LAB 3:

GSDL: Advanced collection configuration

3.1 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./srcli
nk]</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:

   `[link][icon][/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.

3.2 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

8. (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.)

9. 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.*

10. **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.

3.3 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

   ```
   [srcicon]
   ```

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,[srcicon],[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 [srcicon], 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. For versions before 2.82, you need to change to **Library Systems Specialist** mode in order to 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. For versions before 2.82, this option is only visible in **Library Systems Specialist** and **Expert** modes.

For versions before 2.82, 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 `[srcicon]` to `{If}{[NoText] eq "1",[srcicon],[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. Add **SearchVList** 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. 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=&quo
t;_cgiargq_&quot;\">{Or}{[ex.thumbicon],[ex.srcicon]}</a>,
[ex.srclink]{Or}{[ex.thumbicon],[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.

3.4 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 2003, 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. Since versions 2.82, the Librarian Interface operates in three modes, while for versions before 2.82 GLI operates in four modes. Go to **File → Preferences... → Mode** and see the modes and what functionality they provide access to. **Librarian** is the default mode.

6. For versions before 2.82, 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
```

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. For versions before 2.82, 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** for Word 2003). 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.

****

