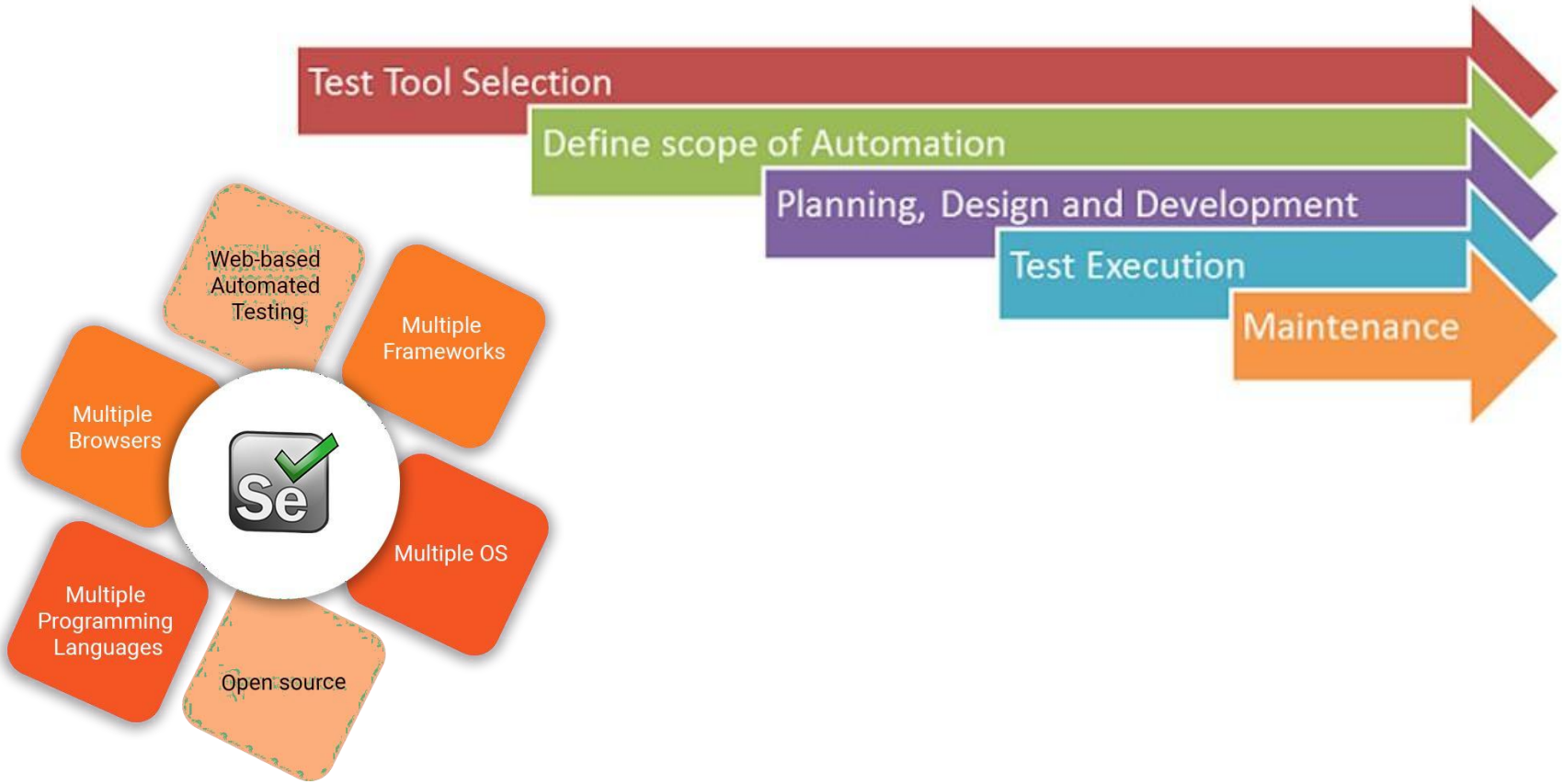
TESTIRANJE SOFTVERA (13S113TS)
LABORATORIJSKA VEŽBA:
AUTOMATSKO TESTIRANJE - SELENIUM WEB DRIVER I TEST NG

Prof. dr Dražen Drašković
Mast. inž. Nikola Stanković

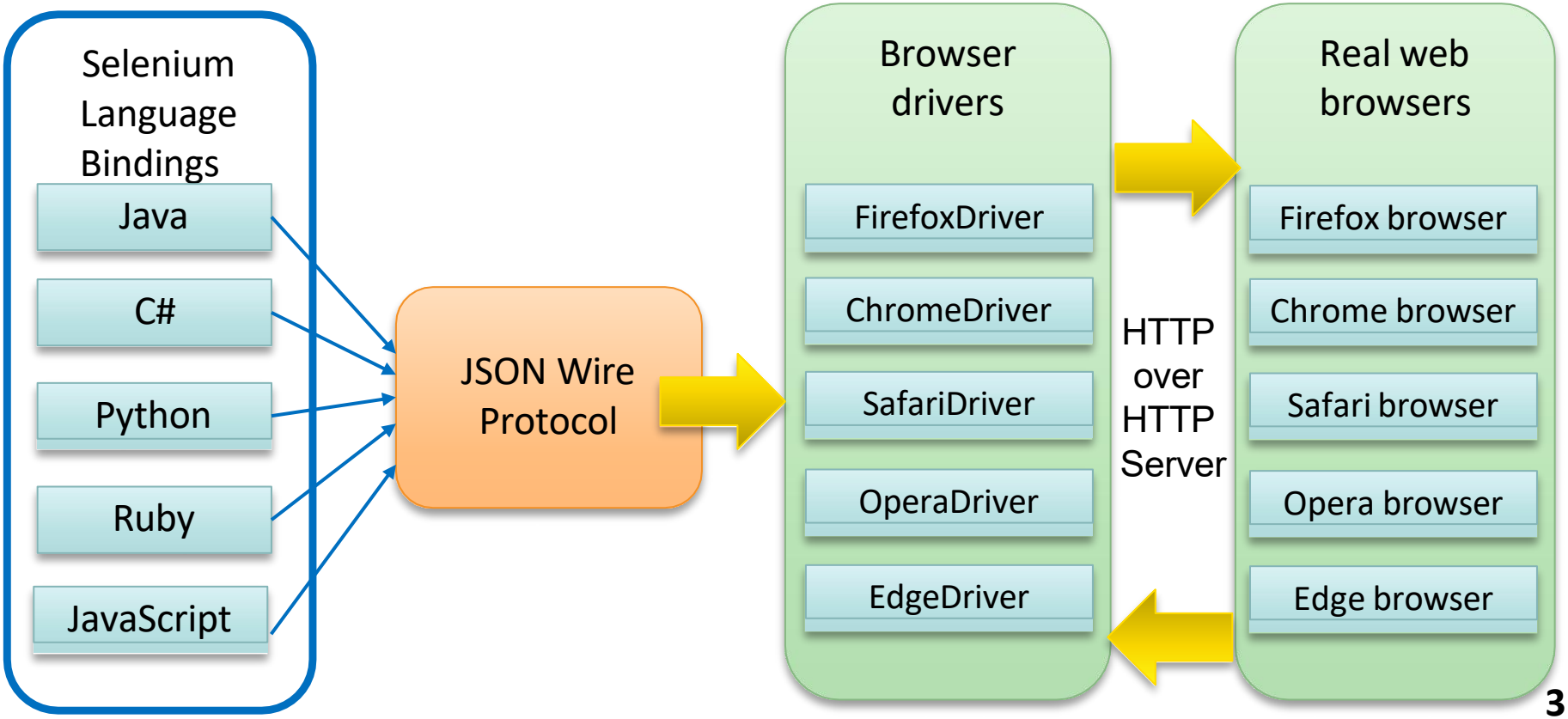
Univerzitet u Beogradu - Elektrotehnički fakultet

UVOD U AUTOMATSKO TESTIRANJE



ARHITEKTURA VEB DRAJVER ALATA

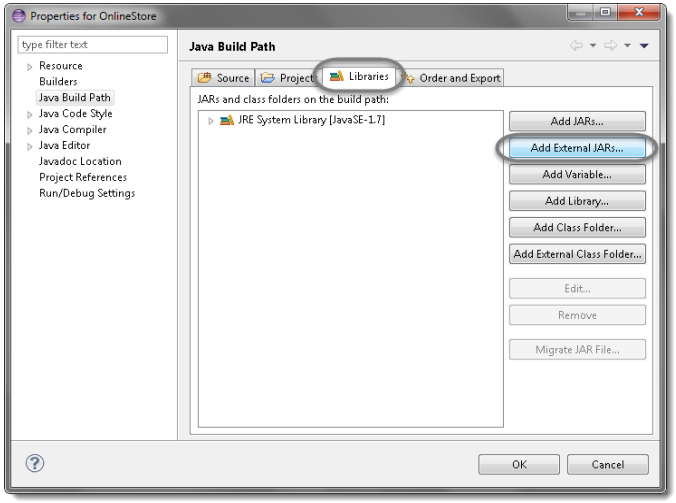
- *Selenium Web Driver* – objektno orijentisani aplikativni interfejs (API).
- Različite implementacije za različite veb pregledače.
- Može se raditi testiranje baze podataka, praviti izveštaji o rezultatima testiranja, upravljati greškama i izuzecima.
- Trenutna aktuelna verzija *Selenium Web Driver*-a je 4.x.

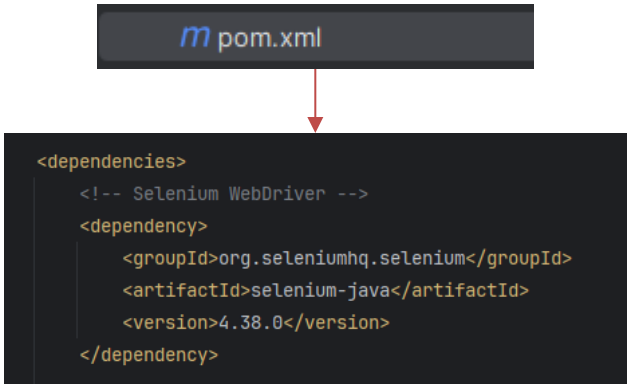


PODEŠAVANJA U RAZVOJNIM ALATIMA

- Dokumentacija: ***https://www.selenium.dev/documentation/webdriver/***
- Dva načina za uvoz biblioteke u projekat:

1. Preuzeti željeni drajver (JAR) sa sajta: ***https://www.selenium.dev/ecosystem/*** ***i*** potom uključiti biblioteku u svoj projekat (u nekom okruženju: *Netbeans*, *Eclipse*, IntelliJ i sl.).
2. Koristiti Maven alat u projektu i postaviti zavisnosti u ***pom.xml*** fajlu. WebDriver u Maven repozitorijumu: https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java





PRIMER: PODEŠAVANJE JAVA PROJEKTA (1)

- Uključivanje drajvera za *Firefox* veb pregledač:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class TestCase {
        private WebDriver driver;
        public static void main(String[] args) {
             driver = new FirefoxDriver();
        }
}
```

- Uključivanje drajvera za *Chrome* veb pregledač:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public static void main(String[] args) {
          String exePath = "C:\\Users\\Drazen\\Desktop\\chromedriver.exe";
          System.setProperty("webdriver.chrome.driver", exePath);
          WebDriver driver = new ChromeDriver();
          driver.get("https://demoqa.com/automation-practice-form");
}
```

PRIMER: PODEŠAVANJE JAVA PROJEKTA (2)

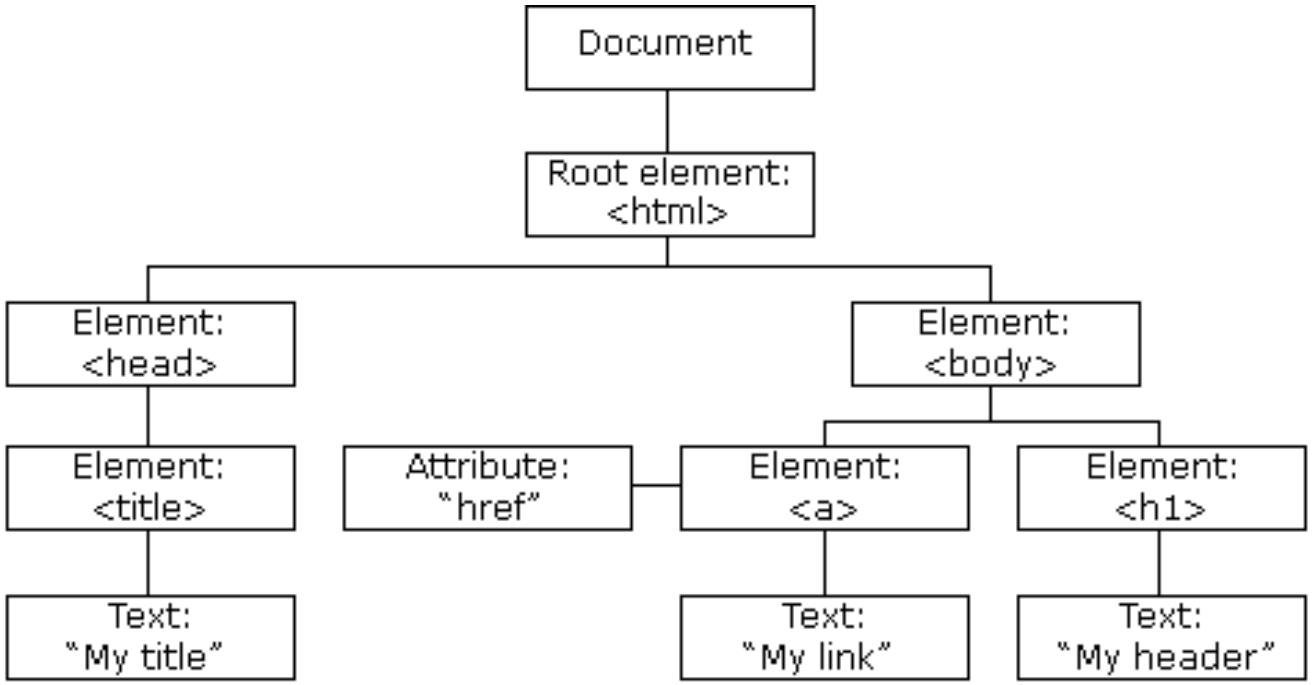
- Drajver koji je preuzet mora biti kompatibilan sa verzijom pretraživača koja je instalirana na računaru na kom se pokreće projekat.
- Postoji open-source biblioteka ***WebDriverManager*** za upravljanje drajverima koja automatski prepoznaje verziju pretraživača i preuzima odgovarajući drajver.
- Maven link: https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager
- Izvorni kod biblioteke: https://github.com/bonigarcia/webdrivermanager
- Potrebno je u *pom.xml* dodati zavisnost i onda je rad sa drajverima mnogo jednostavniji:

```
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class TestCase {
        private WebDriver driver;
        public static void main(String[] args) {
              WebDriverManager.chromedriver().setup();
             driver = new ChromeDriver();
        }
}
```

```
<!-- WebDriverManager -->
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>6.3.3</version>
</dependency>
```

HTML DOM

- Pri učitavanju veb stranice, veb pregledač kreira njen *Document Object Model* (DOM).
- DOM povezuje veb stranice sa skriptovima ili programskim jezicima, tako što reprezentuje veb stranu u memoriji.
- DOM je objektni model za HTML, XHTML i XML fajlove.



HTML DOM API

- HTML DOM definiše:
  - HTML elemente kao objekte
  - HTML element svojstva (*properties*)
  - HTML element metode
  - HTML element događaje (*events*)
- HTML DOM API omogućava pristup različitim funkcijama veb pregledača kao što su prozori, kartice (*tabs*), CSS stilovi, istorijat pregledanja, može da menja HTML elemente, atribute i reaguje na HTML događaje.
- Obično se HTML DOM odnosi na JavaScript, iako modelovanje HTML, SVG ili XML dokumenata kao objekata nije deo osnovnog JavaScript standarda.

LOCIRANJE VEB ELEMENATA

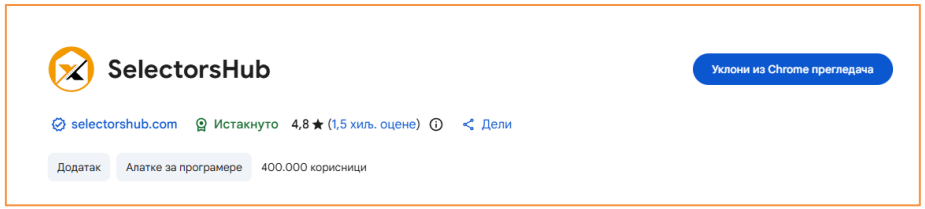
- Načini lociranja elemenata: inspekcija, *XPath* i *CSS.*
- *Selenium* lokatori: *ID*, *Class name*, *Name*, *Link text*, *XPath* i *CSS.*
- *XPath* – jezik koji opisuje način lociranja i obradu elemenata, koristeći sintaksu zasnovanu na putanji kroz logičku strukturu dokumenta ili hijerarhiju.
- *XPath* sintaksa:

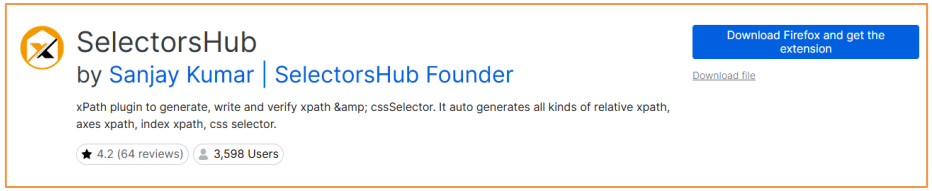
  //tagname[@attribute='value']

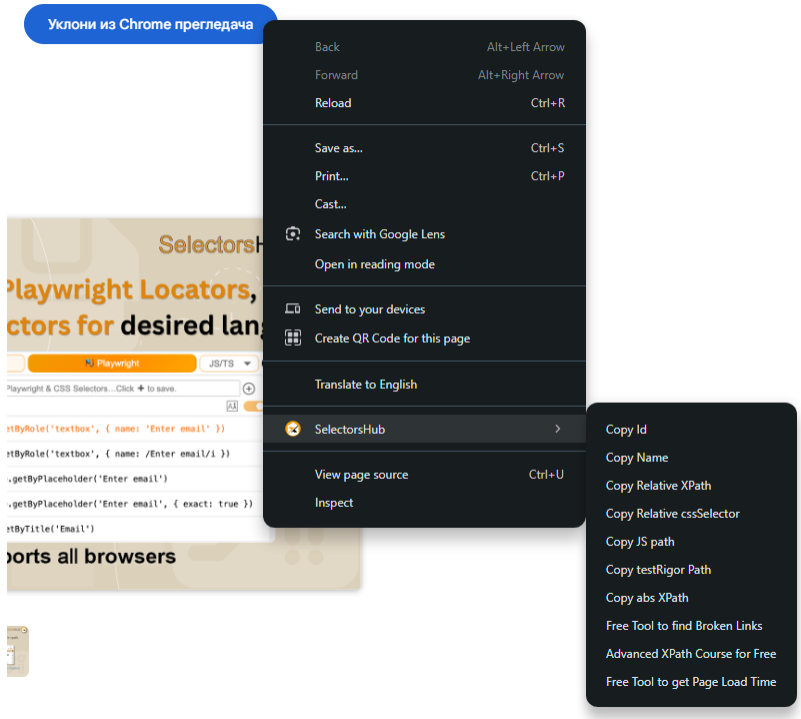
  //input[@type='text']
- Šta znači?
  - // : odaberi tekući čvor
  - tagname : ime određenog čvora/HTML taga (div, a, p,...)
  - @ : izabrati atribut
  - attribute : ime atributa čvora
  - value : vrednost atributa

LOCIRANJE VEB ELEMENATA - DODACI ZA VEB PREGLEDAČE

- *Za Chrome i Firefox*: *SelectorsHub*







XPATH I SELENIUM

- Tipovi *XPath* (*XML path*):
  - apsolutni: iz korenog čvora (root-node)

    html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b

  - relativni: iz bilo kog dela stranice (npr. može početi sa sredine)

    //*[@class='featured-box’]
    //*[text()='Testing’]

- Prednost: ako niste sigurni kako se zove neki element, možete koristiti „contains“ da nađete sva moguća podudaranja.

CSS I SELENIUM

- HTML primer:

- Lokator:
  div[class='ajax_enabled'] [style='display:block’]
- Razlike između korišćenja CSS lokatora i Xpath lokatora?
  - CSS prolazi nadole kroz DOM, a XPath omogućava prolazak i nagore i nadole.
  - CSS pouzdaniji i lakši za korišćenje, često i brži od XPath, ali to može da zavisi od veb pregledača!
  - XPath pokretači u različitim veb pregledačima imaju neke razlike, pa su nekonzistentni!
  - Trend kod XPath je da postane kompleksniji (samim tim teži za čitanje).

UPOREDNI PREGLED SINTAKSE KOD XPATH I CSS LOKATORA

Uslov	CSS selektor	Xpath
All elements	*	//*
All <p> elements	p	//p
All children elements	p>*	//p/*
Select by ID	#start	//*[@id='start']
Select by class	.start	//*[contains(@class, 'start')]
Select by attribute	*[title]	//*[@title]
First child of all <p>	p>*:first-child	//p/*[0]
All <p> elements with a child <a>	impossible to find	//p[a]
Next element	p + *	//p/following-sibling::*[0]
Previous element	impossible to find	//p/preceding-sibling::*[0]

Savet:
- CSS sa klasama, ID, imenima tag-ova; za jednostavne upite zasnovane na atributima elementa; kada su nam potrebne informacije koje nemamo u DOM (pseudo-selektori poput a:visited, input:focus,...).
- XPath koristiti kada tražite roditeljski element ili tražite element po tekstu!

LOKATORI I PRIMER LOGIN FORME

- HTML forma za prijavu korisnika:

  Username: <input id="username" type="text" name="korisnicko_ime" />

  Password: <input id="password" type="password" name= "lozinka" />

  Link: Naziv linka

1. *ID* lokator:

```
WebElement elementPassword = driver.findElement(By.id("password"));
```

2. *Name* lokator:

```
WebElement elementPassword = driver.findElement(By.name("lozinka"));
```

3. Linkovan tekst:

```
WebElement element=driver.findElement(By.linkText("Naziv linka"));
```

4. *Xpath*:

```
//input[@id='username']
WebElement element=driver.findElement(By.xpath("//input[@id='username']"));
input[name='password']
WebElement element=driver.findElement(By.cssSelector("input[name='password']"));
```

LOCIRANJE – NEKE METODE

- click() – pritisnut element korisničkog interfejsa
- clear() – čišćenje polja za unos podataka
- sendKeys(CharSequence tekst) – slanje (unošenje) karaktera
- getText() – vraća tekst elementa korisničkog interfejsa
- isEnabled() – proverava da li je element dostupan
- isSelected() – proverava da li je element selektovan

PROBLEMI U LOCIRANJU – SELENIUM WAIT KOMANDE

- Izbegavati korišćenje *Thread.sleep()*
- Implicitni *wait* (određeno trajanje) – globalno podešavanje za sve elemente koje definiše koliko *Web Driver* čeka da uspešno locira element pre nego što obori test zbog nemogućnosti lociranja (podrazumevano 0):

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

- Eksplicitni *wait* (do ispunjenja nekih uslova) – podešavanje na nivou jednog elementa koje definiše maksimalno vreme čekanja da neki uslov bude ispunjen (npr. element može da se klikne) pre nego što se obori test:

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement element =
wait.until(ExpectedConditions.elementToBeClickable(By.id(<someid>)));
```

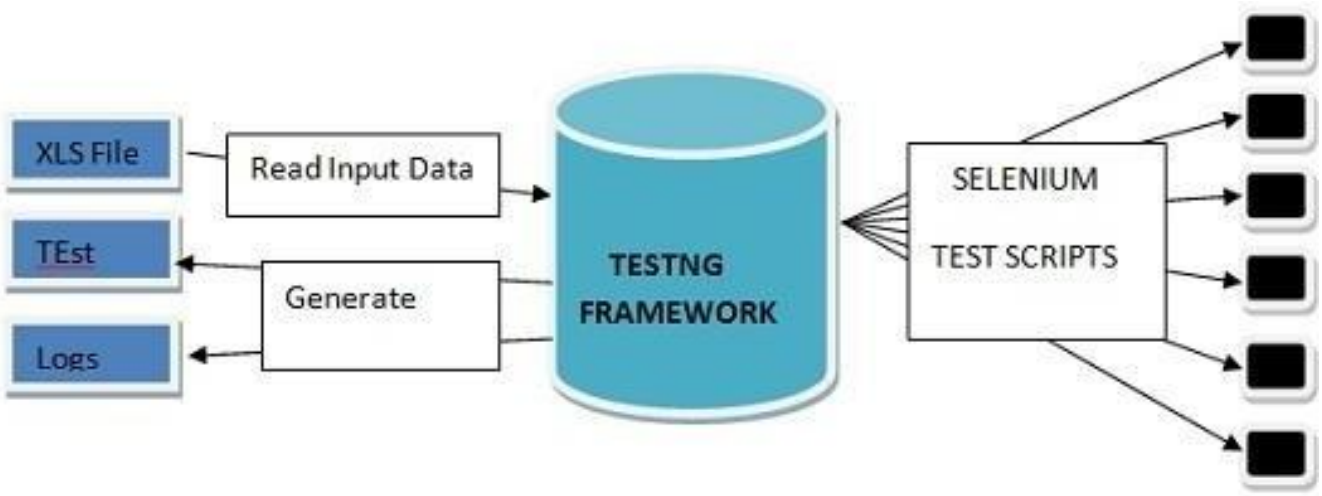
PROBLEMI U LOCIRANJU – SELENIUM WAIT KOMANDE (2)

- Protočni (*fluent*) *wait* (čeka do maksimalno definisanog vremena, ali proverava u određenim intervalima uz mogućnost izuzetka):

```
Wait wait = new FluentWait(driver).withTimeout(30,
SECONDS).pollingEvery(5,
SECONDS).ignoring(NoSuchElementException.class);

WebElement naziv = wait.until(new Function() {
     public WebElement apply(WebDriver driver) {
          return driver.findElement(By.id("naziv"));
     }
});
```

TESTNI OKVIR – TEST NG

- Test NG (*Test Next Generation*) - testni radni okvir za automatsko testiranje koji koristi anotacije.
- Anotacije su linije koda koje kontrolišu kako i kada će metode da budu izvršene.
- Značaj *TestNG* u kombinaciji sa alatom *Selenium WebDriver* je generisanje testnih izveštaja, da li su testovi prošli uspešno (PASSED), nisu prošli (FAILED) ili su preskočeni (SKIPPED).

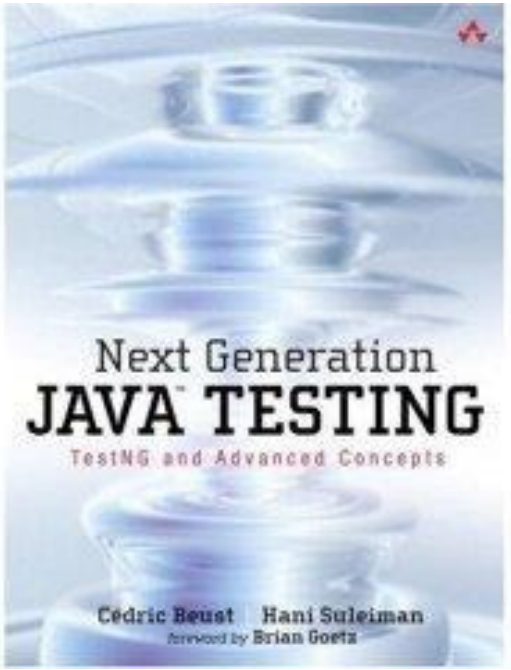


TEST NG – ZNAČAJ

- Napravljen je sa ciljem da olakša izvršavanje Java automatizovanih testova.
- Nastao 2004. godine kao pandan JUnit alatu (inspirisan nedostacima).
- Primenljiv na svim vrstama testova (jediničnim, „end to end“, testovima GUI, itd.).
- Poslednja stabilna verzija: 7.11.0
- Link za dokumentaciju: https://testng.org/doc/

```
<!-- TestNG -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.11.0</version>
    <scope>test</scope>
</dependency>
```

Uvoz u projekat



SELENIUM + TEST NG

Selenium WebDriver	TestNG framework
Web Driver je automatski radni okvir koji koristi JUnit alat (testiranje kod bele kutije).	Radni okvir TestNG koristi anotacije šta se izvršava pre svakog test primera, posle svakog test primera, itd.
Web Driver nema prirodni mehanizam za generisanje izveštaja.	Testni izveštaji se mogu generisati pomoću TestNG okvira.
Za pokretanje neuspešnih test primera potrebno je ponovo da pokrenemo taj skript koji sadrži taj test primer.	Neuspešni test primeri se mogu nezavisno pokretati korišćenjem TestNG alata (bez WebDriver-a).

OSNOVNE ANOTACIJE KOD TEST NG

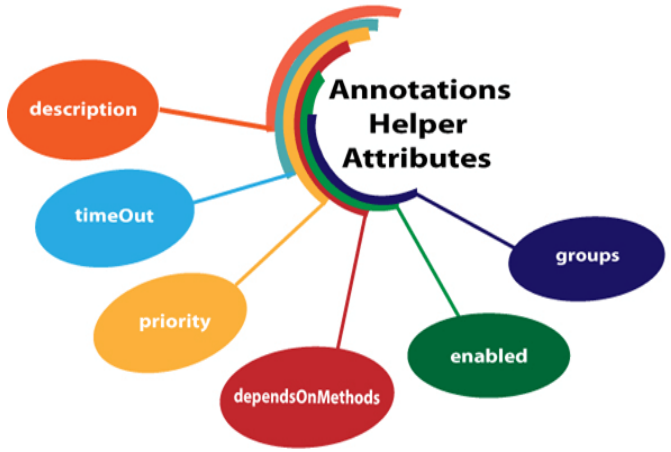
Anotacija	Značenje
@Test	Označava klasu ili metodu koja je deo test primera.
@BeforeTest	Anotirana metoda koja će biti pokrenuta pre svake testne metode (kao preduslov testa), koja pripada <test> tagu.
@AfterTest	Anotirana metoda koja će biti pokrenuta posle svih test metoda (kao postuslov testa), koji su unutar <test> taga.
@BeforeClass	Anotirana metoda će biti pokrenuta jednom pre prve test metode u tekućoj klasi.
@AfterClass	Anotirana metoda će biti pokrenuta jednom posle svih test metoda u tekućoj klasi koja je pokrenuta.
@BeforeGroups	Lista grupa koju će konfigurisani metod pokretati pre. Garantovano će se pokrenuti pre prvog testa u toj grupi.
@AfterGroups	Lista grupa koje će konfigurisani metod pokrenuti posle.
@BeforeSuite	Pokretanje metode jednom pre svih testova.
@AfterSuite	Pokretanje metode jednom posle svih testova.
@Listeners	Definiše osluškivače u testnoj klasi.
@Parameters	Opisuje koji parametri i kako će biti predati testnoj metodi.

Svaka anotacija može imati određeni broj definisanih atributa.

ATRIBUTI UZ ANOTACIJU @TEST

- @Test(atribut=vrednost)
- description - opis testa (podrazumevano: prazan String)
- dependsOnMethods - zavisnost od nekih drugih metoda (očekuje nazive metoda)
- alwaysRun - testna metoda će se uvek izvršiti (podrazumevano: *False*)
- enabled - metoda treba ili ne treba da se izvršava (podrazumevano: *False*)
- parallel - izvršavanje testova u paraleli (podrazumevano: *False*)
- priority - prioritet test primera (podrazumevano: 0)
  Obuhvata ceo broj između -5000 i 5000.
  Najniži prioritet se prvo izvršava, pa naviše.
  Npr. redosled: -5000, -20, 0, 15, 60, 5000.
- groups - pripadnost određenoj funkcionalnosti
- timeOut - vreme izvršavanja

  (ako traje predugo, test treba da padne!)



UPOREĐIVANJE SA ASSERT

- Svaki test primer može da bude uspešan (*success*) i neuspešan (*fail*). Test je uspešan ukoliko se uspešno završi, bez bacanja nekog izuzetka, ili baci izuzetak (koji je očekivan).
- Upoređivanje se radi **assert** metodama, koje verifikuju da li očekivani rezultat i aktuelni rezultat se podudaraju.
- Sintaksa za *Hard Assertions* (obara test, čim prvi uslov nije ispunjen):

  Assert.imeMetode (*actual*, *expected*)

  - *actual* = vrednost koja je dobijena
  - *expected* = vrednost koja je očekivana
- Primeri:

  Assert.assertEquals(*objekat1, objekat2, "Opciona poruka"*);
  Assert.assertTrue(*uslov*, "*Opciona poruka*");
- Dosta sličnih/istih *assert* metoda ćemo raditi i kroz *JUnit* alat!

MEKI ASSERT

- Prvo izvrši sve provere, pa tek onda obaraju test, ukoliko je bilo grešaka.
- Primer:

```
SoftAssert sa = new SoftAssert();
sa.assertTrue(uslov1, "Opciona poruka");
sa.assertFalse(uslov2, "Opciona poruka");
```

TEST PAKET (ENG. TEST SUITE)

- Kolekcija test primera definiše paket/skup (*Suite*) koji testira neku funkcionalnost ili više funkcionalnosti nekog softvera.
- Kod *Test NG*: skup se generiše kao XML fajl (***testng.xml*** fajl), koji se sastoji iz test primera. Koreni čvor je <suite> tag, koji ima više <test> sekcija.

```
<suite name="Test Suite">
    <test name="Tests">
        <classes>
            <class name="tests.LoginTests"/>
        </classes>
    </test>
</suite>
```

- Atributi za <suite> su:
  - *name* – naziv skupa, obavezni atribut;
  - *verbose* – nivo pokretanja;
  - *parallel* – pokretanje više niti kojim se pokreće skup;
  - *thread-count* – broj niti koji se koristi, ako je paralelni režim dozvoljen (u suprotnom se ignoriše ovaj atribut);
  - *annotations* – tip anotacija koje koristite u testovima;
  - *time-out* – podrazumevana vrednost pauze koja se koristi kod svih test metoda u okviru tog test skupa.

PARAMETRIZOVANI TESTOVI (1)

- Iz eksternog fajla (*testng.xml*) – primer:

```
//testni fajl
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;
public class ParameterizedTest1 {
  @Test
  @Parameters("myName")
  public void parameterTest(String myName) {
    System.out.println("Parameterized value is : " + myName);
  }
}
```

IZLAZ TESTA:

```
===============================================
Suite1
Total tests run: 3, Failures: 0, Skips: 0
===============================================
```

```
//testni fajl
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name = "Suite1">
   <test name = "test1">
      <parameter name = "myName" value="drazen"/>
      <parameter name = "myName" value="nikola"/>
      <parameter name = "myName" value="dragan"/>
      <classes>
         <class name = "ParameterizedTest1" />
      </classes>
   </test>
</suite>
```

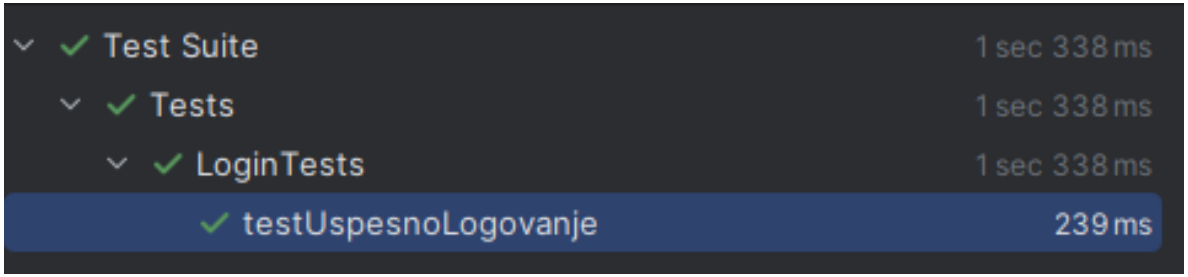
PARAMETRIZOVANI TESTOVI (2)

- Pomoću provajdera podataka:

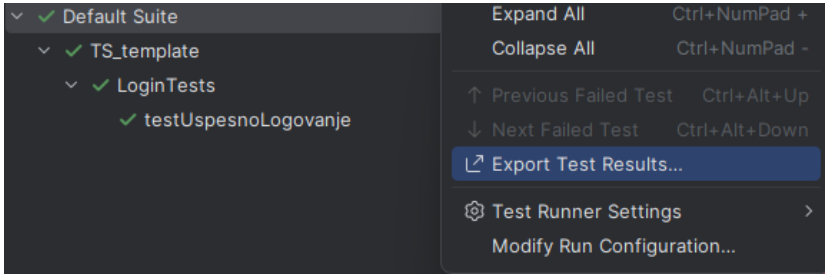
```
import org.testng.Assert;
import org.testng.annotations.*;

public class ParamTestWithDataProvider1 {
  private PrimeNumberChecker primeNumberChecker;

  @BeforeMethod
  public void initialize() { primeNumberChecker = new PrimeNumberChecker(); }

  @DataProvider(name = "test1")
  public static Object[][] primeNumbers() {
    return new Object[][] {{2, true}, {6, false}, {19, true}, {22, false}, {23, true}};
  }

  // This test will run 4 times since we have 5 parameters defined.
  @Test(dataProvider = "test1")
  public void testPrimeNumberChecker(Integer inputNumber, Boolean expectedResult) {
    System.out.println(inputNumber + " " + expectedResult);
    Assert.assertEquals(primeNumberChecker.validate(inputNumber), expectedResult);
  }
```

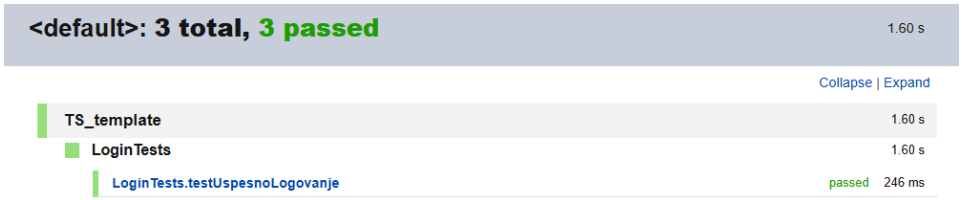
IZVEŠTAJI U TEST NG

- U samom alatu:

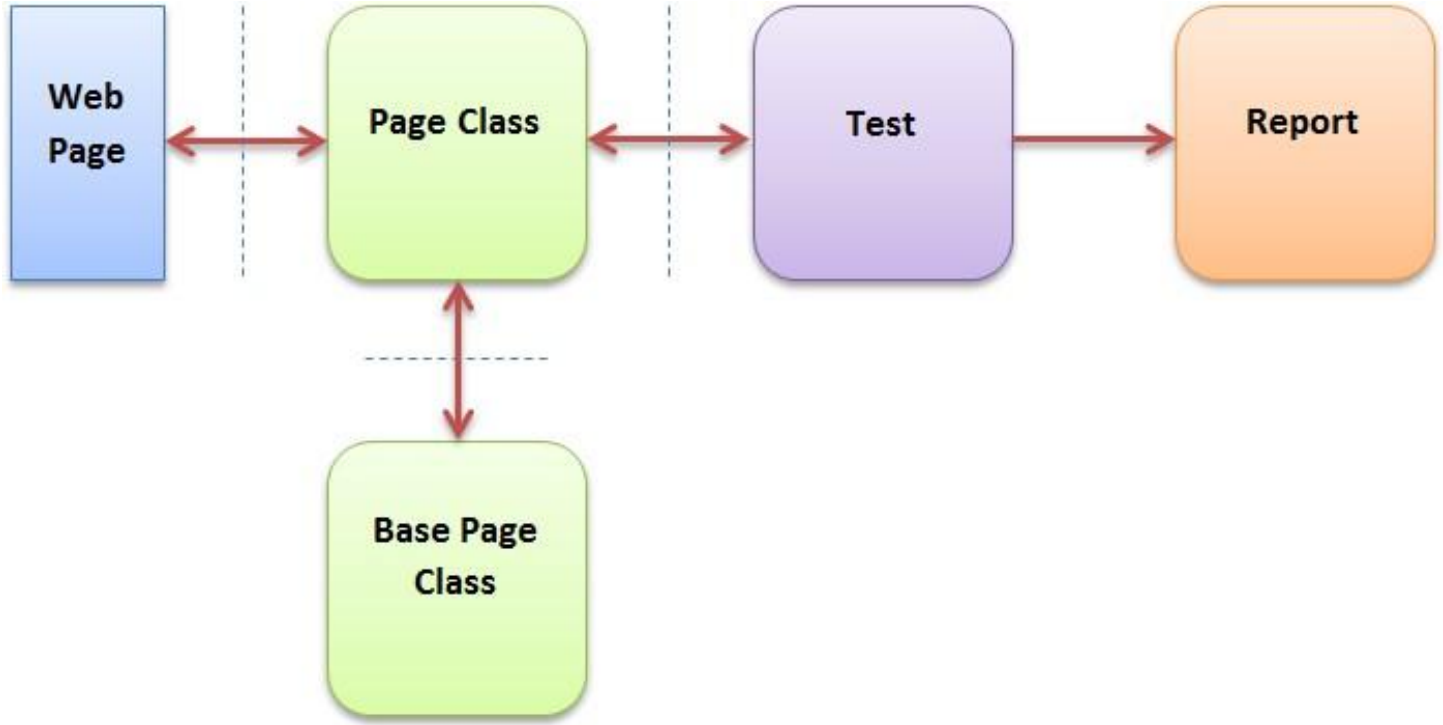


- Ili exportovano kao *HTML/XML* fajl:

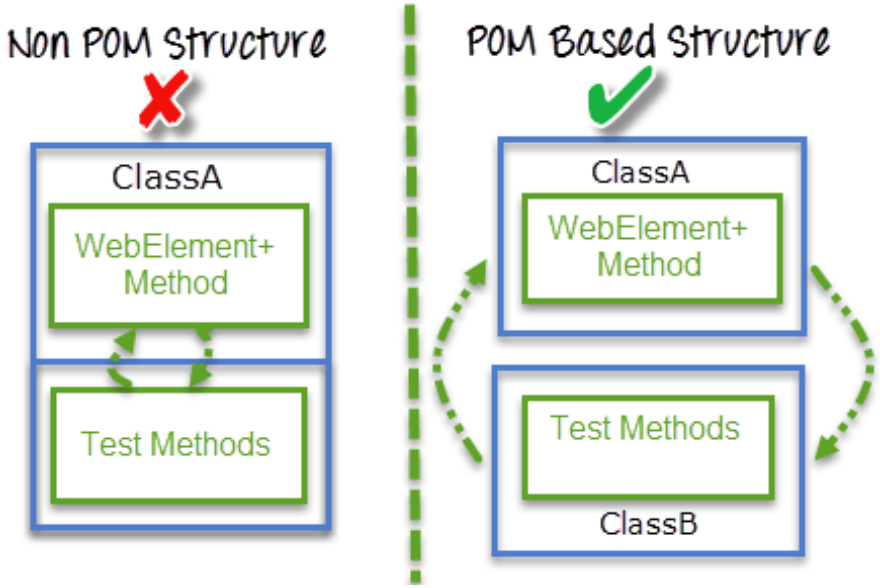




- ***Model objekta stranice (eng. Page Object Model, POM) –*** promeni se veb stranica, ali ne menjamo testove, već samo objekat stranice (*Page Object*).
- Prednost: smanjuje se dupliranje koda i bolje se održavaju testovi!



PREDNOSTI KORIŠĆENJA UZORKA



- Prednosti:
  - *Page object* uzorak – operacije i tokovi u UI treba da budu odvojeni od verifikacije (čistiji i lakši kod za razumevanje).
  - Primena istog repozitorijuma objekata za različite vrste testiranja i različite alate (npr. *POM* u **Selenium + TestNG/JUnit** za funkcionalne testove, i istovremeno sa **JBehave/Cucumber** za testove prihvatljivosti).
  - Programski kod postaje kraći i optimizovaniji, jer postoji reuporebljivost metoda stranice u POM Klasama.
  - Realističnija imena metoda => lako mapiranje sa operacijama u UI.

POM - PRIMER (1)

```
public class SignUpPage extends PageObject {
  @FindBy(id="firstname")
  private WebElement firstName;
  @FindBy(id="lastname")
  private WebElement lastName;
  @FindBy(id="signup")
  private WebElement submitButton;

  public void enterName(String firstName, String lastName) {
    this.firstName.clear(); this.firstName.sendKeys(firstName);
    this.lastName.clear(); this.lastName.sendKeys(lastName);
  }

  public ReceiptPage submit() {
    submitButton.click();
    return new ReceiptPage(driver);
  }
  }
```

POM - PRIMER (2)

- Natklasa *PageObject* gde se nalazi *PageFactory* koja inicijalizuje elemente:

```
public class PageObject {
  protected WebDriver driver;

  public PageObject(WebDriver driver) {
    this.driver = driver;
    PageFactory.initElements(driver, this);
  }
}
```

POM - PRIMER (3)

- Primer još jedne klase izvedene iz *PageObject* klase:

```
public class ReceiptPage extends PageObject {
  @FindBy(tagName = "h1")
  private WebElement header;

  public ReceiptPage(WebDriver driver) {
    super(driver);
  }

  public String confirmationHeader() {
    return header.getText();
  }
}
```

POM - PRIMER (4)

- Test primer:

```
@Test
public void signUp() {
    driver.get("< neki url >");
    SignUpPage signUpPage = new SignUpPage(driver);
    signUpPage.enterName("First", "Last");
    ReceiptPage receiptPage = signUpPage.submit();
    assertEquals("Thank you", receiptPage.confirmationHeader());
}
```

PRAKTIČNI PRIMERI

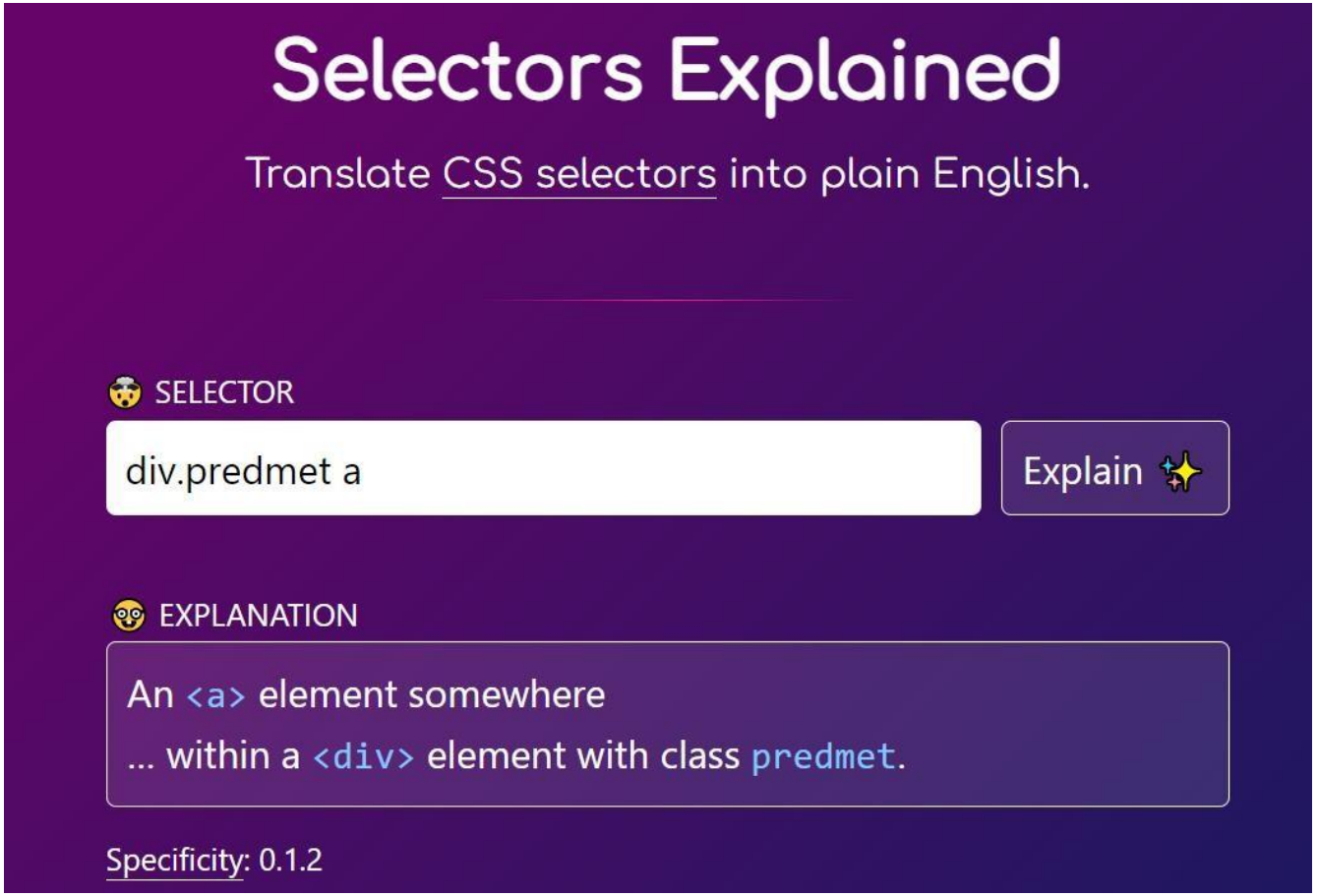
- Primer 1:
  Prijava korisnika (uspešna i neuspešna).
- Primer 2:
  Registracija novog korisnika u sistemu.
- Primer 3:
  Podrška za više veb pregledača (izvršiti iste test primere pokretanjem u veb pregledačima *Chrome*, *Firefox*, *Edge*).

KORISNI LINKOVI (1)

- *Selenium Web Driver* – preuzimanje biblioteka:
  https://www.selenium.dev/downloads/
- *Selenium Web Driver* – dokumentacija:
  https://www.selenium.dev/documentation/
- TEST NG dokumentacija:
  https://testng.org/
- DOM Web API:
  https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- Drajveri:
  - Chrome: chromedriver.chromium.org/downloads
  - Firefox: https://github.com/mozilla/geckodriver/releases

KORISNI LINKOVI (2)

- CSS selektori ...ukoliko ne znate previše dobro CSS ☺
  https://kittygiraudel.github.io/selectors-explained/



HVALA NA PAŽNJI! ☺

PITANJA ?

Kontakt predavača:
drazen.draskovic@etf.bg.ac.rs
nikolas@etf.bg.ac.rs