# Contents

1 Requirements 3
2 Installation 5
3 Setup 7
4 Usage 9
5 Developer Setup 11
   5.1 Contents .................................................. 11
6 Indices and tables 13
A small collection of template tags for debugging and introspecting Django templates

Documentation
CHAPTER 1

Requirements

None, but the latest ipdb is highly recommended.
django-template-debug is available on pypi, so the easiest way to install it is using pip:

```
 pip install django-template-debug
```
Add ‘template_debug’ to the INSTALLED_APPS iterable in your settings file. For example:

```python
INSTALLED_APPS = (  
    ...  
    'template_debug',  
    ...  
)
```

Add `TEMPLATE_DEBUG = True` to your local or development settings if it is not already set.

- Unless `TEMPLATE_DEBUG` is set to True, the django-template-debug templates will return an empty string without doing anything. This behavior prevents your application from calling `set_trace()` or `print` in a production environment if django-template-debug template tags are accidentally committed and deployed.
Add {% load debug_tags %} in any Django template.
The available tags to use are {% set_trace %} {% variables %} {% attributes varname %} and {% details varname %}
See Example Usage docs for more details
CHAPTER 5

Developer Setup

Create a fresh virtualenv and install the test requirements:

```
mkvirtualenv template-debug
pip install -r requirements/test.txt
```

Use manage.py in the project directory along with the example.settings file for local testing.

To run unittests using the virtualenv’s Python and Django, use the `runtests` script. To test all supported versions of Python and Django, run the unittests using tox.

5.1 Contents
Indices and tables

• search