Czym są testy jednostkowe [ang. unit tests]? Jest to test działania małego wycinka systemu - jednostki. Np. metody.Testujemy jedną klasę sprawdzając, czy dla danych wejściowych zwróci oczekiwany rezultat. Z pewnością nie raz robiłeś w kodzie coś takiego (pseudokod):

{codecitation class="brush: php;"}function dodawanie(a, b) {
    return a + b;
 }
 c = dodawanie(2, 5);
 if (7 != c) {
    print "Ej, coś jest nie tak z dodawaniem dla 5 i 2.";
 }{/codecitation}

Właśnie na tym polegają unit testy. Są jednak specjalne biblioteki, które sprawiają, że:

- Nie musisz ingerować w swój kod (powyższy przykład to robił).
- Mogą zostać utrwalone i odpalane po każdej zmianie kodu.
- Niektóre frameworki do unit testów potrafią w bardzo ładny graficzny sposób prezentować wyniki testów.

Zalety testów:

- Pisząc testy dokładnie wiesz jakie dane mają zostać zwrócone przez funckję / metodę dla jakich danych wejściowych. Tak więc zanim zdążysdz napisać linię kodu właściwego już dokładnie wiesz, co on ma robić. często ta wiedza odpowiada także na pytanie: "jak ma to robić?".
- Programując "pod testy" nie możesz sobie pozwolić na metody robiące wszystko! Autoamtycznie starasz się, aby kod był podzielony na logiczne części, łatwe do testowania
- Możesz sobie na więcej pozwolić! Mając zestaw testów pozwalający sprawdzić zachowanie programu w każdej sytuacji możesz sobie czasem pozwolić na dziwne konstrukcje czy uproszczenia - po ponownym uruchomieniu testów (jeśli nadal wszystkie zostaną zaliczone) wiesz, że nic nie popsułeś. Bardzo przydatne podczas refaktoryzacji.
- Testy są jednocześnie dokumentacją projektu i twojej pracy. Nikt nie zarzuci Ci, że coś nie działa z powodu twojej klasy. Starczy, że dodasz (jeśli jeszcze nie masz) test, pokrywający

sytuację o której ktoś mówi i sprawdzasz, czy kod zwróci to czego oczekujesz.

- Pozwalają lepiej zrozumieć istniejący kod. Czasem w wyniku działania testów zauważasz, że twój system działa inaczej niż wydawało ci się, że powinien. Co wcale nie musi znaczyć, że działa błędnie - ty po prostu źle rozumiałeś to jakby on miał się zachować w danej sytuacji.

Wady testów

- Wymagają oderwania od "normalnej pracy". Choć w definicji "normalnej pracy" testowanie i tak występuje. Tyle tylko, że jest bardziej chaotyczne i najczęściej wyklikane. Ile razy już zapomniałeś o jakiejś jednej możliwości? Pisząc testy zawsze możesz dodać kolejny jeśli coś ci przyjdzie do głowy.
- Uważa się, że jest to praca "destrukcyjna" - w przeciwieństwie do kodowania - pracy "kreatywnej". Niesłusznie. Testy pozwalają wykryć błędy, co pozwala je naprawić, co pozwala tworzyć lepsze programy! A więc to praca jak najbardziej kreatywna.

- Same w sobie nie zapewniają poprawności działania programu. To, że twój program przeszedł poprawnie 120 testów, jakie stworzyłeś nie znaczy, że nie wyłoży się na 121-szym teście, który przyjdzie ci do głowy za późno i już klient wykryje ten błąd. Jednak po naprawieniu tego buga możesz uruchomić 121 testów i być pewnym, że załatanie nowo odkrytej dziury nie zepsuło niczego, co działało wcześniej! A to już jest wiele.

Tak naprawdę czy się do tego przyznajesz czy nie robisz testy. Tyle tylko, że często sam sobie robisz pod górkę zmieniając kod testowanej klasy, aby można było wyświetlić wartości poszczególnych zmiennych, czy też za każdym razem wpisując w formularz 40 różnych rodzajów danych. Czy nie czas to zmienić? Zautomatyzować? Ulepszyć?

Jeśli podoba ci się ta idea, czytaj dalej

W najbliższym czasie mam zamiar napisać kilka artykułów o testowaniu kodu (niedawno zacząłęm się tym bawić, więc najpierw sam się muszę podszkolić :P). Oto lista tematów jakie mam zamiar poruszyć z linkami do źródeł. Jeśli chcesz, możesz nie czekać na mnie:)

- Simple test - quick start
- PHPUNIT - quick start
- Selenium