# ExAssist Documentation

Flyaway

Jan 05, 2021

# Contents:

**1 Quickstart** **3**
1.1 Installation . . . . . . . . . . 3
1.2 Hello World . . . . . . . . . . 3

**2 Gathering infomation** **7**
2.1 Create an Assist . . . . . . . . . . 7
2.2 Observe an Experiment . . . . . . . . . . 8
2.3 Assist an Experiment . . . . . . . . . . 9
2.4 Deactivate . . . . . . . . . . 9

**3 Indices and tables** **11**

`ExAssist` is an light-weight assist tool that can save your time from doing experiments. It is designed to help you with:

1. Track the configurations for each experiment.
2. Record any temporary data during experiments.
3. Gather host and environment information for each experiment.

# CHAPTER 1

# Quickstart

## 1.1 Installation

You can install ExAssist directly from pypi like this:

```
pip install ExAssist
```

Also, you can clone the git repo and install it from source file:

```
git clone
cd ExAssist
python setup.py install
```

## 1.2 Hello World

Let's directly jump into it. Here is a miniml experiment using ExAssist:

```
import ExAssist as EA

# Get an instance of ExAssist just like getting a logger.
assist = EA.getAssist('Test')

with EA.start(assist) as assist:
    # Here starts your experiments.
    for i in range(100):
        assist.info['loss'] = 100 - i
        assist.step()
```

**We did following things here:**

- import `ExAssist`
- get a ExAssist instance just like logging library.

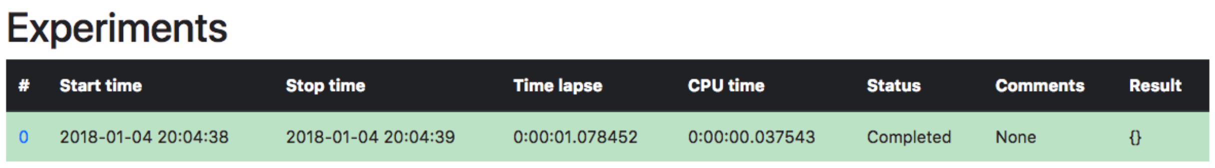
- create a experiment context
- run your own experiments within this context

**Note:** Different from the logginer, ExAssist does not have hierarchy structure. So, `ExAssist.get('a.b')` is useless.

Once you run this simple script, ExAssist will create a directory called `Experiments`, which contains all the information of current experiment. Enter the Experiments directory, run the following code:

```
python -m http.server 8080
```

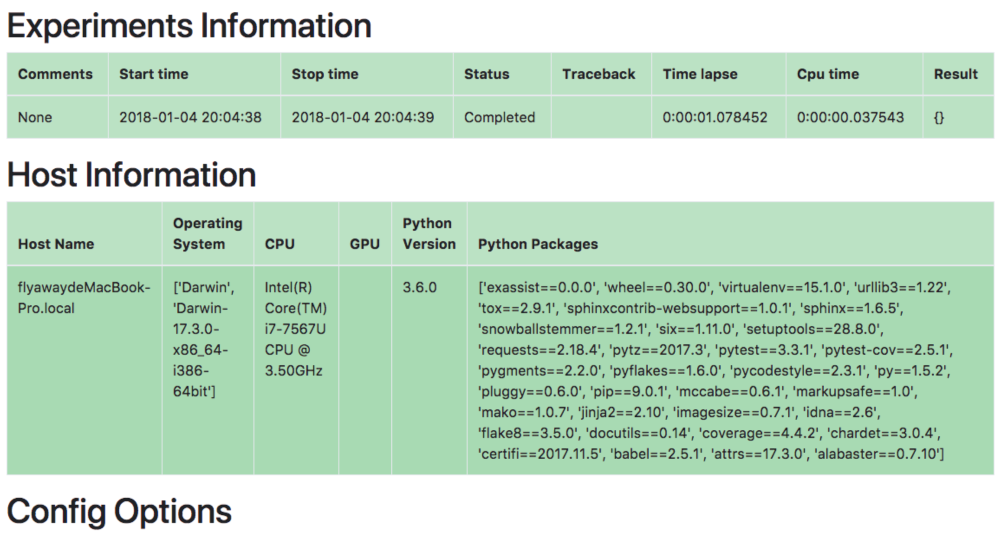
Now, open your browser and open page at http://localhost:8080/. You will see an index page like:



Experiments

| # | Start time | Stop time | Time lapse | CPU time | Status | Comments | Result |
|---|---|---|---|---|---|---|---|
| 0 | 2018-01-04 20:04:38 | 2018-01-04 20:04:39 | 0:00:01.078452 | 0:00:00.037543 | Completed | None | {} |

This table shows basic infomation of your experiments. Click on the number of experiment, you will see more detialed information of this experiment:



## Experiments Information

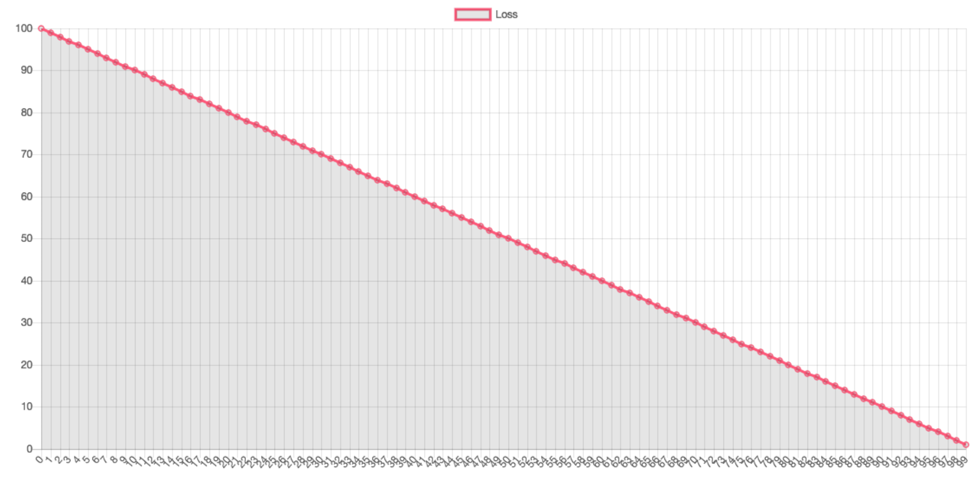
| Comments | Start time | Stop time | Status | Traceback | Time lapse | Cpu time | Result |
|---|---|---|---|---|---|---|---|
| None | 2018-01-04 20:04:38 | 2018-01-04 20:04:39 | Completed | | 0:00:01.078452 | 0:00:00.037543 | {} |

## Host Information

| Host Name | Operating System | CPU | GPU | Python Version | Python Packages |
|---|---|---|---|---|---|
| flyawaydeMacBook-Pro.local | ['Darwin', 'Darwin-17.3.0-x86_64-i386-64bit'] | Intel(R) Core(TM) i7-7567U CPU @ 3.50GHz | | 3.6.0 | ['exassist==0.0.0', 'wheel==0.30.0', 'virtualenv==15.1.0', 'urllib3==1.22', 'tox==2.9.1', 'sphinxcontrib-websupport==1.0.1', 'sphinx==1.6.5', 'snowballstemmer==1.2.1', 'six==1.11.0', 'setuptools==28.8.0', 'requests==2.18.4', 'pytz==2017.3', 'pytest==3.3.1', 'pytest-cov==2.5.1', 'pygments==2.2.0', 'pyflakes==1.6.0', 'pycodestyle==2.3.1', 'py==1.5.2', 'pluggy==0.6.0', 'pip==9.0.1', 'mccabe==0.6.1', 'markupsafe==1.0', 'mako==1.0.7', 'jinja2==2.10', 'imagesize==0.7.1', 'idna==2.6', 'flake8==3.5.0', 'docutils==0.14', 'coverage==4.4.2', 'chardet==3.0.4', 'certifi==2017.11.5', 'babel==2.5.1', 'attrs==17.3.0', 'alabaster==0.7.10'] |

## Config Options

## Loss Curve



# CHAPTER 2

# Gathering infomation

Assist is the only class in ExAssist framework. This section provides basic usage of Assist.

## 2.1 Create an Assist

In order to use ExAssist to assist your experiment, you first need to instantiate an intance of Assist:

```
import ExAssist as EA
assist = EA.getAssist('Test')
```

When calling getAssist method, ExAssist will instantiate a new Assist instance if it does not exist. The name Test is the unique identifier for the new Assist instance. Just like logging you can access the same instance anywhere in your source code:

```
assist = EA.getAssist('Test')
```

After getting an instance of Assist, you need to provide the following information:

- The root path of your experiemnt records.(default: ./Experiments/)
- A config variable that keeps all the configurations of your experiments. This config variable could be:
  - A dict object that contains all the configurations as (key, value)
  - A argparse.namespace object. This namespace object should contains all the configurations.
  - A ConfigParser object that loads configurations from config files.

A simple example is like this:

```
# Get the Assist instance
assist = EA.getAssist('Test')
# Setup the root path of experiment records
assist.ex_dir = 'tests/Experiments/'
```

(continues on next page)

(continued from previous page)

```
# Setup the config variable
assist.set_config(config)
```

Once setting up all the informaion, you can start your experiments.

## 2.2 Observe an Experiment

The first function of ExAssist is to observe en experiment automatically. ExAssist uses context manager to observe your experiment:

```
with EA.start(assist) as assist:
    # Your experiment happens here
```

**When you entering this context, ExAssist will automatically:**

- Create a uniqe directory which will be used to save all the information about this experiment.
- Gather meta information about your experiment, like your starting time and environment information.

**Note:** Once entering the context, you can not modify the basic information about `Assist` covered in last section, see *Create an Assist*.

**When you finish running your experiment and leaving this context, ExAssist will automatically:**

- Record the status (success, failed or interrupted) of this experiment
- Generate an html file that contains all the information of this experiment.

ExAssist collects lots of information about an experiment:

- time it was started, time it stopped and cpu time it used.
- the used configuration
- status of this experiment
- basic information about the machine it runs on
- packages the experiment depends on and their versions
- data added with `assist.info`
- data added with `assist.result`

### 2.2.1 Directory Structure

All the information (except the last two points) above is gathered and saved automatically, you don't need to write any code. For each experiment running, ExAssist will create a new sub-directory in the path of `ex_dir` and stores several files in there:

```
Experiments/
├── 0
│   ├── config.json
│   ├── index.html
│   ├── info.json
│   └── run.json
```

(continues on next page)

(continued from previous page)

```
├── 1
│   ├── config.json
│   ├── index.html
│   ├── info.json
│   └── run.json
```

As we can see above, ExAssist will also generate a report (`index.html`) for each run.

## 2.3 Assist an Experiment

The second function of ExAssist is to assist your experiment. It gives the abilities:

- Record the running information without writing extra IO functions. ExAssis can help you save all the temporary information during the experiment, such as loss and gradients.

```python
import ExAssis as EA

assist = EA.getAssist('Test')
with EA.start(assist) as assist:
    # Here starts your experiments.
    for i in range(100):
        assist.info['loss'] = 100 - i
        assist.step()
```

In the code above, we record `loss` value for each iteration. Method `step()` tells ExAssist that the current iteration is finished. `assis.info` is dictionary which means you can put anything you want into this variable. The `info` dictionary is meant to store temporary information about the experiment, like training loss for each epoch or the total number of parameters. It is updated once you invoke `step` method. You can add whatever information you like to `info`. Code in the above will generate a list like this:

```
[{'loss':100}, {'loss':99}, {'loss':98}, ...]
```

Once you entering the context, you can access and update following variables:

- `assist.info`: You can use `info` to save any temporary value that you need to analysis, like traning loss.
- `assist.result`: `result` are designed to keep the evaluation results of this experiment. `result` does not affeced by `step()` method.
- `assist.run_path`: Read-only. You can access the path of current experiment data. This is useful when you want to save your model in the same directory with its meta information.
- `assist.epoch`: Read-only. Indicates the internal epoch number of ExAssist. It increases every time when you invoke `step()` method.

## 2.4 Deactivate

When publishing the code, you usually do not want ExAssist to observe any experiments. You can deactivate ExAssist by:

```python
import ExAssis as EA

assist = EA.getAssist('Test')
assist.deactivate()
with EA.start(assist) as assist:
    # Here starts your experiments.
    for i in range(100):
        assist.info['loss'] = 100 - i
        assist.step()
```

By invoking `deactivate()`, ExAssis will not do anything during run as if it does not exist.

# CHAPTER 3

# Indices and tables

- genindex
- modindex
- search