statement can be copied and pasted from the file *sample_files → niupepa → formats → doc_tweak.txt*)

    ```
    <table><tr>
    <td valign=top>[srclink][screenicon][/srclink]</td>
    <td valign=top>[Text]</td>
    </tr></table>
    ```

    *Including `[screenicon]` has the effect of embedding the screen-sized image generated by switching the **screenview** option on in **PagedImagePlugin**. It is hyperlinked to the original image by the construct `[srclink]...[/srclink]`. This is a large image but it may be scaled by your browser.*

    This modification will display screenview image, but does nothing about the dummy text "This document has no text.", which will still be displayed. To get rid of this, edit the **DocumentText** format statement again and replace

    ```
    <td valign=top>[Text]</td>
    ```

    with

    ```
    {If}{[NoText] ne '1',<td valign=top>[Text]</td>}
    ```

14. **Preview** the collection and view one of the **Te Waka o Te Iwi** documents. The line "This document has no text." should now be gone.

Searching at page level

15. The newspaper documents are split into sections, one per page. For large documents, it is useful to be able to search on sections rather than documents. This allows users to more easily locate the relevant information in the document.

16. Go to the **Search Indexes** section of the **Design** panel. Remove the **ex.Source** index. Check the **section** checkbox to build the indexes on section level as well as document level. Make section level the default by selecting its **Default** radio button.

17. **Build** and **preview** the collection.

18. Set the display text used for the level drop-down menu by going to the **Search** section on the **Format** panel. Set the document level text to "newspaper", and the section level text to "page".

    Refresh in your web browser. Compare searching at "newspaper" level with searching at "page" level. A useful search term for this collection is "aroha".

19. You will notice that when searching for individual pages, the newspaper image is displayed in the search results. As these images are very large, this is not very useful. Go to **Format Features** section of the **Format** panel in the Librarian Interface, choose **All Features** in **Choose Feature** list, and select the **VList** format statement from the list of assigned format statements. Remove the second line from the **HTML Format String**:

    ```
    <td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./
    srclink]</td>
    ```

    The reason why this is causing a problem is that the **PagedImagePlugin** does not produce ex. thumbicon, and as a consequence this format statement displays ex.srcicon, which is very large

    While we are here, let's remove the filename from the display. Remove the following from the last line of the format string:

    ```
    {If}{[ex.Source],<br><i>([ex.Source])</i>}
    ```

    Preview the collection—the search results should be back to normal.

20. Now you will notice that page level search results only show the Title of the page (the page number), and not the Title of the newspaper. We'll modify the format statement to show the newspaper title as well as the page number. Also, lets add in Volume and Number information too.

    In the **Format Features** section, select **Search** in **Choose Feature**, and **VList** in **Affected Component**. Click <**Add Format**> to add this format to the collection. The previous changes modified **VList**, so they will apply to all **VList**s that don't have specific format statements. These next changes are made to **SearchVList** so will only apply to search results.

    The extracted Title for the current section is specified as `[ex.Title]` while the Title for the parent section is `[parent:ex.Title]`. Since the same **SearchVList** format statement is used when searching both whole newspapers and newspaper pages, we need to make sure it works in both cases.

    Set the format statement to the following text (it can be copied and pasted from the file *sample_files → niupepa → formats → search_tweak.txt*):

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">
    {If}{[parent:ex.Title],[parent:ex.Title] Volume [parent:ex.Volume]
    Number [parent:ex.Number]: Page [ex.Title],
    [ex.Title] Volume [ex.Volume] Number [ex.Number]}
    <br/><i>({Or}{[parent:ex.Date],[ex.Date],undated})</i></td>
    </td>
    ```

    Preview the search results. Items display newspaper title, Volume, Number and Date, and pages also display the page number.

*The collection you have just built involves a fairly complex document structure. There are two series of newspapers, **Te Waka** and **Te Whetu**.*

*In the **Te Waka** series there are two actual newspapers, Volume 1 Numbers 1 and 2. Number 1 has 4*

pages, numbered 1, 2, 3, 4; Number 2 has 4 pages, numbered 5, 6, 7, 8. The page numbers increase consecutively through each volume, despite the fact that the volume is divided into different Numbers. Each page in the Te Waka series is represented by a single file, a GIF image of the page.

*The **Te Whetu** series has three actual newspapers, Volume 1 Numbers 1, 2, and 3. Number 1 has 4 pages, numbered 1, 2, 3, 4; Number 2 has 5 pages, numbered 5, 6, 7, 8, 9; Number 3 has 5 pages, numbered 10, 11, 12, 13, 14. Again the page numbers increase consecutively through each volume. Each page in this series is represented by two files, a GIF image of the page and a text file containing the OCR'd text that appears on it.*

The key to this structure is in the respective .item *files. Here is a synopsis of the information they contain:*

```
(9-1-1) Te Waka Volume 1 Number 1
    p.1 gif
    p.2 gif
    p.3 gif
    p.4 gif
(9-1-2) Te Waka Volume 1 Number 2
    p.5 gif
    p.6 gif
    p.7 gif
    p.8 gif
(10-1-1) Te Whetu Volume 1 Number 1
    p.1 gif text
    p.2 gif text
    p.3 gif text
    p.4 gif text
(10-1-2) Te Whetu Volume 1 Number 2
    p.5 gif text
    …
    p.9 gif text
(10-1-3) Te Whetu Volume 1 Number 3
    p.10 gif text
    …
    p.14 gif text
```

7.2. Advanced scanned image collection

*In this exercise we build upon the collection created in the **Scanned image collection** exercise. We add a new newspaper by creating an item file for it, add a new newspaper using the extended XML item file format, and modify the formatting.*

Adding another newspaper to the collection

Another newspaper has been scanned and OCRed, but has no item file. We will add this newspaper into the collection, and create an item file for it.

1. In the Librarian Interface, open up the Paged Image collection that was created in exercise **Scanned image collection** if it is not already open (**File → Open...**).

2. In the **Gather** panel, add the folder *sample_files → niupepa → new_papers → 12* to your collection.

   Inside the **12** folder you can see that there are 4 images and 4 text files.

3. Create an item file for the collection. Have a look at an existing item file to see the format. Start up a text editor (e.g. WordPad) to open a new document. Add some metadata. The **Title** for this newspaper is "Te Haeata 1859-1862". The **Volume** is 3, **Number** is 6, and the **Date** is "18610902". (Greenstone's date format is **yyyymmdd**.) Metadata must be added in the form:

   ```
   <Metadata name>Metadata value
   ```

   For this document, the metadata looks like:

   ```
   <Title>Te Haeata 1859-1862
   <Date>18610902
   <Volume>3
   <Number>6
   ```

4. For each page, add a line in the file in the following format:

   ```
   pagenum:imagefile:textfile
   ```

   For example, the first page entry would look like

   ```
   1:images/12_3_6_1.gif:text/12_3_6_1.txt
   ```

   Note that if there is no text file, you can leave that space blank. You need to add a line for each page in the document. Make sure you increment the page number for each line.

5. Save the file using **Filename** *12_3_6.item*, and save as a plain text document. (If you are using Windows, make sure the file isn't saved as *12_3_6.item.txt*.) Back in the **Gather** panel of the Librarian Interface, locate the new file in the **Workspace** tree, and drag it into the collection, adding it to the **12** folder.

6. **Build** the collection and **preview**. Check that your new document has been added.

XML based item file

There are two styles of item files. The first, which was used in the previous section, uses a simple text based format, and consists of a list of metadata for the document, and a list of pages. This format allows specification of document level metadata, and a single list of pages.

The second style is an extended format, and uses XML. It allows a hierarchy of pages, and metadata specification at the page level as well as at the document level. In this section, we add in two newspapers which use XML-based item files.

7. In the **Gather** panel, add the folder *sample_files → niupepa → new_papers → xml* (you need to add the **xml** folder, not the **23** folder) to your collection.

8. Open up the file *xml → 23 → 23__2.item* and have a look at the XML. This is **Number** *2* of the newspaper titled *Matariki 1881*. The contents of this document have been grouped into two sections: **Supplementary Material**, which contains an **Abstract**, and **Newspaper Pages**, which contains the page images (and OCR text).

9. **Build** and **preview** the collection. The xml style items have been included, but the document display for these items is not very nice.

Using process_exp to control document processing

10. Paged documents can be presented with a hierarchical table of contents, or with next and previous page arrows, and a "go to page" box (like we have done so far). The display type is specified by the **documenttype (hierarchy|paged)** option to **PagedImagePlugin**. The next and previous arrows suit the linear sequence documents, while the table of contents suits the hierarchically organised document.

    Ordinarily, a Greenstone collection would have one plugin per document type, and all documents of that type get the same processing. In this case, we want to treat the XML-based item files differently from the text-based item files. We can achieve this by adding two **PagedImagePlugin** plugins to the collection, and configuring them differently.

11. Change the mode in the Librarian Interface to **Library Systems Specialist** (or **Expert**) mode (using **File → Preferences... → Mode**), because you will need to change the order of plugins, and use regular expressions in the plugin options.

    For version 2.71, you'll need to close GLI now then restart it to get the list of plugins to update properly.

12. Go to the **Document Plugins** section of the **Design** panel, and add a new **PagedImagePlugin** plugin. Enable the **create_screenview** option, set the **documenttype** option to **hierarchy** and set the **process_exp** option to **xml.*.item$**.

13. Move this **PagedImagePlugin** plugin above the original one in the **Assigned Plugins** list.

14. The XML based newspapers have been grouped into a folder called *xml*. This enables us to process these files differently, by utilizing the **process_exp** option which all plugins support. The

first **PagedImagePlugin** in the list looks for item files underneath the *xml* folder. These documents will be processed as 'hierarchical' documents. Item files that don't match the process expression (i.e. aren't underneath the *xml* folder) will be passed onto the second **PagedImagePlugin**, and these are treated as 'paged' documents.

Rebuild and **preview** the collection. Compare the document display for a paged document e.g. **Te Waka o Te Iwi, Vol. 1, No. 1** with a hierarchical document, e.g. **Matariki 1881, No. 1**.

Switching between images and text

We can modify the document display to switch between the text version and the screenview and full size versions. We do this using a combination of format statements and macro files.

15. First of all we will add a macro file to the collection. Close the collection in the Librarian Interface. In a file browser outside of Greenstone, locate the Paged Image collection in your Greenstone installation: *Greenstone → collect → pagedima*.

    Also in a file browser, locate the file *sample_files → niupepa → macros → extra.dm*. Copy this file and paste it into the *macros* folder inside the pagedima collection.

16. Back in the Librarian Interface, open up the collection again, and go to the **Format Features** section of the **Format** panel.

17. Select **AllowExtendedOptions** in the **Choose Feature** list, and click **<Add Format>**. Tick the **Enabled** checkbox. This gives us more control over the layout of the page—in this case, we want to replace the standard *DETACH* and *NO HIGHLIGHTING* buttons with buttons that switch between images and text.

18. Select the **DocumentHeading** format item and set it to the following text (which can copied from *sample_files → niupepa → formats → adv_doc_heading.txt*).

```
<div class="heading_title">{Or}{[parent(Top):ex.Title],[ex.Title]}
</div>
<div class="buttons" id="toc_buttons">
{If}{[srcicon],_document:viewfullsize_}
{If}{[screenicon],_document:viewpreview_}
{If}{[NoText] ne '1',_document:viewtext_}
</div>
<div class="toc">[DocTOC]</div>
```

    `{Or}{[parent(Top):ex.Title],[ex.Title]}` outputs the newspaper Title metadata. This is only stored at the top level of the document, so if we are at a subsection, we need to get it from the top (`[parent(Top):ex.Title]`). Note that we can't just use `[parent:ex.Title]` as this retrieves the Title from the immediate parent node, which may not be the top node of the document.

    `_document:viewpreview_, _document:viewfullsize_, _document:viewtext_` are macros defined in *extra.dm* which output buttons for preview, fullsize and text versions, respectively. We choose which buttons to display based on what metadata and text the document has. Note you can view the macros by going to the **Collection Specific Macros** section of the **Format** panel.

    `[DocTOC]` is the document table of contents or "go to page" navigation element. Since we are

using extended options, we need to explicitly specify this for it to appear in the page.

The different pieces are surrounded by `<div>` elements, so that the appropriate styling information can be used.

19. Select the **DocumentText** format statement and set it to the following text (which can be copied from *sample_files → niupepa → formats → adv_doc_text.txt*):

```
{If}{_cgiargp_ eq 'fullsize',[srcicon],
{If}{_cgiargp_ eq 'preview',[screenicon],
{If}{[NoText] ne '1',[Text],[screenicon]}}}
```

This format statement changes the display based on the "p" argument (`_cgiargp_`). This is not used normally for document display, so we can use it here to switch between full size image (`[srcicon]`), preview size image (`[screenicon]`) and text (`[Text]`) versions of each page.

20. **Preview** the collection. View some of the documents—once you have reached a newspaper page, you should get fullsize, preview and text options.

Lab 8: Customization

8.1. Customization: macro files and stylesheets

The appearance of all pages produced by Greenstone is governed by macro files, which reside in the folder *Greenstone → macros*, and images and CSS stylesheets reside in *Greenstone → images*.

A macro takes the form `_macroname_ {macro value}`. Macro names start and end with underscores (_), and the macro value is enclosed in curly brackets ({}). Macro values can be text or HTML, and can include other macros.

Macros are grouped into packages, and different packages control the appearance of different pages. For example, the **home**, **help**, **preferences**, **query**, **document** packages control the home, help, preferences, query, and document pages, respectively. Some macro files contain macros for just one package, for example, *home.dm*, *query.dm*, *document.dm*, while others contain macros for many packages. *base.dm* contains macros used globally, *style.dm* controls the common style of each page, *english.dm*, *french.dm* and other language files contain the text fragments for the entire interface, in that language.

The output of the library program is a page of HTML which is viewed in a web browser. CSS (Cascading Style Sheets) are often used alongside HTML pages to control the formatting, such as layout, colour, font etc. The default Greenstone stylesheet is *Greenstone → images → style.css*.

In this exercise, we customize the macros, images and stylesheets to change the appearance of our library.

Collection specific customisation

Macros can be used to customize single collections by adding them to a file called *extra.dm* in the *macros* directory of a collection.

We use the Word and PDF collection (from exercise **A collection of Word and PDF files**) as the example for this exercise, but it can be done with any collection. Open up this collection (**reports**) in the Librarian Interface.

1. Go to the **Format** panel, and select **Collection Specific Macros** from the left hand list. This section allows you to edit the collection's *extra.dm* macro file.

2. First, we change the title of the **About this collection** section of the about page. Add the following text in the edit box (which can be copied from the file *about_tweak.txt* in the *sample_files → custom* folder):

```
package about

_textabout_ {
<div class="section">
<h3>Very Interesting Reports Collection.</h3>
_Global:collectionextra_
</div>
}
```

   Preview the collection by pressing the **<Preview Collection>** button. The *About* page will have a

new title underneath the search form.

3. Next we add a footer to each page. Add the `_footer_` macro to the end of the edit box (which can be copied from the file *footer_tweak.txt* in the *sample_files → custom* folder):

```
package Style

_footer_ {
_pagefooterextra_ <center><small>Copyright 2006 My Awesome Digital
Library</small></center> _endspacer__htmlfooter_
}
```

The `<center>` and `<small>` HTML tags center the text, and make it a smaller size than the rest of the page.

4. Preview the changes in a web browser. Each page should now have the new text at the bottom.

5. Putting text in the main `_footer_` macro adds it to all pages of this collection. To add a footer just to a particular page, use `_pagefooterextra_` in the appropriate package. For example, lets add some more text to the footer, this time just on the *About* page.

   Add the following text immediately after the line `package about` :

```
_pagefooterextra_ {Collection generated by Me.}
```

   Preview the *About* page in a web browser. The *About* page should now display the new text, while the other pages won't.

6. Next we'll do some style customisations. Add the following text below the `_footer_` macro (which can be copied from the file *red_tweak.txt* in the *sample_files → custom* folder)

```
_collectionspecificstyle_ {
<style type="text/css">
/*clear the use of a background image */
body.bgimage \{ background-image: none; \}
/* set the background color to pink */
body \{ background: pink; \}
/* clear the background image for the navigation bar, and set its
color to red */
div.navbar \{ background-image: none; background-color: red; \}
/* clear the background image for the divider bars, and set their
color to red */
div.divbar \{ background-image: none; background-color: red; \}
</style>
}
```

   /*...*/ around a line signals a comment, and this style element will be ignored.

   Preview the collection. The **reports** collection will now have a pink background, and the navigation bar and divider bars will be red. These changes will only affect this collection.

Any macros from the general macro files can be copied into a collection's *extra.dm* file and modified. Remember to include the package declaration to make sure that the macros get applied to the correct page (s).

The style modifications made above were minor. The collection still uses the majority of the standard style file. The style declarations in the `_collectionspecificstyle_` macro get appended to the default ones. To completely change the appearance of a collection, we can use a new style sheet altogether.

7. Add the following text (which can be copied from the file *css_tweak.txt* in the *sample_files → custom* folder) after the last modifications:

```
_cssheader_ {
<link rel="stylesheet" href="_httpcimages_/style-blue.css"
type="text/css"
  title="Blue Style" charset="UTF-8">
}
```

   Outside of the Librarian Interface, locate the collection folder *Greenstone → collect → reports*. Create an *images* folder inside this (if not already present), and copy the file *sample_files → custom → style-blue.css* into this folder.

   Preview the collection; the about page should look radically different.

Changing the colour of the page title and page text

In the previous exercises we changed a single collection. Now we change all the pages in our Greenstone installation by modifying style and macro files outside the Librarian Interface. First, we format the page so that some other parts are blue. Preview any collection after each change to make sure that it has worked properly. On Windows, macro file changes require a restart of the Greenstone local library server. Stylesheet changes may require a forced reload in the web browser.

8. The majority of the style definitions reside in an external style file, *Greenstone → images → style.css*, and most style changes involve modifying that file. Open *Greenstone → images → style.css* in a text editor, e.g. WordPad (and save a .backup copy). Make the following modifications. You might want to preview after each one to see the effect.

   Change some of the colours:

   - Find the `body` style instructions:

```
body {
background: #ffffff;
}
```

     Add `color: teal;`
   - For `a.collectiontitle`, set `color` to `blue`.
   - For `p.collectiontitle`, add `color: blue;`

   Preview the collection. Now text in the page body is a light green color (teal), and the font of the collection title has changed from black to blue.

(If a collection title image is used, you won't see the change on the collection title.)

9. Lets switch the positions of the HOME, HELP and PREFERENCES buttons and the collection

name or image.

- For `div.pageinfo`, set both `float` and `text-align` to `left`.
- For `div.collectimage`, set `float` and `text-align` to `right`.

The look of your library should now be substantially different. The HELP, HOME and PREFERENCES buttons are in the left upper corner whereas the collection title is switched to the right of the page.

10. Now we will customize the default Greenstone header image and the background image. Two new images for this exercise can be found in *sample_files → custom*. Copy *newbgimg.gif*, *newheadimg.gif* from the *custom* folder into the *Greenstone → images* folder.

11. Open the file *Greenstone → macros → home.dm* in a text editor. Find each occurrence of `gsdlhead.gif` in this file (there are two) and replace with `newheadimg.gif`. (If you are using WordPad, you can use **Edit → Find** to search for the text.)

    Save *home.dm* and close the file.

12. Open the file *Greenstone → macros → style.dm* with the text editor. Locate the following part of the file (this is part of the `_cssheader_` macro):

    ```
    <style type="text/css">
    body.bgimage \{ background-image: url("_httpimg_/chalk.gif"); \}
    ```

    Use copy and paste on the `body.bgimage` line to make it look like this:

    ```
    <style type="text/css">
    /*body.bgimage \{ background-image: url("_httpimg_/chalk.gif"); \}
    */
    body.bgimage \{ background-image: url("_httpimg_/newbgimg.gif"); \}
    ```

    Here we are changing the background image for the `bgimage` section of the `body` of the page to `newbgimg.gif`.

    Save *style.dm* and close the file.

13. Preview the home page in a web browser. (On Windows, restart the Greenstone library server.) The header and background of every page of each collection should now use the new graphics.

Make your own Greenstone home page

You can make radical change to a page by changing the macro file completely. For example, here we use a alternative to the home page which we have prepared for you in advance and included in your Greenstone installation.

14. Open the file *Greenstone → etc → main.cfg* in a text editor. Locate the *macrofiles* list:

    ```
    # The list of display macro files used by this receptionist
    macrofiles tip.dm style.dm base.dm query.dm help.dm pref.dm about.
    dm \
    ```

```
                document.dm browse.dm status.dm authen.dm users.dm html.
dm \
                extlink.dm gsdl.dm extra.dm home.dm collect.dm docs.dm \
                bsummary.dm gti.dm gli.dm nav_css.dm usability.dm \
                ...
```

Change the text `home.dm` to `yourhome.dm`. Save and close the file.

15. Preview the newly structured home page in a web browser. (On Windows, restart the Greenstone library server.) Look at the file macros/yourhome.dm in a text editor to see how these changes are expressed.

16. Reverse this last change by changing `yourhome.dm` back to `home.dm` in the file *Greenstone → etc → main.cfg*. You may also like to reverse the other changes you have made.

    The final part of this exercise looks at how we determined which images needed replacing, and which macro files should be edited.

How to determine which images to replace (advanced)

17. In the step 10 of this exercise we replaced the default background (**chalk.gif**) and header (**gsdlhead.gif**) images with new ones. To do this we needed to change the image names in the macro files. How did we know which images we were replacing and which macro files to edit? This exercise shows you how to find out.

18. To find out the names of the images to replace, go to the home page of your digital library in a browser. Right-click on the header image ("Greenstone digital library software") and select "Save picture as". A dialog will pop up and will display the image name: ***gsdlhead.gif*** (or ***newheadimg.gif*** if you are using the new header). Click Cancel to close the dialog—you don't need to save the images. Do the same for the background image by right clicking on the left hand green (or blue) swirly bar. This time choose "Save background as" to find the name: ***chalk.gif*** (or ***newbgimg.gif***), then click Cancel.

19. These instructions apply to Internet Explorer. Other browsers may have other options in the right-click menu. For example, Mozilla provides "View Image" and "View Background Image" options. Using these options will put the path to the image in the browser address box, and the name can be seen from this.

20. Once you have identified the names of the images to be replaced, you need to find out where they occur in the macro files. To do this, search the macro files for the image names using the **find** program, which is run in a command prompt. Open a command prompt using **Start → Programs → Accessories → Command Prompt**, or **Start → Run** and enter `cmd` as the name of the program to run.

    You can type `find/?` to see a description of the program and its arguments.

    To search the macro files for ***gsdlhead.gif*** type

    ```
    find "gsdlhead.gif" "C:\Program Files\Greenstone\macros\*.dm"
    ```

    ***.dm** means all files ending in **.dm**. A list of all macro files will be displayed, along with any matches. You will see that *home.dm* and *exported_home.dm* both contain **gsdlhead.gif**. *home.dm*

in the one you want to edit—*exported_home.dm* is used for the home page when you export a collection to CD-ROM.

Do the same thing for ***chalk.gif***:

```
find "chalk.gif" "C:\Program Files\Greenstone\macros\*.dm"
```

base.dm and *style.dm* are the only files that mention this image.

Close the command prompt.

Lab 9: Sharing collections with OAI-PMH

9.1. Open Archives Initiative (OAI) collection

This exercise explores service-level interoperability using the Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH). So that you can do this on a stand-alone computer, we do not actually connect to the external server that is acting as the data provider. Instead we have provided an appropriate set of files that take the form of XML records produced by the OAI-PMH protocol.

One of Greenstone's documented example collections is sourced over OAI. This exercise takes you through the steps necessary to reconstruct it. (Note: this example is a collection of images: you will not be able to build it unless ImageMagick is installed on your computer.) You may wish to take a look at the documented example collection OAI demo now to see what this exercise will build.

1. Start a new collection called **OAI Service Provider**. Fill out the fields with appropriate information.

2. In the **Gather** panel, locate the folder *sample_files → oai → sample_small → oai*. Drag this folder into the collection and drop it there.

3. During the copy operation, a popup window appears asking whether to add **OAIPlugin** to the list of plug-ins used in the collection, because the Librarian Interface has not found an existing plug-in that can handle this file type. Press the **<Add Plugin>** button to include it.

   The files for this collection consist of a set of images (in *JCDLPICS → srcdocs*) and a set of OAI records (in *JCDLPICS*) which contain metadata for the images.

When files are copied across like this, the Librarian Interface studies each one and uses its filename extension to check whether the collection contains a corresponding plug-in. No plug-in in the list is capable of processing the OAI file records that are copied across (they have the file extension .oai), so the Librarian Interface prompts you to add the appropriate plug-in.

*Sometimes there is more than one plug-in that could process a file—for example, the .xml extension is used for many different XML formats. The popup window, therefore, offers a choice of all possible plug-ins that matched. It is normally easy to determine the correct choice. If you wish, you can ignore the prompt (click **<Don't Add Plugin>**), because plug-ins can be added later, in the **Document Plugins** section of the **Design** panel.*

4. You need to specifiy which document the OAI metadata values should be attached to. In the **Design** panel, select the **Document Plugins** section, then select the **OAIPlugin** and click **<Configure Plugin...>**. Locate the **document_field** option in the popup window and select **dc. Resource Identifier** from the drop-down list. Click **<OK>**.

5. You also need to configure the image plug-in. Select the **ImagePlugin** line in the **Document Plugins** section and click **<Configure Plugin...>**. In the resulting popup window locate the **screenviewsize** option, switch it on, and type the number **300** in the box beside it to create a screen-view image of 300 pixels. Click **<OK>**.

6. Now switch to the **Create** panel and **build** and **preview** the collection.

OAIPlugin will process the OAI records, and assign metadata to the images, which are processed by **ImagePlugin**.

Like other collections we have built by relying on Greenstone defaults, the end result is passable but can be improved. The next steps refine the collection using the metadata harvested by OAI-PMH into the .oai files.

7. In the **Browsing Classifiers** section of the **Design** panel, delete the two **AZList** classifiers (**ex. Title** and **ex.Source**).

8. Add an **AZCompactList** classifier based on **dc.Subject and Keywords** metadata.

9. Now add an **AZCompactList** classifier based on **dc.Description** metadata. In its configuration panel set **mingroup** to **2**, **mincompact** to **1**, **maxcompact** to **10** and **buttonname** to **Captions**.

   Setting **mingroup** to 2 will mean that two or more documents with the same description will be grouped into a bookshelf; the default **mingroup** of 1 means that every document will get a bookshelf. **mincompact** and **maxcompact** control how many documents are grouped into each section of the horizontal A-Z list. In this case, each group can have as few as one document, and no more than ten.

10. In the **Search Indexes** section of the **Design** panel, delete all indexes and add a new one based on **dc.Description** metadata.

11. **Build** the collection and **preview** it.

Tweaking the presentation with format statements

12. In the **Format** panel, select **Format Features**. First replace the **VList** format statement with the following (which can be copied from the file *vlist_tweak.txt* in the *sample_files → oai →format_tweaks* folder).

```
<td>
  {If}{[numleafdocs],[link][icon][/link],[link][thumbicon][/link]}
</td>
<td valign=middle>
  {If}{[numleafdocs],[Title],<i>[dc.Description]</i>}
</td>
```

   This format statement customizes the appearance of vertical lists such as the search results and captions lists to show a thumbnail icon followed by Description metadata. Greenstone's default is to use extracted metadata, so `[Description]` is the same as `[ex.Description]`.

13. Next, select **DocumentHeading** from the **Choose Feature** pull-down list and change its format statement to:

```
<h3>[dc.Subject]</h3>
```

   *The document heading appears above the DETACH and NO HIGHLIGHTING buttons when you get to a document in the collection. By default **DocumentHeading** displays the document's **ex. Title** metadata. In this particular set of OAI exported records, titles are filenames of JPEG*

images, and the filenames are particularly uninformative (for example, 01dla14). You can see them in the ***Enrich*** *panel if you select an image in oai → JCDLPICS → srcdocs and check its* ***ex.Source*** *and* ***ex.Title*** *metadata. The above format statement displays* ***dc.Subject and Keywords*** *metadata instead.*

14. Finally, you will have noticed that where the document itself should appear, you see only "This document has no text.". To rectify this, select **DocumentText** in the **Choose Feature** pull-down list and use the following as its format statement (this text is in *doctxt_tweak.txt* in the *format_tweaks* folder mentioned earlier):

```
<center><table width=_pagewidth_ border=1>
  <tr><td colspan=2 align=center>
    <a href=[dc.OrigURL]>[screenicon]</a></td></tr>
  <tr><td>Caption:</td><td> <i>[dc.Description]</i> <br>
    (<a href=[dc.OrigURL]>original [ImageWidth]x[ImageHeight]
[ImageType] available</a>)
    </td></tr>
  <tr><td>Subject:</td><td> [dc.Subject]</td></tr>
  <tr><td>Publisher:</td><td> [dc.Publisher]</td></tr>
  <tr><td>Rights:<td> [dc.Rights]</td></tr>
</table></center>
```

This format statement alters how the document view is presented. It includes a screen-sized version of the image that hyperlinks back to the original larger version available on the web. Factual information extracted from the image, such as width, height and type, is also displayed.

15. Format statements are processed by the runtime system, so the collection does not need to be rebuilt for these changes to take effect. Click <**Preview Collection**> to see the changes.

To expedite building, this collection contains fewer source documents than the pre-built version supplied with the Greenstone installation. However, after these modifications, its functionality is the same.

9.2. Downloading over OAI

*The previous exercise did not obtain the data from an external OAI-PMH server. This missing step is accomplished either by running a command-line program or by using the **Download** panel in the Librarian Interface. This exercise shows you how to do this using both methods.*

Downloading using the Librarian Interface

1. In the Librarian Interface, switch to the **Download** panel. Select **OAI** from the list of download types on the left hand side.

2. In the **url** box, type in the following URL:

   http://rocky.dlib.vt.edu/~jcdlpix/cgi-bin/OAI/jcdlpix.pl

3. We want to download the documents as well as the metadata, so tick the **Get document** checkbox.

4. If your computer is behind a firewall or proxy server, you will need to edit the proxy settings in the Librarian Interface. Click the **<Configure Proxy...>** button. Switch on the **Use proxy connection?** checkbox. Enter the proxy server address and port number in the **Proxy Host:** and **Proxy Port:** boxes. Click **<OK>**.

5. Now click **<Download>**. If you have set proxy information in **Preferences...**, a popup will ask for your user name and password. Once the download has started, a progress bar appears in the lower half of the panel that reports on how the downloading process is doing.

6. Downloaded files are stored in a top-level folder called **Downloaded Files** that appears on the left-hand side of the **Gather** panel. These files can then be added to a collection.

Downloading using the command line

For command line downloading to work, your computer must have a direct connection to the Internet—being behind a firewall may interfere with the ability to download the information. You will need to use the Librarian Interface for downloading if you are behind a firewall.

7. Close the Librarian Interface.

   We will work with the OAI collection used in exercise **Open Archives Initiative (OAI) collection**. You may have noticed that its internal name is **oaiservi**.

8. In a text editor (e.g. WordPad), open the collection's configuration file, which is in *Greenstone → collect → oaiservi → etc → collect.cfg*. Add the following line (all on one line):

   ```
   acquire OAI -src http://rocky.dlib.vt.edu/~jcdlpix/cgi-bin/OAI/
   jcdlpix.pl -getdoc
   ```

   Although the position of this line is not critical, we recommend that you place it near the

beginning of the file, after the public and creator lines but before the index line. Save the file and quit the editor.

9. Delete the contents of the collection's *import* folder. This contains the canned version of the collection files, put there during the previous exercise. Now we want to witness the data arriving anew from the external OAI server.

10. Open a DOS window to access the command-line prompt. This facility should be located somewhere within your **Start → Programs** menu, but details vary between different Windows systems. If you cannot locate it, select **Start → Run** and enter *cmd* in the popup window that appears.

11. In the DOS window, move to the home directory where you installed Greenstone. This is accomplished by something like:

    ```
    cd C:\Program Files\Greenstone
    ```

12. Type:

    ```
    setup.bat
    ```

    to set up the ability to run Greenstone command-line programs.

13. Change directory into the folder containing the OAI Services Provider collection you built in the last exercise.

    ```
    cd collect\oaiservi
    ```

    Even though the collection name used capital letters the directory generated by the Librarian Interface is all lowercase.

14. Run:

    ```
    perl -S importfrom.pl oaiservi
    ```

    Greenstone will immediately set to work and generate a stream of diagnostic output. The importfrom.pl program connects to the OAI data provider specified in collection configuration file (it does this for each "acquire" line in the file) and exports all the records on that site.

15. The downloaded files are saved in the collection's import folder. Once the command is finished, everything is in place and the collection is ready to be built. Confirm you have successfully acquired the OAI records by rebuilding the collection.

Lab 10: Miscellaneous

10.1. Downloading files from the web

The Greenstone Librarian Interface's Download panel allows you to download individual files, parts of websites, and indeed whole websites, from the web.

1. Start a new collection called **webtudor**, and base it on **-- New Collection --**.

2. In a web browser, visit http://englishhistory.net, follow the link to *Tudor England*, and click <**Enter**>. You should be at the URL

   http://englishhistory.net/tudor.html

   This is where we started the downloading process to obtain the files you have been using for the **tudor** collection. You could do the same thing by copying this URL from the web browser, pasting it into the **Download** panel, and clicking the <**Download**> button. However, several megabytes will be downloaded, which might strain your network resources—or your patience! For a faster exercise we focus on a smaller section of the site.

3. Go to the **Download** panel by clicking its tab. There are five download types listed on the left hand side. For this exercise, we only use the **Web** type. Make sure this is selected in the list.

   Enter this URL

   http://englishhistory.net/tudor/citizens/

   into the **Source URL** box. There are several other options that govern how the download process proceeds. To see a description of an option, hover the mouse over it and a tooltip will appear. To copy just the *citizens* section of the website, switch on the **Only filese below URL** option by checking its box and set the **Download Depth** option to 1. If you don't do this (or if you miss out the terminating "/" in the URL), the downloading process will follow links to other areas of the *englishhistory.net* website and grab those as well.

4. If your computer is behind a firewall or proxy server, you will need to edit the proxy settings in the Librarian Interface. Click the <**Configure Proxy...**> button. Switch on the **Use proxy connection?** checkbox. Enter the proxy server address and port number in the **Proxy Host:** and **Proxy Port:** boxes. Click <**OK**>.

5. Now click <**Download**>. If you have set proxy information in **Preferences...**, a popup will ask for your user name and password. Once the download has started, a progress bar appears in the lower half of the panel that reports on how the downloading process is doing.

   *More detailed information can be obtained by clicking <**View Log**>. The process can be paused and restarted as needed, or stopped altogether by clicking <**Close**>. Downloading can be a lengthy process involving multiple sites, and so Greenstone allows additional downloads to be queued up. When new URLs are pasted into the **url** box and <**Download**> clicked, a new progress bar is appended to those already present in the lower half of the panel. When the currently active download item completes, the next is started automatically.*

6. Downloaded files are stored in a top-level folder called **Downloaded Files** that appears on the left-hand side of the **Gather** panel. You may not need all the downloaded files, and you choose which you want by dragging selected files from this folder over into the collection area on the right-hand side, just like we have done before when selecting data from the *sample_files* folder. In this example we will include everything that has been downloaded.

   Select the *englishhistory.net* folder within **Downloaded Files** and drag it across into the collection area.

7. Switch to the **Create** panel to **build** and **preview** the collection. It is smaller than the previous collection because we included only the *citizens* files. However, these now represent the latest versions of the documents.

10.2. Editing metadata sets

GEMS (Greenstone Editor for Metadata Sets) can be used to modify existing metadata sets or create new ones. GEMS is launched from the Librarian Interface when you want to create a new metadata set, or edit an existing one. In this exercise, we run GEMS outside of the Librarian Interface.

Running GEMS

1. Start the Greenstone Editor for Metadata Sets (GEMS), for versions 2.81 and greater:

   Start → All Programs → Greenstone-2.81 → Metadata Set Editor

   and for versions below 2.81:

   Start → All Programs → Greenstone Digital Library Software → Greenstone Editor for Metadata Sets

2. GEMS starts up with no metadata set loaded. You can start a new set, or open an existing one, from the **File** menu.

Creating a new metadata set

3. In this exercise, we will create a new metadata set. In order to save time, we will base it on an existing one: Development Library Subset. From the **File** menu, select **File → New...**. A popup window appears: **New Metadata Set**. Fill in the fields. Use "My Metadata Set" for the **Metadata set title:**, "my" for the **Metadata set namespace:**, and select "Development Library Subset Example Metadata" from the **Base this metadata set on:** drop down list. Click **<OK>**.

4. The new metadata set will be displayed. The left hand side list the elements (and sub-elements, if any) for the set, and the right hand side displays the set or element attributes. Since the new set was based on the Development Library Subset metadata set, it already contains all the elements from that set.

Adding a new element to a metadata set

5. Right click on **My Metadata Set** in the left hand tree (or in the blank space in the left hand side) and choose **Add Element** from the menu that appears. In the popup window, type "Category" for the new element name, and click **<OK>**. The new element will appear in the list.

6. In the right hand side, the default attributes will appear for the new element. "Label" and "definition" are used in the Librarian Interface when displaying metadata elements and their descriptions (the "definition" is shown as additional text for the element). These attributes can be set in multiple languages.

7. Save the new metadata set by **File → Save**, then close the GEMS by **File → Exit**.

10.3. Building and searching with different indexers

Greenstone supports three indexers **MG**, **MGPP** and **Lucene**.

MG is the original indexer used by Greenstone which is described in the book **"Managing Gigabytes"**. It does section level indexing and compression of the source documents. **MG** is implemented in C.

MGPP is re-implementation of **MG** that provides word-level indexes and enables proximity, phrase and field searching. **MGPP** is implemented in C++ and is the default indexer for new collections.

Lucene (http://lucene.apache.org/) is java-based full-featured text indexing and searching system developed by Apache. It provides a similar range of search functionality to MGPP with the addition of single-character wildcards and range searching. It was added to Greenstone to facilitate incremental collection building, which **MG** and **MGPP** can't provide.

Build with Lucene

1. Start a new collection (**File → New...**) called **Demo Lucene** and base it on the **Greenstone demo (demo)** collection, fill out its fields appropriately.

2. In the **Gather** panel, click **Documents in Greenstone Collections** and click **Greenstone demo (demo)**, it will show the documents in the **Greenstone demo** collection. Drag all 11 folders underneath *Greenstone demo (demo)* into the new collection.

   *If you haven't installed the **Greenstone demo (demo)** collection yet, you can download the demo. zip file from the link above, unzip it and put it into the collect folder in your Greenstone installation.*

3. Go to the **Enrich** panel, look at the metadata that associated with each directory. Go to the **Search Indexes** section in the **Design** panel. The **MG indexer** is in use because the original **Greenstone Demo** collection, which this collection is based on, uses **MG indexer**.

4. Click the **Change...** button at the right top corner of the panel. A new window will pop up for selecting the Indexers. After selecting an indexer, a brief description will appear in the box below. Select Lucene and click **OK**. Please note that the **Assigned Indexes** has changed accordingly.

5. **Build** and **preview** the collection.

Search with Lucene

6. Lucene provides single letter and multiple letter wildcards and range searching. The query syntax could be quite complicated (for more information please see http://lucene.apache.org/java/docs/queryparsersyntax.html). Here we will learn how to use the wildcards while constructing queries.

7. ***** is a multiple letter wildcard. To perform a a multiple letter wildcard search, append ***** to the end of the query term. For example, *econom** will search for words like *econometrics, economist, economical, economy*, which have the common part *econom* but different word endings.

8. To perform a single letter wildcard search, use **?** instead. For example, search for *economi??* will only match words that have two and only two letters left after *economi*, such as *economist*, *economics*, and *economies*.

9. Please note that stopwords are used by default with Lucene indexer, so search for words like *the* will match 0 document. There is also a message on the search page saying that such words are too common and were ignored.

Build with MGPP

10. Start a new collection called **Greenstone Demo MGPP** and also base it on the **Greenstone demo (demo)**.

11. In the **Gather** panel, drag all the 11 folders from → *Greenstone demo (demo)* into the new collection.

12. Go to the **Search Indexes** section in the **Design** panel, click the **Change...** button and select **MGPP**. Click **OK**. Check the **Assigned Indexes** has changed accordingly.

13. There are three options at the bottom of the panel — **Stem**, **Casefold** and **Accent fold**. Notice that **Stem** and **Casefold** are enabled. Once an option is enabled, it will also appear in the collection's **PREFERENCES** page.

14. In the **Indexing Levels** section, also select **section**.

15. **Build** and **preview** the collection.

Search with MGPP

16. MGPP supports stemming and casefolding. By default search in collections built with MGPP indexer is set to **whole word must match** and **ignore case differences**. So search *econom* will return 0 document. Search for *fao* and *FAO* return the same result — 78 word counts and 9 matched documents.

    Go to the **PREFERENCES** page by click the **PREFERENCES** button at the top right corner. You can see that the **Word endings:** option is set to **whole word must match** and the **Case differences:** option is set to **ignore case differences**.

17. Sometimes we may want to ignore word endings while searching so as to match different variations of the term. Go to the **PREFERENCES** page and change the **Word endings:** option from **whole word must match** to **ignore word endings**. Click the **set preferences** button. Click **Search**. This time try search for *econom* again, 9 documents are found.

    Please note that word endings are determined according to the third-party stemming tables incorporated in Greenstone, not by the user. Thus the searches may not do precisely what is expected, especially when cultural variations or dialects are concerned. Besides, not all languages support stemming, only English and French have steming at the moment.

    Go to the **PREFERENCES** page and change back to **whole word must match** to avoid confusion later on. Click the **set preferences** button.

18. Sometimes we may want to search the exact term, that is, differentiate the upper cases from lower cases. Set the **Case differences:** option from **ignore case differences** to **upper/lower case must match**. Click the **set preferences** button. Click **Search**. Now try search for *fao* and *FAO* respectively this time, notice the difference in the results?

    Go back to the **PREFERENCES** page and change the **Case differences:** option back to **ignore case differences** to avoid confusion later on. Click **set preferences** button.

Use search mode hotkeys with query term

*MGPP have several hotkeys to set search modes for a query term. These hotkeys explicitly set the **Word endings:** option and the **Case differences:** option for the query being constructed.*

19. **#s** and **#u** are hotkeys for the **Word endings:** option. Appending **#s** to a query term will specifically enable the **ignore word endings** function. For example, try search for *econom#s*, 7 documents are found, which is the same as in step 17. Remember that we have set it back to **whole word must match**. This means using hotkeys will override the current preference settings.

20. Appending **#u** to a query term will explicitly set the current search to **whole word must match**.

    Note that using hotkeys will only affect that query term. That is, hotkeys are used per term. For example, if a query expresssion contains more than one terms, some tems can have hotkeys and others not, and the hotkeys can be different for different terms. This provides a fine-grained control of the query, whereas changing settings in the **PREFERENCES** page will affect the query as a whole.

21. Hotkeys **#i** and **#c** control the case sensitivity. Appending **#i** to a query term will explicitly set the search to **ignore case differences** (ie. case insensitive).

22. On the contrary, appending **#c** will specifically turn off the casefolding, that is, **upper/lower case must match**. For example, search for *fao#c* returns 0 document.

23. Finally, the hotkeys can also be used in combination. For example, you can append *#uc* to a query term so as to match the whole term (without stemming) and in its exact form (differentiate upper cases and lower cases).

A quick reference of the search mode hotkeys in MGPP

```
Word endings:

    #s    ignore word endings

    #u    whole word must match

Case differences:

    #i    ignore case differences

    #c    upper/lower case must match
```

How to format the output of your collection

From GreenstoneWiki

The full list of formatting options is shown here. But for more information about how to use these options, the developer's guide (http://prdownloads.sourceforge.net/greenstone/Develop-en.pdf) is the place to go.

Contents

- 1 Site-wide formatting options
- 2 Collection-specific formatting options
- 3 Formatting Lists
- 4 Formatstring items
- 5 Extended metadata names
- 6 Extended Formatstring items
- 7 Conditional expressions in formatstrings

Site-wide formatting options

These should be placed in `gsdl/etc/main.cfg`.

Syntax: **SiteFormat <option-name> <option-value>**

Item	Description
HomePageCols int	Set the number of columns used to display collections on the home page. Default: 3
HomePageType pulldown	Display the collection list on the home page as a pulldown menu, rather than using the default table of collection images. This alters the html that appears in the dynamically generated _homeextra_ macro. You can then move this macro around in home.dm. Default: not set

Collection-specific formatting options

These should be placed in gsdl/collect/<collname>/etc/collect.cfg.

Syntax: **format <option-name> <option-value>**

Item	Description
DocumentImages true/false	If true, display a cover image at the top left of the document page Default: false
DocumentTitles true/false	If DocumentImages is false, and this is true, use DocumentHeading to display the title. Default: true

DocumentHeading formatstring	This is used for a document heading at the top left if DocumentImages is false and DocumentTitles is true. Default: `{Or}{[parent(Top):Title],[Title],untitled}  [Title]`
DocumentContents true/false	Display table of contents (if document is hierarchical), or next/previous section arrows and "page k of n" text (if document is paged) Default: true
DocumentButtons string	. Default: "Detach|Highlight"
DocumentText formatstring	Format of the text to be displayed on a document page Default: `<center> <table width=537> <tr><td>[Text]</td></tr> </table> </center>`
DocumentArrowsTop true/false	Display next/previous section arrows at top of document, underneath the navigation bar, on document page Default: false
DocumentArrowsBottom true/false	Display next/previous section arrows at bottom of document page Default: true
DocumentSearchResultLinks true/false	Display *next search result link /previous search result link* at top of a document page Default: false
DocumentUseHTML true/false	If true, each document is displayed in a separate frame. The Preferences page will also change slightly, adding options applicable to a collection of HTML documents. Default: false
NavigationBar pulldown	If set, provides a drop down list in place of the usual navigation bar (that contains search and classifier options). This alters the html that appears in the dynamically generated `_navigationbar_ macro`. Default: not set
AllowExtendedOptions true/false	This allows the entire content of the document page to be controlled by format statements. Use DocumentHeading and DocumentText to format the document. This option prevents the other hard coded stuff (table of contents, buttons etc) from being output. It effectively disables the DocumentContents, DocumentButtons, DocumentImages format options. New format items are provided for use in format statements if AllowExtendedOptions is true (see table below) Default: false

Formatting Lists

The standard use of format statements is for the lists in search results, classifiers etc. Here is a list of the various lists available for format, and what they control. Note that classifiers are numbered from 1 upwards, in the order that they appear in the config file.

Item	Description
VList	Applies to all vertical lists, unless overridden by a more specific format item. These include search results, classifier lists, and document table of contents
HList	Applies to all horizontal lists. Horizontal lists are often used in classifiers, particularly AZ[Compact][Section]Lists
DateList	Applies to all date lists - these are the vertical lists generated by a DateList classifier.
SearchVList	The vertical list of search results
DocumentVList	The document table of contents
CL1VList	Applies only to the vertical list of classifier 1
CL1HList	Applies only to the horizontal list of classifier 1

CL1DateList	Applies only to the DateList in classifier 1

Formatstring items

Item	Description
[link][/link]	Link to the document (Greenstone version)
[href]	The href of the link to the document (Greenstone version), without the <a> tag
[srclink][/srclink]	Link to the original document (only if the original was converted to another form)
[icon]	An appropriate icon for a classifier/document node. E.g. bookshelf, book, chapter, page
[srcicon]	An appropriate icon for the original source document. E.g. Word, PDF, PS icon.
[num]	The document number (position in the search results - useful for debugging)
[numleafdocs]	The number of documents below the current classifier node. This is often used as a test for classifier nodes, as numleafdocs will not be set for documents. This allows different formatting for classifier nodes and document nodes in a hierarchy.
[Text]	The text of the current section
[RelatedDocuments]	Related Documents info (if available). This is a vertical list of Titles (or Subjects if Titles aren't available) that link to the related documents. It is based on "relation" metadata, which is a space separated list of collection,OID pairs.
[highlight][/highlight]	These are used for 'highlighting' (actually bolding) the selected section in a hierarchical table of contents, and the selected node in a classifier. Apart from those two cases, this has no effect. If you actually want to highlight/bold/italicise something in a list, and have it apply to all items in the list, then either use actual html tags, like , <u></u> and , or use the `_starthighlight_` and `_endhighlight_` macros (defined in *macros/base.dm*).
[Summary]	Displays Summary metadata if available, otherwise displays a short summary created on the fly.
[DocOID]	The internal identifier of the current section of the document
[DocTopOID]	The (top level) internal identifier of the current document (available 2.72)
[DocRank]	The rank of the current document - used in search results
[DocImage]	The URL to the cover image of the document
[collection]	The directory name of the collection this document is from - useful in cross-collection searching. (version 2.61)
[collection:meta-name]	A collection metadata for the collection this document is from - useful in cross-collection searching. E.g. [collection:collectionname]. This will display in the current language if an appropriate version is available. (version 2.61)
[metadata-name]	The value of this metadata element for the document

Extended metadata names

There are a few options for displaying metadata. The basic way is to specify e.g. [Title] or [dc.Title]: this displays the value of that particular metadata element for the current document/section. Metadata names can be prefixed by parent: or sibling. The following examples all use Title or Subject metadata, but any metadata could be used, including ones with namespaces (e.g. dc.Title). Any metadata name can also be prefixed by "cgisafe:". This results in the value being formatted so that it is safe to put in a URL.

Item	Description
[parent:Title]	The Title of the immediate parent section
[parent(Top):Title]	The Title of the topmost parent section
[parent(All):Title]	All Titles of the parent sections, separated by nothing
[parent(All': '):Title]	All Titles of the parent sections, separated by ": " (or whatever appears inside the ' ')
[child:Subject]	The Subjects of all child nodes of the current node, separated by ' '. (child modifier avilable from version 2.61)
[child(All'xxx'):Subject]	The Subjects of all child nodes of the current node, separated by xxx
[child(2):Subject]	The Subject of the second child of the current node. Child numbering starts from 1.
[child(last):Subject]	The Subject of the last child of the current node. 'first' is also a valid option.
[sibling:Subject]	All Subjects of the current section, separated by " ". This is used for displaying metadata where there is more than one value. [Subject] will just display the first value.
[sibling(All' '):Subject]	All Subjects of the current section, separated by .
[sibling(2):Subject]	The second Subject metadata value for the current node. Numbering starts from 1.
[sibling(last):Subject]	The last Subject metadata value for the current node. 'first' is also a valid option.
[parent:sibling:Subject]	sibling can be combined with parent to give all (or specific) values for the parent node(s). All parent and sibling qualifiers can be used. (since version 2.61)
[child:sibling:Subject]	sibling can be combined with child to give all (or specific) values for the child node(s). All parent and child qualifiers can be used. (since version 2.61)
[cgisafe:parent(Top):Title]	The Title of the topmost parent section, made safe for URLs.
[cgisafe:sibling(All' '):Subject]	All Subjects of the current section, separated by , made safe for URLs.
[format:Date]	A nicely formatted version of Date metadata. format: can also be used with Language metadata (since version 2.71)

Extended Formatstring items

These items are only available if AllowExtendedOptions is true.

Item	Description
[DocumentButtonDetach]	The Detach button

[DocumentButtonHighlight]	The Highlight button
[DocumentButtonExpandText]	The Expand Text button
[DocumentButtonExpandContents]	The Expand Contents button
[DocTOC]	The table of contents for a hierarchical document, or the next/previous and go to page x bits for a paged document.

Conditional expressions in formatstrings

Item	Description
{If}{[metadata], action-if-non-null, action-if null}	If there is a value for this metadata element, then output the first clause, otherwise output the second clause. Either clause is optional: if empty, nothing will be done for that case.This is useful for displaying classifier nodes differently to document nodes: {If}{[numleafdocs],format for classifier,format for document}
{If}{[metadata] op value, action-if-true, action-if-not-true}	Can do tests on metadata values. These can be string comparisons, or numeric comparisons. Valid operators are: **String Numeric Meaning** eq == equals ne != not equals gt > greater than ge >= greater than or equal to lt < less than le <= less than or equal to sw starts with ew ends with Note that only eq and ne are available for Greenstone versions 2.60 and earlier.
{Or}{[metadata], [metadata2], [metadata3]...}	Each metadata is evaluated in turn, and the first one that exists is output Useful for cases where there are different namespaced version of the same metadata, e.g. {Or}{[dc.Title],[dls.Title],[Title],Untitled}. The last item can be plain text.
nested If/Or	{Or} can have another conditional as its final option, eg {Or}{[BookTitle],[Title],{If}{[XXX],aaa,bbb} }. The following is not valid: {Or}{[BookTitle],[Title]{If}{[XXX],aaa,bbb} }. {If} can have nested conditionals at either true/false option. eg. {If}{[numleafdocs],[Title],[BookTitle]{If}{[Subject],[Subject],unclassified} }