DATENBANKZENTRISCHE
ANWENDUNGEN MIT

SPRING BOOT UND JOOQ

MICHAEL SIMONS

- Entwickler bei ENERKO INFORMATIK in Aachen
  - Datenbankzentrische Anwendungen im Energiemarkt
- Leiter Euregio JUG
- Bloggt zu Java und Spring Themen unter info.michael-simons.eu
- Co-Autor arc(42) by example
- @rotnroll666 auf Twitter

Agenda

- Spring Boot
- jOOQ
- Oracle JET

```
        <property name="fallbackToSystemLocale" value="false"/>
    </bean>
    <!-- Configure the JPA Adapter -->
    <bean id="jpaDialect" class="org.springframework.orm.jpa.vendor.HibernateJpaDialect"/>
    <bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
      <property name="database" value="MYSQL"/>
      <property name="databasePlatform" value="org.hibernate.dialect.MySQLDialect"/>
      <property name="generateDdl" value="false"/>
      <property name="showSql" value="false"/>
    </bean>
    <!-- Configure the local Entity Manager Factory -->
    <bean id="entityManagerFactory"
  class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="persistenceUnitName" value="theEntities"/>
      <property name="dataSource" ref="theDataSource"/>
      <property name="jpaDialect" ref="jpaDialect"/>
      <property name="jpaVendorAdapter" ref="jpaVendorAdapter"/>
      <property name="loadTimeWeaver">
        <bean class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver"/>
      </property>
    </bean>
    <!-- Enable Spring JPA Transactions -->
    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager" p:entity-
manager-factory-ref="entityManagerFactory"/>
    <tx:annotation-driven transaction-manager="transactionManager"/>
    <bean id="exampleBean" class="examples.ExampleBean">
      <!-- setter injection using the nested <ref/> element -->
      <property name="beanOne">
        <ref bean="anotherExampleBean"/>
      </property>
      <!-- setter injection using the neater 'ref' attribute -->
      <property name="beanTwo" ref="yetAnotherBean"/>
      <property name="integerProperty" value="1"/>
    </bean>
    <bean id="anotherExampleBean" class="examples.AnotherBean"/>
    <bean id="yetAnotherBean" class="examples.YetAnotherBean"/>
</beans>
```

A long time ago, in a framework far,
far away

A NEW HOPE...

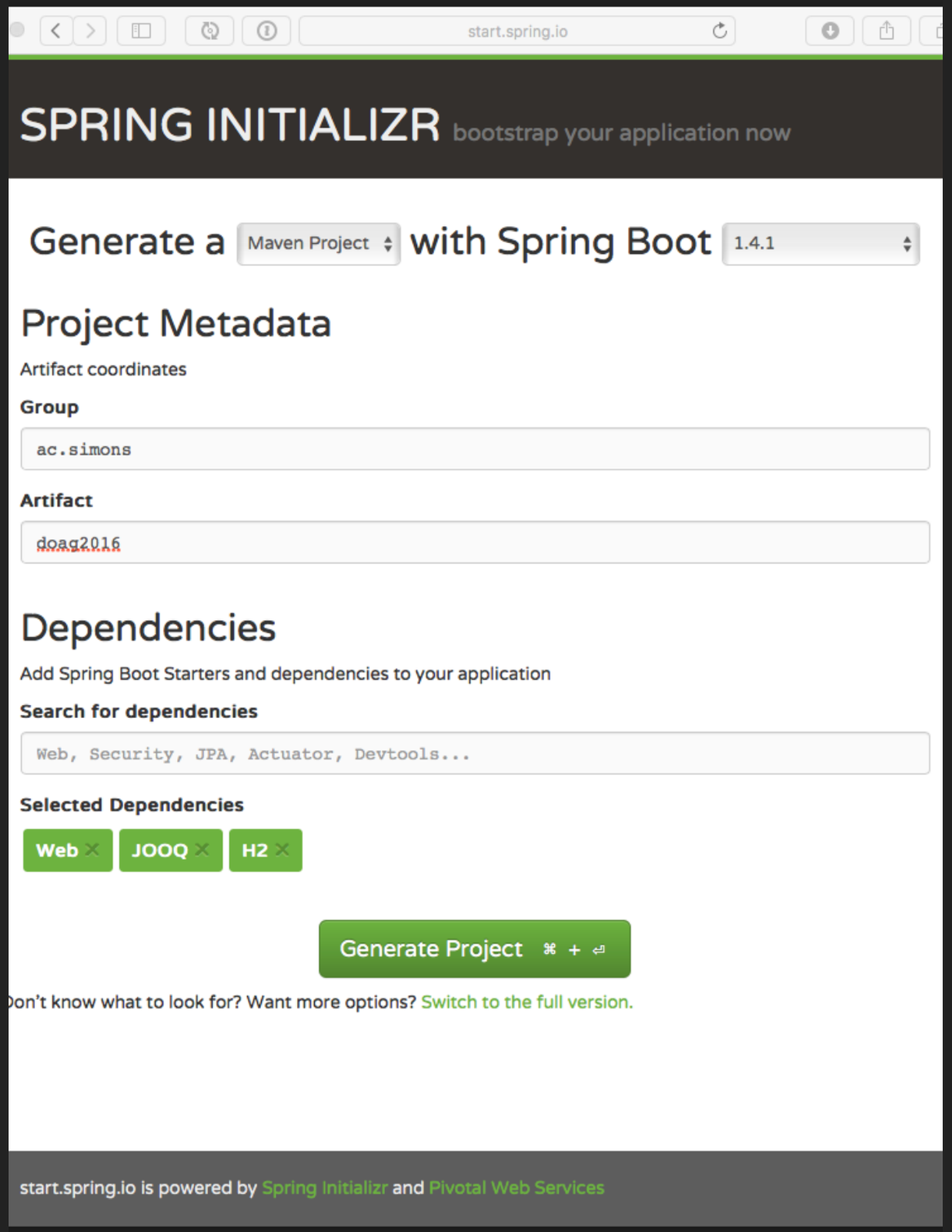
```
package ac.simons.doag2016;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String... args) {
        SpringApplication.run(Application.class, args);
    }
}
```

WAS IST SPRING BOOT?

- „Fertig konfigurierte Instanz des Spring Frameworks“
- *spring-boot-starter-** Jars beinhalten automatische Konfiguration und deklarieren Abhängigkeiten
- *spring-boot-starter-** Jars erlauben gezielte Auswahl von Spring eigenen und unterstützten Technologien
  - unter anderem auch **jOOQ**
- Spring Boot beinhaltet keinen Code Generator!
- XML Konfiguration optional
- Bestmögliche „Out-of-the-box“ Erfahrung mit dem Spring Öko-System

SPRING INITIALIZR

- Über start.spring.io
- Oder direkt aus einer IDE



DEMO

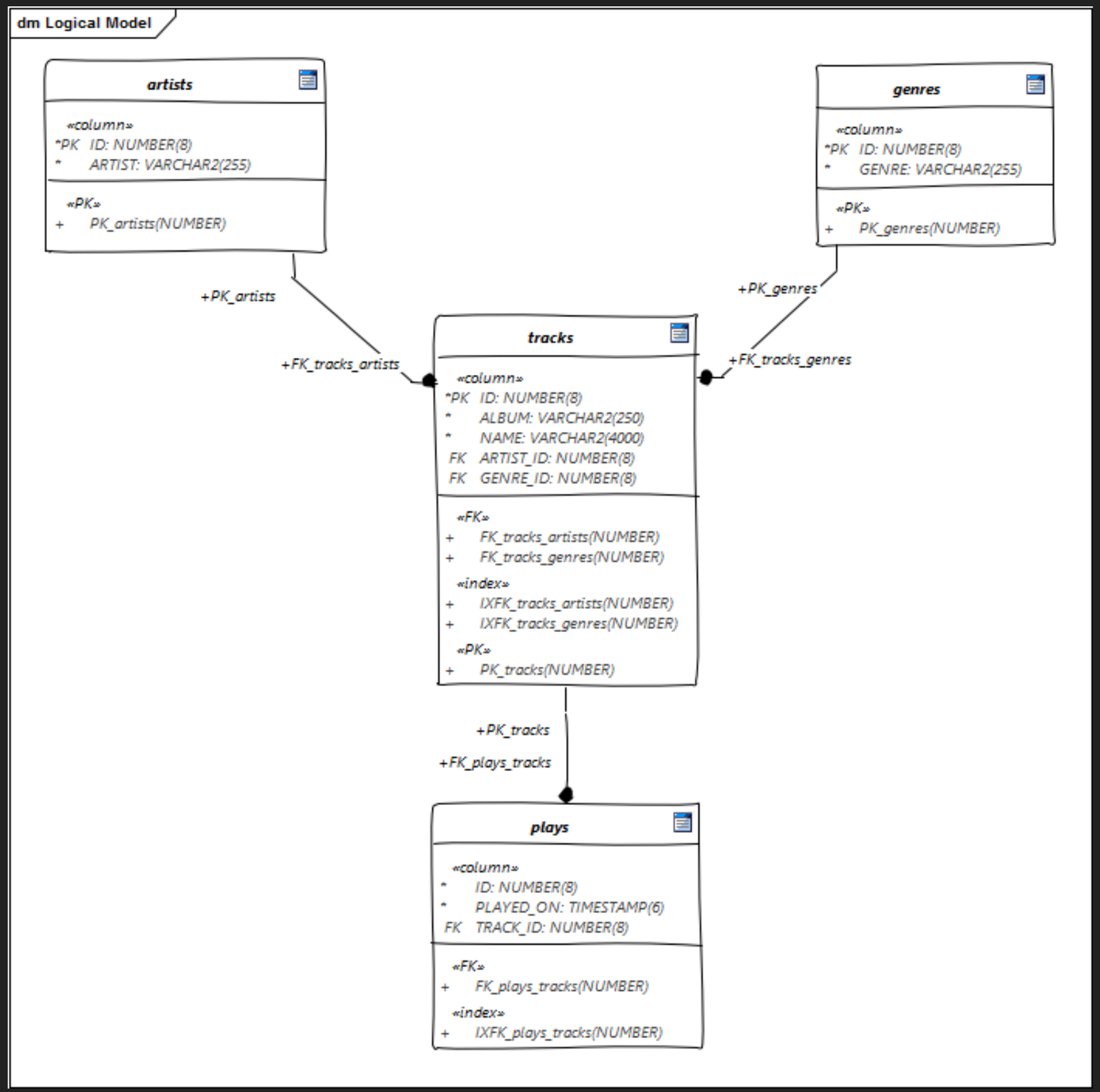
ZUSAMMENFASSUNG

- „Up and running“ in wenigen Augenblicken
- Automatische Konfiguration erfolgt, wenn Abhängigkeiten vorhanden sind
- Kann mit Umgebungsvariablen, Properties oder Konfigurationsservern überschrieben werden

JOOQ

Java und Datenbanken: Plain SQL, ORM oder etwas dazwischen?

DAS BEISPIEL SCHEMA



```sql
-- So?
Select *
  from tracks
 where album = 'True Survivor';
```

UND DAS SQL DAZU?

```java
// oder so?
@Entity
@Table(name = "tracks")
public class TrackEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column
    private String album;


    public static void main(String...a) {
        final EntityManagerFactory factory = Persistence.createEntityManagerFactory("whatever");
        final EntityManager entityManager = factory.createEntityManager();

        List<Track> tracks = entityManager.createQuery("Select t from tracks where album = :album")
                .setParameter("album", "True Survivor")
                .getResultList();
    }
}
```

```java
// Schon viel besser
public interface TrackRepository extends JpaRepository<TrackEntity, Integer> {

    public List<Track> findAllByAlbum(final String name);

    public static void main(String...a) {
        TrackRepository trackRepository;
        final List<Track> tracks = trackRepository.findAllByAlbum("True Survivor");
    }
}
```

ERNSTHAFT?

```
@Entity
@SqlResultSetMapping(
        name = "ChartMapping",
        columns = {
            @ColumnResult(name = "label", type = String.class),
            @ColumnResult(name = "cnt", type = Integer.class),
            @ColumnResult(name = "chage", type = Integer.class)
        })
@NamedNativeQueries(
        @NamedNativeQuery(
                name = "ChartQuery",
                resultSetMapping = "ChartMapping",
                query = ""
                + "WITH \n"
                + "  previous_month AS\n"
                + "    (SELECT p.track_id, count(*) as cnt, \n"
                + "            dense_rank() over(order by count(*) desc) as
position  \n"
                + "       FROM plays p \n"
                + "      WHERE trunc(p.played_on, 'DD') between date'2016-04-01'
and date'2016-04-30' GROUP BY p.track_id),\n"
                + "  current_month AS\n"
                + "    (SELECT p.track_id, count(*) as cnt, \n"
                + "            dense_rank() over(order by count(*) desc) as
position \n"
                + "       FROM plays p \n"
                + "     WHERE trunc(p.played_on, 'DD') between date'2016-05-01'
and date'2016-05-31' GROUP BY p.track_id)\n"
                + "SELECT a.artist || ' - ' || t.name || ' (' || t.album || ')'
as label,\n"
                + "       current_month.cnt,  \n"
                + "       previous_month.position - current_month.position as
change\n"
                + "  FROM tracks t\n"
                + "  JOIN artists a on a.id = t.artist_id\n"
                + "  JOIN current_month current_month on current_month.track_id
= t.id\n"
                + "  LEFT OUTER join previous_month on previous_month.track_id
= t.id\n"
                + " ORDER BY current_month.cnt desc, label asc"
        )
)
public class PlayEntity {
    public static void main(String... a) {
        // Don't do this at home
        EntityManager entityManager;
        List<Object[]> results =
entityManager.createNamedQuery("ChartQuery").setMaxResults(20).getResultList();
        results.stream().forEach((record) -> {
            String label = (String) record[0];
            Integer cnt = (Integer) record[1];
            Integer change = (Integer) record[2];
        });
    }
}
```

SQL TRIFFT JAVA

```
this.create
        .with(currentMonth)
        .with(previousMonth)
        .select(label,
                currentMonth.field("cnt"),
                previousMonth.field("position").minus(
                    currentMonth.field("position")
                ).as("change")
        )
        .from(TRACKS)
        .join(ARTISTS).onKey()
        .join(currentMonth)
          .on(currentMonth.field("track_id", BigDecimal.class)
             .eq(TRACKS.ID))
        .leftOuterJoin(previousMonth)
          .on(previousMonth.field("track_id", BigDecimal.class)
             .eq(TRACKS.ID))
        .orderBy(currentMonth.field("cnt").desc(), label.asc())
        .limit(n)
        .fetch()
        .formatJSON(response.getOutputStream());
```

WAS IST JOOQ?

- „Query builder framework“
- Java DSL zur Generierung datenbankspezifischer Statements
- Das Schema ist die „treibende Kraft“
  - Generierung eines Java-Schemas (Optional, aber empfohlen)
- Typsicher
- OpenSource für OpenSource Datenbanken, $ bis $$ für Enterprise Datenbanken

Open Source

CUBRID 8.4
Derby 10.10
Firebird 2.5
H2 1.3
HSQLDB 2.2
MariaDB 5.2
MySQL 5.5
PostgreSQL 9.0
SQLite

Express

CUBRID 8.4
Derby 10.10
Firebird 2.5
H2 1.3
HSQLDB 2.2
MariaDB 5.2
MySQL 5.5
PostgreSQL 9.0
SQLite

Microsoft Access 2013 [1]
Oracle 10g Express
SQL Server 2008 Express

Professional

CUBRID 8.4
Derby 10.10
Firebird 2.5
H2 1.3
HSQLDB 2.2
MariaDB 5.2
MySQL 5.5
PostgreSQL 9.0
SQLite

Microsoft Access 2013 [1]
Oracle 10g (All editions)
SQL Server 2008 (All editions)

Amazon Redshift [4]
SQL Azure

Enterprise

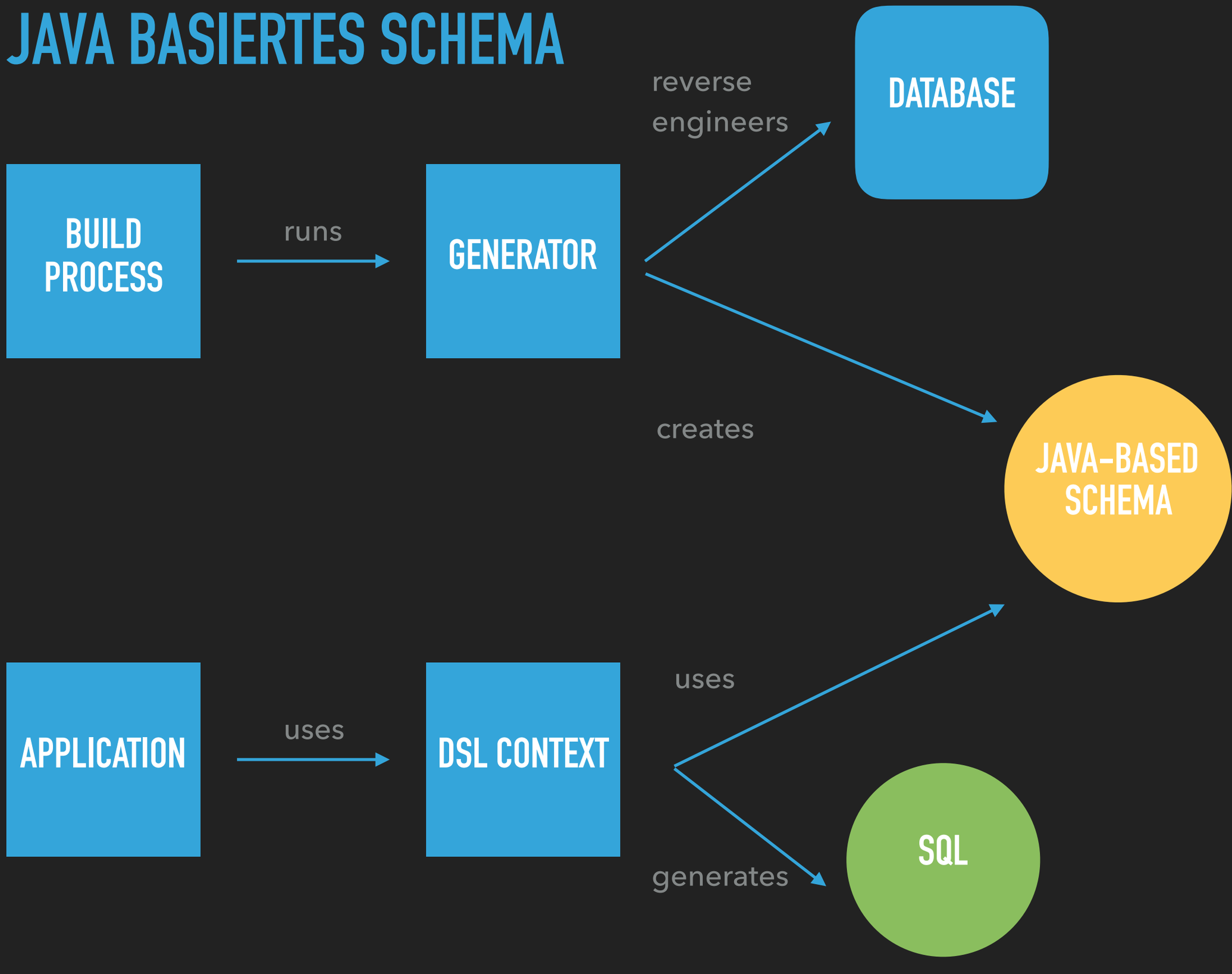
CUBRID 8.4
Derby 10.10
Firebird 2.5
H2 1.3
HSQLDB 2.2
MariaDB 5.2
MySQL 5.5
PostgreSQL 9.0
SQLite

Microsoft Access 2013 [1]
Oracle 10g (All editions)
SQL Server 2008 (All editions)

Amazon Redshift [4]
SQL Azure

DB2 LUW 9.7
HANA (All editions) [3]
Informix 12.10 [2]
Ingres 10.1
Sybase ASE 15.5
Sybase SQL Anywhere 12
Vertica 7.1 [4]

JAVA BASIERTES SCHEMA



DATENBANKMIGRATIONEN SIND ESSENTIELL

- Liquibase
- Flyway

WORKFLOW

- Build gegen Entwicklungsdatenbank
  - startet Migration
  - startet jOOQ Generator
- Anwendung gegen Produktionsdatenbank
  - startet ebenfalls Migration

➡ Java Schema „passt“ immer zur Datenbank

DEMO

ZUSAMMENFASSUNG

- Direkte Abbildung von Abfragen auf URLs
- Von einfach bis kompliziert alles möglich
  - Logging der generierten Queries ist hilfreich
- Einfache Übergabe von Parametern an Queries
- Oft benutzte Fragmente können wiederverwendet werden

THE PROBLEM WITH INTERNET QUOTES IS THAT YOU CANT ALWAYS DEPEND ON THEIR ACCURACY" - ABRAHAM LINCOLN, 1864

THE SQL... IT'S ALWAYS BEEN THERE, IT WILL GUIDE YOU

Lukas Skyeder

ANWENDUNGSFALL ANALYTISCHE FUNKTIONEN

```
WITH
  previous_month AS
   (SELECT p.track_id, count(*) as cnt,
           dense_rank() over(order by count(*) desc) as position
      FROM plays p
     WHERE trunc(p.played_on, 'DD') BETWEEN
           date'2016-04-01' and date'2016-04-30' GROUP BY p.track_id),
  current_month AS
   (SELECT p.track_id, count(*) as cnt,
           dense_rank() over(order by count(*) desc) as position
      FROM plays p
     WHERE trunc(p.played_on, 'DD') BETWEEN
          date'2016-05-01' and date'2016-05-31' GROUP BY p.track_id)
SELECT a.artist || ' - ' || t.name || ' (' || t.album || ')' as label,
       current_month.cnt,
       previous_month.position - current_month.position as change
  FROM tracks t
  JOIN artists a on a.id = t.artist_id
  JOIN current_month current_month on current_month.track_id = t.id
  LEFT OUTER join previous_month on previous_month.track_id = t.id
 ORDER BY current_month.cnt desc, label asc
 FETCH FIRST 20 ROWS ONLY;
```

ANWENDUNGSFALL „UPSERT“ / MERGE STATEMENT

```
MERGE INTO tablename USING table_reference ON (condition)
  WHEN MATCHED THEN
  UPDATE SET column1 = value1 [, column2 = value2 ...]
  WHEN NOT MATCHED THEN
  INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...]);
```

„JUST BECAUSE YOU'RE USING HIBERNATE, DOESN'T MEAN YOU HAVE TO USE IT FOR EVERYTHING.“ GAVIN KING

- Automatische Datenbankmigration (z.B. Flyway)
- JPA / Hibernate zusammen mit Spring Data JPA
- JPQL Queries falls nötig
  - Keine nativen Queries in Annotationen oder ähnlichem verstecken
- Komplexe Abfragen und Projektionen mit jOOQ erstellen
  - Als SQL an Hibernate übergeben oder
  - **den DSL Context direkt nutzen**

KENNT EURE WERKZEUGE!

- https://modern-sql.com
- https://blog.jooq.org
- https://vladmihalcea.com/tutorials/hibernate/
- http://www.thoughts-on-java.org/persistence/
- Und natürlich die jeweilige Referenzdokumentation

WAS IST ORACLE JET?

- Free and Open Source
- Kein neues Framework, vielmehr Bill Of Materials (BOM)
  - RequireJS, Knockout., jQuery UI
- Entwicklung von modernen *Enterprise* Anwendungen für
  - Desktop Browser
  - Mobile

FEATURES

- Große Auswahl an UI Components
- Accessibility support
- Internationalisierung (I18N)
- Unterstützung für hybride, mobile Anwendungen (Cordova)

BEISPIEL MODEL UND VIEW

```
define(['ojs/ojcore', 'knockout', 'jquery', 'moment', 'ojs/ojselectcombobox', 'ojs/ojchart', 'ojs/ojdatetimepicker'],
        function (oj, ko, $, moment) {
            function artistsContentViewModel() {
                    var self = this;
                    self.areaSeriesValue = ko.observableArray([]);
                    self.areaGroupsValue = ko.observableArray([]);
                    var updateCharts = function () {
                        // Fill model
                    }
                };

                self.optionChanged = function (event, data) {
                    updateCharts();
                };
            }
            return new artistsContentViewModel();
        });
```

```
<h2>Cumulative plays per artist and day</h2>
<div id='chart-container'>
    <div id="lineAreaChart" style="max-width:1024px;width:100%;height:320px;" data-bind="ojComponent: {
        component: 'ojChart',
        type: 'lineWithArea',
        series: areaSeriesValue,
        groups: areaGroupsValue,
        timeAxisType: 'enabled',
        animationOnDisplay: 'on',
        animationOnDataChange: 'on',
        stack: 'on',
        hoverBehavior: 'dim',
        zoomAndScroll: 'live',
        overview: {rendered: 'off'},
        dataCursor: dataCursorValue
     }"></div>
</div>
```

DEMO

FAZIT

- Leicht verteilbare Anwendungen / Mikroservices
  - Datenbankzentrisch, aber nicht datenbankabhängig!
- Nutzen von Datenbanktechnik und Wissen
- Wartbares, modernes UI
  - basierend auf offenen „Standards“

DANKE FÜR IHRE AUFMERKSAMKEIT!

- Dieser Vortrag:
  github.com/michael-simons/DOAG2016
- Mehr zum Einstieg in Spring Boot:
  github.com/michael-simons/NetBeansEveningCologne
- Kontakt: michael-simons.eu
- Twitter: @rotnroll666
- Gutschein für **arc42 by example**
  http://leanpub.com/arc42byexample/c/DOAG2016