DICOM Terrier

System requirements

Operating System
Currently the application system has been tested only on Windows XP

For local use only:

Java 1.5	http://java.sun.com/j2se/1.5.0/

For web-interface use:

Java 1.5	http://java.sun.com/j2se/1.5.0/
Java Image I/O tools	http://java.sun.com/products/java-media/jai/current.html
J2EE Application Server	http://java.sun.com/j2ee/1.4/
dcm4che library	http://sourceforge.net/projects/dcm4che/

For development:

Java 1.5	http://java.sun.com/j2se/1.5.0/
Eclipse 3.0 or higher	http://www.eclipse.org/
ANTLR plug-in for Eclipse	http://antlreclipse.sourceforge.net/

(Any other development platform then Eclipse can be used to compile the source of course, but that is not described in this manual)

Terminology

%INSTALL% is the directory where package is extracted.

Compilation

The source consists of the following parts:

- Storage Tool
- Core Engine (this includes a local user-interface)
- DICOM XML parser
- Utility package
- Web-interface

Here the compilation with the Eclipse environment is described. We describe how to compile the core engine. First the source code of the project has to be loaded into the Eclipse environment. To do this we create a new project in Eclipse:

-Select “New project”
-Select “Java Project”
-Select “Create project from existing source”
-Choose the “%INSTALL%/src/dicomterrier” directory
-Click next
-On the “libraries” tab, edit the path names of the used libraries so that they point to the correct location, i.e. “%INSTALL%/lib”
-Finish

Now the project can be build. This can either be done manually by selecting “build project” or by selecting “build automatically” from the project properties.
When the project is build a jar-file can be generated. There is already a jar-file descriptor available in the project; this file is called “dicomterrier.jardesc”. This file can be adjusted so that the jar-file will be placed in the right location. Then right-click this file and select “generate jar-file”.

The above-described process can be applied to all parts of the source distribution.
For the web-interface however we have to perform extra steps, because the files are not packed into a jar-file, but into an ear-file that can be deployed. For this we use the deploytool that comes along with the Sun Application server. In this tool select File->Open. Select “%INSTALL%/webapplication/beta_version.ear” to open. Now we have replace the content of this package with the new content. This can be on three places:

- On the top level ‘beta-version’. Select ‘Add Library Jar’ and ‘Remove Library Jar’ to add new compiled libraries
- On ‘SearchWar’ select ‘Edit Contents’ to replace the class files
- On ‘SearchEngineBean’ select ‘Edit Contents’ to replace the class files.

Click save and the new ear-file is ready for deployment. How to deploy the application is described in the installation section. More information on deploying applications can be found in the extensive documentation on the Application Server.

Installation

The application can be used both locally as through a web-interface. For both types holds that they will look for the %DICOMTERRIER_HOME% variable to determine where the package can be found; it should be set to %INSTALL%. When this variable is not set it will assume the package to be placed in C:\dicomterrier.

Basically the local application now is ready for usage and can be started by “%INSTALL%/bin/desktop_dicomterrier.bat”. However there is not yet an index available for the application to work with. This will be described later.

Now we make the web-interface ready for usage. First the Java Image I/O tools must be installed correctly. Next, Sun’s Application Server must be installed. How to install the application server is described in the documentation that is distributed along with server. In addition to the installation described in that documentation we have to install the dcm4che library for displaying images.
This is done as follows:

- Copy the ‘dcm4che.jar’ to the ‘lib’ directory of the installation directory
- Adjust the file ‘asadmin.bat’ in the ‘bin’ directory of the installation directory. Add to the last line: "%AS_INSTALL%\lib\dcm4che.jar", to make sure that the library is loaded when the server starts up

Now start the server and verify that is functioning correctly.
Next we have to deploy the application. This can be done either with the ‘deploytool’ of the Application Server or through the web-interface of the application server.
With the deploytool:

- Start the deploytool
- Goto File -> Open and select ‘%INSTALL%/webapplication/beta_version.ear’
- Click “Deploy”
- Login with the account for the application server.

Now also the web-application is ready for usage, but still we have no index available. How to create an index is described in the section about usage.

Usage

Storage
The current storage tool is developed to operate in combination with DVT. So DVT must be installed on the system in order the storage tool to function properly. Simply start the tool by clicking “%INSTALL%/tools/StorageTool-vX.X/StorageTool-vX.X.bat”.
Two properties of the tool can be configured:
- Destination directory of the upload
- Path to DvtCmd.exe

After the information about the image(s) is entered correctly the submit button can be clicked. The upload process starts. The progress bar in the bottom gives an indication of the progress of the upload, but because DVT doesn’t have the possibility to show it’s progress, during the validation of the image(s) the progress bar will not move.
After the upload finished, or in case an error occurred, the log file can be viewed.

Retrieval
Webinterface:
This interface consists of two parts: the simple screen and the advanced screen. Both screen have the same purpose, users can enter their queries. Only the advanced screen has more input fields that will help the user to structural build up a complex query.
The time it takes to execute a query is in the order of seconds. However when a query is still taking a long time, it is probably due to one of two causes:
- The searched keywords match with a large number of documents
- The range that is used in a query is very large

Picture Viewing:
Retrieved images can be viewed. By clicking preview, a thumbnail image of the selected image is shown. During the time the image is loaded, the text “Rendering… One moment please” is shown. When it is not possible to show the image, “No preview available” is shown. This can be because of the fact that the retrieved DICOM object simply doesn’t contain pixel information.
It can be useful to know a little bit of the process of showing DICOM images. The selected DICOM object is opened for reading and then the pixel information is streamed as JPEG image to the browser. When the image is very large, this can take of course a considerable amount of time. This is not the case when the DICOM object contains a sequence of images, then only one of the images from the sequence is shown. But when an image in itself is very large, this can take more time.

Local GUI:
This interface contains much of the retrieval functionality of the web-interface, however it misses some options. It is not possible to show previews of the images. Instead a DICOM viewer installed on the system can be used to view the images. Further during retrieval it is not possible to group results for equal datasets. Finally the possibility in the web-interface for showing additional information is here also not available.

Indexing
Next we explain how to create the index. In the “%INSTALL%/tools” the utility “index.bat” can be found. This utility must be run with the directory that is used as destination directory for the storage tool. This utility will successively:
- Shutdown the application server
- List all meta and representation directories
- Run indexing
- Startup the application server

After this process the index is available in %INSTALL%/var/index. Also, if no errors occurred, the web-interface can be used to query the indexed collection. The indexing step can also be called manually by calling "%INSTALL%/bin/desktop_dicomterrier.bat –runindex". This can be useful when the application is run only local, without the application server.

Admin Properties

In the directory %INSTALL%/etc the file "terrier.properties" can be found. In this file the configurable properties of the application are defined. We describe the different properties here. For all property changes holds that the application must be restarted in order for the changes to take effect.

Path properties
The path properties describe where all the files and directories can be found. If these are not set, the default location will be used.

`dicom.viewer`
This property is used when the application is run locally. This application has no functionality of showing (previews of the) images. With this property the path to a DICOM viewer can be set. This viewer will then be run to show the image.

Index properties

`indexing.simplefilecollection.extensionsparsers`
This property is the property that defines which parser (=class) to use for the XML files. Normally this will be DICOMXMLReader but any class that implements the class FileDocument can be used.

`index.properties.file`
This property specifies the name of the index properties file. In this file the parse specific properties for the DICOM XML parser are defined. These are the properties for the current parser:

> `index.process.skip`
> This boolean property is used to specify whether to skip the defined tags or to only process the defined tags.
>
> `index.skippable`
> This property specifies which tags to skip from the XML file while parsing. These tag names will not be taken into account in the structure context of the terms. The terms itself will be processed. For the DVT-generated files this can be for example "Values" and "Value". These tag do not really contain structure information, and would only take up unnecessary space.
>
> `index.processable`
> This property specifies which tags only should be stored in the tag structure of terms. The rest is skipped.
>
> `index.text.skippable`
> This property specifies of which tags the content can be skipped. For the DVT-generated files, this will be for example PixelData, because this contains only absolute path info.
>
> `index.map.tags`
> This property specifies which tag names of an XML-files should be renamed to (possibly multiple) other name(s).
>
> `index.map.targets`

This property specifies the attributes that should be used to create new names for the tags specified in the variable above. The attributes must be ordered corresponding the way the tag names are listed.

`index.map.default.targets`
This property specifies which attributes of an XML tag should be used by default to construct extra tag names.

`invertedfile.processterms`
`invertedfile.docfrequency`
These are two properties that specify how large the blocks are that used in each turn of inverting the index. When there are memory problems these numbers should be decreased, when creating the inverted index should be sped up, then these numbers should be increased.

Weighting properties:

`meta.weight`
This property defines how heavy the information found in the meta description weights in the total score. When this value is increased, the meta information will play a more important role.

`tag.level.weight`
This property specifies the multiplier with which a score of a term is multiplied for each level in the structure between the occurrence of the term and direct containment.

`basic.boolean.tag.weight`
This property specifies the weight that is assigned to boolean values that match. This is the weight they get before the multiplier is applied for the level in which the value occurs (from the previous property).

`context.tags`
This property specifies which tags for which the score is increased for terms that occur in these terms. This information will only be used when the query consists of keywords only, because when specific structure constraints are used, it is not applicable.

`context.multiplier`
This property specifies with which value a weight should be multiplied when in appears in one of the context tags specified in the property above.

Web-interface properties

`results.numberperpage`
This property specifies the number of results that should be showed on one page in the web-interface.

`display.file.info`
This boolean property tells whether additional information of the images should be showed in the web-interface.

`file.info.class`
This property specifies the class that is used to read the file information to show from the XML representation of the file.

`display.meta.info`
This boolean property tells whether the meta information of the datasets should be showed in the web-interface.

`meta.info.class`
This property specifies the class that is used to read the meta information from the meta file.

`webcontext.postfix`
This property can be used to specify the context-name of the shared drive that is used. In our case this is for example .best.ms.philips.com .