jinja.pocoo.org

Introduction — Jinja2 Documentation (2.10)

4 minutes

This is the documentation for the Jinja2 general purpose templating language. Jinja2 is a library for Python that is designed to be flexible, fast and secure.

If you have any exposure to other text-based template languages, such as Smarty or Django, you should feel right at home with Jinja2. It's both designer and developer friendly by sticking to Python's principles and adding functionality useful for templating environments.

Prerequisites¶

Jinja2 works with Python 2.6.x, 2.7.x and >= 3.3. If you are using Python 3.2 you can use an older release of Jinja2 (2.6) as support for Python 3.2 was dropped in Jinja2 version 2.7.

If you wish to use the `PackageLoader` class, you will also need setuptools or distribute installed at runtime.

Installation¶

You have multiple ways to install Jinja2. If you are unsure

what to do, go with the Python egg or tarball.

As a Python egg (via *easy_install*)¶

You can install the most recent Jinja2 version using easy_install or pip:

```
easy_install Jinja2
pip install Jinja2
```

This will install a Jinja2 egg in your Python installation's site-packages directory.

From the tarball release¶

1. Download the most recent tarball from the download page
2. Unpack the tarball
3. `python setup.py install`

   Note that you either have to have *setuptools* or *distribute* installed; the latter is preferred.

   This will install Jinja2 into your Python installation's site-packages directory.

Installing the development version¶

1. Install git
2. `git clone git://github.com/pallets/jinja.git`
3. `cd jinja2`
4. `ln -s jinja2 /usr/lib/python2.X/site-packages`

As an alternative to steps 4 you can also do `python setup.py develop` which will install the package via *distribute* in development mode. This also has the advantage that the C extensions are compiled.

MarkupSafe Dependency¶

As of version 2.7 Jinja2 depends on the MarkupSafe module. If you install Jinja2 via *pip* or *easy_install* it will be installed automatically for you.

Basic API Usage¶

This section gives you a brief introduction to the Python API for Jinja2 templates.

The most basic way to create a template and render it is through `Template`. This however is not the recommended way to work with it if your templates are not loaded from strings but the file system or another data source:

```
>>> from jinja2 import Template
>>> template = Template('Hello {{ name }}!')
>>> template.render(name='John Doe')
u'Hello John Doe!'
```

By creating an instance of `Template` you get back a new template object that provides a method called `render()` which when called with a dict or keyword arguments expands the template. The dict or keywords arguments passed to the template are the so-called “context” of the template.

What you can see here is that Jinja2 is using unicode internally and the return value is an unicode string. So make sure that your application is indeed using unicode internally.

Experimental Python 3 Support¶

Jinja 2.7 brings experimental support for Python >=3.3. It means that all unittests pass on the new version, but there might still be small bugs in there and behavior might be inconsistent. If you notice any bugs, please provide feedback in the Jinja bug tracker.

Also please keep in mind that the documentation is written with Python 2 in mind, so you will have to adapt the shown code examples to Python 3 syntax for yourself.