PRAGMA POD

Example GUI

1 Introduction

A regular ConTeXt workflow is batch oriented. You open a file, edit content, save the file, and ask TeX to process that file. After that you view the result. Usually, editors provide a one-key shortcut for saving, processing and opening the viewer. When you use ConTeXt, you use TeXexec to handle the dirty details of processing the document.

Once you get accustomed to this kind of workflow, and if your documents fit into batch processing, the workflow is not that inconvenient. On the other hand, some tasks can be rather boring, for instance:

- postprocessing documents, like imposing pages
- making letters, envelops, stickers, and alike
- converting graphics in more convenient formats
- testing layouts and playing with ConTeXt features
- produce listings and documentaion files

Some of those tasks, like making booklets, are incorporated in TeXexec. Over the years, more functionality was added to this script, a bit to the extent that part of its functionality becomes fuzzy.

Another development has been that ConTeXt is used in (for instance xml) workflows where the typesetting engine is acting in the background and user must control some aspects of the layout, without the need to bother about details.

The eXaMpLe framework tries to fill in this gap in user interfacing. The task is done by three programs that deal with communication (exampler), processing (examplex) and watching folders (exampleq).

These programs are discussed in more detail in other documents, and their activity is mostly hidden for the ConTeXt user.

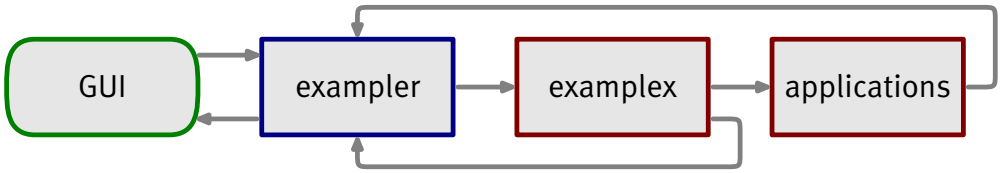

Figure 1 A rough scheme of the user interface.

The (graphical) user interface (currently) comes as pdf document. This document is opened in the full Acrobat version or in a web browser using the Acrobat Reader plugin.

This document communicates with exampler by means of form data in xfdf format. When recognized as valid, exampler will launch examplex, which in turn will control the end-application.[1]

[1] Technically exampler can also control end-applications directly, but this is only done in a few cases.

The exact action is determined by small and simple scripts, in most cases a TEXEXEC run. In this workflow all communication takes place in XML conforming the data definition in `example.rng`.

2 Installation

Installing this lightweight version of this framework is not that complicated, given that you already have CONTEXT running.

- download the archive `cont-exa.zip`
- unpack this archive in your (local) `texmf` tree
- update the file database with `mktexlsr`

The (PDF) interface files can be found in:

```
texroot/texmf(-local)/context/examplap/gui
```

You can open these files in ACROBAT (full version) or in your web browser to get an impression on what they provide you. Clicking on the process button will return an error message. This is because we have to start up EXAMPLER.

```
ruby exampler --continue
```

If this fails, or when processing fails, you may try to add a setuppath specifier:

```
ruby exampler --continue --setuppath=....
```

Here you need to specify the path where `example.exa` can be found, which by default is:

```
texroot/texmf(-local)/context/examplap/scripts
```

The EXAMPLER program listens to the local TCP/IP address `127.0.0.1` better known as local host, on port `8061` in which you can recognize the numerical part of PRAGMA ADE's postal zip code.

You can best launch a dedicated command shell for this, because then you can follow progress. As soon as you hit the process button in the PDF document, you should see some activity.

3 Debugging

If it does not work, you're in trouble, and you need to check the following:

- Is RUBY installed, you can test this by running one of the scripts.
- Does your TEX tree conform to the TDS standard, which is normally the case when you use one of the precooked distributions.

Unless you deeve into the internals of the scripts and the framework, you can try the following:

```
ruby exampler.rb --debug --show
```

You will get a lot of output which you can pipe into `more`, `less`, `list` or another viewer. Watch the environment variables and how they expand. If things are right you should recognize some of the paths reported there.

All scripts get their initialization information from the file `example.exa`, which refers to three other initialization scripts:

- `basic.exa`: the default settings
- `paths.exa`: local path settings
- `local.exa`: more local settings

The last two files are not in the distribution, but can be added by yourself.

4 Colofon

Author	Hans Hagen
Version	December 17, 2002
Copyright	PRAGMA POD, Hasselt NL
Internet	www.pragma-pod.com
Email	j.hagen@xs4all.nl