Gong - Errores #2833

Error en listado de gastos con plugin CPT_CONTABILIDAD no activo

2017-02-17 13:20 - Santiago Ramos

Estado:	Resuelta	**Fecha de inicio:**	2017-02-17
Prioridad:	Normal	**Fecha fin:**	
Asignado a:	Jaime Ortiz	**% Realizado:**	0%
Categoría:		**Tiempo estimado:**	0.00 hora
Versión prevista:	2.54		

Descripción

Al sacar un listado de gastos de agente, se produce un error 500 si el plugin CPT_CONTABILIDAD no esta activo:

```
 NoMethodError in Gasto_agentes#listado

Showing /srv/devel/gong/gor/app/views/gasto_agentes/_gasto.html.erb where line #10 raised:

undefined method `activo' for nil:NilClass

Extracted source (around line #10):

7:     <% if @gasto.proyecto_origen_id.nil? %>
8:       <% avisos += " " + _("La suma por proyectos no es correcta.") unless @gasto.comprobar_pro
yectos %>
9:     <% end %>
10:         <% if Plugin.find_by_codigo("cpt_contabilidad").activo %>
11:             <% avisos += " " + _("El gasto no tiene cuenta de debito (CPT CONTABILIDAD).") unl
ess @gasto.cpt_cuenta_debito_id %>
12:             <% avisos += " " + _("El gasto no tiene cuenta de credito (CPT CONTABILIDAD).") un
less @gasto.cpt_cuenta_credito_id %>
13:         <% end %>

Trace of template inclusion: app/views/gasto_agentes/listado.html.erb
```

No seria mejor invocar ese codigo dentro de alguna acción concreta del plugin invocada desde el modelo de gasto?.

A no ser que fuera algo bastante genérico, no parece buena solución hardcodear nombres particulares plugins en las vistas.

Por otro lado, se podría hacer una invocación genérica a la espera de que algún plugin la resolviera:

https://gong.org.es/projects/gor/repository/entry/trunk/gor/app/models/gasto.rb#L850

pero quizás es algo ineficiente andar invocando un method_missing con para cada plugin (sin problemas puede haber 3 o 4 activos)

Histórico

#1 - 2017-02-28 16:54 - Santiago Ramos

Comento también de momento el código en el formulario de edición de gasto, puesto que también provoca un error 500: r8155

#2 - 2017-03-01 12:52 - Jaime Ortiz

- Estado cambiado Nueva por Resuelta

- Versión prevista establecido a 2.54

#3 - 2017-03-02 11:28 - Santiago Ramos

- Estado cambiado Resuelta por Asignada

Sigue fallando

#4 - 2017-03-02 12:07 - Jaime Ortiz

Perdon por el follon.

Hago un commit ahora mismo provisional para que no falle.

Respecto a la solución, sin poner la referencia en el plugin, no acabo de entender la primera propuesta.

> No seria mejor invocar ese codigo dentro de alguna acción concreta del plugin invocada desde el modelo de gasto?.

Respecto a la segunda propuesta que haces...

> Por otro lado, se podría hacer una invocación genérica a la espera de que algún plugin la resolviera:
>
> https://gong.org.es/projects/gor/repository/entry/trunk/gor/app/models/gasto.rb#L850

Entiendo que la propuesta es que cree un metodo en el modelo de gasto, y que en ese metodo se busque si existe algun plugin que lo implemente ¿Correcto?

Ese decir, desplazar la referencia a los plugin al modelo, y ademas hacerlo mas "generico", consultando si otros plugins lo implementan. ¿Correcto?

En cuanto sepa por donde van los tiros lo vuelvo a corregir

Gracias y un saludo!

#5 - 2017-03-07 15:52 - Santiago Ramos

Jaime Ortiz escribió:

> Entiendo que la propuesta es que cree un metodo en el modelo de gasto, y que en ese metodo se busque si existe algun plugin que lo implemente ¿Correcto?
>
> Ese decir, desplazar la referencia a los plugin al modelo, y ademas hacerlo mas "generico", consultando si otros plugins lo implementan.

¿Correcto?

Sí, más o menos es lo que quiero decir...

Imagina que en la vista de un gasto de agentes, todo el tema de los avisos queda así:

```
<% avisos = @gasto.chequea_avisos_agente %>
<%= icono "alerta", _("¡Atención!.") + avisos.join(" ") unless avisos.blank? %>
```

y en el modelo de gastos se define un metodo:

```
def chequea_avisos_agente
  avisos = []
  avisos.push _("La suma por proyectos no es correcta.") if proyecto_origen_id.nil? && !comprobar_proyectos
  avisos.push _("El gasto no está pagado correctamente.") unless proyecto_origen_id || pagado?
  [...]
  return avisos
end
```

a ese método le podemos añadir luego los chequeos propios de plugins y quedaría todo más o menos como:

```
def chequea_avisos_agente
  avisos = []
  avisos.push _("La suma por proyectos no es correcta.") if proyecto_origen_id.nil? && !comprobar_proyectos
  avisos.push _("El gasto no está pagado correctamente.") unless proyecto_origen_id || pagado?
  [...]
  Plugin.activos.each do |plugin|
    begin
      avisos_plugin = eval(plugin.clase + "::Gasto").chequea_avisos_agente(self)
      avisos += avisos_plugin if avisos_plugin.class_name == "Array"
    rescue => ex
    end
  end
  return avisos
end
```

Así, cada plugin que quiera generar un aviso para una línea de gasto de agentes, tiene que implementar el método

```
Array chequea_avisos_agente(Gasto)
```

y para las líneas de gasto de proyectos lo mismo...

```
Array chequea_avisos_proyecto(Gasto)
```

De todas formas, es una chapu rápida, porque pudiera resultar un tanto lento el tener que recorrer para cada línea de gasto todos los plugins activos y probar el eval del método de enganche, así que no se si sería mejor buscar alguna forma de hacer crecer los modelos en la carga de estos y no en tiempo de ejecución, de forma que no se busque gasto a gasto si existe el método, sino sólo al levantar la app.

Se te ocurre alguna solución para esto? (pongo a Pascal como seguidor para que le llegue este tema por si tiene alguna idea rápida y simple)

#6 - 2017-05-29 16:41 - Jaime Ortiz

- Estado cambiado Asignada por Resuelta