31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Flyway vs. LiquiBase

Kampf der Datenbank-Migrationswerkzeuge

Stephan Kaps

Bundesversicherungsamt

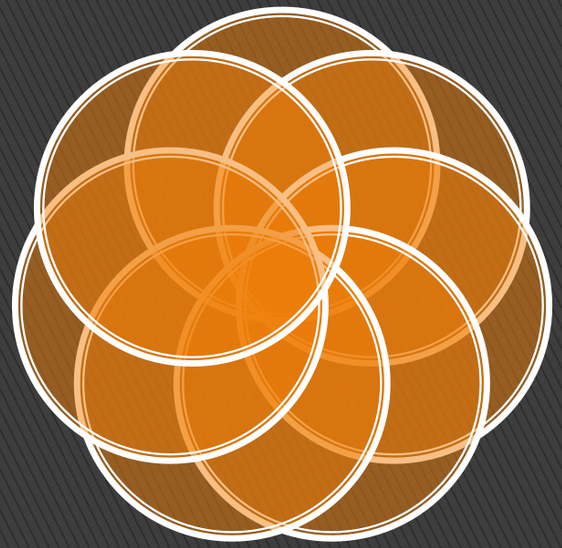
Flyway vs. LiquiBase

Battle der Datenbankmigrationstools

Stephan Kaps

Techn.
Projektleiter

Leidenschaft sind
neue
Technologien
und Methoden



Software-
Architekt

ISTQB, ISAQB,
IREB und ITIL
zertifiziert

JEE – SOA – Host

Speaker & Autor

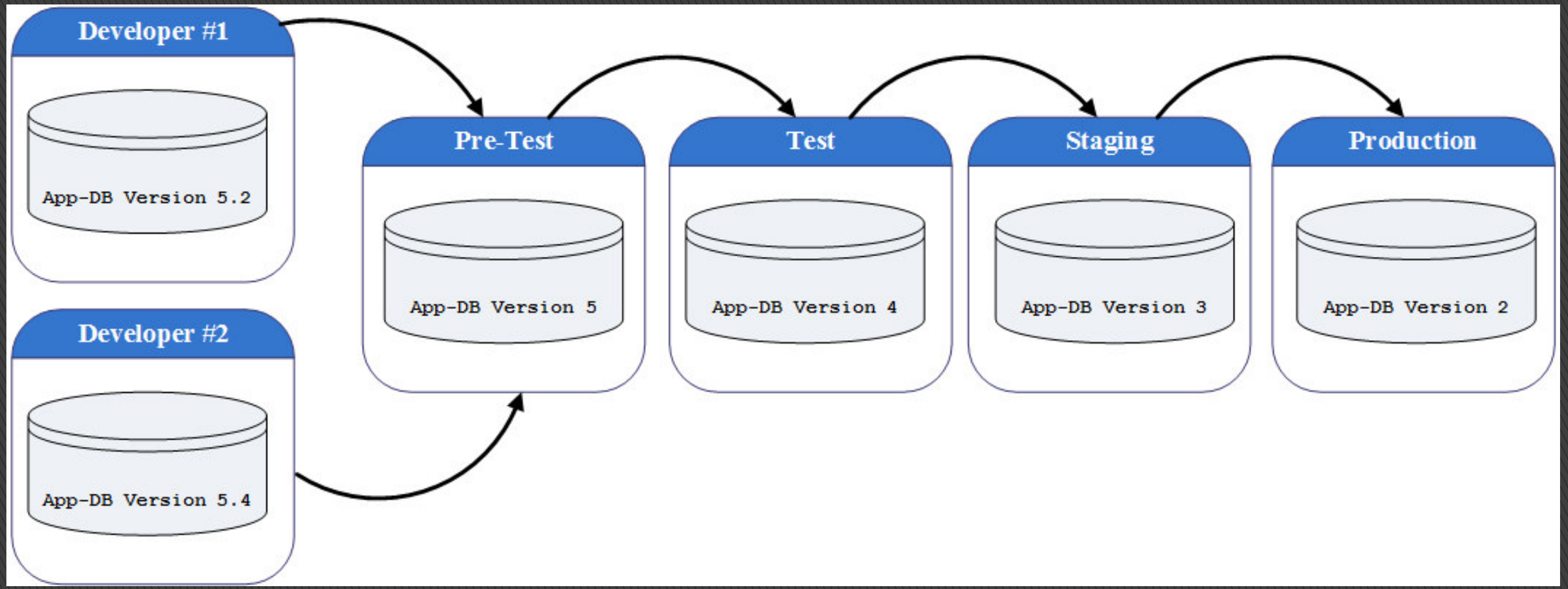
Java seit 2002

Agenda

Einleitung
Funktionsweise / Demo
BATTLE

1. Runde: Funktionen
2. Runde: Einbindung in bestehende Projekte
3. Runde: Integrationsmöglichkeiten
4. Runde: Inbetriebnahme und Handhabung
5. Runde: Verfügbare Plug-Ins
6. Runde: Rahmenbedingungen

Einleitung «


Developer #1
App-DB Version 5.2
Developer #2
App-DB Version 5.4
Pre-Test
App-DB Version 5
Test
App-DB Version 4
Staging
App-DB Version 3
Production
App-DB Version 2

Quellcode wird versioniert,
DB-Änderungen sollten auch!

Funktionsweise «

Funktionsweise

- Tools bieten direkte Integration in die eigene Anwendung (Migration on startup)
- Oder in den Build-Prozess

- Hauptkommandos:
  - migrate (Flyway)
  - update (LiquiBase)
- Aktuelle Stand der DB wird mit vorliegenden Migrationsskripten verglichen
- Kein DB-Administrator mehr notwendig

Funktionsweise

- merken sich in einer von ihnen selbst verwalteten Metatabelle für jedes Schema den aktuellen Versionsstand der jeweiligen Datenbank
  - Welche Skripte wurden in welcher Reihenfolge ausgeführt

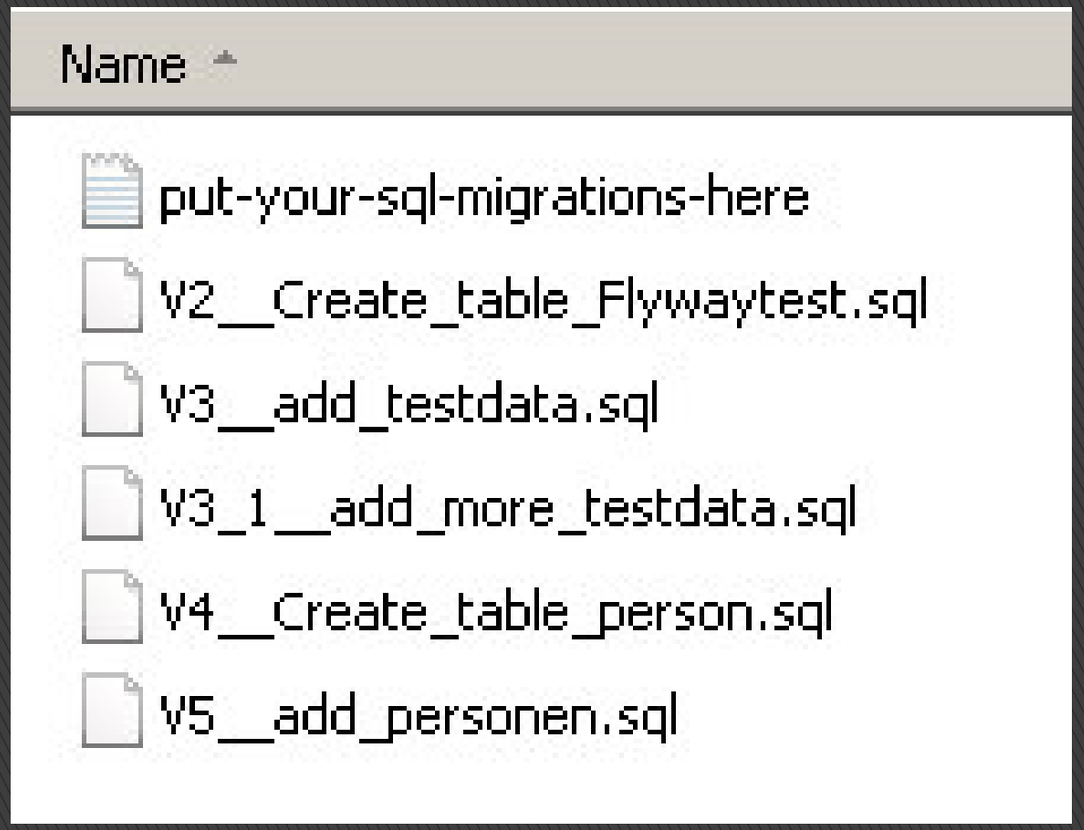
	version_rank integer	installed_rank integer	version [PK] characte	description character varying(200	type character var	script character varying(1000)	checksum integer	installed_by character var	installed_on timestamp w	execution_time integer	success boolean
1	1	1	2	Create test table	SQL	2__Create_test_table.sql	1814441205	postgres	2014-09-10	14	TRUE
2	2	2	3	add testdata	SQL	3__add_testdata.sql	-1342871217	postgres	2014-09-10	7	TRUE
3	3	3	3.1	add testdata	SQL	3_1__add_testdata.sql	916155695	postgres	2014-09-10	7	TRUE
*											

Funktionsweise

- LiquiBase bietet zusätzlich Rollbacks

	id chara	author character varying(	filename character varying(255)	dateexecute(timestamp w	orderexecute integer	exectype character var	md5sum character var	description character varying	comments character varying(255)	tag character varyin	liquibase character var
1	1	TimTester	changelog.xml	2014-12-04	1	EXECUTED	7:a49f5873f	createTable	Tabelle liquibaseTest erstellt	Version 2.0	3.2.2
2	2	System	changelog.xml	2014-12-04	2	EXECUTED	7:3bb63b6af	tagDatabase	''	Version 2.0	3.2.2
3	22	TonyMontana	changelog.xml	2014-12-04	3	EXECUTED	7:1bb937033	createTable	Tabelle person erstellt		3.2.2
4	23	TonyMontana	changelog.xml	2014-12-04	4	EXECUTED	7:77a74212f	sql	''		3.2.2
5	24	TonyMontana	changelog.xml	2014-12-04	5	EXECUTED	7:5e8924b83	addColumn	Spalte web hinzugefügt	Version 3.0	3.2.2
6	20	System	changelog.xml	2014-12-04	6	EXECUTED	7:92e5fe41e	tagDatabase	''	Version 3.0	3.2.2
7	88	RainerZufall	create_sqltest.sql	2014-12-04	7	EXECUTED	7:fa143db4e	sql	''		3.2.2
8	89	RainerZufall	create_sqltest.sql	2014-12-04	8	EXECUTED	7:9a3dd5331	sql	''		3.2.2
9	33	RainerZufall	insert_sqltest.sql	2014-12-04	9	EXECUTED	7:e8099ba25	sql	''	Version 3.1	3.2.2
10	21	System	changelog.xml	2014-12-04	10	EXECUTED	7:552ff2116	tagDatabase	''	Version 3.1	3.2.2

Funktionsweise Flyway



Funktionsweise LiquiBase

```
21
22      <changeSet author="TonyMontana" id="22">
23        <preConditions>
24          <tableExists schemaName="public" tableName="liquibaseTest" />
25        </preConditions>
26          <createTable tableName="person">
27             <column name="VORNAME" type="VARCHAR(100)"/>
28             <column name="NAME" type="VARCHAR(100)"/>
29             <column name="FIRMA" type="VARCHAR2(200)"/>
30             <column name="EMAIL" type="VARCHAR2(150)"/>
31          </createTable>
32      </changeSet>
33
34      <changeSet author="TonyMontana" id="23" context="!dev">
35         <sql>
36             insert into person (vorname, name, firma, email) values
37             ('Stephan', 'Kaps', 'Kitenco', 'info@kitenco.de')
38         </sql>
39         <rollback>
40             delete from person where name='Kaps';
41         </rollback>
42      </changeSet>
43
44      <changeSet author="TonyMontana" id="24">
45         <addColumn tableName="person">
46             <column name="WEB" type="character(200)"/>
47         </addColumn>
48      </changeSet>
49
50
51      <include file="create_sqltest.sql"/>
52      <include file="insert_sqltest.sql"/>
53
```

```
21
22      <changeSet author="TonyMontana" id="22">
23        <preConditions>
24          <tableExists schemaName="public" tableName="liquibaseTest" />
25        </preConditions>
26          <createTable tableName="person">
27              <column name="VORNAME" type="VARCHAR(100)"/>
28              <column name="NAME" type="VARCHAR(100)"/>
29              <column name="FIRMA" type="VARCHAR2(200)"/>
30              <column name="EMAIL" type="VARCHAR2(150)"/>
31          </createTable>
32    </changeSet>
33
34      <changeSet author="TonyMontana" id="23" context="!dev">
35          <sql>
36              insert into person (vorname, name, firma, email) values
37              ('Stephan', 'Kaps', 'Kitenco', 'info@kitenco.de')
38          </sql>
39          <rollback>
40              delete from person where name='Kaps';
41          </rollback>
42    </changeSet>
43
44    <changeSet author="TonyMontana" id="24">
45          <addColumn tableName="person">
46              <column name="WEB" type="character(200)"/>
47          </addColumn>
48    </changeSet>
49
50
51    <include file="create_sqltest.sql"/>
52    <include file="insert_sqltest.sql"/>
53
```

```
1  --liquibase formatted sql
2
3  --changeset tester:88
4
5  create table liquiSqlTest (
6      id int primary key,
7      name varchar(255)
8  );
9  --rollback drop table liquiSqlTest;
10
11 --changeset tester:89
12 insert into liquiSqlTest (id, name) values (1, 'name 1');
13 insert into liquiSqlTest (id, name) values (2, 'name 2');
14 --rollback delete from liquiSqlTest;
15
```

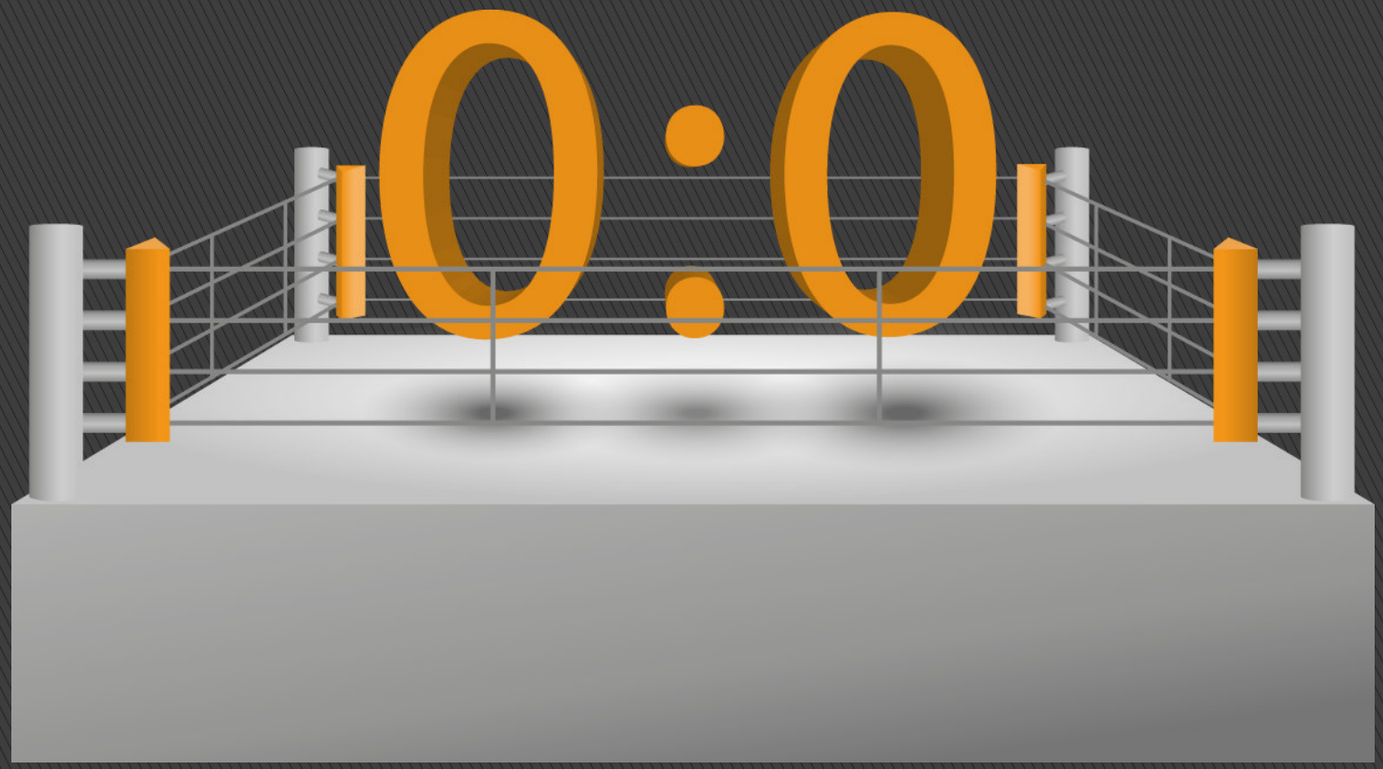
Battle «

V3.2.1 (03/2015)

VS.

LIQUIBASE

V3.4.1 (07/2015)



1.Runde «

Funktionen

Unterstützte Datenbanken

Flyway zusätzlich:

- SQL Azure
- Google Cloud SQL
- HP Vertica
- AWS Redshift
- DB2 on z/OS
- SolidDB

beide:

- **Oracle**
- **SQL Server**
- **DB2**
- **MySQL**
- **PostgreSQL**
- **H2**
- **Hsql**
- **Derby**
- **SQLite**
- MariaDB

LiquiBase zusätzlich:

- Sybase_Enterprise
- Sybase_Anywhere
- Informix
- Firebird

Funktionen

Funktion	Beschreibung	Flyway	LiquiBase
Migration	Migriert die Datenabnk auf eine neue Verison anhand vorliegender SQL-Skripte oder Java-Klassen mit programmierten Migrationen.	+++ `migrate`	+++ `update`
Rollback	Versetzt die Datenbank in einen vorherigen Stand	-	++ `rollback`
Schema leeren	Löscht alle Objekte (Tabellen, Views, Prozeduren, …) im konfigurierten Schema	+ `clean`	+ `dropAll`
Dokumentation	Flyway gibt Details und Statusinformationen (Metadaten) bisheriger Migrationen aus LiquiBase erzeugt eine JavaDoc ähnliche Dokumentation darüber, welche Änderungen an welchen Tabellen stattgefunden haben.	+ `info`	++ `DBDoc`

Funktionen

Funktion	Beschreibung	Flyway	LiquiBase
Vergleich zweier Datenbanken	Vergleicht zwei Datenbanken im Hinblick darauf, in welcher Datenbank noch Tabellen, Spalten, Views usw. fehlen oder zu viel sind. Dabei kann eine Changelog-Datei erzeugt werden, um im Anschluss die eine DB auf den Stand der anderen zu bringen	-	++ `Diff`
Validierung	Prüfung, ob DB sich auf dem aktuellsten Stand befinden oder ob noch Änderungen anstehen (pending)	+ `validate`	-
Initialisierung	Sofern schon Datenbanken vorhanden sind, erstellt und initialisiert diese Funktion die dazugehörige Metadaten-Tabelle	+ `baseline`	-
Wartung	Repariert die Metadaten-Tabelle, d.h. entfernt alle fehlgeschlagenen Migrationen und korrigiert falsche Checksummen	+ `repair`	+ `clearChecksums`

Funktionen

Funktion	Beschreibung	Flyway	LiquiBase
Callbacks	Erlauben es, Aktionen vor oder nach einem Kommando regelmäßig ausführen zu lassen (z.B. afterEachMigrate), z.B. für die Vergabe von Standardrechten an neuen Tabellen	+	-
Vorbedingungen	Müssen erfüllt sein, bevor eine Migration startet. Dadurch kann z.B. geprüft werden, ob ein bestimmtes DBMS existiert	-	+
Kontexte	Bewirkt, dass ein Set an Änderungen nur in einer bestimmten Umgebung ausgeführt wird (z.B. Test oder Produktion). Es kann auch dafür genutzt werden, eine Datenbank mit Testdaten zu versorgen.	-	++
Refactoring	Bietet Kommandos an, um automatisiert Optimierungen an der DB durchzuführen. z.B. addLookupTable: kann aus einer Spalte mit Referenzdaten eine eigene Tabelle erzeugen und diese per Fremdschlüssel mit der aktuellen verknüpfen	-	++

Funktionen

	Flyway	LiquiBase
Endergebnis	9	16

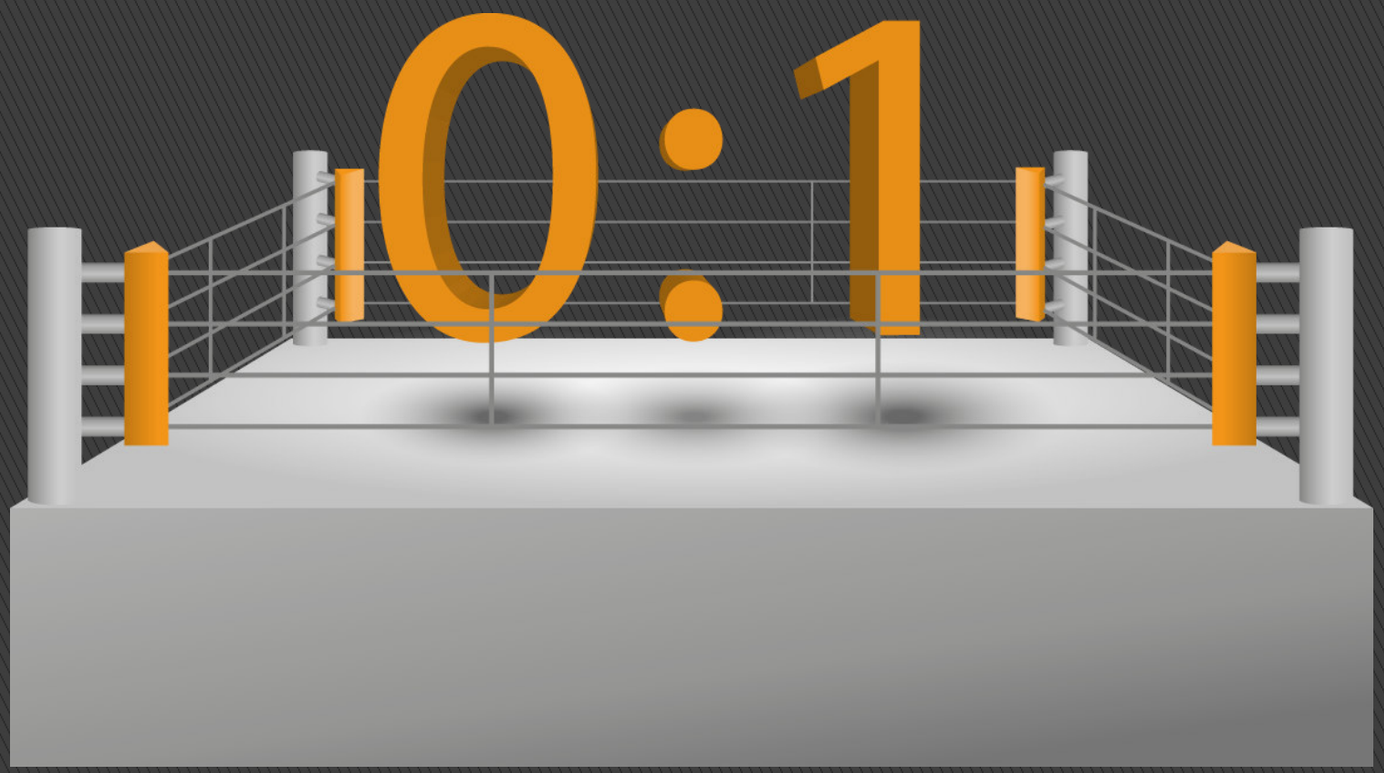
Außerhalb der Wertung:

Die Funktion `updateSQL` von LiquiBase

Ermöglicht, die anstehende Migration zunächst als SQL-Befehle auszugeben, z.B. zu Review-Zwecken.

Ist bei Flyway nicht relevant, da Plain-SQL-Skripte verwendet werden

vs. LIQUIBASE



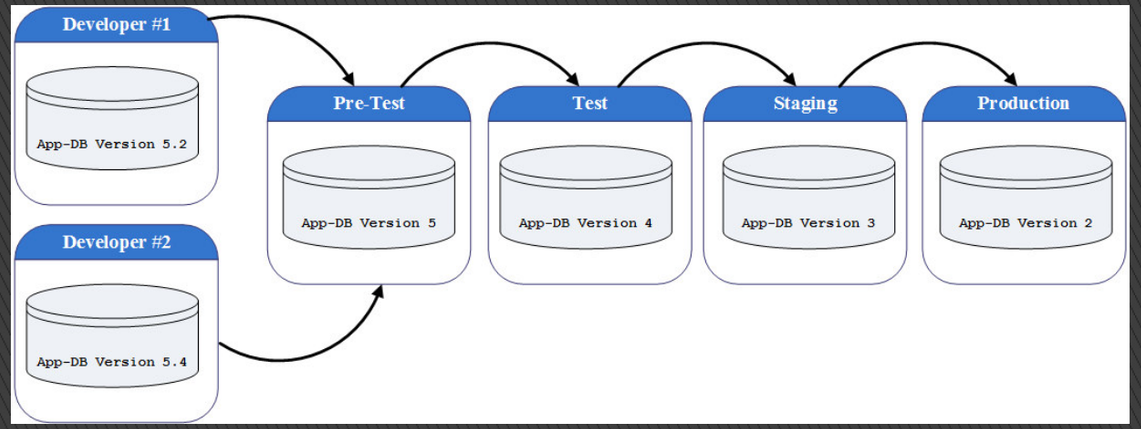
2.Runde «

Einbindung in bestehende Projekte

Einbindung in bestehende Projekte

- Herausforderung

Die aktuellen Datenbank-Stände nachträglich versionieren



Einbindung in bestehende Projekte

Flyway

- Manuellen Abzug (DDL) der Produktion erzeugen als Version 1 Skript
- Per **clean** alle anderen Datenbanken leeren und auf Version 1 migrieren
- Oder manuell auf den Stand der Produktion bringen und per **baseline**-Befehl den Stand als Version 1 setzen

LiquiBase

- Stand der Produktion per **generateChangelog** als Version 1 taggen
- Per **Diff** mit anderen Umgebungen vergleichen und changelogs erzeugen
- changelogs als „bereits gelaufen“ markieren bzw. für neue Läufe ausschließen

Ergebnis

- Flyway geht davon aus, dass die Datenbank-Version der Produktion die aktuellste ist
- Trifft bei den meisten Projekten nicht zu
- Manuelles Anpassen ist großer Schwachpunkt

- Integration bei LiquiBase durch geschickte Kombination mächtiger Funktionen deutlich vielversprechender

VS. LIQUIBASE

3.Runde «

Integrationsmöglichkeiten

Integrationsmöglichkeiten

Flyway:

- Java-API
- Command-Line
- Maven
- Ant
- Gradle
- SBT

LiquiBase:

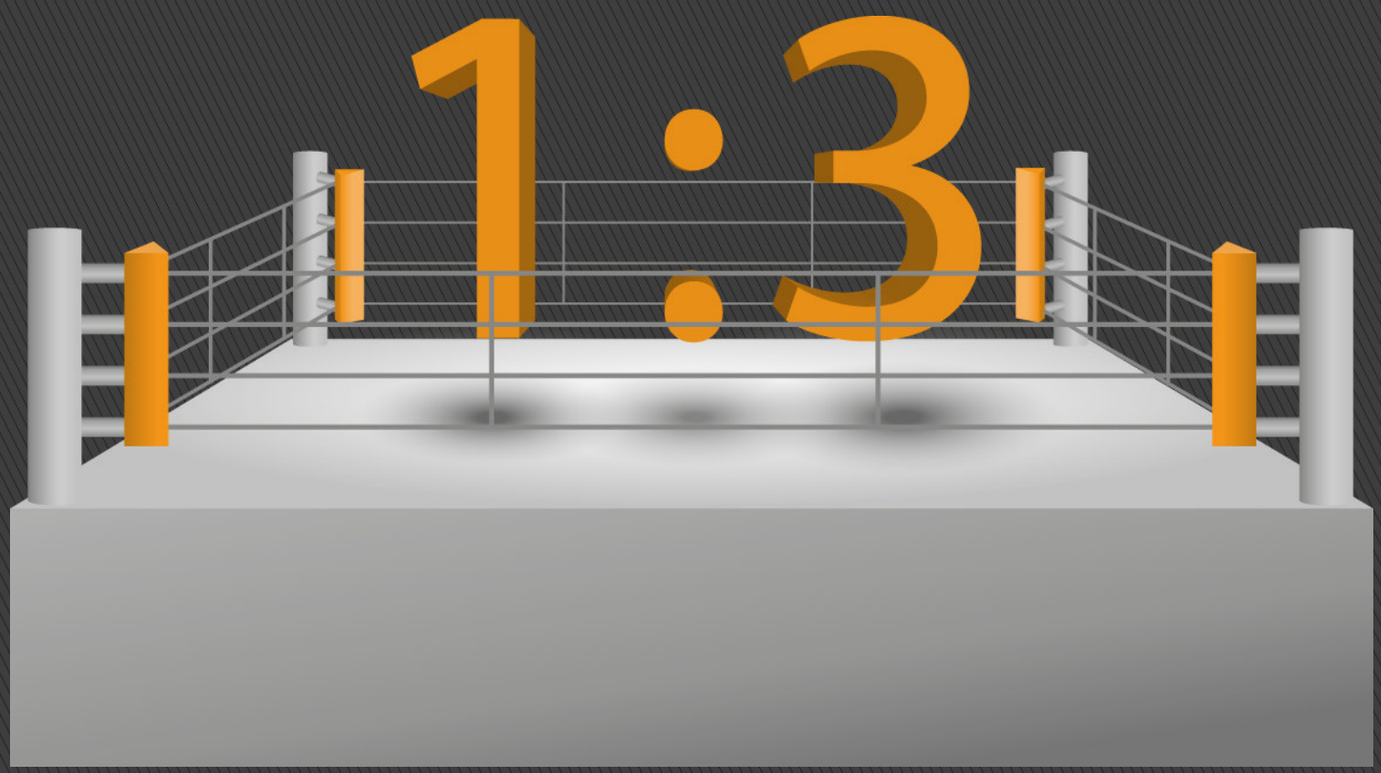
- Java-API
- Command-Line
- Maven
- Ant
- Spring (als Bean)
- Servlet-Listener (startet, wenn die Seite deployed wird)
- CDI Environment (startet wenn der CDI Container Bootet)

Integrationsmöglichkeiten

	Flyway	LiquiBase
Java-API	✓	✓
Command-Line	✓	✓
Ant	✓	✓
Maven	✓	✓
Gradle	✓	⊖
SBT	✓	⊖
Spring	✓	✓
Servlet-Listener	⊖	✓
CDI-Environment	⊖	✓

Flyway

vs. LIQUIBASE



4.Runde «

Inbetriebnahme & Handhabung

Inbetriebnahme & Handhabung

Flyway

- Archiv entpacken
- DB-Treiber inklusive
- Sehr einfach für neue Projekte
- Konfigurationsvorlage
- SQL-Skripte 1:1 verwendbar
- Aufruf mit einer Zeile in der Konsole

LiquiBase

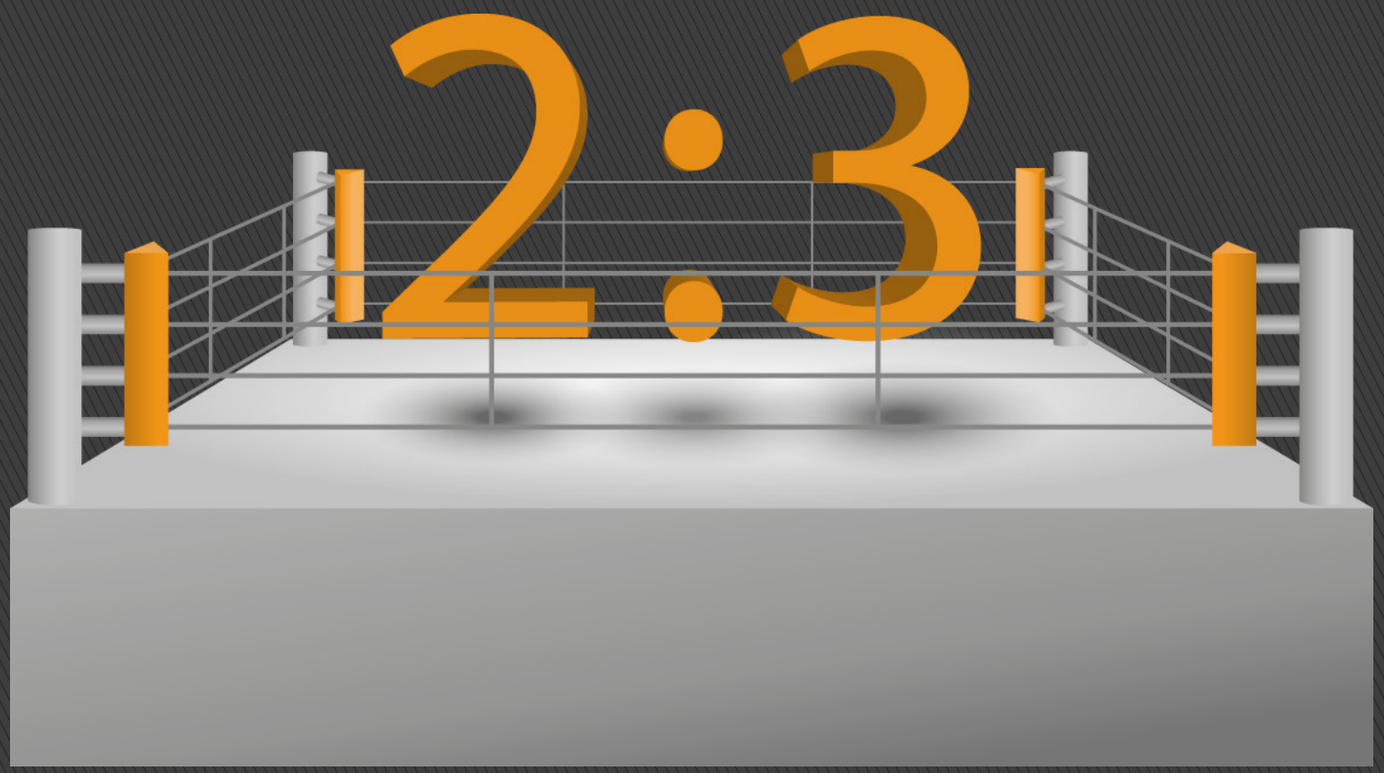
- Archiv entpacken
- DB-Treiber besorgen
- Konfigurationsdatei selbst anlegen
- SQL-Skripte anpassen bzw. **changelog.xml** erstellen
- Aufruf mit einer Zeile in der Konsole

Inbetriebnahme & Handhabung

	Vorteile	Nachteile
Flyway	- Einfache Konfiguration, vieles wird direkt mitgeliefert - Die Einbindung in Java ist sehr einfach - Generell leicht zu erlernen und einzusetzen	- Im Fehlerfall ist kein automatisches Rollback möglich
LiquiBase	- Automatisches Rollback erleichtert die Handhabung im Fehlerfall	- Viele Parameter für die Kommando-Zeile notwendig - Schwerer zu erlernen (z.B. Syntax für Changelog-Datei) - Einbindung in Java ist nicht beschrieben, nur JavaDoc steht online bereit

Flyway

vs. LIQUIBASE



5.Runde «

Verfügbare Plug-Ins

Erweiterungsmöglichkeiten

Flyway:

Spring Boot & Roo, Grails, Play, DropWizard, ...

http://flywaydb.org/documentation/pluginsThirdParty.html

LiquiBase:

Hauptsächlich DB-Extensions

https://liquibase.jira.com/wiki/display/CONTRIB/Liquibase+Extensions+Portal

LiquiBase hat mehr, Flyway aber die interessanteren

Flyway

vs. LIQUIBASE

6.Runde «

Rahmenbedingungen

Rahmenbedingungen

- Community
- Entwicklung
- Dokumentation
- Support

...

1. Community

LiquiBase ca. 3-4 mal so viele Contributor (152 zu 39)

und Commits wie Flyway (4400 zu 1350)

→ Vorteil LiquiBase

...

2. Entwicklung

LiquiBase deutlich mehr Tickets (auch offene)

Aber auch deutlich weitere Verbreitung

Releases bei beiden eher unregelmäßig

→ Unentschieden

...

3. Dokumentation

Flyway:

alles sehr gut beschrieben, kurz aber verständlich

LiquiBase:

alles sehr gut beschrieben, mehr Umfang da mehr Features,

→ Unentschieden

...

4. professioneller Support

Flyway: ja, sogar auf deutsch

LiquiBase: ja, Sprache Englisch, da Firmensitz Texas

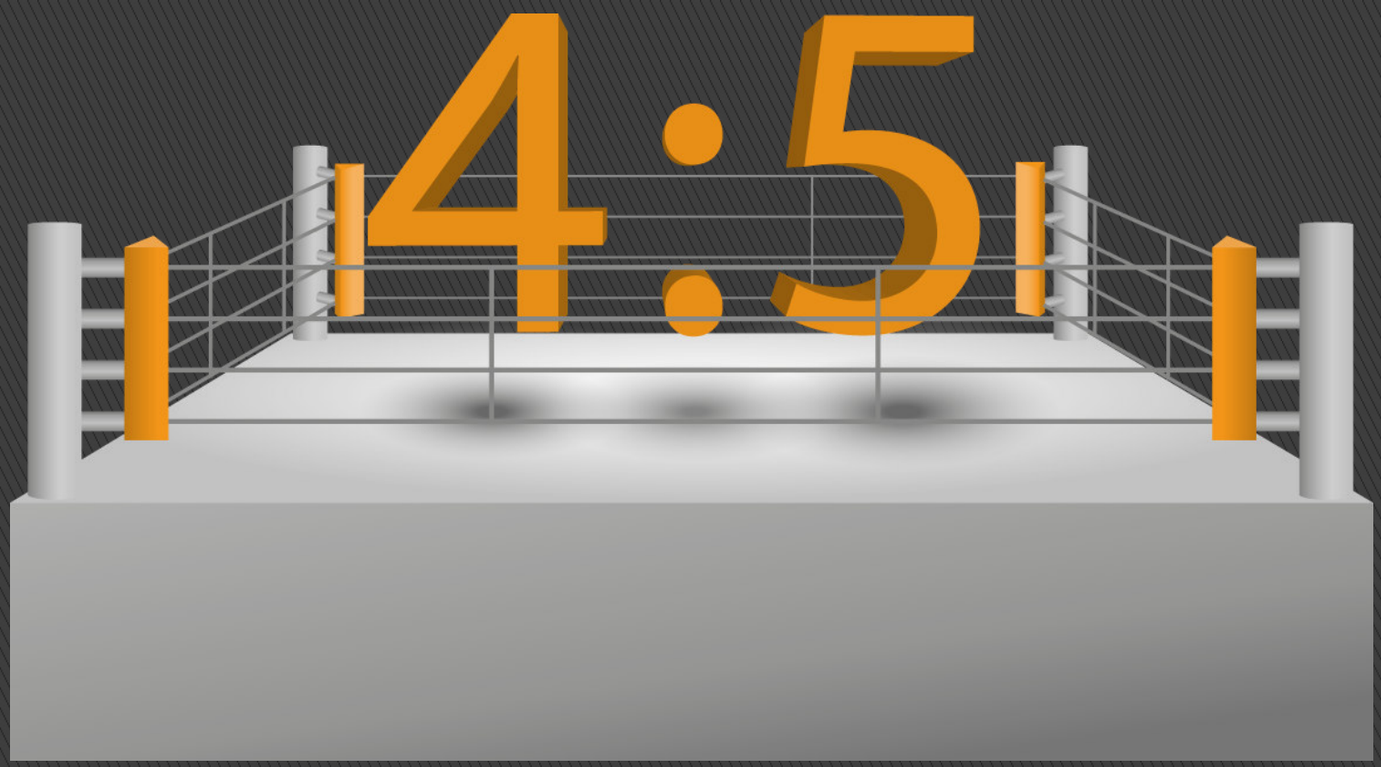
→ Vorteil Flyway

Rahmenbedingungen

- Community → LiquiBase
- Entwicklung → Unentschieden
- Dokumentation → Unentschieden
- Support → Flyway

→ insgesamt Unentschieden

Flyway vs. LIQUIBASE



Fazit

Knapper Sieg für LiquiBase, aber … Entscheidung für Flyway oder LiquiBase hängt von anderen Faktoren ab:

- Wird im Projekt eine spezielle Datenbank verwendet, die nur ein Tool unterstützt?
- Will man seine bisherigen SQL-Skripte 1:1 weiterverwenden?
- Kann man auf Rollbacks verzichten? (z.B. durch geschickte Forward-Migrationen)
- Benötigt man deutschen Support?
- Will man seine Testdaten mit verwalten lassen?
- Will man diverse DBMS unterstützen?
- Wie groß ist das Entwicklerteam?

Fazit

- Unterstützung von NoSQL-Datenbanken ist noch offen

- Starten Sie mit einem kleinen neuen Projekt
- Machen Sie behutsam auf die Vorteile aufmerksam
- Es sind nicht nur technische Hürden zu überwinden

Vielen Dank!

Noch Fragen?

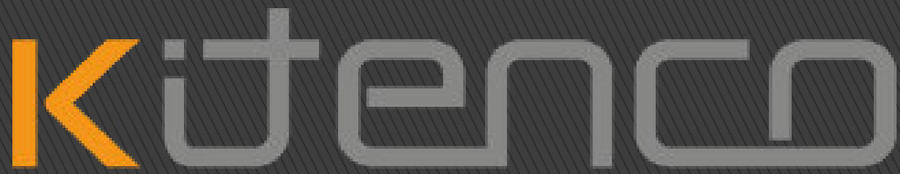
Artikel dazu in ObjektSpektrum 2/2015

Stephan Kaps | @kitenco1 | info@kitenco.de

Vielen Dank an Rebecca Nöll für die Erstellung der Boxring-Grafiken

work just simply smarter

- Continuous Delivery
- Application Lifecycle Management
- BizDevOps



- Schulung
- Coaching
- Training
- JIRA
- Jenkins
- Confluence
- Vorträge
- Artikel
- PoCs