Welcome to JUnit 5

Billy Korando
Developer Advocate - IBM
@BillyKorando
william.korando@ibm.com

Subscribe to the Java Newsletter

https://developer.ibm.com/newsletters/java/

For your Spring & JakartaEE needs

https://cloud.ibm.com/docs/java

think 2019

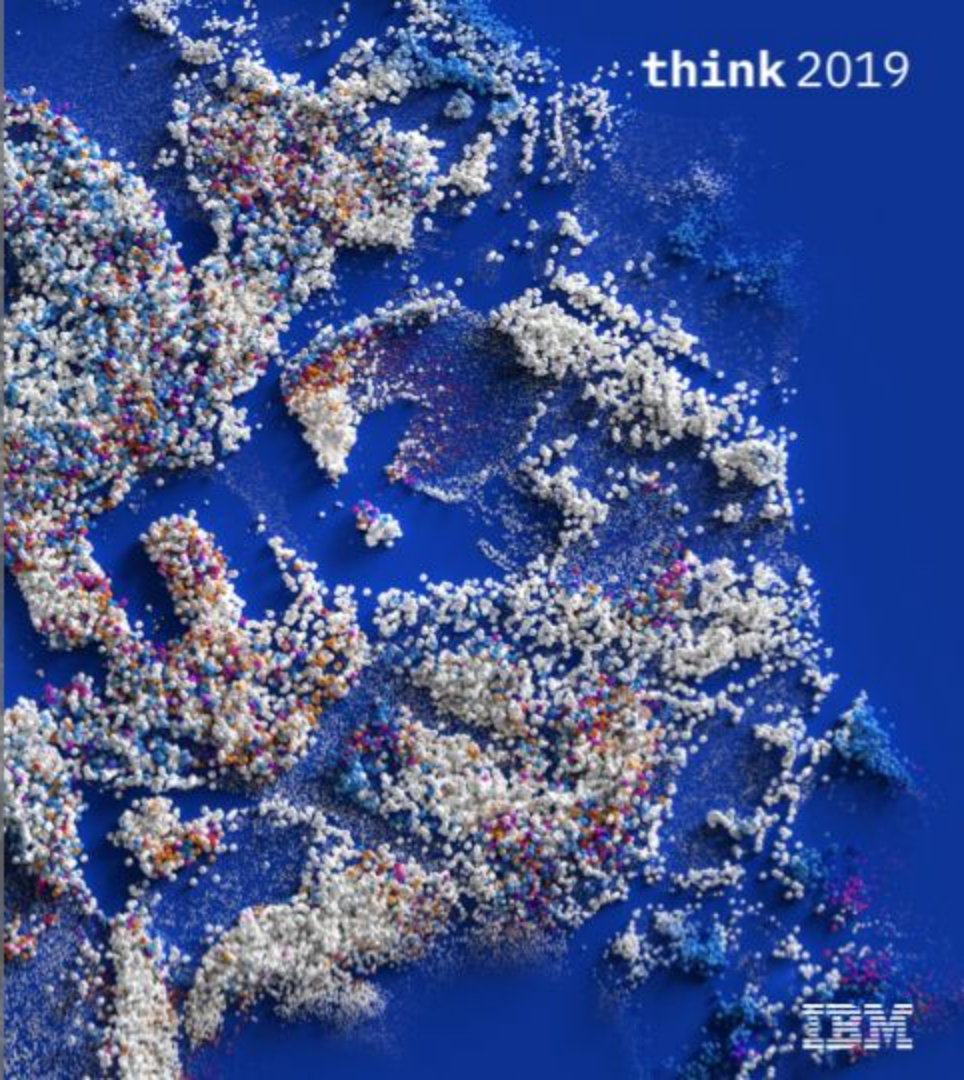
Developer Night

Think. Code. Share.

An evening full of code, live demos and tech talks

June 13, Amsterdam

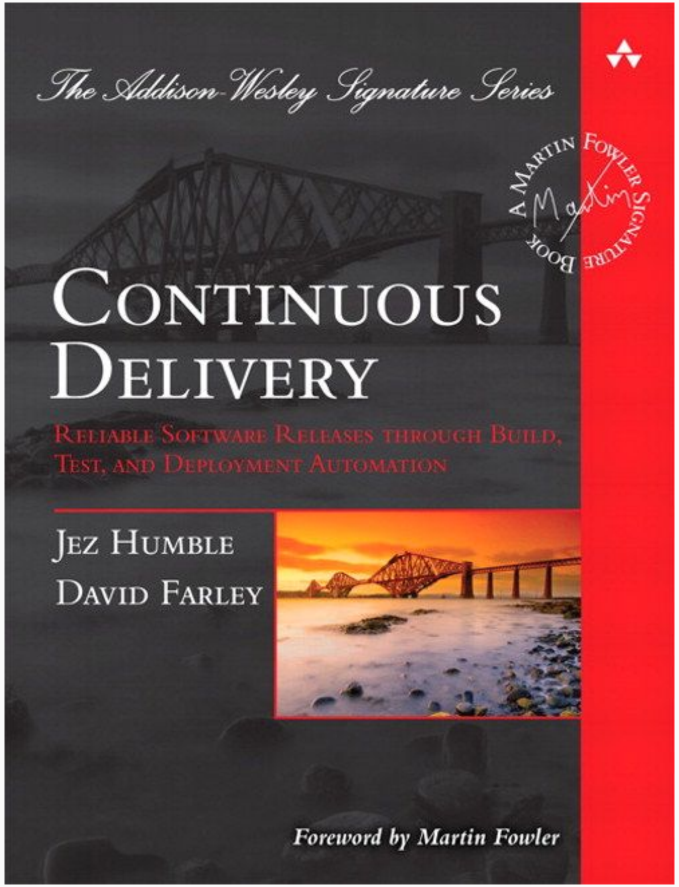
Register now: ibm.biz/DevNight2019



IBM

https://billykorando.com/category/junit-5/

- Get you excited for automated testing
- Get you excited for JUnit 5


The Addison-Wesley Signature Series

A Martin Fowler Signature Book

Continuous Delivery

Reliable Software Releases through Build, Test, and Deployment Automation

Jez Humble

David Farley

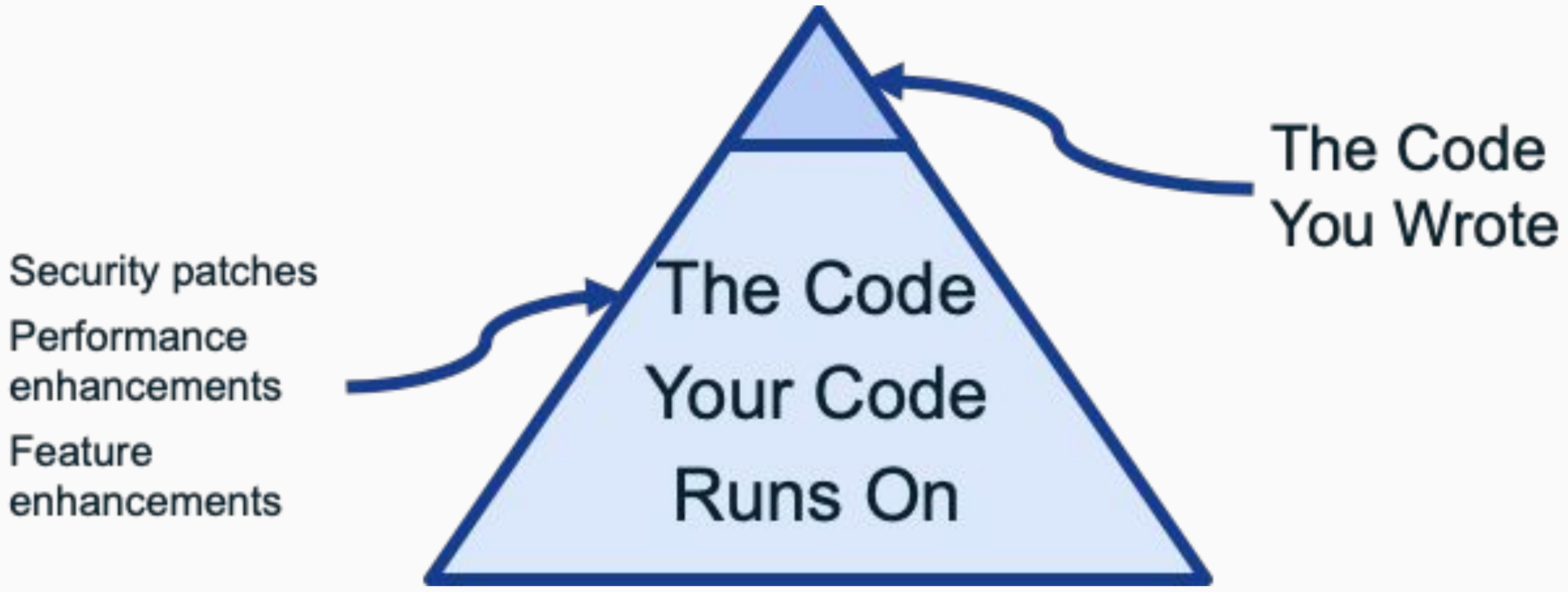
Foreword by Martin Fowler

@BillyKorando

Quality automated testing gives you

- Confidence you are fixing what you set out to fix
- Confidence you are not introducing a new bug

Without automated tests you are building legacy.


The Code You Wrote
Security patches
Performance enhancements
Feature enhancements
The Code Your Code Runs On

Goals of JUnit 5

- Modernize
    - Java 8 baseline
- Extensible
- Backwards compatible

Migration Requirements

- Java 8+
- IntelliJ 2016.2+
- Eclipse 4.7.1a+
- Netbeans 11+
- Gradle 4.6+
- Surefire 2.19.1+ (2.22.0)

Writing Tests with Jupiter

Language Changes

org.junit.Test -> org.junit.jupiter.api.Test

org.junit.Assert.* -> org.junit.jupiter.api.Assertions.*

@Ignore -> @Disabled

@RunWith/@Rule/@ClassRule -> @ExtendWith

@BeforeClass/@Before -> @BeforeAll/@BeforeEach

@AfterClass/@After -> @AfterAll/@AfterEach

Extensions Model

Extension Model

@Rule/@ClassRule/@RunWith -> rolled into the new extension model

Extensions can be registered:

- Declaratively with @ExtendWith
- Programmatically with @RegisterExtension
- Automatically using Java Service Loader

Extension Model Advantages

Can declare multiple extensions in a single class

A lot more control over execution order of extensions

Single API for writing extensions instead of bifurcated Runner and Rule APIs

Dependency Injection

Dependency Injection

Can pass values from container in constructors and methods

TestInfo is always available and can be passed into a test case

Parameterized, Repeated, & Dynamic Tests

Parameterized Tests

@RunWith(Parameterized.class) -> @ParameterizedTest

Put at the individual method level

Various sources for parameterized test; ValuesSource, MethodSource, EnumSource, CsvSource, CsvFileSource

New in 5.2, can use argument aggregator to make CsvSource easier to use

Repeated Tests

Easily execute tests multiple times

@RepeatedTest(value = {repititions}, name={customized test name})

RepititionInfo can be passed in as method argument to get info on current and total repititions

Dynamic Tests

Generate tests dynamically

@TestFactory

Method returns a Collection, Stream, or Iterable of DynamicTest or DynamicContainer

Test Interfaces

Test Interfaces

Tests can be declared on default methods of interfaces

Test class that implements the interface executes those tests

Similar to how abstract classes worked in JUnit 4, but can implement multiple interfaces

Test Interfaces

Test interfaces are great for:

- Making testing a pattern easier
- Encouraging developers to follow pattern guidelines
- When a pattern has distinct portions and may only be partially implemented

Test Interfaces

Test Interfaces in practice:

- Testing RESTful API can be kinda difficult
- REST has a very specific pattern
- Not every RESTful API will need to implement all parts

RESTful api:

`/api/v1/resources` < GET returns all of resource, POST here to add new resource

`/api/v1/resources/{id}` < GET returns specific resource, 404 if not found, PUTs & DELETEs implmented here

`/api/v1/resources/search-by/{typeOfSearch}` < Queries should be prefixed by “search-by” in url

Invalid client data should return 400 with “error message” in response body.

Test Interfaces

```
public interface GetResourceEndpointTest<T, I> extends EndpointTest {
      I getExistingResource();
      I getNonExistingResoruce();
      T foundResource();
      String getFoundResourceJsonContent();
      OngoingStubbing<T> mockExistingBehavior();
      OngoingStubbing<List<T>> mockFindAllResourcesBehavior();
      OngoingStubbing<T> mockNonExistingBehavior();

      @Test
      default void testExistingResource() throws Exception {
            mockExistingBehavior().thenReturn(foundResource());
            getMockMvc().perform(get(baseEndpoint() + "/" +
getExistingResource())).andExpect(status().isOk())
                        .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))
                        .andExpect(content().json(getFoundResourceJsonContent()));
      }

      ...
```

Test Interfaces

```
        @Test
        default void testGetAllResources() throws Exception {
                mockFindAllResourcesBehavior().thenReturn(Arrays.asList(foundResource()));
                getMockMvc().perform(get(baseEndpoint())).andExpect(status().isOk())
                                .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))
                                .andExpect(content().json("[" + getFoundResourceJsonContent() + "]"));
        }

        @Test
        default void testNonExistingResource() throws Exception {
                mockNonExistingBehavior().thenReturn(null);
                getMockMvc().perform(get(baseEndpoint() + "/" +
getNonExistingResoruce())).andExpect(status().isNotFound());
        }
}
```

Test Interfaces

REST API needs to implement GET and search only:
public class TestResourcesController **implements** GetResourceEndpointTest, SearchEndpointTest

REST API needs to implement POST and search only:
public class TestResourcesController **implements** PostResourceEndpointTest

REST API needs to implement everything:
public class TestResourcesController **implements** GetResourceEndpointTest, SearchEndpointTest, PostResourceEndpointTest

Disabling & Filtering

Disabling Tests

@Ignore -> @Disabled

Test can be disabled based on conditions; OS, JDK version, custom logic

Like @Ignore, @Disabled test show as “skipped” in test report

Filtering Tests

Tests can be tagged with @Tag or @Tags

Filtered tests do not show up in test report

Parallel Test Execution

Parallel Test Execution

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.0</version>
    <configuration>
        <properties>
            <configurationParameters>
                junit.jupiter.execution.parallel.enabled=true
                junit.jupiter.execution.parallel.config.dynamic.factor=1
                junit.jupiter.execution.parallel.mode.default = concurrent
            </configurationParameters>
        </properties>
    </configuration>
</plugin>
```

Parallel Test Execution

```
@Execution(CONCURRENT)
public class TestParallelTests {
}
```

```
@Execution(SAME_THREAD)
public class TestParallelTests {
}
```

Parallel Test Execution

```
public class TestParallelTests {
	@Test
	@ResourceLock(value = "RESOURCE_ID", mode = ResourceAccessMode.READ_WRITE)
	public void testA() {
		//Will not be executed at same time as any other test with same resource
	}
	@Test
	@ResourceLock(value = "RESOURCE_ID", mode = ResourceAccessMode.READ_WRITE)
	public void testB() {
		//Will not be executed at same time as any other test with same resource
	}
	@Test
	@ResourceLock(value = "RESOURCE_ID", mode = ResourceAccessMode.READ)
	public void testC() {
		//May be executed at same time as test with same resource, as long it’s not READ_WRITE
	}
	@Test
	@ResourceLock(value = "RESOURCE_ID", mode = ResourceAccessMode.READ)
	public void testD() {
		//May be executed at same time as test with same resource, as long it’s not READ_WRITE
	}
}
```

Ordering Test & Extension Execution

Ordering Test Execution

```
@TestMethodOrder(OrderAnnotation.class)
public class TestClass{
     @Test
     @Order(1)
     public void testExecuteFirst() {
           //ExecutedFirst
     }
     @Test
     @Order(2)
     public void testExecuteSecond() {
           //ExecutedSecond
     }
     @Test
     public void testExecuteSometimeLater() {
           //Executed at some point after ordered tests
     }
     @Test
     public void testExecuteSometimeLaterAswell() {
           //Executed at some point after ordered tests
     }
}
```

Ordering Test Execution

```
@TestMethodOrder(Alphanumeric.class)
public class TestClass{
     @Test
     public void testA() {
           //ExecutedFirst
     }
     @Test
     public void testB() {
           //ExecutedSecond
     }
}
```

Ordering Test Execution

```
@TestMethodOrder(Random.class)
public class TestClass{
     @Test
     public void testRandomOne() {
           //...
     }
     @Test
     public void testRandomTwo() {
           //...
     }
}
<plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.0</version>
      <configuration>
           <properties>
                <configurationParameters>
                     junit.jupiter.execution.order.random.seed=<seed>
                </configurationParameters>
           </properties>
      </configuration>
</plugin>
```

Ordering Programmatically Registered Extension Execution

```
public class TestClass{

      @Order(1)
      @RegisterExtension
      AnExtension anExtension = new AnExtension();//Executed first
      @Order(2)
      @RegisterExtension
      SomeExtension someExtension = new SomeExtension();//Executed second
      @RegisterExtension
      AnotherExtension anotherExtension = new AnotherExtension();//Executed after ordered extensions
      @RegisterExtension
      YetAnotherExtension yetAnotherExtension = new YetAnotherExtension();//Executed after ordered 
extensions

      //...
}
```

That’s not all…

What I didn’t cover

Easily use other assertion frameworks like AssertJ and Hamcrest

Nested tests

Assumptions

ConsoleLauncher

Extendability

And more!

Check out: https://junit.org/junit5/docs/current/user-guide/

Q&A

@BillyKorando

william.korando@ibm.com

IBM Cloud Sign-up: https://ibm.biz/BdzdZs

Slides: http://ibm.biz/welcome-to-junit5

Code: https://github.com/wkorando/WelcomeToJunit5