User Manual of reppy

14.1.2005

Table of contents

Preface

reppy is a python program to create PDF-reports from a SQL-result-set or a CSV-file. The look of the report is declared in an XML-file. This XML-file(called template file) with the file-extension '.xrt' can be created with the included program xtred (the call of this program is 'python xtred.py' or 'xtred.sh'

Usage as stand-alone program

Reppy is written in Python with use of reportlab and database adapters for Mysql,Postgresql and CSV-files(can be created with export of MS-Excel).
After the installation you could create an template XML-file 'test.xtr', to describe the layout of a report. The name of this template-file should have the extension '.XTR'.

Then you can test the program reppy with:

python reppy.py -ftest.xtr

There are other optional arguments of reppy:

-h<hostname> where the database ist stored
-u<username> ⎫
-p<password> ⎬ of used database
-d<database-name> ⎭
-o<pdf-filename>
-v<folder-name> where the templates are stored

The name of the created PDF-file is stored in the template file.
The template file consists of 4 parts: SOURCES,PARAMETERS,GROUPS,LAYOUT

The datasource is declared in SOURCES, and can be an SQL-statement or a table (with an optional mastertable). At the moment there is only one datasource possible.

The PARAMETERS give the possibility to declare text of functions, constants or input from keyboard at run-time.

With GROUPS you can declare, what should be printed, if the group-value changes in the datasource.

The last part is the LAYOUT, where the look of the PDF-file is declared.

A simple example of an template file could be:

```
<?xml version=“1.0“ encoding=“ascii“ ?>
<REPORT filename=“report1.pdf“ name=“example1“>
        <SOURCES>
                <DATASOURCE  database=“test“ host=“localhost“ name=“datasource1“
                                user=“tester“ password=“passwd“ typ=“mysql“
                                sql=“select * from tabelle“>
        </SOURCES>
        <LAYOUT pagesize=“A4“>
                <SECTION typ=“detail“ source=“datasource1“ name=“detailrow“>
                        <FIELD name=“field1“ column=“id“ x=“1.0cm“ y=“0.0cm“ />
                        <FIELD name=“field2“ column=“name“ x=“2.0cm“ y=“0.0cm“
                                fontsize=“12“  font=“Times-Roman“ />
                        <FIELD name=“field3“ column=“city“ x=“12.0cm“ y=“0.0cm“ />
                </SECTION>
        </LAYOUT>
</REPORT>
```

In this report a file report1.pdf is created on a page of size A4 with a line for every record of the table „tabelle“ in the mysql-database „test“.

To ease the development of an template file, there exists a tool „xtred“, to d evelop visually a template file and to test the created report definition.

Usage as function in a python program

You can use reppy as a function in your python program. The PYTHONPATH variable has to point to the two files repPlatypus.py and repParser.py and reportlab and the needed databasedrivers have to be installed. With this, you can call:

```
import repParser
db = {'host':“...“, 'user':“...“, 'password':“...“, 'database':“...“, 'filename':“...“}
folder = ''
templatename = '<templatename>'
repParser.create_rep(templatename, db, folder)
```

templatename is the name of the XTR-template-file. Folder is the name of the directory (absolute or relative), where the template file is stored. The dictionary db may be empty, if all is declared in the templatefile.

Example1:

```
import repParser
db = {'host':'localhost', 'user':“schorg“, 'password':“secret“, 'database':“test“, 'filename':“report.pdf“}
folder = ''
templatename = 'report.xtr'
repParser.create_rep(templatename, db, folder)
```

Example2: (if all data is declared in report.xtr)

```
import repParser
db = {'host':'', 'user':““, 'password':““, 'database':““, 'filename':““}
folder = ''
templatename = 'report.xtr'
repParser.create_rep(templatename, db, folder)
```

Description of the tags in the template file.

1. REPORT

The REPORT-tag is the outermost tag with the optional attributes:

filename:	name of the created PDF-file(default value is „output.pdf“)
author:	name of the author
version:	version of this report(default is „1.0“)
description:	short description of this report
name:	name of this report

2. PARAMETERS:

The PARAMETERS-tag includes the parameters. The purpose of a Parameter is, to create a text to be inserted into a field of a section.

2.1 PARAMETER:

The PARAMETER-tag declares a text-string, which is created dynamically at run-time. The origin of an parameter can be a python function, a sum, input of the command line, fixed text or input during run-time of reppy. The use of an parameter in the FIELD-tag is wished, if you write in the text-attribute the parameter-name with a beginning underscore. If the name of the parameter is 'para1', then write in the text-attribute of an field-tag the text '_para1', to insert the text, created in the parameter.

Needed attributes:

name:	name of parameter
default:	default value
origin:	declares, what type of parameter this is. possible values are: command: the value will be taken from an argument of the commandline (eg: --opt=value) function: the value is teh result of a python function, which is written in the attribute 'function' default: the value will be taken from the default-attribute. input: the value is taken from the system input during run of the program sum: The value is the sum of a column, given in the column-attribute(see later explanation) key: in the default-attribute is the declaration of a key:value combination
function:	python function
column:	database column for sum
group:	database group-column for origin=sum

Optional attributes:

description: description of the parameter
typ: type of the result of this parameter(real|int|string)

The Parameter-tag is very efficient. With it, you can generate text dependent on the run-time situation. The use is controlled by the attribute 'origin'.

If 'origin' is set to '*default*', the text is simply taken from teh default-attribute. This can be used for constants.

If 'origin' is set to '*function*', the content of the function-attributes is given to the python interpreter and the resulting value is inserted as the parameter-value. There are predefined python-functions, which may be used:

db_col(<column-name>) gives the actual value of this database-column
pagenr() gives the actual page number
date() gives the actual date in the form: dd. Mmm YYYY
time() gives the actual time in the form: hh:mm:ss

Some examples may show, how to use this functions:

1) You have a database-table with the columns 'surname' and 'name' and want to print the content of this columns in the form name and first character of surname with a dot. You could write in the function-attribute:
   db_col(name) + ' ' + db_col(surname) + '.'
2) You want to write the pagenr in the page-header:
   'Pagenr. ' + pagenr()
3) You want to write the date of the actual day:
   'Date: ' + date()
4) You want to write the actual time of printing:
   'Time: ' + time()

If 'origin' is set to '*sum*', the content of the attributes 'column' and 'group' determine the result. If 'column' and 'group' are both empty, than the result is the number of records in the result-set of the datasource. When 'column' shows a column-name of the datasource and 'group' is empty, a total sum of all values of the given column is returned.
If 'column' and 'group' contain both names of the datasource, then the momentary sum of all values of the 'column' are resulted, and when the value of the 'group'- column in the table changes, this sum is set to 0. This is useful for group-footers.
If 'column' is empty and 'group' holds the name of a column, the returned value represents the number of records with the same group-value. This value is set to 0, when the value of the 'group'-column changes.

Some examples may show, how to use this parameter-type:

<parameter name= 'recordnumber' origin='sum' /> gives the total record number in datasource
<parameter name='totalsum' origin='sum' column='account' /> gives total sum of the column 'account'
<parameter name='groupsum' origin='sum' column='account' group='persid' typ='real'/>

gives sum of the column 'account' for every group
<parameter name='grouprecords' origin='sum' group='persid' />
gives number of records with the same group value

If 'origin' is set to *'command'*, the value of this parameter is taken from the options of the command-line. For the parameter <parameter name='incomm' origin='command' type='string' />
the reppy-command should be: python reppy.py -ftest.xtr - - incomm='input-value'

If 'origin' is set to *'input'*, the operator of the program is asked for the value of this parameter at run-time.

If 'origin' is set to '*key*', in the 'default'-attribute a table for translation is declared in the form:

column:key1,value1|column:key2,value2|...

Example: If you want, that all possible values in the column 'department' should be translated in speeking words, you could define this parameter:

<parameter name='transtable' origin=key'
default='department:1,store|department:2,marketing|department:3,sale' />

This table will be used, when the 'keyname'-attribute of the datasource-tag is set to <_parameter-name> and the 'key'-attribute of the Field-tag is set to '1'. If the table is too long, to be declared in the parameter, you can create an own table for this key in the database and set the 'keyname'-attribute to the name of the table. This table must be declared with the columns: 'attribute','origin','value'.

3. SOURCES:

The SOURCES-tag includes the datasources (only one datasource is yet allowed)

3.1 DATASOURCE:

The datasource describes the source for all database-based data's. There are two distinct combinations of attributes:

A SQL-command selects all records or a simple table (master-table relation) gives the selection of records.
Needed attributes are:

- name: name of datasource
- typ: type of database (mysql, postgres,CSV, default is mysql)
- table or sql: name of a table or SQL-statement

Optional attributes:

- host: name of host, where database ist stored. Empty, if typ=CSV
- user: name of user, which is granted to access the database
- password: password of user, to access the database
- database: name of database or name of folder, if typ=CSV
- keyname: name of a table, which holds the mapping values (for usage see later)

If table is not empty(has a value), other attributes could be used:

master:	name of a master-table – there are two columns of these tables, which correspond: masterlink and detaillink
masterlink	name of column in mastertable which corresponds to column detaillllink of table
detaillllink:	name of column in table, which correspond to masterlink
sort:	name of column in table, on which the result should be sorted

Examples:
Example1: The table „addresses“ of the mysql-database „schorg“ is used as source of records for the report

```
<sources>
        <datasource name="datasource1"  typ="mysql" database="schorg"  host="localhost"  user="schorg"
                password="schorg"  table="addresses" >
        </datasource>
</sources>
```

Example2: The table „personal“ of the mysql-database „schorg“ is used as source of records linked to the master-table „jobs“ over the link personal.job_id = jobs.id. The result is sorted on the column „name“.

```
<sources>
        <datasource database="schorg" detaillink="job_id" host="localhost" master="jobs" masterlink="id"
         name="ds1" password="schorg" sort="name"  table="personal" typ="mysql" user="schorg" />
</sources>
```

Example3: You will get the same result as in example 2 , if you use the SQL-statement:
select * from personal, jobs where personal.job_id = jobs.id order by name

```
<sources>
        <datasource name="ds1"  typ="mysql" database="schorg"  host="localhost"  user="schorg"
                password="schorg"
                sql="select * from personal, jobs  where personal.job_id = jobs.id order by name"  />
<sources>
```

With the sql-attribute you can use all SQL-commands, which are allowed in the used database. If the database-type is CSV, you cannot use the sql-attribute!

4. GROUPS:

The GROUPS-tag includes the group-tags

4.1 GROUP:

The GROUP-tag declares the connection between the groupfooter-section, groupheader-section and the datasource-column, where the group-change triggers these sections. All attributes are needed.

name:	name of the group-tag
column:	name of the column, which triggers the group change.
header:	name of the group-header section
footer:	name of the group-footer section

Examples:

```
<groups>
        <group name='firstgroup' column='id' header='id_group_header' footer='id_group_footer' />
</goups>
```

5. LAYOUT:

The LAYOUT-tag describes the layout of the report. He contains SECTION-tags, which declare a part of the report. In the LAYOUT-tag the design of a whole page is described.

The optional attributes are:

pagesize: type of pagesize (default is A4. Possible values are A0..A6,B0..B6,LETTER, LEGAL, ELEVENSEVENTEEN)
landscape: orientation of page(0 .. portrait, 1 .. landscape)
leftmargin,rightmargin, bottommargin, topmargin: free place on this side of paper

Example:
The report should be printed in Format A4, Portrait and with the margins left=2cm, bottom=3cm, right=1, top=1 cm

```
<layout bottommargin="3.0cm" leftmargin="2.0cm" pagesize="A4" rightmargin="1.0cm" topmargin="01.0cm" landscape=0>
```

5.1 SECTION:

The SECTION-tag describes a part of the page. There a different types of SECTION's – a section for report header and footer, for page header and footer, for group header and footer for detail header and detail footer and for detail. The most important is detail. Only one pageheader, pagefooter, reportheader, reportfooter,detailheader, detailfooter and detail is possible for one report. But there may be more than one groupheader and/or groupfooter.
A SECTION contains Fields, Lines, Rectangles, Rounded Rectangles, Ellipses, Polygons and Pictures. These tags declare the structure of a Section. The order of the tags in the SECTION is not important.
The origin of the coordinate-system (0,0) is always in the left lower corner. The used measure-unit is cm (e.g. x='2.3cm') or points (e.g. x='203')

Needed attributes:

name: name of section
typ: type of section(„detail“, „detailheader“, „detailfooter“, „groupheader“, „groupfooter“, „pageheader“ , „pagefooter“, „reportheader“, „reportfooter“)
source: name of datasource, as given in SOURCES-tag

Optional attributes:

description: short description of this section
bgcolor: color of background in the RGB-form (255,255,255) is white, (0,0,0) is black
alternate: especially for detail section. The background-colour of the detail section alternates from the given backgroundcolor to white and vice versa. („0“ .. don't change colour, „1“ .. change colour)
table: especially for detail section(„0“ .. no lines drawn between columns, „1“ .. lines between columns)
newpage: „1“ = this section should be placed on an new page, default is „0“

Examples:

```
<section typ="detail" name="zeile" source="datasource1" alternate="1" bgcolor="(0,1,1)" tableform="1" >
<section name="gfooter2" source="datasource1" typ="groupfooter">
```

5.1.1 FIELD:

The FIELD-tag declares an item of a section. It represents a fixed string, a column of the datasource or a number. A text can be written in the attribute 'Text'. If the text begins with an underscore ('_'), the text is replaced with the value given from an Parameter-tag with this name.
If the column-attribute is not empty, the text, which will be printed is the value of a column in the table of the database. If the column-attribute begins with an underscore('_'), the value is taken from the old database record. This is valuable in a section with the type groupheader or groopfooter.

The datasource is given in the SECTION-tag.
Only one of the attributes text or column can be used at the same time.

The origin of all coordinates is the left low corner of the SECTION. The coordinates of all items in a SECTION depend on the same origin for this SECTION. SO it is possible, that a Section is only one line (all x-values are 0.0cm or similar) or a whole page, if the y-values reach from 0.0cm to e.g. 19.7cm.

Needed attributes:

name:	name of the field-tag
text or column:	see the explanation above

Optional attributes:

x:	x-coordinate in cm or inch(from 0.0cm to ... cm)
y:	y-coordinate in cm or inch(from 0.0cm to ..cm)
color:	RGB-values for the colour of the characters
font:	font-type (Courier,**Courier-Bold**,*Courier-Oblique*, Helvetica,**Helvetica-Bold**,***Helvetica-BoldOblique***,*Helvetica-Oblique* Symbol(Σψμβολ),**Times-Bold**, ***Times-BoldItalic***, *Times-Italic*,Times-Roman, ZapfDingbats)(default is Times-Roman)
fontsize:	Size of font from 7 points to ... (default is 12)
align:	Alignment („left“,“center“,“right“)(default is „left“)
length:	length of field in cm or inch
height:	height of field in cm or inch
typ:	type of this field („string“, „int“, „real“, „money“) (default is „string“)
key:	take column value as key and insert the value, found in the keyname-table (this table can be in the datasource or a parameter)

Examples:

```
<field name="city-column" color="(1,0,1)" column="city"  font=“Helvetica“  fontsize=“14“ height=“1.0cm“
      align=“left“ />
<field name="textfeld2" text="_user" length="2.6cm"/>
<field column="state" name="textfeld3" />
```

5.1.2 RECTANGLE:

The RECTANGLE-tag draws a rectangle in the given size in the PDF-file.

Needed Attributes:

name:	name of element
x:	x-coordinate of left lower corner(default 0,0cm)
y:	y-coordinate of lower left corner(default 0.0cm
width:	width of rectangle (default is 1.00cm)
height:	height of rectangle (default is 1.00cm)
linewidth:	width of the line in points
stroke:	RGB-Colour of line (default is black)
fill:	RGB-Colour of inner (default is white)

```
<rectangle name="rechteck"  x="1.0cm" y="0.0cm"  width="2.0cm" height="1.0cm"  linewidth="2"
                stroke="(10,200,30)" fill="(130,20,15)" />
```

5.1.3 ROUNDRECT:

The ROUNDRECT-tag draws a rectangle with rounded edges in the given size in the PDF-file.

Needed Attributes:

name:	name of element
x:	x-coordinate of left lower corner(default 0,0cm)
y:	y-coordinate of lower left corner(default 0.0cm
width:	width of rectangle (default is 1.00cm)
height:	height of rectangle (default is 1.00cm)
linewidth:	width of the line in points
stroke:	RGB-Colour of line (default is black)
fill:	RGB-Colour of inner (default is white)
radius:	radius of circle on edges in cm(default is 6)

```
<roundrect name="rechteck"  x="1.0cm" y="0.0cm"  width="2.0cm" height="1.0cm"  linewidth="2"
                stroke="(10,200,30)" fill="(130,20,15)" />
```

5.1.4 LINE:

The LINE-tag draws a line from point (x1,y1) to point (x2,y2) in the PDF-file.

Needed Attributes:

name:	name of linet
x1:	x-coordinate of point1(default 0,0cm)
y1:	y-coordinate of point1(default 0.0cm
x2:	x-coordinate of point2(default 0,0cm)
y2:	y-coordinate of point2(default 0.0cm
linewidth:	width of the line in points
stroke:	RGB-Colour of line (default is black)

Example:

```
<line name="line1"  x1="1.0cm" y1="0.0cm"  x2="2.0cm"  y2="1.0cm"  linewidth="2" stroke="(10,200,30)"  />
```

5.1.5 ELLIPSE:

The ELLIPSE-tag draws an ellipse in the given rectangle in the PDF-file.

Needed Attributes:

name:	name of element
x1:	x-coordinate of left lower corner(default 0,0cm)
y1:	y-coordinate of lower left corner(default 0.0cm

x2:	x-coordinate of upper right corner(default 0,0cm)
y2:	y-coordinate of upper right corner(default 0.0cm
linewidth:	width of the line in points
stroke:	RGB-Colour of line (default is black)
fill:	RGB-Colour of inner (default is white)

Example:

```
<ellipse name="ellipse"  x1="0.0cm" y1="0.0cm"  x2="2.0cm" y2="1.0cm"  linewidth="2"
        stroke="(10,200,30)" fill="(130,20,15)" />
```

5.1.6 POLYGON:

The POLYGON-tag draws an closed polygon with the given points in the PDF-file.

Needed Attributes:

name:	name of element
points:	pairs of (x,y)-coordinates(in cm or points)
linewidth:	width of the line in points
stroke:	RGB-Colour of line (default is black)
fill:	RGB-Colour of inner (default is transparent)

Example:

```
<polygon name="poly"  points="(0.0cm,0.0cm),(1.0cm,0.0cm),(0.5cm,0.5cm)"  linewidth="2"
        stroke="(10,200,30)" fill="(130,20,15)" />
```

5.1.7 FIXEDIMAGE:

The FIXEDIMAGE-tag draws an image in the given size in the PDF-file.

Needed Attributes:

name:	name of element
x:	x-coordinate of left lower corner(default 0,0cm)
y:	y-coordinate of lower left corner(default 0.0cm
width:	width of rectangle (default is 1.00cm)
height:	height of rectangle (default is 1.00cm)
filename:	name of the image-file

```
<fixedimage name="image1"  x="1.0cm" y="0.0cm"  width="2.0cm" height="1.0cm"
        filename="/home/test/image.jpg" />
```

APPENDIX A: Examples

For these examples, a CSV(Comma-Separated-Values)-based database is assumed.

We have the file numbers.csv with the columns: *id,beschreibung,quantity,pric*e

"id","beschreibung","quantity","price"

"1","Schraube M3","5","0.21"

"1","Schraube M5","5","0.23"

"1","Schraube M10","15","0.40"

"1","Schraube M13","7","0.43"

"1","Schraube M15","3","0.65"

"2","Hammer","105","5.23"

"2","Zange","5","10.23"

"3","Rechen","35","14.00"

"3","Schaufel","10","10.00"

"3","Besen","5","11.23"

The first line represents the names of the table-columns.

A second table, typen.csv, with the columns: *id, type,* can be used as master-table:

"id","type",“rabat“

"1","Screws",“10“

"2","Tools",“5“

"3","Utensils" ,“20“

As keyname table we can use the table keys.csv with the columns: *attribute, origin,value*

```
"attribute","origin","value"
"id","1","Screws"
"id","2","Tools"
"id","3","Utensils"
```

Example1: Simple list of numbers.csv

```
<?xml version="1.0" encoding="ascii"?>
<report filename="test.pdf">
  <sources>
    <datasource datasource0" sort="price"  table="numbers.csv" typ="CSV" />
  </sources>
  <layout>
    <section name="section0" source="datasource0" tableform="0" typ="detail">
      <field  column="id" font="Times-Roman" fontsize="12"  name="id"  x="0.0cm" y="0.0cm"/>
      <field  column="beschreibung" font="Times-Roman" fontsize="11" name="field1"  x="1.5cm" y="0.0cm"/>
      <field  column="quantity" font="Helvetica-Bold" fontsize="12" name="field2"  x="5.0cm" y="0.0cm"/>
      <field  column="price" font="Times-Bold" fontsize="12"  name="field3" rightpadding="0.0cm"  x="10.5cm" y="0.0cm"/>
    </section>
    <section  name="pagehead"  typ="pageheader">
      <field  font="Times-Bold" fontsize="12"  name="head_id"  x="0.0cm" y="0.2cm"/>
      <field align="left" bottompadding="0.0cm" color="(0,0,0)" font="Times-Bold" fontsize="12" key="0" leftpadding="0.0cm"
name="head_beschreibung" rightpadding="0.0cm" text="Description" toppadding="0.0cm" typ="string" x="1.5cm" y="0.2cm"/>
      <field  font="Times-Bold" fontsize="12"  name="head_quantity" text="Quantity"  x="5.0cm" y="0.2cm"/>
      <field  font="Times-Bold" fontsize="12"  name="head_price"  text="Price"  x="10.5cm" y="0.2cm"/>
      <line linewidth="2" name="line0" x1="0.0cm" y1="0.1cm" x2="12.0cm"  y2="0.1cm"/>
    </section>
  </layout>
</report>
```

ID	Description	Quantity	Price
1	Schraube M3	5	0.21
1	Schraube M5	5	0.23
1	Schraube M10	15	0.40
1	Schraube M13	7	0.43
1	Schraube M15	3	0.65
2	Hammer	105	5.23
2	Zange	5	10.23
2	Rechen	35	14.00
2	Schaufel	10	10.00
3	Besen	5	11.23

Example 2:

Create a report with of numbers.csv with a group change, when the id is changing. For the column 'id' use keys.csv, to translate e.g. '1' to 'Sax'. For every position multiply price by quantity and print it. With every group-change draw a line and print the sum of quantity and a sum of the products 'price' * 'quantity'. In the head of the page print the date of creation of the report and the number of the page.

Every detail-line should have a changed backgroundcolor and a group header indicates the meaning of the columns:

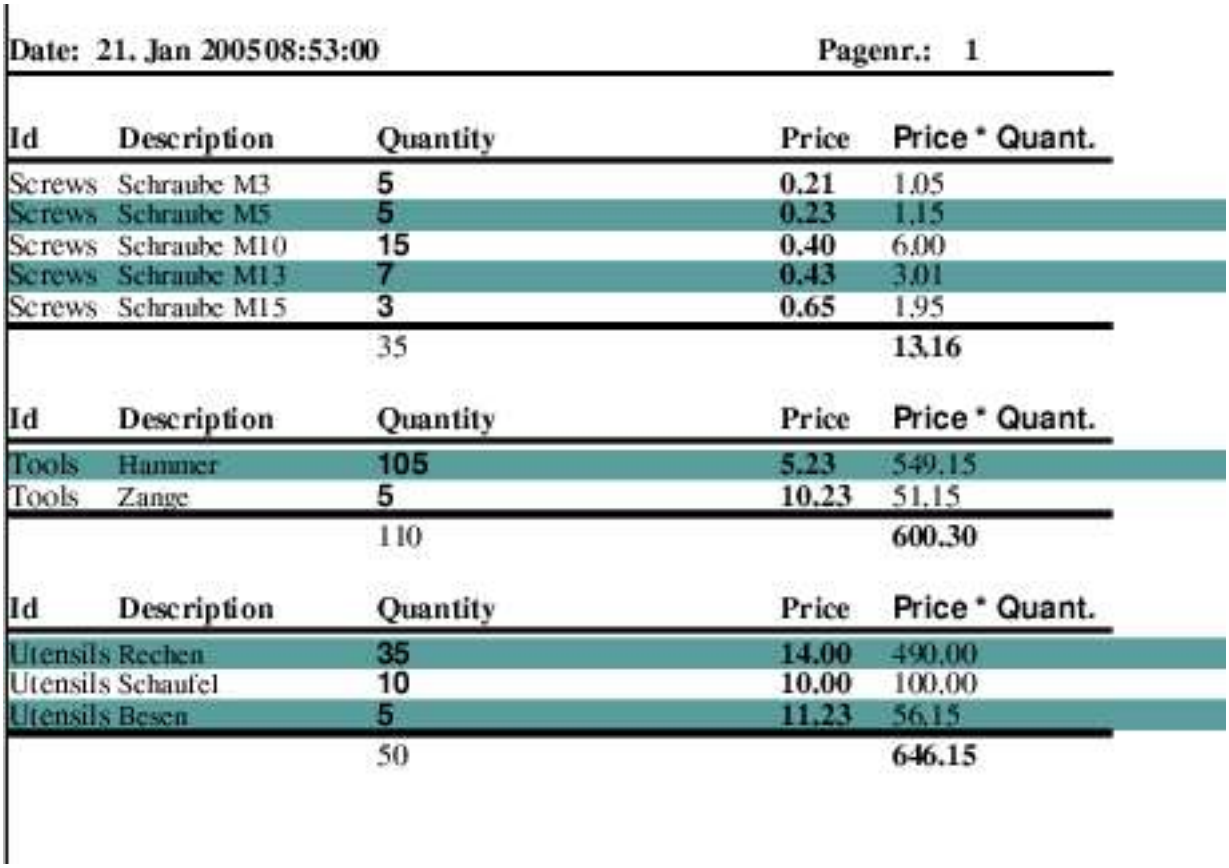
```
<?xml version="1.0" encoding="ascii"?>
<report encoding="ascii" filename="test.pdf" name="report">

 <parameters>
  <parameter name="value" function="float(db_col('price')) * float(db_col('quantity'))" origin="function" typ="real"/>
  <parameter name="quant_sum" column="quantity" group="group0"  origin="sum" typ="int"/>
  <parameter name="p*q_sum" column="_value" group="group0"  origin="sum" typ="real"/>
  <parameter name="date" function="date()"  origin="function" typ="string"/>
  <parameter name="time" function="time()"  origin="function" typ="string"/>
  <parameter name="pagenr" function="pagenr()"  origin="function" typ="string"/>
 </parameters>
 <sources>
  <datasource name="datasource0" keyname="keys.csv" sort="id" table="numbers.csv" typ="CSV" />
 </sources>
 <groups>
  <group name="group0" column="id" footer="groopfooter" header="groupheader" />
 </groups>
 <layout>
  <section name="page_head" typ="pageheader">
   <line name="line1" linewidth="2"  x1="0.0cm" x2="15.0cm" y1="0.1cm" y2="0.1cm"/>
   <field name="page_date_text" font="Times-Bold" fontsize="12" text="Date: " x="0.0cm" y="0.2cm"/>
   <field name="page_date" font="Times-Bold" fontsize="12" text="_date" x="1.2cm" y="0.2cm"/>
   <field name="page_time" font="Times-Bold" fontsize="12" text="_time" x="3.5cm" y="0.2cm"/>
   <field name="pagenr_text" font="Times-Bold" fontsize="12" height="1.0cm" text="Pagenr.:"  x="11.0cm" y="0.2cm"/>
   <field name="pagenr" font="Times-Bold" fontsize="12" height="1.0cm" text="_pagenr" x="13.0cm" y="0.2cm" />
  </section>

  <section name="detail"  alternate="1" bgcolor="(0.368627,0.619608,0.619608)" source="datasource0"  typ="detail">
   <field name="id" column="id"  key="1"  x="0.0cm" y="0.0cm"/>
   <field name="description" column="beschreibung" font="Times-Roman" fontsize="11" x="1.5cm" y="0.0cm"/>
   <field name="quantity"  column="quantity" font="Helvetica-Bold" fontsize="12" x="5.0cm" y="0.0cm"/>
   <field name="price" column="price" font="Times-Bold" fontsize="12" x="10.5cm" y="0.0cm" align="right" length="1.5cm" />
   <field name="p*q" align="right"  font="Times-Roman" fontsize="12" length="1.5cm" text="_value"  typ="real" x="12.0cm" y="0.0cm"/>
  </section>
  <section name="groupheader" source="datasource0" typ="groupheader">
   <field name="head_id" font="Times-Bold" fontsize="12" height="1.0cm" text="Id"  x="0.0cm" y="0.2cm"/>
   <field name="head_beschreibung" font="Times-Bold" fontsize="12" text="Description" x="1.5cm" y="0.2cm"/>
```

```
    <field name="head_quantity" font="Times-Bold" fontsize="12" text="Quantity" x="5.0cm" y="0.2cm"/>
    <field name="head_price" font="Times-Bold" fontsize="12" text="Price" x="10.5cm" y="0.2cm"/>
    <line  name="line0" linewidth="2" x1="0.0cm" x2="15.0cm" y1="0.1cm" y2="0.1cm"/>
    <field name="head_value" font="Helvetica-Bold" fontsize="12" text="Price * Quant." x="12.0cm" y="0.2cm"/>
   </section>

   <section name="groopfooter" source="datasource0" typ="groupfooter">
    <line name="footerline" linewidth="3" stroke="(-1.000000,-1.000000,-1.000000)" x1="0.0cm" x2="15.0cm" y1="0.5cm" y2="0.5cm"/>
    <field name="quant_s" text="_quant_sum"  typ="int" x="5.0cm" y="0.0cm"/>
    <field name="p*q_s" align="right" font="Times-Bold" fontsize="12" length="1.5cm"  text="_p*q_sum" typ="real" x="12.0cm" y="0.0cm"/>
   </section>
  </layout>
</report>
```



Date: 21. Jan 200508:53:00 **Pagenr.: 1**

Id	Description	Quantity	Price	Price * Quant.
Screws	Schraube M3	5	0.21	1.05
Screws	Schraube M5	5	0.23	1.15
Screws	Schraube M10	15	0.40	6.00
Screws	Schraube M13	7	0.43	3.01
Screws	Schraube M15	3	0.65	1.95
		35		13.16

Id	Description	Quantity	Price	Price * Quant.
Tools	Hammer	105	5.23	549.15
Tools	Zange	5	10.23	51.15
		110		600.30

Id	Description	Quantity	Price	Price * Quant.
Utensils	Rechen	35	14.00	490.00
Utensils	Schaufel	10	10.00	100.00
Utensils	Besen	5	11.23	56.15
		50		646.15