This Way

ConTEXt magazine #1101 MkIV
July 2011

Project Structure

Hans Hagen
PRAGMA ADE

A regular document has a simple structure. When we talk about structure here, we only refer to the overall document structure.

```
% style specification

\starttext
  % the document content
\stoptext
```

For practical reasons we delay initial font loading till the first `\starttext` so that one can overload the defaults. This means that when no bodyfont is specified, and is not given, there will be hardly any visible output.

An example of a more elaborate structure is the following:

```
\environment environment-1
\environment environment-2

\startproduct product-1

  \component component-1.tex
  \component component-2.mkiv
  \component component-3.cld

  \component component-1
  \component component-2

\stopproduct
```

Here we have a specific product, made up out of components and using a few environment files that specify the style. By default we assume tex files, but you can be specific and use known suffixes. A less abstract example is the following:

```
\environment my-fonts
\environment my-style
\environment my-abbreviations
\environment my-urls

\startproduct manual

  \component titlepage
  \component contents

  \component chapter-1
  \component chapter-2
  \component chapter-3

  \component index

\stopproduct
```

You can process components and products independently but be aware that you won't get cross document (or chapter) references then.

There is one more level: projects.

```
\environment my-fonts
\environment my-style
\environment my-abbreviations
\environment my-urls

\startproject documentation

  \product manual
  \product faqs

\stopproject
```

This means that we can also define the manual as follows:

```
\project documentation

\startproduct manual

  \component titlepage
  \component contents

  \component chapter-1
  \component chapter-2
  \component chapter-3

  \component index

\stopproduct
```

Environments are only loaded once and when you run a component or product that refers to environments or when environments are picked up from an encapsulating structure you need to be aware of the order of loading.

The names given after the start command are not that important but the names after the simple commands refer to filenames, so in the next case there need to be a file called `index.tex`:

```
\component index
```

Equally valid is:

```
\component[index]
```

Subpaths are also permitted:

```
\component manual/index
```

The meaning of the mentioned commands is not frozen but adapts itself to the current situation. A file can be processed many times, only once or never. The following table shows what will happen when:

	\component	\environment	\product	\project
\start .. \stopcomponent	many	once	none	once
\start .. \stopenvironment	none	once	none	none
\start .. \stopproduct	many	once	many	once
\start .. \stopproject	none	once	none	none
\start .. \stoptext	many	once	many	once

When you load an environment or component, you can specify it to be a Lua file by using the `lua` or `cld` suffix. In that case the file will be loaded in the right way. From the table you can deduce that the following is also valid:

```
\environment mystyle

\starttext
  % the content
\stoptext
```

combined with:

```
\startenvironment mystyle
  % the definitions
\stopenvironment
```

This is about the simplest structure that you can use that still gives a bit of abstraction.

In addition to files in a project structure, you can load predefined modules.

```
\usemodule[mathml]
```

or more specific:

```
\usemodule[x][mathml]
```

Which limits the lookup to the `x` namespace. The first match quits the search and the order of lookups is: `mkvi`, `mkiv`, `tex`, `cld`, `lua`. It follows that modules can be Lua files.

When you use structure in the files you will find an overview in the log file. This looks as follows:

```
system          > structure > start used structure

used structure  > text: product-1
used structure  >   environment: environment-1
used structure  >   environment: environment-2
used structure  >   product: product-1
```

```
used structure  >     component: component-1
used structure  >     component: component-2
used structure  >     component: component-1
used structure  >     component: component-2

system          > structure > stop used structure
```

Some basic logging on the console can be enabled with:

```
\enabletrackers[system.jobfiles]
```

A new command pair is the following:

```
\startdocument[settings]
    structured content
\stopdocument
```

The settings are key/value pairs and the values can be retrieved using:

```
\documentvariable{key}
```

You can set `before` and `after` parameters and by default these are set up as follows:

```
\setvariables
  [document]
  [before=\directsetup{document:start},
   after=\directsetup{document:stop}]
```

You can for instance define these setups to generate a title page (using document variables) and a colophon page. In the future more functionality might be added.

source code of this document

```
% language=uk

% author    : Hans Hagen
% copyright : PRAGMA ADE & ConTeXt Development Team
% license   : Creative Commons Attribution ShareAlike 4.0 International
% reference : pragma-ade.nl | contextgarden.net | texlive (related) distributions
% origin    : the ConTeXt distribution
%
% comment   : Because this manual is distributed with TeX distributions it comes with a rather
%             liberal license. We try to adapt these documents to upgrades in the (sub)systems
%             that they describe. Using parts of the content otherwise can therefore conflict
%             with existing functionality and we cannot be held responsible for that. Many of
%             the manuals contain characteristic graphics and personal notes or examples that
%             make no sense when used out-of-context.

\usemodule[mag-01,abr-02,job-01]

\startbuffer[abstract]
    For a long time already \CONTEXT\ provides a way to organize your document(s)
    in a structure that permits processing of components. This mechanism has been
    upgraded a bit in \MKIV\ and here we will summarize the status quo.
\stopbuffer

\startdocument
  [title={Project Structure},
   author=Hans Hagen,
   affiliation=PRAGMA ADE,
   date=July 2011,
   number=1101 \MKIV]

A regular document has a simple structure. When we talk about structure here, we
only refer to the overall document structure.

\startscite[tex]
% style specification

\starttext
  % the document content
\stoptext
\stopscite

For practical reasons we delay initial font loading till the first \type
{\starttext} so that one can overload the defaults. This means that when no
bodyfont is specified, and {\starttext} is not given, there will be hardly any
visible output.

An example of a more elaborate structure is the following:

% \enabletrackers[context.trace]

\startscite[tex]
\environment environment-1
\environment environment-2

\startproduct product-1

  \component component-1.tex
  \component component-2.mkiv
  \component component-3.cld

  \component component-1
```

source code of this document

```
  \component component-2

\stopproduct
\stopscite

Here we have a specific product, made up out of components and using a few
environment files that specify the style. By default we assume tex files, but you
can be specific and use known suffixes. A less abstract example is the following:

\startscite[tex]
\environment my-fonts
\environment my-style
\environment my-abbreviations
\environment my-urls

\startproduct manual

  \component titlepage
  \component contents

  \component chapter-1
  \component chapter-2
  \component chapter-3

  \component index

\stopproduct
\stopscite

You can process components and products independently but be aware that you won't
get cross document (or chapter) references then.

There is one more level: projects.

\startscite[tex]
\environment my-fonts
\environment my-style
\environment my-abbreviations
\environment my-urls

\startproject documentation

  \product manual
  \product faqs

\stopproject
\stopscite

This means that we can also define the manual as follows:

\startscite[tex]
\project documentation

\startproduct manual

  \component titlepage
  \component contents

  \component chapter-1
  \component chapter-2
  \component chapter-3

  \component index
```

source code of this document

```
\stopproduct
\stopscite
```

Environments are only loaded once and when you run a component or product that refers to environments or when environments are picked up from an encapsulating structure you need to be aware of the order of loading.

The names given after the start command are not that important but the names after the simple commands refer to filenames, so in the next case there need to be a file called \type {index.tex}:

```
\startscite[tex]
\component index
\stopscite
```

Equally valid is:

```
\startscite[tex]
\component[index]
\stopscite
```

Subpaths are also permitted:

```
\startscite[tex]
\component manual/index
\stopscite
```

The meaning of the mentioned commands is not frozen but adapts itself to the current situation. A file can be processed many times, only once or never. The following table shows what will happen when:

```
\ctxlua{moduledata.jobs.showprocessors()}
```

When you load an environment or component, you can specify it to be a \LUA\ file by using the \type {lua} or \type {cld} suffix. In that case the file will be loaded in the right way. From the table you can deduce that the following is also valid:

```
\startscite[tex]
\environment mystyle

\starttext
  % the content
\stoptext
\stopscite
```

combined with:

```
\startscite[tex]
\startenvironment mystyle
  % the definitions
\stopenvironment
\stopscite
```

This is about the simplest structure that you can use that still gives a bit of abstraction.

In addition to files in a project structure, you can load predefined modules.

```
\startscite[tex]
\usemodule[mathml]
\stopscite
```

source code of this document

or more specific:

```
\startscite[tex]
\usemodule[x][mathml]
\stopscite
```

Which limits the lookup to the \type {x} namespace. The first match quits the search and the order of lookups is: \type {mkvi}, \type {mkiv}, \type {tex}, \type {cld}, \type {lua}. It follows that modules can be \LUA\ files.

When you use structure in the files you will find an overview in the log file. This looks as follows:

```
\starttyping
system           > structure > start used structure

used structure   > text: product-1
used structure   >   environment: environment-1
used structure   >   environment: environment-2
used structure   >   product: product-1
used structure   >     component: component-1
used structure   >     component: component-2
used structure   >     component: component-1
used structure   >     component: component-2

system           > structure > stop used structure
\stoptyping
```

Some basic logging on the console can be enabled with:

```
\startscite[tex]
\enabletrackers[system.jobfiles]
\stopscite
```

A new command pair is the following:

```
\starttyping
\startdocument[settings]
    structured content
\stopdocument
\stoptyping
```

The settings are key|/|value pairs and the values can be retrieved using:

```
\starttyping
\documentvariable{key}
\stoptyping
```

You can set \type {before} and \type {after} parameters and by default these are set up as follows:

```
\starttyping
\setvariables
  [document]
  [before=\directsetup{document:start},
   after=\directsetup{document:stop}]
\stoptyping
```

You can for instance define these setups to generate a title page (using document variables) and a colophon page. In the future more functionality might be added.

```
\stopdocument
```