Testowanie

Testy jednostkowe

Plan

- Koncepcja
- Przegląd narzędzi
- JUnit 4.x, Java
- unittest, Python

Koncepcja

- Celem testów jednostkowych jest sprawdzenie pojedynczej jednostki programowej – np. metody, klasy, grupy klas, modułu, funkcji itp.,
- Testy jednostkowe najczęściej są tworzone przez programistę.
- Testy jednostkowe najczęściej są automatyzowane.
- Cechy: niezależność, powtarzalność, jednoznaczność, jednostkowość.

Dostępne narzędzia

- JUnit
  - Jako element aplikacji standalone,
  - MVN,
  - Spring...
- Python - unittest
- TestNG
- NUnit
- Visual Studio
- ...

JUnit

- 3.8
- 4.x
- Integracja np. ze środowiskiem Eclipse/IntelliJ
- xUnit

JUnit

- Podstawowe elementy (dla wersji 4.x):
  - POJO,
  - adnotacje,
  - inicjowanie i finalizowanie (na poziomie metody, klasy),
  - obsługa oczekiwanych wyjątków,
  - obsługa limitów czasowych,
  - wachlarz metod assert...
  - jedna metoda – jeden przypadek testowy.

JUnit

- Adnotacje służą oznaczaniu:
  - przypadków testowych - @Test,
  - oczekiwanych wyjątków - @Test(expected=Exception.class),
  - inicjowania i finalizowania:
    - Poziom testu: @Before, @After,
    - Poziom klasy: @BeforeClass, @AfterClass.

JUnit

- Testowanie składowych niepublicznych:
  - Może zaistnieć potrzeba przetestowania składowych – pól i metod – niepublicznych,
  - Biblioteka JUnit dostarcza mechanizm, który pozwala zmienić widoczność składowych.

unittest

- Podstawowe elementy
  - Podobna struktura do Junit (3.8),
  - Klasy testujące dziedziczą z unittest.TestCase
  - Metody testujące test_...
  - inicjowanie i finalizowanie (na poziomie metody, klasy),
  - obsługa oczekiwanych wyjątków,
  - wachlarz metod assert...
  - jedna metoda – jeden przypadek testowy.

unittest

- Inicjowanie i finalizowanie
  - def setUp(self):
  - def tearDown(self):

JUnit/unittest

- Dane testowe:
  - Należy uważać na położenie danych testowych – testy powinny się dać uruchomić w różnych lokalizacjach, zewnętrzne zasoby mogą stanowić problem,
  - Dane można umieścić wewnątrz testów,
  - Należy ostrożnie operować na danych zależnych od lokalizacji – np. daty.

JUnit/unittest

- Kolejność wykonania:
  - JUnit z założenia nie zapewnia kolejności wykonania przypadków testowych,
  - Dla JUnit istnieją mechanizmy, za pomocą których możliwe jest wymuszenie kolejności.
  - Dla biblioteki unittest kolejność wykonania wynika z sortowania nazw przypadków testowych.

JUnit/unittest

- Testowanie wyjątków:
  - Przypadek testowy może sprawdzać czy w określonej sytuacji system rzuca wyjątek,
  - Przypadek testowy zakończy się wtedy niepowodzeniem, gdy system oczekiwanego wyjątku nie rzuci,
  - JUnit:
    - @Test(expected=Exception.class),
  - unittest:
    - with self.assertRaises(Exception):
      self.g.translate();

JUnit/unittest

- Precyzja:
  - Szczególną uwagę należy przyłożyć do testowania liczb zmiennoprzecinkowych – liczby pozornie równe mogą być reprezentowane różnie, z różną dokładnością,
  - Należy korzystać z dedykowanych porównań, z uwzględnieniem precyzji.

Konwencje

- should...
- given/when/then

Podsumowanie

- JUnit 4.x
  - http://www.junit.org/
- unittest
  - https://docs.python.org/3/library/unittest.html

Q&A