Karol Kasior:

Projekt własnego środowiska testowego przeznaczonego do testów automatycznych

Słowa kluczowe:
Testy, testowanie oprogramowania, testowanie automatyczne, programowanie, Selenium

Wprowadzenie

Tworzenie oprogramowania składa się z wielu aspektów, zarówno biznesowych jak i technicznych. Umiejętne ich połączenie i dopracowanie każdego z nich, przekłada się na powodzenie całego procesu. Głównym aspektem technicznym wytwarzania oprogramowania jest oczywiście pisanie kodu - programowanie. “Przerabianie” instrukcji wcześniej uzyskanych od klienta czy analityka biznesowego na techniczne funkcjonalności. Kiedy mamy już gotową funkcjonalność - poprawnie działający kod programu, przed zaprezentowaniem działającej funkcji systemu czy projektu, powinniśmy ją dokładnie przetestować. To właśnie testowanie oprogramowania jest punktem pośrednim pomiędzy warstwą programowania a prezentacją osiągniętych rezultatów na podstawie kodu programu.
Testowanie oprogramowania polega na weryfikacji działania danej funkcjonalności dostarczonej(najczęściej napisanej) przez programistę oraz dostarczenie informacji na temat przebiegu i rezultatu procesu testowego. Jest to jeden z głównych i najważniejszych procesów wytwarzania oprogramowania. Właśnie testowanie ma na celu potwierdzenie czy kod spełnia wymagania narzucone przez klienta. Testy powinny być napisane i przeprowadzone w taki sposób aby przede wszystkim sprawdzać czy wszystkie wymagania akceptacyjne zostały spełnione.
Jednak aby zapewnić produkt najwyższej jakości, warto wdrożyć i przeprowadzać testowanie na wielu poziomach cyklu wytwarzania oprogramowania. W zależności od tego w jakim momencie procesu tworzenia projektu lub systemu się znajdujemy, stosujemy różne typy i metody testowania:

- Testy modułowe - polegające na testowaniu poszczególnych komponentów oprogramowania - Testy jednostkowe, testowanie wartości brzegowych.
- Testy integracyjne - przeprowadzane w celu weryfikacji działania interakcji pomiędzy zintegrowanymi modułami
- Testy systemowe - weryfikacja działania zintegrowanego systemu na podstawie przyjętych wymagań systemowych - dotyczy zarówno wymagań funkcjonalnych jak i niefunkcjonalnych(wydajność, użyteczność, niezawodność systemu)
- Testy akceptacyjne - w środowisku użytkownika: szereg badań przeprowadzanych po stronie klienckiej w celu odrzucenia bądź akceptacji i późniejszym badaniu w środowisku produkcyjnym; - w środowisku produkcyjnym - testy przeprowadzane zazwyczaj przez operatora lub administratora systemu, w symulowanym rzeczywistym, docelowym środowisku projektu

Przez wszystkie cykle testowania oprogramowania wykorzystujemy testy manualne - przeprowadzane ręcznie przez testera lub użytkownika. W wielu dużych projektach informatycznych jest to bardzo mozolna i czasochłonna praca. Często potęgowana zmieniającymi się wymaganiami projektu czy też dodawaniem nowych funkcjonalności w procesie powstawania systemu. Nawet wykonywana przez cały zespół testerów znacznie

spowalnia cykl wytwarzania oprogramowania. W wielu przypadkach w komercyjnych projektach, musimy zapewnić ogrom konfiguracji, lub danych wejściowych, pozwalających na zweryfikowanie działania danej funkcjonalności. Ułatwieniem pracy testera manualnego są testy automatyczne- polegające w dużej mierze na zautomatyzowaniu czynności wykonywanych przez osobę przeprowadzającą testy manualne, jednak głównym celem testowania automatycznego jest uzupełnienie procesu testowego.

Automatyzacja testów i framework Selenium

Automatyzacja testów może znacznie ułatwić cały proces testowania i wytwarzania oprogramowania. Nie możemy jednak przyjąć, że rozwieje ona wszystkie problemy wynikające w procesie manualnego testowania. Doskonale jednak uzupełnia ona cały proces testowy i ułatwia pracę testerów. Jednym z zasadniczych celów automatyzacji jest również ograniczenie nakładów finansowych na realizację powtarzalnych i stałych testów. Należy jednak pamiętać o obowiązkowym koszcie wdrożenia testów automatycznych do naszego projektu. Często proces automatyzacji wymaga przynajmniej jednej osoby, której głównym zadaniem będzie rozwijanie, pielęgnowanie i utrzymywanie skryptów testujących i ewentualnie całego środowiska testowego - jeśli takowe jest potrzebne. Automatyzacja tego procesu nie oznacza, że czynności w nim wykonywane będą bezobsługowo.
Jakie części projektu informatycznego warto automatyzować? Przede wszystkim zależy to od specyfikacji i przeznaczenia projektu. Przed wdrożeniem automatyzacji, powinniśmy dobrze zastanowić się do jakich części projektu możemy jej użyć. Dobrze napisany skrypt testu, powinien znakomicie sobie poradzić z powtarzalnymi czynnościami, które sprawdzają tę samą logikę systemu. Testowanie tego typu idealnie sprawdza się również przy testach wydajnościowych - przykładowo możemy zaimplementować skrypt, który zostanie wykonany przez określoną liczbę użytkowników(utworzonych w kodzie), nawet w tym samym czasie. Najczęściej testy automatyczne wykorzystywane są przy testowaniu interfejsów użytkownika. Szczególną popularnością cieszą się automatyczne test UI, najczęściej testujące warstwy interfejsów stron internetowych. Takie testy sprowadzają się często do “przeklikania” danej funkcjonalności przy użyciu aplikacji internetowej i weryfikacji rezultatu całego procesu.
Kiedy mamy już wybrane moduły naszej aplikacji, które chcemy zweryfikować za pomocą testów automatycznych - kolejnym krokiem będzie wybranie odpowiednich narzędzi do implementacji i przeprowadzenia naszych testów.
Jednym z najbardziej znanych i powszechnie używanych frameworków do testów automatycznych stron internetowych jest narzędzie Selenium WebDriver. Jest to zestaw narzędzi umożliwiających automatyzację przeglądarek pod wieloma aspektami. Selenium obsługuje automatyzację najbardziej popularnych przeglądarek takich jak: Google Chrome, Firefox, Safari, Opera i Internet Explorer. Framework pracuje głównie z Selenium WebDriver API oraz z danymi udostępnionymi przez użytkownika w przeglądarce. Korzystając z tego narzędzia, możemy przetestować aplikacje webowe dla komputerów stacjonarnych, przeglądarek mobilnych a także środowisk rozproszonych. W zależności od specyfikacji naszego projektu - możemy dowolnie dopasowywać funkcjonalności Selenium. Dodatkowo, skrypty używające tego narzędzia możemy implementować w 4 językach programowania: C#, Java, Ruby i Python. Integracja funkcji selenium z naszymi skryptami, czy z całym projektem jest banalnie prosta. Przykładowo, w przypadku C#, używając Visual Studio, konieczne jest pobranie odpowiedniej paczki NuGet, zawierającej biblioteki Selenium. Przy implementacji testów automatycznych, potrzebujemy tylko odwołać się do konkretnych bibliotek. Jedyne co następnie musimy zrobić to wybrać przeglądarki, na jakich chcemy

wykonywać nasz kod, oraz zintegrować je z projektem. Możemy to bezproblemowo zrobić za pomocą dedykowanych sterowników dostępnych w internecie, np. ChromeDriver.
Samo działanie Selenium w połączeniu ze skryptami testowymi, polega na przetwarzaniu elementów html dostępnych w przeglądarce. Aplikacje webowe są pisane najczęściej przy użyciu templatów HTML oraz stylów CSS. Selenium "pobiera" wskazane elementy jako WebElement - typ obsługiwany przez framework, który później możemy wykorzystywać (przykładowo jako zmienną) w naszych skryptach testowych.

Projekt własnego środowiska testowego

Po wyborze narzędzia odpowiedniego do implementacji skryptów testowych musimy zastanowić się nad stworzeniem i dopasowaniem środowiska testowego, w którym dany kod testowy będzie egzekwowany. Cała architektura takiego środowiska, ponieważ są to testy UI - powinna operować i przetwarzać tylko wizualne elementy projektu. Nie powinna pracować nad elementami z baz danych lub elementów zwracanych przez serwisy czy moduły backendowe implementowane w projekcie.
Istotnym elementem środowiska testowego są scenariusze testowe - wejściowa część projektu, gdzie definiujemy nazwę testu - według wcześniej ustalonej konwencji, jego kryteria akceptowalności, kroki testowe oraz implementacyjne. Tę część środowiska, możemy rozumieć jako plany testowe - według których przeprowadzana będzie implementacja skryptu i testowanie funkcjonalności.
Serwer środowiska testowego - wykonawcza część projektu, oddzielny serwer w wytwarzanym projekcie. Serwer Selenium jest modułem zintegrowanym z wykonywanymi skryptami - jest miejscem, gdzie inicjalizujemy przeglądarkę i wykonujemy na niej przypadki testowe. W przypadku Selenium, mamy możliwość skorzystania z lokalnej inicjalizacji przeglądarki, jednak używanie serwera daje lepsze odwzorowanie środowiska wyjściowego(klienckiego), do którego ma docelowo trafić nasza aplikacja. Nie zakłóca to też pracy testera, skrypty działają w tle - na oddzielnym środowisku.
Moduł skryptowy - aplikacja testerska. Część projektu przeznaczona do implementacji kodu testowego, opartego o scenariusze testowe. Struktura tego modułu jest ściśle powiązana ze strukturą serwisu webowego, który mamy zamiar przetestować. Każda strona html w projekcie, jest reprezentowana przez osobną część programu - przykładowo, może to być klasa, lub repozytorium. W takiej przykładowej klasie implementujemy również wszystkie elementy danej strony, po czym odwołujemy się do nich z poziomu funkcji, zaimplementowanej w zależności od tego, jakie funkcjonalności będzie sprawdzał nasz test. Wyniki uzyskane w module skryptowym - przekazujemy do modułu wyjściowego.
Moduł wyjściowy - jak sama nazwa wskazuje jest to element końcowy działania środowiska testowego. W głównej mierze jest to reprezentacja rezultatów uzyskanych przez egzekwowanie modułu skryptowego na serwerze środowiska. Przyjmuje on wartości zwracane przez programy testowe, a następnie przetwarza je na bardziej przyjazny dla developera bądź testera widok. Poprzez odpowiednią prezentację wyników testowych, możemy w łatwy sposób zweryfikować, czy przykładowo na pewno dane wejściowe do testów zostały zapewnione. Przede wszystkim ten moduł informować nas będzie o powodzeniu bądź niepowodzeniu testu, jednak jego możliwości możemy dostosować pod nasze specyfikacje i wymagania, w zależności od tego, co chcemy uzyskać do późniejszego przetwarzania rezultatów testowych.
Moduł uzupełniania danych - przy opisie tego elementu środowiska, musimy mieć na uwadze, iż nie we wszystkich projektach implementacja tego modułu będzie opłacalna, kolosalnie trudna do wykonania, lub nawet niemożliwa do zrealizowania. Przy ogromnych

projektach komercyjnych - bo w głównej mierze w takich projektach ten moduł będzie sprawdzał się najlepiej, musimy zapewnić ogrom wejściowych konfiguracji i danych, przed egzekwowaniem skryptu testowego, czy nawet przed wykonaniem testu manualnego. Głównym celem tego modułu jest działanie pomiędzy środowiskiem testów automatycznych a środowiskiem implementacyjnym projektu. Jak sama nazwa wskazuje funkcjonalności tego elementu skupiają się wokół zapewniania danych do przeprowadzenia testów. Cała struktura elementu przyjmuje dane zwracane również przez moduł skryptowy, które następnie przetwarza i na tej podstawie zapewnia dane wejściowe do rozpoczęcia procesu testowania. Proces zapewniania danych będzie najprawdopodobniej odnosił się od najniższej warstwy projektu - bazy danych, aż po interfejsy widzialne przez użytkownika. Choć oczywiście, będzie to zależało od struktury oprogramowania projektu. Bardzo dobrym rozwiązaniem jest zintegrowanie powyższego modułu z modułem wyjściowym, przynajmniej pod kątem wspólnego interfejsu i kontrolowanego za jego pomocą przekazywania danych.

Podsumowanie

Cały projekt środowiska został oparty o własne spostrzeżenia i wnioski z powierzonych mi zadań jako programiście/testerowi automatycznemu. Automatyzacja testów nie należy do najłatwiejszych zagadnień, jednak pozwala ona na zaoszczędzenie dużej ilości czasu, kosztów i przede wszystkim mozolnej pracy “klikacza” manualnego.

Literatura:

1. Selenium i testowanie aplikacji. Receptury, Wydanie II, aut: Unmesh Gundecha
2. Testowanie oprogramowania. Podręcznik dla początkujących, aut: Rober Pawlak