

UD06. DESARROLLO DE MÓDULOS. SEGURIDAD

Sistemas de Gestión Empresarial

2 Curso // CFGS DAM // Informática y Comunicaciones

Profesor: Alfredo Oltra

**Cicles
Formatius**

ÍNDIX

1 INTRODUCCIÓN	4
2 PERMISOS	4
3 GRUPOS	6
4 PERMISOS DE REGISTRO	7
5 BIBLIOGRAFÍA	8

Versión: 240107.2159


Licencia




Reconocimiento – NoComercial – CompartirIgual (by-nc-sa). No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 **Atención.** Importante prestar atención a esta información.

 **Interesante.** Ofrece información sobre algún detalle a tener en cuenta.

1 INTRODUCCIÓN

Odoo necesita conocer que permisos tienen los usuarios/roles del sistema para cada modelo particular de nuestro módulo. Estos permisos no son ni mas ni menos que registros que se definen en la base de datos, de una forma similar a lo que hacemos con las vistas o con los datos, por lo que, por una parte, podemos definirlos de dos maneras (a través de ficheros CSV o ficheros XML) y por otra la inserción se realiza desde el apartado `data` del fichero `__manifest__.py`, donde se indica la ruta a los ficheros:

```
'data': [
    'security/ir.model.access.csv',
    'security/permissions.xml',
]
```



La ubicación de estos ficheros puede ser cualquiera dentro del módulo, pero lo recomendado para seguir las líneas de diseño de *Odoo* es que estén dentro de la carpeta *security*



A partir de ahora se va a hacer uso de muchos de los conceptos explicados en el documento de datos, donde se explica el uso de los ficheros CSV y XML para la inserción de datos

2 PERMISOS

El concepto general a tener en cuenta a la hora de definir los permisos en *Odoo* es indicar para cada modelo y grupo de usuarios que permisos tiene de lectura, escritura, creación y borrado, lo que podríamos denominar una ACL (*Access Control List*).

La forma más habitual (de hecho es la que proporciona por defecto el *scaffolding*) es definir esa lista en un fichero CSV llamado *ir.model.access*, en el que cada línea sea uno de los permisos.

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
school_access_course,school_access_course,school.model_school_course,base.group_user,1,1,1,1
```

En ese ejemplo se define:

- Una ACL con un identificador externo *school_access_course*.
- Un nombre que permita identificar al permiso (en muchas ocasiones se indica el mismo valor que el *External ID*).
- El modelo al que se aplican los permisos, en este caso *course*, indicado por su *External ID* como *school.model_school_course*.
- El grupo al que se aplica esta ACL. En este caso se indica el grupo por defecto *base.group_user* que aplica el permiso a todos los usuarios.
- Una lista de permisos (lectura, escritura, creación y borrado) donde *1* indica permiso concedido y *0* indica permiso denegado.

El problema de este fichero es que puede resultar complejo de mantener cuando el número de registros empieza a crecer. Una solución más densa, pero más clara suele ser el uso de un fichero XML.



Otra opción es trabajar el fichero con una hoja de cálculo o con alguna extensión que permita ver de manera tabulada todos los datos.

En este caso, cada una de los permisos queda definido con un registro en xml. Por ejemplo, el fichero *security/acces.xml* que equivaldría al fichero *ir.model.access* definido antes sería.

```
<record id="school_access_course" model="ir.model.access">
  <field name="name">school_access_course</field>
  <!-- otra opción es indicar el campo name con más detalle
    <field name="name">Acceso usuario general a los cursos</field>
  -->
  <field name="model_id" ref="model_school_course"/>
  <field name="group_id" ref="base.group_user"></field>
  <field name="perm_read">1</field>
  <field name="perm_create">0</field>
  <field name="perm_write">0</field>
  <field name="perm_unlink">0</field>
</record>
```



Es importante fijarse en el uso de los *external id* en cada uno de los ficheros. Por ejemplo, en el csv, el modelo se define con el patrón: *[nombre módulo].model_[nombre interno]*, donde *nombre módulo* es *school* y *nombre interno* es *school_course*, es decir, el nombre definido en la variable *_name* del modelo (*school.course*) pero cambiando los puntos por *_*. Sin embargo, en el xml no es necesario indicar el nombre del módulo



¡RECUERDA! un permiso no concedido es un permiso denegado. Si no aparece explícitamente indicado nadie podrá acceder a ninguna operación sobre el modelo, incluido visualizarlo.

3 GRUPOS

Sin la definición de grupos, la flexibilidad a la hora de definir permisos es casi nula. Para definirlos podemos optar por las dos maneras antes comentadas (en este caso el nombre del fichero csv seria *res.groups*) pero lo más habitual es crearlos via XML.

```
<odoo>
  <data>
    <record id="school.category_group" model="ir.module.category">
      <field name="name">Grupos módulo School</field>
    </record>
    <record id="school.group_TEACH" model="res.groups">
      <field name="name">Profesorado</field>
      <field name="category_id" ref="school.category_group"/>
    </record>
  </data>
</odoo>
```

En este ejemplo se crea una categoría para el grupo (no es necesario, pero ayuda a tener los grupos clasificados, sobre todo en sistemas con muchos módulos instalados) y posteriormente se crea el grupo *Profesorado* asociado a esa categoría.



Para que los permisos sean leídos correctamente por *Odoo*, la definición de los grupos tienen que cargarse desde el apartado *data* del fichero `__manifest__.py` antes que la de los permisos.



Es posible asignar usuarios al grupo en el proceso de creación de los grupos utilizando el campo *users*. Por ejemplo: `<field name="users" eval="[(4, ref('base.user_admin'))]" />` añadiríamos al grupo los usuarios del grupo administradores. Aún así, esta opción no es recomendable, siendo siempre mejor asignar el grupo al usuario en el momento de creación de este último.

Un campo que puede ser interesante es *implied_ids* que permite que los usuarios añadidos a ese grupo sean automáticamente incluidos en los referenciados. Por ejemplo, los usuarios del grupo *Coordinadores* serán incluidos automáticamente en el grupo *Profesorado*.

```
<odoo>
  <data>
    <record id="school.group_COORD" model="res.groups">
      <field name="name">Coordinadores</field>
      <field name="category_id" ref="school.category_group"/>
      <field name="implied_ids" eval="[(4,ref('school.group_TEACH'))]" />
    </record>
  </data>
</odoo>
```

4 PERMISOS DE REGISTRO

Odoo permite ir más allá y definir reglas que controlen no sólo el acceso a un modelo sino a determinados registros de ese modelo. Las reglas se almacenan en la tabla *ir.rule*, y deben definir:

- *name*, que define el nombre descriptivo de la regla.
- *model_id*, que utilizando el atributo *ref* indica el *External ID* del modelo al que hace referencia la regla.
- *groups*, que indica los *External Id* de los grupos a los que les afecta la regla.
- *domain_force*, el dominio de selección del modelo *model_id* de los registros a los que tendrán acceso los usuarios de los grupos definidos en *groups*.

```
<odoo>
<data>
  <record id="school.courses_rule" model="ir.rule">
    <field name="name">Acceso de los profesores a los cursos activos</field>
    <field name="model_id" ref="model_school_course"/>
    <field name="groups" eval="[(4, ref('school.group_TEACH'))]" />
    <field name="domain_force">[('state', 'in', ['2'])]</field>
  </record>
</data>
</odoo>
```

En este ejemplo se limita el acceso únicamente a los cursos que están activos (los que tienen el estado 2) a los profesores.

5 BIBLIOGRAFÍA

1. Documentación de Odoo