Prof. Dr. Stefan Betermieux
Fakultät Informatik
Hochschule Furtwangen

Open Source-basierte Softwareentwicklung

Praktikum 7 - Integrationstests

Szenario

In dieser Woche lernen Sie die Vorgehensweise, um mit dem Test-Werkzeug »JUnit 4« und der Bibliothek »EasyMock« Integrationstests zu schreiben. Zu diesem Zweck verwenden wir das in den letzten Praktika angelegten Maven-Projekt.

Arbeitsschritte

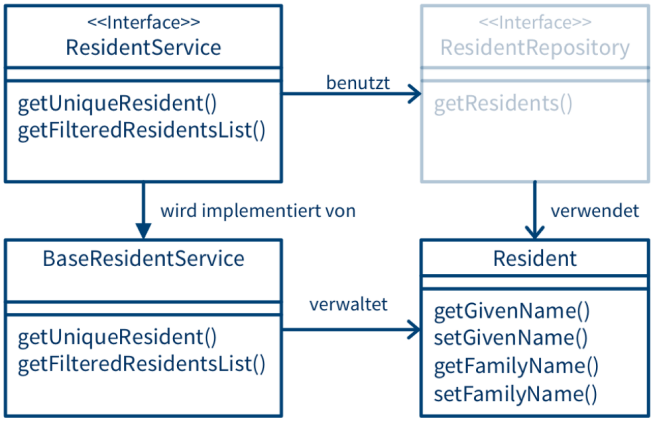
Schritt 1

Öffnen Sie Ihr angelegtes Maven-Projekt in Eclipse. Vergewissern Sie sich, dass es von Eclipse als…

- …Maven-Projekt erkannt wird (kleines blaues M links oben im Projekt-Icon)
- …Java-Projekt erkannt wird (kleines blaues J rechts oben im Projekt-Icon)
- …Git-Projekt erkannt wird (kleine gelbe Tonne rechts unten im Projekt-Icon)

Schritt 2

Laden Sie von http://webtech.informatik.hs-furtwangen.de/oss/07-Integrationstests.zip die zu testenden Dateien herunter. Integrieren Sie die Dateien in Ihr Projekt unter src/main/java. Passen Sie die Package-Struktur bei Bedarf an. Eclipse zeigt keine Compiler-Fehler an.



Die heruntergeladenen Dateien sind Teil eines Programms zur Verwaltung von Einwohnermeldedaten in Kommunen. In der Klasse »`Resident`« wird der Datensatz eines gemeldeten Einwohners gespeichert (Vorname, Nachname, Geburtsdatum, etc…). Das Interface »`ResidentRepository`« dient als Schnittstelle zu einer Datenbank und liefert alle gemeldeten Einwohner zurück (es gibt noch keine Implementierung zu dieser Schnittstelle). Die Schnittstelle »`ResidentService`« bietet öffentliche Funktionen, die von außen aufgerufen werden können:

`getFilteredResidentsList(Resident filterResident)` → es werden alle Datensätze im ResidentRepository gesucht, die auf den im Parameter gelieferten filterResident passen. Leere Felder im filterResident werden bei der Suche ignoriert, das »*« dient als Wildcard (He* als Vorname liefert also alle Helga, Herbert, Hermann, etc…)

`getUniqueResident(Resident filterResident)` → es wird überprüft, ob der im Parameter übergebene filterResident im ResidentRepository auftaucht. Im filterResident dürfen keine Wildcards verwendet werden und das Ergebnis muss eindeutig sein.

Schritt 3

Sorgen Sie dafür, dass Sie (falls noch nicht geschehen) JUnit4 in Ihrem Projekt verwenden. JUnit3 wird in einigen Fällen beim Erstellen eines Projekts in der pom.xml eingetragen, dort müssen Sie die Versionsnummer z.B. auf 4.12 ändern.

Schritt 4

Schreiben Sie einen Integrationstest, indem Sie ein Stub-Objekt für das ResidentRepository erstellen und Sie mindestens drei verschiedene Testfälle für getFilteredResidentsList() und drei verschiedene Testfälle für getUniqueResident() schreiben.

Schritt 5

Erzeugen Sie Testmetriken, indem Sie JaCoCo in die Maven pom.xml integrieren (ist im Augenblick noch an zwei Stellen — reporting/plugins und build/plugins — auskommentiert) und »mvn site« aufrufen. Sorgen Sie dafür, dass Ihre Tests für die Klasse BaseResidentService eine C_0 Anweisungs-überdeckung (Line Coverage) größer 80% erreicht. Integrieren Sie folgenden Bereich in Ihre pom.xml:

Schritt 6

Schreiben Sie einen weiteren Integrationstest, indem Sie das Stub-Objekt durch einen Mock austauschen. Verwenden Sie dafür EasyMock, welches Sie zuerst als Abhängigkeit der Maven pom.xml hinzufügen (ist im Augenblick noch als dependency auskommentiert).

Schritt 7

Laden Sie Ihre Änderungen in das entfernte Git-Repository hoch.