Command line

You can run PlantUML using the command line. (See running for ways to run PlantUML from various other

The most basic way to run it is:

```
java -jar plantuml.jar file1 file2 file3
```

This will look for `@startXYZ` into `file1`, `file2` and `file3`. For each diagram, a `.png` file will be cre

For processing a whole directory, you can use:

```
java -jar plantuml.jar "c:/directory1" "c:/directory2"
```

This command will search for `@startXYZ` and `@endXYZ` into `.txt`, `.tex`, `.java`, `.htm`, `.html`, `.c`
`.hh` or `.md` files of the `c:/directory1` and `c:/directory2` directories.

Docker images are released as Github Packages and to Docker Hub.

```
docker run ghcr.io/plantuml/plantuml
```

Wildcards

You can also use wildcards :

- For a single character, use `?`
- For zero or more characters, use `*`
- For zero or more characters, (including `/` or `\`), use a double `**`

So to process any `.cpp` files in all directories starting by *dummy* :

```
java -jar plantuml.jar "dummy*/*.cpp"
```

And to process any `.cpp` files in all directories starting by *dummy*, and theirs subdirectories :

```
java -jar plantuml.jar "dummy*/**.cpp"
```

Excluded files

You can exlude some files from the process using the `-x` option:

```
java -jar plantuml.jar -x "**/common/**" -x "**/test/Test*" "dummy*/**/*.cpp"
```

Output Directory

You can specify an output directory for all images using the `-o` switch:

```
java -jar plantuml.jar -o "c:/outputPng" "c:/directory2"
```

If you recurse into several directory, there is a slight difference if you provide an absolute or a relative path fo

- An absolute path will ensure that all images are output to a single, specific, directory.
- If you provide a relative path then the images is placed in that directory relative to the location of the **in** even if the path begins with a `.`). When Plantuml processes files from multiple directores then the cor computed output directory.

Types of Output File

Images for your diagrams can be exported in a variety of different formats. By default the format will be a PN following extensions:

Param name	Short param name	Output format	Comment
`-tpng`	`-png`	PNG	Default
`-tsvg`	`-svg`	SVG	Further details can be found here
`-teps`	`-eps`	EPS	Further details can be found here
`-teps:text`	`-eps:text`	EPS	This option keeps text as text
`-tpdf`	`-pdf`	PDF	Further details can be found here
`-tvdx`	`-vdx`	VDX	Microsoft Visio Document
`-txmi`	`-xmi`	XMI	Further details can be found here
`-tscxml`	`-scxml`	SCXML	
`-thtml`	`-html`	HTML	Alpha feature: do not use
`-ttxt`	`-txt`	ATXT	ASCII art. Further details can be found
`-tutxt`	`-utxt`	UTXT	ASCII art using Unicode characters
`-tlatex`	`-latex`	LATEX	Further details can be found here
`-tlatex:nopreamble`	`-latex:nopreamble`	LATEX	Contains no LaTeX preamble creating
`-tbraille`	`-braille`	PNG	Braille image *[Ref. QA-4752]*

Example:

```
java -jar plantuml.jar yourdiagram.txt -ttxt
```

Configuration File

You can also provide a configuration file which will be included before each diagram:

```
java -jar plantuml.jar -config "./config.cfg" dir1
```

Metadata

- If you does not want plantuml to save the diagram's source code in the generated PNG Metadata, you c `nometadata` to disable this functionality (To NOT export metadata in PNG/SVG generated files).
- It is possible to retrieve this source with the `-metadata` option. This means that the PNG is almost "e you cannot install plugins, and someone in the future can update the diagram by getting the metadata, e stand-alone.
- Conversely, the `-checkmetadata` option checks whether the target PNG has the same source and if t thus saving all processing time. This allows you to run PlantUML on a whole folder (or tree with the -

Sounds like magic! No, merely clever engineering :-)

Example:

```
java -jar plantuml.jar -metadata diagram.png > diagram.puml
```

Unfortunately this option works only with local files. It doesn't work with `-pipe` so you cannot fetch a URI

However, the Plantuml server has a similar feature, where it can get a PNG from a URL and extract its metad

Exit code

When there are some errors in diagrams the command returns an error (-1) exit code. But even if some diagra which can be time consuming for large project.

You can use the `-failfast` flag to change this behavior to stop diagram generations as soon as one error oc and some will not.

There is also a `-failfast2` flag that does a first checking pass. If some error is present, no diagram will be even faster than `-failfast`, which may be useful for huge project.

Standard report [stdrpt]

Using the `-stdrpt` (standard report) option, you can change the format of the error output of y

With this option, a different error output of your diagram is possible:

- none: two lines
- `-stdrpt` : single line
- `-stdrpt:1` : verbose
- `-stdrpt:2` : single line

[Ref. Issue#155 and QA-11805]

Examples, with the bad file `file1.pu`, where `as` is written `aass` :

```
@startuml
participant "Famous Bob" aass Bob
@enduml
```

Without any option

```
java -jar plantuml.jar file1.pu
```

The error output is:

```
Error line 2 in file: file1.pu
Some diagram description contains errors
```

-stdrpt option

```
java -jar plantuml.jar -stdrpt file1.pu
```

The error output is:

В начало
Что нового ?
Быстрый старт
Online Server
Запуск
F.A.Q.
Скачать
Форум
Theme
Препроцессинг
Стандартная библиотека
Hitchhiker's Guide
PDF Guide

-stdrpt:1 option

```
java -jar plantuml.jar -stdrpt:1 file1.pu
```

The error output is:

```
protocolVersion=1
status=ERROR
lineNumber=2
label=Syntax Error?
Error line 2 in file: file1.pu
Some diagram description contains errors
```

-stdrpt:2 option (like -stdrpt)

```
java -jar plantuml.jar -stdrpt:2 file1.pu
```

The error output is:

```
file1.pu:2:error:Syntax Error?
```

Standard Input & Output

Using the `-pipe` option, you can easily use PlantUML in your scripts.

With this option, a diagram description is received through standard input and the PNG file is generated to sta system.

Example:

```
cat somefile.puml | java -jar plantuml.jar -pipe > somefile.png
```

The `-pipemap` option can be used to generate PNG map data (hyperlink rectangles) for use in HTML, eg:

```
cat somefile.puml | java -jar plantuml.jar -pipemap > somefile.map
```

The map file looks like this:

```
<map id="plantuml_map" name="plantuml_map">
<area shape="rect" id="id1" href="http://plantuml.com" title="http://plantuml.com"
      alt="" coords="1,8,88,44"/>
</map>
```

Note: Also take a look at `-pipedelimitor` and `-pipeNoStderr` to implement proper multiplexing of seve multiple diagrams), and error handling.

Help

You can have a help message by launching :

```
java -jar plantuml.jar -help
```

This will output:

```
        (to execute the GUI)
    or java -jar plantuml.jar [options] [file/dir] [file/dir] [file/dir]
        (to process files or directories)

You can use the following wildcards in files/dirs:
        *       means any characters but '\'
        ?       one and only one character but '\'
        **      means any characters (used to recurse through directories)

where options include:
    -darkmode           To use dark mode for diagrams
    -gui                To run the graphical user interface
    -tpng               To generate images using PNG format (default)
    -tsvg               To generate images using SVG format
    -teps               To generate images using EPS format
    -tpdf               To generate images using PDF format
    -tvdx               To generate images using VDX format
    -txmi               To generate XMI file for class diagram
    -tscxml             To generate SCXML file for state diagram
    -thtml              To generate HTML file for class diagram
    -ttxt               To generate images with ASCII art
    -tutxt              To generate images with ASCII art using Unicode characters
    -tlatex             To generate images using LaTeX/Tikz format
    -tlatex:nopreamble  To generate images using LaTeX/Tikz format without preamble
    -o[utput] "dir"     To generate images in the specified directory
    -DVAR1=value        To set a preprocessing variable as if '!define VAR1 value' we
    -Sparam1=value      To set a skin parameter as if 'skinparam param1 value' were u
    -Ppragma1=value     To set pragma as if '!pragma pragma1 value' were used
    -I\path\to\file     To include file as if '!include file' were used
    -I\path\to\*.puml   To include files with pattern
    -theme xxx          To use a specific theme
    -charset xxx        To use a specific charset (default is windows-1251)
    -e[x]clude pattern  To exclude files that match the provided pattern
    -metadata           To retrieve PlantUML sources from PNG images
    -nometadata         To NOT export metadata in PNG/SVG generated files
    -checkmetadata              Skip PNG files that don't need to be regenerated
    -version            To display information about PlantUML and Java versions
    -v[erbose]          To have log information
    -quiet              To NOT print error message into the console
    -debugsvek          To generate intermediate svek files
    -h[elp]             To display this help message
    -testdot            To test the installation of graphviz
    -graphvizdot "exe"  To specify dot executable
    -p[ipe]             To use stdin for PlantUML source and stdout for PNG/SVG/EPS g
    -encodesprite 4|8|16[z] "file"      To encode a sprite at gray level (z for compr
    -computeurl|-encodeurl      To compute the encoded URL of a PlantUML source file
    -decodeurl          To retrieve the PlantUML source from an encoded URL
    -syntax             To report any syntax error from standard input without genera
    -language           To print the list of PlantUML keywords
    -checkonly          To check the syntax of files without generating images
    -failfast           To stop processing as soon as a syntax error in diagram occu
    -failfast2          To do a first syntax check before processing files, to fail e
    -noerror            To skip images when error in diagrams
    -duration           To print the duration of complete diagrams processing
    -nbthread N         To use (N) threads for processing
    -nbthread auto      To use 4 threads for processing
    -timeout N          Processing timeout in (N) seconds. Defaults to 15 minutes (90
    -author[s]          To print information about PlantUML authors
    -overwrite          To allow to overwrite read only files
    -printfonts         To print fonts available on your system
    -enablestats        To enable statistics computation
    -disablestats       To disable statistics computation (default)
    -htmlstats          To output general statistics in file plantuml-stats.html
    -xmlstats           To output general statistics in file plantuml-stats.xml
    -realtimestats      To generate statistics on the fly rather than at the end
    -loopstats          To continuously print statistics about usage
    -splash             To display a splash screen with some progress bar
    -progress           To display a textual progress bar in console
    -pipeimageindex N   To generate the Nth image with pipe option
    -stdlib             To print standard library info
    -extractstdlib      To extract PlantUML Standard Library into stdlib folder
    -filedir xxx        To behave as if the PlantUML source is in this dir (only affe
    -filename "example.puml"    To override %filename% variable
    -preproc            To output preprocessor text of diagrams
    -cypher             To cypher texts of diagrams so that you can share them
    -picoweb            To start internal HTTP Server. See https://plantuml.com/picow

If needed, you can setup the environment variable GRAPHVIZ_DOT.
```