# **odoo V10中文参考手册（六：安全、测试用例、web控制器）**

[鼎易](https://www.jianshu.com/u/6fdae8ec06bc)软件

2017.04.11

# 安全

odoo提供了两种方式来管理数据权限而不需要通过手写权限管理相关代码，每种方式都通过用户组来指定用户：一个用户属于多个用户组，安全机制通过用户组关联作用到用户

## 访问控制

1.通过ir.model.access记录来管理，定义对整个模型的访问权限  
2.每一条记录包含有 (1)对应的授权模型 (2)授予的权限 (3) 可选的用户组  
3.访问权限控制是附加的，对于一个模型用户拥有赋予它所在组的权限：如果用户所在用户组1拥有写权限，用户组2有删除权限，那么该用户拥有写和删除权限）  
4.如果没有指定用户组，这条权限控制就会作用在所有用户上，否则只会作用在指定组的用户上  
5.可用的权限是：写(perm\_create),搜索和读取(perm\_read),更新现有记录(perm\_write),删除记录(perm\_unlink)

## 记录行规则

记录行规则规定进行(create, read, update or delete)操作时数据需要满足的条件，它会在应用了访问控制权限之后逐条记录进行应用

记录行规则的要素：  
1.需要被应用规则的model  
2.需要应用的规则（如果设置了perm\_read ，那么只在读取记录时检查）  
3.需要应用规则的用户组，如果没指定用户组，那么该规则就是全局的  
4.用来检查记录是否满足条件的domain表达式，如果true表示可访问，false表示不可访问，domain表达式使用两个上下文变量：user - 当前用户记录，time - python time

全局规则和基于用户组的规则使用方式不同：

* 全局规则必须全部满足才能赋予权限
* 用户组规则是附加的，只要满足其中一个就可以有权限（需要满足全局权限）

记录行规则对于管理员用户是没有作用的

## 字段权限

一个ORM字段可以通过groups属性来指定一个可访问的组列表（以逗号分隔）

如果当前用户不在指定的任何一个用户组中，那么不能访问该字段

* 没访问权限的字段会自动从视图中移除
* 没权限的字段也会从fields\_get() 函数的返回中移除
* 尝试去读或写未授权字段会报出一个权限错误

## 工作流转换规则

工作流也可以通过指定用户组来控制权限，用户不在指定组内的将不会触发相应动作

# 模块测试

odoo支持通过单元测试来测试模块，写测试用例只需要在模块文件夹建立一个test文件夹，测试模块文件名须以test\_开头，并且在\_\_ init \_\_.py里导入

your\_module

|-- ...

`-- tests

|-- \_\_init\_\_.py

|-- test\_bar.py

`-- test\_foo.py

#\_\_init\_\_.py

from . import test\_foo, test\_bar

默认只能运行python官方提供的部分测试，odoo提供了一些用来测试odoo内容的工具

* class odoo.tests.common.TransactionCase(methodName='runTest')  
  每个测试方法运行在自己的事务中的测试用例，使用自己的数据库游标，当用例执行完自动回滚
* browse\_ref(xid) 返回指定id的记录对象
* ref(xid) 返回提供的external identifier对应的数据库id，与get\_object\_reference一致
* class odoo.tests.common.SingleTransactionCase(methodName='runTest')  
  所有测试方法都在同一个事务中执行的测试用例，事务从第一个测试方法开始，直到最后一个完成才回滚
* browse\_ref(xid) 返回指定id的记录对象
* ref(xid) 返回提供的external identifier对应的数据库id，与get\_object\_reference一致

默认情况下，在安装好对应模块后会自动执行对应的测试用例，也可配置为在所有模块安装完成后再运行

* odoo.tests.common.at\_install(flag)  
  设置安装测试用例，flag参数(boolean)设置该测试用例是否在安装时执行，默认情况下会在安装完该模块后，安装下一个模块前自动运行用例
* odoo.tests.common.post\_install(flag)  
  设置安装后测试用例，flag参数(boolean)表示是否在安装后自动运行用例，默认情况下测试用例只会在所有模块安装完成后才运行

class TestModelA(common.TransactionCase):

def test\_some\_action(self):

record = self.env['model.a'].create({'field': 'value'})

record.some\_action()

self.assertEqual(

record.field,

expected\_field\_value)

# other tests...

当启动odoo时如果指定--test-enable ，测试用例会在安装或更新模块时运行

# web控制器

## 路由

odoo.http.route(route=None, \*\*kw) 装饰器可以装对应方法装饰为处理对应的http请求，该方法须是Controller的子类

参数列表：

* route -- 字符串或数组，决定哪个http请求匹配所装饰的方法，可以是当个字符串、或多个字符串的数组，用的是werkzeug的路由http://werkzeug.pocoo.org/docs/routing/
* type -- 请求的类型，可以是http或json
* auth -- 认证方法的类型，可以是以下几种：
  + user - 必须是认证的用户，该请求基于已认证的用户
  + public - 当不通过认证访问时使用公用的认证
  + none - 相应的方法总是可用，一般用于框架和认证模块，对应请求没有办法访问数据库或指向数据库的设置
* methods 这个请求所应用的一系列http方法，如果没指定则是所有方法
* cors 跨域资源cors参数
* csrf(boolean) 是否开启CSRF保护，默认True

1.如果表单是用python代码生成的，可通过request.csrf\_token() 获取csrf  
2.如果表单是用javascript生成的，CSRF token会自动被添加到QWEB环境变量中，通过require('web.core').csrf\_token使用  
3.如果终端可从其他地方以api或webhook形式调用，需要将对应的csrf禁用，此时最好用其他方式进行验证

## 请求

请求对象在收到请求时自动设置到odoo.http.request

### class odoo.http.WebRequest(httprequest)

所有odoo WEB请求的父类，一般用于进行请求对象的初始化

* httprequest 原始的werkzeug.wrappers.Request对象
* params 请求参数的映射
* cr 当前方法调用的初始游标，当使用none的认证方式时读取游标会报错
* context 当前请求的上下文键值映射
* env 绑定到当前请求的环境
* session 储存当前请求session数据的OpenERPSession
* debug 指定当前请求是否是debug模式
* db 当前请求所关联的数据库，当使用none认证时为None
* csrf\_token(time\_limit=3600) 为该请求生成并返回一个token（参数以秒计算，默认1小时，如果传None表示与当前用户session时间相同）

### class odoo.http.HttpRequest(\*args)

用于处理http类型请求的函数，匹配路由参数、查询参数、表格参数，如果有指定文件也会传给该方法。为防止重名，路由参数优先级最高。  
该函数的返回有三种：

* 无效值，HTTP响应会返回一个204（没有内容）
* 一个werkzeug 响应对象
* 一个字符串或unicode，会被响应对象包装并使用HTML解析

#### make\_response(data, headers=None, cookies=None)

用于生成没有HTML的响应 或 自定义响应头、cookie的html响应  
由于处理函数只以字符串形式返回html标记内容，需要组成一个完整的响应对象，这样客户端才能解析

参数：

1. data (basestring) -- 响应主体
2. headers ([(name, value)]) -- http响应头
3. cookies (collections.Mapping) -- 发送给客户端的cookie

#### not\_found(description=None)

给出404 NOT FOUND响应

#### render(template, qcontext=None, lazy=True, \*\*kw)

渲染qweb模板，在调度完成后会对给定的模板进行渲染，同时模板和qcontext可以被静态响应修改或替换

参数：

1. template (basestring) -- 用于渲染的模板
2. qcontext (dict) -- 用于渲染的上下文环境
3. lazy (bool) -- 渲染动作是否应该拖延到最后执行
4. kw -- 转发到werkzeug响应对象

### class odoo.http.JsonRequest(\*args)

处理通过http发来的json rpc格式请求

说明：  
1.method -- 忽略  
2.params -- 须是一个json格式对象  
3.处理方法返回的结果是一个json-rpc格式的，以JSON-RPC Response对象的形式组装

正确的请求：

--> {"jsonrpc": "2.0",

"method": "call",

"params": {"context": {},

"arg1": "val1" },

"id": null}

<-- {"jsonrpc": "2.0",

"result": { "res1": "val1" },

"id": null}

出错的请求:

--> {"jsonrpc": "2.0",

"method": "call",

"params": {"context": {},

"arg1": "val1" },

"id": null}

<-- {"jsonrpc": "2.0",

"error": {"code": 1,

"message": "End user error message.",

"data": {"code": "codestring",

"debug": "traceback" } },

"id": null}

## 响应

### class odoo.http.Response(\*args, \*\*kw)

响应对象通过控制器的路由传递，在werkzeug.wrappers.Response之外，该类的构造方法会添加以下参数到qweb的渲染中

1. template (basestring) -- 用于渲染的模板
2. qcontext (dict) -- 用在渲染中的上下文环境
3. uid (int) -- 用于调用ir.ui.view渲染的用户id，None时使用当前请求的id

上面的参数在实际渲染之前可以随时作为Response对象的属性修改

* render() - 渲染响应对象的模板，并返回内容
* flatten() - 强制渲染响应对象的模板，将结果设置为响应主体，并将模板复原

## 控制器

控制器像Model一样需要定义成可扩展的，但不能用同样的机制，所以采用了另外一套机制  
一般通过继承的形式创建：class odoo.http.Controller  
以route装饰器来装饰定义的方法：

class MyController(odoo.http.Controller):

@route('/some\_url', auth='public')

def handler(self):

return stuff()

为了覆盖一个controller，可以继承它并将相关方法都覆盖：

class Extension(MyController):

@route()

def handler(self):

do\_before()

return super(Extension, self).handler()

* 需要通过route装饰器来让controller方法对外部可访问，如果函数被重定义时没有route装饰，它就会对外面不可见。
* 装饰器方法是相关联的，如果覆盖的方法没有参数那么原方法参数会自动保留；如果覆盖方法有提供参数就会将原参数覆盖，如：

#下例会将handler方法由public变为user认证模式，需要登录

class Restrict(MyController):

@route(auth='user')

def handler(self):

return super(Restrict, self).handler()

译自odoo官方文档：http://www.odoo.com/documentation/10.0/reference/security.html，http://www.odoo.com/documentation/10.0/reference/testing.html ,http://www.odoo.com/documentation/10.0/reference/http.html,不当之处欢迎批评指正。

内容发布自[http://www.jianshu.com/u/6fdae8ec06bc](https://www.jianshu.com/u/6fdae8ec06bc)，转载请注明出处