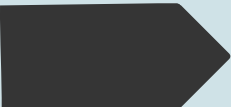




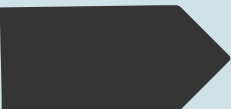
Testowanie automatyczne I - Wprowadzenie

Wykład 3



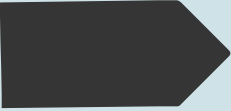
Efektywne Testy Automatyczne

- Mówiliśmy o wyzwaniach testowania ale jakie są częste błędy w automatyzacji i jak ich unikać?
 - Przenoszenie scenariuszy oraz mentalności testów Eksploracyjnych do Automatyzacji.
 - Zagrożenie: Testy tracą czytelność odpowiedzialności za testowaną przestrzeń i walidują oraz weryfikują wiele rzeczy w jednym teście.
 - Jak mówiliśmy kilkakrotnie testy automatyczne powinny być Jednoznaczne i zrozumiałe.
 - Zatem metodą uniknięcia tego problemu jest:
 - Jeden czytelny cel testu
 - Prostota (Keep it Simple)



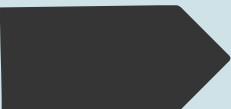
Efektywne Testy Automatyczne

- Uzależnianie testów od siebie nawzajem lub od kolejności
 - Zagrożenie: Testy zależne od siebie utrudniają lub uniemożliwiają niezależne i równoległe testowanie co wiąże się wydłużeniem się czasowym testowania i dodaje nowe problemy w rozpoznawaniu wadliwych testów.
 - Zatem metodą uniknięcia tego problemu jest:
 - Niezależne przypadki testowe
 - Analiza warunków wejścia
 - Projektowanie testów by można było je wykonywać równolegle



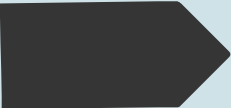
Efektywne Testy Automatyczne

- Duże repozytorium testów Zombie
 - Zagrożenie: Nieznane testy wykonujące się poprawnie albo z błędami od zawsze które chyba testują coś.
 - Zatem metodą uniknięcia tego problemu jest:
 - Prowadzenie dokumentacji testowej
 - Kontrola jakości testów
 - Stosowanie czytelnych technik opisywania testów (Keyword, BDT)




Efektywne Testy Automatyczne

- Celowanie w 100% pokrycia lub Automatyzacja na siłę
 - Zagrożenie: Tworzymy testy trudne do utrzymania i zwielokrotniamy ich ilość w repozytorium wydłużamy czas testowania i implementacją testami które nie przyniosą nam realnego zysku a wręcz zaczną nam generować większe koszt
 - Zatem metodą uniknięcia tego problemu jest:
 - Projektowanie testów automatycznych by były budowane z re-używalnych elementów
 - Określanie Wartości testu przed jego automatyzacją
 - Nakładanie Ograniczeń
 - Prostota (Keep it Simple)



Efektywne Testy Automatyczne

- Brak metryk lub historii
 - Zagrożenie: Brak prowadzenia historii i metryk uniemożliwia wykrywania: niestabilności testów, wykrywania różnic między testami i odpowiedzi na pytanie kiedy błąd się pojawił
 - Zatem metodą uniknięcia tego problemu jest:
 - Jest Prowadzenie historii
 - Najprostsze to zapisywanie raportów generowane przez testy
 - Bardziej zaawansowane jest używanie narzędzi CI\CD
 - Jenkin
 - Azure Devops
 - Są też dedykowane narzędzia chmurowe



Efektywne Testy Automatyczne

- Tylko tyle?
 - Oczywiście że nie
 - Przed wami wyzwania i ryzyka: analityczne, technologiczne, logiczne, optymalizacyjne, programistyczne
 - W których rozwiązaniu pomogą wam:
 - Wzorce projektowe
 - Standardy
 - Sami producenci narzędzi
 - Doświadczenie
 - Dobre praktyki
 - Fora

Selektory Xpath

- Czym jest XPATH
 - XML Path Language
 - Standard stworzony przez W3C dla XML
 - Skoro dla XML to czemu działa na stronie?

Selektory Xpath

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

```
<h1>Demo of the selectors</h1>
<div class="style-scope">
  <p> p in style-scope</p>
</div>
<div class="style-transition">
  <p> p in style-transition</p>
</div>
<div class="scope-transition">
  <p> p in scope-transition</p>
</div>
<div class="scope">
  <p> p in style</p>
</div>
<div class="style">
  <p> p in style</p>
</div>
<div class="scope transition">
  <p> p in scope transition</p>
</div>
<div class="rodzic">
  <div class="dziecko">
    .dziecko in .rodzic
  </div>
</div>
<div class="rodzic">
  <div class="problem">
    <div class="dziecko">
```



Selektory Xpath

- XPATH jest zbudowany z
 - XPath
 - Reprezentacja ścieżki
 - XQuery
 - Język zapytań i funkcji przeszukiwania reprezentacji dokumentu



Selektory Xpath

- Podstawy ścieżki
 - // - Oznacza dowolną pozycję
 - / - Oznacza bezpośrednią pozycję lub początek bezpośredniej ścieżki
 - * - Oznacza dowolny element
 - Zatem
 - //* - zwróci nam wszystkie elementy
 - /* - zwróci nam pierwszy element



Selektory Xpath

- Podstawy ścieżki
 - By wybrać dowolny element p
 - `//p`
 - By wybrać specyficzny p element o konkretnej ścieżce
 - `/html/body/div/p`



Selektory Xpath

- ▶ Pierwsze rozszerzenie
 - ▶ [] – zawiera xquery i funkcje
 - ▶ "//img[@src]" – pobierze wszystkie elementy z atrybutem src
- ▶ Xpath w porównaniu do CSS nie rozróżnia typów atrybutów zatem do wszystkich atrybutów elementu odwołujemy się przez @



Selectory XPath

- Porównanie sposobów odwoływania się
 - CSS:
 - #id
 - Xpath
 - `//*[@id='id']`
 - CSS:
 - .class
 - Xpath
 - `//*[@class='class']`



Selectory XPath

- Porównanie sposobów odwoływania się
 - CSS:
 - `div[class*="-"]`
 - Xpath
 - `//div[contains(@class,'-')]`
 - CSS:
 - `div[class^="scope-"]`
 - Xpath
 - `//div[starts-with(@class,'scope-')]`



Selectory XPath

- Porównanie sposobów odwoływania się
 - CSS
 - `div[class$="scope"]`
 - XPath
 - `//div[ends-with(@class,'scope')]` - Jeżeli oprogramowanie wspiera XPath w wersji 2.0 lub wyższej
 - `//div[substring(@class, string-length(@class) - string-length('scope')+1)='scope']`



Selectory XPath

- Porównanie sposobów odwoływania się
 - CSS:
 - `div[class~="scope"]`
 - XPath
 - `//div[contains(concat(' ', @class, ' '), ' scope ')]`



Selectory XPath

➤ Scalanie

➤ Wewnątrz warunku

- `"//img[contains(@src,'.jpg') and contains(@title,'flower')]"`
- `"//img[contains(@src,'.jpg') or contains(@title,'flower')]"`

➤ Zewnątrz warunku

- `//img[contains(@src,'.jpg')][contains(@title,'flower')]` – odpowiednik `and`
 - Uwaga kolejność ma znaczenie



Selectory XPath

➤ Scalanie

➤ Rodzeństwo

- `//div/following-sibling::img[1]`
 - Pierwszy sibling odpowiednik div + img z css
- `//div/following-sibling::img`
 - Dowolny sibling img od div odpowiedni div ~ img
- `//img/preceding-sibling::div`
 - Dowolny sibling div poprzedzający img
- `//img/preceding-sibling::div[1]`
 - Pierwszy sibling div poprzedzający img



Selectory XPath

➤ Scalanie

➤ Rodzeństwo

- `//img/preceding-sibling::div[@class='scope']`

- Dowolny sibling div z class scope poprzedzający img

➤ Rodzice

- `//p/..`

- Dowolny Bezpośredni Rodzic elementów p

- `//p/parent::div[@class='scope']`

- Dowolny Bezpośredni Rodzic typu div z class = scope

- `//div[child::p[text()=' p in style']]`

- Element div który ma dziecko z określonym tekstem