1 Setup and installation

This chapter gives a quickstart description of how to setup the software, possibly in a way so to make it work together with existing document sets and workflows. Some internals are described in more detail as to give a starting point for further integration extension of the software.

1.1 A use case

Default directories. Some paths can be configured (config.ini).

- For a PDF file in the local filesystem, a file describing table structure in "ICDAR format" [1] exists.
- The user wants to see the table structure as an overlay to the original document and possibly make changes to the table structure.
- `http://localhost:8080/annotate?file=var/pdf/test.pdf` look for a PDF file at `/var/pdf/test.pdf` and show it in the application (and it will be added to the internal document index, etc.).
- A file named `test-str.xml` must either be present in the source folder (/var/pdf) or in `resources/repos/external` (relative to the application path) .
- Choosing the "exttable" layer to be visible will show the annotated tables from `test-str.xml` .
- With a double click, the table can be edited. After it has been save, it will now be part of the local table layer. (These tables are on a different layer, "tables" which is visible by default).
- After tables have been edited and saved, the "Save" button next to the "Exports" button will save the edited export to `resources/repos/external/exports/test-str.xml` . (*All tables from the original annotation file must be "imported" to the local layer by double click and save, otherwise the will not be present in the newly generated export*) .

[1] http://www.tamirhassan.com/competition/dataset-format.html

1.2 Architecture and means of operation

The application can work in two slightly different modes - (i) working only with files below a specific directory that have been previously "imported"; or by specifiying a local path as a GET parameter to `/annotate` (which also does an import of the file "on the fly" if, it is not in the repositry yet) .

Usage with repository PDF files an and data files representing the annotations are kept below the directory `resources/repos`.

This is the location for subdirectories, so called "repositories". Each repository (i.e., subdirectory) has an index file, the file "documents.json". The file stores an integer id to each PDF source document, through wich they are referenced from then on.

In each repository directory, PDF source files are located in the "source" directory, annotation files are kept inside the "data" directory, in a directory that is the same as the document's id.

```
resources/repos
  /[name]
    /src
      /file1.pdf
      /file2.pdf
      /documents.json
    /data
      /1/tables_v2.json
      /2/tables_v2.json
```

Figure 1: Layout for a repository with two files in it.

Files can be imported into the repository either by upload or by the import script.

Usage with external paths Even when using file paths as get parameter, the file will be added to the index file "documents.json" , located in the repository named "external" (resources/repos/external). This directory has a subdirectory "data", but no source directory, as the source file stay in their original place. For this case too, a "basedir" can be specified, so that the subtree from which documents can be accessed can be restricted.

The table structure export in ICDAR format can also be conveniently saved to disk with the "Save" button next to the "Exports" button.

```
resources/repos
  /external
    /documents.json
    /1/tables_v2.json
    /2/tables_v2.json
```

Figure 2: Using "external" source PDFs, the repository will look like this.

The location will be `resources/repos/[reponame]/data/exports` .

External applications could access and even modify these datafiles. See below for (re-)importing data from external sources.

Import annotation data from external sources The application will look for files that follow the pattern "[FILENAME]-str.xml" in the source-directory and in the root of the external repository (resources/repos/external).

This table data can be accessed (viewed and imported) by activating the "Ext.table" layer in the frontend.

1.3 Begin working on documents

With files imported into the repository `http://localhost:8080/`

Per default, the application will use the "test" repository at `resources/repos/test`. The location will be empty at first. Use the upload servlet at `http://localhost:8080/repo` or the import batch-script to add files.

With files anywhere in the local filesystem `http://localhost:8080/annotate?file=path/to/file01.pdf`

Accessing the application like this will import the specified file and the user can begin workng on it.

Display annotation/table definitions from external sources External data can be opened with the file, if it is placed in the `resources/repos/external` directory - or in the same directory as the source PDF file.

For external data to be recognized, a naming convention must be met; '-str.xml' is appended to the original filename: For a file that is named `file01.pdf`, the filename must be `file01-str.xml` or `file01.pdf-str.xml` .

View: To view that data, activate the layer "exttable".

1.4 Data exchange formats

1.4.1 Internal Json format

```
{"pages":
  {"0":
    [
      { "page":0,
        "tn":31,
        "trs":[
          [
            {"visgrid":[448.67,280.0,514.0,291.33],
            "classes":["header"],
            "rowspan":1,
            "colspan":2,
            "startrow":0,
            "startcol":0}
          ],
          [
            {"visgrid":[448.67,292.0,483.33,310.67],
            "classes":[null],
            "rowspan":1,
            "colspan":1,
            "startrow":1,
            "startcol":0},

            {"visgrid":[484.0,292.0,514.0,310.67],
            ...
            "startrow":1,
            "startcol":1},
            }
          ]
        ]
      }
    ]}}
```

Figure 3: Example of the internal annotation data in JSON, as it is stored to disk (tables_v2.json).

Internally, table annotations are stored as JSON files. The field "pages" holds is an indexed array (page number) holding a list of tabl objects, which consist of the page number, an id ("tn") and a two-dimensional array (i.e. a list of lists) ("trs") representing the table cells. Within the cell object, its area is stored under the "visgrid" member, as a list of float values, in

```
1
2   // Set up a table with one row and two cells
3   DocTable table = new DocTable();
4   table.trs.add(new ArrayList<DefTd>());
5   table.trs.get(0).add(new DefTd(40,40,100,50));
6   table.trs.get(0).add(new DefTd(120,40,200,50));
7
8   // Add table on page two
9   TableDefinitions tables = new TableDefinitions();
10  tables.addTable(2, table);
11
12  // Save to disk
13  File = new File("test.json");
14  RepositoryAccess.saveTableDefinitions(file, tdefs);
15
16  // Load from disk
17  tdefs = RepositoryAccess.loadTableDefinitions(file);
```

Figure 4: Example of how to use the API to write and read JSON files.

the order left, top, right, bottom. The values are relative to the page 0, 0 coordinates. The y value of the pdf is considered to be 0 at the top, and "page height" at the bottom.

1.4.2 XML export of structure and region data

For specification , see [2] .

1.4.3 HTML

The tables can also be exported in HTML. This export can be viewed when choosing the "Table contents" button, and downloaded afterwards. There are data-attributes added to certain tags: These are the bounding box of the whole table region and the pagenumber to the `<table>`-tag; for the `<td>`-tag, it is also the bounding box of the characters in the cell, as well as the rectangle that was used to define this cell. The `rowspan`- and `colspan`-attributes of Html tables are also used.

1.5 Command line usage and batch functionality

Certain functions are best performed via the command line script.

[2] http://www.tamirhassan.com/competition/dataset-format.html

1.5.1 Add documents do repository in bulk

Documents can be imported into the repository in bulk.

An invocation like the following would import a directory into `resources/repos/repository_name` . That directory has to exist beforehand.

```
$ mvn exec:java -Dexec.args="import /path/to/pdfs/
    repository_name"
```

1.5.2 Inspect repository / working-set contents

Lists all documents and the number of annotated tables for each document.

```
$ mvn exec:java -Dexec.args="info <repo> <workingset> <
    export_type>"
```

1.5.3 Create external representations of annotations (exports)

Export tables of a single file:

```
$ mvn exec:java -Dexec.args="export <repo> <workingset> <
    docid> <export_type>"
```

Or for complete workingsets.

```
$ mvn exec:java -Dexec.args="exportall <repo> <workingset> <
    export_type>"
```

Export type can be one of 'html', 'csv', 'structure', 'region', or 'functional'.