Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники

**Лабораторная работа №2**

по дисциплине

«Тестирование программного обеспечения»

**Выполнил**

Ореховский А.,

группа P3317

**Преподаватель**

Исаев И. В.

Санкт-Петербург

2020

Оглавление

[Цель 3](#_Toc51878286)

[Описание приложения 3](#_Toc51878287)

[Тестовые сценарии 3](#_Toc51878288)

[Листинг кода 8](#_Toc51878289)

[Компонент Header 8](#_Toc51878290)

[Компонент About 9](#_Toc51878291)

[Компонент Learn 10](#_Toc51878292)

[Выводы 14](#_Toc51878293)

# Цель

Доработать консольное приложение (по вариантам):

1. Прикрутить веб UI;
2. Покрыть UI тестами на Selenium/по согласованию что-то другое.

# Описание приложения

Приложение для «микрообучения». Постройте приложение, высылающее вам ежедневно (или другой интервал) одну страницу по теме, которую вы хотите изучить. Это может быть случайная страница из Википедии, из документации по Kotlin, любое другое.

# Тестовые сценарии

1. Стандартный тестовый случай, который проверяет соответствие заголовка страницы ожидаемому значению.

*Код отвечающий за тестовый случай*

    it('should have a title', () => {

    page.navigateTo();

    expect(browser.getTitle()).toEqual('About');

  });

1. Проверка того, что интервал по умолчанию равен 2 секундам.

*Код отвечающий за тестовый случай*

  it('should have 2 seconds interval by default', () => {

    page.navigateTo();

    var intervalSelect = element(by.id('interval'));

    expect(intervalSelect.getAttribute('value')).toEqual('2');

  });

1. Проверка возможности изменения интервала.

*Код отвечающий за тестовый случай*

  it('should be able to change the select value', () => {

    page.navigateTo();

    var intervalSelect = element(by.id('interval'));

    expect(intervalSelect.getAttribute('value')).toEqual('2');

    element(by.css("#interval [value='60']")).click();

    expect(intervalSelect.getAttribute('value')).toEqual('60');

    element(by.css("#interval [value='3600']")).click();

    expect(intervalSelect.getAttribute('value')).toEqual('3600');

  });

1. Проверка возможности перемещения на страницу обучения при помощи ссылки.

*Код отвечающий за тестовый случай*

  it('should route to the learn page via anchor', () => {

    page.navigateTo();

    expect(browser.getTitle()).toEqual('About');

    element(by.id('learnAnchor')).click();

    expect(browser.getTitle()).toEqual('Learn');

  });

1. Проверка возможности перемещения на страницу обучения при помощи ссылки из навигационной панели.

*Код отвечающий за тестовый случай*

  it('should route to the learn page via nav-link', () => {

    page.navigateTo();

    expect(browser.getTitle()).toEqual('About');

    element(by.id('LearnNavLink')).click()

      .then(() => {

        expect(browser.driver.getTitle()).toEqual('Learn');

      });

  });

1. Проверка того, что поиск топика работает корректно.

*Код отвечающий за тестовый случай*

it('should find a topic', () => {

    page.navigateToLearnPage().then(() => {

      expect(browser.driver.getTitle()).toEqual('Learn');

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(trueTopics[0]);

      expect(topic.getAttribute('value')).toEqual(trueTopics[0]);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementLocated(by.id('topicHeader')));

          var topicHeader = browser.driver.findElement(by.id('topicHeader'));

          expect(topicHeader.getText()).toEqual(trueTopics[0]);

        });

    })

  });

1. Проверка того, что показывается сообщение об ошибке при некорректном топике.

*Код отвечающий за тестовый случай*

  it('should show an error when the topic is wrong', () => {

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(fakeTopic);

      expect(topic.getAttribute('value')).toEqual(fakeTopic);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementLocated(by.id('error')));

          var error = browser.driver.findElement(by.id('error'));

          expect(error.getText()).toEqual('No data found by given topic.');

        });

    });

  });

1. Проверка корректной загрузки страницы из заданного топика.

*Код отвечающий за тестовый случай*

  it('should load a page', () => {

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(trueTopics[0]);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementsLocated(by.css('.btn-primary')));

          var titleButtons = browser.driver.findElements(by.css('.btn-primary'));

          titleButtons.then((titles: any[]) => {

            titles[0].click().then(()=> {

              browser.driver.wait(until.elementLocated(by.id('currentTitle')));

              var currentTitle = browser.driver.findElement(by.id('currentTitle'));

              expect(currentTitle.getText()).toBeDefined();

            });

          });

        });

    });

  });

1. Проверка того, что заголовки страниц по данном топику приходят с интервалом в 2 секунды по умолчанию.

*Код отвечающий за тестовый случай*

it('should receive titles from hub within 2 seconds by default', () => {

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(trueTopics[0]);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementsLocated(by.css('.btn-primary')));

          browser.driver.sleep(6000);

          var titleButtons = browser.driver.findElements(by.css('.btn-primary'));

          titleButtons.then((titles: any[]) => {

            expect(titles.length).toBeGreaterThan(1);

          });

        });

    });

  });

1. Проверка того, что названия страниц могут приходить и с другим интервалом

*Код отвечающий за тестовый случай*

  it('should receive titles from hub with interval different from 2', () => {

    page.navigateTo();

    element(by.css("#interval [value='60']")).click();

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(trueTopics[0]);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementsLocated(by.css('.btn-primary')));

          browser.driver.sleep(6000);

          var titleButtons = browser.driver.findElements(by.css('.btn-primary'));

          titleButtons.then((titles: any[]) => {

            expect(titles.length).toEqual(1);

          });

        });

    });

  });

1. Проверка того, что сообщение об ошибке исчезает, когда вводится корректный топик.

*Код отвечающий за тестовый случай*

it ('should hide an error when new given topic is correct', () => {

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(fakeTopic);

      expect(topic.getAttribute('value')).toEqual(fakeTopic);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementLocated(by.id('error')));

          var error = browser.driver.findElement(by.id('error'));

          expect(error.getText()).toEqual('No data found by given topic.');

        }).then(() => {

          topic.clear();

          topic.sendKeys(trueTopics[0]);

          expect(topic.getAttribute('value')).toEqual(trueTopics[0]);

          searchButton.click()

            .then(() => {

              browser.driver.findElements(by.id('error')).then((items) => {

                expect(items.length).toBe(0);

              });

            });

        });

    });

  });

1. Проверка того, что после ввода нового топика, все старые данные исчезают.

*Код отвечающий за тестовый случай*

  it ('should remove old topic and page when new topic is given', () => {

    page.navigateToLearnPage().then(() => {

      var topic = browser.driver.findElement(by.id('topic'));

      var searchButton = browser.driver.findElement(by.id('searchButton'));

      topic.sendKeys(trueTopics[0]);

      searchButton.click()

        .then(() => {

          browser.driver.wait(until.elementsLocated(by.css('.btn-primary')));

          var titleButtons = browser.driver.findElements(by.css('.btn-primary'));

          titleButtons.then((titles: any[]) => {

            titles[0].click().then(()=> {

              browser.driver.wait(until.elementLocated(by.id('currentTitle')));

              var currentTitle = browser.driver.findElement(by.id('currentTitle'));

              expect(currentTitle.getText()).toBeDefined();

            });

          });

          topic.clear();

          topic.sendKeys(trueTopics[1]);

          expect(topic.getAttribute('value')).toEqual(trueTopics[1]);

          searchButton.click()

            .then(() => {

              browser.driver.wait(until.elementTextContains(browser.driver.findElement(by.id('topicHeader')), trueTopics[1]));

              var topicHeader = browser.driver.findElement(by.id('topicHeader'));

              expect(topicHeader.getText()).toEqual(trueTopics[1]);

              browser.driver.findElements(by.id('currentTitle')).then((items) => {

                expect(items.length).toBe(0);

              });

            });

        });

    });

  });

# Листинг кода

## Компонент Header

*header.component.html*

<header>

    <nav class="navbar navbar-dark navbar-expand-md bg-dark">

        <div class="navbar-brand">LearnApp</div>

        <ul class="navbar-nav mr-auto">

            <li \*ngFor="let item of menuItems">

                <a class="nav-link" routerLink="{{item.route}}" routerLinkActive="active" id="{{item.name}}NavLink">{{item.name}}</a>

            </li>

        </ul>

    </nav>

</header>

*header.component.ts*

import { Component, OnInit } from '@angular/core';

import { MenuItem } from 'src/models/menuItem';

import { Router } from '@angular/router';

@Component({

  selector: 'app-header',

  templateUrl: './header.component.html',

  styleUrls: ['./header.component.css']

})

export class HeaderComponent implements OnInit {

  menuItems: MenuItem[];

  private readonly aboutMenuItem: MenuItem = {

    name: 'About',

    route: '/about'

  }

  private readonly learnMenuItem: MenuItem = {

    name: 'Learn',

    route: '/learn'

  }

  constructor(private router: Router) { }

  ngOnInit(): void {

    this.menuItems = [this.aboutMenuItem, this.learnMenuItem];

  }

}

## Компонент About

*about.component.html*

<div class="container">

    <h1>

        Hello stranger!

    </h1>

    <p>

        This application is developed for those who wants to learn some new cool topics.

    </p>

    <p>

        All you have to do is visit our "Learn" page, search for a topic and then just pick one of the available pages with the given title.

    </p>

    <p>

        You can also choose the interval you want to receive a new page within.

    </p>

    <p>

        Cool! Now you can visit the <a routerLink="/learn" routerLinkActive="active" id="learnAnchor">Learn</a> page!

    </p>

</div>

<div class="container">

    <h2>Select the interval</h2>

    <select name="interval" id="interval" #interval (change)="onIntervalChanged(interval.value)">

        <option value="2" selected>2 seconds</option>

        <option value="60">1 minute</option>

        <option value="3600">1 hour</option>

    </select>

</div>

*about.component.ts*

import { Component, OnInit } from '@angular/core';

import {Title} from '@angular/platform-browser';

@Component({

  selector: 'app-about',

  templateUrl: './about.component.html',

  styleUrls: ['./about.component.css']

})

export class AboutComponent implements OnInit {

  constructor(private titleService:Title) {

    this.titleService.setTitle('About');

  }

  ngOnInit(): void {

    localStorage.setItem('interval', '2');

  }

  onIntervalChanged(interval) {

    localStorage.setItem('interval', interval);

    console.log('The interval has been set to ' + interval + ' seconds.')

  }

}

## Компонент Learn

*learn.component.html*

<div class="search">

    <div class="row">

        <div class="col-4"></div>

        <div class="col-4">

            <div class="row">

                <div class="col-8">

                    <input type="text" name="topic" id="topic" class="form-control" placeholder="Find a topic">

                </div>

                <div class="col-4">

                    <button id="searchButton" class="btn btn-outline-success" (click)="subscribe()" onclick="return false;">Learn!</button>

                </div>

            </div>

        </div>

        <div class="col-4"></div>

    </div>

    <div \*ngIf="error" class="row">

        <div class="col-4"></div>

        <div class="col-4" id ="error" class="alert alert-danger">

            {{error}}

        </div>

        <div class="col-4"></div>

    </div>

</div>

<div>

    <div class="row">

        <div class="col-2" \*ngIf="topic">

            <div class="titles alert ">

                <h3 id="topicHeader">

                    {{topic}}

                </h3>

                <div \*ngFor="let title of allTitles" class="row">

                    <div \*ngIf="isPageAvailable(title); then availablePage else unavailablePage"> </div>

                    <ng-template #availablePage>

                        <button (click)="getPageContent(title)" onclick="return false;" class="btn btn-primary">

                            {{title}}

                        </button>

                    </ng-template>

                    <ng-template #unavailablePage>

                        <button (click)="getPageContent(title)" onclick="return false;" class="btn btn-secondary" disabled>

                            {{title}}

                        </button>

                    </ng-template>

                </div>

            </div>

        </div>

        <div class="col-10"  \*ngIf="pageContent">

            <div class="page-content">

                <h1 class="centered" id="currentTitle">{{currentTitle}}</h1>

                <div id="pageContent" class="row" [innerHTML]="pageContent">

                </div>

            </div>

        </div>

    </div>

</div>

*learn.component.ts*

import { Component, OnInit } from '@angular/core';

import {LearnService} from './learn.service'

import { HttpErrorResponse } from '@angular/common/http';

import {Title} from '@angular/platform-browser';

@Component({

  selector: 'app-learn',

  templateUrl: './learn.component.html',

  styleUrls: ['./learn.component.css'],

  providers: [LearnService]

})

export class LearnComponent implements OnInit {

  public topic: string;

  public allTitles: string[];

  public currentTitle: string;

  public error: string;

  public pageContent: any;

  constructor(private learnService: LearnService, private titleService: Title) {

      this.titleService.setTitle('Learn');

  }

  ngOnInit(): void {

    this.learnService.startConnection();

    this.learnService.addBroadcastPageTitleListner();

    this.pageContent = '';

    this.topic = '';

    this.allTitles = [];

    this.learnService.availablePages = [];

    this.currentTitle = '';

  }

  getPageContent(title:string):void {

    this.currentTitle = title;

    this.learnService.getPageContent(title)

      .subscribe( (content) => {

        this.pageContent = content;

      }, (error: HttpErrorResponse) => {

        console.log(error);

      });

  }

  isPageAvailable(title):boolean {

    return this.learnService.availablePages.includes(title);

  }

  subscribe():void {

    var topic:string = (<HTMLInputElement>document.getElementById('topic')).value;

    this.pageContent = '';

    this.learnService.availablePages = [];

    this.allTitles = [];

    this.error='';

    this.learnService.isTopicExists(topic)

      .subscribe((isTopicExists) => {

        if (isTopicExists) {

          this.learnService.broadcastPageTitle(topic);

          this.topic = topic;

          this.learnService.getPages(topic)

            .subscribe((titles) => {

              this.allTitles = titles;

            });

        }

        else {

          this.error='No data found by given topic.';

          this.pageContent = '';

        }

      }, (error: HttpErrorResponse) => {

        console.log(error);

      });

  }

}

*learn.service.ts*

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import \* as signalR from "@aspnet/signalr";

import { Router, ActivatedRoute } from '@angular/router';

import { Observable } from 'rxjs';

@Injectable()

export class LearnService {

    private baseUrl = 'https://localhost:44395/api/onetime/';

    // Receive messages from server within <interval> seconds.

    private hubConnection: signalR.HubConnection;

    public availablePages: string[];

    constructor (private http:HttpClient) { }

    public startConnection = () => {

        this.hubConnection = new signalR.HubConnectionBuilder()

            .withUrl('https://localhost:44395/broadcast')

            .build();

        this.hubConnection

            .start()

            .then(() => console.log('Connection started'))

            .catch(err => console.log('Error while starting connection: ' + err));

    }

    // Receive from server

    public addBroadcastPageTitleListner = () => {

        this.hubConnection.on('receivePageTitle', (title) => {

            this.availablePages.push(title);

        });

    }

    // Send to serverByTopic

    public broadcastPageTitle = (topic) => {

        var interval:number = parseInt(localStorage.getItem('interval'));

        this.hubConnection.invoke('broadcastPageTitle', topic, interval)

        .catch(err => console.error(err));

    }

    // Get Page Content

    public getPageContent(title):Observable<any> {

        var requestOptions:Object = {

            responseType: 'text'

        }

        return this.http.get<any>(`${this.baseUrl}getPageByTitle?title=${title}&format=html`, requestOptions);

    }

    // Check if topic exists

    public isTopicExists(topic):Observable<boolean> {

        return this.http.get<boolean>(`${this.baseUrl}isTopicExists?topic=${topic}`);

    }

    // Get list of pages by Topic

    public getPages(topic):Observable<string[]> {

        return this.http.get<string[]>(`${this.baseUrl}getPagesByTopic?topic=${topic}`);

    }

}

# Выводы

Клиентское приложение, разработанное в течение данной работы, получилось весьма небольшим, в сравнении даже с предыдущей работой. Тем не менее я изучил основы UI-тестирования angular приложений. Процесс тестирования как модульного, так и пользовательского, отличается от более привычного для меня тестирования ООП. Так как код в клиентском приложении основан на большом количестве promise и observable, необходима подготовка окружения для тестов. Так, например, используется функция beforeEach, которая проделывает одни и те же действия перед каждым тестом.