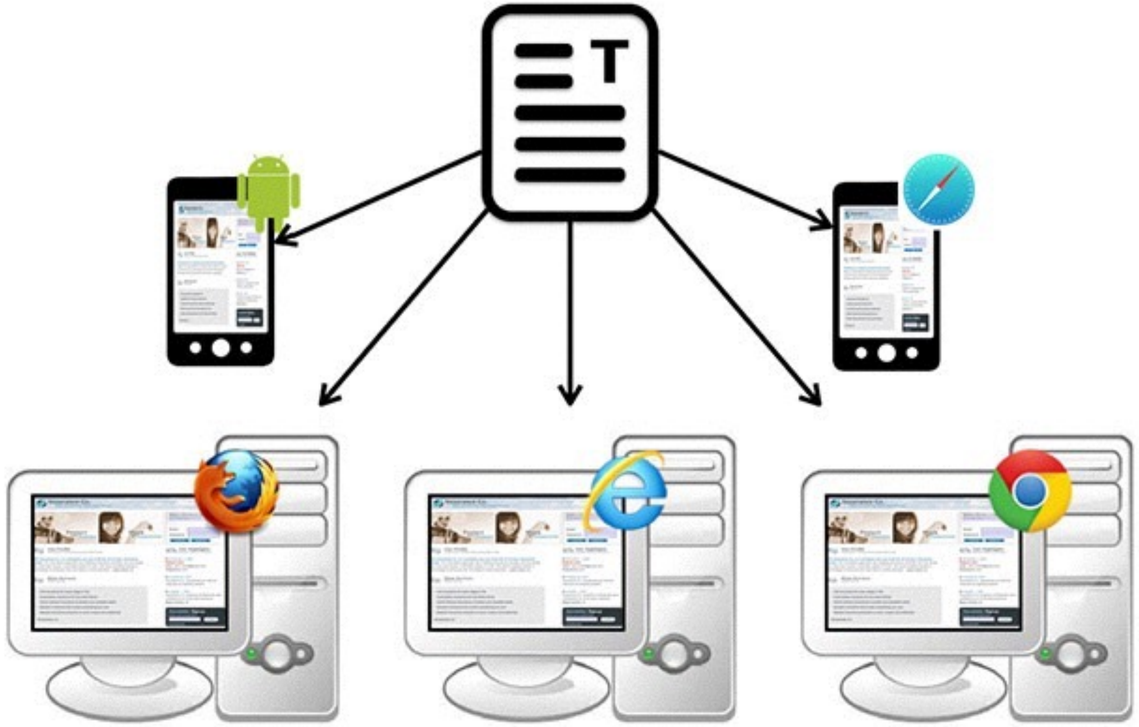


RUTGERS | CODING BOOTCAMP

Session 16.1 - Behavior-driven development (BDD) and Load Testing

Behavior-driven development (BDD) and Load Testing

Objectives

- Students will understand what Behavior Driven Development is
- Students will understand how BDD is similar and differs from TDD.
- Students will run BDD tests on the Eat-Da-Burger app from Week 14
- Students will learn about and run load testing on the Eat-Da-Burger app

What is BDD?

Behavior-driven development (BDD) is a software development methodology in which an application is specified and designed by describing how its behavior should appear to an outside observer.

BDD vs TDD

Within the confines of the requirements and testing dictated by BDD, you will use "ordinary" TDD to develop the software. The unit tests so created will serve as a test suite for your implementing code, while the BDD tests will function more or less as acceptance tests for the customer.

Both TDD and BDD run tests on the code but BDD also tests the output of the complete application to ensure that parts of it make sense to the user.

BDD Testing Tools For Javascript

- **Jasmine** - Behavior-Driven JavaScript
- **Karma** - Spectacular Test Runner for Javascript
- **Selenium** - Web Browser Automation
- **Selenium Webdriver** - makes direct calls to the browser using each browser's native support for automation.

Writing Tests

- Tests are defined in spec files in your application that test functions in your code.
- Tests are in the spec/ folder of your codebase and end in *spec.js*. Only files ending in spec.js will be considered tests.

BDD Tests with Jasmine-Node & Selenium

In order to BDD a node application using BDD. We need to install the following modules.

- Jasmine - http://jasmine.github.io/2.0/node.html
- jasmine-node
- Selenium
- selenium-webdriver - http://webdriver.io/

Example BDD Tests

- Test the homepage of a site exists and is displaying properly
- Verify that the expected elements are properly displaying in the page (for example H2 tg or the page's form)
- Verify database interaction is working
- Verify connect form can be sent

Example Test

```
describe('Eat-da-burger BDD Testing Homepage', function() {

    // Open the Eat-da-burger website in the browser before each test is run
    beforeEach(function(done) {
        this.driver = new selenium.Builder().
            withCapabilities(selenium.Capabilities.chrome()).
            build();

        this.driver.get('http://127.0.0.1:3000/').then(done);
    });

});
```

Define describe block and before each test is run, start a new browser with default route opened.

Example Test

```
describe('Eat-da-burger BDD Testing Homepage', function() {

    // Open the Eat-da-burger website in the browser before each test is run
    beforeEach(function(done) {
        this.driver = new selenium.Builder().
            withCapabilities(selenium.Capabilities.chrome()).
            build();

        this.driver.get('http://127.0.0.1:3000/').then(done);
    });

    // Close the website after each test is run (so that it is opened fresh each time)
    afterEach(function(done) {
        this.driver.quit().then(done);
    });
});
```

When the tests are finished make sure the browser started is closed.

Example Test

```
describe('Eat-da-burger BDD Testing Homepage', function() {

…

    // Test to ensure we are on the home page by checking the <body> tag id attribute
    it('Should be on the home page', function(done) {
        var element = this.driver.findElement(selenium.By.tagName('h2'));

         element.getText().then(function(value) {
            // Show the value of getText
            // console.log(value);
            expect(value).toBe('Welcome To Eat-Da-Burger');
            done();
         });
    });

});
```

Run a test inside the describe block. Look for h2 tag on the page and test to see if it contained the text “Welcome to Eat-Da-Burger”.

Run a More Complex Test

```
describe("Eat-da-burger BDD Testing for Home Page", function () {
  // Open the Eat-da-burger website in the browser before each test is run
  beforeEach(function(done) {
      this.driver = new selenium.Builder().
          withCapabilities(selenium.Capabilities.chrome()).
          build();
      this.driver.get('http://127.0.0.1:3000').then(done);
  });
  // Close the website after each test is run (so that it is opened fresh each time)
  afterEach(function(done) {
      this.driver.quit().then(done);
  });
  it("should create a new burger", function (done) {
    /// Name the new burger and then create it by adding it to the Submit button
    var burgerName="My New Burger, Bro " +(Math.floor(Date.now() / 1000))
    this.driver.findElement(selenium.By.id('enter_text')).sendKeys(burgerName);
    this.driver.findElement(selenium.By.id('text-enter-button')).click();

    var element = this.driver.findElement(selenium.By.id(burgerName));

    element.getAttribute('id').then(function(id) {
      expect(id).toBe(burgerName);
      done();
    });

  });
});
```

Create a new burger with a distant name that will create a div id for that new burger ONLY IF the burger is successfully created,

Exercise

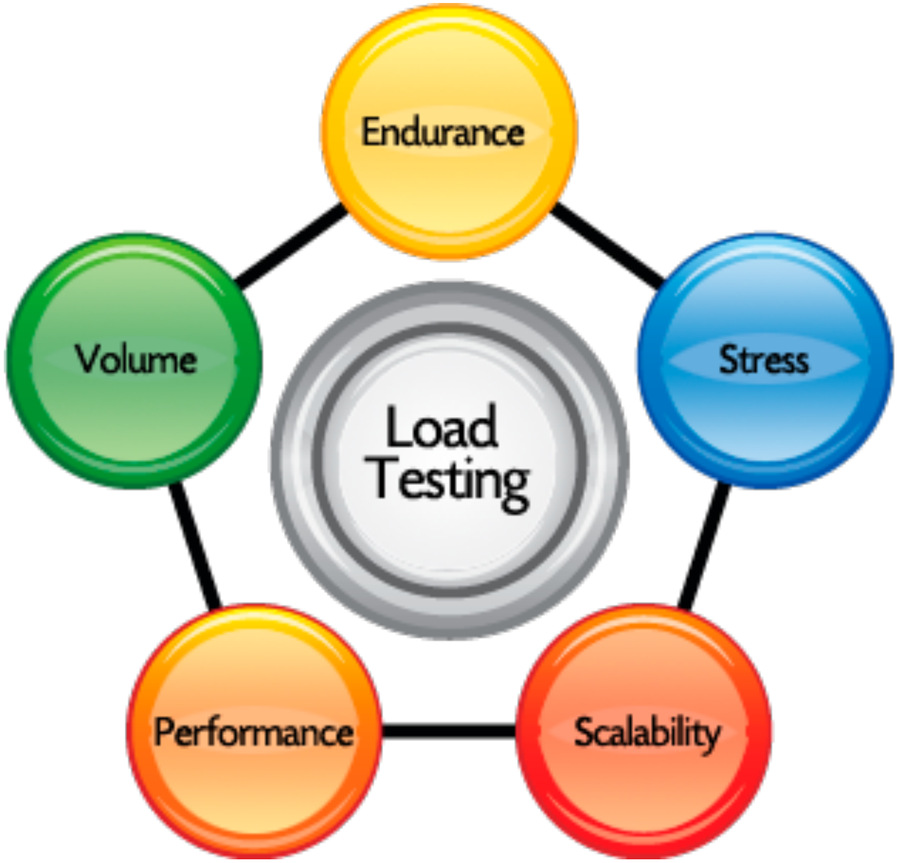
- Write a BDD test for Eat-Da-Burger that
  - “Should fill out form and successfully send email"

Performance Testing

Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device.

Load Testing

Load testing is the process of putting demand on a software system or computing device and measuring its response. Load testing is performed to determine a system's behavior under both normal and anticipated peak load conditions.



Stress Testing

Stress testing is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.

Although Selenium as an automated test tool will run quite fast, making a mini stress test. If you put the same automation running on a couple of computers on your network at the same time you'll be able to see how it behaves.

Load Testing Tools

Server Based Tools

loadtest (NiodeJS module)

Apache JMeter

Gatling

Cloud Based Tools

https://smartbear.com

https://loadimpact.com

https://www.loadsterperformance.com

https://loader.io

Different tools have different strengths and compatibilities

Load Testing Exercise

Using *loadtest* module to find stress points in your Eat-Da-Burger application (or use the MVC Cats application from last week (15.3)).

Instructions:

- Install your app
- Run various tests with loadtest for example:
    - node node_modules/loadtest/bin/loadtest.js -n 2000 -c 100 http://localhost:3000/
- Identify stress points on various routes in the application. Test all paths on the site.
- Why is there stress on those underperforming routes?
- Suggestions for fixing those stress points.
- Fixes for those stress points.