# Custom Template Tags

To be a valid tag library, the module must contain a module-level variable named **register** that is a **template.Library** instance, in which all the tags and filters are registered. So, near the top of your module, put the following:

# from django import template

# register = template.Library()

The **Library.filter()** method takes two arguments:

1. The name of the filter – a string.
2. The compilation function – a Python function (not the name of the function as a string).

You can use **register.filter()** as a decorator instead:

# @register.filter(name='cut')

# **def** cut(value, arg):

# **return** value.replace(arg, '')

# @register.filter

# **def** lower(value):

# **return** value.lower()

If you leave off the **name** argument, as in the second example above, Django will use the function’s name as the filter name.

Finally, **register.filter()** also accepts three keyword arguments, **is\_safe**, **needs\_autoescape**, and **expects\_localtime**. These arguments are described in [filters and auto-escaping](https://docs.djangoproject.com/en/2.2/howto/custom-template-tags/#filters-auto-escaping) and [filters and time zones](https://docs.djangoproject.com/en/2.2/howto/custom-template-tags/#filters-timezones) below.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Bound and unbound forms

A [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance is either **bound** to a set of data, or **unbound**.

1. If it’s **bound** to a set of data, it’s capable of validating that data and rendering the form as HTML with the data displayed in the HTML.
2. If it’s **unbound**, it cannot do validation (because there’s no data to validate!), but it can still render the blank form as HTML.

To create an unbound [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance, simply instantiate the class:

**>>>** f = ContactForm()

To bind data to a form, pass the data as a dictionary as the first parameter to your [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) class constructor:

# **>>>** data = {'subject': 'hello',

# **...** 'message': 'Hi there',

# **...** 'sender': 'foo@example.com',

# **...** 'cc\_myself': **True**}

# **>>>** f = ContactForm(data)

In this dictionary, the keys are the field names, which correspond to the attributes in your [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) class. The values are the data you’re trying to validate. These will usually be strings, but there’s no requirement that they be strings; the type of data you pass depends on the [**Field**](https://docs.djangoproject.com/en/2.2/ref/forms/fields/#django.forms.Field), as we’ll see in a moment.

**Form.is\_bound**

If you need to distinguish between bound and unbound form instances at runtime, check the value of the form’s [**is\_bound**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form.is_bound) attribute:

# **>>>** f = ContactForm()

# **>>>** f.is\_bound

# False

# **>>>** f = ContactForm({'subject': 'hello'})

# **>>>** f.is\_bound

# True

Note that passing an empty dictionary creates a bound form with empty data:

# **>>>** f = ContactForm({})

# **>>>** f.is\_bound

# True

If you have a bound [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance and want to change the data somehow, or if you want to bind an unbound [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance to some data, create another [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance. There is no way to change data in a [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance. Once a [**Form**](https://docs.djangoproject.com/en/2.2/ref/forms/api/#django.forms.Form) instance has been created, you should consider its data immutable, whether it has data or not.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from django import template

from django.utils.safestring import mark\_safe

register = template.Library()

@register.filter

def highlight\_search(text, search):

highlighted = text.replace(

search, '<span class="highlight">{}</span>'.format(search))

return mark\_safe(highlighted)

<a href="{{ post.get\_absolute\_url }}">{{ post.title|highlight\_search:query }}</a>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

try:

from urllib.parse import quote\_plus #python 3

except:

pass

from django import template

register = template.Library()

@register.filter

def urlify(value):

return quote\_plus(value)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_