Open Office PDF Forms

v1.6.0

Introduction

OpenOffice.org is an office package that can be used on a number of platforms and languages. It is compatible with all the other major office software packages. OpenOffice is a free software that can be downloaded from www.openoffice.org. Version 2 of OpenOffice is the first office package that uses the new OASIS OpenDocument format. The word processor of OpenOffice is called Writer: it can be used to edit text from one page letters to full books containing embedded pictures, cross references, table of contents, indexes and source material lists. Writer is capable of opening other document formats (e.g. MS Office doc), so documents produced in other word processors can be used as a starting point for editing. OpenOffice Writer supports the creation and filling out of forms. It supports Xforms 1.0, that is a W3C standard. The other important feature of OpenOffice is that it is able to directly export any document into PDF format. When a document created in OpenOffice is exported to PDF all forms related information is converted into PDF form information that is called Acroform. The PDF that contains the Acroform prepared in this way can be opened and edited with Acrobat Reader, but unfortunately it cannot be saved directly. PL/PDF provides a solution to save AcroForms prepared in OpenOffice and filled out with data from the database using PL/SQL. The actual filling out of forms can be individual filling out of forms by users using a web application or mass filling out of forms from a database table using PL/SQL programming. The resulting PDF can be saved into the database, can be sent as an e-mail attachment or can be printed directly.
If national characters are used then Acrobat Reader version 7.0.9 or above is recommended.

Overview

These are the general steps to create a form and to fill it out:

1. Create the form in OpenOffice: an already existing document may be used or new document can be created in OpenOffice Writer. >>>
2. Create the Acroform: the Xforms has to be converted to PDF format. Export the document in OpenOffice. >>>
3. Verify the created PDF (optional). >>>
4. Upload the created Acroform into the database: the PDF containing the Acroform has to be placed in a BLOB field/variable, so that the PL/SQL program may access it. Any PL/SQL developer tool (PLSQL Developer, TOAD) or built-in tool available in the database (SQL*Loader, UTL_FILE) may be used to achieve this.
5. Write the PL/SQL code: the PL/SQL programs will prepare the the Acroform, fill out the document and save or provide the filled out PDF document to the user. >>>

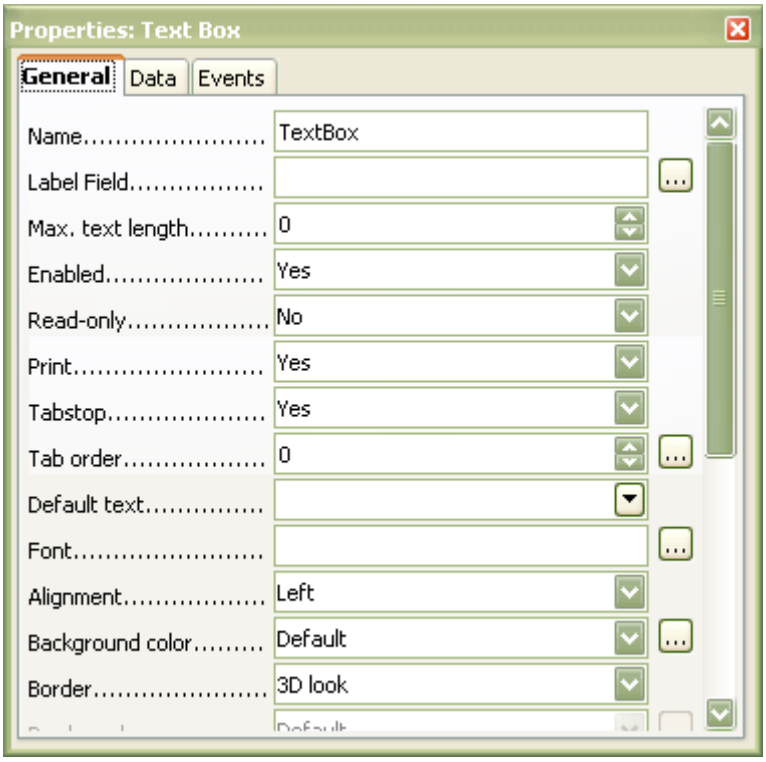
1. Create the form in OpenOffice

To create a form in OpenOffice an existing document may be used or a completely new document can be created. Text, pictures and other objects can be added to the document so that document in the end looks as the user wishes it to. If the user wants to add a form related object to the document then the „Form Controls” toolbar needs to be activated (menu: Tools/Toolbars).

In the toolbar use the „Design Mode On/Off” button to change into Design Mode. OpenOffice has a lot of form controls, but it can only convert some of them into Acroform, so we suggest the usage of the following controls:

- Check box
- Text box
- Push Button
- Option Button: if the „Wizards On/Off” button is On, then all the elements in the radio button group can be created at the same time.
- List Box
- Combo Box

The properties of the controls can be set on the „Properties” panel.



The most important properties as far as PL/PDF is concerned are:

- „Name”: it is crucial that the „Name” property is set to a unique name within the document because this field will be used to identity the form field to be filled by the PL/SQL program. If possible, only use lower case English characters “a” to “z” and numeric characters from “0” to “9” to avoid issues later (reason: OpenOffice is case sensitive and it converts any character that is different from the above).

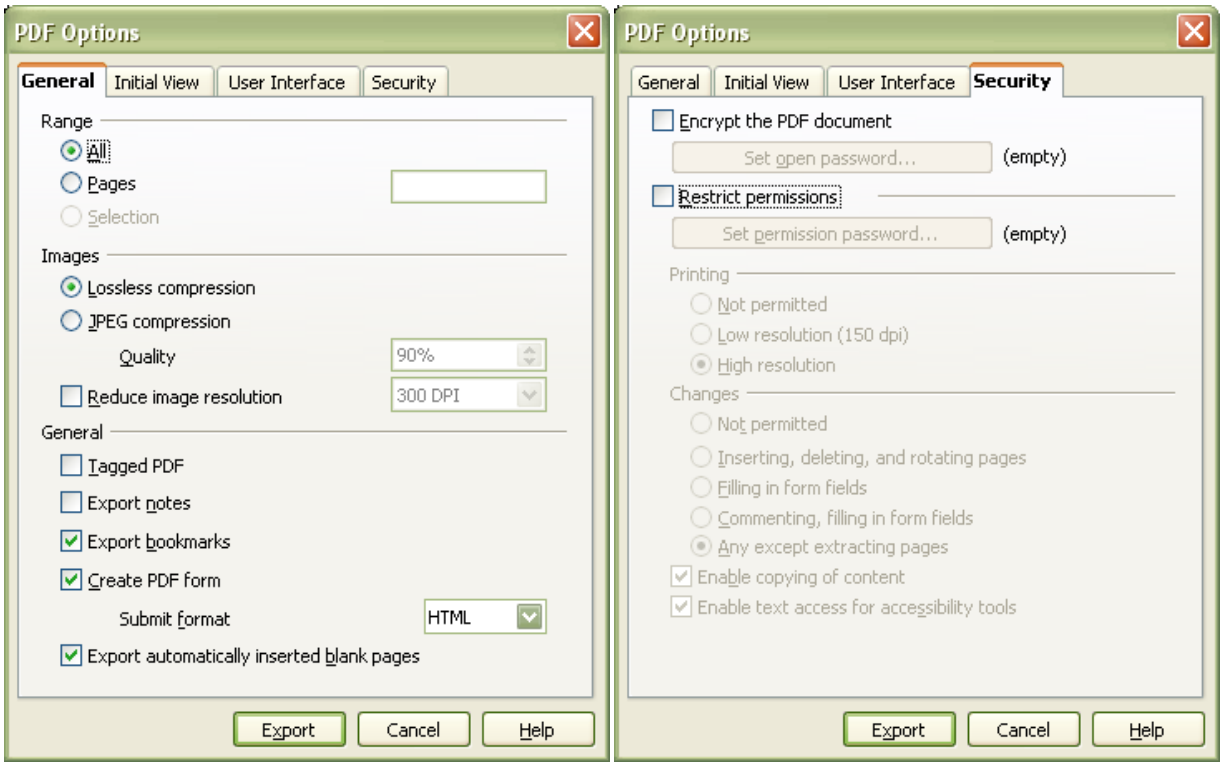
- „Print”: if set to „No” then the field does not appear in the PDF
- „Border”: if a field has a border then it will always be visible, even in the PDF regardless whether it is filled out or not.
- „Text type”: it may be used for single or multiple line fields.

It is fairly easy to use OpenOffice to prepare forms. Should any problems occur, the on-line help is very useful. For further information can be found at „How to create OpenOffice.org forms” (see Links) page.

2. Create the AcroForm

Once the form is created, then the next step is to convert the OpenOffice form into an AcroForm. There are two ways of doing this:

- Directly from the toolbar: „Export Directly as PDF”
- From the menu: „File”/”Export as PDF”. A dialog window appears where the properties can be set. It is important to set the following:
  - „Create PDF form”: set to yes, otherwise the form related information will not be converted.
  - „Security” tab: please do not use this option.



3. Verify the created PDF (optional)

Before uploading the Acroform into the database the field information in the PDF AcroForm can be checked to make sure that all the fields will be visible to PL/PDF. This can be done using the „plpdf_oo_form.GetPdfFieldsInfo” function. This function takes a BLOB variable where the PDF can be given as an input,

collect all the field related information and returns it in a „plpdf_type.t_form_fields" type variable (see plpdf_type package declaration for the structure of the type). The collected information can be displayed as an output either by custom code or by the „plpdf_oo_form.WriteOutPdfFieldsInfo" procedure, that uses „dbms_output.put_line".

4. Upload the created Acroform into the database

The PDF containing the Acroform has to be placed in a BLOB field/variable, so that the PL/SQL program may access it. Any PL/SQL developer tool (PLSQL Developer, TOAD) or built-in tool available in the database (SQL*Loader, UTL_FILE) may be used to achieve this.

5. Write the PL/SQL code

The PDF containing the Acroform can be stored in a BLOB type field of a table. This can be a simple table that can store the original AcroForm as well as the prepared version (see later). The suggested structure is the following:

```
create table plpdf_OO_form_store
(
  id number not null,
  filename varchar2(255 char),
  orig_pdf blob,
  prepform plpdf_type_prepform
)
nested table prepform.xref store as xref_table,
nested table prepform.objs store as objs_table
;
```

In this structure the PDF should be uploaded into the orig_pdf field.
The first step is to prepare the PDF. This procedure collects all the AcroForm related information from the PDF and stores it for later usage when the form is filled out. It is practical to do this work intensive part only once, when the PDF is uploaded into the database, as this information can only change if the AcroForm is changed. The procedure stores the information into a „plpdf_type_prepform" tipe field, in this case in the same table.
The PL/SQL code is the following:

```
procedure prepform(
    p_id number
    ) is
  l_blob blob;
  l_prepform plpdf_type.t_prepform;
  l_pf plpdf_type_prepform;
begin
  -- select PDF BLOB from table
  select t.orig_pdf
    into l_blob
  from plpdf_OO_form_store t
  where t.id = p_id
  ;

  -- prepare
  l_prepform := plpdf_oo_form.PrepPDF(l_blob);

  -- convert to stored format
  l_pf := plpdf_oo_form.Convert2Object(l_prepform);
```

```
  -- store prepared form in the table
  update plpdf_OO_form_store
    set prepform = l_pf
    where id = p_id;

  commit;

end;
```

The most important step in the code is:
l_prepform := plpdf_oo_form.PrepPDF(l_blob);
where the PDF stored in the BLOB variable is "prepared". The result is a „plpdf_type.t_prepform" type variable, that can be used when the form is filled out. If this variable needs to be stored (i.e. The recommended option) then it needs to be converted to a „plpdf_type_prepform" type, which is a type that exists in the database. The conversion can be done by the „plpdf_oo_form.Convert2Object" function. The result of this example code is a prepared Acroform, that is stored in a table in the database.
The next step is to fill out the prepared Acroform. For this the „plpdf_type_prepform" type data structure needs to be read from the database, with a code like this:

```
function loadform(
  p_id number
  ) return plpdf_type.t_prepform is
  l_pf plpdf_type_prepform;
  l_ret plpdf_type.t_prepform;
begin
  select prepform
    into l_pf
  from plpdf_OO_form_store
  where id = p_id
  ;

  l_ret := plpdf_oo_form.Convert2Type(l_pf);

  return l_ret;
end;
```

Part of the converts the „plpdf_type_prepform" type data structure into a „plpdf_type.t_prepform" type variable that can be used when the effective filling out of fields takes place. The „plpdf_oo_form.Convert2Type" function can be used to do this.
Once the prepared PDF is in a „plpdf_type.t_prepform" type variable, then the effective filling out can begin. The following procedures may be used:
- plpdf_oo_form.SetValue: Sets the value of a field
- plpdf_oo_form.SetReadonly: Sets the field to read-only in the PDF
- plpdf_oo_form.SetValueReadonly: Sets the value of the field and sets the field to read-only

Whether a field exists in the prepared PDF can be checked by the „plpdf_oo_form.FieldExist" function.
The last step is to convert the prepared and filled out PDF back into PDF form and into a BLOB variable: plpdf_oo_form.GetPDF.
Here is an example code (this saves the resulting PDF into the database):

```
procedure fillform_1(
  p_prepform in out plpdf_type.t_prepform,
```

```
  p_clubname varchar2,
  p_contactperson varchar2,
  p_phone varchar2,
  p_email varchar2,
  p_phonetype varchar2,
  p_city varchar2,
  p_association varchar2,
  p_kids boolean,-- cb
  p_cadets boolean, -- cb
  p_juniors boolean, -- cb
  p_open boolean, -- cb
  p_womengirls boolean, -- cb
  p_comments varchar2,
  p_date date-- date
  ) is
  --
begin
  plpdf_oo_form.SetValueReadonly(p_prepform,'clubname',p_clubname); -- CLUB NAME
  plpdf_oo_form.SetValueReadonly(p_prepform,'contactperson',p_contactperson); -- Contact Person
  plpdf_oo_form.SetValueReadonly(p_prepform,'phone',p_phone); -- Phone
  plpdf_oo_form.SetValueReadonly(p_prepform,'email',p_email); -- Email
  plpdf_oo_form.SetValueReadonly(p_prepform,'phonetype',p_phonetype); -- Phone type
  plpdf_oo_form.SetValueReadonly(p_prepform,'city',p_city); -- City
  plpdf_oo_form.SetValueReadonly(p_prepform,'association',p_association); -- Association
  plpdf_oo_form.SetValueReadonly(p_prepform,'kids',plpdf_oo_form.boolean2cb(p_kids)); -- kids
  plpdf_oo_form.SetValueReadonly(p_prepform,'cadets',plpdf_oo_form.boolean2cb(p_cadets)); -- cadets
  plpdf_oo_form.SetValueReadonly(p_prepform,'juniors',plpdf_oo_form.boolean2cb(p_juniors)); --
juniors
  plpdf_oo_form.SetValueReadonly(p_prepform,'open',plpdf_oo_form.boolean2cb(p_open)); -- open
  plpdf_oo_form.SetValueReadonly(p_prepform,'womengirls',plpdf_oo_form.boolean2cb(p_womengirls)); --
womengirls
  plpdf_oo_form.SetValueReadonly(p_prepform,'comments',p_comments); -- comments
  plpdf_oo_form.SetValueReadonly(p_prepform,'date',to_char(p_date,'YYYY-MM-DD')); -- date
end;
--
procedure test1 is
  l_prepform plpdf_type.t_prepform;
  l_prepform2 plpdf_type.t_prepform;
  l_blob blob;
begin
  l_prepform := loadform(1);
  fillform_1(
  p_prepform => l_prepform,
  p_clubname => 'San Jose W-Squad',
  p_contactperson => 'Joe Schlabotnik',
  p_phone => '408-555-7871',
  p_email => 'joe_schlabotnik@example.com',
  p_phonetype => '2',
  p_city => 'San Jose, CA',
  p_association => 'SCWA',
  p_kids => true,-- cb
  p_cadets => true, -- cb
  p_juniors => false, -- cb
  p_open => false, -- cb
  p_womengirls => true, -- cb
  p_comments =>
    'Lets say youre in charge of a database of chartered clubs for an amateur sports association. '
||
    'Club directors send you papers like the one in Appendix A, and you enter the data into an XML
file, ' ||
    'which is used to create an online searchable database of the clubs.  Clearly, the better option
is ' ||
    'to send the club directors a machine-readable document with form fields that they fill in. ' ||
    'The directors send the file back to you, and, rather than having to decipher their handwriting
```

```
' ||
    'and re-enter the data, you run a program to extract the information.',
  p_date => sysdate
  );

  plpdf_oo_form.GetPDF(
  p_prepform => l_prepform,
  p_blob => l_blob
  );

 insert into store_blob (blob_file,created_date) values
 (l_blob,sysdate);

 commit;

end;
```

You may use the following procedure to convert logical values (true/false) to an On/Off state in a check box: „plpdf_oo_form.boolean2cb”.

Client-side PDF filling

The forms created using OpenOffice can be used for manual input by users in Acrobat Reader. The problem with Acrobat Reader is that filled out forms cannot be saved, only printed. With PL/PDF OpenOffice extension this limitation can be overcome and an application can be built that will simulate saving the PDF in Acrobat Reader (the MOD_PLSQL environment needs to be set up for this).
We can use the OpenOffice „Push Button" for this, which can submit the form to the server. According to the PDF data PL/PDF can fill out the form (that is prepared in the database) and can resend the filled out document back to the user. Some steps that may be used for the application are:

- The „Push Button" property „Action" needs to be set to:"Submit Form",.
- The button („Push Button") should not be on the filled out form so the „Print" property of the button should be set to „No". This way the OpenOffice export will not put the button into the PDF.
- The „URL" property of the „Form Properties" should be set to the address of the procedure that will execute the filling out of the PDF. The „Type of submission" should be „Post".
- When we are „Exporting as PDF...", „Submit format" should be set to „HTML".
- The PL/SQL code needs to be written that will do the effective filling out. This will be the procedure that will be provided as a „URL" above. Naturally, the PDF that will be filled out needs to be in the database, since the program will reference it. The PL/SQL program parameter names must match the Acroform field names. The button needs to be a parameter as well because all the fields are passed from Acrobat Reader. The button will have no value, so it needs to be set as „default null".
- If the character set of the data sent by Submit is different from the charset of the database then the data may need to be converted using the CONVERT function.

After this the PDF can be supplied to the web page and when it is submitted (using the button) by the user to the server, then the program will return the filled out PDF.

Links

OpenOffice.org

OpenDocument software

The Forms Working Group

How to create OpenOffice.org forms

Using XForms in Office Applications

OpenXML/ODF Translator Add-in for Office

Acrobat PDF Forms: A Step-by-step Introduction