1、 控制显示和隐藏

invisible 是一个隐藏属性，**'true/false'** 则控制隐藏/显示，也可以使用 **True/False**

```
attrs="{'invisible': [('state', '!=', 3)]}"
```

当字段 **state** 满足条件 **state != 3** 时，隐藏

```
attrs="{'invisible': [('state', 'not in', (3, 4))]}"
```

当需要判断的值有多个时，可以使用元组，并使用 **in/not in** 判断字段值是否在元组范围内

```
attrs="{'invisible': [('state', '!=', 4), ('state', '!=', 5)]}"
attrs="{'invisible': ['|', ('state', '!=', 4), ('state', '!=', 5)]}"
```

当然，可以可以使用多个判断条件，这样默认是 **与** 操作，加上 **'|'** 则是 **或**，但是这样远没有使用元组来的方便

2、控制字段只读

readonly 是控制只读的属性，只对 **field** 生效，不能用在 **group** 上

```
<field name="desc" attrs="{'readonly':True}/>
```

表示字段 **desc** 只读

```
<field name="desc" attrs="{'readonly': [('state', 'not in', (4, 5))]}"/>
```

同样，也可以使用判断逻辑

3、默认打开form视图

action中会根据顺序显示，第一个则是默认视图

```
<record id="test_view" model="ir.actions.act_window">
 <field name="name">测试视图</field>
 <field name="res_model">test.data</field>
 <field name="view_type">form</field>
 <field name="view_mode">form,tree,search</field>
</field>
```

如上，示例中的 **view_mode** 表示打开默认显示 **form**

如果改成下面这样，则默认显示 **tree** 视图

```
<field name="view_mode">tree,form,search</field>
```

4、视图内的编辑和创建按钮

可以使用 **form_no_edit** 控制form视图是否可以编辑（与创建无关）

```
<record id="test_view" model="ir.actions.act_window">
 <field name="name">测试视图</field>
 <field name="res_model">test.data</field>
 <field name="view_type">form</field>
 <field name="view_mode">form,tree,search</field>
   <field name="context">
 {"form_no_edit": [('state', '!=', 'new')]}
 </field>
```

```
</field>
```

示例中，**form_no_edit** 可以使用判断语句，也可以直接修改值 **"true/false"**

另外，也可以直接的视图上使用 **edit** 和 **create** 属性修改

```
<form string="隐患处理数据通用视图" create="false" edit="false">
  ...
</form>
<tree string="隐患处理数据列表视图" edit="false" create="false">
</tree>
```

5、只显示列表，点击不触发数据的form视图

修改action的 **view_type** 即可

```
<record id="test_view" model="ir.actions.act_window">
 <field name="name">测试视图</field>
 <field name="res_model">test.data</field>
 <field name="view_type">tree</field>
 <field name="view_mode">form,tree,search</field>
</field>
```

示例中，**view_type** 值为tree，则打开后，值显示列表，点击数据行不会跳转

6、xxx2Many字段删除创建并编辑选项

一对多和多对多的字段，在下拉选择时，默认下方会出现 **创建并编辑** 的选项

在 **field** 的选项中的 **no_create_edit** 选项设置为 **True** 即可

```
<field name="task_id"  options="{'no_create_edit': True}"/>
```

7、菜单

使用 **menuitem** 定义菜单栏

```
<menuitem id="main_menu" name="主菜单"/>
<menuitem id="second_menu" name="二级菜单" parent="main_menu"/>
<menuitem id="third_menu" name="三级菜单" sequence="1" parent="se
cond_menu" action="action_view"/>
<menuitem id="third_menu_2" name="三级菜单2" sequence="2"
parent="second_menu" action="test_action_view"/>
<menuitem id="second_menu_2" name="二级菜单2" parent="main_menu"/
>
```

说明：

- 没有设置 **parent**，则默认是主菜单，会显示在导航栏中，以下的菜单项可以通过指定 **parent** 设置层级关系；
- **name** 表示菜单的显示名称；
- **sequence** 指定同一级菜单的顺序；

8、同一个数据模型，多个视图

有些时候，我们需要针对一个数据模型，显示多个不同的视图

```
<record id="action_view" model="ir.actions.act_window">
<field name="name">主action</field>
<field name="res_model">test.data</field>
<field name="view_ids" eval="[(5, 0, 0),
 (0, 0, {'view_mode': 'tree', 'view_id': ref('main_tree_vie
w')}),
 (0, 0, {'view_mode': 'form', 'view_id': ref('main_form_vie
w')})]"/>
</record>
```

如上例，如果 **menuitem** 触发该action，则会显示指定的视图main_tree_view和main_tree_view

9、必填字段

必填字段可以再py文件中限定，也可以在xml中指定

```
field = fields.Char(string=u'名称', arequired=True)
```

安装后，则会在数据库层面设置此字段必填

因为是修改数据库，所以这里的修改需要卸载重装才能生效

```
<field name="hazard_device" required="1"/>
```

此时，会在前端视图层面限制字段必填

10、视图中的字段不显示标签

默认显示字段的时候，会在字段前面显示该字段的名称，也可以使用 **nolabel** 设置是否显示

```
<field name="location_city" nolabel="1"/>
```

11、视图中的tab

用于在视图中使用tab标签翻页

```
<notebook>
 <page string="检测数据">
```

```
<field name="name"/>
<field name="desc"/>
</page>
<page string="检测环境">
<field name="price"/>
<field name="wind"/>
</page>
</notebook>
```

- **notebook** 表示显示一个tab控件，内部则是需要显示的页；
- **page** 表示一页，**string** 表示该页显示的标题；

12、根据时间自动生成序号

py文件

```
order = fields.Char(string=u'任务单编号', default=u'保存时自动生成')

@api.model
def create(self, vals):
 if not vals.get('order'):
 vals['order'] = self.env['ir.sequence']
 .next_by_code('test.data')
 result = super(ElectricityProtectTask, self).create(vals)
 return result
```

xml文件

既然是自动生成，则视图中需要设置为只读

```
<field name="task_order" attrs="{'readonly':True}"/>
```

13、在视图中嵌入自定义页面

使用 **ir.actions.client** 即可

```
<record id="info_manager_action_view" model="ir.actions.client">
 <field name="name">信息管理</field>
 <field name="tag">web_report_ext.main</field>
 <field name="context">
 {'search_default_url':
 'https://map.baidu.com/search/ditu/@13444421,3648634,13z'}
```

```
 </field>
</record>
<menuitem id="info_manager" name="信息管理" sequence="3"
parent="menu" action="info_manager_action_view"/>
```

14、过滤数据

（1）视图中过滤显示

可以直接在action中使用domain过滤数据

```
<record id="action_view" model="ir.actions.act_window">
 <field name="name">数据</field>
 <field name="res_model">test.data</field>
 <field name="domain">[('state','in', (1, 2))]</field>
</record>
```

（2）多对一下拉框过滤

在字段中使用domain过滤

```
type_id = fields.Many2one('test.type', string='类型',
domain=['|', ('name', '=', '类型1'), ('name', '=', '类型2')])
```