

Sommaire

- Règles de nommage python
- Les tests
- Git
- REST
- Architecture N-tiers

Règles de nommage python

- Pourquoi des règles de nommages ?
 - Facilite la maintenabilité.
 - Permet à plusieurs personnes de travailler sur le même code.

Règles de nommage python

Check si les conventions sont respectées

```
1 import unittest
2 import json
3 from unittest.mock import patch
4
5 from app import app
6 from app.database.database import Database
7 from tests.utils import content_type
8 from tests.utils.mock_connection_factory import MockConnectionFactory
9
10
11 class TestUserController(unittest.TestCase):
12
13     @patch.object(Database, 'get_connection', new=MockConnectionFactory.get)
14     def test_create(self):
15         Database.execute_schema()
16         test_app = app.test_client()
17
18         response = test_app.post('/users', data=json.dumps({
19             "firstname": "test"
20         }), content_type=content_type.JSON)
21         assert response.status_code == 200
22         user_id = response.data.decode('utf-8')
23
24         response = test_app.get('/users/' + user_id)
25         assert response.status_code == 200
26         user = json.loads(response.data)
27         assert user['firstname'] == 'test'
28
29     @patch.object(Database, 'get_connection', new=MockConnectionFactory.get)
30     def test_get_null(self):
31         Database.execute_schema()
32         test_app = app.test_client()
33
34         response = test_app.get('/users/1')
35         assert response.status_code == 404
36
```

Nom classe en camel case

Noms méthodes
séparés par underscore

Nom variables séparés
par underscore

4 espaces d'indentation

Espaces de chaque
côtés du signe égal



Règles de nommage python

- Eviter les lignes trop longues (80 char max) :

```
30 TriggerClientMocker.init() \
31     .with_file(url='/contracts/sps/firstactivecontracts', file='data_05.json') \
32     .default_data(data={'results': []}) \
33     .start()
```

- Eviter les chaines de caractères qui pourraient être utilisées souvent => utiliser des constantes à la place :

```
66     return time.strftime(DateFormat.ISO)
```

Les tests

- Pourquoi faire des tests ?
 - Eviter les régressions.
 - Etre plus efficace lors du développement.
 - Plus grande confiance dans le code.

Les tests

- Les différents types de tests:
 - Tests unitaires : une classe, une méthode.
 - Tests fonctionnels : une fonctionnalité.
 - Tests d'intégrations : plusieurs composants entre eux (plusieurs serveurs, interfaces graphiques, bases de données...).

Les tests

- TDD (test driven development):
 - Tester d'abord, développer ensuite.

“Puisque les tests sont utiles, ils seront faits systématiquement avant chaque mise en œuvre”

- Deuxième principe de l'extreme programming

Les tests

- Les mocks:
 - Utilisés dans le cadre de tests unitaires ou fonctionnels.
 - Permet de limiter le test à une partie de l'application.
 - Utilisé notamment pour éviter d'appeler une base de donnée ou un service externe à l'application testée.

Git

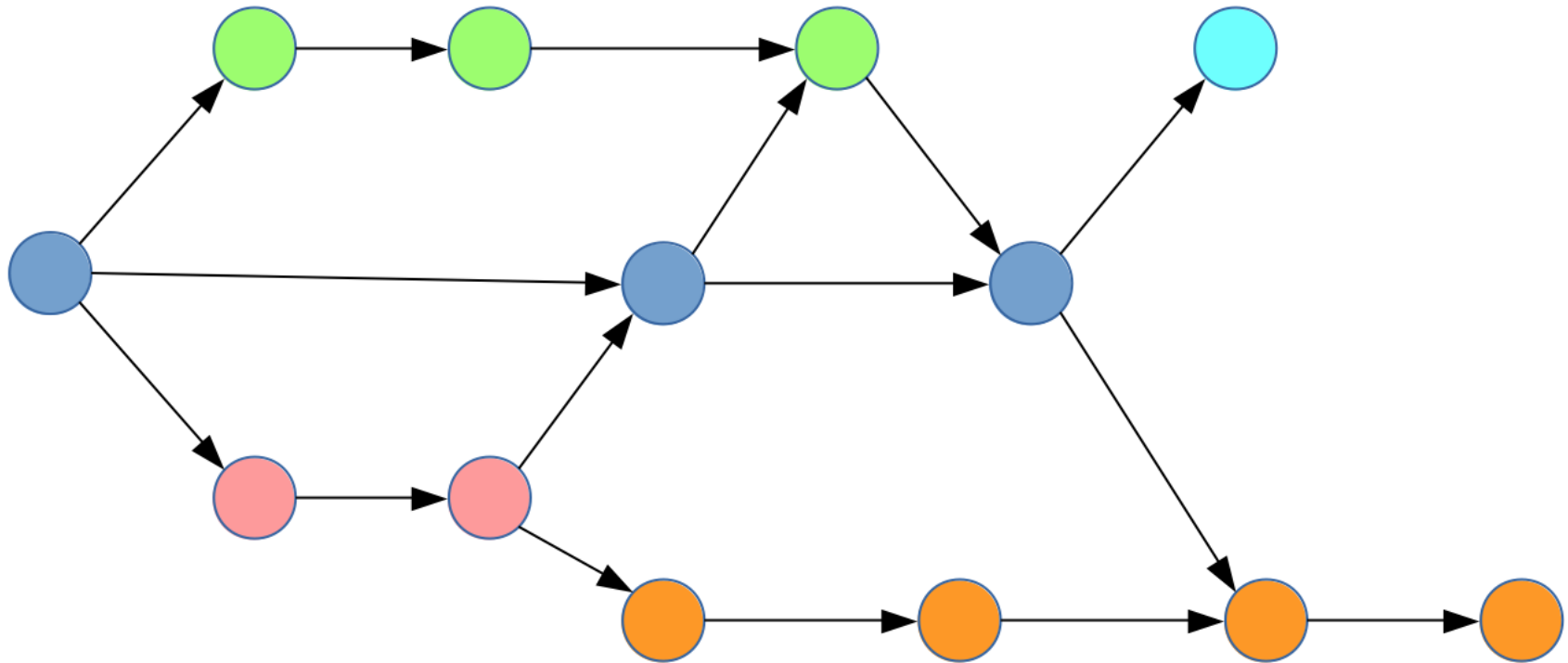
- Commandes à retenir:

- git checkout
- git branch
- git commit
- git pull
- git push
- git merge
- git clone

“Puisque l'intégration des modifications est cruciale, nous l'effectuerons plusieurs fois par jour”

Sixième principe de l'extreme programming

Git



REST

- Services webs basés sur les verbes HTTP et les erreurs HTTP.
- 4 verbes très couramment utilisés:
 - GET : pour la lecture.
 - POST : pour la création.
 - PUT : pour la modification.
 - DELETE : pour la suppression.

REST

- Codes réponse courants :
 - 200 : tout s'est bien passé.
 - 201 : création réussie.
 - 400 : erreur dans la requête du client.
 - 401 : le client n'est pas authentifié.
 - 403 : le client n'a pas les droits d'accès requis.
 - 404 : la ressource demandée n'existe pas.
 - 500 : erreur interne du serveur.

Architecture N-tiers

- Architecture serveur organisée en couche applicative.
- Les couches sont généralement les suivantes :
 - Controllers : chargé de recevoir les requêtes et d'envoyer les réponses.
 - Services : contient le code métier de l'application.
 - DAO : chargé de lire et d'écrire dans la base de données.
- Les appels se font comme suit :
 - Controller → service → DAO