
django-currencies Documentation

Release 0.9.0

Panos Laganakos

Dec 14, 2018

Contents

1	Installation	3
2	Set Up	5
3	Views	7
4	Context Processors	9
5	Template Tags and Filters	11
5.1	Tag reference	11
5.2	Filter reference:	11

`django-currencies` allows you to define different currencies, and includes template tags/filters to allow easy conversion between them.

CHAPTER 1

Installation

To install it, run the following command inside this directory:

```
python setup.py install
```

if you have the Python `easy_install` utility available, you can also type the following to download and install in one step:

```
easy_install django-currencies
```

or if you're using `pip`:

```
pip install django-currencies
```

or if you'd prefer you can simply place the included `currencies` directory somewhere on your `PYTHON_PATH`, or symlink to it from somewhere on your Python path; this is useful if you're working from a checkout.

Note that this application requires Python 2.3 or later, and a functional installation of Django 1.0 or newer. You can obtain Python from <http://www.python.org/> and Django from <http://www.djangoproject.com/>.

To use the currencies system with all its default settings, you'll need to do the following:

1. Add `currencies` to the `INSTALLED_APPS` setting of your Django project.
2. Add `currencies.context_processors.currencies` in your `TEMPLATE_CONTEXT_PROCESSORS` setting of your Django project.
3. Add this line to your site's root `URLConf`:

```
urlpatterns += patterns('',
    url(r'^currencies/', include('currencies.urls')),
)
```

4. Run the command `./manage.py syncdb`.

The `syncdb` command creates the necessary database tables and creates permission objects for all installed apps that need them.

That's it!

CHAPTER 3

Views

django-currencies defines a `set_currency` view, in which you need to pass the new currency you want as a `currency_code` variable (as POST), and it will be set. `urls.py`, includes the named url, so you can do:

```
{% url currencies_set_currency [currency_code] %}
```

A form that could handle the currency switching could be defined like so:

```
<form id="currency_switcher" method="POST" action="{% url 'currencies_set_currency' %}"
→">{% csrf_token %}
  <select name="currency" onchange="$('#currency_switcher').submit()">
    {% for curr in CURRENCIES %}
      <option value="{{ curr.code }}" {% ifequal curr.code CURRENCY.code %}selected=
→"selected"{% endifequal %}>
        {{ curr.symbol }} {{ curr.name }}
      </option>
    {% endfor %}
  </select>
  <noscript>
    <input type="submit" value="Set" />
  </noscript>
</form>
```


CHAPTER 4

Context Processors

`django-currencies` provides a `currencies.context_processors.currencies`, which gives you the following template variables:

```
# A list of the active currencies.  
CURRENCIES  
  
# The currently set currency.  
CURRENCY
```

Template Tags and Filters

The `currencies.templatetags.currency` module defines a template tag and filter which may be used to work with currencies.

5.1 Tag reference

5.1.1 `change_currency`

Retrieves a list of Tag objects associated with a given model and stores them in a context variable.

Usage:

```
{% change_currency [price] [currency_code] %}
```

i.e:

```
{% change_currency product.price "USD" %}  
  
<!-- or if you have the ``currencies.context_processors.currencies`` available -->  
{% change_currency product.price CURRENCY.code %}
```

5.2 Filter reference:

5.2.1 `currency`

Usage:

```
{{ [price] | currency:[currency_code] }}
```

i.e.:

```
{{ product.price|currency:"USD" }}
```