Odoo 13 Technocal Training - Udemy - HNET

Edgar Josué Benedetto Godoy

0801-1997-23600

ebenedetto@hnetw.com

edgar.benedetto@unah.hn

+504 3330-0171

12/07/2021

---

10. Crear base de datos

Ruta para administrar bases de datos en Odoo

```
localhost:8069/web/database/manager
```

---

13. Modificar una vista desde la interfaz de Odoo

- Modificar la vista de una lista
  1. Identificar el campo (field) o elemento de la vista que se quiera cambiar
  2. Clic en **Edit View List**
  3. Reconocer el elemento y realizar el cambio correspondiente
  4. Refrescar la página
- Modificar la vista de un formulario
  1. Identificar el campo (field) o elemento de la vista que se quiera cambiar
  2. Clic en **Edit View Form**
  3. Reconocer el elemento y realizar el cambio correspondiente
  4. Refrescar la página

---

20. Crear un módulo con scaffold

Formato para crear modulo con scaffold Linux

```
"path_to_odoo_bin" scaffold module name "path_to_your_proyect"
```

Formato para crear modulo con scaffold Windows

```
"path_to_python" "path_to_odoo_bin" scaffold module name "path_to_your_proyect"
```

---

21. Crear un módulo con scaffold

---

1. **Controlador**
   - Se utiliza principalmente para funcionalidades que involucren el e-commerce, es decir website
2. **Modelo:**
   - Objeto de negocio de la base de datos

---

31. Atributos en los campos

---

1. Invisible: Campo Invisible.
   `<field name="door_number" invisible="1"/>`

2. **Readonly: No se puede cambiar el valor**
   `<field name="door_number" readonly="1"/>`
3. **Required: Es un campo obligatorio**
   `<field name="door_number" required="1"/>`

Uso de `attrs`

- **Se pueden colocar condiciones para que se habilite o deshabilite un atributo**

Sentencia AND con `attrs`

- **Se utiliza `&` al inicio de la lista para denotar un AND entre 2 condiciones**

```
attrs="{'invisible':['&',('filed_name_1','=',False),('fieled_name_2','=',False)]}"
```

Sentencia OR con `attrs`

- **Se utiliza `|` al inicio de la lista para denotar un AND entre 2 condiciones**

```
attrs="{'invisible':['|',('filed_name_1','=',False),('fieled_name_2','=',False)]}"
```

Multicondición en un solo atributo con `attrs`

- **Cada tupla invoca una condición, las tuplas o condiciones se separan con una coma**

Multiples atributos en un solo `attrs`

- **Se puede hacer uso de la etiqueta `attributes` la cual también da el mismo resultado que utilizar `attrs`**

```
<field name="sale_information" position="attributes">
  <attribute name="invisible">0</attribute>
  <attribute name="attrs">
    {'invisible': [('sale_ok', '=', False)], 'readonly': [('editable', '=', False)]}
  </attribute>
</field>

attrs="{'invisible': [('sale_ok', '=', False)], 'readonly': [('editable', '=', False)]}"
```

---

39. Wizard en Odoo

- **Los asistentes describen sesiones interactivas con el usuario (o cuadros de diálogo) a través de formularios dinámicos.**
- **Un asistente es simplemente un modelo que extiende la clase en TransientModellugar de Model. La clase `TransientModel` amplía `Model` y reutiliza todos sus mecanismos existentes, con las siguientes particularidades:**

1. **Los registros del asistente no deben ser persistentes; se eliminan automáticamente de la base de datos después de cierto tiempo. Por eso se les llama transitorios**
2. **Los modelos de asistente no requieren derechos de acceso explícitos: los usuarios tienen todos los permisos sobre los registros del asistente**
3. **Los registros de asistente pueden hacer referencia a registros regulares o registros de asistente a través de campos many2one, pero los registros regulares no pueden referirse a registros de asistente a través de un campo many2one**

Lanzamiento de asistentes o wizards

Los asistentes se inician por `ir.actions.act_windowregistros` , con el campo `target` establecido en el valor `new` . Este último abre la vista del asistente en una ventana emergente. La acción puede desencadenarse mediante un elemento del menú.

Hay otra forma de iniciar el asistente: utilizando un `ir.actions.act_window` registro como el anterior, pero con un campo adicional `src_modelque` especifica en el contexto de qué modelo está disponible la acción. El asistente aparecerá en las acciones contextuales del modelo, encima de la vista principal. Debido a algunos ganchos internos en el ORM, dicha acción se declara en XML con la etiqueta `act_window` .

```
<act_window id="launch_the_wizard"
            name="Launch the Wizard"
            src_model="context.model.name"
            res_model="wizard.model.name"
            view_mode="form"
            target="new"
            key2="client_action_multi"/>s
```

43. Crear botón inteligente

1. **Generar en la vista el espacio correspondiente al botón**
   - **El esqueleto de los botones inteligentes es el siguiente**
   - **En `name` se debe colocar el nombre de la función**

```
<button class="oe_stat_button" name="get_cars">

</button>

<div name="button_box" position="inside">
    <button class="oe_stat_button" type="object" name="get_cars" icon="fa-car">
        <field string="Car" name="car_count" widget="statinfo"/>
    </button>
</div>
```

1. **Crear la variable que contenga al campo de la vista en el modelo**

```
car_count= fields.Integer(string="Count")
```

2. **Crear el método para obtener los carros en vista de lista, el cual se menciona en el botón inteligente**

```
def get_cars(self):
  self.ensure_one()
  return {
    'type':'ir.actions.act_window',
    'name':'Cars',
    'view_mode':'tree',
    'res_model':'flet.vahicle'
    'domain':[('driver_id','=',self.id)],
    'context':"{'create':False}"
  }
```

1. **Hacer del campo incluido en el botón inteligente, un campo calculado para mostrar el numero de carros a los que el usuario actual es el driver**

```
def get_car_number(self):
  for rec in self:
    rec.car_count = self.env['fleet.vehicle'].search_count([('driver_id','=',self.id)])
```