Django Model Utils

Casi siempre se va a necesitar en los modelos de los atributos fecha de creación y fecha de modificación.
Sin embargo es algo que se debe hacer en todos los modelos y para no estar repitiendo codigo pues nos apoyaremos de la herramienta Django Model Utils.
pip install django-model-utils
al instalar esta app no es necesario añadirla dentro de los entornos ya que es una aplicación que ya la trae en realidad Django sólo que se necesitamos activarlo.

Entonces al instalarlo automaticamente ya se lo ha hecho dentro de todo lo que es Django, por ende para usarlo solo se lo debe importar

get_context_data

Como su nombre indica obtiene los datos del contexto. Son los datos, variables, objectos, etc, que le vas a pasar al template para maquetarlos. Por ejemplo es donde pasarías un formulario en caso de que necesites un template con 2 o más formularios.

A menudo, es necesario presentar información adicional además de la proporcionada por la vista genérica. Por ejemplo, piense en mostrar una lista de todos los libros en cada página de detalles del editor.
La DetailView vista genérica proporciona al editor el contexto, pero ¿cómo obtenemos información adicional en esa plantilla?
La respuesta es subclasificar DetailView y proporcionar su propia implementación del get_context_data método. La implementación predeterminada agrega el objeto que se muestra a la plantilla, pero puede anularlo para enviar más:

```python
from django.views.generic import DetailView
from books.models import Book, Publisher

class PublisherDetailView(DetailView):

    model = Publisher

    def get_context_data(self, **kwargs):
        # Call the base implementation first to get a context
        context = super().get_context_data(**kwargs)
        # Add in a QuerySet of all the books
        context['book_list'] = Book.objects.all()
        return context
```

este es el nombre de mi contexto y con este le llamo en el html

```python
class HomePageView(TemplateView):
    template_name = "home/index.html"

    def get_context_data(self, **kwargs):
        context = super(HomePageView, self).get_context_data(**kwargs)
        # contexto de portada
        context["portada"] = Entry.objects.entrada_en_portada() #Contexto/Variable portada
        return context
```

```html
<div class="cell small-12 large-6">
    <div class="card" style="width: 100%;">
        <img src="{{portada.image.url}}">
        <div class="card-section">
            <h4>{{ portada.title }}</h4>
            <p>{{ portada.resume }}</p>
        </div>
    </div>
</div>
```

Vista para Guardar una Subscripción

Para poder hacer que cargue la url dentro del modal sin hacer uso de otro template pues lo que se hace es de la misma manera usar el POST, token y el boton submit pero además se debe poner un action en el formulario

Dentro del mismo lo que se va a poner es la url para redireccionar

```html
<div class="reveal" id="exampleModal2" data-reveal style="width: 300px;">
    <p class="lead">Ingresa tu correo</p>
    <form class="grid-x grid-margin-x" method="POST" action="{% url 'home_app:add-subscription' %}"> {% csrf_token %}
        <div class="cell small-12">
            <!-- <input type="text" placeholder="E-mail.."> -->
            {{form.email}}
        </div>
        <div class="cell small-12">
            <button type="submit" class="success button">Suscribirme</button>
        </div>
    </form>
</div>
```

Formulario de Contacto

En esta clase lo que se plantea es la funcionalidad del formulario en el footer que se muestra en todas las páginas, inicialmente se pensaría hacer como lo que se hizo con el Modal sin embargo se deberia mandar el get context en cada una de las vistas por lo que se repetiría el código...

Cuando hacemos uso del context y en el HTML mandamos el {{form}} Django propiamente lo que hace es ya hacer la relacion con el modelo

Sin embargo al no poder mandar el contexto del formulario {{form.atributo}} en el HTML debido a la redundancia de codigo pues lo que se hace es

maquetar manualmente los input con el id y el name para que Django los relacione con el modelo en el form

De esta forma se ahorra codigo y se lo ve más elegante