Thumbor Community Core

Oct 08, 2018

Contents

CHAPTER 1

Contents

1.1 Installing Thumbor Community Core

1. Install thumbor (see Thumbor repository)
2. Clone this repository
3. If you've set a virtualenv up for thumbor, activate it.
4. Install the Thumbor Community Core:

```
$ cd thumbor-community/core
$ pip install .
```

5. Install the extensions you wish to load.
6. Register the extensions to load within Thumbor's configuration file:

```
COMMUNITY_EXTENSIONS = [
    'my_extension',
    ...
]
```

7. Launch thumbor with the Thumbor Community custom application:

```
$ thumbor --conf=my_configuration_file -a tc_core.app.App
```

1.2 Available Thumbor Community Extensions

- URL Shortener

1.3 How to create your own extension?

The best way to understand how to do it is by example, you can check out Shortener (particularly the `__init__.py` file).

1.3.1 What's an extension?

An extension is a python module that will be integrated within thumbor, through Thumbor Community's core package. This module may contain autoloaded thumbor's modules (such as Filters, Storages, ...), tornado handlers, or configuration values.

1.3.2 Define your extension

You extension must be registered to Thumbor Community's core. To do so, you'll need create an extension instance as follows:

```
from tc_core import Extension, Extensions

// Extension name (here `my_extension` must match your python module name)
extension = Extension('my_extension')

// ... configure your extension

Extensions.register(extension)
```

So within this example we have a project containing a python module named `my_extension` which will be registered in the core. To enable it, we will have to edit our Thumbor's configuration to add the newly registered extension to Thumbor:

```
COMMUNITY_EXTENSIONS = [
    'my_extension'
]
```

1.3.3 Setup the extension

You've registered your extension and it is now loaded, but what if you'd like to use some of Thumbor's mechanisms to interconnect a bit further?

Add config

You can register new configuration options fairly easily as follows:

```
from derpconf.config import Config

Config.define('MY_CONFIG_KEY', 'my_default_value',  'Description for my key.',
↪'Configuration section')
```

And voilà! It's done, you can now fill in this value in your config and retrieve it from the config object by calling it:

```
my_val = self.context.config.get('MY_CONFIG_KEY')
```

But this is given you have access to the context object... To do that, check out how to add modules in the section below.

Add modules

Modules are context-aware classes which will be instantiated upon receiving the request.

To register them:

```
from tc_core import Extension, Extensions

extension = Extension('my_extension')

extension.add_module(
    config_key='MY_MODULE',
    class_name='MyModuleClass'
)

Extensions.register(extension)
```

In the configuration:

```
MY_MODULE = 'my_extension.my_module'
```

Given you have the following class:

```
# File my_extension/my_module.py

class MyModuleClass(object):

    def __init__(self, context):
        self.context = context
```

This will instantiate the class named `MyModuleClass` within the python module `my_extension.my_module` by passing the context to the constructor if need be.

Add handlers

You may also add tornado handlers that will allow you to create custom controllers to answer the request to thumbor.

```
from tc_core import Extension, Extensions
from my_extension.handlers import MyHandler

extension = Extension('my_extension')

extension.add_handler(MyHandler.regex(), MyHandler)

Extensions.register(extension)
```

Given you have the following handler:

```
# File my_extension/handlers/__init__.py

from thumbor.handlers import ContextHandler

class MyHandler(ContextHandler):
```

(continues on next page)

(continued from previous page)

```
    @classmethod
    def regex(cls):
        return r'/my_handler/(?P<url>.+)'

    // ...
```

1.4 Contributing

If you wish to contribute to Thumbor-Community, please abide by the Code of conduct.