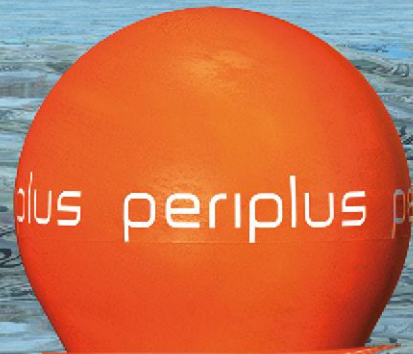


Modelbased Testing with FitNesse Tests for REST Services with periplus autogen

Dierk Ehmke



autogen: 100% automatisch synthetisierte Testdaten

- Erstellung der Testdaten **rein auf Metadaten-Basis**
- Erstellung der Testdaten **ohne manuelle Skript-Erstellung**
 - keine besonderen Kenntnisse von Programmiersprachen erforderlich
- Testen **ohne Produktivdaten**
 - keine Datenschutzprobleme
 - keine Gefahr von Datenlecks oder ungewollten Transaktionen
- Testen **ohne Liefersysteme**
 - Testdaten können bereits produziert werden, bevor die Liefersysteme ausgerollt werden (Smoke-Testdaten für Betriebliche Tests und Portierungstests)



autogen Editor

autogen - rev110

Datei Bearbeiten Suchen Ansicht Einstellungen Fenster ?

autogen Projekt rev110 Sourcen Testplan Business Cases Testfall rot

Sources:

Tabelle: **BUCHER**

Spalten

	Name	Typ	Einschränkungen	Extra 1	Extra 2
ISBN		char	not null	30	0
TITEL		char	not null	30	0
AUTOR		char	not null	30	0
VERLAG		char	not null	30	0

ISBN	TITEL	AUTOR	VERLAG
1		Hesse	
2			periplus
3			

Orakel (SQL-Skript):

```
SELECT * FROM BUCHER ;
```

Testdaten Generieren Testfall speichern

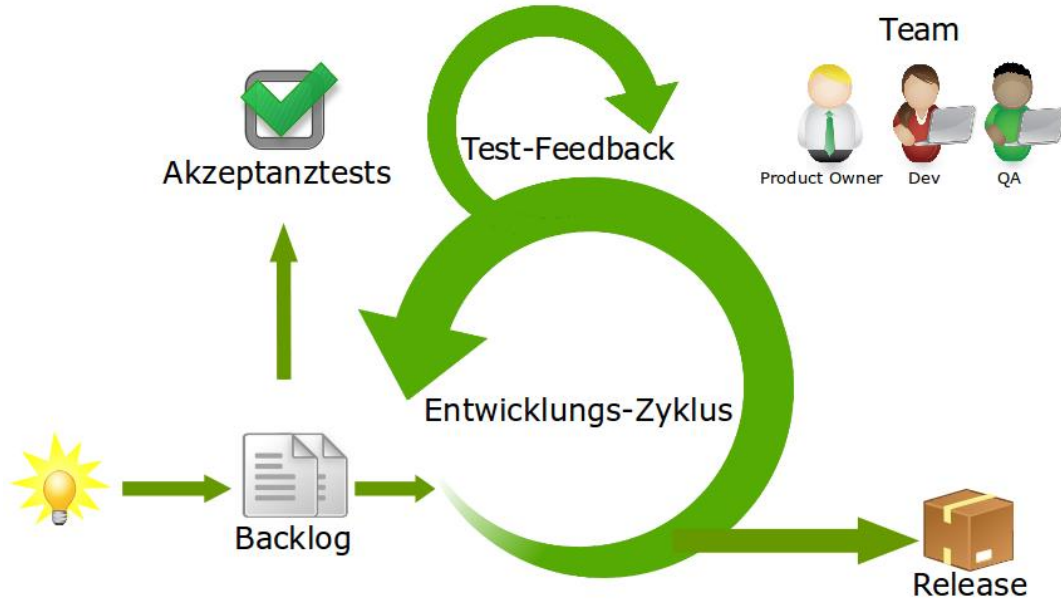
Unterstützte Typen

- Integer
- Fixpunkt
- Datum
- Gleitkomma
- Zeichenketten

autogen Graph

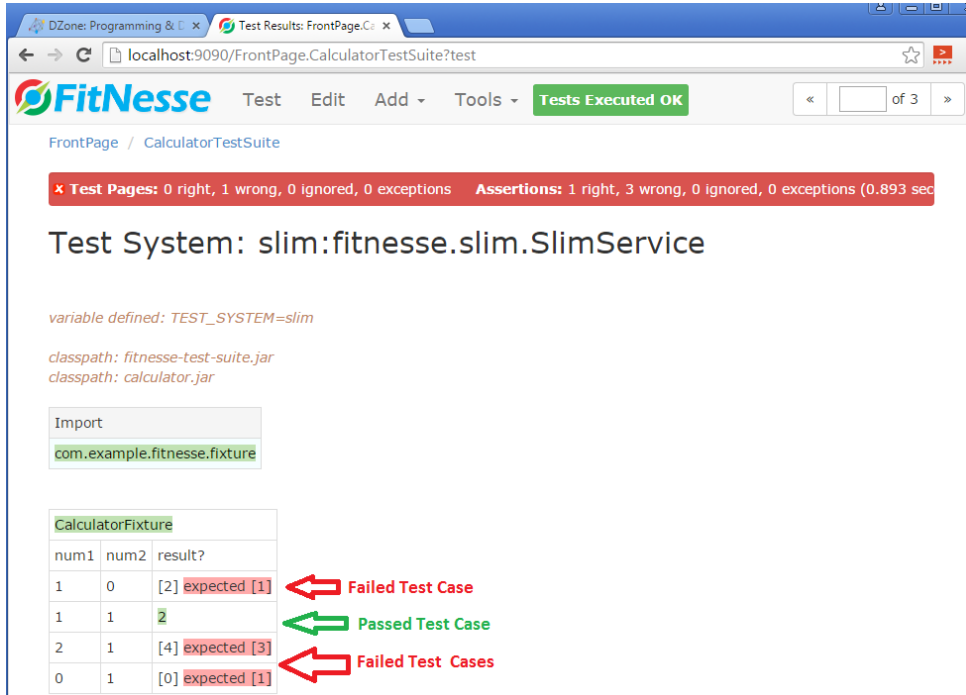


Agiles (Test-) Vorgehen



<http://www.agile-software-quality-assurance.com/traditional-vs-agile-qa/>

FitNesse - fachliche Sicht



The screenshot shows the FitNesse web interface in a browser. The address bar shows the URL `localhost:9090/FrontPage.CalculatorTestSuite?test`. The FitNesse logo is in the top left, and a green button says "Tests Executed OK". Below the navigation bar, the breadcrumb "FrontPage / CalculatorTestSuite" is visible. A red status bar indicates: "x Test Pages: 0 right, 1 wrong, 0 ignored, 0 exceptions Assertions: 1 right, 3 wrong, 0 ignored, 0 exceptions (0.893 sec)". The main heading is "Test System: slim:fitnesse.slim.SlimService". Below it, the variable `TEST_SYSTEM=slim` is defined. The classpath is listed as `fitnesse-test-suite.jar` and `calculator.jar`. An "Import" box contains `com.example.fitnesse.fixture`. A table titled "CalculatorFixture" shows test results:

num1	num2	result?
1	0	[2] expected [1]
1	1	2
2	1	[4] expected [3]
0	1	[0] expected [1]

Annotations with arrows point to the test results:

- Red arrow pointing to the first row: Failed Test Case
- Green arrow pointing to the second row: Passed Test Case
- Red arrow pointing to the third and fourth rows: Failed Test Cases

<http://fitnesse.org/> (Logo)
<https://dzone.com/articles/fitnesse-test-part-1> (Tabelle)

FitNesse - technische Sicht

The screenshot shows the FitNesse web interface in a browser. The page title is "FrontPage". Below the title, there are input fields for "Page name:" (containing "CalculatorTestSuite"), "Help text:", and "Tags:" (containing "add a tag"). Below these are buttons for "Spreadsheet to FitNesse", "FitNesse to Spreadsheet", "Format", "Insert Template", and a "wrap" checkbox. There is also a radio button for "autoformat".

The main content area displays a code editor with the following code:

```
1 |# define TEST_SYSTEM as 'slim' By default it is 'fit'
2 |
3 |!define TEST_SYSTEM {slim}
4 |
5 |!path fitness-test-suite.jar
6 |!path calculator.jar
7 |
8 |Import
9 |com.example.fitness.fixture
10 |
11 |!| CalculatorFixture
12 |num1|num2|result?
13 |1|0|1
14 |1|1|2
15 |2|1|3
16 |0|1|1
17 |
```

Red arrows point to specific parts of the code with explanatory text:

- Arrow 1 points to line 1: "Comment in FitNesse"
- Arrow 2 points to line 3: "Define test system as 'slim'"
- Arrow 3 points to line 5: "Add fixture classes and system under test's classes to classpath"
- Arrow 4 points to line 11: "Name of Fixture"
- Arrow 5 points to line 12: "Wiki Table. This is decision table in fitness where values of num1 and num2 are passed as inputs to fixture. It will be set using method setNum1() and setNum2() for each test case. Fixture will returns result using method result()."

At the bottom of the code editor, there are "Save" and "Cancel" buttons.

<http://fitnesse.org/> (Logo)

<https://dzone.com/articles/fitnesse-test-part-1> (Tabelle)

SQL als Beschreibungssprache

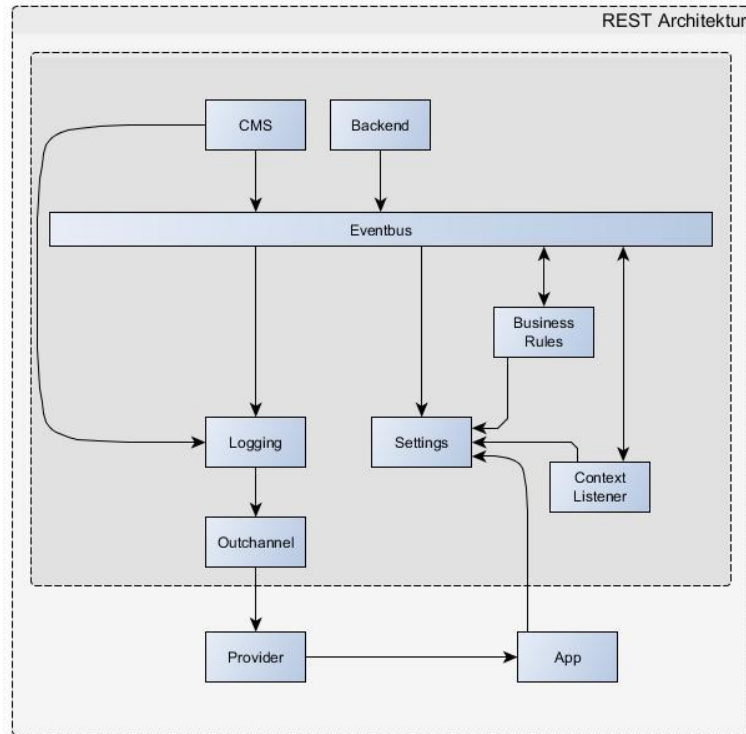
- **Große Verbreitung, besonders Entwickler, aber auch Tester und Analytiker**
- **Geeignet für Beschreibung komplexer Datenflüsse**
- **es muss keine graphische Notation gelernt werden**
- **Kompakte textuelle Darstellung (Bildschirm, Drucken)**



www.integrant.com/sql-20052008-master-database-recovery

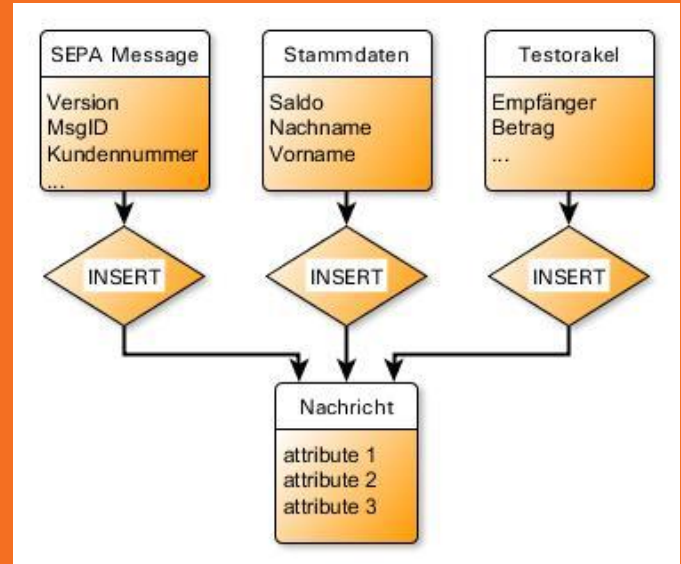


RESTfull Architektur



Beispiel

- Formulieren der Zusammenhänge (Constraints und INSERT-Statements)
- Ausgabeformatierung festlegen
- Festlegen der Testfälle (nur testrelevante Spalten)
- Testdaten, Stammdaten und Testorakel werden erzeugt



Informationen im Source Code

Tabellendefinitionen

```
KUNDENNR CHAR(10),  
KONTONR      INTEGER,
```

Constraints

```
GK_TYPE IN ( 'GKUM', 'GKVW', 'GKST' )  
KONTONR > 1000000000 AND KONTONR <= 9999999999
```

WHERE-Clauses

```
GK_ZIELBUCHUNGSDATUM > GK_BUCHUNGSDATUM
```

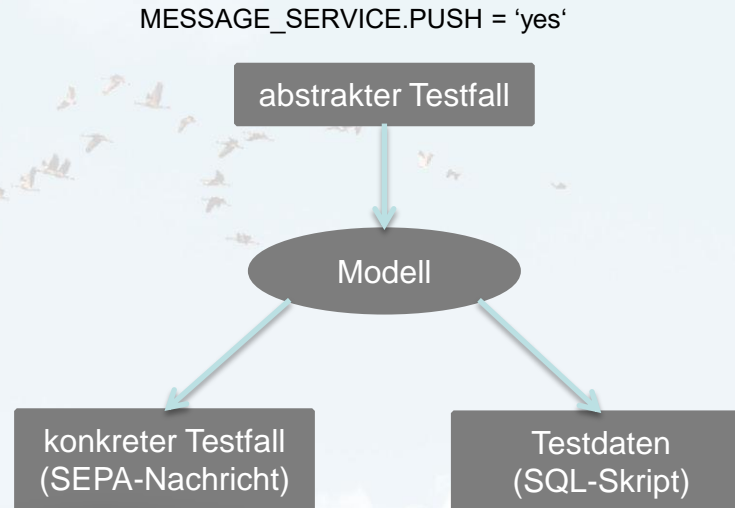
Berechnungen in der Select-Liste

Testfälle

1. **Smoketest**
2. **Table RULES_FIXTURE**

COM_KOMMENTAR	OUT_APPLE_CHANNEL	OUT_ANDROID_C HANNEL	OUT_BALANCE _ALARM
'does android channel work,	NOPUSH	PUSH	BLANK
'does apple channel work	PUSH	NOPUSH	BLANK
'do both channels work at the same time'	PUSH	PUSH	BLANK
'does no channel work :),	NOPUSH	NOPUSH	BLANK

Vom abstrakten zum konkreten Testfall



[1]G|KUMU0000000000000001|000000000001|0000000001|GIS
EPA|NIXIDE00000000000000000001||DE00000000000000000000
1|ABC|20170809|20170809|20170809|200000|UEBERWEISUNG
S|0000000000000001220|H|000000000006722180|H|0000000000
00000000|020AANDINO|TPROVIDED|20170809000000000000
0000000311757000||yes|1234567890|0001 an A überwiesen|

```
INSERT INTO SETTINGS (KUNDENNR,GERAETEADDRESS,LOCKED,ENABLED) VALUES ('00000000000001','1234567890','no','yes');
```

Synthese MBT, KDT und BDT

Schlüsselwortnotation als Beschreibungssprache

- Macht Testfälle unabhängig von der Dialogimplementierung
- (und macht sie lesbar)

„Verhaltensmodell“

- Ein- und Ausgaben (im Dialog)
- Übergänge (Vorgänger, Nachfolger)
- Ausgezeichnete End- und Anfangsknoten
- Statt Konstanter Werte werden Variablen und ihre Beziehungen zueinander beschrieben
- Tester gibt wieder nur Zielwerte vor
- Eingabesequenzen werden automatisch berechnet
- Macht Testfälle unabhängig von der Dialogsyntax

DANKE.



periphus autogen