
dataflake.wsgi.bjoern

Jens Vagelpohl

Feb 04, 2023

CONTENTS

1	Narrative documentation	3
1.1	Installation	3
1.2	Using this package	3
1.3	Development	4
1.4	Changelog	5
2	Support	7
3	Indices and tables	9

This package provides a PasteDeploy-compatible entry point to easily integrate the [bjoern WSGI server](#) into an environment that uses PasteDeploy-style `.ini` files to compose a WSGI application.

It also includes a script to create a basic WSGI configuration file for Zope, similar to Zope's own `mkwsgiinstance`, but specifying `bjoern` instead of `waitress` as WSGI server.

Please note that `bjoern` requires the `libev` event library to be installed.

NARRATIVE DOCUMENTATION

Narrative documentation explaining how to use `dataflake.wsgi.bjoern`.

1.1 Installation

1.1.1 Install with pip

```
$ pip install dataflake.wsgi.bjoern
```

1.1.2 Install with `zc.buildout`

Just add `dataflake.wsgi.bjoern` to the `eggs` setting(s) in your buildout configuration to have it pulled in automatically:

```
...  
eggs =  
    dataflake.wsgi.bjoern  
...
```

1.2 Using this package

Warning: Bjoern does not use Python logging facilities when encountering errors. It just writes to `STDERR`. That means you will see exception output on the console if you run the application in the foreground, but not in the configured event log.

1.2.1 Using the PasteDeploy entry point

You can use the PasteDeploy entry point in your WSGI configuration file to define a bjoern server:

```
[server:main]  
use = egg:dataflake.wsgi.bjoern#main  
host = 127.0.0.1  
port = 8080  
reuse_port = True
```

If you leave out the host specification, `bjoern` will listen on all IPv4 interfaces (`0.0.0.0`). If no port is specified `bjoern` will choose a random port (probably not what you want). `reuse_port` sets `SO_REUSEPORT` if it is available on your platform.

Alternatively you could use `listen` directive instead of `host` and `port`:

```
[server:main]
use = egg:dataflake.wsgi.bjoern#main
listen = 127.0.0.1:8080
reuse_port = True
```

```
[server:main]
use = egg:dataflake.wsgi.bjoern#main
listen = 8080
reuse_port = True
```

1.2.2 Creating a basic WSGI configuration for Zope

This package defines a console script named `mkbjoerninstance` that works just like Zope's own `mkwsgiinstance`. It will ask you for a location, a username and a password to create a basic Zope instance home with a WSGI configuration, in this case it will be `bjoern`-based as opposed to Zope's default, `waitress`.

Note: Just like `mkwsgiinstance`, the script will not overwrite an existing WSGI configuration file at `etc/zope.ini`. You need to move the existing file to the side to get a fresh configuration.

```
$ bin/mkbjoerninstance
Please choose a directory in which you'd like to install
Zope "instance home" files such as database files, configuration
files, etc.

Directory: .
Please choose a username and password for the initial user.
These will be the credentials you use to initially manage
your new Zope instance.

Username: admin
Password: (enter password)
Verify password: (re-enter password)
```

1.3 Development

1.3.1 Bug tracker

For bug reports, suggestions or questions please use the GitHub issue tracker at <https://github.com/dataflake/dataflake.wsgi.bjoern/issues>.

1.3.2 Getting the source code

The source code is maintained on GitHub. To check out the main branch:

```
$ git clone https://github.com/dataflake/dataflake.wsgi.bjoern.git
```

You can also browse the code online at <https://github.com/dataflake/dataflake.wsgi.bjoern>

1.3.3 Preparing the development sandbox

The following steps only need to be done once to install all the tools and scripts needed for building, packaging and testing. First, create a Virtual environment. The example here uses Python 3.11, but any Python version supported by this package will work. Then install all the required tools:

```
$ cd dataflake.wsgi.bjoern
$ python3.11 -m venv .
$ bin/pip install -U pip wheel
$ bin/pip install -U setuptools zc.buildout tox twine
```

1.3.4 Running the tests

You can use `tox` to run the unit and integration tests in this package. The shipped `tox` configuration can run the tests for all supported platforms. You can read the entire long list of possible options on the [tox CLI interface documentation page](#), but the following examples will get you started:

```
$ bin/tox -l      # List all available environments
$ bin/tox -pall  # Run tests for all environments in parallel
$ bin/tox -epy311 # Run tests on Python 3.11 only
$ bin/tox -elint # Run package sanity checks and lint the code
```

1.3.5 Building the documentation

`tox` is also used to build the Sphinx-based documentation. The input files are in the `docs` subfolder and the documentation build step will compile them to HTML. The output is stored in `docs/_build/html/`:

```
$ bin/tox -edocs
```

If the documentation contains doctests they are run as well.

1.4 Changelog

1.4.1 2.1 (unreleased)

1.4.2 2.0 (2023-02-02)

- Drop support for Python 2.7, 3.5, 3.6.

1.4.3 1.2 (2022-06-27)

- Added support for Python 3.9 and 3.10
- Switched to GitHub Actions for CI (#2)

1.4.4 1.1 (2019-08-14)

- Added the `listen` parameter to pass a port or a host and port. This is more in line with what `plone.recipe.zope2instance` expects.

1.4.5 1.0 (2019-04-20)

- Initial release

SUPPORT

If you need commercial support for this software package, please visit <https://www.zetwork.com>.

INDICES AND TABLES

- genindex
- glossary