project_manager Documentation

Release 0.0.1

kpj

May 06, 2019

Contents:

A utility which makes running the same projects with various configurations as easy as pie.

CHAPTER 1

Installation

```
$ pip install project_manager
```

CHAPTER 2

Usage

2.1 Configuration file

Create a configuration:

```
project_source: <url or path>  # project you want to run
working_dir: <path>  # where everything will run

exec_command:  # list of commands that will be executed in each project setup
    - <python ..>
result_dirs:  # list of files/folders that will be extracted after successful␣
↪execution
    - <result dir>

base_config: <path>  # path to the raw configuration file (typically part of your␣
↪project)
symlinks:  # list of symlinks to include in each project setup
    - <path 1>
    - <path 2>
config_parameters:  # how to modify the configuration
    - key: param1
      values: [0, 1, 2]
      paired:
        - key: param2
          values: [a, b, c]
    - key: [nested, param3]
      values: ['a', 'b', 'c']
extra_parameters:  # special extra parameters
    git_branch: ['master']
    repetitions: 1
```

2.2 Commands

After setting up the configuration file, you can run all commands.

```
$ project_manager build
$ project_manager run
$ project_manager gather
```

In order, these commands do the following:

1. Create individual folders for each run and adapt the configuration accordingly
2. Run the specified commands per previously created setup
3. Retrieve all specified results into a single directory. Each individual files is annotated with its origin.

2.3 Example

A quick usage overview:

```
$ tree
.
├── config.yaml
└── dummy_project
    ├── my_conf.yaml
    └── run.py
```

config.yaml:

```
project_source: dummy_project
working_dir: tmp

exec_command:
    - python3 run.py
result_dirs:
    - results

base_config: dummy_project/my_conf.yaml
config_parameters:
    - key: message
      values: [A, B, C]
```

dummy_project/my_conf.yaml:

```
message: 'this is important'
```

run.py:

```
import os
import yaml


def main():
  with open('my_conf.yaml') as fd:
      config = yaml.load(fd)
```

(continues on next page)

(continued from previous page)

```
  os.makedirs('results')
  with open('results/data.txt', 'w') as fd:
      fd.write(config['message'])


if __name__ == '__main__':
  main()
```

We can then run the pipeline:

```
$ project_manager build
Setting up environments: 100%|| 3/3 [00:00<00:00, 477.57it/s]
$ project_manager run
run.message=A
 > python3 run.py
run.message=B
 > python3 run.py
run.message=C
 > python3 run.py
$ project_manager gather
run.message=A
 > data.txt
run.message=B
 > data.txt
run.message=C
 > data.txt
```

Here's the result:

```
$ tree tmp/
tmp/
├── aggregated_results
│   └── results
│       ├── data.message=A.txt
│       ├── data.message=B.txt
│       └── data.message=C.txt
├── run.message=A
│   ├── my_conf.yaml
│   ├── results
│   │   └── data.txt
│   └── run.py
├── run.message=B
│   ├── my_conf.yaml
│   ├── results
│   │   └── data.txt
│   └── run.py
└── run.message=C
    ├── my_conf.yaml
    ├── results
    │   └── data.txt
    └── run.py
$ cat tmp/aggregated_results/results/*
ABC
```