Serenity BDD

Beyond the Basics!

Stefan Schenk

DE TESTERS

@stefanschenk_

Mike van Vendeloo

jpoint

@mikevanvendeloo

Agenda

- Serenity / JBehave introduction
- BDD in a microservice landscape
- Challenges
  - Reusing stories and steps
  - Aliases & Converters
  - Timeouts
  - Stale Elements
  - Continuous Integration
  - Reporting
- Work in Progress
- Questions

Serenity / JBehave introduction

- There are many frameworks available for testing that rely on BDD techniques.
- The one that we use here at the customer is Serenity BDD in combination with JBehave.

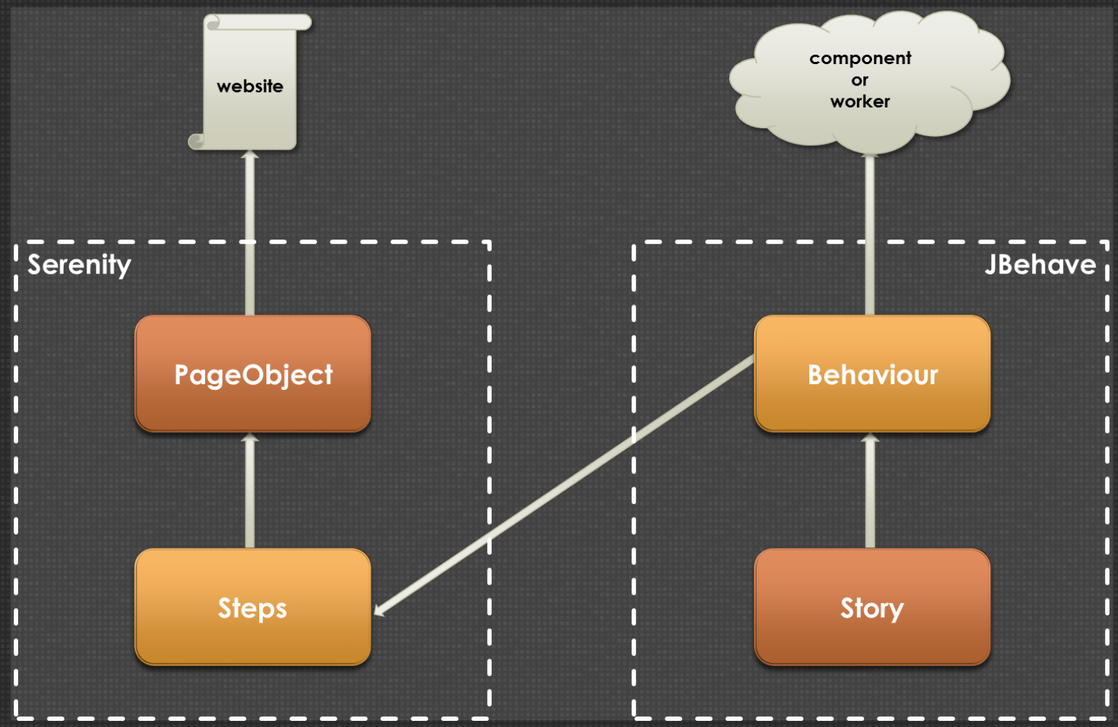
JBehave

- JBehave is a BDD framework, which allows us to write stories in plain text...
- ...and map the steps in these stories to Java.
- jbehave.org website

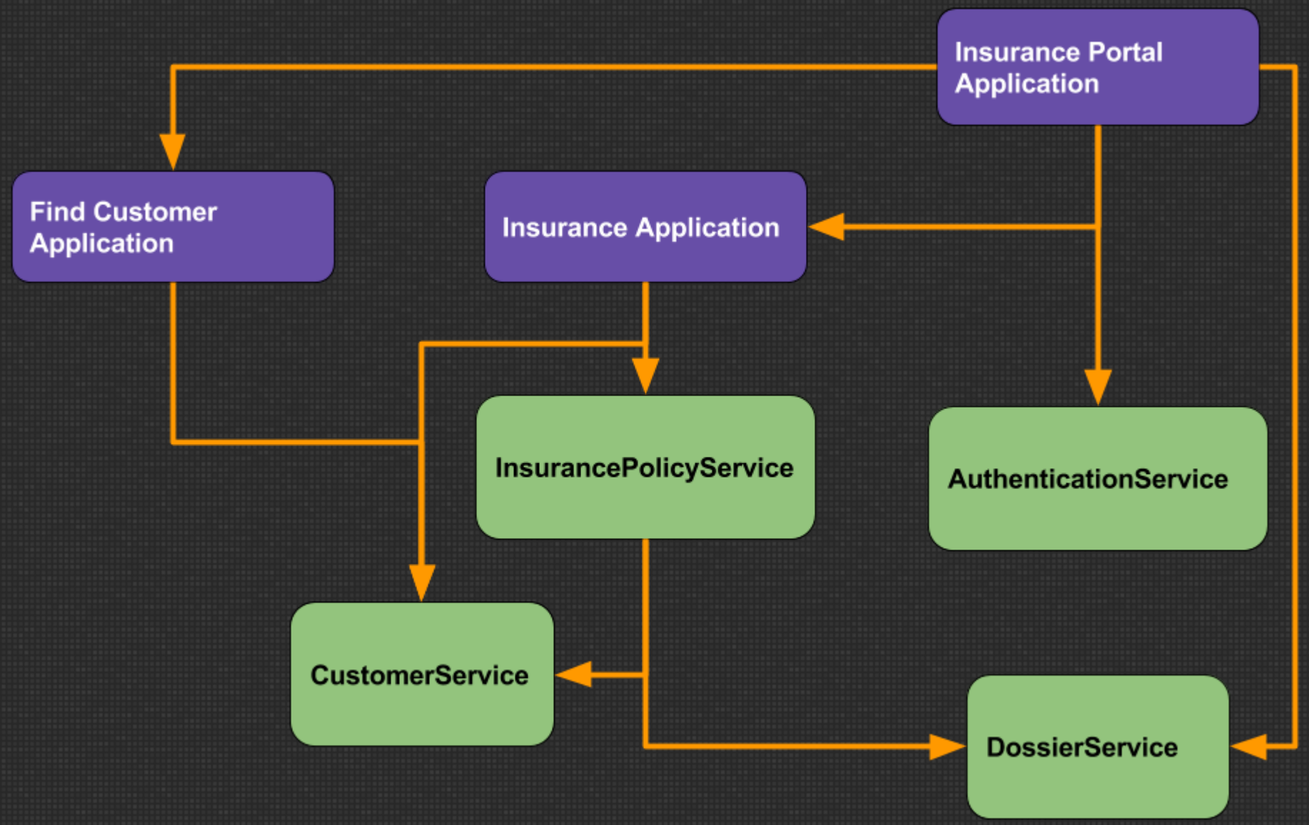
Serenity BDD

- Serenity is a framework that enables us to write, maintain and run tests on all our microservice components.
- By working together with the JBehave framework to test components and workers and by using the built-in Selenium support to test the web applications.
- It provides a way to start and run the tests using IntelliJ and Maven and generate reports about the test results, whether running locally or via Jenkins.
- serenity bdd website

Connecting stories to testobjects



BDD in a microservices landscape



Reusing stories and steps

Packaging

Each application has its own regression test set

which is packaged as a maven artifact

Packaging (2)

And unpacked to be able to reuse the steps

```
<plugin>
        <groupid>org.apache.maven.plugins</groupid>
        <artifactid>maven-dependency-plugin</artifactid>
        <version>2.10</version>
        <executions>
        <execution>
                <id>unpack-test-dependencies</id>
                <phase>generate-test-resources</phase>
                <goals>
                        <goal>unpack</goal>
                </goals>
                        <configuration>
                                <artifactitems>
                                        <artifactitem>
                                                <groupid>nl.jpoint.detesters.searchapp</groupid>
                                                <artifactid>regression-test</artifactid>
                                                <version>${searchapp.version}</version>
                                                <type>test-jar</type>
                                                <overwrite>false</overwrite>
                                                <outputdirectory>${project.build.directory}/test-classes</outputdirectory>
                                        </artifactitem>
                                </artifactitems>
                        </configuration>
                </execution>
        </executions>
</plugin>
```

Reusing steps from other packages (1)

Difference between Cucumber and JBehave

Cucumber

```
@RunWith(CucumberWithSerenity.class)
@CucumberOptions(features="src/test/resources/features/bla.feature")
public class DefinitionTestSuite {}
```

JBehave

```
public class AcceptanceTestSuite extends SerenityStories {}
```

Reusing steps from other packages (2)

```
/**
 * The root package on the classpath containing the JBehave stories to be run.
 */
protected String getStoryPath() {
        return (StringUtils.isEmpty(storyFolder)) ? storyNamePattern :
                                storyFolder + "/" + storyNamePattern;
}
```

```
@Override
public InjectableStepsFactory stepsFactory() {
        return CustomStepFactory.withStepsFromPackage(getRootPackage(),
                                configuration(),
                                getExtraPackagesFromEnvironment());
}
```

Reusing steps from other packages (3)

```
public class CustomStepFactory extends SerenityStepFactory {
        private final String rootStepPackage;
        private List<String> extraStepPackages = new ArrayList<>();

        @Override
        protected List<Class<?>> stepsTypes() {
        List<Class<?>> types = new ArrayList<Class<?>>();
        types.addAll(getStepTypesFromPackage(rootStepPackage));
        for (final String packageName : extraStepPackages) {
                        types.addAll(getStepTypesFromPackage(packageName));
        }
        return types;
        }
}
```

Reusing stories example: Given stories

```
GivenStories:   stories/authentication/login/login.story#{usage:precondition demo_user},
                stories/portal/show_dossier/create_dossier.story#{usage:precondition}
Given the dossier is created
When opening the dossier
Then do usefull stuff with the dossier
```

stories/authentication/login/login.story

```
Meta:
@usage precondition demo_user
Given the user is logged out and on the login page
When the user logs in with:
|username  |password       |
|demo_user |secretpassword |
Then demo_user is logged in
```

Reusing steps example

```
Given a new insurance application
When my contact details are filled in
And the insurance application is saved
Then the insurance application has an reference number
```

```
package nl.jpoint.detesters.insuranceapplication;

@When("my contact details are filled in")
public void enterContactDetails() {
        insuranceApplicationSteps.enterContactDetails();
}
```

```
package nl.jpoint.detesters.commonbehaviour;

@When("the insurance application is saved")
public void Save() {
        commonSteps.save();
}
```

Writing your own parameter converters

example converter for BigDecimal

```
public class BigDecimalConverter() implements ParameterConverter {
        @Override
        public boolean accept(Type type) {
                if (type instanceof Class<?>) {
                        return BigDecimal.class.isAssignableFrom((Class<?>) type);
                }
                return false;
        }

        @Override
        public Object convertValue(String value, Type type) {
                return new BigDecimal(value);
        }
}
```

custom object to use with converter

```
public String value;

private String convertValue(String value) {
        String convertedValue = value;

        while (convertedValue.matches(".*<.*>.*")) {
                String parameter = convertedValue.substring(convertedValue.indexOf("<") + 1, convertedValue.indexOf(">"));

                if (parameter.startsWith("datum:")) {
                        String storyDate = parameter.split(":")[1];
                        String date;

                        if (storyDate.contains(",")) {
                                DateTimeFormatter storyDateFormat = DateTimeFormat.forPattern(storyDate.split(",")[1]);
                                date = TestDataUtilities.convertStoryDateSmart(storyDate.split(",")[0], storyDateFormat);
                        } else {
                                date = TestDataUtilities.convertStoryDateSmart(storyDate, TestDataUtilities.screenDateFormat);
                        }

                        convertedValue = convertedValue.replace("<" + parameter + ">", date);
                } else if (parameter.equalsIgnoreCase("uuid")) {
                        convertedValue = convertedValue.replace("<" + parameter + ">", UUID.randomUUID().toString());
                } else {
                        String valueFromSession = Behaviour.getSessionInterface().get(parameter).toString();
                        convertedValue = convertedValue.replace("<" + parameter + ">", valueFromSession);
                }
        }

        return convertedValue;
}
```

our converter to utilize the object

```
public class StoryStringConverter implements ParameterConverter {
        @Override
        public boolean accept(Type type) {
        if (type instanceof Class<?>) {
                        return StoryString.class.isAssignableFrom((Class<?>) type);
                }
                return false;
        }

        @Override
        public Object convertValue(String value, Type type) {
        return new StoryString(value);
        }
}
```

extending the list of converters

```
public class ExtendedSerenityStory extends SerenityStory {
        @Override
        public Configuration configuration() {
        if (configuration == null) {
                Configuration thucydidesConfiguration = getSystemConfiguration();

                if (environmentVariables != null) {
                        thucydidesConfiguration = thucydidesConfiguration.withEnvironmentVariables(environmentVariables);
                        }

                configuration = SerenityJBehave.defaultConfiguration(thucydidesConfiguration, formats, this);
                configuration.parameterConverters().addConverters(ConverterUtils.customConverters());
        }
        return configuration;
        }
}
```

```
public final class ConverterUtils {
        public static ParameterConverter[] customConverters() {
        List<ParameterConverter> converters = new ArrayList<ParameterConverter>();
        converters.add(new BigDecimalConverter());
        converters.add(new StoryStringConverter());

        return converters.toArray(new ParameterConverter[converters.size()]);
        }
}
```

How to use it in your story / behaviour

```
And I expect a file with the following information
|information                                                  |
|<filenumber>                                                 |
|1                                                            |
|This file is valid from <date:today>                         |
|VALUES:Summary of amounts and values with file <filenumber>|
```

```
@Then("I expect a polisblad with the following information $information")
public void thenIExpectPolisbladWithInformation(@Named("information") final ExamplesTable information) {
        for (Parameters parameters : information.getRowsAsParameters()) {
        String expectedInfo = parameters.valueAs("information", StoryString.class).value;

        assertThat("", actual, is(expectedInfo));
        }
}
```

Aliases and Optional parameters in a story

- Aliases can be declared by using @Alias
- or by using @Aliases
- JBehave also has a automatic aliases mechanism
- use { option 1 | option x } in your behaviour step definition

Aliases and Optional parameters in a story

Example with parameters outside the alias options

- You have selected a product in your Given step
- And you would like to use different texts in a When step
- You could write an @When with multiple @Aliases, but you could also write it as a oneliner

```
@When("I add $number {bananas|pears|apples} to my cart")
```

```
When I add 5 bananas to my cart
When I add 3 pears to my cart
When I add 10 apples to my cart
```

```
@When("I add $number bananas to my cart")
@When("I add $number pears to my cart")
@When("I add $number apples to my cart")
```

Aliases and Optional parameters in a story

- You can use parameters in your aliases
- If you use the same type and number in all aliases it will work without trouble

```
@When("I {remove|add|change} $items {from|to|in} the database")
private void whenAlteringTheDatabase(@Named("items") final String items)
```

```
@When("I {remove $items from|add $items to|change $items in} the database")
private void whenAlteringTheDatabase(@Named("items") final String items)
```

Aliases and Optional parameters in a story

```
@When("I want the parameter {$parameter |}to be optional")
private void whenUsingOptionalParameters(@Named("parameter") final String myParameter)
```

```
@When("I want the parameter {$parameter |}to be optional")
private void whenUsingOptionalParameters(@Named("parameter") final int myParameter)
```

```
@When("I want the parameter {$parameter |$0}to be optional")
private void whenUsingOptionalParameters(@Named("parameter") final Optional<int> myParameter)
```

```
Given I am logged in
When I post some json to a component => response
Then I expect a certain result <= response
```

```
@When("I post some json to a component {$sessionIO|$0}")
private void whenIPostSomething(@Named("sessionIO") final Optional<SessionIO> sessionIO)
```

Timeouts

Finding elements

```
# How long does Serenity wait for elements that are not present on the screen to load
webdriver.timeouts.implicitlywait = 6000

# How long webdriver waits by default when you use a fluent waiting method, in milliseconds.
webdriver.wait.for.timeout = 10000
```

```
@FindBy(id="slow-loader")
public WebElementFacade slowLoadingField;

@FindBy(id="top-advertisement")
WebElementFacade topAdvertisement;
...
public WebElementState topPositionAdvertisement() {
    return withTimeoutOf(15, TimeUnit.SECONDS).waitFor(topAdvertisement);
}
```

Timeouts

Story timeout

```
story.timeout.in.secs=3600
```

Note: the default timeout is 300 seconds (5 minutes)

Timeouts

Ajax wait until ready timeout

Based on a solution found on StackOverflow for use with PrimeFaces

```
new FluentWait(webDriver).withTimeout(TIME_OUT_SECONDS, TimeUnit.SECONDS)
        .pollingEvery(POLLING_MILLISECONDS, TimeUnit.MILLISECONDS)
        .until(new Function<WebDriver, Boolean>() {
        @Override
        public Boolean apply(WebDriver input) {
                ...
        }
}
```

http://stackoverflow.com/questions/23300126/selenium-wait-for-primefaces-4-0-ajax-to-process

Stale Elements (1)

Have you ever tried googling for StaleElementReferenceException?
You probably found http://docs.seleniumhq.org/exceptions/stale_element_reference.jsp

But did you find a solution on this page?

Stale Elements (2)

```
SmartElementHandler smartElementHandler = (SmartElementHandler) java.lang.reflect.Proxy.getInvocationHandler(webElementFacade);

Field fieldLocator = smartElementHandler.getClass().getSuperclass().getDeclaredField("locator");
fieldLocator.setAccessible(true);
SmartAjaxElementLocator smartAjaxElementLocator = (SmartAjaxElementLocator) fieldLocator.get(smartElementHandler);

Field fieldBy = smartAjaxElementLocator.getClass().getSuperclass().getDeclaredField("by");
fieldBy.setAccessible(true);

if (fieldBy.get(smartAjaxElementLocator) instanceof By) {
        By by = (By) fieldBy.get(smartAjaxElementLocator);

        LOG.info("Element ({}) found in RenderedView: {}", by, renderedPageObjectView.elementIsPresent(by));
        relocatedElement = renderedPageObjectView.find(by);
} else {
        LOG.warn("Unknown selector found: {}", fieldBy.get(smartAjaxElementLocator));
}
```

Stale Elements (3)

```
@FindBy(linkText = "Save")
private WebElementFacade buttonSave;

try {
        buttonSave.click();
} catch (StaleElementReferenceException ex) {
        RelocateElement.find(this.getRenderedView(), buttonSave).click();
}
```

Continuous Integration - Environment profiles

Using profiles for environment selection

```
<profile>
        <id>demo</id>
        <properties>
                <environment>acceptance</environment>
                <application.base.url>http://searchapp.acc.insurance.nl</application.base.url>
                <number.of.threads>4</number.of.threads>
        </properties>
</profile>
<profile>
        <id>prod</id>
        <properties>
                <environment>production</environment>
                <application.base.url>http://searchapp.insurance.nl</application.base.url>
                <number.of.threads>8</number.of.threads>
        </properties>
</profile>
```

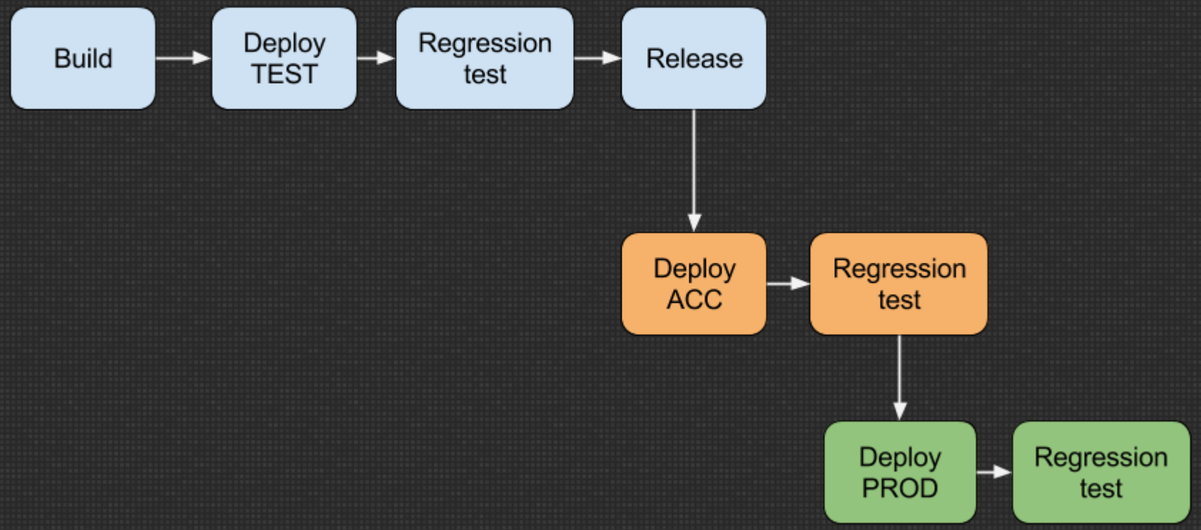
Continuous Integration - Optimization

Use the forkcount to optimize the duration of your test set

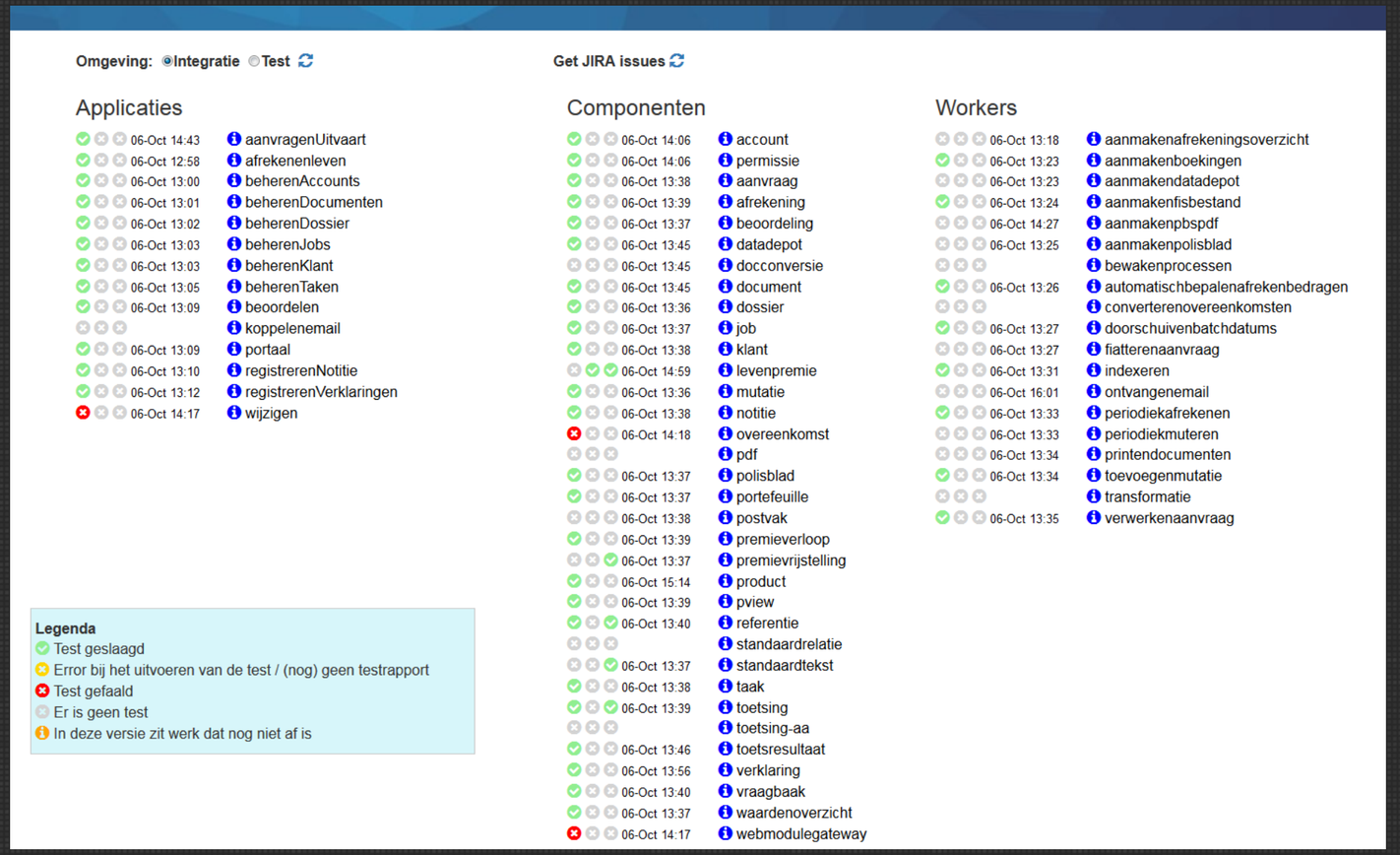
```
<profile>
        <id>regression-test</id>
        ...
        <plugin>
        <artifactid>maven-failsafe-plugin</artifactid>
        <version>2.18.1</version>
        <configuration>
                        <skip>false</skip>
                        <includes>
                                <include>${test.runner.class}</include>
                        </includes>
                        <forkcount>${number.of.threads}</forkcount>
                        <systempropertyvariables>
                                <webdriver.driver>${webdriver.driver}</webdriver.driver>
                                <webdriver.base.url>${application.base.url}</webdriver.base.url>
                                <environment>${environment}</environment>
                        </systempropertyvariables>
        </configuration>
  </plugin>
  ...
</profile>
```

Running in a Jenkins Pipeline

Running the regression test / acceptane test after every deploy

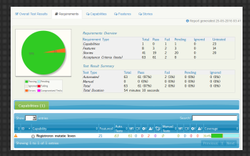


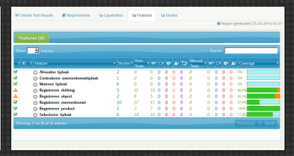
Report integration

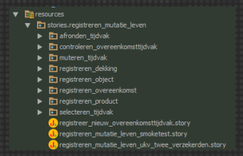


Reporting, requirements and story structure

- Serenity can focus in the reports on what features were delivered.
- For this to work, Serenity needs to know how the requirements are structured.
- The simplest way to represent a requirements structure in Serenity is to use a hierarchical directory structure, where each top level directory describes a high level capability or functional domain.
- You might break these directories down further into sub directories representing more detailed functional areas, or just place feature files directly in these directories







Reporting, requirements and story structure

open a report

Linking Jira issues in Serenity reports

Linking Jira issues in Serenity reports

First, create or update the serenity.properties file with connection settings to Jira

```
jira.url = http://myjiraserver
jira.project = PRJ
jira.username = jirauser
jira.password = t0pSecret
```

Linking Jira issues in Serenity reports

To link an issue to one of your stories, open or create a story and add the Meta tag @issue to that story

```
Meta:
@issue PRJ-38

Login
Narrative:
As a user I would like to login on the portal

Scenario: Login succesfully
```

Linking Jira issues in Serenity reports

It is also possible to link an issue to a specific scenario in your story

```
Meta:

Login
Narrative:
As a user I would like to login on the portal

Scenario: Login succesfully
Meta:
@issue PRJ-2
```

Linking Jira issues in Serenity reports

Linking Jira issues in Serenity reports

Commenting test status by Serenity

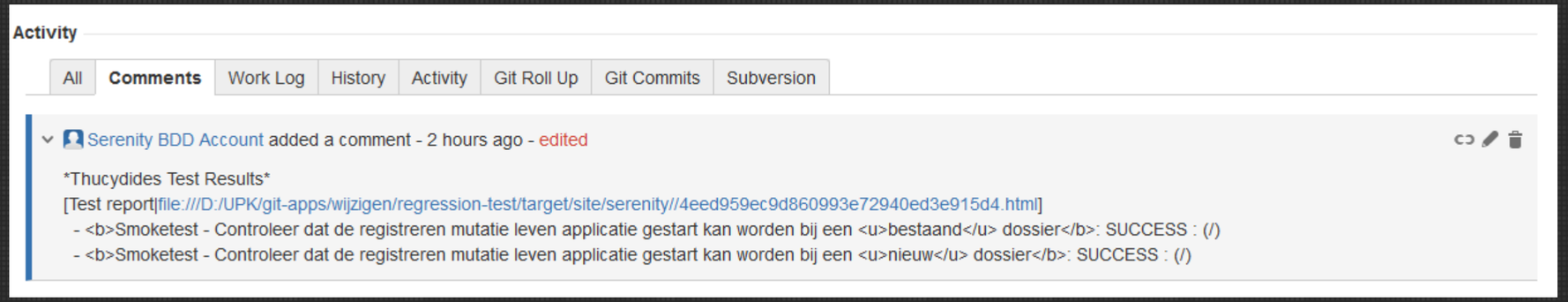
We can go a step further with our Jira integration
We can tell Serenity to comment on a Jira issue with the status of the tests run

```
<dependency>
  <groupid>net.serenity-bdd</groupid>
  <artifactid>serenity-jira-plugin</artifactid>
  <version>1.1.2-rc.1</version>
</dependency>
```

```
serenity.public.url=http://my-jenkins-server/tests/site/serenity
```

```
serenity.public.url=file:///my-project/test/target/site/serenity/
```

Commenting test status by Serenity



Change the state of the issue by Serenity

After linking issues in you reports and add test results in a comment
We come to the final Serenity / Jira integration

```
serenity.jira.workflow.active=true
```

Change the state of the issue by Serenity

Workflow Configuration

```
when'Open',{
  'success' should:'Resolve Issue'
}

when'Reopened',{
  'success' should:'Resolve Issue'
}

when'Resolved',{
  'failure' should:'Reopen Issue'
}

when'In Progress',{
  'success' should:['Stop Progress','Resolve Issue']
}

when'Closed',{
  'failure' should:'Reopen Issue'
}
```

```
thucydides.jira.workflow=my-workflow.groovy
```

Work in Progress (1)

- As testers we were enthousiastic about Serenity and JBehave and its reporting capabilities
- It always was an effort shared between technical testers and developers like us
- And now we chose to use this framework to use with unit testing

Work in Progress (2)

- You don't need PageObjects or a Serenity Steps class to test
- But we do use JBehave stories and Behaviour steps to write our tests
- Which we run and report about with the Serenity framework

Work in Progress (3)

```
Meta:

Narrative:
As a user
I want to login with my username and password
So I can execute my tasks at hand

Scenario: successful authentication with username and password
Given searching in UserRepository on username will return a user with id = 123, pki = DUS, name = Demo User, password = Secr3t
And user has 0 failed login attempts
And user has changed password 5 days ago
And changing user in UserRepository results in success
When authentication with username = userd, password = Secr3t
Then authenticationState = OK
And result contains id = 123, code = DUS, naam = Demo User
And UserRepository has been searched for username userd
And successful authentication is processed in UserRepository
```

Work in Progress (4)

```
public class AuthenticationSteps implements WithHamcrest, WithMockito {
  private UserRepository userRepository;
  protected User user;
  protected Authentication authentication;
  protected AuthenticationResult result;

  @BeforeScenario
  public void before()
  {
    TokenProvider tokenProvider = mock(TokenProvider.class);
    TokenManager.add(tokenProvider);
    when(tokenProvider.sign(any())).thenReturn(new Token());
    user = null;
    authentication = new Authentication();
  }
```

```
  @Given("searching in UserRepository on username will return a user with id = $id, pki = $pki, name = $name, password = $password")
  public void givenUserRepositoryByUsername(Id id, String pki, String name, Octets password)
  {
    user = mock(User.class);
    when(user.getId()).thenReturn(id);
    when(user.getPKI()).thenReturn(pki);
    when(user.getFormattedName()).thenReturn(name);
    when(user.getPassword()).thenReturn(password);
    when(user.success()).thenReturn(user);
    when(user.failure()).thenReturn(user);
    when(user.changed(any(),any())).thenReturn(user);
    when(user.getByUsername(any())).thenReturn(Try.success(user));
  }
```

Work in Progress (5)

Serenity BDD

Home

Overall Test Results | Requirements | Capabilities | Features | Stories

Report generated 06-10-2016 18:15

Test Results: All Tests

8 test scenarios

8 passed , 0 pending, 0 failed, 0 errors, 0 compromised, 0 ignored, 0 skipped [CSV]

Test Count | Weighted Tests

Total number of tests that pass, fail, or are pending.

Show/Hide Pie Chart

100%

Passing Pending Ignored
Failing Errors Compromised

Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	8	8 (100%)	0 (0%)	0 (0%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	8	8 (100%)	0 (0%)	0 (0%)	0 (0%)
Total Duration	1 seconds				

Related Tags	% Passed	Test count
Capabilities		
Account	100%	8
Features		
Process	100%	8
Stories		
Authenticeren J Behave Test	100%	8

Tests

Show entries Search:

Tests	Steps	Duration (seconds)
authenticeren met een correct maar verlopen wachtwoord	7	0.03
authenticeren met een foutief en verlopen wachtwoord	8	0.03
authenticeren met een geblokkeerde gebruiker	8	0.03
authenticeren met foutief wachtwoord	8	0.03
authenticeren met GebruikerRepository leesfout	3	0.02
authenticeren met GebruikerRepository schrijffout	7	0.03
authenticeren met niet bestaande gebruikersnaam	3	0.03
succesvol authenticeren met gebruikersnaam en wachtwoord	9	0.48

Showing 1 to 8 of 8 entries Previous 1 Next

Serenity BDD

Beyond the Basics!

Thank you for listening
are there any questions?