PropGen Documentation

Release 0

Holger Karl

February 20, 2012

CONTENTS

Contents:

CHAPTER

ONE

WHY THIS TOOL?

Writing an application for a research project is a challenging task: good ideas are needed, background research checks, a research hypothesis and a research program have to be formulated. When applying for a larger project, this is typically done by a group of people, coming from different organizations. A lot of work goes into the creative process - the mere act of writing the proposal, collecting information about the program structure, putting it in Gantt charts and tables of deliverables etc. should get out of the way!

Anybody who has tried to write a proposal for one of the European Union's Framework Programs knows that it can be a cumbersome process. The EU provides a relatively strict template which information to provide: information about work packages, tasks, deliverables, milestones, Gantt charts, etc. Much of this information is repeated at several places in the document, in various forms of presentation (tables, charts, free text). Merely keeping this information in synch can be a formidable challenge, in particular, when several people work on a proposal. To make matters worse, the EU only provides an MS Word template (and not a particularly well done one, either). There is no support to get all the administrative work out of the way.

This was the very situation we were in when we developed a proposal for a reasonably large EU proposal (an integrated project with about 15 partners). Instead of going down the Word-road, we decided to put all the information on a Wiki and to generate the actual proposal from there, using LaTeX to typeset the actual document and generating all the administrative information automatically. This has three main advantages:

- Wikis are easy to use even for novel users who are not used to using version control systems for collaborative work (let alone trying to distribute these files via email). Wikis naturally split up text in separate sections, circumventing the often problematic features of word processors to split up a document in smaller files.
- All the administrative information only needs to be entered ONCE. All possible presentations are automatically generated. They are guaranteed to stay synchronized. There is no time wasted for such work. Even non-trivial operations can be done until late in the proposal preparation without any risk (e.g., we decided to move a deliverable around a few hours before proposal submission - that would have been impossible with conventional tools).
- Wikis allow us to concentrate on the content, on our research ideas. We do not have to waste time fighting with a word processor.

To give one example: a task description for a workpackage looks as described in the figure in a moinmoin wiki. This then gets translated automatically into a Gantt chart for the workpackage (and into a Gantt chart for the project as a whole, and in tables, and in ...).

Tasks

Label	Start	Duration	Name	Lead partner
architectureDesign	1	3	Architecture Design	UE
archImprovement	4	2	Improvement of the Architecture	ABC
archImprovement	8	2	Improvement of the Architecture	ABC
archImprovement	12	2	Improvement of the Architecture	ABC
archFalsify	10	10	Falsification of the proposed Architecture	ISC

Figure 1.1: Example of a Wiki page, specifying several tasks for a workpackage (the architecture improvement task even has three phases).

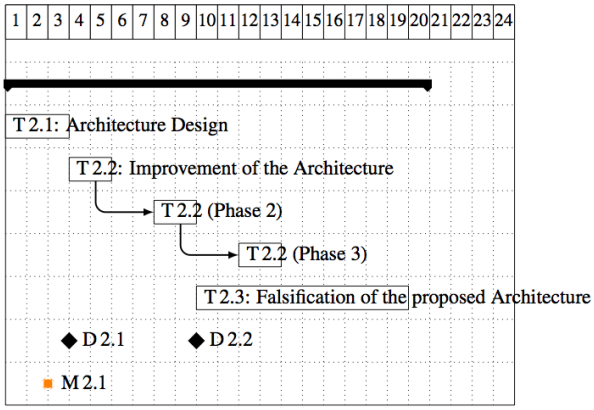

Figure 1.2: Resulting Gantt chart from the example task table (deliverable and milestones shown in this Gantt chart are defined in other Wiki tables)

Hence, the approach to go from a wiki to latex to PDF, and to submit this PDF file, has worked out nicely. It has produced a workflow that was reasonably easy for everybody, with full version control support without less IT-savy users needing to worry about it.

We felt that such a tool might be beneficial for a wider audience. So here it is - feel free to use it, to modify it, and to write interesting proposals using it. Our hope is that it will free up time from the mundane and boring tasks and enable all of us to concentrate more on the creative aspects of research.

Holger Karl

PS: When I write “we”, I refer to the team of colleagues engaged in the writing of said initial integrated research proposal. In particular, Bengt Ahlgren, Dirk Kutscher and Börje Ohlman deserve my thanks and gratitude for bearing through the rough-shot development of the inital version of this tool. I am indebted to them for constructive criticisms, ideas, and encouragement.

CHAPTER

TWO

INSTALLATION

2.1 Quick and dirty

The fastest possible way to get everything set up and produce a proposal PDF, assuming you have latex and python set up:

```
$ cd ~/tmp
$ wget --no-check-certificate https://github.com/hkarl/propgen/zipball/master --output-document=p
$ unzip propgen.zip
$ cd hkarl-propgen-7d7fd2d/
$ cd moin
$ python wikiserver.py &
$ cd ..
$ make
```

That will leave TestProject.pdf in the current directory. Use your webbrowser to go to http://127.0.0.1:8080/TestProject, make some changes to this page or to the Wiki pages linked from there, save the changes. Type make again in the shell. Gives an updated PDF file.

2.2 Usage scenarios

2.2.1 External Wiki

A typical usage scenario of this PropGen tool is the following:

- A proposal is to be prepared by a group of people.
- One of them runs the wiki, or an external Wiki provider is used
- Several people install the PropGen tool (ignoring the built-in Wiki) and can then built the PDF file for the proposal.
- Not everybody needs to install PropGen. Ideally, a version control system like SVN is integrated and whoever generates a new PDF file commits it to this version control system. Then, everybody has access to reasonably up-to-date versions.

This scenario is fairly straightforward to set up. I assume here that you have your external Wiki set up and know how to administer it.

2.2.2 Built-in Wiki

An alternative is to use the MoinMoin Wiki included in the PropGen distribution. Then, one partner has to run this wiki. Ideally and typically, the same machine is then able to run PropGen and to generate the proposal PDF. This can conveniently be triggered via a crontab (e.g., do an hourly build) and the result can be put into a version control system similar to above.

If other partners want to setup PropGen as well to locally generate the PDF, that is no problem at all.

Advantage of this approach is that the generation scripts can directly talk to the wiki and there is no need to go over the network to pull the wiki files. This is substantially more reliable, faster, and easier to setup (in particular, if there are firewalls or proxies in place, which can be real trouble). The disadvantage is that often, a Wiki is already in place, people have their accounts on it, are accustomed to its syntax and quirks, it can have powerful features not present in the provided MoinMoin installation (e.g, Twiki has a very useful butracker that can be very beneficial during proposal writing). The choice is yours!

2.2.3 Integrating a version control system

As outlined above, it can be extremely useful to integrate a version control system like SVN. I would recommend to limit this to the latex directory, only committing files in this directory.

Nothing is done automatically here since the variety of VCS systems is large. But it should be a simple exercise to integrate corresponding commit commands in the Makefile.

However, some care has to be taken in that symbolic links from the LaTeX directory to the "generated" directory are used. The reason is that it can be convenient, towards the end of a proposal preparation process, to stop pulling some parts of a proposal from the Wiki and to rather work on the LaTeX files directly. This allows better fine-tuning then working via the Wiki. The process is simple: replace the symbolic link by the actual file. Then, this file is used and it is not touched by the generation process.

2.3 Actual installation

2.3.1 Prerequisites

You need the following software installed:

Python You will need Python version 2.7.2 or later. Python 3 is known not to work at this time, Python 2.6 is too old.

Mechanize If you want to pull in Wiki files over the network (e.g., from a remote Twiki), then you need the python mechanize module installed. Details can be found on the Mechanize webpage (http://wwwsearch.sourceforge.net/mechanize/).

Tex An up-to-date LaTeX installation. TexLive 2011 was used for development. Non-standard packages or packages which needed patches are provided in the distribution.

Make There is a simple makefile in place. It is not absolutely needed and could quite easily be replaced by shell scripts or batch files.

Bash The makefile uses some simple loop and test constructs of bash. (See the clean target, e.g.) It should not be difficult to do without or provide a version for another shell.

Operation system Development and testing took place on Mac OSX Snow Leopard. Normal Linux distributions should pose no problems at all. Installation on Window is likely to be problematic because of symbolic links, and makefiles, bash etc. is likely to require at least cygwin - but I have very little clue of Windows and dare not make any statements here. Your mileage might vary.

Sphinx If you should want to generate the documentation for the reStructuredText markup (I have no idea why you would want to do that), you will also need Sphinx (http://sphinx.pocoo.org/) , at least version 1.1.2.

2.3.2 Installation

- Download the PropGen package and unpack to a folder of your choice.
  - From github:
    - Main page: https://github.com/hkarl/propgen

        * GIT Read-Only: git://github.com/hkarl/propgen.git
        * ZIP file: https://github.com/hkarl/propgen/zipball/master
    - Other sources still to come (possibly even a virtual machine)
- Decide which Wiki to use and set it up correctly.
    - Internal wiki: See *Setting up the MoinMoin Wiki included in distribution*
    - For both internal and external wiki: simply add information to settings.cfg (see *Setting up wiki in settings.cfg*)
- Add the templates to an external Wiki
    - Example templates for MoinMoin and Twiki are included in the templates folder. Ignore the directories; they are just to group the wiki files a bit. Each file becomes one Wiki page with the corresponding filename.
    - This step is not necessary when using the internal Wiki; it is pre-populated with an example pseudo project which should be easy to modify.
    - It might make sense to rename the "TestProject" page to some more specific for your project. (Then, remember to also rename the corresponding entry in settings.cfg.)
- Once you have setup Wiki access and the Wiki is running, try to generate a PDF file. cd into the main propgen directory and type make.

2.4 Setting up the MoinMoin Wiki included in distribution

If you want to use an external wiki (e.g., an existing Twiki), you can skip this section.

For more details, check the documentation of the MoinMoin Wiki.

2.4.1 Preconfigured account

The distributed version of MoinMoin is setup to support accounts, require login to edit or download material, and to deny anonymous access.

It has a preconfigured account ProjectMaster with password 123abc. This account ProjectMaster is configured as a superuser in MoinMoin. Check the lines

```
acl_rights_default = u"ProjectMaster:read,write,delete,revert,admin Known:read,write,revert,delete
```

and

```
superuser = [u"ProjectMaster"]
```

in the file moin/wikiconfig.py. It gives admin rights to the ProjectMaster account, and usual read, write, revert, delete rights to all other known accounts.

2.4.2 Change password

Obviously, you really, really want to change the password of this superuser. Log in as the user for which you want to change password, go to "Settings" (link on the very top of the page, to the left), then click "Change password".

Or go directly to http://127.0.0.1:8080/ProjectMaster?action=userprefs&sub=changepass (and replace "ProjectMaster" by the account name for which you want to change the password, of course).

2.4.3 Adding accounts

You could stick to the preconfigured account and distribute this account name and password to all members of your team. However, then it will not be possible to track who did which changes.

Hence, it is usually preferable to assign a dedicated username/password to each team member.

To add a user, you need to login as the superuser ProjectMaster. Go to the Wiki page NewUser (e.g., if you run it locally on the default port, goto http://127.0.0.1:8080/NewUser), and create as many users as you like.

2.4.4 Rename the main project page

In case you want a different main page name, simply use the "Rename page" action of the Wiki. Remember to rename the corresponding setting in settings.cfg as well!

2.4.5 Run the MoinMoin Wiki

Simple:

```
$ cd moin
$ python wikiserver.py &
```

You might want to start this as a daemon, possibly start automatically after reboot. Consult your own operating system how to do that.

2.5 Setting up wiki in settings.cfg

The file settings.cfg contains both basic configurations to ensure that the download script talks to the right wiki server as well as basic configuration options about what kind of information to generate. The latter content-customization options are described elsewhere. Here, we concentrate on basic connectivity settings.

2.5.1 Wiki

Set up the necessary information to access the wiki: which type, where can it be found, what is the start page, which account and password to use to log in.

projectName
: This option specificies the root Wiki page where the main project information can be found. It also serves as the filename of the file pdf file.

  Default: TestProject

wikitype
: The wikitype setting selects which type of wiki access is to be used. Wikitypes currently supported are: twiki, moinmoin and moinmoin-local

  Option 1: wikitype = moinmoin-local This option specifies that the a moinmoin wiki can be accessed in the same file system where the generation system executes. This option requires to sepcify the moinmoinpath option as well. No user and password need to be given, but the files must be accessible for reading.

  Option 2: moinmoin

  A moinmoin wiki is used, to be access remotely, using the mechanize library. To this end, both the wikiuser and wikipassword option need to be specified; they are used for login. Also, the baseURL option needs to be specified: it provides a URL where the wiki can be accessed, without any concrete page name. Example:

  baseURL = http://hk-vm.cs.uni-paderborn.de:8080/

  Option 3: twiki

It needs the same options as moinmoin. The difference is that the obtained files are parsed assuming the twiki syntax. The distribution's configuration file assumes a moinmoin-local

Default: moinmoin-local

moinmoinpath
Specify the moinmoin path in the standard PropGen distribution

Default: ../moin/

baseURL
For remote access to a wiki. It is ignored for the moinmoin-local wikitype and only used for other wikitypes.

Default: http://hk-vm.cs.uni-paderborn.de:8080/

wikiuser
The wikiuser account name to use to log in. Ignored in the moinmoin-local wikitype. The default corresponds to the account name preset in the distribution's moinmoin wiki. Change this option to reflect your own user name.

Default: ProjectMaster

wikipassword
Password used to log in. The default is the one used in the distribution's example moinmoin wiki. It is ignored in the moinmoin-local wikitype and only needed for remote access.

Default: 123abc

loginURL
Some remote wikis usually need a special login URL, e.g., Twiki wikis. Specify here. This setting is ignored in both the moinmoin-local wikitype (where no login is needed at all) and in the moinmoin wikitype (were the login URL is constructed directly from the baseURL and no separate URL is needed).

Default: https://twiki.sics.se/bin/login

httpProxyIP
For remote Wiki access through a proxy: These variables are used by the mechanize module, for access via an HTTP or HTTPS proxy. You can specify the proxy's IP, port number, and user and password, if needed. You can do that separately for HTTP and HTTPS access. But this does not always works out well. Also, this functionality is not well tested. Your mileage WILL vary! All the defaults are just empty.

Default:

httpProxyport
Default:

httpProxyuser
Default:

httpProxypassword
Default:

httpsProxyIP
Default:

httpsProxyport
Default:

httpsProxyuser
Default:

httpsProxypassword
Default:

CHAPTER

THREE

HOW TO USE PROPGEN

What to put on Wiki. How to trigger builds. What should go in which tables. What to watch out for when editing.

CHAPTER

FOUR

HOW TO CUSTOMIZE PROPGREN

How to customize the project.

4.1 Simple customization

Turn on, off certain parts, set colors.

4.2 Customize LaTeX templates

All the stuff that happens in template/latexTemplate.cfg

4.3 Complex customization

When you really have to work with the pyton code...

CHAPTER

FIVE

OPEN ISSUES

5.1 Known bugs

None, of course :-). If you find any, let me know!

5.2 Things still to do (TODO)

1. Put the bibtex file onto the Wiki as well. Probably better than to rely on version control to distribute it.

5.3 Debatable aspects

1. The settings files could be put on the Wiki as well. Two-edged sword: Might make it easier for everybody to configure things, but that is a serious downside as well. Technically, this would not be difficult to do. Not made up my mind yet.

5.4 Ideas for future features

1. Integrate a version control system like SVN for the produced LaTeX files
2. Generate PDF files directly from the Wiki, make it possible to trigger that at least.
3. Integrate Etherpad into Wiki
4. Build a bridge to the financial planning of a project.

- Either by parsing from/ writing to an Excel (or similar) spreadsheet. Relatively easy, but hard to make this general
- Or by putting spreadsheet-functionality onto the wiki. Hard to do for different wiki types (a nightmare, probably).

5. Build support for latexdiff. Possibly triggered from wiki as well?
6. Better support for figures in both wiki in latex. One idea might be to upload a PDF to the wiki and have the Wiki convert it to a PNG file. And the pull the PDF file directly from the wiki, without need to manually put it in the LaTeX figure directory.

CHAPTER

SIX

SOURCE CODE DOCUMENTATION

The code described here lists in the bin directory. Some general remarks:

- The invocation sequence is pullproject -> generateXML -> latexFromWiki -> latexFromXML -> ensureSymbolicLinks
- Details are in the makefile in the main directory
- Many functions get passed a parameter "config". This is the content of settings.cfg, as parsed by the standard python configuration file parser ConfigParser.SafeConfigParser (see python library documentation for details).

6.1 pullproject

Pull the raw wiki files from wherever is specified in settings.cfg. Store the raw wiki syntax in the wikipath directory.

`pullProject.`**`ensureDirectories`**(*config*)
A small helper function that makes sure that all the directories that are mentioned in settings.cfg PathNames section actually exist. This can be useful after a make clean or in case directories have been manually and inadvertently removed.

`pullProject.`**`getPartners`**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
get all the partner description files

`pullProject.`**`getProposalStructure`**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
Extract all the relevant files for the actual proposal text from the wiki.

`pullProject.`**`getWorkpackages`**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
Identify all the workpackages and download them

6.2 wikiParser

6.2.1 The wikiParser module as such

We need to parse various wiki formats into useable latex. This module provides an abstract base class wikiParser that implements a lot of basic functions e.g., to extract tables, lists, etc.

This base class has to be subclassed to specialize for specific Wiki syntax variants. The subclasses can be fairly slim and mostly specify regular expressions to use (e.g., how to recognize headings).

A factory function is called to obtain an instance of such a parser.

`wikiParser.`**`wikiParserFactory`**(*config*)
Construct an instance of the correct parser class, choice depends on what is selected in settings.cfg.

6.2.2 The wikiParser base class

class wikiParser.**wikiParser**
Base class to get the interface for turning wiki syntax into useful stuff

constructLabel(*t*)
Given a heading, construct a suitable label out of it. Remove whitspaces and obvious strange characters.

getLaTeX(*t*)
turn all of the wiki into LaTeX

getList(*wiki*)
turn the first itemize in the wiki into a list

getListAsDict(*wiki*, *delimiter=':'*)
tmp is an aray of strings, assumed to be key/values delimited by delimited split them up, return a proper dictionary for that

getSection(*wiki*, *title*, *level*)
extract the section with title at level

getTable(*wiki*)
turn the first table into list of dictionaries, using the first row as keys for the dictionaries. Removing boldfacing from the first row entries.

moveCommissionHints(*t*)
Make sure that commission hints appear after the first heading!

6.2.3 The moinmoin parser

class wikiParser.**wikiParserMoinmoin**(*config*)
Specialized for Moinmoin

getSection(*wiki*, *title*, *level*)
extract the section with title at level

6.2.4 The twiki parser

class wikiParser.**wikiParserTwiki**(*config*)
Specialized for Twiki

getSection(*wiki*, *title*, *level*)
extract the section with title at level

localHeading(*title*, *level*)
How does a heading with the given title, at the given level, look in this wiki style? Describe it as a regular expression.

6.3 latexFromWiki

latexFromWiki.**handleFile**(*f*, *outdir*, *parser*, *config*, *verbose=False*)
Translate a wiki file with name f to the corresponding LaTeX file.

Information where and how to translate are giving in config. Parser is a parser object for the correct wiki style.

6.4 latexFromXML

general docu for latexFromXML

6.5 settings

settings.**getSettings**(*filename*)
Try to find the settings file, turn it into a configParser object, and do some first preprocessing on it.

6.6 ensureSymbolicLinks

Make sure that the symbolic links from the manual to the generated subtree exist, unless there is already a real file there.

ensureSymbolicLinks.**createLinks**(*config*)
Look into config, in the Paths section, for any directory with generated as prefix, and has a corresponding manual directory as peer. Put symbolic links there if necessary to all files in generated.

6.7 utils

utils.**roundPie**(*l*)
round the values to 100%, input: list of (name, value) tuples

utils.**searchListOfDicts**(*l*, *key*, *value*, *returnkey*)
Search a list l which contains dictionaries for an entry where key has value, and return the value of returnkey.

utils.**treeReduce**(*l*, *reducefct*)
recursively apply a reduce function to a nested list structure. Atomic elements must be boolean values.

utils.**writefile**(*t*, *f*)
Write text t into file f. If flag utf8conversion is set, try to run a conversion into UTF-8.

6.8 settings.cfg

6.8.1 Wiki

Set up the necessary information to access the wiki: which type, where can it be found, what is the start page, which account and password to use to log in.

projectName
This option specificies the root Wiki page where the main project information can be found. It also serves as the filename of the file pdf file.

Default: TestProject

wikitype
The wikitype setting selects which type of wiki access is to be used. Wikitypes currently supported are: twiki, moinmoin and moinmoin-local

Option 1: wikitype = moinmoin-local This option specifies that the a moinmoin wiki can be accessed in the same file system where the generation system executes. This option requires to sepcify the moinmoinpath option as well. No user and password need to be given, but the files must be accessible for reading.

Option 2: moinmoin

A moinmoin wiki is used, to be access remotely, using the mechanize library. To this end, both the wikiuser and wikipassword option need to be specified; they are used for login. Also, the baseURL option needs to be specified: it provides a URL where the wiki can be accessed, without any concrete page name. Example:

baseURL = http://hk-vm.cs.uni-paderborn.de:8080/

Option 3: twiki

It needs the same options as moinmoin. The difference is that the obtained files are parsed assuming the twiki syntax. The distribution's configuration file assumes a moinmoin-local

Default: moinmoin-local

moinmoinpath
Specify the moinmoin path in the standard PropGen distribution

Default: ../moin/

baseURL
For remote access to a wiki. It is ignored for the moinmoin-local wikitype and only used for other wikitypes.

Default: http://hk-vm.cs.uni-paderborn.de:8080/

wikiuser
The wikiuser account name to use to log in. Ignored in the moinmoin-local wikitype. The default corresponds to the account name preset in the distribution's moinmoin wiki. Change this option to reflect your own user name.

Default: ProjectMaster

wikipassword
Password used to log in. The default is the one used in the distribution's example moinmoin wiki. It is ignored in the moinmoin-local wikitype and only needed for remote access.

Default: 123abc

loginURL
Some remote wikis usually need a special login URL, e.g., Twiki wikis. Specify here. This setting is ignored in both the moinmoin-local wikitype (where no login is needed at all) and in the moinmoin wikitype (were the login URL is constructed directly from the baseURL and no separate URL is needed).

Default: https://twiki.sics.se/bin/login

httpProxyIP
For remote Wiki access through a proxy: These variables are used by the mechanize module, for access via an HTTP or HTTPS proxy. You can specify the proxy's IP, port number, and user and password, if needed. You can do that separately for HTTP and HTTPS access. But this does not always works out well. Also, this functionality is not well tested. Your mileage WILL vary! All the defaults are just empty.

Default:

httpProxyport
Default:

httpProxyuser
Default:

httpProxypassword
Default:

httpsProxyIP
Default:

httpsProxyport
Default:

httpsProxyuser
Default:

httpsProxypassword
Default:

6.8.2 PathNames

binpath
Caution: best not to touch these paths!!! You should know what you are doing here! note: these paths are relative to the main directory where are all the scripts?

Default: bin

wikipath
where should downloaded wiki files be stored?

Default: generated/wiki

wikiwppath
Default: generated/wiki/wp

wikipartnerpath
Default: generated/wiki/partners

xmlpath
where do generated xml files go?

Default: generated/xml

xmlwppath
Default: generated/xml/wp

latexTemplates
where is the LaTeX templates file?

Default: template/latexTemplates.cfg

genlatexpath
where do generated LaTeX files, wp paths, partner files go?

Default: generated/latex

genlatexfigurespath
Default: generated/latex/figures

genlatexganttspath
Default: generated/latex/gantts

genlatextablespath
Default: generated/latex/tables

genlatexpiespath
Default: generated/latex/pies

genlatexwppath
Default: generated/latex/wp

genlatexpartnerspath
Default: generated/latex/partners

manuallatexpath
where are the MANUAL latex files? usage: in that directory, no files are EVER overwritten however, it is ensured that for all generated files, there is either a regular file of the same name in the manual directory (then nothing happens), or a symbolic link is created in the manual directory. NOTE: no good idea how to replicate that behavior

Default: latex

manuallatexfigurespath
Default: latex/figures

manuallatexganttspath
Default: latex/figures/gantts

manuallatextablespath
Default: latex/figures/tables

manuallatexpiespath
Default: latex/figures/pies

manuallatexwppath
Default: latex/wp

manuallatexpartnerspath
Default: latex/partners

6.8.3 Gantts

TODO: excel interface! excelFile = ../Administration/Steam-resources.xls control which and how Gantt charts are typeset

WpMilestonesUncompressedShow
toggle to turn on/off various figures: some simple examples - generate various WP-specific Gantt charts for milestones, deliverables

Default: True

WpMilestonesShow
Default: True

WpDeliverablesUncompressedShow
Default: True

ShowWPBar
for the full-project milestones/deliverable gantt charts: build a WP bar to separate WPs ?

Default: True

ganttPerWPShowsLegend
should the WP-specifc Gantt charts show a legend?

Default: True

ganttLegendTwoColumn
should a Gantt legend be one or two columns?

Default: True

ganttTaskbarsShowTaskname
task bars show task names in the gantt chart? easier to read, but risk of text extending to the right if not, maybe center the task identifier in the bar? (difficult!)

Default: True

ganttDistanceBetweenMS
More detailed control about looks of Gantt charts: number of months that a deliverable/milestone marker text occupies (horizontally)

Default: 4

milestoneDecoration = fill=orange, rounded corners
how should a milestone look like? Put a "decoration string" according to the pdfgantt package here

Default: 5pt

milestonesShowCrossWP
show cross-WP milestones?

Default: True

deliverablesShowCrossWP
show cross-WP milestones?

Default: True

milestoneLegendTemplate
template to format the milestone captions in the Gantt charts

Default: item textbf{${id}}: ${Title}

deliverableLegendTemplate
Default: item textbf{${id}}: ${Title}

6.8.4 Summaries

a plausible alternative for the legend strings is to use a description environment instead. That requires then a corresponding change in latexTemplates.cfg. Not difficult to do, look for compactitem there. which summary tables, figures should appear in the document

showEffortPartnerWPs
one table showing efforts only per partner and workpackage, over all workpackages

Default: True

showEffortPartnerTasks
one table showing effort per partner and task, over all tasks

Default: True

piePMsWPs
a pie chart, showing person month summaries for WPs

Default: True

piePMsPartners
a pie chart, showing person month summaries for each partner

Default: True

piePMsPartnerTypes
a pie chart, showing person month summaries for each partner type (industry, SME, academic)

Default: True

piePMsNations
a pie chart, showing person month summaries for each nation

Default: True

6.8.5 WPTables

the following pie charts are budget-related; makes only sense once the spreadsheet coupling is implemented a pie chart, showing total/contributed budget, per partner pieTotalPerPartner = True pieContribPerPartner = False a piec chart, showing total/contributed budget, per partner TYPE pieTotalPerPartnerType = False pieContribPerPartnerType = False a piec chart, showing total/contributed budget, per NATION pieTotalPerNation = False pieContribPerNation = False all information that is relevant for workpackage tables

maxPartnersPerRow
how many partners should be typeset in one row of the WP tables? choice depends mostly on how long the partner shortnames are

Default: 8

firstColumnWidth
how wide should the first column be? give it as percent of textwidth!

Default: 15

wptablespacing
how to influence the spacing of the wp tables? wptablespacing = @{hskip 2.8ex}

Default: @{hskip 0ex}

tabularCorrection
correction factor to eadjust total width of the WP tabular (Note: I'd much appreciate help from a LaTeX wizard to ensure a tabular environment is at most textwidth wide)

Default: 0.95

tasklistShowsDuration
what information should the task list include (in the WP table):

Default: True

tasklistShowsPartners
Default: True

tasklistShowsDeliverables
Default: True

tasklistShowsMilestones
Default: True

wpdescriptionShowsLeader
how much details should the workpackage description box report?

Default: True

taskboxShowsLeader
how much details should the individual task boxes report? (the boxes containing the description of each task) (this is used in latex-templates.cfg in the WpTasksDescriptions section)

Default: True

taskboxShowsObjectives
showsLeader: refers to the head of the taskbox, not to the list at the end

Default: True

taskboxShowsDescription
Default: True

taskboxShowsDeliverables
Default: True

taskboxShowsMilestones
Default: True

taskboxShowsPartners
Default: True

deliverablesWPshowDue
how much details in the deliverables list per WP? similar to tasklist options

Default: True

deliverablesWPshowTasks
Default: True

deliverablesWPshowPartners
Default: True

deliverablesWPshowDescription
Default: True

milestonesWPshowTasks
how much details in the milestones list per WP? similar to tasklist options

Default: True

milestonesWPshowPartners
Default: True

milestonesWPshowDue
Default: True

milestonesWPshowDescription
Default: True

colorInactivePartner
color highlight for inactive partners? (color names as defined by the LaTeX xcolor package) (no highlight: simply use black)

Default: gray

6.8.6 Participants

Options to control the individual participant descriptions in Section 2.2

newpageAfterEachPartner
Default: True

6.8.7 LaTeX

Options controlling LaTeX processing produces file setttings.tex, included by frame.tex NOTE: these settings are only processed in the makefile; changing them and directly running pdflatx will have no effect

showCommissionHints
should the PDF file include the commission hints text?

Default: True

useShowkeys
should the showkey package be used, highlighting the label, ref and cite commands?

Default: False

showWarnings
should warnings and fixmes be printed?

Default: True

showListOfTables
should the file show list of tables?

Default: True

showListOfFigures
should the file show list of figures?

Default: True

showAcronymList
should there be a list of acronyms?

Default: False

useMultipageDeliverableTable
multipage deliverables/milestone Table? (defaults False / True only for demonstration of the options, pick what you prefer...)

Default: False

useMultipageMilestoneTable
Default: True

useMultipageEffortTable
should the effort table for the entire project be typesetting across multiple pages?

Default: True

effortTableLandscape
turn the effort table sideways? Can be useful for large consortia

Default: True

6.8.8 CustomLaTeX

custom LaTeX commands Rationale: there might be some things you'd like to include in your proposal that are not fit for making in general, but can be computed based on the numbers contained in files pulled form the wiki E.g., the toal number of person months to this end, this section allows you to write python code that is executed once every thing else is done and assign the result to a LaTeX command. The defining command will end up in settings.tex the command is the option name CAUTION: you can really screw up everything here. It can delete your disk and kill your pet. You are WARNED! To make use of this feature, you have to understand the Python code!

totalPM
e.g.: total person months should be defined like this:

Default: sum([int(e['resources']) for e in allEfforts])

tocLevel
and it will turn into a LaTeX command in settings.tex like this (where 999 is of course replaced by the result of computing based on actual data): newcommand{totalPM}{999} toc level

Default: 3

secNumDepth
how deeply should headings be numbered?

Default: 3

6.9 latexTemplates.cfg

To get this right: First line must be on the same line as the
Default: ;

6.9.1 titleheader

we build the titlepage in three steps: the header, the table rows for the partners, and then the titlepage complete out of these two parts. This third part is then written to file

template
Default: begin{center} {LARGE ${instrument} } \[.2cm] {large ${call} } \[.4cm] {LARGE textbf{ ${projectname} }} \[.3cm] {LARGE Acronym: textbf{ ${projectshort} }} \[.3cm] end{center} {large Date of Preparation: today }\[1em] begin{large} begin{description} item[Work program topics addressed:] ${topics} item[Coordinator:] ${coordinatorname} item[e-mail:] {url{${coordinatoremail}}} item[tel/fax:] ${coordinatorphone} end{description} end{large} noindent

dict
Default: titlepageDict

6.9.2 partnerTableRow

rows for partner table on titlepage:

template
Default: ${Number} & ${Name} & ${Shortname} & ${Nation}

list
Default: partnerList

joiner
Default: \ n

sorter
to demonstrate how to sort such a list, let's sort if by number Note: sorter is optional, but only available in conjunction with joiner attribute

Default: lambda x: int(x['Number'])

6.9.3 titlepage

some alternative examples (which make no sense here, just to demonstrate): sorter = lambda x: x['Shortname'] sorter = lambda x: x['Nation'] and the actual titlepage:

template
Default: ${titleheader} { begin{tabular}{cp{8cm}ll} toprule Participant no. & Participant organisation & Short name & Country \ midrule ${partnerTableRow} \ bottomrule end{tabular} }

dict
Default: expanded

file
Default: True

6.9.4 wpSummaryRows

the wp summary list

template
Default: WP ${Number} & ${Name} & ${Type} & ${Leadernumber} & ${Leadership} & ${wpeffort} & ${Start} & ${End}

list
Default: allWPDicts

joiner
Default: \ n

6.9.5 wpsummarytable

template
Default: begin{table}[bhtp] caption{Summary table of all work packages} label{tab:wpsummary} begin{tabular}{cp{0.25textwidth}cccrcc} toprule WP No. & WP name & Type of & Lead & Lead & Person- & Start & End \ & & activity & part. no. & short name & months & month & month \ midrule ${wpSummaryRows} \ midrule multicolumn{2}{l}{Total:} & & & & totalPM & & \ bottomrule end{tabular} end{table}

dict
Default: expanded

file
Default: True

dir
Default: tables

6.9.6 ganttPrefix

Gantt charts! First, some building blocks for various Gantt charts

template
Default: begin{tikzpicture} begin{ganttchart}[vgrid,hgrid, x unit=0.371cm, y unit chart = 0.75cm, title label font={footnotesize}, bar height = 0.55, bar top shift = 0.225, inline, milestone label font=color{black}small, milestone label inline anchor={right=.1cm}, bar label inline anchor={anchor=west}, bar label font=small, link={-latex, rounded corners=1ex, thick}]{${duration}} gantttitlelist{1,...,${duration}}{1}

dict
Default: titlepageDict

6.9.7 ganttPostfix

template
Default: end{ganttchart} end{tikzpicture}

6.9.8 WpMilestonesUncompressedShow

and the actual gantts: first, the wp-specific gantts

template
Default: begin{figure}[htbp] ${ganttPrefix} \${milestoneUncompressedGanttString} ${ganttPostfix} caption{Gantt chart of all milestones of Work package ${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

6.9.9 WpMilestonesShow

template
Default: begin{figure}[htbp] ${ganttPrefix} \${milestoneGanttString} ${ganttPostfix} caption{Gantt chart of all milestones of Work package ${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

6.9.10 WpDeliverablesUncompressedShow

template
Default: begin{figure}[htbp] ${ganttPrefix} \ ${deliverableUncompressedGanttString} ${ganttPostfix} caption{Gantt chart of all deliverables of Work package ${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

6.9.11 WpDeliverablesShow

template
Default: begin{figure}[htbp] ${ganttPrefix} \ ${deliverableGanttString} ${ganttPostfix} caption{Gantt chart of all deliverables of Work package ${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

6.9.12 ganttWP

template
Default: begin{figure}[htbp] centering{${ganttPrefix}\ ifthenelse{boolean{Gantts-ShowWPBar}}{ ${groupbar}}{} ${taskGantt} ${deliverableGanttString} \ ${milestoneGanttString} ${ganttPostfix}} ifthenelse{boolean{Gantts-ganttPerWPShowsLegend}}{ ifthenelse{boolean{Gantts-ganttLegendTwoColumn}} {begin{multicols}{2} begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} end{compactitem} end{multicols} } { % single-column legends: begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} end{compactitem} } }{} caption{Gantt chart for Work package ${Number}: ${Shortname}} label{fig:gantt-WP${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

numerator
do not change the numerator; else, wPInclude.tex will look for the wrong files

Default: value['Shortname']

dir
Default: gantts

6.9.13 ganttWPLegend

template
separate legends per WP gantts only make sense if they are not already included in the

Default: begin{figure}[htbp] ifthenelse{boolean{Gantts-ganttLegendTwoColumn}} {begin{multicols}{2} begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} end{compactitem} end{multicols} } {% single-column legends: begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} end{compactitem} } caption{Gantt chart of Work package ${Number}: ${Shortname}} label{fig:gantt-Legend-WP${Number}} end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: False

numerator
do not change the numerator; else, wPInclude.tex will look for the wrong files

Default: value['Shortname']

dir
Default: gantts

6.9.14 allTaskDelMSList

and prepate the complete Gantt chart for the entire project this happens in several steps

template
Default: ifthenelse{boolean{Gantts-ShowWPBar}}{ ${groupbar}}{} ${taskGantt} ${deliverableGanttString} \ ${milestoneGanttString}

list
Default: allWPDicts

dir
Default: gantts

6.9.15 allTaskDelMS

the next one is an example with an empty template: it is only used to turn a list into a string

template
Default:

list
Default: expanded['allTaskDelMSList']

joiner
Default: \ n

6.9.16 allDelLegend

template
Default: ${ganttLegend}

list
Default: allDeliverables

joiner
Default: n

6.9.17 allMSLegend

template
Default: ${ganttLegend}

list
Default: allMilestones

joiner
Default: n

6.9.18 CompleteGantt

template
Default: begin{figure}[htbp] centeringmaxsizebox{0.95textwidth}{0.95textheight}{ ${ganttPrefix} \ ${allTaskDelMS} ${ganttPostfix} } % closes adjustbox caption[Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones]{Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones (legend in Table~ref{fig:allDelMSLegend})} label{fig:completeGantt} end{figure}

dict
Default: expanded

file
Default: True

dir
Default: gantts

6.9.19 allLegend

template
Default: begin{table} caption[List of all deliverables and milestones]{List of all deliverables and milestones shown in Figure~ref{fig:completeGantt}} label{fig:allDelMSLegend} begin{multicols}{2} begin{compactitem} ${allDelLegend} ${allMSLegend} end{compactitem} end{multicols} end{table}

dict
Default: expanded

file
Default: True

dir
Default: gantts

6.9.20 CompleteGanttFacingLegend

template
this is particularly useful for a double-sided printing layout chart is on a left page, Legend on a right page

Default: cleardoubleevenstandardpage centeringmaxsizebox{0.95textwidth}{0.95textheight}{ ${ganttPrefix} \ ${allTaskDelMS} ${ganttPostfix} } % closes adjustbox captionof{figure}[Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones]{Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones (legend in Table~ref{fig:allDelMSLegend})} label{fig:completeGantt} clearpage captionof{table}[List of all deliverables and milestones]{List of all deliverables and milestones shown in Figure~ref{fig:completeGantt}} label{fig:allDelMSLegend} begin{multicols}{2} begin{compactitem} ${allDelLegend} ${allMSLegend} end{compactitem} end{multicols}

file
Default: True

dict
Default: expanded

dir
Default: gantts

6.9.21 WpTasks

and finally: the actual WP files!

template
Default: labitem{${taskId}}{task:${Label}} ${Name} ifthenelse{boolean{WPTables-tasklistShowsDuration}}{ hfill (%{ ‘, ‘.join([(‘M,’ + str(t[’Start’]) + ‘ – M,’ + str(t[’Start’] + t[’Duration’] -1)) for t in allTasks if t[’Label’] == ‘${Label}’]) %}) }{} ifthenelse{boolean{WPTables-tasklistShowsPartners}}{ \ Contributing partners: %{ ‘, ‘.join([((r”textbf{%s}” % x) if (x == ‘${Leadpartner}’) else x) for x in sorted([p[’partner’] for p in allEfforts if p[’task’] == ‘${Label}’], key = lambda x: [int(pl[’Number’]) for pl in partnerList if pl[’Shortname’] == x][0])]) %} }{} ifthenelse{boolean{WPTables-tasklistShowsDeliverables}}{ %{ (r’\ Contributing to Deliverables: ‘ + ‘, ‘.join([((r”textbf{ %s }” % d[’id’]) if (d[’ProducingtaskMain’] == ‘${Label}’) else d[’id’]) for d in allDeliverables if ‘${Label}’ in d[’Producingtask’]]) if [d[’id’] for d in allDeliverables if ‘${Label}’ in d[’Producingtask’]] else ‘’) %} }{} ifthenelse{boolean{WPTables-tasklistShowsMilestones}}{ %{ (r’\ Contributing to Milestones: ‘ + ‘, ‘.join([((r”textbf{ %s }” % d[’id’]) if (d[’ProducingtaskMain’] == ‘${Label}’) else d[’id’]) for d in allMilestones if ‘${Label}’ in d[’Producingtask’]]) if [d[’id’] for d in allMilestones if ‘${Label}’ in d[’Producingtask’]] else ‘’) %} }{}

list = [t for t in allTasks if t[’Main’]=
we only show this for the main task, not for all the individual tasks need to think about the duration, though!

Default: ‘True’]

groupby
Default: wp

joiner
Default: n

6.9.22 WpTasksDescriptions

template
Default: begin{framed} noindent textbf{Description of Task ${taskId}: ${Name}} ifthenelse{boolean{WPTables-taskboxShowsLeader}}{(Task leader: ${Leadpartner})}{} ifthenelse{boolean{WPTables-taskboxShowsObjectives}}{ ~\[0.2cm] textbf{Task objectives:} ${taskobjectives}}{} ifthenelse{boolean{WPTables-taskboxShowsDescription}}{~

\[0.2cm] textbf{Description of work:} ${taskdescription}}{} ifthenelse{boolean{WPTables-taskboxShowsDeliverables}}{ %{ (r"""\[0.3cm] noindent {centering begin{tabular}{lp{0.7textwidth}l} multicolumn{3}{l}{textbf{Deliverables contributed to by Task ${taskId}:} } \ toprule Del.no. & Deliverable name & Due \ midrule %s \ bottomrule end{tabular} }""" % (r'\ '.join([(d['id'] + " & " + d['Title'] + " & M," + str(d['Monthdue'])) for d in allDeliverables if '${Label}' in d['Producingtask']])) if [d for d in allDeliverables if '${Label}' in d['Producingtask']] else "") %} }{} ifthenelse{boolean{WPTables-taskboxShowsMilestones}}{ %{ (r"""\[0.3cm] noindent {centering begin{tabular}{lp{0.7textwidth}l} multicolumn{3}{l}{textbf{Milestones contributed to by Task ${taskId}:} } \ toprule MS.no. & Milestone name & Due \ midrule %s \ bottomrule end{tabular} }""" % (r'\ '.join([(d['id'] + " & " + d['Title'] + " & M," + str(d['Monthdue'])) for d in allMilestones if '${Label}' in d['Producingtask']])) if [d for d in allMilestones if '${Label}' in d['Producingtask']] else "") %} }{} ifthenelse{boolean{WPTables-taskboxShowsPartners}}{ ~ \[0.2cm] noindent textbf{Partners contributing to this task:} %{ ", ".join([(r'textbf{%s}' % x['partner'] if x['partner'] == '${Leadpartner}' else x['partner']) for x in allEfforts if x['task'] == '${Label}' and int(x['resources']) > 0]) %} }{} end{framed}

list = [t for t in allTasks if t['Main']=
: Default: 'True']

groupby
: Default: wp

joiner
: Default: n

6.9.23 WpDeliverables

the deliverable and milestone list per WP, along with more detailed description

template
: Default: item textbf{${id}}: ${Title} ifthenelse{boolean{WPTables-deliverablesWPshowDue}}{ hfill (M,${Monthdue})}{} ifthenelse{boolean{WPTables-deliverablesWPshowTasks}}{ \ Contributing tasks: ${ProducingtaskString}}{} ifthenelse{boolean{WPTables-deliverablesWPshowPartners}}{ \ Contributing partners: ${ContributorString}}{} ifthenelse{boolean{WPTables-deliverablesWPshowDescription}}{ \ Brief description: ${Description}}{}

list
: Default: allDeliverables

groupby
: Default: wp

joiner
: Default: n

6.9.24 WpMilestones

template
: Default: item textbf{${id}}: ${Title} ifthenelse{boolean{WPTables-milestonesWPshowDue}}{ hfill (M,${Monthdue})}{} ifthenelse{boolean{WPTables-milestonesWPshowTasks}}{ \ Contributing tasks: ${ProducingtaskString}}{} ifthenelse{boolean{WPTables-milestonesWPshowPartners}}{ \ Contributing partners: ${ContributorString}}{} ifthenelse{boolean{WPTables-milestonesWPshowDescription}}{ \ Brief description: ${Description}}{}

list
: Default: allMilestones

groupby
: Default: wp

joiner
: Default: n

6.9.25 Wp

template
Default: newpage noindent addcontentsline{toc}{subsubsection}{WP ${Number}: ${Shortname} } ${tableheader} begin{framed} noindent textbf{Objectives of Workpackage ${Number}:} ${objectives} end{framed} begin{framed} noindent textbf{Tasks of Workpackage ${Number}:} begin{compactdesc} ${WpTasks_${Number}} end{compactdesc} {footnotesize emph{Lead partners are shown in bold.}} end{framed} begin{framed} noindent textbf{Description of Workpackage ${Number}:} ifthenelse{boolean{WPTables-wpdescriptionShowsLeader}}{(workpackage leader: ${Leadership})}{} ${wpdescription} end{framed} ${WpTasksDescriptions_${Number}} begin{framed} noindent textbf{Deliverables for Workpackage ${Number}:} begin{compactdesc} ${WpDeliverables_${Number}} end{compactdesc} noindent textbf{Milestones for Workpackage ${Number}:} begin{compactdesc} ${WpMilestones_${Number}} end{compactdesc} end{framed} % let us pull in the gantt chart for this WP directly here, % no need to use a separate file: ${ganttWP_${Shortname}}

list
Default: allWPDicts

dict
Default: expanded

numerator
Default: value['Shortname']

file
Default: True

dir
Default: wp

6.9.26 DeliverableTableRows

a table to summarize all the deliverables again: two steps: build the rows, and then the table

template
Default: ${id} & ${Title} & ${Monthdue} & ${Nature} & ${Dissemination} & ${ProducingtaskString}

list
Default: allDeliverables

joiner
Default: \ n

sorter
this sorts according to due date, and where the due date is the same, use deliverable id:

Default: lambda x: '%03d' % x['Monthdue'] + x['id']

6.9.27 DeliverableTableShort

if you just want to sort by id, then simply use: sorter = lambda x: x['id']

template
Default: begin{table}[hbtp] caption{Summary of all deliverables (Nature: O=Other, P=Prototype, R=Report; Dissemination: PU=Public, RE=Restricted, CO=Confidential)} label{tab:deliverablesummary} begin{center} begin{tabular}{|p{0.4textwidth}|c|c|c|p{0.1textwidth}|} toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\ midrule ${DeliverableTableRows} \ bottomrule end{tabular} end{center} end{table}

dict
Default: expanded

6.9.28 DeliverableTableLong

template
Default: topcaption{Summary of all deliverables} label{tab:deliverablesummary} tablefirsthead{ toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\ midrule } tablehead{toprule multicolumn{6}{r}{emph{Table~ref{tab:deliverablesummary} continues from previous page}} \ toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\ midrule } tabletail{bottomrule multicolumn{6}{r}{emph{Table~ref{tab:deliverablesummary} continues on next page}} \ bottomrule} tablelasttail{multicolumn{6}{r}{Table~ref{tab:deliverablesummary} ends} \ bottomrule} begin{center} begin{mpxtabular}{|l|p{0.4textwidth}|c|c|c|p{0.1textwidth}|} ${DeliverableTableRows}\ bottomrule end{mpxtabular} end{center}

dict
Default: expanded

6.9.29 DeliverableTable

template
Default: ifthenelse{boolean{LaTeX-useMultipageDeliverableTable}}{${DeliverableTableLong}}{${DeliverableTableShort}

dict
Default: expanded

file
Default: True

dir
Default: tables

6.9.30 MilestoneTableRows

and a table for the milestones - same structure as for the deliverable table

template
Default: ${id} & ${Title} & ${Monthdue} & ${Verificationmeans} & ${ProducingtaskString}

list
Default: allMilestones

joiner
Default: \ n

sorter
Default: lambda x: '%03d' % x['Monthdue'] + x['id']

6.9.31 MilestoneTableShort

template
Default: begin{table}[hbtp] caption{Summary of all milestones} label{tab:milestonessummary} begin{center} begin{tabular}{|l|p{0.3textwidth}|c|p{0.3textwidth}|p{0.1textwidth}|} toprule Number & Title & Due date & Means of verification & Contributing task(s) \ midrule ${MilestoneTableRows}\ bottomrule end{tabular} end{center} end{table}

dict
Default: expanded

6.9.32 MilestoneTableLong

template
Default: topcaption{Summary of all milestones} label{tab:milestonessummary} tablefirsthead{ toprule Number & Title & Due date & Means of verification & Contributing task(s) \ midrule } tablehead{toprule multicolumn{5}{r}{emph{Table~ref{tab:milestonessummary} continues from previous page}} \ toprule Number & Title & Due date & Means of verification & Contributing task(s) \ midrule } tabletail{bottomrule multicolumn{5}{r}{emph{Table~ref{tab:milestonessummary} continues on next page}} \ bottomrule} tablelasttail{multicolumn{5}{r}{Table~ref{tab:milestonessummary} ends} \ bottomrule} begin{center} begin{mpxtabular}{|p{0.3textwidth}|c|p{0.3textwidth}|p{0.1textwidth}|} ${MilestoneTableRows}\ bottomrule end{mpxtabular} end{center}

dict
Default: expanded

6.9.33 MilestoneTable

template
Default: ifthenelse{boolean{LaTeX-useMultipageMilestoneTable}}{${MilestoneTableLong}}{${MilestoneTableShort}}

dict
Default: expanded

file
Default: True

dir
Default: tables

6.9.34 summaryEffortRows

template
Default: ${Shortname} & %{ ' & '.join([str(x['partnereffort']['${Shortname}']) for x in allWPDicts]) %} & %{ str(sum([x['partnereffort']['${Shortname}'] for x in allWPDicts])) %}

list
Default: partnerList

joiner
Default: \ n

6.9.35 summaryEffort

template
Default: ifthenelse{boolean{Summaries-showEffortPartnerWPs}}{begin{table} caption{Summary of all effort over all partners and workpackages} label{tab:summaryEffortperWP} begin{center} begin{tabular}{l%{'c'*len(allWPDicts)%}r} toprule Partner & %{ ' & '.join(['WP,%s' % x['Number'] for x in allWPDicts]) %} & Total \ midrule ${summaryEffortRows} \ midrule Total & %{ ' & '.join([x['wpeffort'] for x in allWPDicts]) %} & %{ str(sum([int(x['wpeffort']) for x in allWPDicts])) %} \ bottomrule end{tabular} end{center} end{table} }{}

dict
Default: expanded

file
Default: True

dir
Default: tables

6.9.36 effortPerTaskRows

template
Default: ${taskId} & ${Name} & {cellcolor[gray]{0.9} %{ str(sum([int(e['resources']) for e in allEfforts if e['task'] == '${Label}'])) %} } & %{ ' & '.join([((r'textbf{%s}' % e['resources']) if '${Leadpartner}' == p['Shortname'] else (e['resources'] if int(e['resources']) > 0 else (r'{textcolor{%s}0}' % config.get('WPTables','colorInactivePartner')))) for p in partnerList for e in [ee for ee in allEfforts if ee['task'] == '${Label}' and ee['partner'] == p['Shortname']]]) %}

list = [t for t in allTasks if t['Main']=
Default: 'True']

groupby
Default: wp

joiner
Default: \ hline n

6.9.37 effortPerTaskRowsWP

template
Default: ${effortPerTaskRows_${Number}} \ hline rowcolor[gray]{0.9} multicolumn{2}{ll}{cellcolor[gray]{0.9} textbf{WP ${Number}:}} & %{ str(sum(${partnereffort}.values())) %} & %{ ' & '.join([r'textbf{%s}' % str(${partnereffort}[p['Shortname']]) if p['Shortname']=='${Leadership}' else str(${partnereffort}[p['Shortname']]) for p in partnerList]) %}

list
Default: allWPDicts

joiner
Default: \ hline hline n

dict
Default: expanded

6.9.38 effortHeader

template
Default: hline Task & Task name & Total PM & %{ ' & '.join([p['Shortname'] for p in partnerList]) %} \ hline

dict
Default: expanded

6.9.39 effortSum

template
Default: rowcolor[gray]{0.8} multicolumn{2}{ll}{cellcolor[gray]{0.8} textbf{Project total: }} & textbf{totalPM} & %{ ' & '.join([r'textbf{%s}' % str(sum([int(e['resources']) for e in allEfforts if e['partner'] == p['Shortname']])) for p in partnerList]) %}

dict
Default: expanded

6.9.40 effortPerTaskTableShort

template
Default: ifthenelse{boolean{LaTeX-effortTableLandscape}}{begin{landscape}}{} begin{table} caption{Effort per tasks and partners for entire project (in personmonths)} label{tab:effortPerTasks} begin{center} small begin{tabular}{c|p{0.15textwidth}|c%{‘ |c’ * len(partnerList)%}} ${effortHeader} ${effortPerTaskRowsWP} \ hline hline ${effortSum} \ hline hline end{tabular} end{center} end{table} ifthenelse{boolean{LaTeX-effortTableLandscape}}{end{landscape}}{}

dict
Default: expanded

6.9.41 effortPerTaskTableMultipage

template
Default: ifthenelse{boolean{LaTeX-effortTableLandscape}}{begin{landscape}}{} topcaption{Effort per tasks and partners for entire project (in personmonths)} label{tab:effortPerTasks} tablefirsthead{ ${effortHeader} } tablehead{ toprule multicolumn{%{str(3+len(partnerList))%}}{r}{emph{Table~ref{tab:effortPerTasks} continues from previous page}} \ ${effortHeader}} tabletail{bottomrule multicolumn{%{str(3+len(partnerList))%}}{r}{emph{Table~ref{tab:effortPerTasks} continues on next page}} \ bottomrule} tablelasttail{ multicolumn{%{str(3+len(partnerList))%}}{r}{emph{Table~ref{tab:effortPerTasks} ends}} \ bottomrule} begin{center} begin{mpxtabular}{c|p{0.15textwidth}|c%{‘ |c’ * len(partnerList)%}} ${effortPerTaskRowsWP} \ hline hline ${effortSum} \ hline hline end{mpxtabular} end{center} ifthenelse{boolean{LaTeX-effortTableLandscape}}{end{landscape}}{}

dict
Default: expanded

6.9.42 effortPerTaskTable

template
Default: ifthenelse{boolean{Summaries-showEffortPartnerTasks}} {ifthenelse{boolean{LaTeX-useMultipageEffortTable}} {${effortPerTaskTableMultipage}} {${effortPerTaskTableShort}} }{}

dict
Default: expanded

file
Default: True

dir
Default: tables

6.9.43 piePMsPartners

pie charts a pie chart showing person months distributed over partners

template
Default: ifthenelse{boolean{Summaries-piePMsPartners}}{ begin{figure}[htbp] centering begin{tikzpicture} pie[scale font]{ %{ utils.roundPie ([(x[’Shortname’], sum([int(e[’resources’]) for e in allEfforts if e[’partner’]==x[’Shortname’]])) for x in partnerList]) %}} end{tikzpicture} caption{Distribution of person months over partners (in percent)} label{fig:pie:pm:partner} end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

6.9.44 piePMsWPs

template
Default: ifthenelse{boolean{Summaries-piePMsWPs}}{ begin{figure}[htbp] centering begin{tikzpicture} pie[scale font]{ %{ utils.roundPie ([('WP,%s: %s'%(x['Number'], x['Shortname']), sum(x['partnereffort'].values())) for x in allWPDicts]) %}} end{tikzpicture} caption{Distribution of person months over work packages (in percent)} label{fig:pie:pm:wp} end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

6.9.45 piePMsNations

pie chart over person months assigned to different nations a little bit more complicated: we need to pull out the (Nation/Effort) pairs. those we get from allEfforts, where we look up the nation in the partnerList that leaves us with many entries in the list with the same nation we add up those efforts by a mapReduce operation to which we pass a suitable reduce function: adding up two values

template
Default: ifthenelse{boolean{Summaries-piePMsNations}}{ begin{figure}[htbp] centering begin{tikzpicture} pie[scale font]{ %{ utils.roundPie(utils.mapReduce ([(utils.searchListOfDicts(partnerList, 'Shortname', e['partner'], 'Nation'), int(e['resources'])) for e in allEfforts], lambda a,b: a+b)) %}} end{tikzpicture} caption{Distribution of person months over nations (in percent)} label{fig:pie:pm:nations} end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

6.9.46 piePMsPartnerTypes

same thing that worked for the nation pie charts works for the partner type pie charts as well

template
Default: ifthenelse{boolean{Summaries-piePMsPartnerTypes}}{ begin{figure}[htbp] centering begin{tikzpicture} pie[scale font]{ %{ utils.roundPie(utils.mapReduce ([(utils.searchListOfDicts(partnerList, 'Shortname', e['partner'], 'Type'), int(e['resources'])) for e in allEfforts], lambda a,b: a+b)) %}} end{tikzpicture} caption{Distribution of partner types over nations (in percent)} label{fig:pie:pm:partnertype} end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX