O/ZB Webanwendung

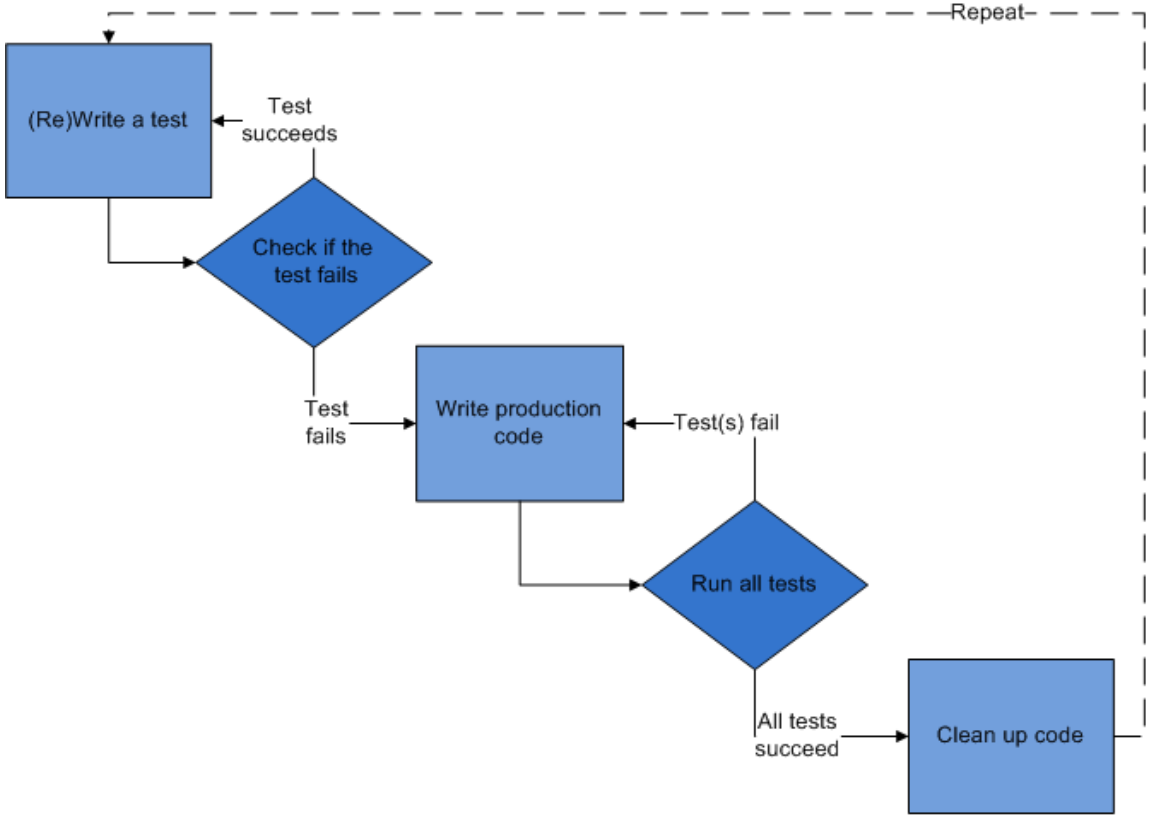
TDD - Zwischenstand

Übersicht

- Was ist Testdriven Development?
- Unit Test
- Controller Test
- Datenbankmodell, Umsetzung in MySQL
- Offene Fragen, Issues

Testdriven Development

- Hauptmerkmal
  - Software-Tests werden konsequent *vor* den zu testenden Komponenten erstellt



Testdriven Development

Ziele und Umsetzung

- Das Hauptmerkmal kann in diesem Projekt erst wieder bei neuen Funktionalitäten angewandt werden
- Mit Hilfe der Tests soll der bisherige Zustand, der historisch gewachsenen Webanwendung, aufgenommen werden
  - Konsistenz: Ruby Anwendung <-> Datenbank
  - Fachlich und technische Validierung der einzelnen Modelle (Attribute, Beziehungen und Funktionen) und Controller (Logik, Controller übergreifende Funktionalitäten)

Unit Test

Ziele

- Es soll sichergestellt werden, dass nur fachlich und technisch korrekte Objekte des Models erzeugt werden können.
- Sicherstellung der korrekten Funktionalität der im Model enthaltenen Funktionen

Unit Test

Voraussetzung

- Fachlich und technisch korrekter Testdatensatz
- Vollständige Implementierung des Models:
  - Vollständigkeit der fachlichen und technischen Attribute, die mit der Datenbank übereinstimmen
  - Validierungen der vorhandenen Attribute (z.B. Beziehungen, Datenformate, Wertebereiche, usw.)
  - Historierungsfunktionalität bereits implementiert, wo diese notwendig ist

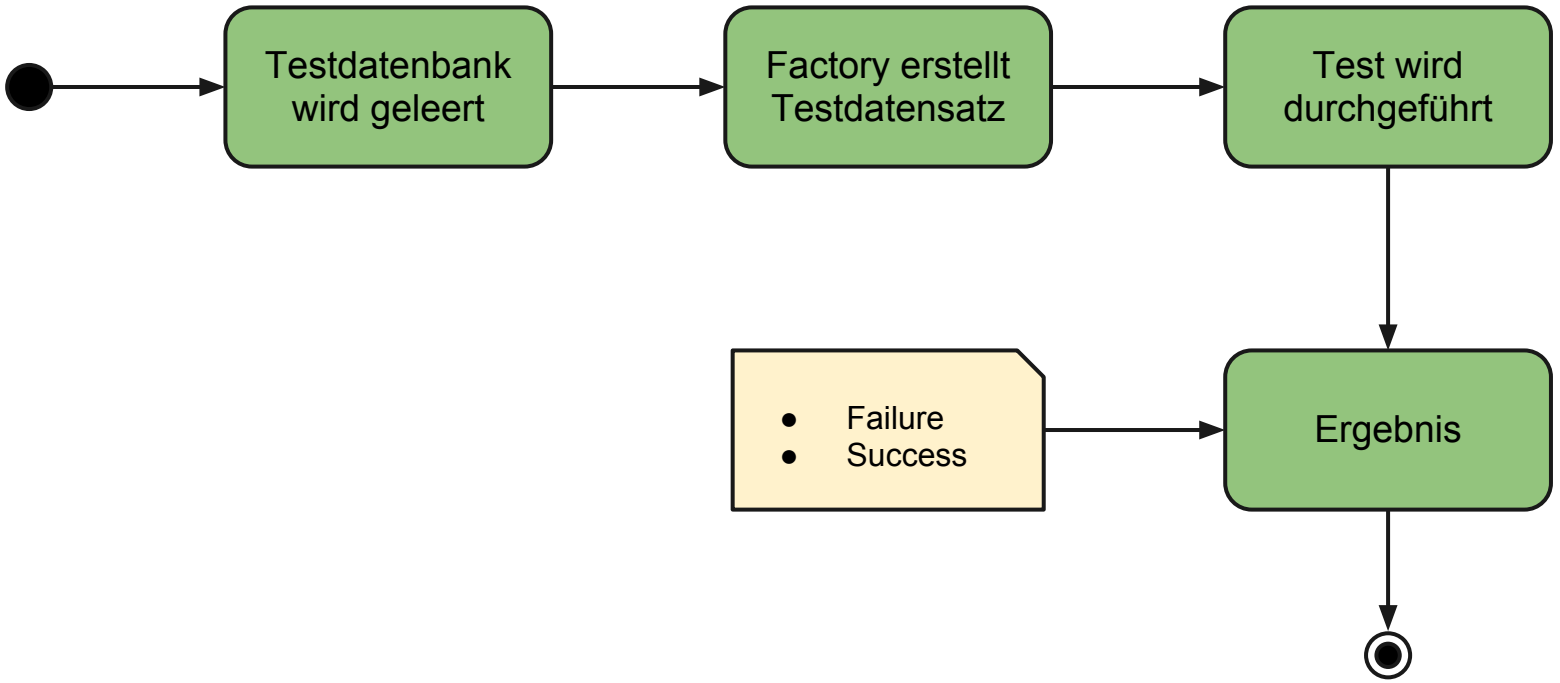
Unit Test

Umsetzung

- Zur Generierung der Testdaten wird eine Factory erstellt
  - Diese wird im Test zuerst validiert
- Jedes Attribut und jede Funktion wird isoliert getestet
  - Dabei werden sowohl die gültigen als auch ungültigen Fälle berücksichtig (z.B. Verletzung des Wertebereiches oder nicht vorhande Werte/Beziehungen)

Unit Test

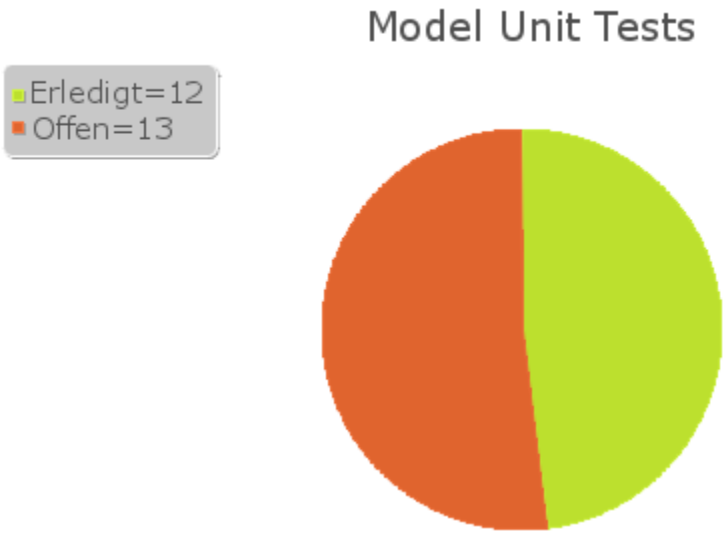
Ablauf



Unit Test

Ist-Stand

- 354 examples, 0 failures, 79 pending
- Die Kern-Modelle sind getest.
  - z.B. Person, OZBPerson, OZBKonto, EE-Konto, ZE-Konto, usw.



Unit Test

Fazit

- Es sind Inkonsistenzen aufgefallen
  - Model stimmte nicht mit der Datenbank überein
  - Validierungen sind nicht vorhanden oder fehlerhaft
  - Umsetzung der Modelfunktionenmit gleicher Funktionalität sind unterschiedlich, oder auch fehlerhaft umgesetzt
- Änderungen im Datenmodell, im Model wirken sich direkt auf die Tests und auf die Anwendunge aus
- Mit einem festgelegten Datenmodell kann die Konsistenz der Modelle, sowie der Anwendung, sichergestellt werden.

Controller Test

Ziele

- Stellt die korrekte Funktionalität der Funktionen im Controller sicher
  - Reagiert der Controller auf die Eingaben des Benutzers korrekt?
- Sicherstellung der korrekten Ausführung privater Funktionen

Controller Test

Voraussetzungen

- Fachlich und technisch korrekter Testdatensatz
- Alle Modelle, die in Verbindung mit dem Controller stehen, sind mit Hilfe von Unit Tests getestet worden
- Anwendungsfälle sind definiert
  - z.B. konkrete Benutzereingaben mit Ergebnissen die erwartet werden

Controller Test

Umsetzung

- Zur Generierung der Testdaten wird eine Factory erstellt
  - Diese wird im Test zuerst validiert
- Jede Funktion wird isoliert getestet
  - Dabei werden sowohl die gültigen als auch ungültigen Fälle berücksichtig (z.B. Verletzung des Wertebereiches oder nicht vorhande Werte/Beziehungen)
  - Vordefinierte Eingabeparameter werden hierbei vorbereitet und mit Hilfe der HTTP Operationen (GET, POST, PUT, DELETE) angewandt
  - Das Ergebnis (= Wert der verfügbaren Instanzvariable des Modells) wird mit den erwarteten Ergebnis verglichen
  - Somit können die Benutzereingaben 1:1 verarbeitet und nachgestellt werden

Controller Test

Ist-Stand

- Testcase: Darlehensverlauf
  - Testdatensatz hierfür ist der vorhandene Datensatz aus der Testdatenbank ozb_test. Er wird mit Hilfe eines Batchskriptes vor jedem Test migriert
  - Implementierung der Test Struktur abgeschlossen
  - Umsetzung für den ersten Testfall abgeschlossen:

```
context "Show Darlehensverlauf of EEKonto 70073"
        context "parameters: anzeigen, vonDatum and bisDatum are nil"
                it "returns the 10 latest buchungen"
```

Controller Test

Fazit

- Controller Tests bieten die Möglichkeit, die Funktionaliäten mit Benutzereingaben zu testen
- Für einen konsistenten Controller Test sind die möglichen Szenarien = Benutzereingaben zu definieren, sowie auch die erwarteten Ergebnisse
- Die Grundvoraussetzung sind die vollständig getesteten Modelle, die auf einem festgelegten Datenmodell basieren.

Datenbankmodell

Nach Änderungen am Datenbankmodell sind folgende Anpassungen notwendig :

- create_tables.txt
- ER-Diagram
- Migration-Tool
- Tabellenübersicht
- Model
  - Validierung, Datenformat, Wertebereich
- Unit Tests
- Testdaten (Factories, SQL-Dumps)
- ...

=> Priorität sollte sein die Korrektheit des Datenbankmodells sicherzustellen und zu finalisieren.

Denn jede Änderung kostet extrem viel Zeit und Aufwand.

Offene Fragen, Issues, Kommentare, ...

Siehe hierzu auch:
https://github.com/Avenel/FirstApp/issues

Issue: KKL Verlauf

- Wenn ein KKL-Verlauf Eintrag gelöscht wird, werden alle zeitlich vergangene (KKLAbDatum) Einträge gelöscht. Weshalb werden ältere Verläufe der Kontenklasse mitgelöscht?

Issue: Buchung

- Beschreibungsnamen für die Felder "wSaldoAcc" und "pSaldoAcc" fehlen.

Issue: Unit Test

- Ist der Sachbearbeiter nun Pflicht?

  JA [ ]
  NEIN [ ]

- Test: Bank: Gibt es einen invaliden Banknamen?
- Bankverbindung:
  - Warum ist BankKtoNr ein varchar und kein Integer?
  - Warum gibt es Datensätze, die einen Bindestrich in der BankKtoNr enthalten?

Issue: Unit Test

Create_Tables:

Ist es richtig, dass in dem Schema des EEKontos:

- der SachPnr *nicht* als ForeignKey ausgewiesen ist?
- die BankID *nicht* als NOT NULL deklariert ist?
- Kreditlimit auf NOT NULL gesetzt, es *muss* einfach ein Kreditlimit geben, sonst machts keinen Sinn.

Habe das mal eben in der create_tables angepasst.

Darf SachPnr NULL sein? Ich lasse dies erst einmal so.

Issue: Unit Test

- pgnr (Projektgruppen-Nr) wird vom Model gefordert, aber in der create tables ist es nicht so. Ich ändere das in der create tables af NOT NULL ab.

Buergschaft:

- Warum sind dort die FOREIGN Keys nicht eingetragen?

Ich ändere das mal.

Issue: Unit Test

Buergschaft: Dort wurde noch keine Historisierung aktiviert. Sprich, jegliche callback Methoden fehlen. Primary Key stimmt nicht. NOT NULL Constraints stimmen auch nicht überein. SichKurzbez, ist das ein enum = {Einzelbuergschaft, Teilbuergschaft} oder nicht? Noch alte Validierungsmethodik. Es wurde offensichtlich seit 1.5 Jahren nix mehr hier getan.

Issue: Unit Test

ZE_Konto:

- Welche Zahlmodi gibt es?
- Welche Möglichkeiten ergeben sich für ZEStatus? Ich habe gerad nur N, D und A gefunden.