adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE 28th – 30th SEPTEMBER 2020

Behaviour vs Implementation Testing in AEM Daniel Strmečki, ecx.io, part of IBM iX



How, Why, When and What to Test?



How to test?

This question has already been answered

- <u>https://junit.org/junit5/</u>
- https://site.mockito.org/
- https://wcm.io/testing/aem-mock/
- In this session
 - We will not talk about tools and libraries
 - Instead we will focus on our testing strategy



Why test?

Writing tests does take time, but... It makes us faster in the long run It improves quality, design and maintainability



When to test?

- Whenever you write business logic
- Writing test suites just to have high code coverage leads to test that are
 - brittle (easy to break)
 - hard to read and
 - do not support easy refactoring



What to test?

- Testing strategy guidelines on what to test
- Some basics
 - Test requirements behaviour or functionality over implementation details
 - Test the business logic your write, not the framework or libraries you use
 - Test the APIs you expose, not internal details



Unit vs Integration Testing?



The basic difference

- Unit tests are written by developers to test a single unit of code (service or model) we developed
- Integration testing is a type of testing to check if different components, services or modules are working together towards a common goal (requirement)



How to decide?

- In general, prefer integration testing
 - Focus on behaviour and requirements, not internal implementation details
 - Mocking is hard work and requires maintenance
 - Takes less time to achieve the same coverage
- Unit tests are still welcome to test complicated business logic of specific units



What's Specific in AEM?



- Integration tests can be written and run using mocks, not only running instances as recommended by Adobe
- JUnit, Mockito and AEM Mocks are mature libraries for writing **integration** and unit tests
 Similar approach to Spring Mock MVC



What (not) to test in AEM?

- Useless testing
 - Testing (or mocking) a Utility class
 - Testing a data class or a Sling Model, that contains no business logic
- In AEM, integration tests should usually be written for Sling Models, REST Services and Servlets



Time for Live Demo!



Key Takeaways!



- Invest the needed time to describe what you are testing (given-when-then)
- Write business logic in services and create mocks based on their interfaces
- Separate mocked classes from test classes to make tests cleaner



- Test the business logic you write, not the framework or libraries you are using
- Only test the things you expose, your public API, nothing else
- Secure time for writing tests in advance, it needs to be planned



Let's end with a quote

Managers may defend the schedule and requirements with passion; but that's their job. It's our job to defend the code quality with equal passion.

By Robert C. Martin



- Examples available on GitHub
 - https://github.com/dstr89/aem-unit-vs-integration
- Test behaviour, not implementation
 - https://dev.to/mkovacek/test-behaviour-notimplementation-3g2j
 - by Matija Kovaček



The End



 Java, Web, AEM, software craftsmanship, testing, reusability, architecture, knowledge sharing...

<u>https://www.linkedin.com/in/</u> <u>strmecki/</u>

daniel.strmecki@ecx.io



Thank you.