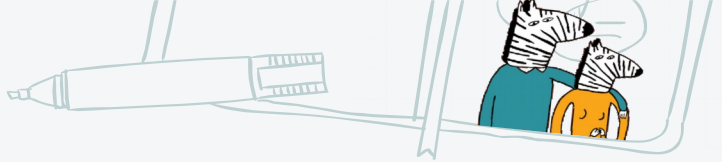
Použití PostgreSQL v zonky.cz

P2D2 – 15.2.2018
Martin Swiech
martin.swiech@zonky.cz

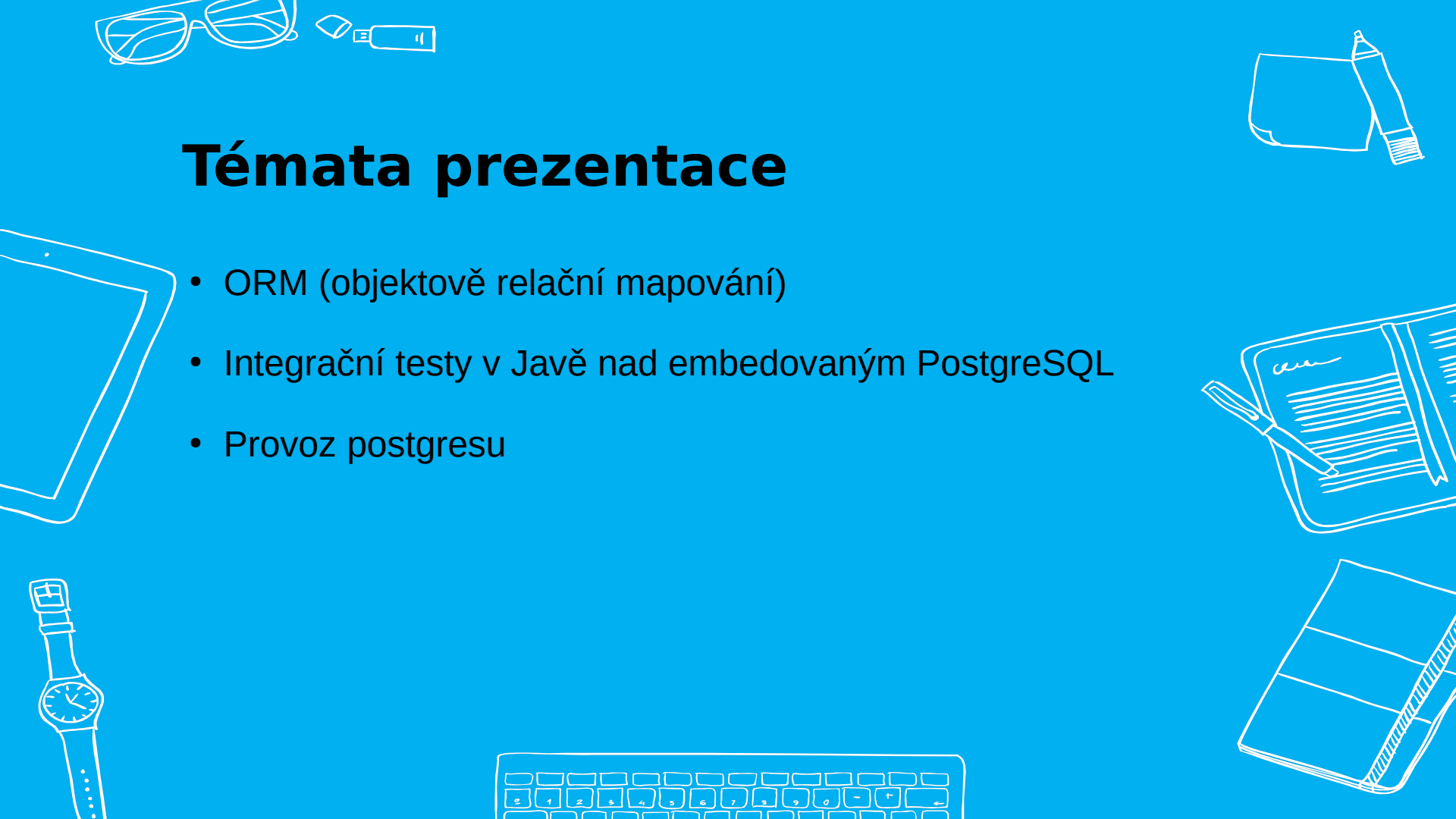


Kdo jsme?

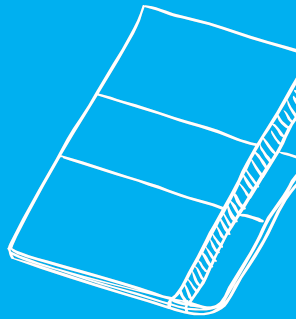
- Peer-to-peer landing platforma (lidé půjčují lidem)
- 15.000 aktivních půjček
- 16.000 investorů
- 1.500.000 investic
- BE: Java8 (Spring) + PostgreSQL 9.6
- Velikost DB cca 200 GB

Témata prezentace

- ORM (objektově relační mapování)
- Integrační testy v Javě nad embedovaným PostgreSQL
- Provoz postgresu



ORM

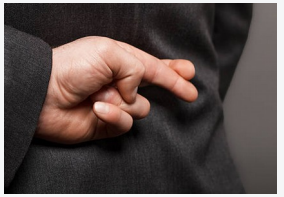


ORM

- Mapování DB relací na doménové objekty
- Dobrý sluha, zlý pán
- ORM – The Vietnam War of Computer Science
  ~Ted Neward 2006
- Zonky: Hibernate + Spring Data JPA

ORM - motivace

- Jednoduchost práce s daty
- JPA standard je součást Java EE
- Přenositelnost na jinou DB změnou konfigurace =>



```java
@Entity
@Table(name = "LOAN", schema = "P2P")
public class Loan {

    @Column(name = "contract_no", nullable = false)
    private String contractNo;

    @OneToOne
    @JoinColumn(name = "rating_id")
    private Rating rating;

    @OneToMany
    @JoinColumn(name = "loan_id")
    private List<Note> notes = new ArrayList<>();
```

ORM – obecné rady

- Vzájemné mapování doménových objektů
  (nesnažte se propojit vše se vším - buďte si vědomi hranic doménových oblastí)

- Používejte lazy fetching mapování na mapované objekty, které nepotřebujete vždy

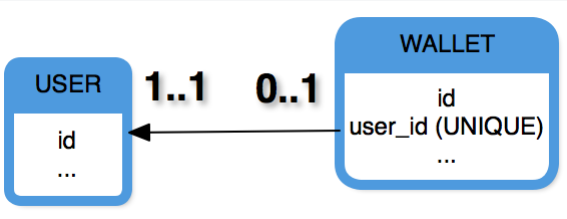
- Na složitější dotazy a operace používejte native SQL a prostředky, které vám Postgres dává

ORM – Lazy mapping

Pozor na “zpětné” one-to-one mapování

```
public class Wallet {
    @OneToOne
    @JoinColumn(name = "user_id", unique = true)
    private User user;
}

public class User {
    @OneToOne(fetch = FetchType.LAZY, mappedBy = "user")
    private Wallet wallet;
}
```



Hibernate “neví”, zda objekt Wallet v User je či není null a tak ho loaduje v samostatných dotazech, i když se jedná o Lazy mapování. Řešit se to dá použitím PROXY objektů, ale to nemusí být vždy žádoucí. Nebo otočením vazby.

ORM – Lazy mapping

Eager loading Lazy mapovaných objektů – JPQL

Klíčové slovo FETCH

```
SELECT loan FROM Loan loan
JOIN FETCH loan.borrower borrower
...
```

ORM – Lazy mapping

Eager loading Lazy mapovaných objektů – CriteriaBuilder

metoda fetch(...)

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Loan> cq = cb.createQuery(Loan.class);
Root<Loan> root = cq.from(Loan.class);
root.join( attributeName: "borrower");
root.fetch( attributeName: "borrower");
cq.select(root);
List<Loan> loans = em.createQuery(cq).getResultList();
```

ORM – Lazy mapping

Eager loading Lazy mapovaných objektů – JPA Spring Repository Loading EntityGraph

```
@NamedEntityGraphs({
        @NamedEntityGraph(
                name = Investment.ENTITY_GRAPH_GET_INVESTMENTS_WITH_INVETORS,
                attributeNodes = {
                        @NamedAttributeNode(value = "loan", subgraph = "loan"),
                        @NamedAttributeNode(value = "investor", subgraph = "user")
                },
                subgraphs = {
                        @NamedSubgraph(name = "loan", type = Loan.class, attributeNodes = {
                                @NamedAttributeNode(value = "borrower", subgraph = "user"),
                                @NamedAttributeNode(value = "ratingChangeProposal")
                        }),
                        @NamedSubgraph(name = "user", type = User.class, attributeNodes = {
                                @NamedAttributeNode(value = "wallet")
                        })
                })
})
public class Investment {
```

ORM – usecase view

Vytvoření další sady doménových objektů pro view konkrétního usecase.

Budou to @Immutable objekty nad stejnými tabulkami, jako je primární sada doménových objektů.

```
@Entity
@Table(name = "LOAN", schema = "P2P")
@SequenceGenerator(name = AbstractEntity.SQ_GENERATOR, sequenceName = "P2P.SQ_LOAN")
@AttributeOverride(name = "id", column = @Column(name = "ID"))
public class Loan extends AbstractEntity {
```

```
@Immutable
@Entity
@Table(name = "LOAN", schema = "P2P")
public class LoanQueueView extends AbstractViewEntity {
```

ORM - ladění

Zobrazení vygenerovaných SQL dotazů.

- Nastavení hibernate property `**hibernate.show_sql**` na `**true**`, případně `**hibernate.format_sql**` (persistence.xml, persistence.properties, jpaProperties EntityManagerFactoryBean)

- Nastavení loggeru na DEBUG
  <logger name="org.hibernate.SQL" level="DEBUG"/>

Testy v Javě nad PostgreSQL

Embedded PostgreSQL (for TEST)

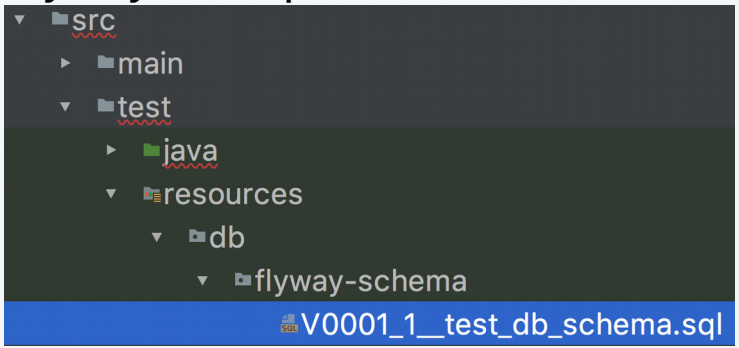
com.opentable.components:otj-pg-embedded:0.9.0
https://github.com/opentable/otj-pg-embedded

Simple example:

```
public class SimplePostgresTest {
    @Rule
    public SingleInstancePostgresRule pg = EmbeddedPostgresRules.singleInstance();
    @Test
    public void test() throws Exception {
        final DataSource dataSource = pg.getEmbeddedPostgres().getPostgresDatabase();
```

Embedded PostgreSQL (for TEST)

Flyway pro inicializaci a změnu DB schematu (používáme i produkčně)
Flyway example:




```
@Rule
public PreparedDbRule pg = EmbeddedPostgresRules.preparedDatabase(
        FlywayPreparer.forClasspathLocation("db/flyway-schema"));
```

Embedded PostgreSQL (for TEST)

Inicializace jinak než pomocí anotace @Rule

Empty:

```
PreparedDbProvider provider = PreparedDbProvider
        .forPreparer(ds -> {});
DataSource dataSource = provider.createDataSource();
...
```

Flyway:

```
PreparedDbProvider provider = PreparedDbProvider.forPreparer(
        FlywayPreparer.forClasspathLocation("db/flyway-schema"));
DataSource dataSource = provider.createDataSource();
...
```

Embedded PostgreSQL (for TEST)

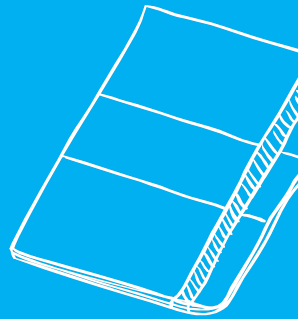
Zonky knihovna pro použití embedded Postgres se Springem:
io.zonky.test:embedded-database-spring-test:1.1.0
https://github.com/zonkyio/embedded-database-spring-test

```
@RunWith(SpringRunner.class)
@FlywayTest(locationsForMigrate = "db/test-data")
@AutoConfigureEmbeddedDatabase(beanName = "dataSource")
@ContextConfiguration(classes = SpringAppConfiguration.class)
public class SpringPostgresTest {

    @Autowired
    private SomeServiceBean someServiceBean;

    @Test
    public void test() throws Exception {
        final Long a = someServiceBean.getValue( key: "a");
        Assertions.assertThat(a).isEqualTo(1L);
```

Provoz Postgresu



Provoz Postgresu

zálohování a replikace

2x barman

postgresql.conf:

wal_level = replica

archive_mode = on

archive_command = '*rsync -a %p barman@1.1.1.1:/{barman}/zonky_incoming/%f*'

2x replika

1 hot-standby (manuální)

1 pro analytické dotazy businessu a risk managementu

Repliky nejsou napojeny přes sloty, ale v případě, že “nestihnou” stáhnout WAL z master, tak si o ně řeknou barmanovi.

recovery.conf:

recovery_target_timeline = 'latest'

restore_command = '*ssh barman@1.1.1.1 barman get-wal zonky %f > %p*'

Provoz Postgresu

server:

CPU	2 x 10 core Xeon 2.4GHz
RAM	64GB

postgresql.conf:

```
max_connections = 200
shared_buffers = 8192MB
work_mem = 64MB
maintenance_work_mem = 1GB
synchronous_commit=off
max_wal_size = 2GB
min_wal_size = 640MB
random_page_cost = 2.0
```

Provoz Postgresu

Logování dotazů

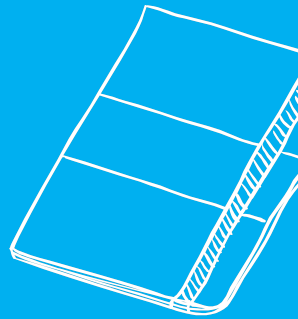
postgresql.conf:

```
log_line_prefix = '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '
log_checkpoints = on
log_min_duration_statement = 20
log_temp_files = 5
log_lock_waits = on
log_autovacuum_min_duration = 20
session_preload_libraries = 'auto_explain'
auto_explain.log_min_duration = 20
auto_explain.log_analyze = off
```

A jejich analýza

pgbadger

Praktiky



Práce s časem

- Používáme `timestamp without time zone` a ukládáme záznamy v GMT.
  date_from timestamp without time zone,
- Client si je vědom své časové zóny
- Zápis do tabulky:
  UPDATE SET date_from=now() AT TIME ZONE 'GMT'
- Čtení z tabulek:
  SELECT date_from AT TIME ZONE 'GMT'

Plánujeme

- Postgres 10
- Aplikační nastavování `***set work_mem=....;***` pro složitější úkoly
  - dopad: analýza a úprava connection poolu, aby prováděl `***RESET ALL;***` při vracení connection do poolu
- Automatické testy obnovitelnosti zálohy z Barmana

Otázky (a odpovědi)

Děkuji za pozornost.

Zdroje a odkazy

- Prezentace “ORM & how not to get crazy” by Dominik Mostek
- https://github.com/opentable/otj-pg-embedded
- https://github.com/flyway/flyway
- https://github.com/zonkyio/embedded-database-spring-test

Examples:

- https://github.com/mswiech/pgtestdemo

martin.swiech@zonky.cz [or @gmail.com]