جافا - الوحدة العاشرة: الاختبار باستخدام JUnit والخطوات التالية

قد يتعطل برنامج يعمل اليوم غدا، عندما يغير أحد سطرا واحدا. والاختبارات الآلية برامج صغيرة تفحص كودنا وتخبرنا فورا عندما ينكسر شيء. وتبني أربع خطوات اختبارا موثوقا: التهيئة، والتنفيذ، والتحقق، والأتمتة. وفي هذه الوحدة تستخدمها مايا، مسؤولة الجودة في فريق الفوترة بشركة اتصالات، لتحمي حساب فاتورة، وتجيب كل خطوة عن سؤال عملي واحد.

1. التهيئة

يفحص الاختبار الجزئي unit test قطعة صغيرة من الكود، كطريقة، بمدخلات معروفة ونتيجة متوقعة. و JUnit إطار اختبار مجاني لجافا يجد هذه الاختبارات ويشغلها. والاختبار طريقة عادية موسومة بالتعليق التوضيحي Test. ويتبع كل اختبار الشكل نفسه: تهيئة، ثم تنفيذ، ثم تحقق. وفي مرحلة التهيئة نجهز كل ما يحتاجه الاختبار، مثل فاتورة فيها سطران نستطيع حساب إجماليها باليد. والطريقة التي تحمل التعليق BeforeEach تعمل قبل كل اختبار وتبني كائنات جديدة، فلا يعتمد اختبار على آخر. وسؤال هذه الخطوة هو: أي موقف نجهز، وماذا نعرف مسبقا عن الإجابة الصحيحة؟

مثال: تختبر مختبرة في مصرف الطريقة التي تضيف الفائدة السنوية إلى حساب توفير. فتهيئ الموقف أولا: حسابا برصيد ألف دولار ونسبة خمسة في المئة. وقبل أن يعمل الاختبار تحسب الفائدة الصحيحة على الورق وتكتب خمسين دولارا. هذه هي التهيئة: موقف بداية معروف وإجابة صحيحة معروفة.

في العمل: تبدأ مايا بأبسط موقف وتعطي الاختبار اسما يقول ما يفحصه: `@Test void emptyInvoiceHasZeroTotal()`. وفي التهيئة تنشئ فاتورة فارغة. ويهيئ اختبار ثان فاتورة بسطرين، وتحسب الإجمالي المتوقع باليد أولا. وتعطي طريقة BeforeEach كل اختبار فاتورة جديدة.

2. التنفيذ

في مرحلة التنفيذ نستدعي الشيء الواحد الذي نختبره، وهو عادة استدعاء طريقة واحدة. والاقتصار على استدعاء واحد يجعل موضع الفشل سهل التحديد. وتغطي الاختبارات الجيدة الحالة العادية والحواف وحالات الخطأ: القيمة التي تقل عن الحد قليلا، والتي تساويه تماما، والتي تزيد عليه قليلا، ومدخلا فارغا، ومدخلا يجب رفضه. ويفحص كل اختبار فكرة واحدة وله اسم يصف السلوك. ويجب أن تكون الاختبارات مستقلة وسريعة وقابلة للتكرار، أي تعطي الإجابة نفسها في كل مرة وبأي ترتيب. وسؤال هذه الخطوة هو: أي إجراء واحد نختبر، وأي حالات حدية تهم؟

مثال: يختبر فريق شحن الطريقة التي تحسب رسوم التوصيل، وترتفع الرسوم عندما يتجاوز الطرد حد الوزن. ومرحلة التنفيذ هي الاستدعاء الواحد نفسه في كل مرة، مع ثلاثة طرود مختلفة: طرد يقل عن الحد قليلا، وطرد يساويه تماما، وطرد يزيد عليه قليلا. وتبين ثلاثة اختبارات صغيرة أين تتغير الرسوم بالضبط. ويكتشف الخطأ عند الحد في اليوم نفسه، وهذا يبين لماذا تهم الحالات الحدية.

في العمل: مرحلة التنفيذ عند مايا سطر واحد: تطلب من الفاتورة إجماليها. وللخصم تكتب ثلاثة اختبارات، بإجمالي يقل عن حد الولاء قليلا، وآخر يساويه تماما، وآخر يزيد عليه قليلا. ويستدعي اختبار آخر الطريقة بكمية سالبة، ويجب أن ترفض. ولكل اختبار اسم مثل discountStartsAtTheThreshold.

3. التحقق

تقارن مرحلة التحقق ما حدث بما توقعناه. وتأخذ الطريقة assertEquals القيمة المتوقعة أولا والقيمة الفعلية ثانيا، وتفحص assertTrue شرطا. وللتأكد من أن طريقة ترفض مدخلا سيئا، تسمي assertThrows الاستثناء المتوقع وتشغل الكود الذي يجب أن يرميه. وعندما يفشل تحقق، يبلغ JUnit بالقيمة المتوقعة والقيمة الفعلية واسم الاختبار، فنبدأ بدليل دقيق. ومع BigDecimal تعتبر equals مبلغا بخانة عشرية واحدة والمبلغ نفسه بخانتين مختلفين، لذلك نقارن

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)

بالطريقة compareTo أو نعطي القيمتين العدد نفسه من الخانات. والاختبار الذي لا يستطيع الفشل لا يثبت شيئا، فنتأكد من أن كل اختبار يفشل عندما يكون الكود خاطئا. وسؤال هذه الخطوة هو: ماذا يجب أن يتحقق بالضبط عندما يكون الكود صحيحا؟

مثال: تتوقع مختبرة رواتب أن يكون صافي راتب موظف ألفي دولار، لكن الاختبار يبلغ بقيمة فعلية هي ألف وتسعمئة. ويظهر التقرير القيمة المتوقعة والقيمة الفعلية واسم الاختبار. ويفتح المطور تلك الطريقة ويجد خصما مطبقا مرتين في دقائق. هذه هي قيمة التحقق: إخفاق معه دليل دقيق.

في العمل: تؤكد مايا أن إجمالي الفاتورة الفارغة يساوي الصفر، وتستخدم compareTo بسبب قاعدة عدد الخانات العشرية. وللكمية السالبة تستخدم assertThrows مع IllegalArgumentException. وتكسر قاعدة الخصم عمدا، فيفشل اختبار الحد ويظهر الإجمالي المتوقع والفعلي. وتصلح القاعدة، فتنجح الاختبارات كلها من جديد.

4. الأتمتة

تنفع الاختبارات أكثر عندما تعمل باستمرار. وتستطيع بيئات التطوير IDE تشغيل كل الاختبارات بنقرة واحدة، وتشغلها أدوات البناء مثل Maven و Gradle خطوة معتادة في كل بناء. وتنزل هذه الأدوات أيضا مكتبات مثل JUnit وتترجم التطبيق وتغلفه. ويشغل الفريق عادة الاختبارات تلقائيا عند كل تغيير، فتكتشف القاعدة المكسورة قبل الإصدار. والنتيجة الخضراء لا تثبت غياب الأخطاء، لكن النتيجة الحمراء تحذير دقيق. ولنمضي في التعلم نستطيع التعمق في المجموعات والتدفقات، وقواعد البيانات عبر JDBC، وأطر مثل Spring لخدمات الويب. وسؤال هذه الخطوة هو: متى تعمل الاختبارات، ومن يرى النتيجة؟

مثال: يستخدم فريق برمجيات أداة بناء تشغل كل الاختبارات كلما اقترح مطور تغييرا. وذات عصر يغير مطور قاعدة ضريبة، فيتحول اختبار واحد إلى الأحمر. وتمنع الأداة التغيير وتنبه كاتبه، فيصلحه قبل أن يراه أحد غيره. هذه هي الأتمتة: تعمل الاختبارات في كل مرة، فقلما يصل الكود المكسور إلى العملاء.

في العمل: تطلب مايا من الفريق تشغيل اختبارات الفوترة بأداة البناء عند كل تغيير. وفي يوم الجمعة التالي يغير مطور قاعدة الخصم بسطر واحد. فيتحول البناء إلى الأحمر في دقائق، ويصلحه الفريق قبل الإصدار. ويرى العملاء يوم الاثنين فواتيرهم الصحيحة.

الخاتمة

في الخلاصة، الاختبار الجيد يهيئ موقفا معروفا، وينفذ باستدعاء واحد، ويتحقق من النتيجة المتوقعة، ويعمل تلقائيا. ويجد JUnit الطرق الموسومة بالتعليق Test ويبلغ بإخفاقات واضحة، وتقارن مبالغ BigDecimal بالطريقة compareTo. وانتقلنا في هذه السلسلة من أول برنامج إلى تطبيق أعمال مختبر: قيم وقرارات وطرق ومجموعات وأصناف وواجهات واستثناءات وملفات واختبارات. وقاعدة الفاتورة عند مايا محمية الآن، ويظهر التغيير السيئ بناء أحمر في دقائق. والخطوات التالية هي التدرب على مسائل حقيقية، والمزيد من المكتبة القياسية، وأطر خدمات الويب. ومن العادات الجيدة كتابة الاختبار أولا عندما يبلغ عن خطأ، والاحتفاظ به إلى الأبد.

المصادر

- JUnit: دليل مستخدم JUnit (التعليقات التوضيحية والتحققات وبنية الاختبار) – https://junit.org/junit5/docs/current/user-guide/
- أوراكل: دروس جافا، التعليقات التوضيحية – https://docs.oracle.com/javase/tutorial/java/annotations/index.html
- مشروع Apache Maven: مقدمة إلى دورة حياة البناء – https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
- dev.java: تعلّم جافا – https://dev.java/learn/
- JUnit: كتابة الاختبارات - أصناف الاختبار وطرقه – https://docs.junit.org/current/writing-tests/test-classes-and-methods.html
- JUnit: كتابة الاختبارات - التحققات – https://docs.junit.org/current/writing-tests/assertions.html
- Oracle: واجهة BigDecimal البرمجية – https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/math/BigDecimal.html
- Java علامة تجارية لشركة Oracle و/أو الشركات التابعة لها. JUnit و Maven و Gradle مشاريع تعود إلى أصحابها. هذا الفيديو مادة تدريبية مستقلة وغير تابعة لـ Oracle.

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)