

Technical Practices Workshop

# Participant Guide

# Overview

Hands-on introduction to key software engineering practices that support an Agile and Lean approach to solution development and delivery.

## Duration

Three days

## Authors

Dave Nicolette, LeadingAgile Technical Practie Lead

Jann Thomas, author of the original version of the workshop

# Technical Prerequisites

Each computer that will be used in the technical exercises needs the following:

* Public Internet access (not firewalled corporate network)
* The Java IDE they will use at work
* If they're using Ruby Cucumber, the Ruby IDE they will use at work
* Git client (if not provided by the IDE)
* MySQL database system installed locally
* Ruby 2.3.1 or later installed locally

# Outline

## Day 1 Morning

### 00:00-0020 Getting Started

* Name Tents (activity)
* Team Agreement (activity)
* Team Goals (activity)
* Why are you here? (activity)
* Parking Lot

### 00:20-01:00 Section 1: Business Drivers

* Predictability (activity)
* Quality (activity)
* Speed (activity)
* Value (activity)

### 01:00-01:45 Collaboration

* Collaboration game

### 01:45-02:15 Self-Organization

* Ball Point game

### 02:15-02:45 Transparency and Trust

* Transparency and Trust game

### 02:45-03:00 Day 1 Morning Retrospective

* Facilitated retrospective

### 03:00-03:45 Verify technical setup

* Ensure everyone has what they need installed

## Day 1 Afternoon

### 00:00-00:45 TDD Kickstart

* Greenfield TDD round 1 (FizzBuzz)
* Debrief
* Explanation and demonstration

### 00:45-01:45 TDD for Greenfield Development

* Story decomposition
* Greenfield TDD round 2 (FizzBuzz)
* Debrief

### 01:45-02:00 TDD Introduction - Retrospective

* Facilitated recap and retrospective on the TDD intro

### 02:00-02:10 Compass Reading

* Basecamps and technical practices

### 02:10-02:25 Test Automation Pyramid

* Lecture and discussion

### 02:25-02:55 Continuous Delivery

* Lecture and discussion

## Day 2 Morning

### 00:00-02:00 Refactoring Kickstart

* Gilded Rose round 1
* Debrief
* Lecture and demonstration
* Gilded Rose round 2

### 02:00-02:30 Agile Planning

* Multiple Planning Horizons

### 02:30-03:00 Agile Requirements

* Big Design Up Front (BDUF)
* Insufficient Design
* Just Enough Design Initially (JEDI)
* Personas (Activity)
* Story Mapping introduction

## Day 2 Afternoon

### 00:00-1:00 Story Mapping

* Activity

### 01:00-03:30 Specification by Example

* Concrete vs. abstract descriptions (Activity)
* Reading Gherkin (Activity)
* What is Specification by Example?
* Writing Gherkin (Activity)
* Value of Specification by Example
* Demonstration of test automation (World's Smallest Bookstore)

### 03:30-04:00 Day 2 Retrospective

* Facilitated retrospective

## Day 3 Morning

### 00:00-00:30 Git Introduction

* Basic interaction with the local git repository
* Basic interaction with the remote git repository
* Semantic versioning

### 00:30-03:00 Working on the World's Smallest Bookstore

* Pomodoro-based randori or pair develpment of User Stories

## Day 3 Afternoon

### 00:00-03:30 Working on the World's Smallest Bookstore

* Continue working User Stories for the World's Smallest Bookstore

### 03:30-04:00 The Rest of the Story

* Platform-as-a-Service (PaaS)
* Continuous Delivery Again

### 04:00-04:15 Section 7: Name Tent Contest

* Vote on the name tents and award prizes

### 04:15-04:30 Workshop Retrospective

* Were expectations met?
* Did you learn anything useful?
* What can we do to improve the workshop?

# Bookstore Build and Test

## First, the bad news

* Cucumber-JVM and Spring Boot will not work together.
* Gradle unit test reporting and JUnit5 will not work together
* MockitoJUnitRunner does not work with JUnit5 (for example, if you write @BeforeEach it will compile but have no effect.
* Spring Boot has JUnit4 baked in, so you may experience classpath-related problems when using JUnit5 even if your POM looks okay.

## Build with Gradle

|  |  |
| --- | --- |
| **gradle test** | build and run unit tests |
| **gradle integrationTest** | build and run integration tests |
| **gradle bootRepackage** | create executable jar |
| **gradle clean** | delete generated files |
| **gradle bootRun** | execute the application with gradle |
| **gradle tasks** | list available gradle tasks |
| **java -jar build/libs/bookstore-1.0.0.jar** | execute the application as packaged |
|  |  |

Notes:

* Gradle's output directory is build.
* Gradle's unit test results HTML formatter doesn't work with JUnit 5. You won't get unit test reports from Gradle.

## Build with Maven

*Known problems with this project (not the tools themselves)*

* mvn integration-test goal does not find any tests.

|  |  |
| --- | --- |
| **mvn test** | build and run unit tests (not working) |
| **mvn integration-test** | build and run integration tests (not working) |
| **mvn clean** | delete generated files |
| **mvn clean package spring-boot:repackage** | create executable jar |
| **mvn spring-boot:run** | execute the application with maven |
| **mvn fr.jcgay.maven.plugins:buildplan-maven-plugin:list** | list phases and goals in execution order |
| **mvn fr.jcgay.maven.plugins:buildplan-maven-plugin:list-phase** | list goals per phase |
| **mvn fr.jcgay.maven.plugins:buildplan-maven-plugin:list-plugin** | list goals by plugin |
| **mvn surefire-report:report** | generate unit test report |
| **target/java -jar bookstore.1.0.0.jar** | execute the application as packaged |
|  |  |

Notes:

* Maven's output directory is *target*.
* Surefire 2.20+ and Failsafe 2.20+ do not work properly with Junit5. Use 2.19.1 *and* specify the JUnit Platform Surefire Provider dependency (see pom.xml).

## Ruby Cucumber

You can use the Ruby version of Cucumber, as Cucumber-JVM is not compatible with Spring Boot, and JBehave is not compatible with JUnit5. The cukes are included with the Bookstore project on Github.

Assumes you have Ruby 2.3.1 or later installed. The Ruby installation is not included with the Bookstore project. Once you have Ruby on your system and you've cloned the Bookstore project repo, you can run these commands from the project root directory to install the necessary gems:

**gem install bundler**

**bundle install --path vendor/bundle**

*Run the Cukes*

Start the application server (see above). Then run:

**bundle exec rake**

## Bookstore UI application (separate)

This workshop focuses on the RESTful service application, but there is a lightweight web-based user interface application that can talk to the Bookstore service. If it would be helpful for you to use it in this workshop, or if you would like to practice "full stack" development when completing the User Stories for the workshop, you can find the UI application here:

* https://github.com/neopragma/bookstore-ui

It's a "single-page" webapp built on Node.js and the Express framework.

*Setup*

Clone the repo.

**rpm install**

*Run the server*

**rpm start**

# Resources

## Test-Driven Development

* Empirical Studies Show Test-Driven Development Improves Quality. Chris Sims. https://www.infoq.com/news/2009/03/TDD-Improves-Quality
* Introduction to Test-Driven Development. Scott Ambler. http://agiledata.org/essays/tdd.html
* The Transformation Priority Premise. Bob Martin. https://8thlight.com/blog/uncle-bob/2013/05/27/TheTransformationPriorityPremise.html
* TDD Guided by ZOMBIES. James Grenning http://blog.wingman-sw.com/archives/677
* Detroit and London Schools of TDD. Justin Searls. https://github.com/testdouble/contributing-tests/wiki/London-school-TDD

## Refactoring

* What is refactoring? C2 Wiki. http://wiki.c2.com/?WhatIsRefactoring
* Refactorings Catalog. Martin Fowler. https://refactoring.com/catalog/
* Refactoring to Patterns Catalog. Joshua Kerievsky. https://industriallogic.com/xp/refactoring/catalog.html

## Trunk-Based Development

* Much information at trunkbaseddevelopment.com. https://trunkbaseddevelopment.com
* From Git Flow to Trunk-Based Development. Robert Ecker. https://team-coder.com/from-git-flow-to-trunk-based-development/

## Semantic Versioning

* Semantic Versioning. https://semver.org

## Infrastructure as Code

* Infrastructure as Code. Martin Fowler. https://martinfowler.com/bliki/InfrastructureAsCode.html
* Configuration Drift. Kief Morris. http://kief.com/configuration-drift.html
* Moving to the Phoenix Server Pattern. Ama Asare. https://www.thoughtworks.com/insights/blog/moving-to-phoenix-server-pattern-introduction
* Canary Deployments. Octopus Deploy. https://octopus.com/docs/deployment-patterns/canary-deployments
* Blue-Green Deployment. Cloud Foundry. https://docs.cloudfoundry.org/devguide/deploy-apps/blue-green.html
* Continuous Delivery. Continuousdelivery.com. https://continuousdelivery.com

## Microservices

* Microservices Architecture. Chris Richardson. http://microservices.io/patterns/microservices.html
* The Twelve-Factor App. Heroku. https://12factor.net
* Java walkthrough to build a microservice and CI/CD pipeline. Dave Nicolette. https://github.com/neopragma/javahellolib

## Personas

* Personas. Interaction Design Foundation. https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them
* Personas. Hubspot. https://knowledge.hubspot.com/contacts-user-guide-v2/how-to-create-personas
* Personas. Shlomo Goltz. https://www.smashingmagazine.com/2014/08/a-closer-look-at-personas-part-1/

## Feature Mapping / Story Mapping

* Story Mapping. Sunit Parekh. https://www.thoughtworks.com/insights/blog/story-mapping-visual-way-building-product-backlog
* How Story Mapping Complements Agile Development. Matt Heusser. https://www.cio.com/article/2377375/agile-development/how-story-mapping-complements-agile-development.html

## Specification by Example

## Cucumber

* Cucumber documentation