Owl

Creazione Componente nel Pos

per poter creare un componente nel pos di odoo 14 bisogna eseguire questi passaggi:

1. andare in `static/src/js` e creare un nuovo file js che si chiamerà `EpsonEPOSWidget.js` all'interno inserire questo codice `js`

```
    var PosComponent = require('point_of_sale.PosComponent');
    var Registries = require('point_of_sale.Registries'); // questa variabile di
permette di accedere a tutti i metodi per registrare e estendere i componenti

    class EpsonEPOSWidget extends PosComponent {
        async onClick() {
                    $('.epson-fp81ii-widget').removeClass('oe_hidden');
            } // questo componente contente un event onClick
    }


    EpsonEPOSWidget.template = 'EpsonEPOSWidget'; // il nome del template deve
essere uguale al nome del componente in questo caso il nome della classe

    Registries.Component.add(EpsonEPOSWidget); // questo metodo permette di
registrare il componente e di renderlo disponibile nel APP

    return EpsonEPOSWidget;
```

2. andare in `static/src/xml` e creare un nuovo file `xml` che deve avere lo stesso nome del file precedentemente creato `EpsonEPOSWidget.xml`. Questo file al suo interno deve avete questo blocco di codice.

```
<?xml version="1.0" encoding="UTF-8"?>
<templates id="template" xml:space="preserve">

    <t t-name="EpsonEPOSWidget" owl="1">
        <div class="header-button" t-on-click="onClick"> <!--  per eseguire eventi
sui componenti bisogna esguire la sintassi del liguaggio Qweb -->
            <img src="/fiscal_epos_print/static/src/img/epos.svg" alt="Epson ePOS"
title="Epson ePOS Operation" />
        </div>
    </t>
</templates>
```

N.B. sia i file xml e che js si devonono chiamare uguale, si devono trovare nello stesso posto (vedi punto 1 e 2) quindi per esempio se creo un componente in questo path `static/src/js/Component/EpsonEPOSWidget.js`, il file xml deve seguire lo stesso percorso del file js `static/src/xml/Component/EpsonEPOSWidget.xml` a livello di struttura delle cartelle

3. inserie i file js nel file `views/assets.xml`

```
<!-- views/assets.xml -->

<?xml version="1.0" encoding="utf-8"?>
<odoo>

    <template id="assets" inherit_id="point_of_sale.assets">
        <xpath expr="." position="inside">
            <script type="text/javascript"
src="/fiscal_epos_print/static/src/js/EpsonEPOSWidget.js"/>
        </xpath>

    </template>

</odoo>
```

e i file xml creati nel `__manifest__.py` del modulo di Odoo 14

```
#__manifest__.py

'data': [
    'views/assets.xml',
        ],
'qweb': [
    'static/src/xml/EpsonEPOSWidget.xml'
]
```

4. infine il template del componente deve essere chiamato nella pagina `static/src/xml/pos.xml` per essere mostrato nel frontend del Pos

```
    <!-- views/pos.xml -->

    ?xml version="1.0" encoding="UTF-8"?>
    <templates id="template" xml:space="preserve">

        <t t-name="Chrome" t-inherit="point_of_sale.Chrome" t-inherit-
mode="extension" owl="1">

            <xpath expr="//MobileMode" position="after">
                <EpsonEPOSWidget/>
            </xpath>

        </t>

    </template>

    </odoo>
```

Metodi per registrare e estendere i componenti

tutti i metodi per gestire la registrazione o l'estensione dei componenti sono presenti nel file

`/src/odoo/addons/point_of_sale/static/src/js/ClassRegistry.js`

- aggiungere un **costruttore** alla classe del componente

```
    constructor() {
        //inserire i parametri o campi che la classe deve inizializzare
    }
```

- registrare un componente creato da zero

```
      Registries.Component.add(baseClass);
```

- aggiungere un nuovo **componente classe** nel registro basato su una altra **classe**

```
      Registries.Component.addByExtending(baseClassExtend, baseClass);
```

- aggiunge un nuovo **componente estendendo** tutte le **funzionalità** di un altra **classe**

```
Registries.Component.extend(baseClassse, baseClassExtend)
```

- ritorna il **componente** contente tutte le **estensioni** basate su una classe

```
Registries.Component.get(baseClass)
```