Firenado Framework Documentation

Release 0.2.15

Flavio Garcia

Jan 30, 2022

Contents

Firenado is a Python web framework that encapsulates and extends Tornado organizing the application in components also adding a server side session layer, yaml based configuration files as other features common that will help developers building web applications and services.

Contents:

CHAPTER 1

User's guide

1.1 Introduction

Firenado is a Python web framework that primarily extends the Tornado framework and runs over it's web server.

A Firenado application is organized in components wired by yaml config files. This design makes it possible to develop shareable components between applications and be distributed separately.

Firenado also provides a server side session layer based on files or redis. If you prefer it is also possible to create a new backend to the session layer or disable it.

Other features shipped with the framework are configurable data sources, a service layer that could be injected to handles via decorators, a command line that helps start a project and run an application.

Firenado provides many other features and resources that will help a developer to create and manage Tornado applications.

1.1.1 Instalation

```
pip install firenado
```

1.1.2 Usage

Creating and running a new application:

```
firenado project init helloworld
cd helloworld
firenado app run
```

By default an application will be created with a redis based session and a redis data source defied and linked to the session.

Firenado don't install redispy so it is necessary to either install it or turn the session as file based. You can disable the session engine too.

To change the session type to file go to helloworld/conf/firenado.yml and change the session definition to:

```
# Session types could be:
# file or redis.
session:
 type: file
 enabled: true
 # Redis session handler configuration
 # data:
 #   source: session
 # File session handler related configuration
 path: /tmp
```

If your helloworld project isn't on the python path just go helloworld/conf/firenado.yml and configure the application settings:

```
app:
  component: helloworld
  data:
  sources:
     # Set here references from sources defined on data.sources
     - session
  pythonpath: ..
  port: 8888
```


1.2 Command line

Firenado provides a command line structure that helps you to execute or create management tasks. Those tasks are organized by category, commands and sub commands.

The framework provides a category called Firenado with the following commands:

- app Application related commands
- projec(ect) Project related commands
- rand(om) Random related commands

Firenado has a command line program that helps to create a new project, run an application and other things.

Setting up a redis based session:

1.3 Components

Firenado is organized in components that could be distributed as a full application or loose-coupled parts made to be shared between applications.

A component could provide new handlers, templates, services, models as other resources to an application in an organized manner.

An application is a component and it is defined into the app.component parameter in the app application configuration file:

```
app:
  component: skell
```

Components are mapped into the framework, system or application configuration files. They are defined by the component section:

```
components:
  - id: skell
    class: skell.app.SkellComponent
    enabled: true
  - id: internal
    class: skell.components.internal.component.SkellInternalComponent
    enabled: true
```

A component configuration item is contains the follow parameters:

- id

The component id to be registered in the application during the load time.

The id is used to get the component from the application during run time and to reference a template from another component.

- class

The component class to be loaded during the load time.

- enabled

Defines if the component is enabled or disabled. If isn't informed the component will be disabled.

Disabled components are not available at run time.

1.3.1 Before and after handler methods

For all handler returned by the component it is possible to implement the methods before_handler and after_handler that will be executed before and after the some http method be triggered by the Tornado server.

If a component TestComponent has a handler TestHandler that provides a get method, mapped at /test url, when a request to this method is executed by the server, the follow methods will be executed in this order:

1. TestComponent.before_handler
2. TestHandler.get
3. TestComponent.after_handler

Both before_handler and after_handler have the current handler as a parameter and the handler session will be initialized correctly during their execution.

The execution of those methods will happen all the time a handler method returned by the component is requested. In another words before_handler and after_handler are global to the handlers returned by the component.

```
class TestComponent(firenado.tornadoweb.TornadoComponent):

    def get_handlers(self):
        return [
            (r"/test", handlers.TestHandler),
        ]
```

(continues on next page)

(continued from previous page)

```
    def after_handler(self, handler):
        print("Doing something after handler: %s" % handler)

    def before_handler(self, handler):
        print("Doing something before handler: %s" % handler)
```

1.4 Schedule

If you need to crate an application with scheduled actions Firenado provides a structure to do it.

Using a scheduled component is possible to define a scheduler via configuration and point to tasks that could be run once or in a interval.

1.4.1 Setting up a scheduled component:

Instead of creating extending TornadoComponent extend it from ScheduledTornadoComponent.

```
from . import handlers
from firenado import schedule


class SchedappComponent(schedule.ScheduledTornadoComponent):
```

We added the component to conf/firenado.yml and enabled it.

```
components:
  - id: schedapp
    class: schedapp.app.SchedappComponent
    enabled: true
```

This component could be used as the application component if you need.

```
app:
  component: 'schedapp'
```

By default Firenado will look for <component_id>_schedule.[ymllyaml] config file. Here is an example with a config based scheduler running a cron based scheduled task(the file is conf/schedapp_schedule.yml):

```
schedulers:
  - id: scheduler1
    name: Scheduler Example 1
    jobs:
      - id: job1
        class: schedapp.jobs.PrintTestJob
        cron: "*/1 * * * *"
        custom_property: Custom property from job1.
      - id: job2
        class: schedapp.jobs.PrintTestJob
        interval: 15000
        custom_property: Custom property from job1.
```

The configuration file is define the scheduler scheduler1 that manages a job identified as job1 that runs every minute and a custom property to be used by the the class schedapp.jobs.PrintTestJob. Here is the the job implementation:

A scheduled job can be executed by interval and cron string.

The interval is defined in milliseconds and will take priority over the cron string.

The cron string will defined intervals using the cron like format and is resolved by croniter.

```
from firenado.schedule import ScheduledJob


class PrintTestJob(ScheduledJob):

    def __init__(self, scheduler, **kwargs):
        super(PrintTestJob, self).__init__(scheduler, **kwargs)
        self._custom_property = kwargs.get('custom_property', None)

    def run(self):
         print("This is the custom property value: %s" % self._custom_property)
```

As demonstrated above we need to create a class that extends from ScheduledJob and implements the run method. We used the constructor to consume the custom property defined in the schedule config file and used it to print a message.

1.5 Services

The way firenado organizes the logic to be executed in several parts of an application is defining services.

Those services can be injected with the decorator `firenado.service.served_by`. This decorator will add an instance of a service to a method of any data connected object. Examples of data connected classes are `firenado.tornadoweb.TornadoHandler` and any descendent of `firenado.service.FirenadoService`.

Creating a service and decorating a handler:

```
from firenado import service, tornadoweb
# Importing a package with some services
import another_service_package


class MyService(service.FirenadoService):
    def do_something(self):
        # Self consumer will be the handler where this service was
        # called from.
        self.consumer.write("Something was done")


class MyHandlerBeingServed(tornadoweb.TornadoHandler):
    # A good way to keep the reference is keeping the type hint
    my_service: MyService
    service_from_another_package: another_service_package.AnotherService

    @service.served_by(MyService)
    # you can also set the attribute/property name to be used
    @service.served_by(another_service_package.AnotherService,
        attribute_name="service_from_another_package"
    )
    def get(self):
        # The anotation service.served_by added self.my_service
        # here. The attribute/property name will be converted from the
        # cammel cased class to dashed separated.
```

(continues on next page)

(continued from previous page)

```
        self.my_service.do_something()
        self.service_from_another_package.do_another_thing()
```

You can also add services to another services using the decorator:

1.6 Session

Firenado provides a session layer supporting redis or files backends.

To setup the application session it is necessary configure the session section into the firenado.yml file.

Setting up a redis based session:

```
session:
 type: redis
 enabled: true
 data:
   source: session
```

Setting up a file based session:

```
session:
 type: file
 enabled: true
 path: /tmp
```

Once a session is set and enabled the developer can persist or retrieve session data in the handler.

```
class SessionCounterHandler(firenado.tornadoweb.TornadoHandler):

    def get(self):
        reset = self.get_argument("reset", False, True)
        if reset:
            # Deleting a session variable
            self.session.delete('counter')
            self.redirect(self.get_rooted_path("/session/counter"))
            return None
        counter = 0
        # Checking if a variable is set in the session
        if self.session.has('counter'):
            # Retrieving a value from the session
            counter = self.session.get('counter')
        counter += 1
        # Setting a value in the session
        self.session.set('counter', counter)
        self.render("session.html", session_value=counter)
```

CHAPTER 2

Configuration

Firenado extends the Tornado web framework which uses ini files to help developers to configure theirs applications.

Besides the fact the ini configuration from Tornado is pretty easy and straight forward to use, it just define key/value structures.

Let's assume that someone wants to define hierarchical configuration structures . It is necessary create indexes that represent the hierarchy(i.e. my.hierarchical.index) and the file will be organized in a key/value manner. In this case the developer assumes the index is structured hierarchically or some extra development is needed to represent this data as should be in memory.

A key/value structure isn't a problem if the application configuration is simple. When a project requires lots of configurable parameters ini files can be overwhelming.

Firenado uses yaml files instead that are organized in a hierarchical structure and can define lists and dictionaries values instead of only strings. With yaml boolean and numeric values are resolved with the same time when we consume them in the python side.

A Firenado application is defined by a configuration file. This file will define application aspects like session, data sources or the port to listen. The application file will overload configuration set by framework and system levels.

The framework config file will define the framework parts like:

- Firenado components offered by the framework
- Data connectors, used to connect to databases
- Log configuration
- Firenado cli management commands
- Session handlers and encoders

On the system config file level it is possible to define system level:

- Firenado components
- Data sources
- Custom data connectors

- Log configuration
- Custom cli managment commands
- Custom Session handlers and encoders

The app config file will define the application component and tornado behaviour(ie. port or socket, number of processes forked, etc). It is also possible to configure on this level:

- Data sources used by the application
- If session is enabled and what session handler and encoder is being used by

the application - Override framework and system configuration

2.1 Application

The application configuration section set properties to the application to be launched by Firenado.

The configuration items will set aspects like application data sources or addresses and ports to listen for requests.

Here is an example of an app section:

```
app:
 component: "myapp"
 data:
   sources:
     - session
     - mydata
 port: 9091
```

2.1.1 Configuration Items

2.1.2 addresses

List of addresses the application will be listen for requests.

- Type: list
- Default value: [“::”, “0.0.0.0”]

2.1.3 component

Firenado component to be set as the application component. This is the main application’s component.

When running *firenado proj init <project_name>* command a component is created in the app.py file and set as in the conf/firenado.yml file.

- Type: string
- Default value: None

```
app:
 component: "myapp"
```

2.1.4 data

Dictionary containing application data related properties.

2.1.5 sources

List of data sources to be created into the application during the launch process. The list items are names of data sources defined in the configuration data section.

Data sources can be defined in system and application levels.

- Type: list
- Default value: []

```
app:
 data:
   sources:
     - mydata
     - session
```

2.1.6 pythonpath

Paths to be added to PYTHONPATH environment variable during the application launch process.

- Type: string
- Default value: None

```
app:
 pythonpath: "..:/a/path/somewhere:"
```

2.1.7 port

Port the application will be listen for requests.

- Type: int
- Default value: 8888

```
app:
 port: 9092
```

2.1.8 process

Configuration to fork the Tornado process to be launched by Firenado.

If the num_processes value is set to None the ioloop will be started without forking.

If num_processes is set to a value bigger than 0 the ioloop will be forked with this amount as number of child processes. If num_processes is set to zero the number of cpu will be used to fork the main process.

The max_restarts value will only be used if num_processes is not none.

- Type: dictionary

- Default value: {'num_processes': None, 'max_restarts': 100}

```
app:
 process:
   num_processes: 4
   max_restarts: 150
```

- See:
- https://www.tornadoweb.org/en/stable/process.html#tornado.process.fork_processes
- https://www.tornadoweb.org/en/stable/httpserver.html#tornado.httpserver.HTTPServer

2.1.9 settings

Settings to be passed to the Tornado application to be launched by Firenado.

- Type: dictionary
- Default value: {}

```
app:
 settings:
   cookie_secret: "kljasdf;lkasjdf;lasdkfjasd;lfkjasdf;lkasdjfasd"
   debug: true
   xsrf_cookies: true
```

- See:
- http://www.tornadoweb.org/en/stable/web.html#tornado.web.Application.settings

2.1.10 socket

Unix socket path the application will be listen. When socket is defined either addresses and port are ignored.

- Type: string
- Default value: None

```
app:
 pythonpath: "/tmp/myapp_socket"
```

2.1.11 wait_before_shutdown

Time in seconds to wait before trigger the application shutdown.

- Type: int
- Default value: 0

```
app:
 wait_before_shutdown: 5
```

CHAPTER 3

Release notes

3.1 What's new in Firenado 0.2.15

3.1.1 Jan 30, 2022

We are pleased to announce the release of Firenado 0.2.15.

At this release we added a service decorator to help handle sqlalchemy sessions.

Here are the highlights:

Bug Fixes

- Change dashed parameters to underscored ones in setup.cfg. #385

Features

- Create a decorator to handle shared sqlalchemy sessions between mehtods. #377
- Add shortcut properties to help get components from the scheduler and scheduled job. #387

3.2 What's new in Firenado 0.2.14

3.2.1 Dec 09, 2021

We are pleased to announce the release of Firenado 0.2.14.

Fixed pagination api as implemented in the new Cartola 0.14 and replaced travis by github actions for testing.

Here are the highlights:

Refactory

- Update cartola paginator. #380
- Replace travis by github actions. #381

3.3 What's new in Firenado 0.2.13

3.3.1 Oct 31, 2021

We are pleased to announce the release of Firenado 0.2.13.

This release is adding a new way to create sqlalchemy data sources and fixing the xheaders warning when no xheaders parameter is defined in the application configuration.

Here are the highlights:

Bug Fixes

- If xheaders isn't defined in an application firenado will always warn about it. #374

New Features

- Define a sqlalchemy using parameters instead url. #70

3.4 What's new in Firenado 0.2.12

3.4.1 Oct 23, 2021

We are pleased to announce the release of Firenado 0.2.12.

Here are the highlights:

Features

- Add xheaders configuration item to the application section. #370
- Create a sqlalchemy fast count method. #371

3.5 What's new in Firenado 0.2.11

3.5.1 Oct 17, 2021

We are pleased to announce the release of Firenado 0.2.11.

The fix presented at 0.2.10 won't cover methods not allowed, bug #366 was reopened. This version fixed the issue.

Here are the highlights:

Bug Fixes

- When writing an error the component handler will enter in infinite loop. #366

3.6 What's new in Firenado 0.2.10

3.6.1 Oct 16, 2021

We are pleased to announce the release of Firenado 0.2.10.

This release is fixing an infinite loop when the component handler was handling a http error.

Here are the highlights:

Bug Fixes

- When writing an error the component handler will enter in infinite loop. #366

3.7 What's new in Firenado 0.2.9

3.7.1 Apr 03, 2021

We are pleased to announce the release of Firenado 0.2.9.

Logging basic configuration will be executed in the launcher instead of the import of *firenado.conf*. This will make it easier to evolve logging functionalities and fix some undesired behaviors caused by all logging logic being handled in *firenado.conf*.

Here are the highlights:

Refactory

- Move log handling to the launcher. #361
- Use log_level_from_string from carcara. #362

3.8 What's new in Firenado 0.2.8

3.8.1 Mar 27, 2021

We are pleased to announce the release of Firenado 0.2.8.

Removed six from the project directly. Some other packages(like croniter) may depend on six but our logic is tied to python 3 at this point. It was also removed *import from __future__* as python 2.7 support was removed.

The dynamic configuration is being handled by cartola.config and logging from scheduler level was changed to debug instead of info.

As sqlalchemy was changed to 1.4.3 we added the future parameter to the data source configuration to make it possible port application to the 2.0 style.

Here are the highlights:

New Features

- Add future parameter to the sqlalchemy data source data feature. #354

Refactory

- Remove six from the project refactory. #352
- Handle dinamic configuration from yaml files using cartola.config. #355
- Change logging from in the schedule to debug instead of info. #356

3.9 What's new in Firenado 0.2.7

3.9.1 Mar 14, 2021

We are pleased to announce the release of Firenado 0.2.7.

Upgraded to Tornado 6.1. Python 2.7 and 3.5 are no longer supported.

Significant refactory to unify TornadoHandler and TornadoWebSocketHandler logic.

Here are the highlights:

Bug Fixes

- Launcher won't launch on Windows with python 3.9. #339

New Features

- Upgrade tornado to 6.1. #344
- Remove Python 2.7 and 3.5 from the support list. #343

Refactory

- Unify TornadoHandler and TornadoWebSocketHandler logic. #342

3.10 What's new in Firenado 0.2.6

3.10.1 Mar 14, 2021

We are pleased to announce the release of Firenado 0.2.6.

Here are the highlights:

Bug Fixes

- Pyyaml conflict with cartola when installing 0.2.5. #338

3.11 What's new in Firenado 0.2.5

3.11.1 Mar 02, 2021

We are pleased to announce the release of Firenado 0.2.5.

Here are the highlights:

Bug Fixes

- TornadoHandler write error will never hit the default RequestHandler write error. #335

3.12 What's new in Firenado 0.2.4

3.12.1 Feb 28, 2021

We are pleased to announce the release of Firenado 0.2.4.

Here are the highlights:

New Features

- Add http error handling to the framework. #242

3.13 What's new in Firenado 0.2.3

3.13.1 Nov 28, 2020

We are pleased to announce the release of Firenado 0.2.3.

Here are the highlights:

Bug Fixes

- Fix python 3.9 warnings. #328

New Features

- Reference data sources in different files. #171

3.14 What's new in Firenado 0.2.2

3.14.1 Oct 09, 2020

We are pleased to announce the release of Firenado 0.2.2.

Here are the highlights:

New Features

- Create and store data sources at run time‘#230 <https://github.com/candango/firenado/issues/230>‘_
- Make a scheduled job to be executed by interval #325

3.15 What's new in Firenado 0.2.1

3.15.1 Sep 11, 2020

We are pleased to announce the release of Firenado 0.2.1.

This release will add couple minor features and updating cartola.

We'll increment the release number instead of minor.

Here are the highlights:

New Features

- Add function to create HTTPRequests‘#131 <https://github.com/candango/firenado/issues/131>‘_
- Create a function to transform sqlalchemy objects to dict #321

3.16 What's new in Firenado 0.2

3.16.1 Aug 30, 2020

We are pleased to announce the release of Firenado 0.2.

This release adds the scheduled component able to run jobs defined in a configuration file. Jobs are scheduled using cron strings.

Now before_handler and after_handler methods from application where renamed to before_request and after_request and they are also present in the handler.

By now releases will increment the minor. Next release will be 0.3.

Here are the highlights:

New Features

- Create scheduled component and module #45
- Implement before and after request methods #315
- Add has conf property to TornadoComponent #316
- Create a method to resolve the config file to TornadoComponent #317

3.17 What's new in Firenado 0.1.9

3.17.1 Jul 23, 2020

We are pleased to announce the release of Firenado 0.1.9.

That release fixes session, launcher issues and fixes installation under pip 20.1 up.

The paginator component was moved to cartola 0.7.

Here are the highlights:

Bug Fixes

- If PYTHONPATH isn't defined launcher will clean sys.path. #306
- Pip 20.1 will break installation. #310
- Redis session prefix isn't working. #311

Refactory

- Remove Paginator from the project. #309

3.18 What's new in Firenado 0.1.8.3

3.18.1 Apr 10, 2020

We are pleased to announce the release of Firenado 0.1.8.3.

Right now python 3.x will install tornado 6 and python 2.7 will keep with tornado 5.1.1.

Here are the highlights:

New Features

- Install new tornado if python is 3.x. #303

3.19 What's new in Firenado 0.1.8.1

3.19.1 Mar 04, 2020

We are pleased to announce the release of Firenado 0.1.8.1.

We still support python 2.7 in the main development contrary to the announce from 0.1.7.8 but we started to relax tests against 2.7 and prepare the code base for a python 3.x only.

Firenado gave birth to Cartola and we added this project to the basic dependency.

This version fixes definitely the process launcher shutdown making easier to spam and teardown Firenado application in a test and that's the reason we still keep with python 2.7.

A new json session encoder were added to the available list of session encoders making it easier to share session data between applications that don't have pikle encoding/serialization.

Here are the highlights:

New Features

- Add json session encoder to the project. #295
- Clean sys.path and set PYTHONPATH before launch a process. . #299

Bug Fixes

- Process launcher shutdown doesn't work. #290

Refactory

- Use pymobiledetect instead of util.browser. #292
- Use cartola instead of util.sysexits. #293
- Use cartola instead of util.file. #294
- Use cartola instead of local security package. #296

3.20 What's new in Firenado 0.1.7.8

3.20.1 Dec 25, 2019

We are pleased to announce the release of Firenado 0.1.7.8.

Last Firenado supporting python 2.7 in the main development we will move Tornado 5.x codebase to a legacy state and only fixing bugs if needed.

Next release we're moving to Tornado 6.x.

We added an extra options to install all extra dependencies and random string and uuid commands to the command line program.

We fixed bugs with processes launcher and added interface to send lines to the process running.

Here are the highlights:

New Features

- Add to setup.py an extra option to install all extras. #280
- Random string and uuid command line tasks #284. #284

Bug Fixes

- ProcessLauncher logfile parameter assignment overrides socket. #279
- Control ProcessLauncher shutdown properly. #281

Examples

- Async timeout example. #283

3.21 What's new in Firenado 0.1.7.7

3.21.1 Sep 07, 2019

We are pleased to announce the release of Firenado 0.1.7.7.

Introduced multi app and multi process configurations. Now its is possible to define and launch more than one applications in the same project with more than one processes.

Here are the highlights:

Refactory

- Move tornado app configurations to app settings. #263

New Features

- Use bind instead listen when starting a Tornado Application. #94
- Multiapp configuration. #177
- Multi process configuration. #264

Bug Fixes

- Using localhost and 127.0.0.1 in the addresses will break the application execution. #229
- Command project init will throw an error if no module name is informed. #268
- Pexpect is being imported when process launcher is still being resolved bug. #273

3.22 What's new in Firenado 0.1.7.6

3.22.1 Dec 22, 2018

We are pleased to announce the release of Firenado 0.1.7.6.

Firenado 0.1.7.5 pip package was not correct use this release instead.

Here are the highlights:

Bug Fixes

- Cannot install 0.1.7.5 because the pexpect is missing on the manifest. #258

3.23 What's new in Firenado 0.1.7.5

3.23.1 Dec 22, 2018

We are pleased to announce the release of Firenado 0.1.7.5.

A new launcher were added to spawn an application in another process and the socket parameter was added to the firenado launcher to overrides the configuration file.

Here are the highlights:

Refactory

- Move random_string to the security module. #247
- Provide a function to return the class reference by it's name. #251
- Rename skell to testapp. #252

New Features

- Add socket parameter to the app run command. #248
- Add a process launcher. #249

Bug Fixes

- Fix command match with parenthesis. #250

3.24 What's new in Firenado 0.1.7.4

3.24.1 Dec 01, 2018

We are pleased to announce the release of Firenado 0.1.7.4.

With launcher loading separately from the launch method we can use it to retrieve applications in management tasks and tests.

Services will retrieve a data connected consumer when calling the get data connected method.

Fixed an attempt to write a none session. This is happening when we have a handler with error set by tornado. The error was noticed in a post method without xsrf argument/cookie. The handler was set with a 403 and other exception as raised because we were waiting for a valid session. Now a warning will be logged regarding the issue of a none session in this case.

Here are the highlights:

Refactory

- Launcher should load the application before launch. #243

New Features

- Service should return the consumer if it is a data connected. #152

Bug Fixes

- When xsrf argument is missing from post we try to write an empty session. #241

3.25 What's new in Firenado 0.1.7.3

3.25.1 Nov 23, 2018

We are pleased to announce the release of Firenado 0.1.7.3.

Now it is possible to change the address and port the application will listen and the application directory using parameters with app run command.

Fixed some bugs and upgraded redis-py to 3.0.1.

Here are the highlights:

Refactory

- Upgrade redis to 3.x. #232

New Features

- Service by decorator must only create the service if not defined. #228
- Add addresses, application directory and port parameters to the app run command. #231

Bug Fixes

- Handle manager help better. #217
- Session purge limit check is broken. #233
- Session read and write breaking when session is disabled. #234
- Tornado Web Socket handler breaking on python 3. #235

3.26 What's new in Firenado 0.1.7.2

3.26.1 Oct 21, 2018

We are pleased to announce the release of Firenado 0.1.7.2.

The system configuration file was implemented and improvements to session and components were added to the framework.

Here are the highlights:

New Features

- Implement system conf file. #222
- Add after and before handler methods to the component. #223
- Add method to execute a sql script to the sqlalchemy util. #225

Bug Fixes

- Run session write after the handler on_finish. #39
- Redis session expiration control is blocking. #221

3.27 What's new in Firenado 0.1.7.1

3.27.1 Oct 05, 2018

We are pleased to announce the release of Firenado 0.1.7.1.

Added some minor features and fixed the management commands stack.

Here are the highlights:

New Features

- Add session parameters to the sqlalchemy data source configuration. #213
- Create a decorator to limit a handler method only to XMLHttpRequest(ajax). #214

Bug Fixes

- Application management commands configuration removes the default framework ones. #216

3.28 What's new in Firenado 0.1.7.0

3.28.1 Aug 24, 2018

We are pleased to announce the release of Firenado 0.1.7.0.

Tornado 5 added to the project and python 3.3 removed from the supported versions.

Here are the highlights:

New Features

- Add tornado 5. #174
- Add to the handler a method to check if user is authenticated. #204
- Add addresses configuration item to the application. #205

Bug Fixes

- Handle empty static maps config files. #175
- The configuration file stream is opened for a while after we load the yaml file. #206

Refactoring

- Replace nose with python unittest. #185

3.29 What's new in Firenado 0.1.6.2

3.29.1 May 06, 2018

We are pleased to announce the release of Firenado 0.1.6.2.

Fixed error installing Firenado on pip 10.

Here are the highlights:

Bug Fixes

- Cannot install on pip 10. #197

3.30 What's new in Firenado 0.1.6.1

3.30.1 March 05, 2018

We are pleased to announce the release of Firenado 0.1.6.1.

This is the release we changed the package to production ready as the session engine will expire and clean invalid session data.

Here are the highlights:

New Features

- Implement session timeout #151
- Session id should be defined by a config item #169
- Provide data_connected method to the service #173

Bug Fixes

- Handle better data source key error during configuration process #187

3.31 What's new in Firenado 0.1.6.0

3.31.1 November 12, 2017

We are pleased to announce the release of Firenado 0.1.6.0.

With this release we added the idea of url root path to the application.

By default the url root path will be set as / and that is ideal for applications that runs over the url root (ie. http://example.com). If an application isn't located at the root of the domain (ie. http://example.com/my/app/address) we should change the url root path to /my/app/adress.

The configuration is set in the firenado.yml:

```
app:
  url_root_path: "my/app/address"
```

A method on the request and an ui module were added to help build links regarding the url root path configuration:

On the request:

```
self.get_rooted_path("/an_url_path")
```

On a template:

```
{% module RootedPath('/an_url_path') %}
```

Both will return <url_root_path>/an_url_path.

It is possible to set the time to wait before the shutdown by a configuration item.

A new toolbox component was added with a full pagination functionality.

Also a shortcut to the firenado project command as added. Here is an example:

```
firenado proj init
```

A bug fix made possible to run firenado on windows. The skell application is running without problems after the fix.

Here are the highlights:

New Features

- Added url_root_path to the app config. #160
- New toolbox component. #161
- Added pagination to the toolbox component. #170
- New wait_before_shutdown configuration item. #178
- Added shortcut proj to the firenado project command. #179

Bug Fixes

- Adding sinal handler for SIGTSTP will break application start on windows. #181

3.32 What's new in Firenado 0.1.5.3

3.32.1 October 09, 2016

We are pleased to announce the release of Firenado 0.1.5.3.

We added couple features to give more flexibility to configure the application and check if the request was sent by a mobile device.

The Tornado examples to handle Facebook and Twitter Oauth were added to the project and a Google one was created based on the Twitter one.

Here are the highlights:

New Features

- Create method to check if the handler is dealing with a mobile. #161
- Add static_url_prefix to the config file. #162
- Add settings configuration list to the app config. #163

Examples

- Create examples using facebook, twitter and google authentication. #164

3.33 What's new in Firenado 0.1.5.2

3.33.1 September 18, 2016

We are pleased to announce the release of Firenado 0.1.5.2.

This release add some features to secure cookies, prevent xsrf attacks and add login urls to the app.

Here are the highlights:

Features

- Add cookie_secret to the config core feature session tornadoweb. #123
- Add login url to the app config . #153
- Add xsrf_cookies to the app config. #154
- Add management task to generate random cookie secrets. #155

3.34 What's new in Firenado 0.1.5.1

3.34.1 September 10, 2016

We are pleased to announce the release of Firenado 0.1.5.1.

This release fixes some management tasks bugs and handles the sqlalchemy data source disconnect issue better.

Here are the highlights:

Bug Fixes

- Firenado project init is broken. #138
- Sub-command help is not being parsed correctly. #139
- Sqlalchemy session has errors after some time. #145

3.35 What's new in Firenado 0.1.5.0

3.35.1 September 01, 2016

We are pleased to announce the release of Firenado 0.1.5.0.

On this release the build was improved in order to provide extra requirements.

Right now it is possible to install redis and sqlalchemy using extra requirements with pip or on the requirements file.

Python 2.6 and 3.2 are not supported on this version to comply with Tornado 4.4.1.

Here are the highlights:

Refactory

- Depreciate python 2.6 and 3.2. #118
- Change application file from application.py to app.py. #120

New Features

- Create the static maps component. #57
- Add optional requirements to setup.py. #119

Bug Fixes

- Fix managment command help. #13
- Six is not being installed by setup.py. #121
- Sqlalchemy connection is missed when database service restarts or the connection is renewed. #127

3.36 What's new in Firenado 0.1.4

3.36.1 April 9, 2016

We are pleased to announce the release of Firenado 0.1.4.

On this release the focus was to get the framework refactored to be more "Pythonic".

This version fixes couple bugs and introduce a way to set the application static_path via configuration.

Here are the highlights:

Refactoring

- Package refactoring to move code from __init__.py's #109

New Features

- Added static_path to the config file #104

Bug Fixes

- A disabled session still needs a session type #91
- The service by decorator is broken #107
- The initialize method runs just if the component has a file defined #103

3.37 What's new in Firenado 0.1.3

3.37.1 January 31, 2016

We are pleased to announce the release of Firenado 0.1.3.

On this release yml config files will be used by default but yaml files will be considered also.

Services can be served by other services and couple more minor features were added to the framework.

No bug fixes were covered on this release.

Here are the highlights:

New Features

- Served_by decorator can be used by another service. #91
- Components can add ui_modules to an application. #85
- Tornado application debug mode added. #84
- Config files can be used with yml and yaml. Yml is default. #83
- Firenado was prepared to start with application types. Tornado is default. #41
- A Firenado Tornado application can be set to start listening on a unix socket. #38

3.38 What's new in Firenado 0.1.2

3.38.1 October 30, 2015

Hotfix release to solve a severe bug that was preventing firenado command to run.

Here are the highlights:

New Features

- Added root path to the assets component. #77

Bug Fixes

- Firenado doesn't run because core.management.tasks was not shipped on 0.1.1. #76

3.39 What's new in Firenado 0.1.1

3.39.1 October 28, 2015

We are pleased to announce the release of Firenado 0.1.1. Now we start to support Python 3 and have fixed some bugs fixed.

The project CI is being tracked by travis and tests are being added to the project.

Here are the highlights:

New Features

- Python 3 support
- Added installation capabilities to the framework.

Migration from iFlux to Firenado

- Security (It will be decided if this module still be needed on 0.1.2).

Bug Fixes

- Firenado requirements are not wired to the setup script. #52
- When Redis connection fails application exits without warning. #58
- Deadlock with the string generation on the util package. #64

3.40 What's new in Firenado 0.1.0

3.40.1 September 28, 2015

We are pleased to announce the release of Firenado 0.1. This was a complete rewrite from the iFlux framework an here are the highlights:

New Features

- Yaml based configuration files

Migration from iFlux to Firenado

- Data tools
- Modules
- Session
- Services

Migration changes

- Configuration
- Command line tool

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

4.1 Discussion and support

Report bugs on the GitHub issue tracker.

Firenado is one of Candango Open Source Group initiatives. It is available under the Apache License, Version 2.0.