HighQSoft

AoDoc GUI

Version 1.4

User Manual

Martin Kreyling 08.03.2005

Contents

Chapter 1

Introduction

1.1 Introduction of AoDoc

AoDoc is a tool for reading, checking, completing, correcting and documenting Asam Transport Format (ATF) files. AoDoc can read both, ATF/XML and ATF/Classic files. It is designed to find errors in ATF files via so-called proofers. AoDoc already contains some proofers, e.g. for required attributes or for relation checking. It can be enhanced by any kind of proofer.

AoDoc can be used in a batch mode as well as in an interactive mode (aodocgui). This utility has not yet been integrated as a plugin into the AsamCommander.

A given file is compared against a specified ASAM ODS Base Model (ao_baseXXX.htm) and automatically completed and corrected into a separately specified output file. The supported ASAM ODS Base Model file is defined in an extended HTML file containing some special tags to allow the coding of the ASAM ODS rules. This file is available in different versions which reflect the versions of the ASAM ODS standard (e.g. 5.1 = athos/etc/ao_base51.htm). This file is maintained by HighQSoft according to the latest results of the ASAM ODS standard decisions and it is used as the backbone base model definition for the Athos software.

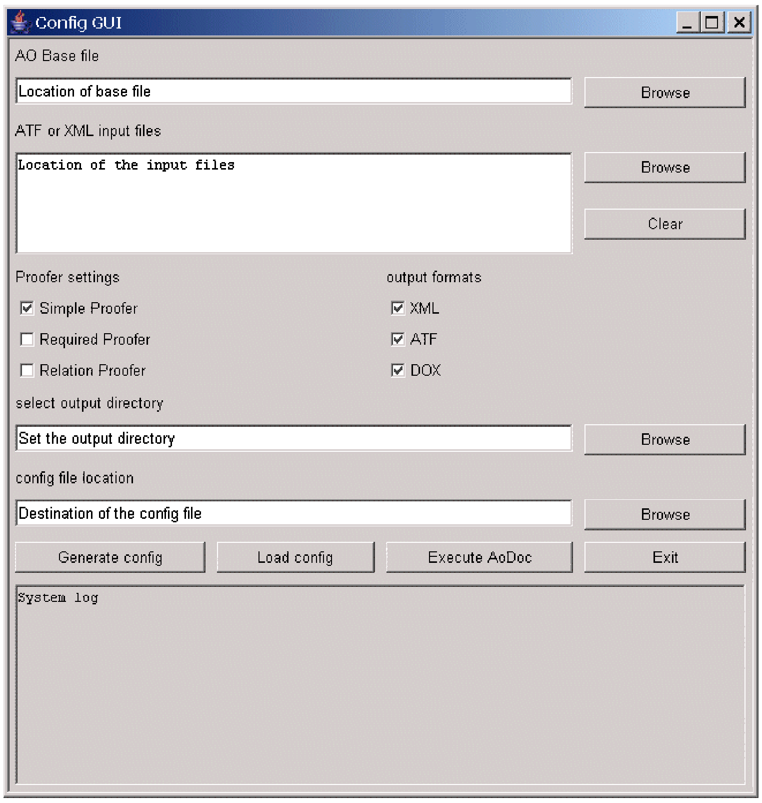
The standard ASAM ODS Base Model Definition in Step-Express notation and the ASAM ODS XML Base Schema file are not yet evaluated by AoDoc.

Output files can be ATF/XML, ATF/Classic and Doxygen documentation files to be used in a further typesetting process.

There are three different checker modules, so-called 'proofers', to verify if an ASAM ODS Application Model contains all the required information. The ASAM ODS Application Model is automatically extended with required information from the ASAM ODS Base Model. The result model can be stored in a specified ATF/XML, ATF/Classic od Doxygen file.

1.2 Introduction to AoDocGui

AoDoc GUI is a graphical user interface, which should simplify the usage of the AoDoc program. It allows to generate a config file for the AoDoc and/or it executes the AoDoc program.



Location of base file

Location of the input files

[✓] Simple Proofer
[ ] Required Proofer
[ ] Relation Proofer

[✓] XML
[✓] ATF
[✓] DOX

Set the output directory

Destination of the config file

System log

Figure 1.1: An overview on the progam

Chapter 2

The AoDoc

2.1 Using Aodoc

Aodoc can be installed to any directory. It is recommend to add the '../aodoc/' directory to the system-path to make the usage more comfortable.

2.1.1 The config file

The config file contains all input data for the Aodoc tool. This is the point where you choose your input atf/xml files and other required information. An example of the default config.dat looks like this (without line numbers):

```
(01) ATF_XML_FILE = c:/tools/aodoc/input_files/base.xml
(02) ATF_XML_FILE = c:/tools/aodoc/input_files/demo.atf
(03) ATF_XML_FILE = c:/tools/aodoc/input_files/combustionmodel.atf
(04)
(05) AO_BASE_FILE = C:/tools/aodoc/src/java/com/highqsoft/aodoc/ao_base50_ralf.htm
(06)
(07) //possible proofer are: 'SIMPLE_PROOFER', 'REQUIRED_PROOFER' and 'RELATION_PROOFER'
(08) PROOFER      = REQUIRED_PROOFER, RELATION_PROOFER
(09)
(10) //possible outputs are: xmlout, atfout, doxout
(11) OUTPUT       = xmlout, atfout, doxout
(12)
(13) OUTPUT_DIR   = c:/tools/aodoc/output/
```

All input files have to be declared with 'ATF_XML_FILE = ..' (line 1-3). The user may choose multiple files in different directories. The path name has to be absolut.

Line (5) declares where Aodoc takes the AoBase.html from. Please note that not every AoBase.htm will work with this programm. A working version is delivered within the source directory of Aodoc. The difference to former base files is, that a new tag

```
ao_description
```

was added to parse the description of the elements more easily. If using another version, all descriptions will be missing.

Line (8) declares which proofer Aodoc will use to merge the AoBase file with the atf/xml file. The three choices are: 'SIMPLE_PROOFER', 'REQUIRED_PROOFER' and 'RELATION_-PROOFER'. Each proofer may be used as a stand-alone, but can also be used in combination with the other two proofers. The proofer need to be seperated with a ', '. The order of execution will always be: 'SIMPLE_PROOFER' -> 'REQUIRED_PROOFER' -> 'RELATION_PROOFER'.

In line (11) the output mode is selected. Allowed values are 'xmlout', 'atfout' and 'doxout'. All modes can be in combination with each other, but have to be seperated with ', '.

Line (12) declares the output directory, where all outputfiles will be saved.

Note: A line can be commented with '//' and with '\'.

2.1.2 What would Aodoc produce with the example Config.dat?

Aodoc parses the three input files and compares them with the given AoBase file using the selected proofers. Since all three output modes are selected the output directory will contain nine files:

- base.atf
- base.dox
- base.xml
- demo.atf
- demo.dox
- demo.xml
- combustionmodel.atf
- combustionmodel.dox
- combustionmodel.xml

As you can see the filename from the input file will be taken over.

2.1.3 Starting Aodoc

Aodoc can be started with the Aodoc.bat file. It simply uses the Config.dat which is in the same directory.

Since you might have different config files for different projects in different directories, just start the console, go to the desired directory and type in 'aodoc'. Aodoc will now take the 'Config.dat' out of the actual diretory. The aodoc directory must therefor be set in the system path. The config file is of course allowed to have a different name and also the path might be absolut. This will also work:

The position of the log file can be changed with a second argument.

```
aodoc example.dat
aodoc c:/projekt/projekt_a/test.dat
aodoc temp/test.dat
```

The console also allowes 'backslash' instead of 'slash'.

2.2 The Proofer

As explained above there are three different proofers which can be used. The choise is between 'SIMPLE_PROOFER', 'REQUIRED_PROOFER' and 'RELATION_PROOFER'. Every possible combination of the proofers is possible.

The **SIMPLE_PROOFER** is more likely a merger of the atf/xml and the AoBase file. It compares all application elements and application attributes found in the atf/xml file with the matching element/attribute of the AoBase file. All the attributes of the matching AoElement/AoAttribute will be copied the the atf/xml file element/attribute if they are missing. As a result the user will have the original atf/xml extended with all available information from the AoBase file. As an example: If an application element has no description, it will be extended with the one from the AoBase file. If a description is avialable, nothing will be done.

- checks if application/relation attribute has a description

The **REQUIRED_PROOFER** checks the atf/xml file for all required attributes and relations:

- if a required attribute is missing, it will be added with a default name, which has to be changed by user.
- checks if required 'or' relations are available only once. If such a relations does not occur, or occurs multiple times, the user has to make the change
- if a required 'non-or' relation is missing, Aodoc adds it with default 'name' and default 'ref-to'.
- all attributes and relations are checked for double occurances

The **RELATION_PROOFER** proofs all the relation targets:

- checks if base relation is available (not required)
- checks if target application element is available (name and basetype need to be correct)

No matter which combination of the proofers a user chooses. It will always be called the way they are ordered above. This can be changed in the Aodoc.java itself, or Aodoc can be executed multiple times with only one proofer.

Note: It is also possible to choose no proofer. This is useful, if the user just wants to convert atf/xml files to another file. It is possible to create the doxygen out but from atf/xml and it is prossible to convert a complete xml file to atf. Converting atf to xml also works, but the user has to add some tags to the xml header the base model version. This information is not required in a atf file, and therefore has to be added manually.

To convert files it is recommened to use the SIMPLE_PROOFER. In most of the cases just the description for each element/attribute is added.

2.3 The logging file

The logging output of aodoc will normaly be saved in a temp dir'/AodocLogging.htm'. If an URL is given by the second argument, the file only will be stored in the given URL.

The logging file contains all warnings of the proofers when they detect missing object or a wrong relation in the atf/xml file. All these occurances are registered with some extra information so that it becomes quite easy for the user to correct the atf/xml file. For example AoDoc adds missing relations, but they still have to be renamed by the user (target element name). Since there is no GUI yet, this is the only way to do it.

The displayed warnings do not necessarily cause the ASAM Commander to display a wrong tree if you load the original atf/xml file. If there are no red warnings you atf/xml file is complete. The warnings displayed green just inform the user that the specified relation has no base relation which is totally conform with the base model. More such (green) warnings may be added, but there is no need to work on it.

The SIMPLE_PROOFER throws purple info-warnings, when there is no description for an application/relation attribute. These attributes wheter are not part of the base model, or the base model does not contain a description for the object.

There are some other information included to the logfile which are most likely self-explaining.

2.4 AoDoc.java

This section will describe a few things the user can change in the AoDoc.java to test single steps of the programm.

First of all the user should roughly understand how the programm works:

The whole AoDoc consist of three main procedure. First of all the input files are parsed with the specified parses. The parser aoutamatically build a DOM out of the file. No matter what file it was, they all look the same in the DOM. Then the atf/xml DOM is taken to be compared with the AoBase DOM with the rules of one of the proofers. The third task is to print the output via one of the printers.

2.4.1 The logger

The logger can be switched off or be set to another level.

```
logger.setLevel(Level.FINEST);
```

To turn the logger of select 'Level.OFF'. To just printout the warning lines it can also be changed to 'Level.WARNING'. For further explaination, see the java documentation.

2.5 How to comment atf/xml files correctly.

The ATF Classic file syntax allows block comments of the type

```
/* ... */
```

and single line comments of the type

```
// ...
```

.

Those comments may appear anywhere within the code, but they are ignored in string definitions of the type

```
" ... "
```

The AoDoc parser adds a few additional rules which allow to distinguish certain types of comments without the need to mark those comments with special tags. The rules have been defined in a way that most files can be processed without major modifications.

Rule 1: AoDoc ignores the block comments within slash-star comment signs completely. Slash-star block comments do not appear in the documentation of an ATF file.

Rule 2: Single line comments starting with

```
//
```

are used as valid ATF file documentation for further processing.

Rule 3: Continuous single line comments are treated as a block comment unless they appear behind APPLELEM or APPLATTR keywords on the same line. This means, as long as no empty line or a line without a comment is detected, each additional single line comment is added to the block comment.

2.5.1 The ATF comments.

Comments are interpreted by their position. AoDoc recognizes the following patterns. Each comment block may have as many lines as desired. For simplicity each block is shown here in a few lines per block only.

```
// Document title
// If there are only two or less comment blocks available, this comment block is assumed to be missing.

// Document Synopsis
// This is a short desription or abstract of the document.
// If there are only three or less comment blocks available, this comment block is assumed to be missing.

// Document Description
// This is the full and probably lengthy description of the document.

// Application Element Synopsis
// This is a short desription or abstract of the following application element.
// If there are only four or less comment blocks available, this comment block is assumed to be missing.

// Application Element Description
// This is the full and probably lengthy description of the following application element.
APPLELEM Environment, BASETYPE AoEnvironment
   APPLATTR Id, BASEATTR id, DATATYPE DT_LONGLONG;                 // The Id comment (optional).
   APPLATTR Name, BASEATTR name, DATATYPE DT_STRING;               // The Name comment (optional).
   APPLATTR Description, BASEATTR description, DATATYPE DT_STRING; // The Description comment (optional).
   APPLATTR myAttr, DATATYPE DT_STRING;                            // And so on...
ENDAPPLELEM

// Application Element Synopsis
// This is a short desription or abstract of the following application element.

// Application Element Description
// This is the full and probably lengthy description of the following application element.
APPLELEM Test, BASETYPE AoTest
   APPLATTR Id, BASEATTR id, DATATYPE DT_LONGLONG;                 // The Id comment (optional).
   APPLATTR Name, BASEATTR name, DATATYPE DT_STRING;               // The Name comment (optional).
   APPLATTR Description, BASEATTR description, DATATYPE DT_STRING; // The Description comment (optional).
   APPLATTR myAttr, DATATYPE DT_STRING;                            // And so on...
ENDAPPLELEM


/*
```

```
 Created Thu Jun 30 11:31:26 2005
 This is some atf-file description or history information that should not
 show up in the final documentation.
*/

// This is the general description of the ATF file which
// is used in the documenation. The comment signs are removed
// for documentation purposes
//
// Empty lines as the line above this one are used to introduce a new
// paragraph in the documentation
//
// There may be any number of paragraphs as long as the single line
// comment sign is available on every line.

// By the way, this line overwrites the single line comment block from
// above, because we started a new one. This is only allowed for application
// element descriptions..................................

ATF_FILE V1.4;

// APPLE Kommentar1
// APPLE Kommentar2
/* APPLE Kommentar3 */
APPLELEM Environment, BASETYPE AoEnvironment // APPLE Kommentar4
   APPLATTR EID, BASEATTR id, DATATYPE DT_LONGLONG;
// APPLA Kommentar1
// APPLA Kommentar2
// APPLA Kommentar3
   APPLATTR Description, BASEATTR description, DATATYPE DT_STRING; // APPLA Kommentar4
   APPLATTR Name, BASEATTR name, DATATYPE DT_STRING; // APPLA Kommentar4
ENDAPPLELEM;
```

It is allowed to write comments on top of every file, element and attribute. It is also allowed to write the comments behind an element or attribute until the end of line (10, 15). Comment chars are

```
/* ... */
```

and '

```
//
```

. It does not work using

slash star comments in itselfs

```
/* ... /* ... */ .. */
```

. The following output will be (using no proofer or SIMPLE_PROOFER):

```
(01) /*
(02)   Created Tue Mar 27 11:31:26 2001
(03)   this is the atf-file description
(04) */
(05) ATF_FILE V1.4;
(06)
(07) /*
(08)   APPLE Kommentar1
(09)    APPLE Kommentar2
(10)    APPLE Kommentar3
(11)    APPLE Kommentar4
```

```
*/
APPLELEM Environment, BASETYPE AoEnvironment
   APPLATTR EID, BASEATTR id, DATATYPE DT_LONGLONG;
   /*
     APPLA Kommentar1
      APPLA Kommentar2
      APPLA Kommentar3
      APPLA Kommentar4
   */
   APPLATTR Description, BASEATTR description, DATATYPE DT_STRING;
   /*  APPLA Kommentar4 */
   APPLATTR Name, BASEATTR name, DATATYPE DT_STRING;
ENDAPPLELEM;
ATF_END;
```

2.5.2 The XML Comments

There is onyl ony way to comment xml files. The comment has to be above the start-tag of the element/attribute. An example file is listed here:

```
<!--
  Created Tue Mar 27 11:31:26 2001
  by user karst on host ODIN with program ASCOBA.
-->
<atfx_file version="atfx_file V1.4">

<!--
Locale of Document
-->
   <locale>de_DE
   </locale>

<!--
 Based on ODS Base Model version
-->
   <base_model_version>27
   </base_model_version>

<!--
 declare application model meta data
-->
   <application_model>

<!--
  APPLE Kommentar1
  APPLE Kommentar2
-->
      <application_element>
         <name>Environment</name>
         <basetype>AoEnvironment</basetype>
<!--
 Unique ID for the instances of an application element
-->
         <application_attribute>
            <name>EID</name>
           <base_attribute>id</base_attribute>
            <datatype>DT_LONGLONG</datatype>
         </application_attribute>
      </application_element>
```

This example will not be discussed, because it should be clear how it works.

Chapter 3

Using the GUI

3.1 Configure the settings

3.1.1 The ASAM ODS Base file

First of all the absolut path of the AO Base should be set. There for it can be entered into the textfield or it can be choosen by using the browser button and the filechooser. Here only one file will be given to the program. If you don't give a location of an AoBase the AoDoc a standart location. Which is "ATHOS_ROOT".htm, either give the path of a AoBase or make sure that this one is existing. This is the standart of AoDoc.

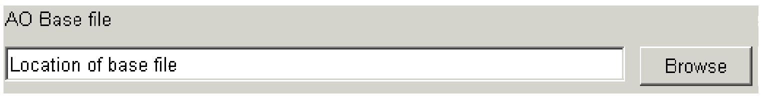


Figure 3.1: Select the location of an AoBas

3.1.2 The ATF or XML input files

Now the files, which have to be checked, must be selected. Here it is only posible to use the filechooser which will open after pressing the button. In this filechooser it is posible to selecte multiple files by using the "Shift" or the "Ctrg" button. The "Clear" button resets the textarea.

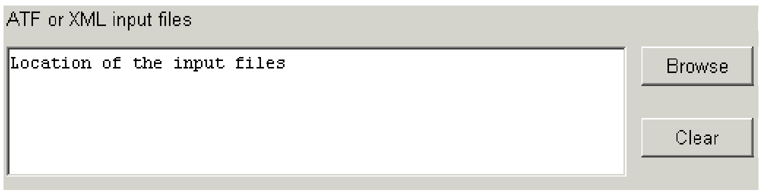


Figure 3.2: Select the locations of the input files

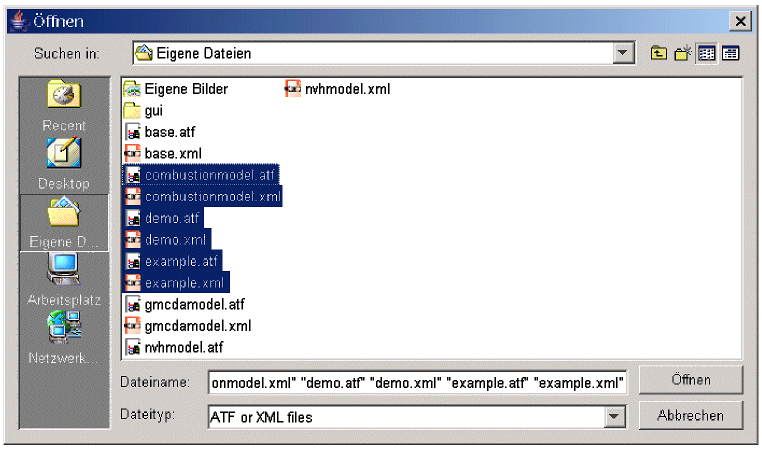
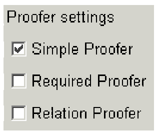


Figure 3.3: Multiple files selected

3.1.3 The Proofer Option

Now the Proofer will be selected. The Proofers are described in the AoDoc documentation. For more information about **the Proofers** (p. 5)


Proofer settings

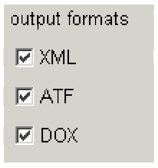
[✓] Simple Proofer
[ ] Required Proofer
[ ] Relation Proofer

Figure 3.4: Un/check the proofer you need or not

All combinations are posible. If no Proofer is selected the input file will only be converted, but the SIMPLE PROOFER is a better way to convert the files.

3.1.4 Output format

The posible output formats are XML, ATF and DOX. The user can combine the different output formats as he likes, but there must be at least one format selected.


output formats

[✓] XML
[✓] ATF
[✓] DOX

Figure 3.5: Un/check the format you want

3.1.5 Output directory

Here the user has to select where the files which will be created during the AoDoc is running will be stored. The destination can be entered or be selected through the filechooser. The filechooser only accepts directorys!

3.1.6 Config file destination

Here the user has to give the destination of the config file. The destination is used for three things:

- for the creation of the config file
- for loading a config file
- for the execution of the AoDoc program

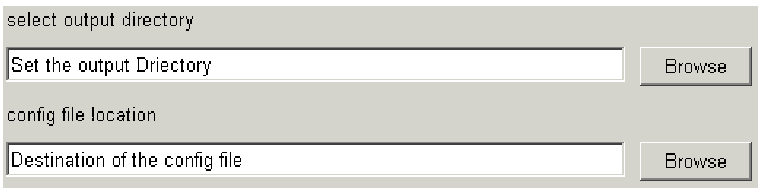


Figure 3.6: The output and config settings

3.2 The functions of AoDocGUI

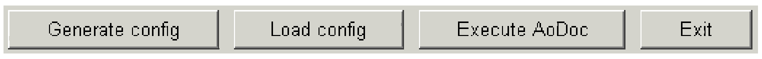


Figure 3.7: The buttons which make the most important actions

3.2.1 Generate config button

If the "Generate config" button is pressed, some essental conditions are checked, befor the config file is writen :

- is there at least one input file
- is the destination of the config file set
- is an AoBase path given
- is there at least one output format chosen

If all conditions are fulfilled, a procedure produces the config file and it creates it if necessary. The apprence of the config file itself is described in the AoDoc documentation.

3.2.2 Load config dutton

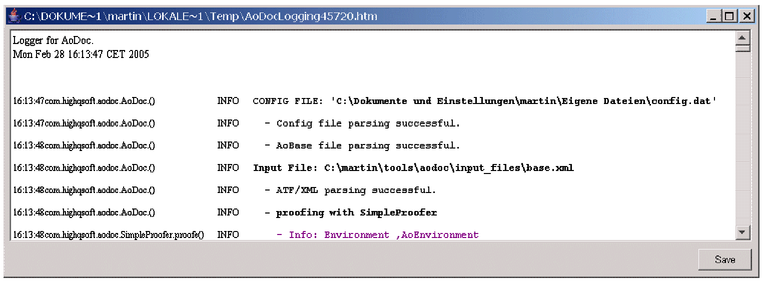
If the "Load config" button is pressed, the destination of the config file is taken and the file is loaded. Then the option spezified in there will be read and displayed on the GUI.

3.2.3 Execute AoDoc button

If the "Execute AoDoc" button is pressed, the destination of the config file is taken and starts the AoDoc program as a thread, so you can work on. When the AoDoc finishes a new Window comes up and the result is shown in an HTML file which can be saved.

IMPORTANT:

The GUI only gives the location of the config file to the AoDoc not the values which are selected on the GUI itself! That means first make a config file or get the location of an exsiting config file, and the execute!



```
Logger for AoDoc.
Mon Feb 28 16:13:47 CET 2005


16:13:47com.highqsoft.aodoc.AoDoc.()            INFO   CONFIG FILE: 'C:\Dokumente und Einstellungen\martin\Eigene Dateien\config.dat'
16:13:47com.highqsoft.aodoc.AoDoc.()            INFO     - Config file parsing successful.
16:13:48com.highqsoft.aodoc.AoDoc.()            INFO     - AoBase file parsing successful.
16:13:48com.highqsoft.aodoc.AoDoc.()            INFO   Input File: C:\martin\tools\aodoc\input_files\base.xml
16:13:48com.highqsoft.aodoc.AoDoc.()            INFO     - ATF/XML parsing successful.
16:13:48com.highqsoft.aodoc.AoDoc.()            INFO     - proofing with SimpleProofer
16:13:48com.highqsoft.aodoc.SimpleProofer.proofe() INFO      - Info: Environment ,AoEnvironment
```

Figure 3.8: Here is result shown

Note: The file shown is only a temp file it will be deleted after closing the window.

3.2.4 Exit button

This button stops the hole program. The thread for of the execution of AoDoc will be stop, too. If the AoDoc is running don't close the main frame or every thing will be stop. That can lead to problems because of some file operations that are running.

3.3 The system output

The normal console output is all shown in the textarea instead of in the console.

```
System log
```

Figure 3.9: The system output

3.4 Starting the AoDocGUI.java

The AoDocGUI is the main program. The GUI doesn't take any paramenters, when called.

The AoDocGui needs:

- CONFIG_FILE_Maker.java
- HTMLFilter.java
- ATF_XMLFilter.java
- EXEThread.java
- LOGStream.java

Chapter 4

Document Historie

Date	**Description**	**Author**
March 1st 2005	basic version	Martin Kreyling
March 2st 2005	AoDoc and AoDocGUI manual joint	Martin Kreyling