

d) Commandes Pytest et html report

Les commandes pytest

À venir

HTML Reporting

Tout d'abord, installons la librairie qui nous permettra d'avoir des rapports de test plus simples à lire.

```
pip install pytest-html
```

Une fois que cette librairie est installée, la commande pytest sera enrichie d'arguments. Tapons tout simplement:

```
pytest --html=report.html
```

Et allons voir ensemble la page qui s'est créée à l'endroit où nous avons lancé la commande :

report.html

Report generated on 19-Feb-2021 at 15:24:53 by pytest-html v1.22.1

Environment

Packages	{'py': '1.10.0', 'pytest': '4.6.11', 'pluggy': '0.13.1'}
Platform	Darwin-19.0.0-x86_64-i386-64bit
Plugins	{'metadata': '1.11.0', 'html': '1.22.1', 'cov': '2.11.1'}
Python	2.7.16

Summary

1 tests ran in 0.11 seconds.

(Un)check the boxes to filter the results.

✓ 0 passed, ✓ 0 skipped, ✗ 1 failed, ✓ 0 errors, ✓ 0 expected failures, ✓ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Failed (hide details)	test_app.py::test_http_return	0.00	
<pre>tmpdir = local('/private/var/folders/50/f7j6kckd5dd_37g6c1nptmw000gn/T/pytest-of-keryiclain/pytest-20/test-http_return0'), monkeypatch = <_pytest.monkeypatch.MonkeyPatch object at 0x10b874350> def test_http_return(tmpdir, monkeypatch): p = tmpdir.mkdirc("program").join("test.txt") # run script app.main(["--dest", str(p)]) print(p) local_res = "Hello" with open(str(p), 'r') as f: test = f.readlines()[0].strip() > assert local_res == test E AssertionError: assert 'Hello' == 'Hello world' E E Hello E Hello world test_app.py:14: AssertionError ----- Captured stdout call ----- /private/var/folders/50/f7j6kckd5dd_37g6c1nptmw000gn/T/pytest-of-keryiclain/pytest-20/test-http_return0/program/test.txt</pre>			

C'est bien plus compréhensible que de lire dans une console, non ?!

Json reporting

Si l'on souhaite créer notre propre interface, on pourra créer des reports sous forme de json. Pour cela, on installera la librairie à partir de la commande suivante:

```
pip install pytest-json-report
```

Et on utilisera les arguments de la librairie json :

```
pytest --json-report-file=json_report.json --json-report --json-report-indent=2 --json-report-summary
```

Définition des arguments:

- -json-report demande à ce que l'on crée un report en json
- -json-report-file=PATH, indique le nom du fichier json qui sera créé
- -json-report-indent=2, on met un peu en forme
- -json-report-summary, on n'affiche que le principal sinon c'est illisible !

Résultat :

```
{
  "environment": {
    "Python": "2.7.16",
    "Platform": "Darwin-19.6.0-x86_64-i386-64bit",
    "Packages": {
      "py": "1.10.0",
      "pytest": "4.6.11",
      "pluggy": "0.13.1"
    },
    "Plugins": {
      "json-report": "1.2.4",
      "metadata": "1.11.0",
      "html": "1.22.1",
      "cov": "2.11.1"
    }
  },
  "created": 1613745598.360417,
  "duration": 0.11444091796875,
  "exitcode": 1,
  "root": "/Users/kerylclain/Desktop/test",
  "summary": {
    "collected": 1,
    "total": 1,
    "error": 1
  }
}
```