QWeb Reports

Reports are written in HTML/QWeb, like website views in Odoo. You can use the usual QWeb control flow tools (qweb.html#reference-qweb). The PDF rendering itself is performed by wkhtmltopdf (https://wkhtmltopdf.org).

Reports are declared using a report action (actions.html#reference-actions-report), and a Report template for the action to use.

If useful or necessary, it is possible to specify a Paper Format for the report report.

Report template

Report templates will always provide the following variables:

`time`
: a reference to `time` (https://docs.python.org/3/library/time.html#module-time) from the Python standard library

`user`
: `res.user` record for the user printing the report

`res_company`
: record for the current `user`'s company

`website`
: the current website object, if any (this item can be present but `None`)

`web_base_url`
: the base url for the webserver

`context_timestamp`
: a function taking `datetime.datetime` (https://docs.python.org/3/library/datetime.html#datetime.datetime) in UTC[1] and converting it to the timezone of the user printing the report

Minimal viable template

A minimal template would look like:

```
<template id="report_invoice">
    <t t-call="web.html_container">
        <t t-foreach="docs" t-as="o">
            <t t-call="web.external_layout">
                <div class="page">
                    <h2>Report title</h2>
                    <p>This object's name is <span t-field="o.name"/></p>
                </div>
            </t>
        </t>
    </t>
</template>
```

Calling `external_layout` will add the default header and footer on your report. The PDF body will be the content inside the `<div class="page">`. The template's `id` must be the name specified in the report declaration; for example `account.report_invoice` for the above report. Since this is a QWeb template, you can access all the fields of the `docs` objects received by the template.

By default, the rendering context will also expose the following items:

`docs`
records for the current report

`doc_ids`
list of ids for the `docs` records

`doc_model`
model for the `docs` records

If you wish to access other records/models in the template, you will need a custom report, however in that case you will have to provide the items above if you need them.

Translatable Templates

If you wish to translate reports (to the language of a partner, for example), you need to define two templates:

- The main report template
- The translatable document

You can then call the translatable document from your main template with the attribute `t-lang` set to a language code (for example `fr` or `en_US`) or to a record field. You will also need to re-browse the related records with the proper context if you use fields that are translatable (like country names, sales conditions, etc.)

⚠ Warning

If your report template does not use translatable record fields, re-browsing the record in another language is *not* necessary and will impact performances.

For example, let's look at the Sale Order report from the Sale module:

```
<!-- Main template -->
<template id="report_saleorder">
    <t t-call="web.html_container">
        <t t-foreach="docs" t-as="doc">
            <t t-call="sale.report_saleorder_document" t-lang="doc.partner_id.lang"/>
        </t>
    </t>
</template>

<!-- Translatable template -->
<template id="report_saleorder_document">
    <!-- Re-browse of the record with the partner lang -->
    <t t-set="doc" t-value="doc.with_context(lang=doc.partner_id.lang)" />
    <t t-call="web.external_layout">
        <div class="page">
            <div class="oe_structure"/>
            <div class="row">
                <div class="col-6">
                    <strong t-if="doc.partner_shipping_id == doc.partner_invoice_id">Inv
                    <strong t-if="doc.partner_shipping_id != doc.partner_invoice_id">Inv
                    <div t-field="doc.partner_invoice_id" t-options="{&quot;no_marker&qu
                <...>
            <div class="oe_structure"/>
        </div>
    </t>
</template>
```

The main template calls the translatable template with `doc.partner_id.lang` as a `t-lang` parameter, so it will be rendered in the language of the partner. This way, each Sale Order will be printed in the language of the corresponding customer. If you wish to translate only the body of the document, but keep the header and footer in a default language, you could call the report's external layout this way:

```
<t t-call="web.external_layout" t-lang="en_US">
```

> Please take note that this works only when calling external templates, you will not be able to translate part of a document by setting a `t-lang` attribute on an xml node other than `t-call`. If you wish to translate part of a template, you can create an external template with this partial template and call it from the main one with the `t-lang` attribute.

Barcodes

Barcodes are images returned by a controller and can easily be embedded in reports thanks to the QWeb syntax (e.g. see attributes (qweb.html#reference-qweb-attributes)):

```
<img t-att-src="'/report/barcode/QR/%s' % 'My text in qr code'"/>
```

More parameters can be passed as a query string

```
<img t-att-src="'/report/barcode/?
    type=%s&amp;value=%s&amp;width=%s&amp;height=%s'%('QR', 'text', 200, 200)"/>
```

Useful Remarks

Twitter Bootstrap and FontAwesome classes can be used in your report template

Local CSS can be put directly in the template

Global CSS can be inserted in the main report layout by inheriting its template and inserting your CSS:

```
<template id="report_saleorder_style" inherit_id="report.style">
  <xpath expr=".">
    <t>
      .example-css-class {
        background-color: red;
      }
    </t>
  </xpath>
</template>
```

If it appears that your PDF report is missing the styles, please check these instructions (../howtos/backend.html#reference-backend-reporting-printed-reports-pdf-without-styles).

Paper Format

Paper formats are records of `report.paperformat` and can contain the following attributes:

`name` (mandatory)
only useful as a mnemonic/description of the report when looking for one in a list of some sort

`description`
a small description of your format

`format`
either a predefined format (A0 to A9, B0 to B10, Legal, Letter, Tabloid,...) or `custom`; A4 by default. You cannot use a non-custom format if you define the page dimensions.

`dpi`
output DPI; 90 by default

`margin_top`, `margin_bottom`, `margin_left`, `margin_right`
margin sizes in mm

`page_height`, `page_width`

page dimensions in mm

`orientation`
Landscape or Portrait

`header_line`
boolean to display a header line

`header_spacing`
header spacing in mm

Example:

```
<record id="paperformat_frenchcheck" model="report.paperformat">
    <field name="name">French Bank Check</field>
    <field name="default" eval="True"/>
    <field name="format">custom</field>
    <field name="page_height">80</field>
    <field name="page_width">175</field>
    <field name="orientation">Portrait</field>
    <field name="margin_top">3</field>
    <field name="margin_bottom">3</field>
    <field name="margin_left">3</field>
    <field name="margin_right">3</field>
    <field name="header_line" eval="False"/>
    <field name="header_spacing">3</field>
    <field name="dpi">80</field>
</record>
```

Custom Reports

By default, the reporting system builds rendering values based on the target model specified through the `model` field.
However, it will first look for a model named `report.`***`module.report_name`*** and call that model's `_get_report_values(doc_ids, data)` in order to prepare the rendering data for the template.
This can be used to include arbitrary items to use or display while rendering the template, such as data from additional models:

```
from odoo import api, models

class ParticularReport(models.AbstractModel):
    _name = 'report.module.report_name'

    def _get_report_values(self, docids, data=None):
        # get the report action back as we will need its data
        report = self.env['ir.actions.report']._get_report_from_name('module.report_name
        # get the records selected for this rendering of the report
        obj = self.env[report.model].browse(docids)
        # return a custom rendering context
        return {
            'lines': docids.get_lines()
        }
```

⚠ Warning

When using a custom report, the "default" document-related items (`doc_ids` , `doc_model` and `docs`) will *not* be included. If you want them, you will need to include them yourself.

In the example above, the rendering context will contain the "global" values as well as the `lines` we put in there but nothing else.

Custom fonts

If you want to use custom fonts you will need to add your custom font and the related less/CSS to the `web.reports_assets_common` assets bundle. Adding your custom font(s) to `web.assets_common` or `web.assets_backend` will not make your font available in QWeb reports.

Example:

```
<template id="report_assets_common_custom_fonts" name="Custom QWeb fonts" inherit_id="wel
    <xpath expr="." position="inside">
        <link href="/your_module/static/src/less/fonts.less" rel="stylesheet" type="text
    </xpath>
</template>
```

You will need to define your `@font-face` within this less file, even if you've used in another assets bundle (other than `web.reports_assets_common`).

Example:

```
@font-face {
    font-family: 'MonixBold';
    src: local('MonixBold'), local('MonixBold'), url(/your_module/static/src/fonts/Monix
}

.h1-title-big {
    font-family: MonixBold;
    font-size: 60px;
    color: #3399cc;
}
```

After you've added the less into your assets bundle you can use the classes - in this example **`h1-title-big`** - in your custom QWeb report.

Reports are web pages

Reports are dynamically generated by the report module and can be accessed directly via URL:
For example, you can access a Sale Order report in html mode by going to http://<server-address>/report/html/sale.report_saleorder/38
Or you can access the pdf version at http://<server-address>/report/pdf/sale.report_saleorder/38

[1] it does not matter what timezone the **`datetime`** object is actually in (including no timezone), its timezone will unconditionally be *set* to UTC before being adjusted to the user's