

---

# **django-currencies Documentation**

*Release 0.2.1*

**Panos Laganakos**

**Apr 24, 2017**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Set Up</b>	<b>5</b>
<b>3</b>	<b>Views</b>	<b>7</b>
<b>4</b>	<b>Context Processors</b>	<b>9</b>
<b>5</b>	<b>Template Tags and Filters</b>	<b>11</b>
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:

```
(r'^currencies/', include('currencies.urls')),
```

4. Run the command `python 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!



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

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

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

```
<form id="currency_switcher" method="POST" action="{% url currencies_set_currency %}">
  <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>
```



---

## Context Processors

---

Django `currencies`, provides a `currencies.context_processors.currency`, which gives you the following template variables:

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



---

## Template Tags and Filters

---

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

### Tag reference

#### `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 we have the currencies.context_processors.currencies  
# available:  
  
{% change_currency product.price CURRENCY.code %}
```

### Filter reference:

#### `currency`

Usage:

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

i.e.:

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