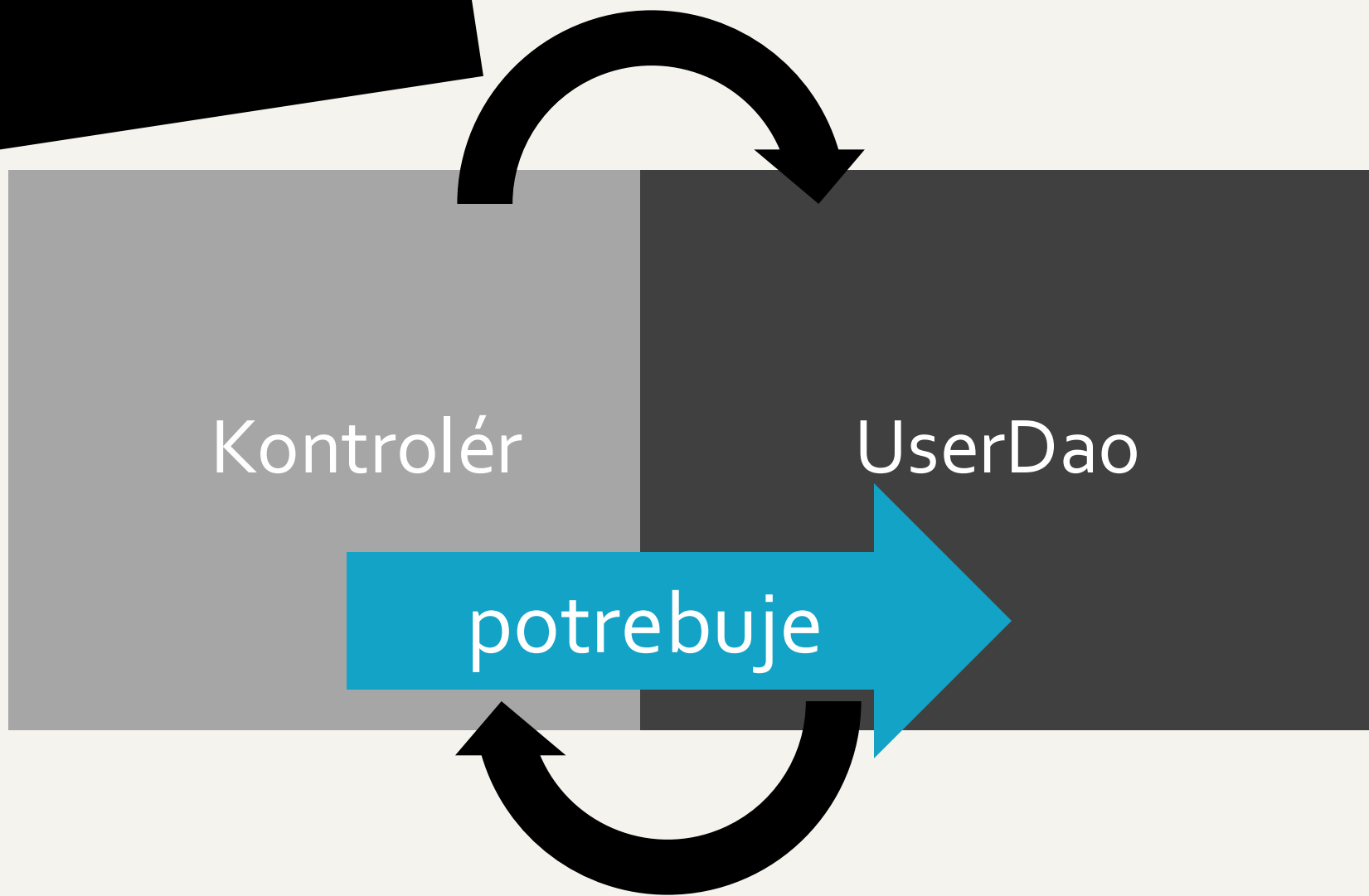


Kontrakty,
Továrne,
Jedináčikovia,
Vymenované typy,
Databázy a JDBCTemplate,
Docker

UINF/PAZ1c
4. prednáška



prepojenie
GUI a DAO



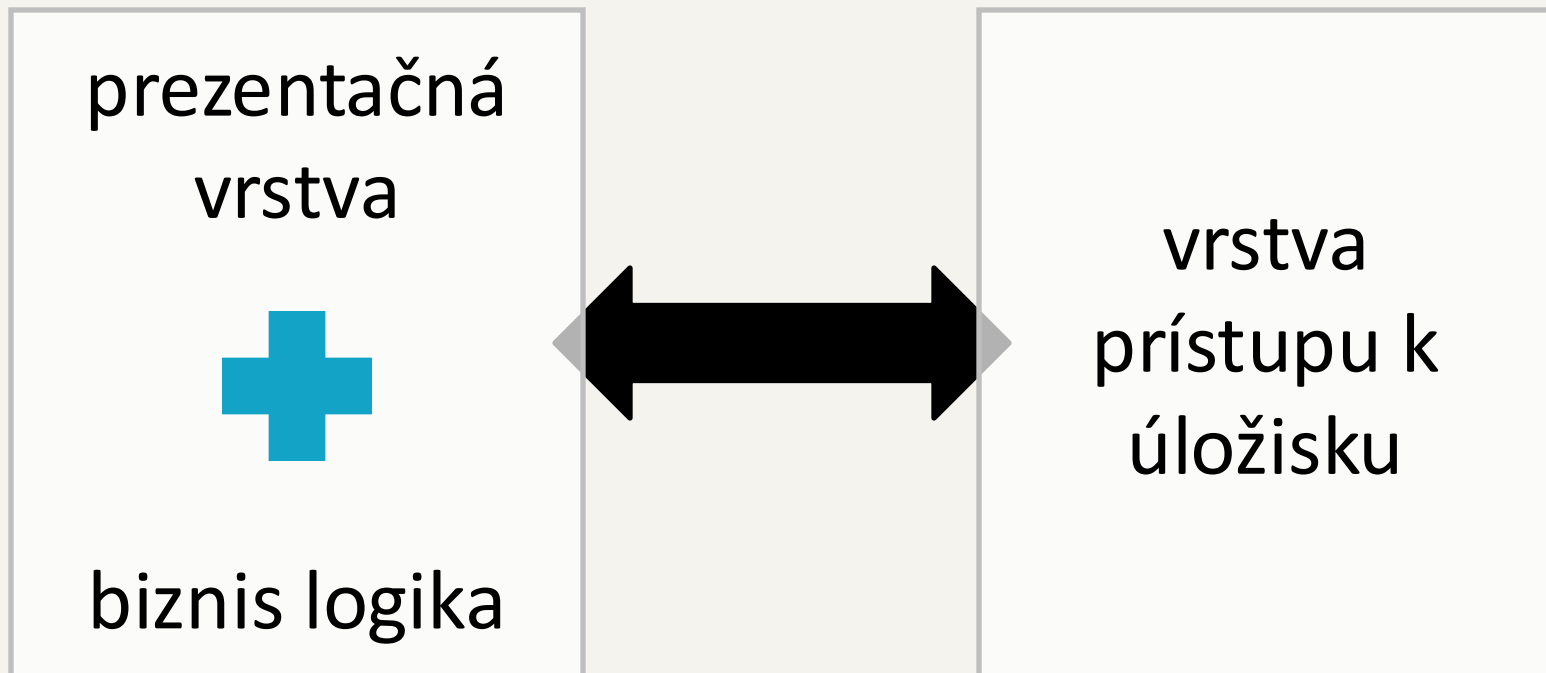
dependency

objekt vyžaduje pre fungovanie iný objekt

kolaborácia

závislosť

2vrstvové aplikácie



kód:
inštančná
premená

```
class EntranceMainController {  
    private UserDao userDao;  
    ...  
}
```

```
class UserDao {  
    ...  
}
```

kód

```
class EntranceMainController {  
    private UserDao userDao = new UserDao();  
    ...  
}
```

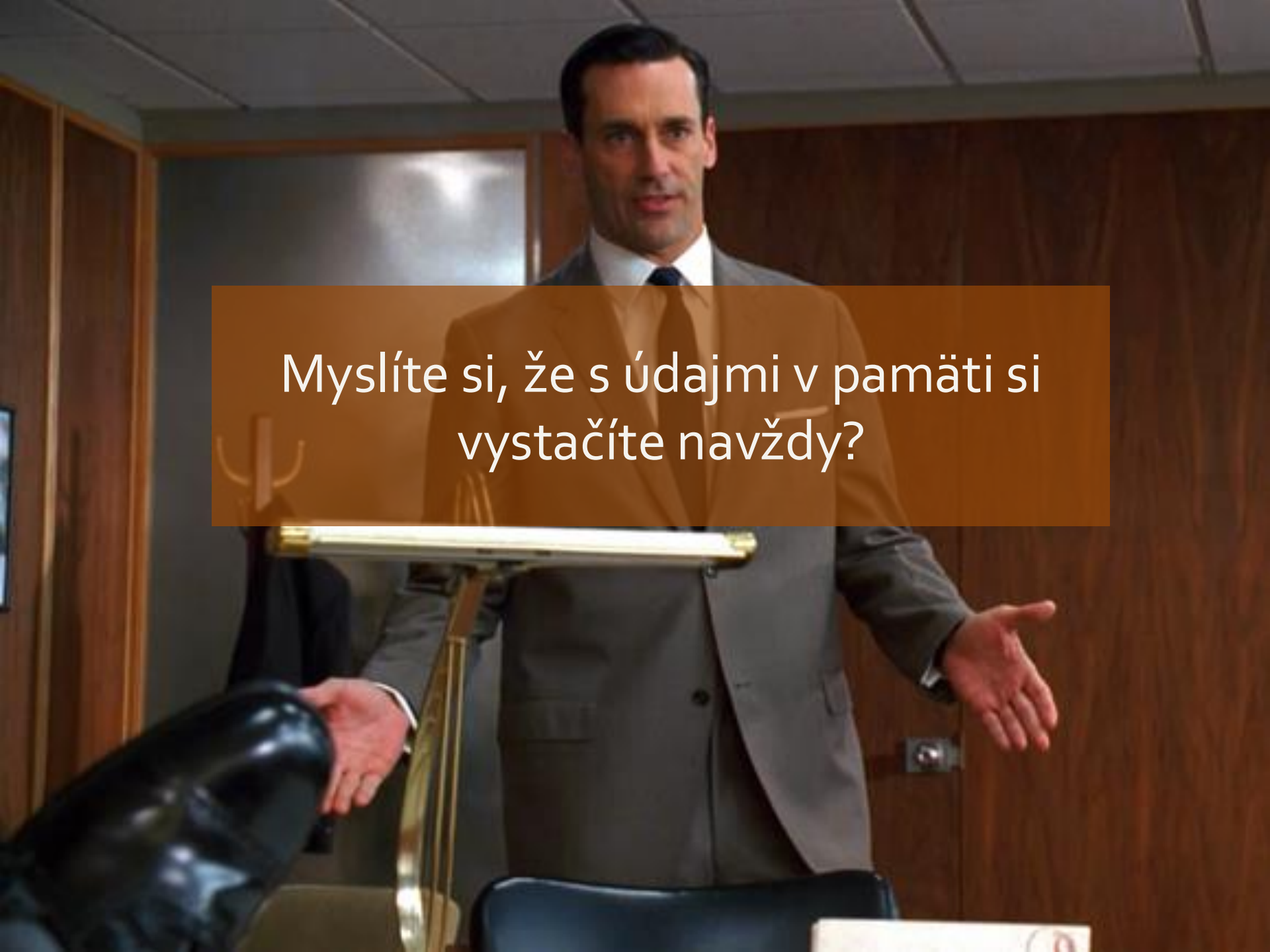
trieda si vytvorí závislosť sama

„jeden modul programu vyrába iný“



trieda sa zadrôtuje s implementáciou

čo ak chceme
implementáciu meniť?

A man in a grey suit, white shirt, and dark tie stands behind a gold-colored podium in a conference room. He is gesturing with his hands, looking towards the camera. The room has wood-paneled walls and a window in the background. A black leather chair is visible in the foreground.

Myslíte si, že s údajmi v pamäti si
vystačíte navždy?



migrujme
použivateľov
na disk / do databázy



vymieňame
perzistentnú vrstvu!

refaktor





zahodíme UserDao nad pamätou?



uvažujme
úplne inak



Honeywell
**FLIGHT
RECORDER
DO NOT OPEN**

Demo Unit
For Display Only

trieda je čierna skrinka

čo, nie ako

pri návrhu triedy začíname
od schopností

implementácia až v druhom
rade

spomeňme na TDD – najprv testy
nad prázdnyimi metódami

CONTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis sodales magna sed pretium. Proin ac diam nec velit tempus blandit. Etiam lobortis auctor ligula eget dignissim. Mauris et ultrices sem. Nunc tincidunt massa nec arcu consequat, in dignissim sem imperdiet. Curabitur venenatis id pretium condimentum, lacus ligula blandit turpis, id faucibus nibh mauris eu ante. Proin vel iaculis interdum diam turpis, sodales luctus neque lobortis nec. Suspendisse at ipsum eget tortor.

vonkajší pohľad na triedu

Nulla ipsum felis, gravida id velit eget, maximus tincidunt nulla. Fusce sit amet erat diam. Nullam tincidunt arcu placerat aliquam lacinia. Aenean ullamcorper lacus metus, non varius sapien faucibus sit amet. Curabitur pulvinar ornare venenatis. Aliquam ultricies orci vitae eros maximus tincidunt. Duis viverra leo lectus, tristique dignissim urna iaculis sed. Vivamus diam diam, efficitur sit amet ultricies in, luctus eget justo. Aliquam mollis euismod leo, a fermentum ante vestibulum ut. Nullam est ipsum, volutpat tristique tempus id, dapibus non velit. Fusce vestibulum purus dui, scelerisque tempus libero imperdiet sed. sit amet luctus maximus. Praesent fermentum suscipit placerat. ut molestie eros lacinia. Pellentesque finibus, magna odio sit amet lacus. bicula. Nulla

klient
slušné správanie

kontrakt

trieda
prísľuby
zodpovednosti
garancie

interfejs

sada operácií, ktoré objekt
poskytuje navonok

= rozhranie

= protokol

= kontrakt

```
public interface UserDao {  
    User findById(Integer id);  
    List<User> findByName(String name);  
    User findById(String chipId);  
    List<User> activeUsers();  
    void save(User user);  
    void delete(User user);  
}
```

ktokoľvek vie
naplniť kontrakt

```
public class MemoryUserDao implements UserDao
{
    private List<User> users = new ArrayList<>();

    public User findById(Integer id) {
        for(int i = 0; i < users.size(); i++)
            if (users.get(i).getId()==id)
                return users.get(i);
        return null;
    }
    /* implementácie ostatných metód */
}
```

ktokoľvek vie
naplniť kontrakt

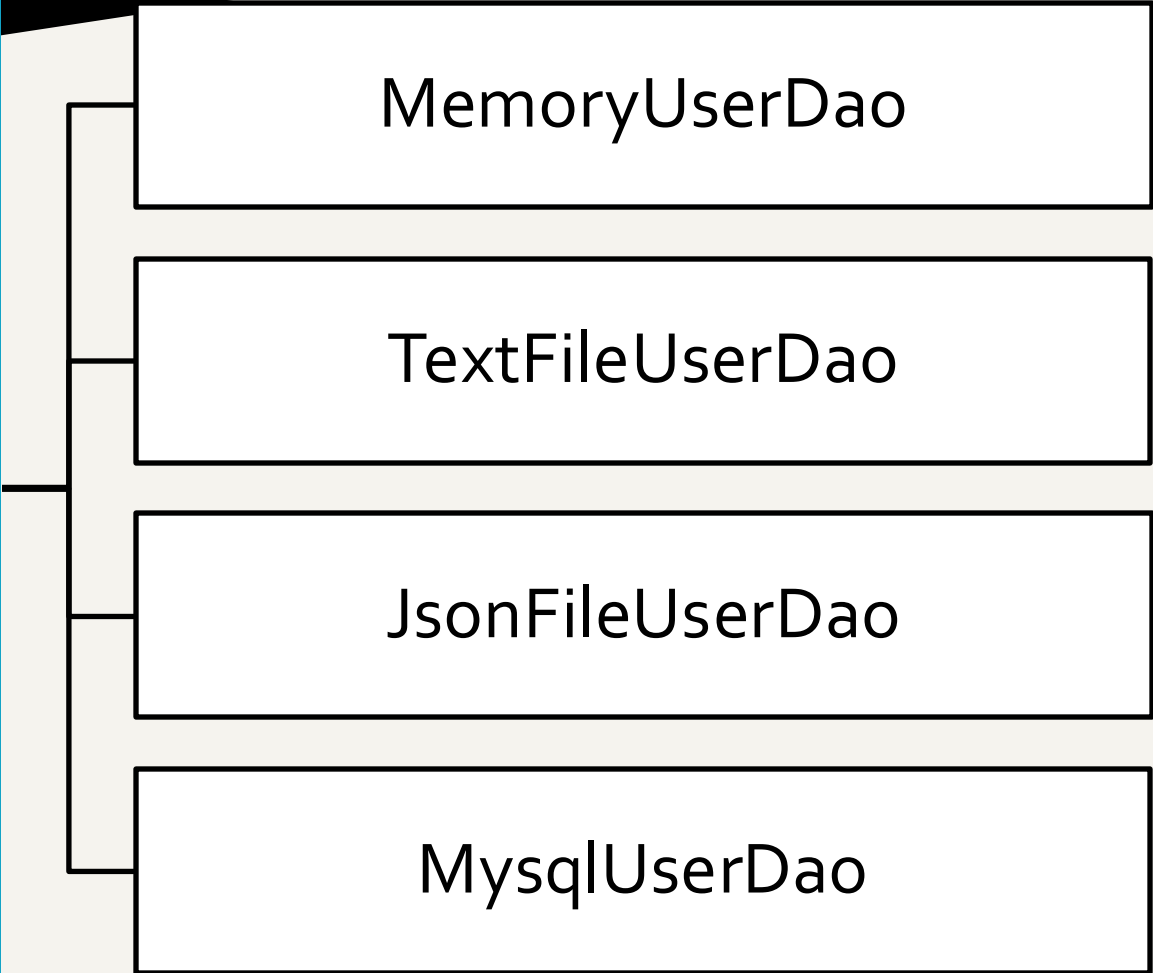
UserDao

MemoryUserDao

TextFileUserDao

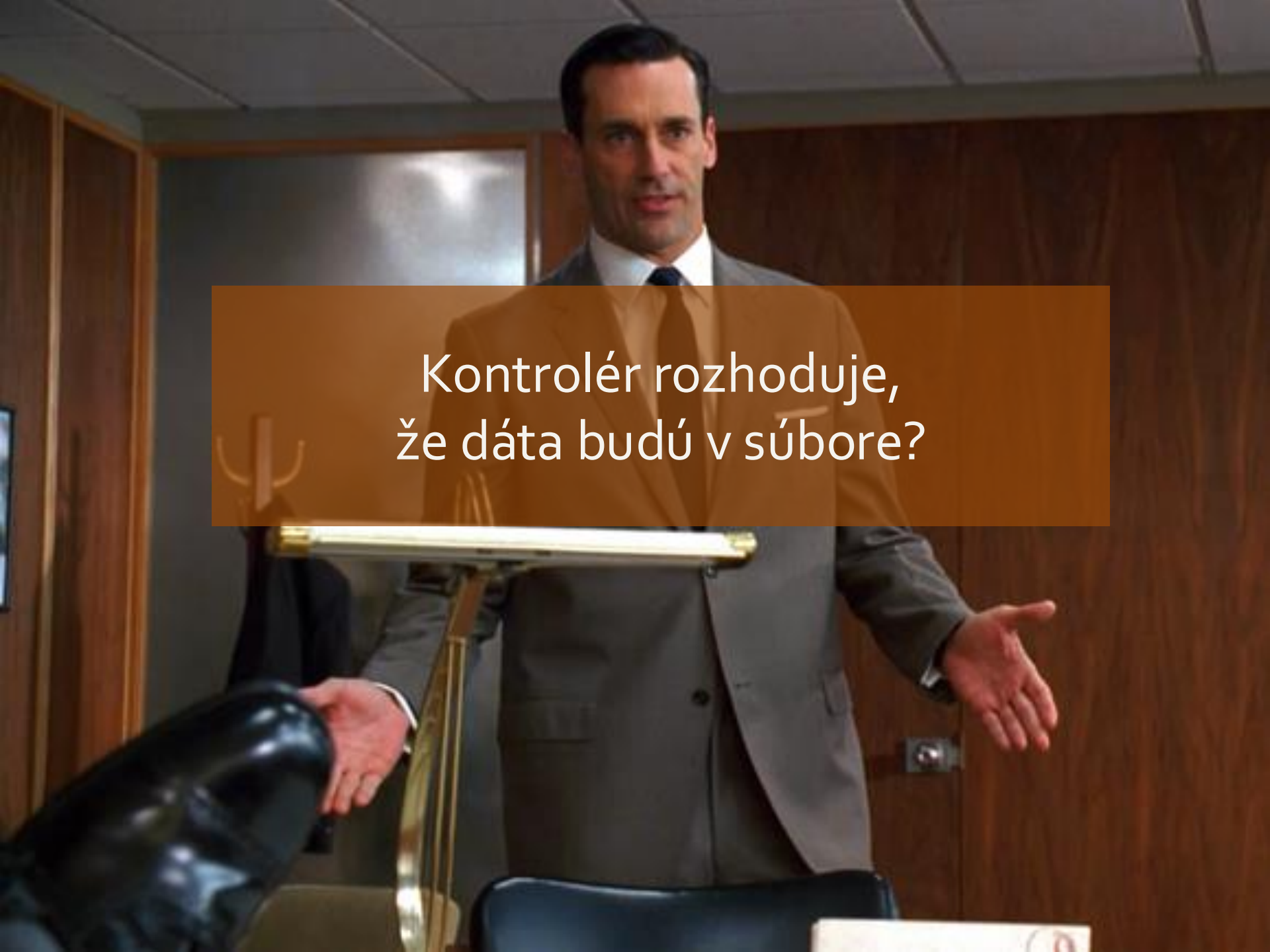
JsonFileUserDao

MysqlUserDao



použitie

```
public class EntranceMainController {  
    private UserDao userDao  
        = new TextFileUserDao();  
    ...  
}
```

A man in a grey suit and tie stands behind a gold podium in a wood-paneled conference room. He is gesturing with his hands as if speaking. A semi-transparent brown box is overlaid on the image, containing white text.

Kontrolér rozhoduje,
že dáta budú v súbore?

Čo ak máme viac tried v GUI?

- Viac scén a kontrolérov, „vyskakovacie okná“
- Každá trieda z GUI bude mať inštančnú premennú DAO
 - Bude si každá trieda GUI vyrábať vlastnú perzistentnú vrstvu?
 - Budú si musieť objekty hovoriť medzi sebou referenciu na spoločné DAO?
 - A ktorý z nich ju má vyrábať?



Tight Coupling

- úzka zviazanosť tried
- priveľká závislosť
- krehké pri zmenách





HOLLYWOOD

PRINCIPLE

Nevolajte nám, my sa vám ozveme.

The image shows a grassy hillside with the word "HOLLYWOOD" written in large, white, block letters. The letters are slightly weathered and appear to be made of a material like concrete or stone. The background consists of green grass and some trees under a clear blue sky.

HOLLYWOOD

PRINCIPLE

Trieda poprosí niekoho, nech jej vyrobí a dodá kontraktorov.

továreň

objekt, ktorý vyrába iné objekty

továreň

- Rozhoduje, ktorá implementácia interfejsu sa vyberie
 - pamäť, súbor, databáza, ...
- **Našteluje danú implementáciu**
 - login, heslo, umiestnenie, ...

...Factory

```
public class UserDaoFactory {  
    public UserDao getUserDao() {  
        /* tu sa vráti inštancia vybranej  
        implementácie UserDao*/  
    }  
}
```

```
getUserDao()
```

nový objekt?

rovnaký
objekt?

z poolu?

akého typu?

použitie továrne

kontrolér potrebuje DAO



továreň vytvorí DAO



kontrolér potrebuje
továreň

trieda potrebuje továreň

```
public class EntranceMainController {  
    private UserDaoFactory factory  
        = new UserDaoFactory();  
}
```

trieda potrebuje továreň

```
public class EntranceMainController {  
    private UserDaoFactory factory  
        = new UserDaoFactory();  
}
```

lolwut?



„Každý problém v
informatike možno
vyriešiť pridaním ďalšej
úrovne nepriameho
prístupu / abstrakcie.“
– Butler Lampson, 1972

A photograph of a man with a shaved head, wearing a maroon shirt, sitting in a chair and covering his face with his hands. The background is a plain, light-colored wall.

**I had a problem so I
thought to use Java**

Now I have a ProblemFactory



kolko tovární
potrebujeme?

FROM ANOTHER TIME COMES A MAN
OF GREAT POWER.
A MAN OF INCREDIBLE STRENGTH.
AN IMMORTAL ABOUT TO FACE HIS
GREATEST CHALLENGE...

SINGLETON

jediná inštancia v
systéme

Special Edition Music by QUEEN Music Score by MICHAEL KAMEN Executive Producer E.C. MONELL Produced by PETER
SOUNDTRACK AVAILABLE ON TWO RECORDS & TAPE RELEASED BY COLUMBIA TRISTAR RECORDS
© CANNON

FEATURING
ORIGINAL SONGS
BY
QUEEN

Joshua Bloch

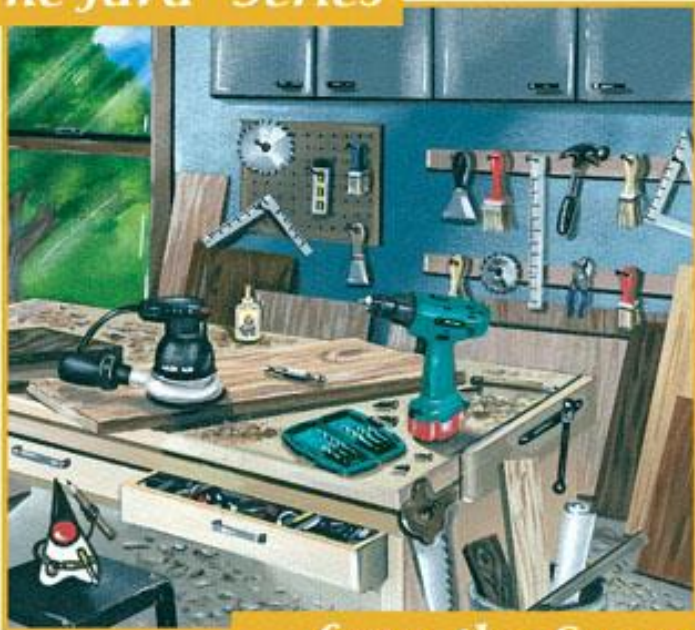
Revised and
Updated for
Java SE 6



Effective Java™

Second Edition

The Java™ Series



...from the Source



Rada 3: singleton
cez privátny
konštruktor alebo
enum

továreň v kóde

```
public enum UserDaoFactory {  
    INSTANCE;
```

```
    public UserDao getUserDao() {
```

```
        /* ... */
```

```
    }
```

```
}
```


továreň v kóde

```
UserDao userDao =  
    UserDaoFactory.INSTANCE  
    .getUserDao();
```

hack,
ale
korektný



migrácia: 1 riadok!

```
public enum UserDaoFactory {  
    INSTANCE;
```

```
    public UserDao getUserDao() {  
        return new MemoryUserDao();  
        return new MysqlUserDao();  
    }  
}
```

prístup k databázam

A black and white photograph of a modern building's facade. The word "ORACLE" is prominently displayed in large, raised, sans-serif capital letters on a curved section of the building. Below the text, several horizontal metallic bands or railings are visible, creating a rhythmic pattern. The building's surface is highly reflective, mirroring the surrounding environment. The overall composition is clean and architectural.

ORACLE

Relačné databázy

dominantný spôsob ukladania dát

hojný výskum

kilá implementácií

Naša entita User v relačnej databáze

```
CREATE TABLE user (  
  id INT NOT NULL AUTO_INCREMENT,  
  chip_id VARCHAR(20) NOT NULL,  
  name VARCHAR(45) NOT NULL,  
  active BOOLEAN,  
  PRIMARY KEY (`id`)  
);
```

Ako prepojiť
Javu a DB?



JDBC = Java API
pre SQL



IT'S A CRAP!

JDBC

- Funkčné, ale ťažkopádne
- Potenciálne nebezpečné
 - Nutnosť korektne obsluhovať a uvoľňovať prostriedky
 - V tutoriáloch sa kvôli prehľadnosti ignorujú závažné chyby

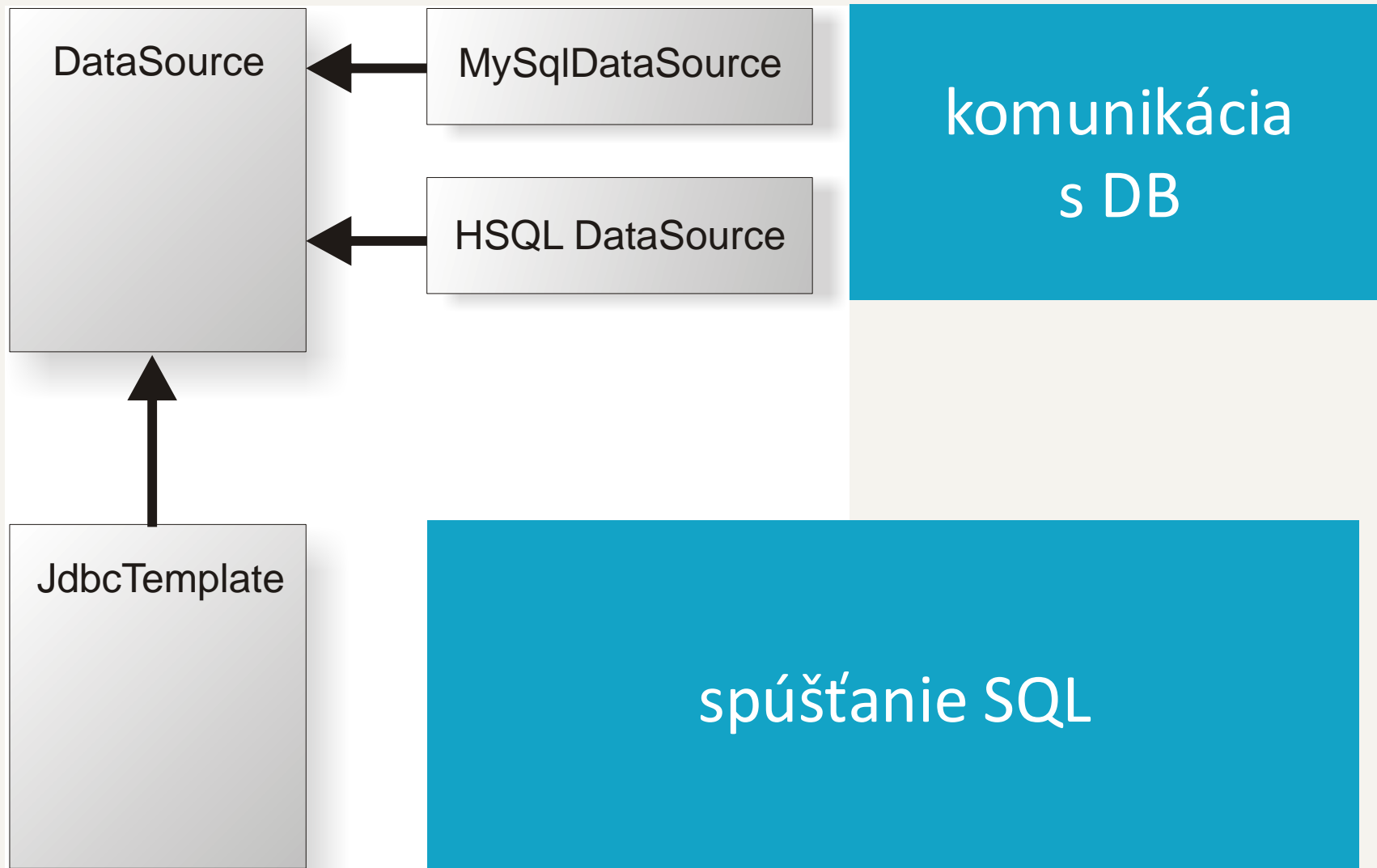
DB spojenia

- Časté vytváranie spojenia je drahé
 - Bežne chceme niekoľko (aj tisíce) dopytov za sekundu
 - Robiť všetko cez jedno trvalé spojenie je pomalé
 - Trpíme: „head of line blocking“ – krátke dopyty zbytočne čakajú na predchádzajúce dlhé operácie
- Typicky databáza dovolí max. 20 - 50 paralelných spojení
 - Potrebujeme to nejakto spravovať

Spring
Framework

„Budujme lepšie enterprise aplikácie“

Spring JDBC



JdbcTemplate v Mavene

mysql-connector-j

spring-jdbc

Inicializácia

```
MySQLDataSource dataSource = new MySQLDataSource();
```

```
dataSource.setDatabaseName(„entrance_system“);
```

```
dataSource.setUser("paz1c");
```

```
dataSource.setPassword("paz1clsGreat");
```

```
JdbcTemplate jdbcTemplate = new
```

```
    JdbcTemplate(dataSource);
```

Dopyty

```
String sql = "SELECT id, chip_id, name, active FROM user";  
RowMapper<User> rowMapper = new UserRowMapper();  
List<User> users = jdbcTemplate.query(sql, rowMapper);
```

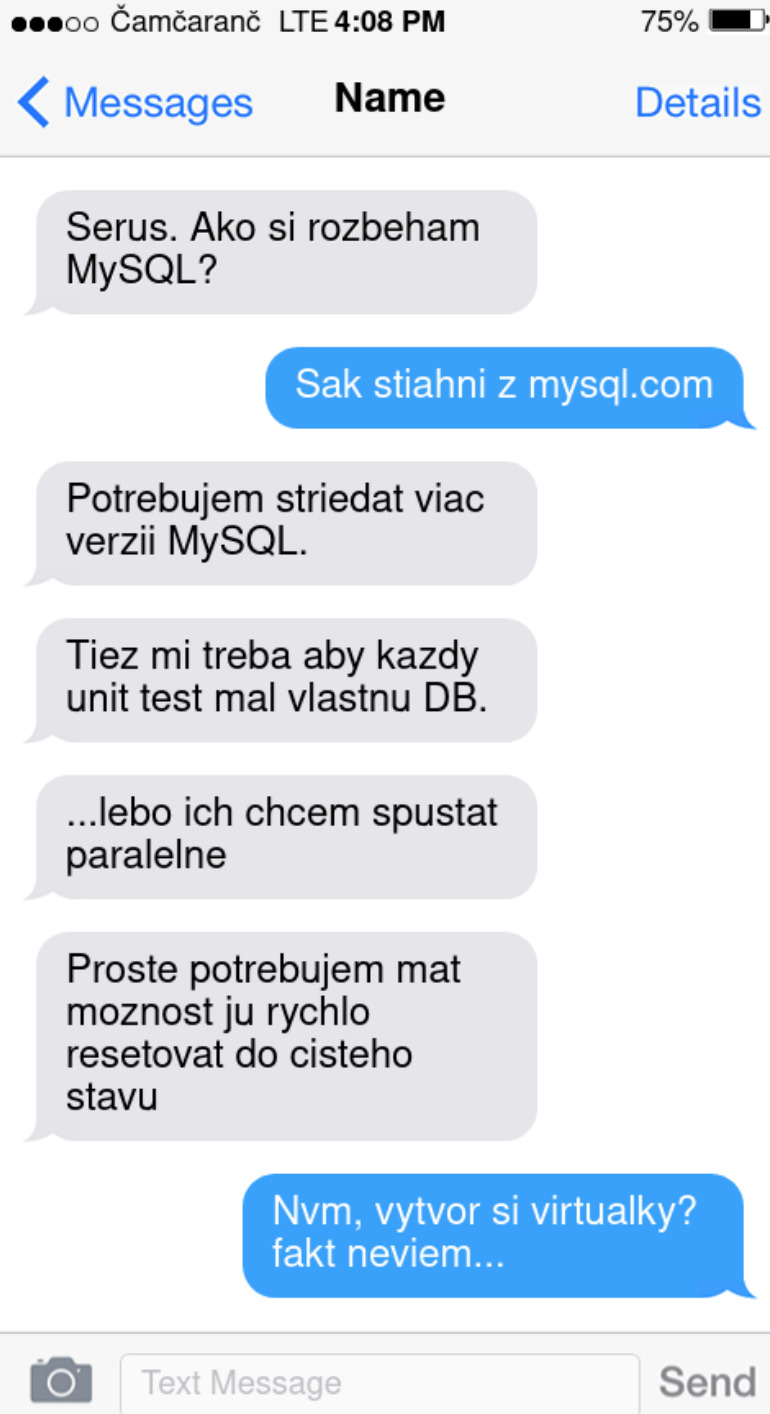
- Pre každý riadok výslednej tabuľky sa zavolá metóda `mapRow()` objektu `rowMapper`
 - mapovač vráti jednu inštanciu triedy `User`
- Metóda `query()` vyzbiera všetky inštancie a vráti ich ako zoznam

Dopyty

```
String sql = "SELECT id, chip_id, name, active FROM user";  
RowMapper<User> rowMapper = new UserRowMapper();  
List<User> users = jdbcTemplate.query(sql, rowMapper);
```

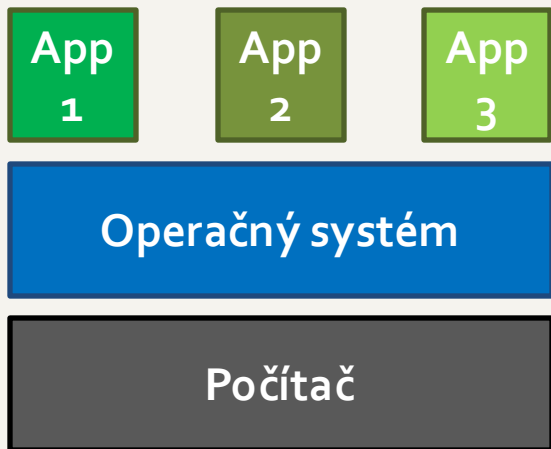
```
public class UserRowMapper implements RowMapper<User> {  
    public User mapRow(ResultSet rs, int rowNum)  
        throws SQLException {  
        User user = new User();  
        user.setId(rs.getInt("id"));  
        /* ďalšie volania setterov */  
        return user;  
    }  
}
```


Spúšťanie DB

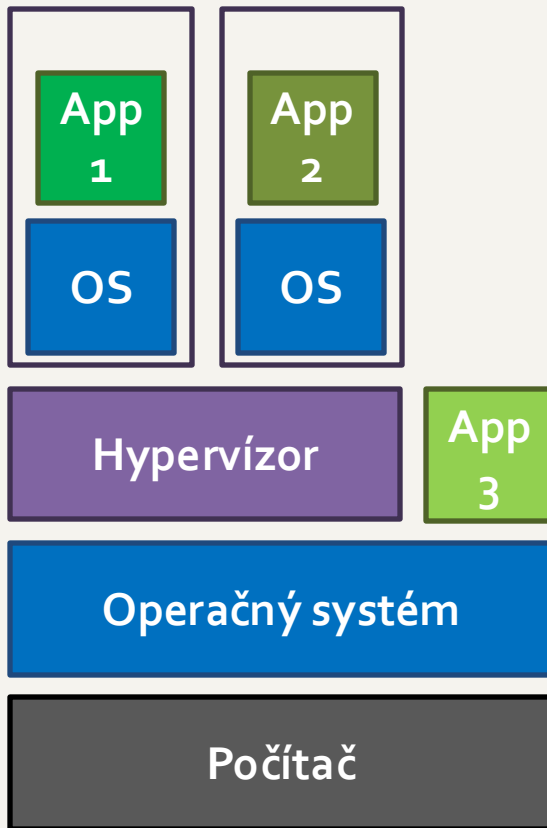


Kontajnerizácia

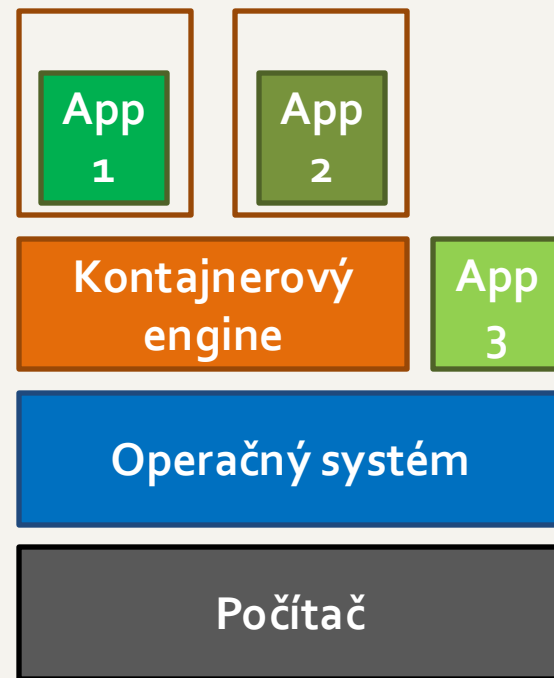
Bežný PC



Virtualizácia
(typu 2)



Kontajnerizácia





- Docker je kontajnerizačný nástroj
 - kontajner - "odľahčená virtuálka"
- Moderná appka potrebuje viacero iných appiek
 - MySQL/PostgreSQL, Redis, Kafka, ElasticSearch, ...
- Docker umožní na jeden klik/príkaz tieto appky
 - sťahovať, spúšťať, vypínať, **resetovať**...
 - skvelé aj na testy (cez TestContainers knižnicu)

MySQL cez Docker Compose

docker-compose.yml

```
services:
  mysql:
    image: mysql:8.4.2
    environment:
      MYSQL_ROOT_PASSWORD: superheslo
      MYSQL_DATABASE: entrance
      MYSQL_USER: entrance
      MYSQL_PASSWORD: hesielko
    ports:
      - "3306:3306"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

stiahni nejaký MySQL

konfigurácia

mapuj port host<-kontajner

(voliteľne) sprístupni súbor init.sql do magického priečinka `/docker-entrypoint-initdb.d` v kontajneri. MySQL ho načíta pri prvom spustení.

Spustíme príkazom `docker compose up`

alebo `docker compose up -f docker-compose.yml`

Otázky?