```
        <optional> <ref name="contact.telefax"/> </optional>
        <optional> <ref name="contact.email"/> </optional>
        <optional> <ref name="contact.address"/> </optional>
        <optional> <ref name="contact.information"/>  </optional>
      </interleave>
      <attribute name="label"/>
    </element>
</define>
```

A contact group is a collection of references to contacts:

```
<define name="contacts.contactgroup">
  <element name="contactgroup">
    <zeroOrMore>
      <ref name="contactgroup.member"/>
    </zeroOrMore>
    <attribute name="label"/>
  </element>
</define>

<define name="contactgroup.member">
  <element name="member">
    <text/>
  </element>
</define>
```

A contact file is just a reference to a to be included file:

```
<define name="contacts.contactfile">
  <element name="contactfile">
    <text/>
  </element>
</define>
```

The fields that make up a contact are normally just text. The address needs to be preformatted with <p> elements (more details can be found in the file x-corres.rng).

A simple database will look like this:

```
<contacts>

   <contact label=' hagen.j' >
     <initials>J.</initials>
     <formalname>Hagen</formalname>
     <informalname>Hans</informalname>
```

```
    <title>Drs.</title>
    <prefix>Heer</prefix>
    <address>
      <p>J. Hagen</p>
      <p>PRAGMA ADE</p>
      <p>Ridderstraat 27</p>
      <p>8061GH Hasselt NL</p>
    </address>
  </contact>

  <contact label='otten.a.f'>
    <initials>J.</initials>
    <formalname>Otten</formalname>
    <informalname>Ton</informalname>
    <title>Drs.</title>
    <prefix>Heer</prefix>
    <address>
      <p>A.F. Otten</p>
      <p>PRAGMA ADE</p>
      <p>Ridderstraat 27</p>
      <p>8061GH Hasselt NL</p>
    </address>
  </contact>

  <contact label='marle.k.van'>
    <initials>K. van</initials>
    <formalname>Marle</formalname>
    <informalname>Kees</informalname>
    <title>Ir.</title>
    <prefix>Heer van</prefix>
    <address>
      <p>K. van Marle</p>
      <p>PRAGMA POD</p>
      <p>Ridderstraat 27</p>
      <p>8061GH Hasselt NL</p>
    </address>
  </contact>

  <contactgroup label='pragma-ade'>
    <member>hagen.j</member>
    <member>otten.a.f</member>
  </contactgroup>
```

```
  <contactgroup label=' pragma-pod' >
    <member>hagen.j</member>
    <member>otten.a.f</member>
    <member>marle.k.van</member>
  </contactgroup>

</contacts>
```

The labels plays an important role later on, when we will filter data from the database, so they should be unique and consistent.

3 The Interface

You can use the database from within ConTEXt after you've loaded the file x-corres.tex:

```
\usemodule[corres]
```

The macro that deals with processing is:

```
\XMLprocesscontacts[filename]
```

The argument is optional, in which case the selection needs to be set up with the following command:

```
\setvariables
  [contacts]
  [selection=,
  file=]
```

This command is also used for specifying the data file.

The fields of a contact are stored and can be used in the typesetting process. This process is hooked into the xml engine and style by the following setup (here we show the default definition):

```
\startsetups[contact:handle]

  \XMLflush{address}

\stopsetups
```

And so we have arrived at the main process, which may look like something:

```
\setvariables [contacts] [selection=pragma-ade,file=pragma.xml]

\startsetups [contact:handle]
  \XMLflush{initials} \XMLflush{formalname} \endgraf
\stopsetups

\XMLprocesscontacts
```

A selection can be a comma separated list of labels and/or groups. In a first pass, the list is resolved into a list of valid contacts, while the second pass typesets the result based on the handle.

4 The Style

In the previous chapters we presented the xml database and its interface, here will will spend some words on typesetting a (series of) letter(s). The style is quite independent of the database but uses the same fields and is called m-letter.tex.

It does not make sense to discuss all the details of the style because you need to look into the file anyway, so we stick to discussing the methodology used. There is an eXaMpLe interface that permits you to play with the settings.

The layout setup is rather simple and uses standard ConTEXt setup commands. The individual components that make up the letter are defined in so called setups, like:

```
\startsetups [letter:place:address]
   ...
\stopsetups
```

We use the layering mechanism to collect and place the components. Inside such a setup, variables are used to control the behaviour. For instance, the location of the address on the page grid is determined by:

```
\setvariables
   [letter:address]
   [line=3]
```

The address itself again a setup:

```
\setsetups[letter:address]
   ...
\stopsetups
```

and although a definition like:

```
\startsetups[letter:address]
   A Fancy Name \endgraf
   A Nice Address \endgraf
   The Place To Go
\stopsetups
```

is okay, it makes more sense to use variables again, like in:

```
\startsetups[letter:address]
   \def\\{\endgraf}\getvariable{letter:data}{address}
\stopsetups
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/886142243002010033