DIAGRAM SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROGRAMACIÓN DE OPENERP/ODOO

1.3. VISTAS

José Zambudio Bernabeu
jose.zambudio@diagram.net

NubeaERP global cloud solutions

INTRODUCCIÓN VISTAS

- Tipos de elementos
  1. Business object
     - Clases de python las cuales extienden de la clase orm.Model. La gestión en base de datos de dichos módulos la mantiene el ORM.
  2. Data
     - Vienen dados por ficheros XML o CSV, y nos proporcionan todos aquellos datos que serán introducidos en base de datos. Entre estos tenemos: la declaración de vistas, workflows, datos demo, datos de configuración…
  3. Reports
     - Ficheros RML, HTML/MAKO o templates de OpenOffice con los que junto con la información de nuestro módulo, utilizaremos para generar informes PDF, HTML o ODT.

CREANDO MENUS

- Los menús son registros para el modelo ir.ui.menu
- Como cualquier registro podemos definir un record en un xml:

```
<record id="record_menu_ejemplo" model="ir.ui.menu">
    <field name="name">Mi menu</field>
    <field name="action" ref="id_del_action_a_ejecutar"/>
    <field name="sequence" eval="5"/>
    <field name="parent_id" ref="id_del_record_del_menu_padre"/>
</record>
```

- Pero es recomendable utilizar el shortcout que OpenERP nos habilita:

```
<menuitem id="record_menu_ejemplo"
    name="Mi menu"
    action="id_del_action_a_ejecutar"
    sequence="5"
    parent="id_del_record_del_menu_padre"
    groups="listado_de_grupos_con_permiso"/>
```

CREANDO MENUS

- @id: Obligatorio
  - Especificamos el id (debe ser único)
- @name:
  - Nombre del menú a mostrar en la interfaz del usuario. Por defecto aplica el nombre del @action.
- @action:
  - Especificamos el id del action a ejecutar (ir.actions). No es obligatorio.
- @groups:
  - Especificamos cada uno de los grupos de usuarios que pueden ver el menu. Los separamos por ‘,’.
  - Si no lo especificamos indicamos que todos los grupos pueden verlo.
- @sequence:
  - Integer. Se utiliza para ordenar los menús.
  - En caso de no especificarse, OpenERP establece por defecto 10. En caso de tener el mismo sequence OpenERP los ordena por id (orden de creación)
- @parent:
  - Especificamos la jerarquía, el id del menu padre del que vamos a depender.

CREANDO ACTIONS

- Los actions definen el comportamiento del programa para responder ante las acciones de un usuario, por ejemplo accionar un menuitem, “Imprimir”, “Más”…
- Podemos definir varios tipos de actions:

Ventanas	*Informes*	*Ejecución*
ir.actions.act_window	*ir.actions.report.xml*	*ir.actions.act_url*
~~*ir.actions.wizard*~~		*ir.actions.server*
ir.actions.act_window_close		*ir.actions.client*

CREANDO ACTIONS - WINDOW

- ir.actions.act_window

```
<record model="ir.actions.act_window" id="id_de_nuestro_action">
    <field name="name">nombre.de.nuestro.action</field>
    <field name="res_model">modelo</field>
    <field name="view_type">form|tree</field>
    <field name="view_mode">tree,form,calendar...</field>
    <field name="view_id" ref="id_vista_por_defecto"/>
    <field name="domain">[('total', '=', '0')]</field>
    <field name="context">{'nuestro_flag': True}</field>
</record>
```

CREANDO ACTIONS - WINDOW

- ir.actions.act_window
  - **name**: Nombre del action.
  - **view_id**: ID de la vista (ir.ui.view) a mostrar cuando el act_window se active.
  - **domain**: Lista de constraints para filtrar los registros a mostrar.
  - **context:** Diccionario (Por defectos dinamicos, flag...). {}
  - **res_model**: Modelo al que vincular el action.
  - **target**: [current, new, inline, inlineview]
  - **view_type**: [tree, form]
  - **view_mode**: Sólo si view_type = form, establecemos los tipos de vistas que tendrá nuestro modelo. tree,form
  - **limit**: 80
  - **auto_refresh**: 0
  - **search_view:** ID de la vista search (ir.ui.view)

INTRODUCCIÓN VISTAS

- Las vistas describen la forma por la que se va a presentar los registros de los modelos a los usuarios.
- Más específicamente… Podemos definir un (o muchos) vistas, para un mismo modelo.
- Estas vistas pueden ser o no del mismo tipo.
- Tipos de vistas básicas:
  - Form
  - Tree
- Más tipos de vistas:
  - Calendar
  - Graph
  - Diagram
  - Gantt
  - Kanban
  - Search

INTRODUCCIÓN VISTAS

- @model: Para definir una vista debemos realizar un record al modelo “ir.ui.view”
- @id: Por convención “view_*_[tree|form…]”
  - <field name=“name”>…</field>
    - Indicamos un name para la vista. Por convención mismo que @id cambiando ‘_’ por ‘.’
  - <field name=“inherit_id” ref=“id_vista”/>
    - Heredamos de una vista…
  - <field name=“model”>…</field>
    - Modelo de los records que va a mostrar la vista.
  - <field name=“field_parent”>…</field>
    - Indicamos a la vista que el contenido debe representar el contenido siguiendo su jerarquía.

INTRODUCCIÓN VISTAS

- <field name="groups_id" eval="[(6, 0, [ref('record_group_id')])]" />
  - ACLs…
- <field name="priority">…</field>
  - Orden para seleccionar las vistas
  - Por defecto 16.
  - _order = "priority,name"
- <field name="arch" type="xml">…</field>
  - Contenido de la vista. Aquí establecemos el tipo y el estilo.
    - <tree…></tree>
    - <form…></form>
    - …

INTRODUCCIÓN VISTAS

- Atributos generales
  - create
    - Determinamos si se pueden crear nuevos registros desde dicha vista.
    - default = 1 ~ true
  - delete
    - Determinamos si se pueden borrar registros desde dicha vista.
    - default = 1 ~ true
  - edit
    - Determinamos si se pueden modificar registros desde dicha vista.
    - default = 1 ~ true
  - string
    - Nombre de la vista.

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Field tag: <field [attributos]/>**
    - Definimos la columna a mostrar en la vista lista.
    - Atributos:

attrs	*Modificaciones al resto de atributos vía una condición previa.*
string	*Modificamos el label definido para el field.*
width	*No funciona en los tree. Para el form indica el ancho del campo.*
on_change	*Trigger que ejecuta un método del servidor cuando el valor del field es modificado.*
filter_domain	*Search.* *Permite modificar completamente el domain, sobre-escribiendo al del action.*

INTRODUCCIÓN VISTAS

invisible	*Indica que el field va a estar oculto en el navegador.*
password	*Reemplaza el contenido del valor del field por "*"*
sum	*Tree. Indicamos que el campo contiene el sumario de todas las líneas. Lo pinta en "left bottom"*
avg	*Mismo que sum pero "average" (promedio, media…)*
select	*1 -> búsqueda básica. 2 -> búsqueda avanzada. Por defecto 1.*
groups	*Indicamos para que grupos de usuarios será visible.*
operator	*Reemplaza el operador por defecto para el field.*
colspan	*Número de columnas que debe utilizar el field. Por defecto: 4.*

INTRODUCCIÓN VISTAS

nolabel	*No muestra el label del field. Por defecto = 1 a no ser que indiquemos el field dentro de un group.*
required	*Indica que para poder modificar/crear un registro, dicho campo debe estar introducido.*
readonly	*Indica que el field es de sólo lectura.*
domain	*Restringe vía un filtro el contenido que va a mostrar el field.*
context	*"flags", "default_", "form_view_ref", "tree_view_ref", "group_by"…*
states	*Estados en los que va a ser visible el field.*
digits	*Mismo uso que en la declaración del field en el model.*
icon	*openerp/tools/misc.py:608 - 644 addons/web/src/img/icons*

INTRODUCCIÓN VISTAS

mode	*Sequence de las vistas cuando se can a ver.*
filename	*Para binary, nombre por defecto del fichero a descargar.*
height	*No funciona en los tree. Para el form indica el alto del campo.*
default_focus	*Al inicio de un formulario, se hace focus a aquel field con dicho campo activo.*
statusbar_visible	*Mismo caso que states. Indicamos los estados en los que la barra será visible.*
statusbar_colors	*Especificamos los colores que tendrá para cada estado: "blue" | "red"*
options	
placeholder	*Mismo uso que en html…*

INTRODUCCIÓN VISTAS

group	
widget	*Indica el widget que va a utilizar el field para mostrar el contenido:* • *one2many_list* • *many2many_tags* • *many2many_kanban* • *many2many_binary* • *many2onebutton* • *selection* • *statusbar* • *handle* • *monetary* • *mail_followers* • *mail_thread* • *many2many_tags_email* • *char* • *float_time* • *html* • *url* • *one2many* • *email* • *integer* • *progressbar* • *image* • *pad* • *date*

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Button tag: <button [attributos]/>**
    - Definimos la columna a mostrar en la vista lista.
    - Atributos:

attrs	***Modificaciones al resto de atributos vía una condición previa.***
invisible	***Indica que el botón va a estar oculto en el navegador.***
name	***Id del action - nombre del método - nombre del signal -> Ver @type.***
icon	***Muestra un icono sobre el botón. Si no se indica el botón es de sólo texto.***
string	***En vistas tree los botones no tienen label. En los form indica el label a mostrar.***

INTRODUCCIÓN VISTAS

states	*Estados en los que va a ser visible el botón.*
confirm	*Salta un alert del navegador:* *js: confirm("Press a button!");*
special	*Único valor disponible: "cancel". Indica que el popup debe cerrarse sin realizar ninguna acción.*
default_focus	*Al inicio de un formulario, se hace focus a aquel field con dicho campo activo.*
colspan	*Número de columnas que debe utilizar el botón.*
target	*En caso de @type="action" define el target del action definido.*
readonly	*Muestra el botón como disable.*
context	*"flags", "default_", "form_view_ref", "tree_view_ref", "group_by"…*

INTRODUCCIÓN VISTAS

help	***Al igual que el field en la definición del modelo, podemos indicar al usuario una ayuda del funcionamiento del botón.***
class	***Añadimos una clase css.***
type	***Define el tipo de acción a realizar:*** • ***workflow (default)**: Manda una señal (signal) para el modelo actual utilizando el @name del botón como nombre de esta.* • ***object**: El botón ejecuta un método del modelo actual utilizando el @name del botón como nombre de este.* • ***action**: El botón ejecuta un action (ir.actions) con id indicado en el @name de este.*

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Group tag: <group [attributos]/>**
    - Utilizado tanto en vistas “form,search”
    - Agrupa los fields que contiene.
    - Atributos:

Expand	***En las vistas search, podemos agrupar fields bajo un icono de agrupado. 1 (expandido - default), 0 (agrupado)***
String	***Label para el grupo.***
attrs	***Modificaciones al resto de atributos vía una condición previa.***
colspan	***Número a columnas a usar.***
col	***Número de columnas a generar (internas)***

INTRODUCCIÓN VISTAS

rowspan	*Número de filas a utilizar.*
states	*Estados en los que va a ser visible el grupo.*
height	*Indica el alto del grupo.*
width	*Indica el ancho del grupo.*
name	
color	
invisible	*Indica que el grupo va a estar oculto en el navegador.*

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Separator tag: <separator [attributos]/>**
    - Utilizado tanto en vistas “form,search”
    - Atributos:

invisible	***Indica que el separador va a estar oculto en el navegador.***
name	
colspan	
rowspan	
string	

INTRODUCCIÓN VISTAS

col	
select	
orientation	

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Label tag: <label [attributos]/>**
    - Utilizado tanto en vistas “form,search”
    - Atributos:

string	***Especificamos el texto a mostrar.***
for	***Muestra el label del field con @name que se indica en este campo.***
attrs	
invisible	
align	

INTRODUCCIÓN VISTAS

nolabel	
colspan	
angle	
fill	
help	
width	
wrap	
name	

INTRODUCCIÓN VISTAS

- Elementos generales
  - **Newline tag: <newline/>**
    - Fuerza un salto de línea incluso si no se han rellenado todas las columnas.

INTRODUCCIÓN VISTAS

nolabel	
colspan	
angle	
fill	
help	
width	
wrap	
name	

TREE VIEWS

- Las utilizamos para mostrar información en forma de lista.
- Son las vistas más simples.

```
<record id="view_invoice_line_tree" model="ir.ui.view">
    <field name="name">account.invoice.line.tree</field>
    <field name="model">account.invoice.line</field>
    <field name="arch" type="xml">
        <tree string="Invoice Line">
            <field name="name"/>
            <field name="account_id" groups="account.group_account_user"/>
            <field name="quantity"/>
            <field name="uos_id" groups="product.group_uom"/>
            <field name="price_unit"/>
            <field name="discount" groups="sale.group_discount_per_so_line"/>
            <field name="price_subtotal"/>
        </tree>
    </field>
</record>
```

- En el contenido del la vista vamos añadiendo los fields a mostrar. Estos fields se irán mostrando por columnas.

TREE VIEWS

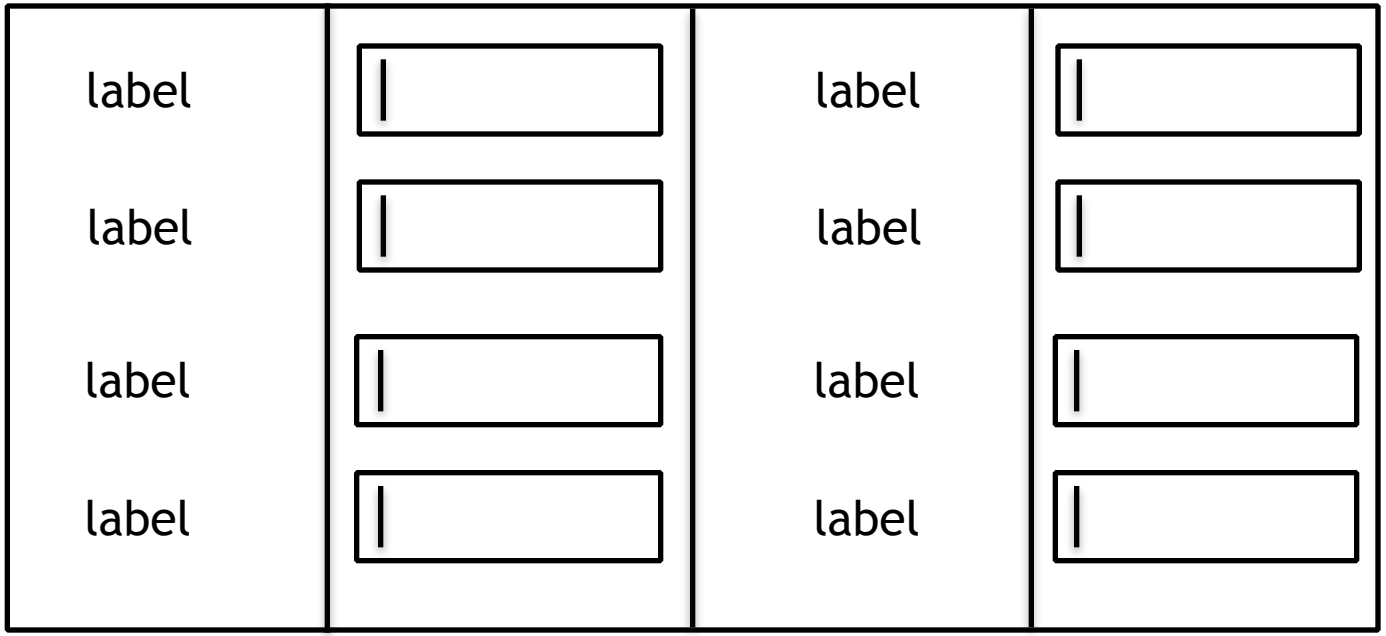
- Atributos:
  - ...generales... más...
  - colors:
    - colors="[colour]:[condition];"
    - default: colors="black:True"
    - ejemplo: colors="red:state == 'cancelled';blue: state != 'cancelled' and total > 0;"
  - fonts:
    - fonts="**bold**:message_unread==True"
  - editable:
    - Opciones: "top" | "bottom"
  - toolbar:
    - Opciones: "1" | "True"
    - Mostrar acciones "Print", "Más"

TREE VIEWS

- Elementos:
  - **<field [attributos]/>**
    - Definimos la columna a mostrar en la vista lista.
    - Atributos:
      - **name**=“” —> Nombre del field a mostrar.
      - **attrs**=“” —> mismo funcionamiento que states en el field
        - attrs=“{‘readonly’:[(‘is_company’, ‘=’, True)]}”
      - **string**=“” —> Especificamos nombre de la columna. Por defecto tiene el string especificado en el field del modelo.
      - **width**=“” —> Podemos especificar anchos de columna.
      - **on_change**=“” —> …siguientes diapositivas…
      - **domain**=“” —> Filtro para la selección de datos.
      - **invisible**=“” —> Ocultar columnas.
      - password=“” —> Mostramos **** en vez del valor.
      - sum=“” —>
      - avg=“” —>
      - required=“” —>
      - readonly=“”—>
  - <button [attributos]/>

FORM VIEWS

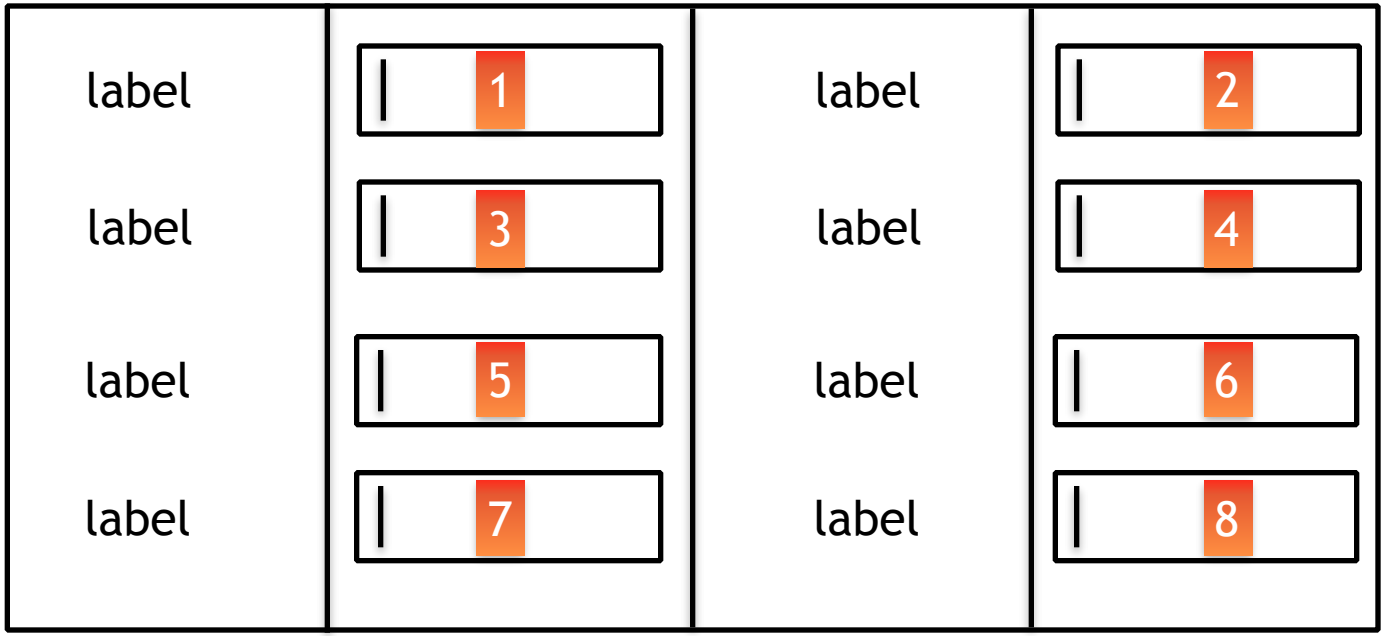
- Disposición:
  - Por defecto…



  - 4 columnas, label - input + label - input

FORM VIEWS

- Disposición:
  - Por defecto…



  - Los fields se van añadiendo desde la izquierda a la derecha, y de arriba a abajo

FORM VIEWS

- Disposición:
  - Por defecto…

  - Un field puede utilizar varias columnas.