Django-widget-tweaks

<https://simpleisbetterthancomplex.com/2015/12/04/package-of-the-week-django-widget-tweaks.html>

The problem

A basic example of a form using the Bootstrap classes would be:

<form>

<div class="form-group">

<label for="id\_email">Email address</label>

<input type="email" class="form-control" id="id\_email">

</div>

<div class="form-group">

<label for="id\_password">Password</label>

<input type="password" class="form-control" id="id\_password">

</div>

<button type="submit" class="btn btn-default">Submit</button>

</form>

The problem usually lives in the need to add some extra attributes to the HTML element, preserving those old ones that are automatically generated by Django, based on your models.

In this case, it would be the div element with form-group class, as well as the form-control class in the input element.

For example, if we consider the following model:

from django.db import models

class Person(models.Model):

first\_name = models.CharField(max\_length=30)

last\_name = models.CharField(max\_length=30)

email = models.EmailField(max\_length=254)

phone = models.CharField(max\_length=20)

bio = models.TextField(max\_length=500)

And for this model, we create a Django Form:

from django import forms

from simple\_forms.apps.core.models import Person

class PersonForm(forms.ModelForm):

class Meta:

model = Person

fields = ('first\_name', 'last\_name', 'email', 'phone', 'bio',)

If we render this form right away, using the following code:

<form method="post">

{% csrf\_token %}

{{ form }}

<div class="form-group">

<button type="submit" class="btn btn-success">

<span class="glyphicon glyphicon-ok"></span> Save

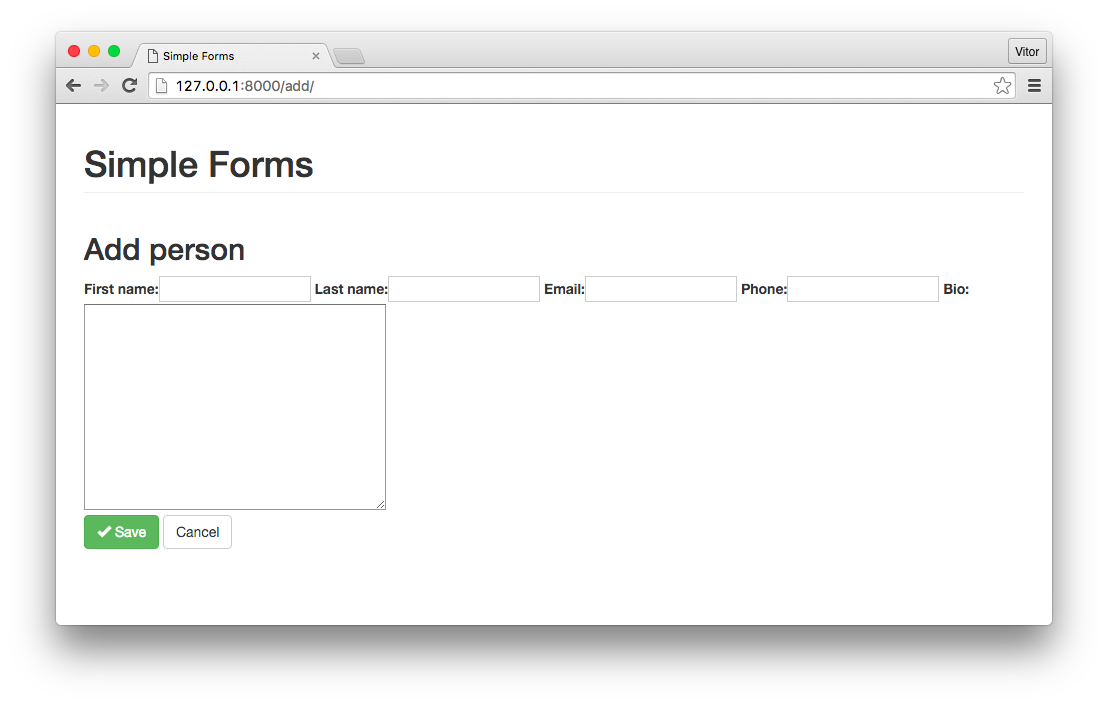
</button>

<a href="{% url 'home' %}"> Cancel</a>

</div>

</form>

It would look broken, like the picture below:



Installation

You can install it with pip, or download it from [PyPI](https://pypi.python.org/pypi/django-widget-tweaks) if you prefer:

$ pip install django-widget-tweaks

Now add widget\_tweaks to your INSTALLED\_APPS:

INSTALLED\_APPS = [

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'widget\_tweaks',

'simple\_forms.apps.core',

]

Usage:

I will show just a few of the many options the package offers. You can learn more reading the official docs.

To start using it, you must load the template tag in the template you want to use its functions:

{% load widget\_tweaks %}

Now expand your form by iterating through its fields, in order to expose the input tags, replacing this:

{{ form }}

With this:

{% for hidden in form.hidden\_fields %}

{{ hidden }}

{% endfor %}

{% for field in form.visible\_fields %}

<div class="form-group">

<label for="{{ field.id\_for\_label }}">{{ field.label }}</label>

{{ field }}

{% for error in field.errors %}

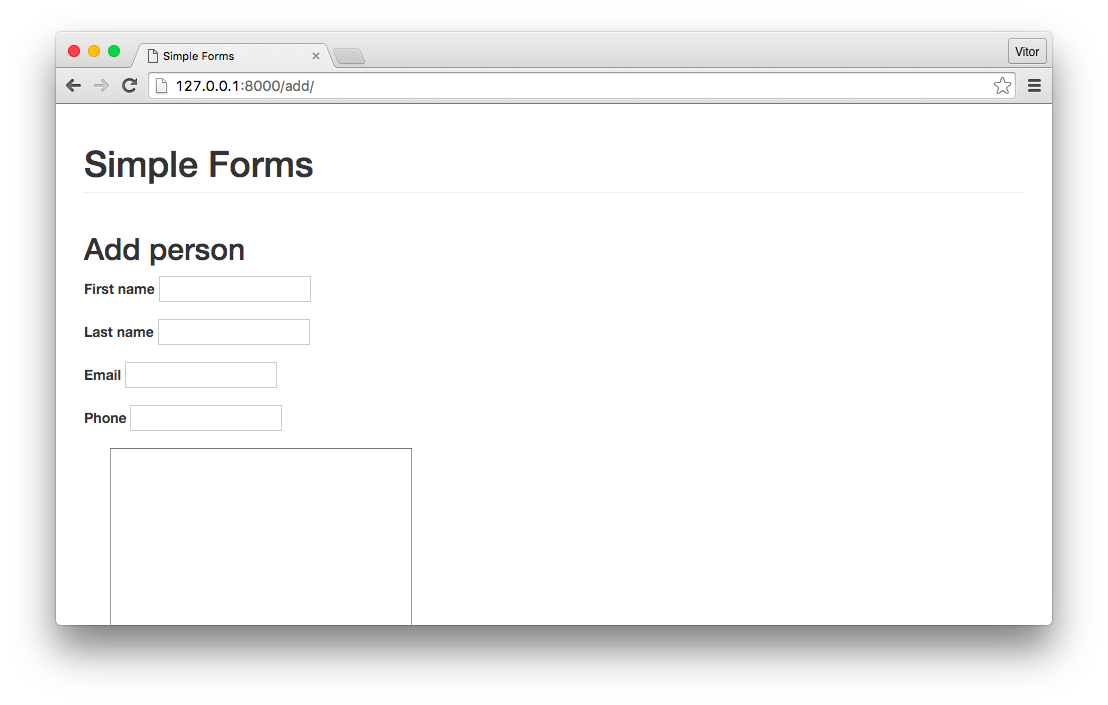
<span class="help-block">{{ error }}</span>

{% endfor %}

</div>

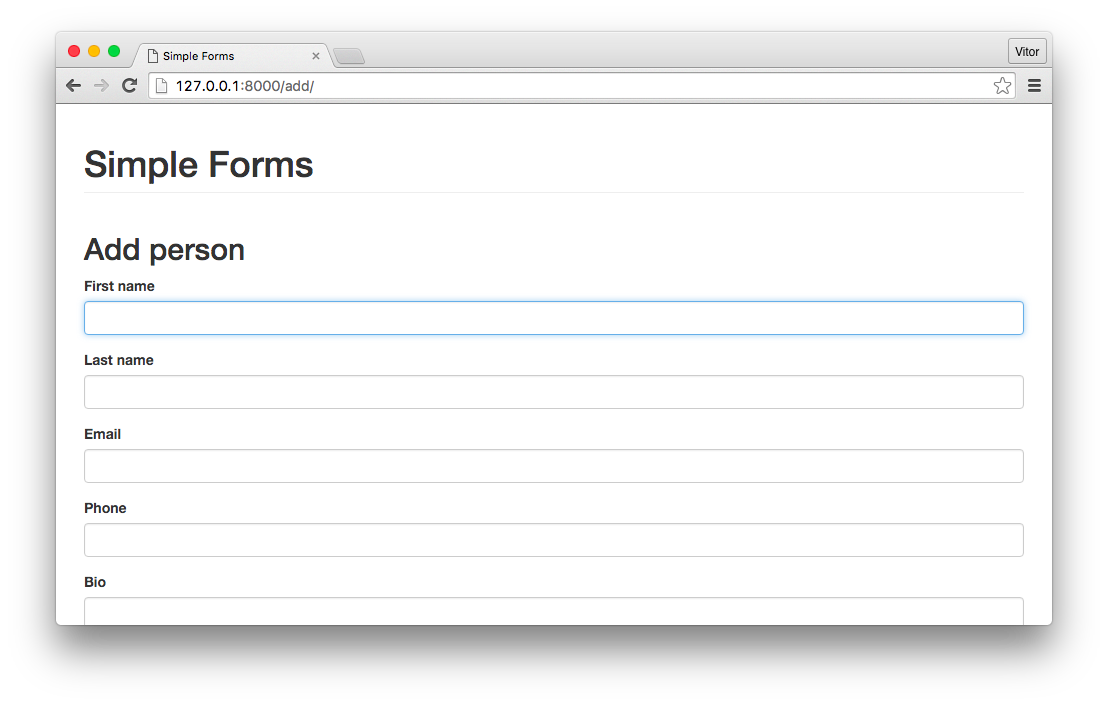
{% endfor %}

At this point we already added several Bootstrap elements, but our form still looks broken:



Now to put Django Widget Tweaks in action, add an extra attribute to the field element:

{{ field|add\_class:'form-control' }}



The final result of our template is shown below:

{% extends 'base.html' %}

{% load widget\_tweaks %}

{% block content %}

<h2>Add person</h2>

<form method="post">

{% csrf\_token %}

{% for hidden in form.hidden\_fields %}

{{ hidden }}

{% endfor %}

{% for field in form.visible\_fields %}

<div class="form-group">

<label for="{{ field.id\_for\_label }}">

{{ field.label }}</label>

{{ field|add\_class:'form-control' }}

{% for error in field.errors %}

<span class="help-block">{{ error }}</span>

{% endfor %}

</div>

{% endfor %}

<div class="form-group">

<button type="submit" class="btn btn-success">

<span class="glyphicon glyphicon-ok"></span> Save

</button>

<a href="{% url 'home' %}" class="btn btn-default">Cancel</a>

</div>

</form>

{% endblock %}

Another way to render the fields is using the render\_field template tag, which gives you a flexible way to render Django fields using a HTML-like syntax:

{% render\_field form.first\_name class="form-control" %}

**My Note**: ***first\_name****,* ***last\_name*** are names of our fields.

You can also use the template variables as attribute values:

{% render\_field form.first\_name class="form-control" placeholder=form.first\_name.label %}

Personally I find this package really useful, because it let you customize your form elements in a non intrusive way, without having to add extra css class inside the form definition. Also it’s more clear this way, because afterall the css classes are related to the page layout.