Testowanie automatyczne I - Wprowadzenie

Wykład 2

Testowanie automatyczne Aplikacji Web

- Aplikacje web z powodu swojej architektury mają wiele potencjalnych sposobów automatyzacji które można wykorzystać do uzyskania pożądanego pokrycia testami.
- Dwa główne elementy poddawane testom to:
  - UI / Frontend
  - API / Endpoint

WEB UI/Frontend Tests

- Testowanie przez warstwę frontendową aplikacji web jest jednym z najpopularniejszych sposobów testowania funkcyjnego aplikacji
- Jego zalety to:
  - Przeglądarka/applikacja odpowiada za wszystkie nie istotne z naszego punktu widzenia elementy takie jak ciasteczka, cash, rendering
  - Symulujemy prawdziwe interakcje użytkownika z aplikacją
  - Łatwo zobrazować proces testowania

WEB UI/Frontend Tests

- Jego wady to:
  - Delikatność na zmiany w warstwie które bardzo łatwo zrobić zmianę niezauważalną dla użytkownika
  - Kosztowna czasowo często obciążona oczekiwaniami na elementy które powinny się pojawić
  - Zależnie od projektu UX brak możliwości chodzenia na „skróty”

WEB UI/Frontend Tests

- Testy frontowe również mogą testować nie funkcjonalnie
  - Ale z powodu ich prędkości raczej będą to testy niemierzące wydajność a bardziej niezawodność, kompatybilność z wersją przeglądarki czy też rzeczy specyficzne dla rynku

API / Endpoint Tests

- Testowanie przez warstwę API czy endpointy
- Jego zalety to:
  - Szybkość – nie czekamy na renderingi aplikacji
  - Prostota projektowania – dosłownie jak wyślę A to otrzymam B
  - Mierzalność – Czasy, wielkości przesyłanych pakietów, ilość wywołań / przekierowań

API / Endpoint Tests

- Jego wady to:
  - Skomplikowanie samych zapytań – by stworzyć poprawną treść zapytania potrzebujemy często wielu elementów z innych zapytań które skądś trzeba otrzymać
  - Zarządzanie elementami i wyłuskiwanie danych z ciasteczek itp.
  - API bez wersjonowania potrafią przedstawiać wykolejające zmiany.

API / Endpoint Tests

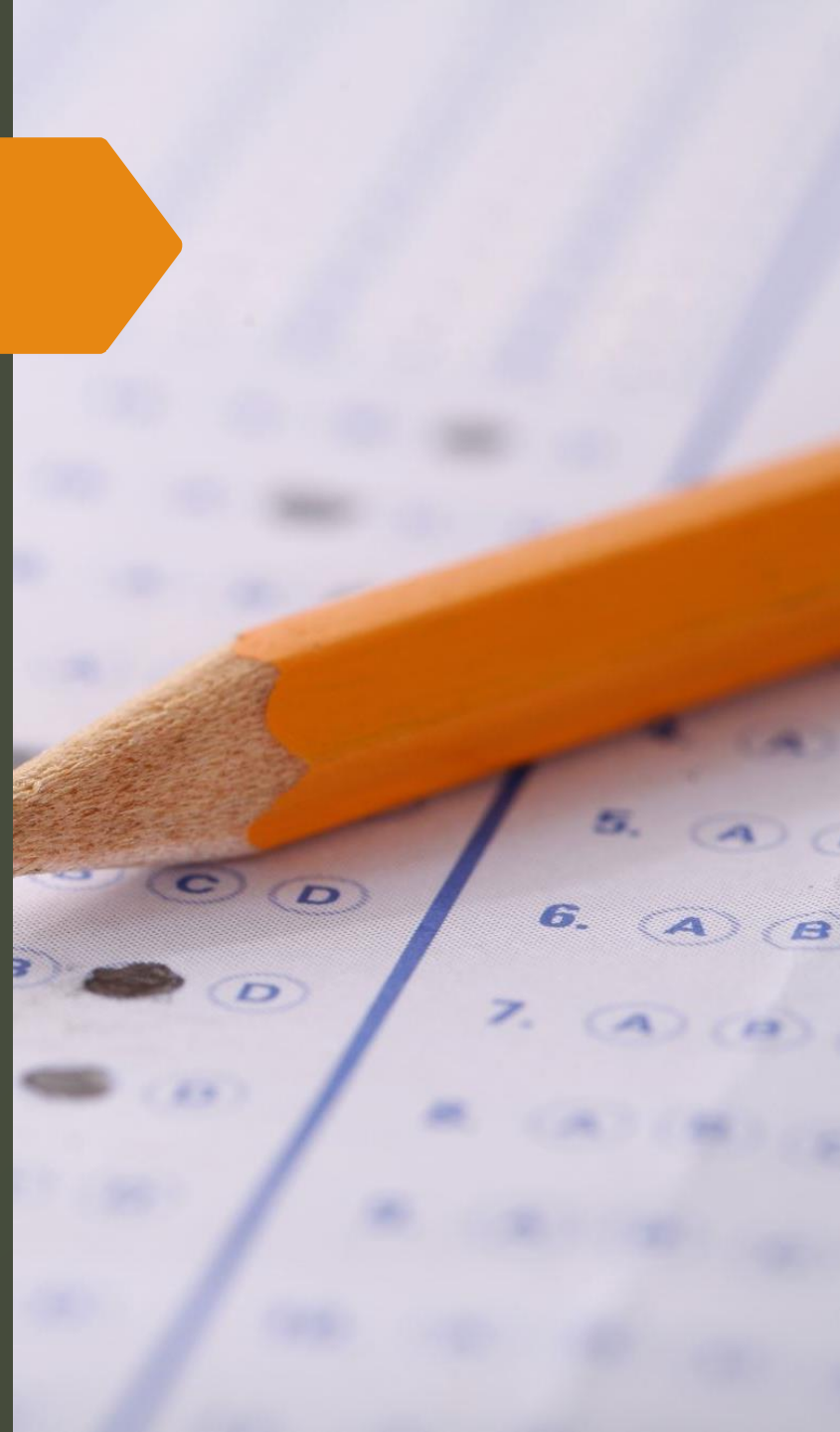
- Testy tego łatwo się skalują i zazwyczaj są szybkie zatem idealnie nadają się do testów nie funkcjonalnych takich jak load, performance, skalowania
- Testy tego typu często są wykorzystywane do potwierdzania sprawności API partnerów przed innymi testami

Popularne Narzędzia

- UI testy przeglądarkowe
  - Selenium
    - OpenSource
    - Dostarcza trzy projekty zależnie od potrzeb organizacji
      - Selenium WebDrive
        - Wykorzystuje WebDriver czyli zestawy instrukcji do kontrolowania przeglądarki
      - Selenium IDE
        - Rozszerzenie do przeglądarek pozwalające na egzekucje skryptów testowych
      - Selenium Grid
        - Rozszerza WebDriver by dostarczyć skalowalność

Popularne Narzędzia

- UI testy przeglądarkowe
  - Selenium
  - Cypress
    - Dostarcza dwa rozwiązania
      - Cypress Test Runner
        - OpenSource
        - Oparty na JavaScript
      - Cypress Dashboard
        - Rozwiązanie Chmurowe nastawione na skalowanie i współpracę z CI/CD



Popularne Narzędzia

- API
  - Postman
    - Dostarcza szereg rozwiązań do tworzenia, projektowania i testowania api
    - Wspiera wiele formatów takich jak REST, SOAP, JSON, cURL
    - Pozwala na łatwe skalowanie testów i optymalizacje procesów CI/CD

Popularne Narzędzia

- API
  - Postman
  - SoapUI
    - Dostarcza szereg narzędzi
      - SoapUI jest OpenSource
        - Umożliwia bez skryptowe pisanie skryptów testowych
      - ReadyAPI
        - Płatne narzędzie nastawione współpracę, integracje z innym narzędziami i CI/CD

Popularne Narzędzia

- Wyspecjalizowane narzędzia API
  - Apache JMeter™
    - Narzędzie OpenSource nastawione na funkcjonalne load testy i mierzenie wydajności
    - Jak na razie nieśmiertelny dziadek testowania aplikacji webowych
    - Dostarcza narzędzie interfejsowe umożliwiające „wyklikanie” testu oraz bibliotekę dla piszących w Java

Popularne Narzędzia

- Wyspecjalizowane narzędzia API
  - Apache JMeter™
  - K6
    - Dostarcza dwa narzędzia
      - K6 Open Source lokalne ale darmowe
      - Cloud płatne SaaS
    - Narzędzie nastawione na load testing
    - Wykorzystuje JavaScript do pisania swoich testów
    - Młodziak który próbuje ubić dziadzia

Selektory CSS

- Odzwierciedlają użycia konstrukcji CSS na stronie
- Podstawową zaletą jest ich prostota
- Podstawową wadą jest nieskończenie wielka elastyczność CSS
  - Przykładowo:
    ```
    div.layout-align-start-center.layout-row.flex > button.md-raised.md-primary.buttonEdit.md-button.md-configurationTool-theme.md-ink-ripple
    ```
  - Co to oznacza?

Selektory CSS klocki

- Podstawowe Selektory
  - gwiazdka oznacza wszystko
    - *
  - Nazwa Elementu wybierze wszystkie elementy o tej nazwie
    - p
  - Kropka z nazwą klasy wybierze wszystkie elementy które mają daną klasę
    - .class

Selektory CSS klocki cz. 2

- Hasztag nazwa identyfikatora = wszystkie elementy z danym identyfikatorem
  - #id
- Nazwa w nawiasach kwadratowych = wszystkie elementy z danym atrybutem
  - [href]
  - Może być rozszerzony o wartość atrybutu
    - [href="https://wi.pb.edu.pl"

Selektory CSS klocki cz. 3

- Selektory Atrybutowe mogą być elastycznie pisane
  - ="" wartość atrybutu musi mieć identyczną wartość
  - *="" wartość atrybutu musi mieć zawierać wartość
    - div[class*="-"]
      - Wybierze elementy zawierające klasy z nazwą zawierającą –
        - Zatem zostaną wybrane np.:
          - <div class="style-scope" \>
          - <div class="style-transition" \>

Selektory CSS klocki cz. 4

- ^="" wartość atrybutu musi mieć wartość z przodu
  - div[class^="scope-"]
    - Zatem zostaną wybrane np.:
      - <div class="scope-transition" \>
    - Ale nie
      - <div class="style-scope" \>

Selektory CSS klocki cz. 5

- $="" wartość atrybutu musi mieć wartość z końca
  - div[class$="scope"]
    - Zatem zostaną wybrane np.:
      - <div class="style-scope" \>
    - Ale nie
      - <div class="scope-transition" \>

Selektory CSS klocki cz. 6

- ~="" wartość atrybutu musi zawierać słowo o wartości
  - div[class~="scope"]
    - Zatem zostaną wybrane np.:
      - <div class="scope" \>
      - <div class="scope transition" \>
    - Zatem nie zostaną wybrane np.:
      - <div class="style-scope" \>
      - <div class="scope-transition" \>

Selektory CSS klocki cz. 7

- |= "" wartość atrybutu musi zawierać dokładnie wskazaną wartość lub wartość zaczynającą się
  - div[class|="scope"]
    - Zatem zostaną wybrane np.:
      - <div class="scope" \>
      - <div class="scope-transition" \>
    - Zatem nie zostaną wybrane np.:
      - <div class="style-scope" \>
      - <div class="scope transition" \>

Selektory CSS klocki cz. 8

- Scalanie
  - .rodzic .dziecko
    - Wybierze wszystkie elementy z klasą dziecko które mają dowolnego przodka z klasą .rodzic
  - .rodzic > .dziecko
    - Wybierze wszystkie elementy z klasą dziecko które mają bezpośredniego przodka z klasą .rodzic

Selektory CSS klocki cz. 9

- Scalanie
  - .dziecko + .rodzeństwo
    - Wybierze element z klasą rodzeństwo które przylegają do klasy dziecko
  - .dziecko ~ .rodzeństwo
    - Wybierze wszystkie element z klasą rodzeństwo które przylegają do klasy dziecko
  - Jeżeli nie umieścimy spacji między selektorami zakładamy że element musi spełnić wszystkie warunki
    - img.kon#a[title=landscape]