QWEB

目录

[数据输出 2](#__RefHeading___Toc124_4165040904)

[条件 2](#__RefHeading___Toc126_4165040904)

[循环 3](#__RefHeading___Toc128_4165040904)

[属性 5](#__RefHeading___Toc130_4165040904)

[设置变量 6](#__RefHeading___Toc132_4165040904)

[调用子模板 6](#__RefHeading___Toc134_4165040904)

[Python 8](#__RefHeading___Toc136_4165040904)

[专用指令 8](#__RefHeading___Toc138_4165040904)

[资源包 8](#__RefHeading___Toc140_4165040904)

[“智能记录”字段格式化 8](#__RefHeading___Toc142_4165040904)

[调试 8](#__RefHeading___Toc144_4165040904)

[帮助 8](#__RefHeading___Toc146_4165040904)

[基于请求 8](#__RefHeading___Toc148_4165040904)

[基于视图 9](#__RefHeading___Toc150_4165040904)

[API 10](#__RefHeading___Toc152_4165040904)

[Javascript 10](#__RefHeading___Toc154_4165040904)

[专用指令 10](#__RefHeading___Toc156_4165040904)

[定义模板 10](#__RefHeading___Toc158_4165040904)

[模板继承 10](#__RefHeading___Toc160_4165040904)

[调试 11](#__RefHeading___Toc162_4165040904)

[帮助 12](#__RefHeading___Toc164_4165040904)

[API 12](#__RefHeading___Toc166_4165040904)

QWeb是Odoo使用的主要模板引擎。它是一个XML模板引擎，主要用于生成HTML片段和页面。

模板伪指令被指定为以 **t-** 为前缀的XML属性，例如 **t-if** 用于 [条件](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "reference-qweb-conditionals)，元素和其他属性直接呈现。

为了避免元素渲染，一个占位符元素 **<t>** 也是可用的，它执行它的指令，但不生成任何输出:

**<t t-if="condition">**

**<p>Test</p>**

**</t>**

将导致:

**<p>Test</p>**

如果 **condition** 是对的，但:

**<div t-if="condition">**

**<p>Test</p>**

**</div>**

将导致:

**<div>**

**<p>Test</p>**

**</div>**

## 数据输出

QWeb有一个主输出指令，当显示用户提供的内容: **esc** 时，HTML会自动转义其限制 [XSS](http://en.wikipedia.org/wiki/Cross-site_scripting) 的内容。

**esc** 接受一个表达式，计算它并打印内容:

**<p><t t-esc="value"/></p>**

使用值 **value** 设置为 **42** ，产生:

**<p>42</p>**

还有一个其它输出指令 **raw**，它的行为分别与 **esc** 相同，但是 不会对其输出进行HTML转义 。显示单独构建的标记（例如，从函数方法中）或用户提供的安全标记可能是有用的。

## 条件

QWeb有一个条件指令 **if** ，它计算表达式作为属性的值:

**<div>**

**<t t-if="condition">**

**<p>ok</p>**

**</t>**

**</div>**

如果条件是成立的，则呈现元素:

**<div>**

**<p>ok</p>**

**</div>**

但是如果条件不成立，则从结果中删除:

**<div>**

**</div>**

条件渲染适用于指令的载体，它不必是 **<t>**

**<div>**

**<p t-if="condition">ok</p>**

**</div>**

将给出与前面例子相同的结果。

额外的条件分支指令 **t-elif** 和 **t-else** 也可用:

**<div>**

**<p t-if="user.birthday == today()">Happy bithday!</p>**

**<p t-elif="user.login == 'root'">Welcome master!</p>**

**<p t-else="">Welcome!</p>**

**</div>**

## 循环

QWeb有一个迭代指令 **foreach**，它接受一个表达式返回集合迭代，第二个参数 **t-as** 提供迭代的 "当前项目"的名称:

**<t t-foreach="[1, 2, 3]" t-as="i">**

**<p><t t-esc="i"/></p>**

**</t>**

将呈现为:

**<p>1</p>**

**<p>2</p>**

**<p>3</p>**

跟条件一样， **foreach** 适用于带有指令属性的元素

**<p t-foreach="[1, 2, 3]" t-as="i">**

**<t t-esc="i"/>**

**</p>**

相当于前面的例子。

**foreach** 可以对一个数组进行迭代 (当前项将是当前值)，一个映射 (当前项将是当前键)或一个整数 (相当于在0和所提供的整数之间的数组迭代).

除了通过 **t-as** 传递名字， **foreach** 为各种数据点提供了一些其他变量：

**警告**

**$as** 将被传递给 **t-as** 的名称所取代

**$as\_all**

该对象被迭代

**$as\_value**

当前迭代值，对于列表和整数与 **$as** 相同，但是对于映射，它提供了值(**$as** 提供了键)

**$as\_index**

当前迭代索引（迭代的第一项具有索引0）

**$as\_size**

集合的大小（如果可用）

**$as\_first**

是否当前项是第一次迭代 (等价于 **$as\_index == 0**)

**$as\_last**

当前项是否是迭代的最后一个(等价于 **$as\_index + 1 == $as\_size**)，需要迭代的大小可用

**$as\_parity**

即 **"even"** 或 **"odd"**，当前迭代的奇偶性

**$as\_even**

指示当前迭代轮次在偶数索引上的布尔标志

**$as\_odd**

指示当前迭代轮次在奇数索引上的布尔标志

提供的这些额外的变量和创建到 **foreach** 的所有新变量只在 **foreach** 的范围内可用。如果变量存在于 **foreach** 的上下文之外，则该值将在foreach的末尾处复制到全局的上下文中。

**<t t-set="existing\_variable" t-value="False"/>**

**<!-- existing\_variable now False -->**

**<p t-foreach="[1, 2, 3]" t-as="i">**

**<t t-set="existing\_variable" t-value="True"/>**

**<t t-set="new\_variable" t-value="True"/>**

**<!-- existing\_variable and new\_variable now True -->**

**</p>**

**<!-- existing\_variable always True -->**

**<!-- new\_variable undefined -->**

## 属性

QWeb 可以即时计算属性并设置计算结果在输出节点。这是 通过 **t-att** (属性)指令来实现的，它存在3种不同的形式：

**t-att-$name**

创建一个名为 **$name** 的属性，对属性值进行求值并将结果设置为属性的值:

**<div t-att-a="42"/>**

将被渲染为:

**<div a="42"></div>**

**t-attf-$name**

与上一个相同，但是参数是一个 [format string](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/glossary.html" \l "term-format-string) 而不是一个表达式，通常用于混合字面值和非 字面值字符串（例如类）:

**<t t-foreach="[1, 2, 3]" t-as="item">**

**<li t-attf-class="row {{ item\_parity }}"><t t-esc="item"/></li>**

**</t>**

将呈现为:

**<li class="row even">1</li>**

**<li class="row odd">2</li>**

**<li class="row even">3</li>**

**t-att=mapping**

如果参数是映射，则每个（键，值）对将生成一个新属性及其值:

**<div t-att="{'a': 1, 'b': 2}"/>**

将呈现为:

**<div a="1" b="2"></div>**

**t-att=pair**

如果参数是一对（2元素的元组或数组），则该对的第一项是属性的名称，第二项是值:

**<div t-att="['a', 'b']"/>**

将呈现为:

**<div a="b"></div>**

## 设置变量

QWeb允许从模板内创建变量，记住计算（使用它多次），给一堆数据一个明显的名称 ...

这是通过 **set** 指令完成的，它接受要创建的变量的名称。要设置的值可以通过两种方式提供:

* 一个包含表达式的 **t-value** 属性，其计算结果将被设置:

**<t t-set="foo" t-value="2 + 1"/>**

**<t t-esc="foo"/>**

将会打印 **3**

* 如果没有 **t-value** 属性，则渲染节点的主体并将设置为变量的值:

**<t t-set="foo">**

**<li>ok</li>**

**</t>**

**<t t-esc="foo"/>**

将会生成 **&lt;li&gt;ok&lt;/li&gt;** (内容被转义，因为我们使用 **esc** 指令)

**注解**

使用这个操作的结果是一个重要的用例的 **raw** 指令。

## 调用子模板

QWeb模板可以用于顶级渲染，但也可以从另一个模板（避免重复或给部分模板命名）通过使用 **t-call** 指令来使用它们:

**<t t-call="other-template"/>**

这将调用具有具有执行上下文命名的父模板，如果 **other\_template** 定义为:

**<p><t t-value="var"/></p>**

上面的调用将被渲染为 **<p/>** (无内容)，但是:

**<t t-set="var" t-value="1"/>**

**<t t-call="other-template"/>**

将被渲染为 **<p>1</p>**。

然而，这里有一个问题是从 **t-call** 外部可见的。或者在 **call** 指令的主体中设置的内容将在调用模 板 之前 被计算，并且可以改变本地上下文:

**<t t-call="other-template">**

**<t t-set="var" t-value="1"/>**

**</t>**

**<!-- "var" does not exist here -->**

**call** 指令的主体可以是任意复杂的(而不仅仅是 **set** 指令)，并且它的呈现形式将在被调用的模板中作为一个神奇的 **0**变量使用:

**<div>**

**This template was called with content:**

**<t t-raw="0"/>**

**</div>**

因此被称为:

**<t t-call="other-template">**

**<em>content</em>**

**</t>**

将导致:

**<div>**

**This template was called with content:**

**<em>content</em>**

**</div>**

## Python

### 专用指令

#### 资源包

#### “智能记录”字段格式化

**t-field** 指令智能在“智能记录”( **browse** 方法)的结果中执行字段访问( **a.b** )时使用。它能够基于字段类型自动格式化，并集成在网站的副文本版本中。

**t-options** 可以用于自定义字段，最常见的选项是 **widget** ，其它选项是依赖于字段或者widget的。

### 调试

**t-debug**

使用PDB的 **set\_trace** API调用调试器。参数应该是一个模块的名称，在其上调用 **set\_trace** 方法:

**<t t-debug="pdb"/>**

相当于 **importlib.import\_module("pdb").set\_trace()**

### 帮助

#### 基于请求

QWeb的大多数Python端使用都在控制器中(在HTTP请求期间)，在这种情况下，存储在数据库中的模板(如 [views](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/views.html" \l "reference-views-qweb)) 可以简单地通过调用 **odoo.http.HttpRequest.render()**:

**response = http.request.render('my-template', {**

**'context\_value': 42**

**})**

这自动创建一个 **Response** 对象可以从控制器返回（或进一步定制合适的）。

#### 基于视图

在比以前的帮助者更深层次的是 **ir.ui.view** 的 **render** 方法:

###### **render(cr, uid, id[, values][, engine='ir.qweb][, context])**

通过数据库id或 [external id](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/glossary.html" \l "term-external-id) 呈现QWeb视图/模板。模板从 **ir.ui.view** 记录自动加载。

在呈现上下文中设置多个默认值：

**request**

当前 **WebRequest** 对象，如果有

**debug**

当前请求（如果有）是否在 **debug** 模式

**[quote\_plus](http://werkzeug.pocoo.org/docs/urls/" \l "werkzeug.urls.url_quote_plus)**

url编码效用函数

**[json](https://docs.python.org/2/library/json.html" \l "module-json)**

相应的标准库模块

**[time](https://docs.python.org/2/library/time.html" \l "module-time)**

相应的标准库模块

**[datetime](https://docs.python.org/2/library/datetime.html" \l "module-datetime)**

相应的标准库模块

**[相关文章](https://labix.org/python-dateutil" \l "head-ba5ffd4df8111d1b83fc194b97ebecf837add454)**

参考模块

**keep\_query**

**keep\_query** 帮助函数

**参数**

* **values** -- 上下文值传递到QWeb进行渲染
* **engine** (**[str](https://docs.python.org/2/library/functions.html" \l "str)**) -- 用于渲染的Odoo模型的名称，可以用于在本地扩展或自定义QWeb（通过创建一个

基于 **ir.qweb** 改变的"新" qweb)

### API

也可以直接使用 **ir.qweb** 模型（并扩展它，继承它）:

## Javascript

### 专用指令

#### 定义模板

**t-name** 指令智能放在模板文件的顶层（直接子文件到文档根）:

**<templates>**

**<t t-name="template-name">**

**<!-- template code -->**

**</t>**

**</templates>**

它不需要其它参数，但可以使用一个 **<t>** 元素或任何其它的。使用 **<t>** 元素，**<t>** 应该有一个子元素。

模板名称是任意字符串，但是当多个模板相关时（例如称为子模板），通常使用点分割的名称来指示分层关系。

#### 模板继承

模板继承用于就地改变现有模板，例如，以向其他模块所创建的模板添加信息。

模板继承通过 **t-extend** 指令执行，它将模板的名称作为参数修改。

然后使用任意数量的 **t-jquery** 子命令执行更改:

**<t t-extend="base.template">**

**<t t-jquery="ul" t-operation="append">**

**<li>new element</li>**

**</t>**

**</t>**

**t-jquery** 指令采用一个 [CSS selector](http://api.jquery.com/category/selectors/) 。此选择器用于扩展模板以选择应用指定的 **t-operation** 的 上下文节点 :

**append**

节点的主体被附加在上下文节点的末尾（在上下文节点的最后一个子节点之后）

**prepend**

节点的主体被添加到上下文节点（在上下文节点的第一个子节点之前插入）

**before**

节点的主体被插入在上下文节点之前

**after**

节点的主体被插入在上下文节点之后

**inner**

节点的主体替换上下文节点的子节点

**replace**

该节点的主体用于替换上下文节点本身

**无操作**

如果没有指定 **t-operation** ，那么模板主体被解释为javascript代码，并使用上下文节点作 为 **this** 来执行。

**警告**

同时比其它操作更强大，这种模式也更难调试和维护，建议避免它

### 调试

javascript QWeb实现提供了一些调试:

**t-log**

使用表达式参数，在呈现过程中计算表达式并使用 **console.log** 记录其结果:

**<t t-set="foo" t-value="42"/>**

**<t t-log="foo"/>**

将打印 **42** 到控制台

**t-debug**

在模板渲染期间触发调试器断点:

**<t t-if="a\_test">**

**<t t-debug="">**

**</t>**

如果调试活跃度较高（确切的条件取决于浏览器及其开发工具）将停止执行

**t-js**

节点的主体是在模板渲染期间执行的JavaScript代码。获取一个 **context** 参数，这是 在 **t-js** 的主体中 可以使用的渲染上下文的名称:

**<t t-set="foo" t-value="42"/>**

**<t t-js="ctx">**

**console.log("Foo is", ctx.foo);**

**</t>**

### 帮助

###### **core.qweb**

(核心是 **web.core** 模块)一个实例 **[QWeb2.Engine()](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "QWeb2.Engine)** ，所有模块定义的模板文件被加载，引用标准帮助器对象 **\_** (下划线)， **\_t** (翻译函数) 和 [JSON](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON) 。

**[core.qweb.render](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "QWeb2.Engine.render)** 可以用来轻松渲染基本模块模板

### API

###### **class QWeb2.Engine()**

QWeb "renderer" ，处理大多数QWeb的逻辑（加载，解析，编辑和渲染模板）。

OpenERP Web在核心模块中为用户实例化一个，并将其导出到 **core.qweb** 。它还将各种模块的所有模板文件加载到QWeb实例中。

A **[QWeb2.Engine()](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "QWeb2.Engine)** 也用作"模板命名空间".

###### **QWeb2.Engine.render(template[, context])**

使用 **context** (如果提供有)将先前加载的模板呈现给字符串，以找到在模板呈现期间访问的变量(例如要显示的字符串)。

**参数**

* **template** (**String**) -- 要呈现的模板的名称
* **context** (**Object**) -- 用于模板渲染的基本命名空间

**返回**

字符串

引擎提供了另一种方法，在某些情况下可能是有用的(例如，如果你需要一个单独的模板命名空间，在OpenERP Web中，看板视图有自己的 **[QWeb2.Engine()](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "QWeb2.Engine)** 实例，所以他们的模板不会与更一般的“模块”模板碰撞):

###### **QWeb2.Engine.add\_template(templates)**

在QWeb实例中装入模板文件（模板集合）。模板可以指定为:

**XML字符串**

QWeb尝试将其解析为XML文档，然后加载它。

**网址**

QWeb将尝试下载网址内容，然后加载生成的XML字符串。

**一个 Document 或 Node**

QWeb将遍历文档的第一级（提供根的子节点）并加载任何命名的模板或模板覆盖。

**[QWeb2.Engine()](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "QWeb2.Engine)** 还暴露了行为定制的各种属性:

###### **QWeb2.Engine.prefix**

前缀用于在解析期间识别指令，字符串。默认情况下为 **t** 。

###### **QWeb2.Engine.debug**

布尔标志将引擎置于“调试模式”。通常，QWeb拦截在模板执行期间引发的任何错误。在调试模式下，它使所 有异常通过，而不拦截它们。

###### **QWeb2.Engine.jQuery**

在模板继承处理期间使用的jQuery实例。默认为 **window.jQuery** 。

###### **QWeb2.Engine.preprocess\_node**

**Function** 。如果存在，在将每个DOM节点编译为模板代码之前调用。在OpenERP Web中，这用于 自动翻译文本内容和模板中一些属性。默认为 **null**。

[[1]](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "id2) XML模板引擎 它类似于 [Genshi](http://genshi.edgewall.org/) , 虽然它不使用（不支持） [XML namespaces](http://en.wikipedia.org/wiki/XML_namespace)

[[2](http://odoo-10-dev-translation.readthedocs.io/zh_CN/latest/reference/qweb.html" \l "id1) odoo  尽管它使用了一些其他的，由于历史原因，或者因为它们更适合用例。Odoo 9.0 仍然取决于[Jinja](http://jinja.pocoo.org/)和[Mako](http://www.makotemplates.org/)。