Sistemas de Gestión Empresarial

UD 08. Cliente Web de Odoo

Actualizado Septiembre 2021

Licencia

Nomenclatura

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

📖 Importante

❕ Atención

💬 Interesante

ÍNDICE DE CONTENIDO

UD08. CLIENTE WEB ODOO

1. INTRODUCCIÓN

> ! A dia de hoy (Febrero de 2021), Odoo está en la versión 14 en la que conviven dos frameworks de javascript diferentes: el antiguo de Odoo hasta la versión 13 y Owl que es un framework propio similar a react o vue.js. Como estamos en esa transición, esta unidad estará incompleta hasta futuras versiones en las que será realmente interesante explicar Owl. No obstante, hay cosas interesantes que no cambiarán demasiado.

En anteriores unidades hemos modificado el cliente web del backend de Odoo creando o modificando las vistas declaradas en XML. También hemos hecho algo un poco más avanzado con el QWeb en los reports o en los Web Controllers o incluso dentro de las vistas kanban.

Odoo recomienda utilizar las vistas tree, form, kanban… para todo, pero en ocasiones necesitamos ir un poco más allá y modificar la vista a un nivel inferior.

Antes de entrar en detalles sobre Odoo hay que hacer una breve explicación sobre el tipo de cliente web que es.

Las páginas web más simples son estáticas con hiperenlaces entre documentos HTML. La siguiente evolución a este tipo de web son las generadas desde el servidor ya sea con PHP, Java, Python, Node… Estas son iguales de cara al navegador web. Para darles un poco más de usabilidad, se incorpora un poco de Javascript en el cliente para validar formularios o animaciones. El gran cambio en el cliente web se produce con las técnicas de AJAX (Javascript asíncrono y XML). Ahora el cliente puede pedir fragmentos de su HTML al servidor y sustituirlos de forma dinámica sin refrescar el navegador. El ejemplo más completo de esta manera de hacer webs son las SPA (Single Page Application) que es lo que es Odoo.

En Odoo, al refrescar la web, se carga un HTML muy básico con unos pocos enlaces a archivos .js. Al cargar y ejecutar este javascript, comienza a pedir el resto de la aplicación web y la genera totalmente en el lado del cliente. A partir de ese momento, la comunicación con el servidor se limita a pedir datos en JSON y vistas en XML. Esas peticiones son mucho más ligeras que el contenido HTML que finalmente genera para el usuario.

Hacer una SPA simple con JS Vanilla no es demasiado complicado, pero la complejidad de Odoo y el hecho de poder definir las vistas por XML, hace necesario utilizar un Framework. El de Odoo tiene una estructura MVC como el propio Odoo.

El objetivo de esta unidad es hacer módulos que añadan funcionalidades al cliente web ya existente de Odoo. Tanto el backend como la web o el terminal punto de venta. Lo que veremos será cómo modificar estilos, crear y añadir elementos HTML o crear Widgets, Componentes o Vistas.

Empezaremos entendiendo cómo funciona el cliente web. A continuación haremos modificaciones al cliente más superficiales para acabar creando nuestros própios widgets y vistas.

2. ARQUITECTURA DEL CLIENTE WEB

Cuando creamos un módulo para Odoo en anteriores unidades, creamos ficheros Python que definian el modelo y controlador y XML que definían plantillas para la vista. Los módulos que modifican el **WebClient** de Odoo tienen archivos XML para QWeb, Javascript y CSS entre otros. El cliente web tiene internamente también una arquitectura MVC de forma análoga al própio Odoo completo.

El modelo en el cliente web es la gestión de las peticiones AJAX al servidor y los datos resultantes. El controlador es la manipulación de esos datos y la vista la renderización de los mismos. Cada componente (Widget en Odoo) de la interfaz suele tener un archivo Javascript para cada parte de esta arquitectura.

> 💬 Se puede obviar toda esta arquitectura, al fin y al cabo es Javascript. Pero sería un error no respetarla porque está pensada para facilitar la comunicación con el servidor y reducir los errores.

Los módulos principales del cliente web parten del módulo **web**, que proporciona el núcleo. Los otros módulos lo complementan. Hay módulos que lo amplían como, por ejemplo, **web_kanban**. Pero en realidad cualquier módulo lo puede hacer. Si estamos haciendo un módulo que necesita un widget específico, podemos ampliar el cliente web desde este módulo y ese widget lo podremos utilizar en otros módulos. Para hacerlo, debemos poner todo esto en el directorio static del módulo:

- static/src/js
- static/src/css
- static/src/xml : Los ficheros de plantillas.
- static/img : Las imágenes
- static/libs : Las librerías estáticas de javascript que necesita el módulo.

El servidor no manipula estos ficheros, pero los procesa y los envía al cliente.

3. ENVÍO DEL CLIENTE WEB AL NAVEGADOR

Pensemos que cada módulo puede tener, potencialmente, un módulo de cliente web. Esto son muchos ficheros javascript, css y demás. El envío de muchos ficheros supone un problema en HTTP, ya que tiene que establecer la conexión muchas veces y eso es muy lento. Para evitar saturar la red, el servidor hace una compresión de los datos de la siguiente manera:

- Todo el CSS y Javascript es concatenado en un sólo fichero. La concatenación se hace ordenando según las dependencias entre módulos.
- El JS es minimizado quitando espacios y refactorizando las variables por nombres más cortos.
- Se envía un HTML básico con los enlaces a estos CSS y JS.
- Todo esto se comprime con gzip para enviar menos por la red. El navegador es capaz de descomprimir.

3.1 ASSETS

Para saber todo lo que el servidor debe enviar al cliente se necesita saber qué quiere enviar cada módulo. Además, es posible que un módulo sea sólo para el cliente web del backend o de la página web pública. Por eso Odoo tiene tres colecciones de links JS y CSS en lo que llama **bundles**. Estos son:

- **web.assets_common**: Cosas comunes.
- **web.assets_backend**: Cosas del Backend.
- **web.assets_frontend**: Cosas de la web pública.

Estos bundles están hechos en XML y guardados en la base de datos. Para añadir algo a ese XML se puede usar la misma técnica que con la herencia en la vista:

```
<template id="assets_backend" name="school assets" inherit_id="web.assets_backend">
    <xpath expr="//script[last()]" position="after">
        <link rel="stylesheet" href="/school/static/src/less/school.less"/>
        <script type="text/javascript" src="/school/static/src/js/school.js"></script>
    </xpath>
</template>
```

4. AÑADIR CSS A NUESTRO MÓDULO

Veamos un ejemplo de lo visto anteriormente pero sólo con el CSS, ya que es mucho más sencillo. Para ello vamos a ponerle una clase al tree de nuestro módulo para que la letra sea más pequeña y quepan más columnas. Lo primero será hacer el CSS en /<modulo>/static/src/css/<modulo>.css:

```
 .school_tree { font-size: 0.8em; }
```

A continuación añadiremos una referencia ese archivo en el bundle de **assets_backend**

```
<template id="assets_backend" name="school assets" inherit_id="web.assets_backend">
    <xpath expr="//script[last()]" position="after">
     <link rel="stylesheet" href="/school/static/src/css/school.css"/>
    </xpath>
</template>
```

Y por último podemos usar la clase CSS:

```
<field name="courses" limit="10" class="school_tree">
```

5. AÑADIR FRAGMENTOS DE HTML AL MÓDULO

6.

7. BIBLIOGRAFÍA

- Sistemes de Gestió Empresarial IOC: https://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M10/web/html/index.html
- Wikipedia: https://es.wikipedia.org/wiki/Sistema_de_planificaci%C3%B3n_de_recursos_empresariales
- Documentación de Odoo: https://www.odoo.com/documentation/

8. AUTORES (EN ORDEN ALFABÉTICO)

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento.

- Jose Castillo Aliaga
- Sergi García Barea