TESTING JAVA CODE EFFECTIVELY

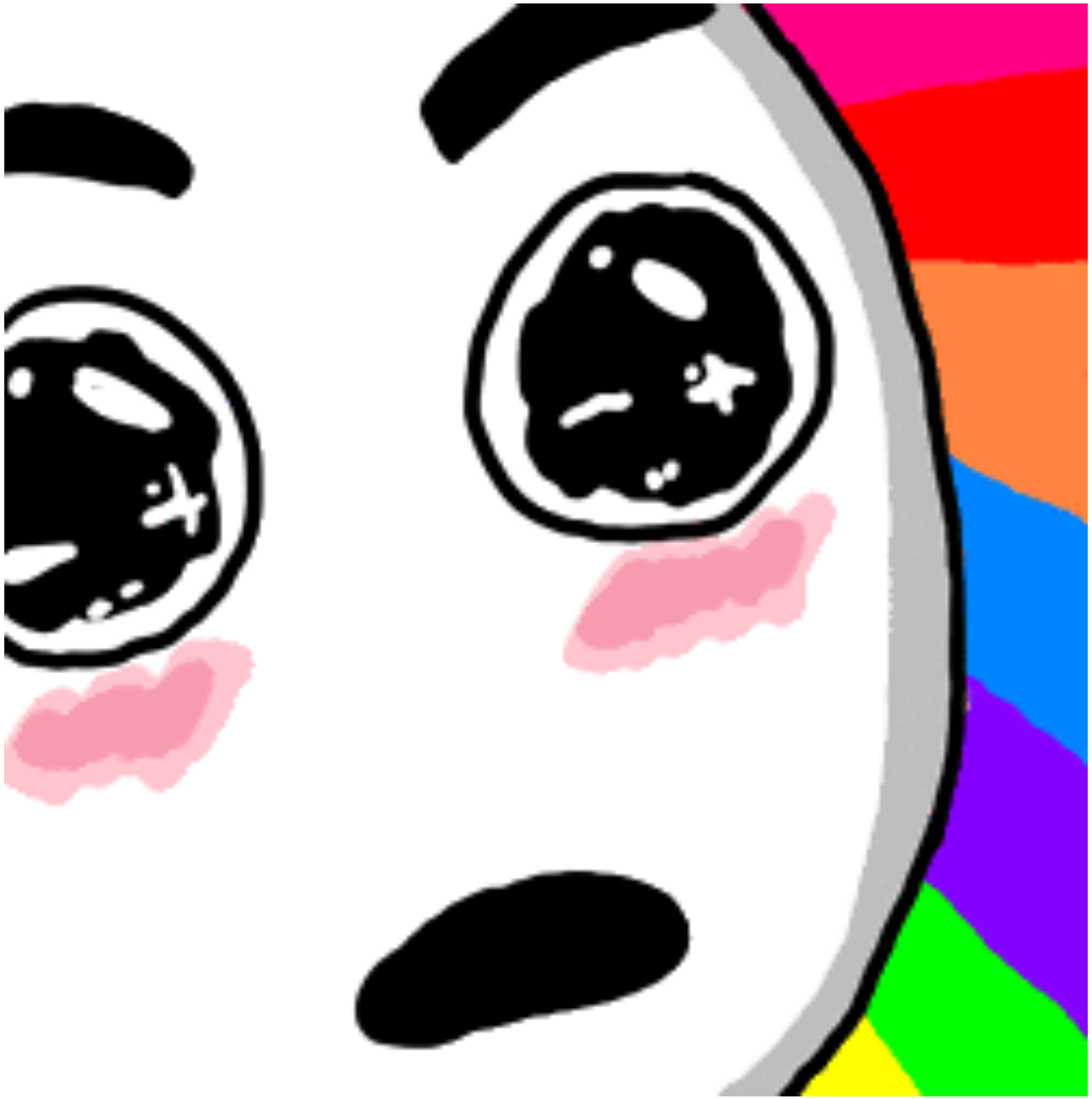
ANDRES ALMIRAY

@AALMIRAY

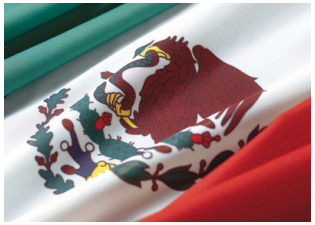
ANDRESALMIRAY.COM

DISCLAIMER

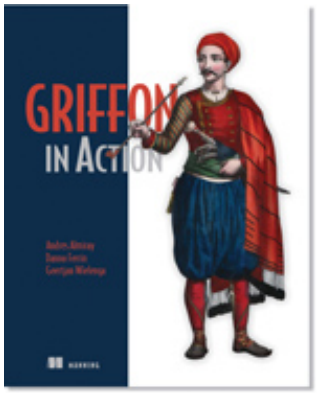

TM
open source



@aalmiray



canoo



JAVA™
CHAMPIONS

HACKERGARTEN

GET THE CODE

https://github.com/aalmiray/javatrove/

ASSERTIONS

Hamcrest - http://hamcrest.org/JavaHamcrest/

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.equalTo;

@RunWith(JUnitParamsRunner.class)
public class HelloServiceHamcrestTest {
    @Test
    @Parameters(source = HelloParameters.class)
    public void sayHello(String input, String output) {
        // given:
     HelloService service = new DefaultHelloService();
        // expect:
     assertThat(service.sayHello(input), equalTo(output));
   }
}
```

Hamcrest - http://hamcrest.org/JavaHamcrest/

- Descriptive conditions
- Easily extensible
- Bundled with JUnit

AssertJ-

http://joel-costigliola.github.io/assertj/

```
import static org.assertj.core.api.Assertions.assertThat;

@RunWith(JUnitParamsRunner.class)
public class HelloServiceAssertJTest {
    @Test
    @Parameters(source = HelloParameters.class)
    public void sayHello(String input, String output) {
        // given:
        HelloService service = new DefaultHelloService();
        // expect:
        assertThat(service.sayHello(input)).isEqualTo(output);
    }
}
```

AssertJ-

http://joel-costigliola.github.io/assertj/

- Fluent interface design for assertions
- Inspired by FEST-assert and Hamcrest
- Soft assertions

Truth- http://google.github.io/truth/

```
import static com.google.common.truth.Truth.assertThat;

@RunWith(JUnitParamsRunner.class)
public class HelloServiceThruthTest {
    @Test
    @Parameters(source = HelloParameters.class)
    public void sayHello(String input, String output) {
        // given:
        HelloService service = new DefaultHelloService();
        // expect:
        assertThat(service.sayHello(input)).isEqualTo(output);
    }
}
```

Truth- http://google.github.io/truth/

- Fluent interface design for assertions/ propositions
- Inspired by FEST-assert and Hamcrest
- Soft assertions
- Curated by Google
- http://google.github.io/truth/comparison

JGoTesting-

https://gitlab.com/tastapod/jgotesting

```
import org.jgotesting.rule.JGoTestRule;
import static org.hamcrest.Matchers.equalTo;

@RunWith(JUnitParamsRunner.class)
public class HelloServiceJGoTestingTest {
    @org.junit.Rule
    public final JGoTestRule test = new JGoTestRule();

    @Test
    @Parameters(source = HelloParameters.class)
    public void sayHello(String input, String output) {
        // given:
        HelloService service = new DefaultHelloService();
        // expect:
        test.check(service.sayHello(input), equalTo(output));
    }
}
```

JGoTesting-

https://gitlab.com/tastapod/jgotesting

- Inspired in Go's testing support
- Multiple checks may result in a single test failure -> soft assertions

P14N
(PARAMETERIZATION)

```
@RunWith(Parameterized.class)
public class HelloServiceParameterizedTest {
    @Parameterized.Parameters(name = "Invoking HelloService.sayHello("{0}") yields "{1}"")
    public static Collection<Object[]> data() {
        return Arrays.asList(
            new String[]{"", "Howdy stranger!"},
            new String[]{"Test", "Hello Test"}
        );
    }

    private final String input;
    private final String output;

    public HelloServiceParameterizedTest(String input, String output) {
        this.input = input;
        this.output = output;
    }

    @Test
    public void sayHello() {
        // given:
        HelloService service = new DefaultHelloService();
        // expect:
        assertThat(service.sayHello(input), equalTo(output));
    }
}
```

```
@RunWith(Parameterized.class)
public class HelloServiceParameterizedTest {
    @Parameterized.Parameters(name = "Invoking HelloService.sayHello("{0}")
yields "{1}"")
    public static Collection<Object[]> data() {
        return Arrays.asList(
            new String[]{"", "Howdy stranger!"},
            new String[]{"Test", "Hello Test"}
        );
    }

    @Parameterized.Parameter(0)
    public String input;

    @Parameterized.Parameter(1)
    public String output;

    @Test
    public void sayHello() {
        // given:
        HelloService service = new DefaultHelloService();
        // expect:
        assertThat(service.sayHello(input), equalTo(output));
    }
}
```

JUnitParams - https://github.com/Pragmatists/JUnitParams

```
@RunWith(JUnitParamsRunner.class)
public class SampleServiceTest {
    @Test
    @Parameters({",Howdy stranger!",
                "Test, Hello Test"})
    public void sayHello(String input, String output) {
        // given:
        SampleService service = new SampleService();

        // expect:
        assertThat(service.sayHello(input), equalTo(output));
    }
}
```

JUnitParams - https://github.com/Pragmatists/JUnitParams

- Parameterize multiple methods with different argument cardinality
- Different data provider strategies

MOCKING

Mockito - http://mockito.org

```
@Test @Parameters({",Howdy stranger!", "Test, Hello Test"})
public void sayHelloAction(String input, String output) {
    // given:
    SampleController controller = new SampleController();
    controller.setModel(new SampleModel());
    controller.setService(mock(SampleService.class));

     // expectations
    when(controller.getService().sayHello(input)).thenReturn(output);

    // when:
    controller.getModel().setInput(input);
    controller.sayHello();

    // then:
    assertThat(controller.getModel().getOutput(), equalTo(output));
    verify(controller.getService(), only()).sayHello(input);
}
```

Mockito - http://mockito.org

- Fluid DSL based on static methods
- Provides support for Stubs, Mocks, and Spies
- Mock interfaces, abstract classes, and concrete classes

Jukito - https://github.com/ArcBees/Jukito

```
@RunWith(JukitoRunner.class)
public class SampleControllerJukitoTest {
    @Inject private SampleController controller;

    @Before
    public void setupMocks(SampleService sampleService) {
        when(sampleService.sayHello("Test")).thenReturn("Hello Test");
    }

    @Test
    public void sayHelloAction() {
        controller.setModel(new SampleModel());
        controller.getModel().setInput("Test");
        controller.sayHello();
        assertThat(controller.getModel().getOutput(),
                        equalTo("Hello Test"));
        verify(controller.getService(), only()).sayHello("Test");
    }
}
```

Jukito - https://github.com/ArcBees/Jukito

- Combines JUnit, Guice, and Mockito
- Bind multiple values to the same source type
- Can be used to parameterize test methods

BDD

JGiven - http://jgiven.org/

```
@Test
public void default_greeting_is_returned_when_blank_input_is_given() {
    given().an_instance_of_the_service();

    when().the_sayHello_method_is_invoked_with_$s("");

    then().the_result_is_equal_to_$s("Howdy stranger!");
}

@Test
public void greeting_is_returned_when_input_is_given() {
    given().an_instance_of_the_service();

    when().the_sayHello_method_is_invoked_with_$s("Test");

    then().the_result_is_equal_to_$s("Hello Test");
}
```

JGiven - http://jgiven.org/

```
public static class TestSteps extends Stage<TestSteps> {
    private HelloService service;
    private String result;

    public void an_instance_of_the_service() {
        service = new DefaultHelloService();
    }

    public TestSteps the_sayHello_method_is_invoked_with_$s(String
input) {
        result = service.sayHello(input);
        return this;
    }

    public void the_result_is_equal_to_$s(String output) {
        assertThat(result, equalTo(output));
    }
}
```

JGiven - http://jgiven.org

- Java based BDD DSL
- Scenarios can be composed and reused
- Scenarios can be parameterized
- Custom HTML/Asciidoc reports

Spock- http://spockframework.org

```
@spock.lang.Unroll
class HelloServiceSpecification extends Specification {
    def "Invoking HelloService.sayHello('#input') yields '#output'"() {
        given: "an instance of HelloService"
        HelloService service = new DefaultHelloService()

        when: "the sayHello method is invoked with '#input'"
        String result = service.sayHello(input)

        then: "the result should be equal to '#output'"
        result == output

        where:
        input  | output
        ''     | 'Howdy stranger!'
        'Test' | 'Hello Test'
    }
}
```

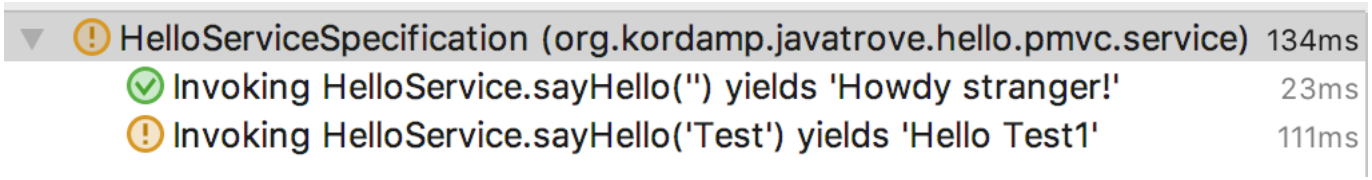
Spock- http://spockframework.org

```
@spock.lang.Unroll
class HelloServiceSpecification extends Specification {
    def "Invoking HelloService.sayHello('#input') yields '#output'"() {
        given: "an instance of HelloService"
        HelloService service = new DefaultHelloService()

        when: "the sayHello method is invoked with '#input'"
        String result = service.sayHello(input)

        then: "the result should be equal to '#output'"
        result == output

        where:
        input   << ['', 'Test']
        output << ['Howdy, stranger!', 'Hello Test']
    }
}
```




```
Condition not satisfied:

result == output
|      |  |
|      |  Hello Test1
|      false
|      1 difference (90% similarity)
|      Hello Test(-)
|      Hello Test(1)
Hello Test

Expected :Hello Test1

Actual   :Hello Test
```

@aalmiray

Spock- http://spockframework.org

```
@spock.lang.Unroll
class SampleControllerSpec extends spock.lang.Specification {
    def "Invoke say hello with #input results in #output"() {
        given:
            SampleController controller = new SampleController()
            controller.model = new SampleModel()
            controller.service = Mock(SampleService) {
                sayHello(input) >> output
            }
        when:
            controller.model.input = input
            controller.sayHello()
        then:
            controller.model.output == output
            1 * controller.service.sayHello(input)
        where:
            input   | output
            ''      | 'Howdy, stranger!'
            'Test'  | 'Hello Test'
    }
}
```

Spock- http://spockframework.org

- Groovy based DSL
- Parameterize multiple methods with different argument cardinality
- Parameterize test method names
- JUnit friendly (can use extensions and rules)
- Soft assertions (verifyAll)
- https://github.com/opaluchlukasz/junit2spock

WAITING

NETWORK AWARE JAVAFX APP

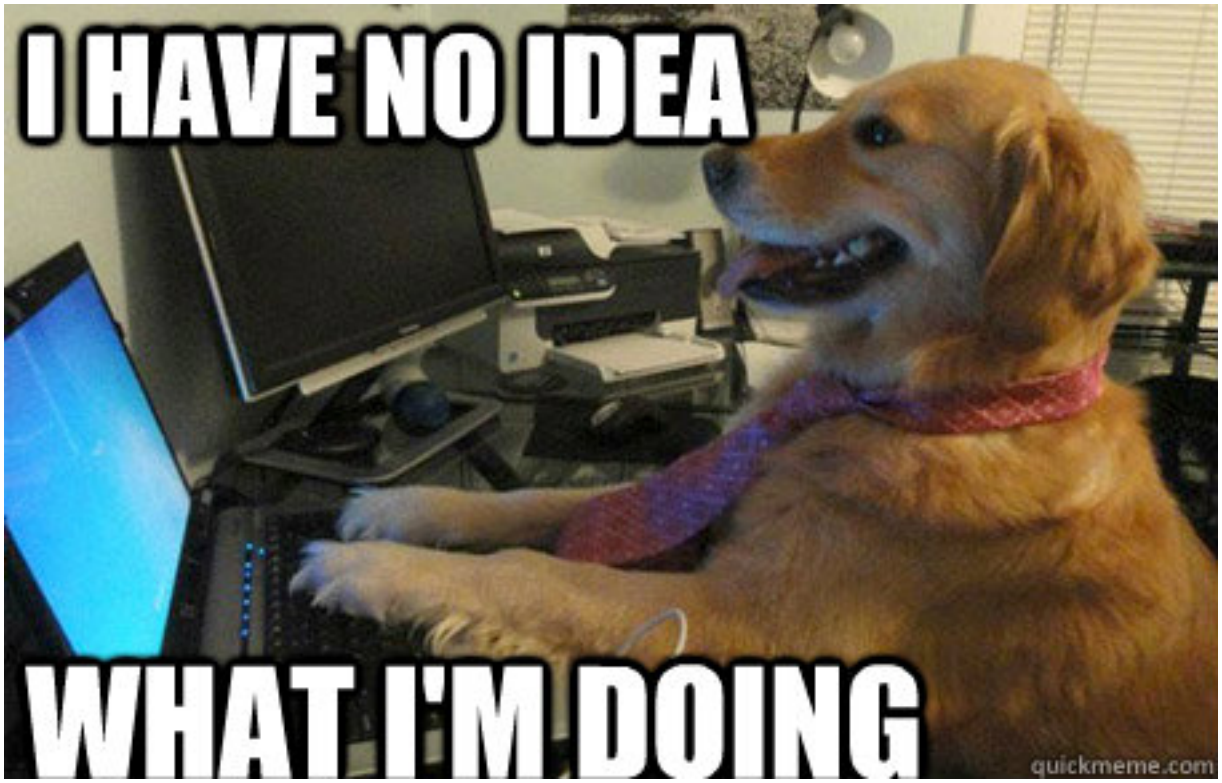
Write an application that consumes a REST API.

Components must be small and reusable.

Say no to boilerplate code.

Behavior should be easy to test.

DEMO



@aalmiray

Awaitility - https://github.com/awaitility/awaitility

```
@Test
public void happyPath(Github github) {
    // given:
    Collection<Repository> repositories = createSampleRepositories();
    when(github.repositories(ORGANIZATION))
        .thenReturn(Observable.from(repositories));

    // when:
    model.setOrganization(ORGANIZATION);
    controller.load();
    await().timeout(2, SECONDS).until(model::getState, equalTo(State.READY));

    // then:
    assertThat(model.getRepositories(), hasSize(10));
    assertThat(model.getRepositories(), equalTo(repositories));
    verify(github, only()).repositories(ORGANIZATION);
}
```

Awaitility - https://github.com/awaitility/awaitility

- DSL for testing multi-threaded code
- Extensions available for Java8, Groovy, and Scala
- Conditions can be customized with Hamcrest matchers

WireMock- http://wiremock.org/

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;

String nextUrl = "/organizations/1/repos?page=2";
List<Repository> repositories = createSampleRepositories();
stubFor(get(urlEqualTo("/orgs/" + ORGANIZATION + "/repos"))
    .willReturn(aResponse()
        .withStatus(200)
        .withHeader("Content-Type", "text/json")
        .withHeader("Link", "<http://localhost:8080" + nextUrl + ">; rel=\"next\"")
        .withBody(repositoriesAsJSON(repositories.subList(0, 5), objectMapper))));
stubFor(get(urlEqualTo(nextUrl ))
    .willReturn(aResponse()
        .withStatus(200)
        .withHeader("Content-Type", "text/json")
        .withBody(repositoriesAsJSON(repositories.subList(5, 10), objectMapper))));
```

WireMock- http://wiremock.org/

- Supply custom HTTP response payloads
- DSL for matching HTTP requests
- Supports record and playback

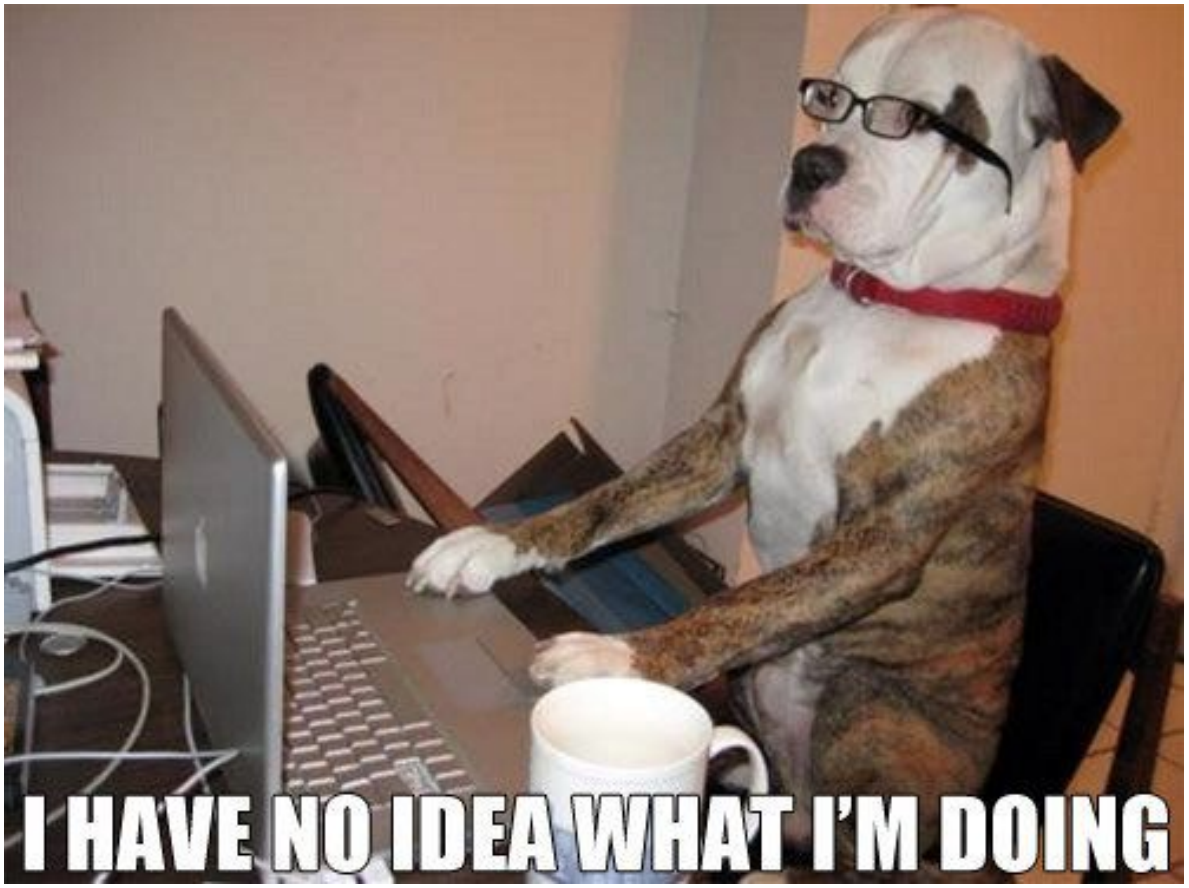
WEB

TRIVIAL TODO WEBAPP

Store todo items in a database

Expose operations via REST API

DEMO



@aalmiray

REST-assured -

https://github.com/rest-assured/rest-assured

```
import static io.restassured.RestAssured.given;
import static java.util.Arrays.asList;
import static org.hamcrest.Matchers.equalTo;

    @Test
    public void _01_initial_data_is_loaded() {
        given().port(4567).
        when().
            get("/todos").
        then().
            body("todos.description", equalTo(asList("Add Javadoc")));
    }
```

REST-assured -

https://github.com/rest-assured/rest-assured

- DSL for issuing HTTP requests
- Validate and navigate response body if JSON, XML, or HTML

Arquillian- http://arquillian.org

```
@Deployment
public static WebArchive createDeployment() throws Exception {
    File rootDir = new File(System.getProperty("user.dir"));

    File[] files = Maven.resolver()
        .resolve(...).withTransitivity().asFile();

    return ShrinkWrap.create(WebArchive.class, "application.war")
        .addPackage(”com.acme")
        .setWebXML(new File(rootDir, "src/main/webapp/WEB-INF/web.xml"))
        .addAsLibraries(files);
}
```

Arquillian- http://arquillian.org

- Container based testing, supporting Jetty, Tomcat, Glassfish
- Wildfly and others
- Create deployable archives with Shrinkwrap
- Combine with REST-assured for better results

MISC

DbUnit- http://dbunit.sourceforge.net/

```
@Test
public void testCleanInsert() throws Exception {
    // when:
    performInsertion();

    // then:
    String tableName = resolveDTOTableName();
    IDataSet databaseDataSet = resolveDatabaseConnection().createDataSet();
    ITable actualTable = databaseDataSet.getTable(tableName);

    StringWriter w = new StringWriter();
    FlatXmlWriter xml = new FlatXmlWriter(w);
    xml.write(databaseDataSet);
    IDataSet expectedDataSet = createDataSetForInsertion();
    ITable expectedTable = expectedDataSet.getTable(tableName);

    w = new StringWriter();
    xml = new FlatXmlWriter(w);
    xml.write(expectedDataSet);
    assertEqualsIgnoreCols(expectedTable, actualTable, ignoredColumns());
}
```

XMLUnit- http://www.xmlunit.org

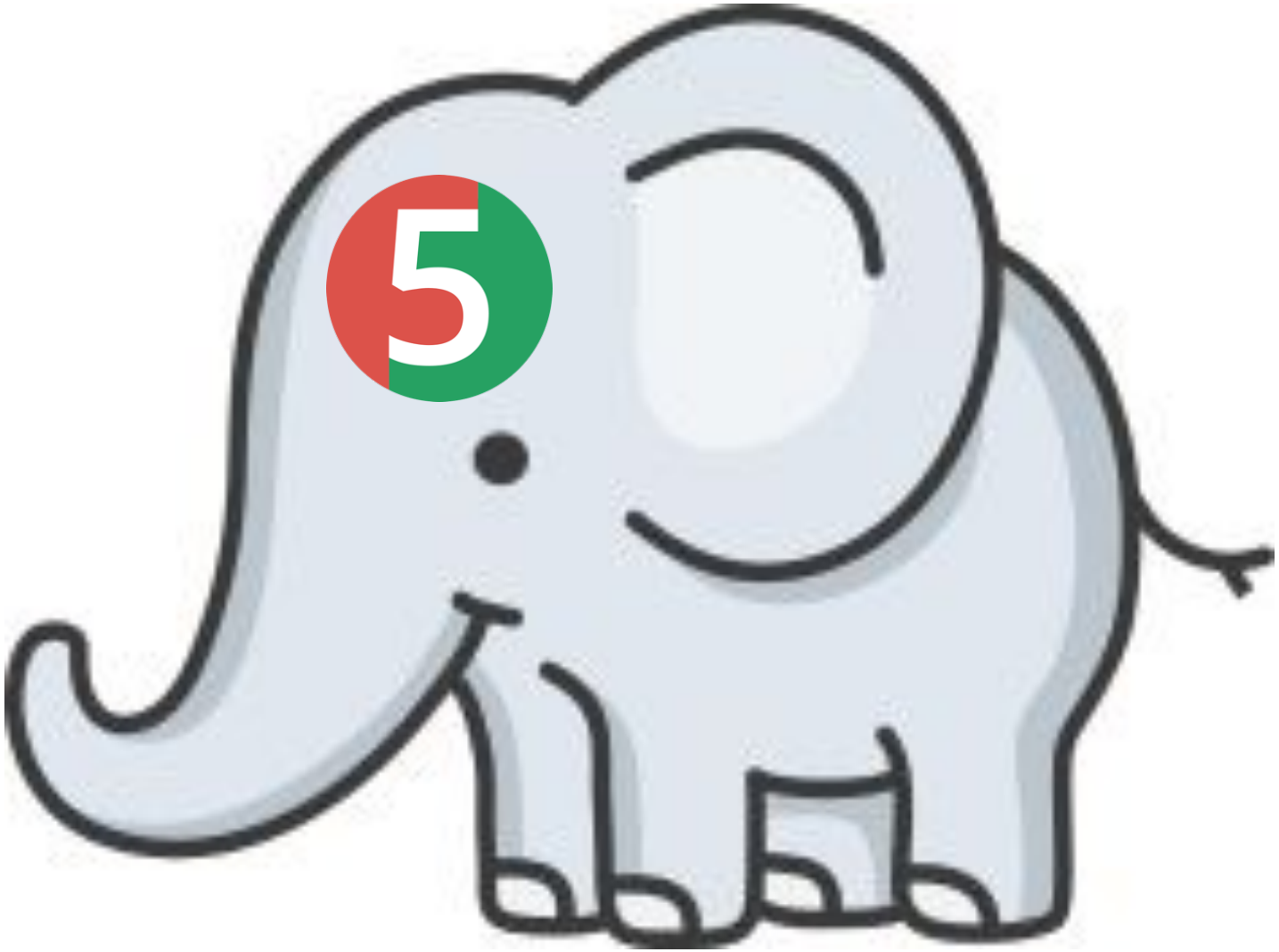
```
import org.custommonkey.xmlunit.XMLTestCase;

public class TestXmlContainingTypeAttribute extends XMLTestCase {
  public void testXmlWithTypeAttribute() throws Exception {
    String xml = "<data type=\"someType\">" +
      "<nested type=\"some_other_type\">value</nested></data>";
    XMLSerializer tested = new XMLSerializer();
    tested.setTypeHintsEnabled(false);
    tested.setTypeHintsCompatibility(false);
    tested.setRootName("data");

    JSON jsonRepresentation = tested.read(xml);
    String result = tested.write(jsonRepresentation);
    assertXMLEqual(xml, result);
  }
}
```

PdfUnit- http://www.pdfunit.com

```
@Test
public void hasText_OnFirstPage_InRegion() throws Exception {
    String pdfUnderTest = "documentUnderTest.pdf";
    int leftX = 17; // in millimeter
    int upperY = 45;
    int width = 80;
    int height = 50;
    PageRegion addressRegion = new PageRegion(leftX, upperY, width, height);
    AssertThat.document(pdfUnderTest)
          .restrictedTo(FIRST_PAGE)
          .restrictedTo(addressRegion) .hasText()
          .containing("John Doe Ltd.");
}
```



https://i.pinimg.com/736x/6b/47/05/6b47050f0f15b955cf725609749e1c87--elephant-baby-baby-elephants.jpg

@aalmiray

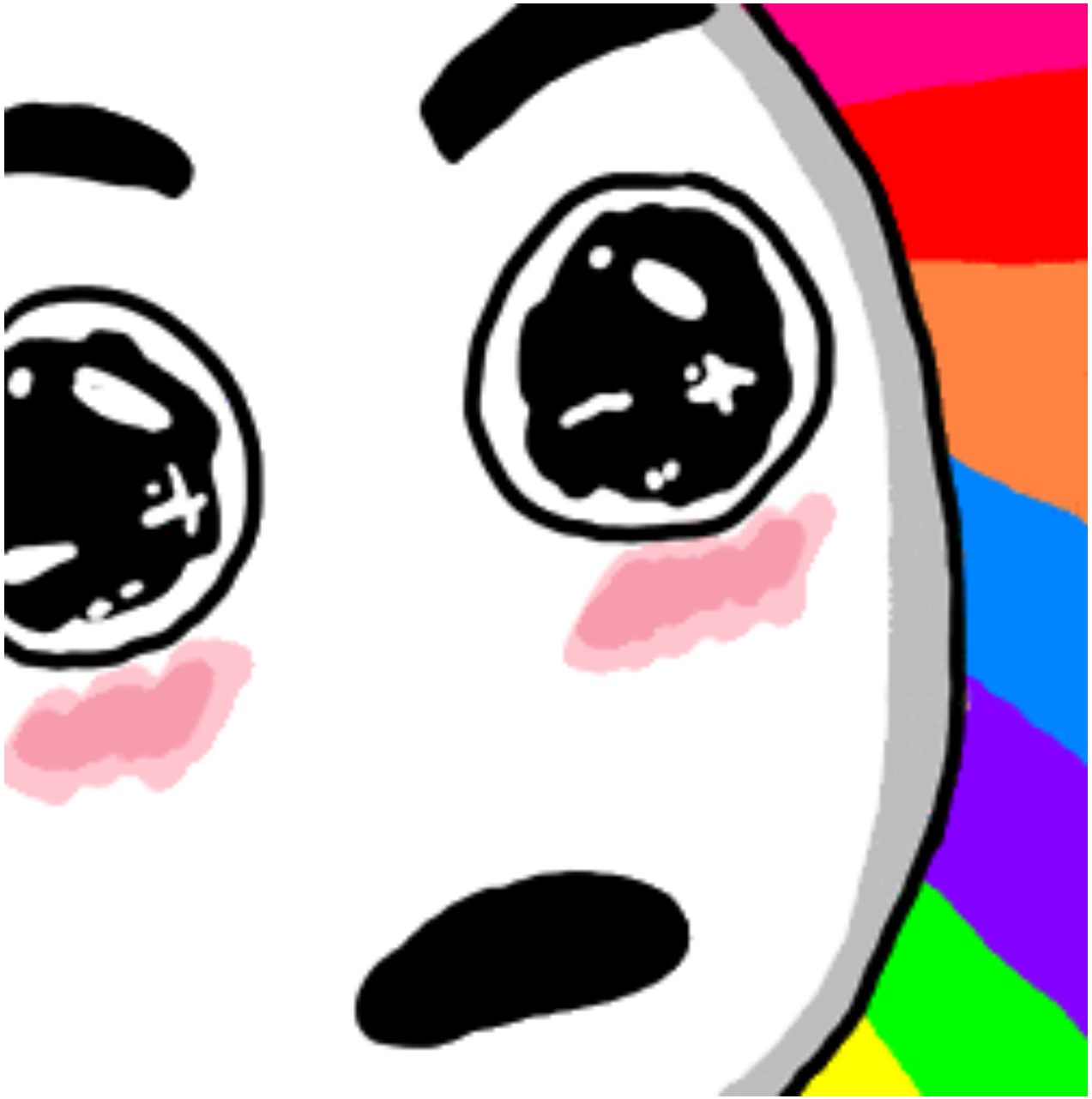
JUnit5 - http://junit.org/junit5/

- Redesigned with Java8 in mind. Harness lambda expressions/method references
- Homogeneous Extension API (no more @Rule and @ClassRule)
- Can run JUnit 4.x tests
- Experimental parameterization API
- http://junit.org/junit5/docs/current/user-guide/#migrating-from-junit4

@aalmiray

HTTP://ANDRESALMIRAY.COM/NEWSLETTER

HTTP://ANDRESALMIRAY.COM/EDITORIAL



@aalmiray

THANK YOU!

ANDRES ALMIRAY
@AALMIRAY

ANDRESALMIRAY.COM