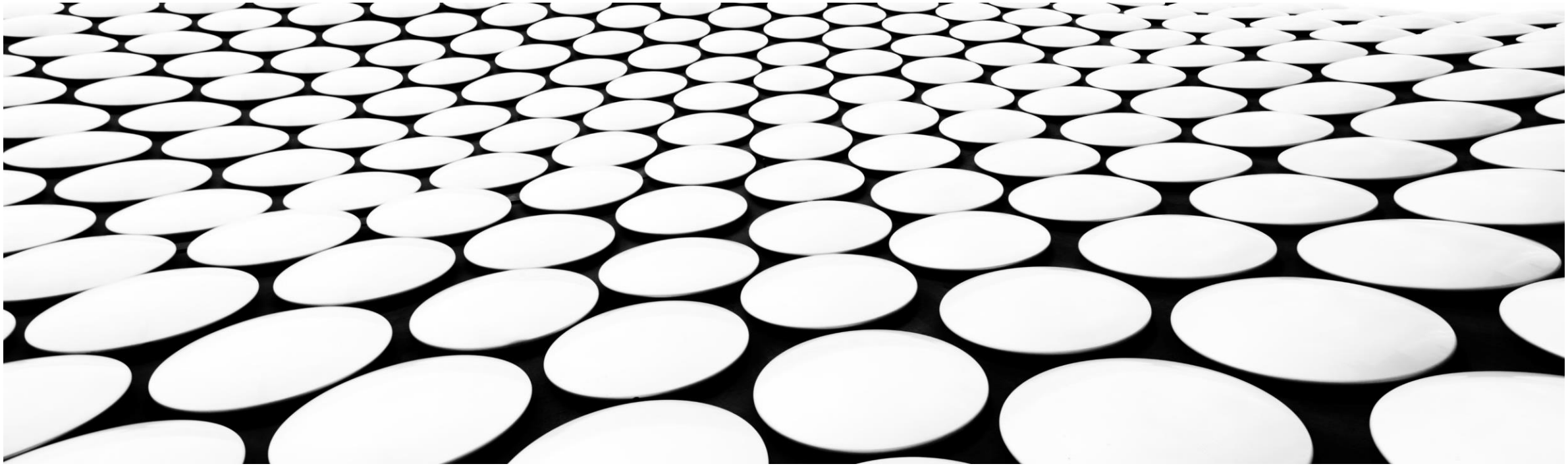
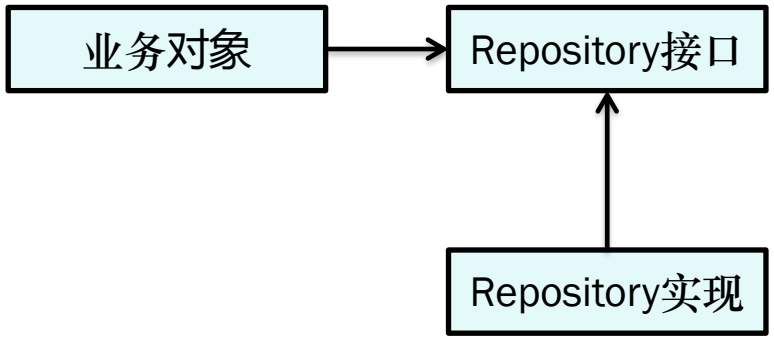


服务端开发-Spring Data JDBC、JPA

陶召胜

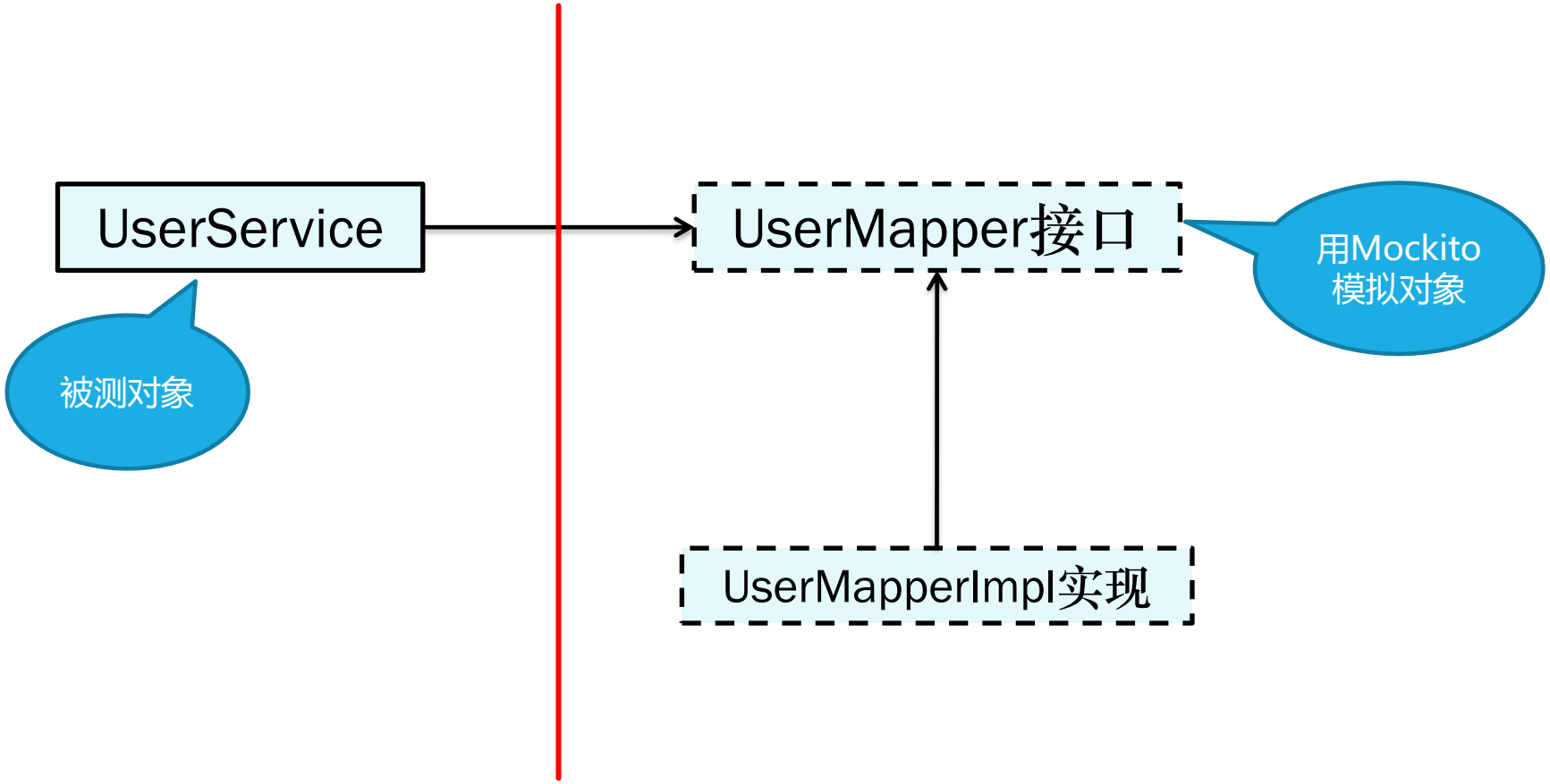
数据访问层开发

- 关系型数据库
  - ✓ 使用JdbcTemplate简化JDBC访问（spring-boot-starter-jdbc）
  - ✓ Spring Data JDBC（spring-boot-starter-data-jdbc）
  - ✓ Spring Data JPA（spring-boot-starter-data-jpa）
  - ✓ 与MyBatis、 Hibernate等集成
- 非关系型数据库（ NoSQL）
  - ✓ spring-boot-starter-data-mongodb
  - ✓ spring-boot-starter-data-redis
  - ✓ 等



数据访问层对象模拟

- 数据访问对象（data access object ,DAO)或Repository



内容

1. 使用JdbcTemplate简化JDBC访问（spring-boot-starter-jdbc）
2. Spring Data JDBC（spring-boot-starter-data-jdbc）
3. Spring Data JPA（spring-boot-starter-data-jpa）

使用原始的JDBC访问数据库

- RawJdbcIngredientRepository
- 样板式代码（ResultSet、PreparedStatement、Connection）
- SQLException，checked异常

异常体系

- SQLException
  - ✓ 发生异常时很难恢复
  - ✓ 难确定异常类型
- Hibernate异常
  - ✓ 定义了许多具体异常，方便定位问题
  - ✓ 对业务对象的侵入
- Spring所提供的平台无关的持久化异常
  - ✓ DataAccessException
  - ✓ 具体异常，方便定位问题
  - ✓ 隔离具体数据库平台

使用JdbcTemplate

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

配置生成的H2数据库名，用于console访问

```
spring:
  datasource:
    generate-unique-name: false
    name: tacocloud
```

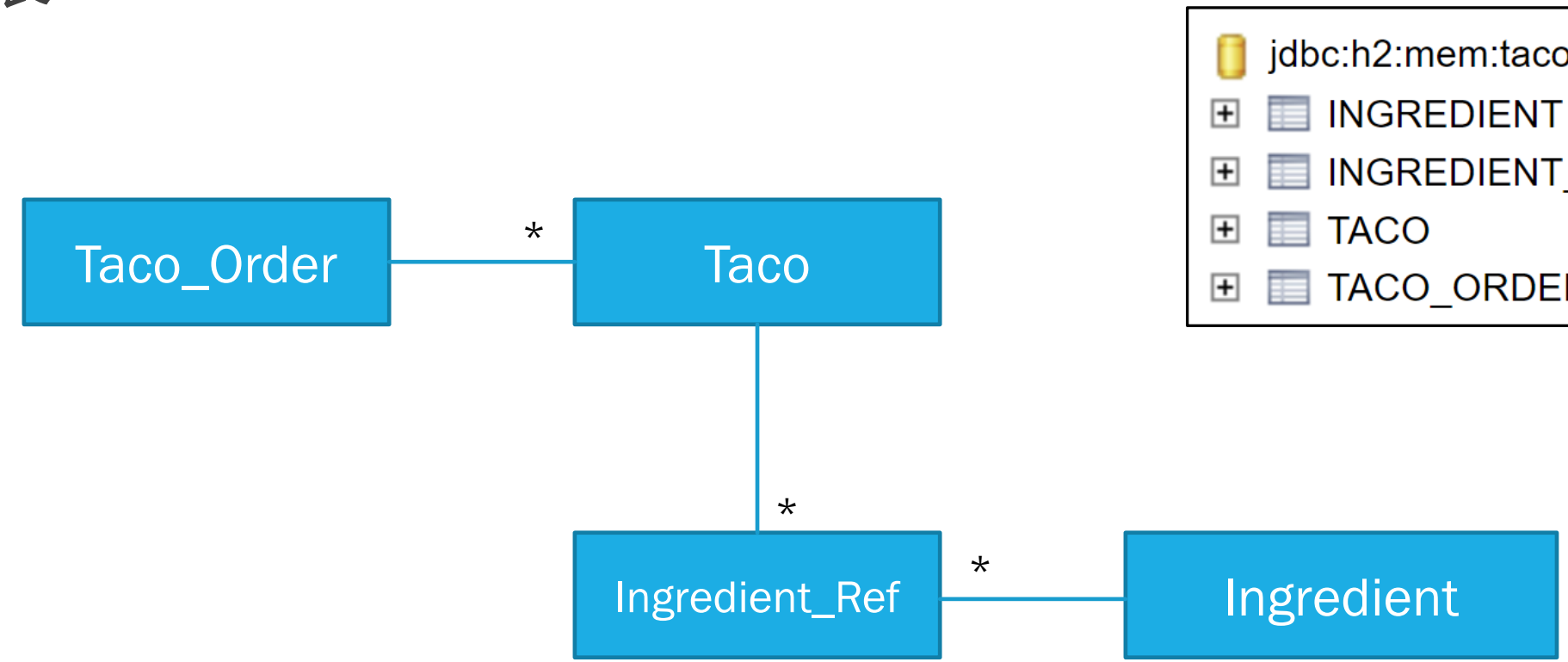
H2访问

- http://localhost:8080/h2-console
- 驱动：org.h2.Driver
- JDBC URL：jdbc:h2:mem:tacocloud
- 用户名：sa

IngredientRepository的实现

- 注入JdbcTemplate，如果只有一个构造方法可以省去@Autowired
- @Repository
- 接口：RowMapper<T>，可以使用lambda表达式
- 注入DesignTacoController，使用
- IngredientByIdConverter实现优化

数据库表


Taco_Order
*
Taco
*
Ingredient_Ref
*
Ingredient

jdbc:h2:mem:tacocloud
INGREDIENT
INGREDIENT_REF
TACO
TACO_ORDER

数据库表创建与数据初始化

- schema.sql-表创建
- data.sql-数据初始化

save(TacoOrder order)的实现

- Taco不能脱离TacoOrder而存在，聚合关系
- JdbcOrderRepository
- identity字段由数据库自动生成值，获取返回的ID，GeneratedKeyHolder
  - ✓ PreparedStatementCreatorFactory
  - ✓ PreparedStatementCreator
  - ✓ jdbcOperations.update
- 注入OrderController，使用

内容

1. 使用JdbcTemplate简化JDBC访问（spring-boot-starter-jdbc）
2. Spring Data JDBC（spring-boot-starter-data-jdbc）
3. Spring Data JPA（spring-boot-starter-data-jpa）

Spring Data项目

- Spring Data JDBC
- Spring Data JPA
- Spring Data MongoDB
- Spring Data Neo4j
- Spring Data Redis
- Spring Data Cassandra

Spring Data JDBC

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>
```

定义持久化接口

```
import org.springframework.data.repository.CrudRepository;

public interface IngredientRepository
        extends CrudRepository<Ingredient, String>
```

为领域类添加持久化的注解

- @Table，对象会基于领域类的名称映射到数据库的表上
  - ✓ TacoOrder会映射到Taco_Order表
- @Id
- @Column
  - ✓ deliveryName会映射到delivery_Name列

程序预加载

- org.springframework.boot.CommandLineRunner
- org.springframework.boot.ApplicationRunner

内容

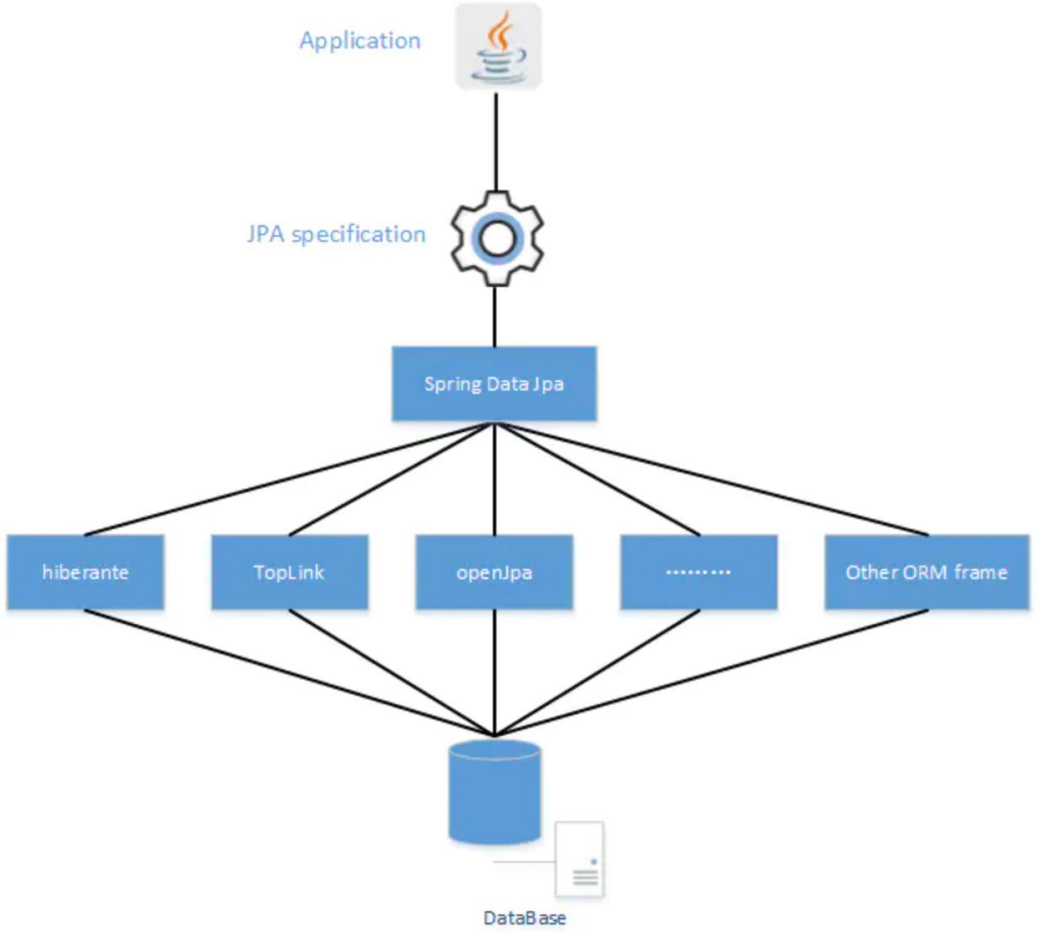
1. 使用JdbcTemplate简化JDBC访问（spring-boot-starter-jdbc）
2. Spring Data JDBC（spring-boot-starter-data-jdbc）
3. Spring Data JPA（spring-boot-starter-data-jpa）

Spring Data JPA

- JPA：Java Persistence API
- JPA的宗旨是为POJO提供持久化标准规范
- JPQL是一种面向对象的查询语言
- 依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

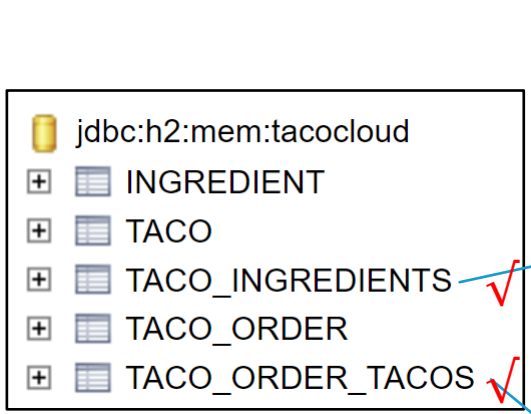
Jpa、Hibernate、Spring Data Jpa三者之间的关系



@Entity

- javax.persistence.*

自动生成的数据库表

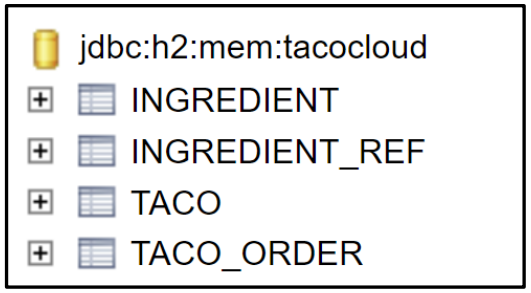


SELECT * FROM TACO_INGREDIENTS;

TACO_ID	INGREDIENTS_ID
2	FLTO
2	CHED
2	TMTO
2	LETC
3	CARN
3	CHED
3	JACK
3	TMTO

(8 rows, 2 ms)

与之前jdbc的对比



SELECT * FROM TACO_ORDER_TACOS;

TACO_ORDER_ID	TACOS_ID
1	2
1	3

(2 rows, 3 ms)

自定义的查询方法

- 定义查询方法，无需实现
  - ✓ 领域特定语言（domain-specific language，DSL），spring data的命名约定
  - ✓ 查询动词 + 主题 + 断言
  - ✓ 查询动词：get、read、find、count
  - ✓ 例子：

```
List<TacoOrder> findByDeliveryZip( String deliveryZip );
```

- 声明自定义查询
  - ✓ 不符合方法命名约定时，或者命名太长时

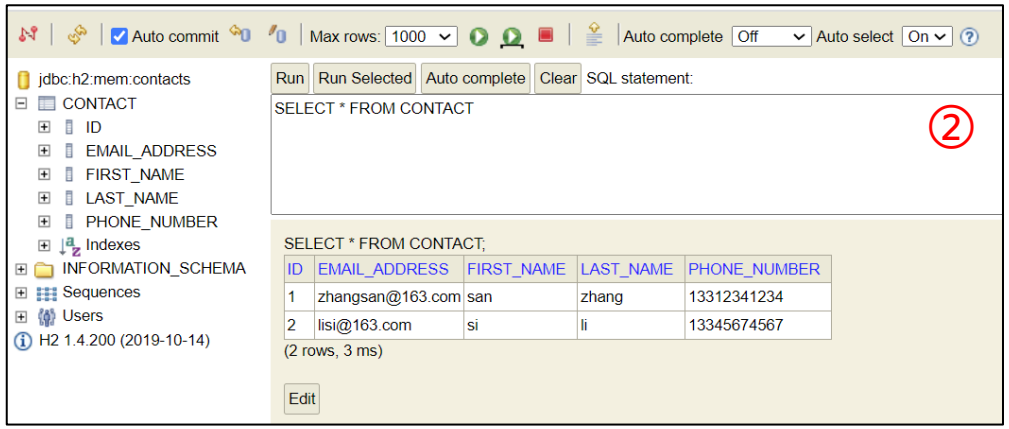
```
@Query(“Order o where o.deliveryCity = 'Seattle'”)
List<TacoOrder> readOrdersDeliveredInSeattle( );
```

作业

- 将上节课的作业完善，使用Spring Data JPA做数据库持久化，数据库使用H2

提交

- 所有源代码，压缩成一个文件。不包含编译后的class文件（删除target目录）
- 截图：①录入载图；②H2 console访问截图（http://localhost:8080/h2-console）



下节课准备：下载安装两个NoSQL数据库软件

- MongoDB
  - ✓ https://www.mongodb.com/download-center/community
  - ✓ 请使用默认端口号27017
  - ✓ mongosh：MongoDB Shell是MongoDB自带的交互式Javascript shell，用来对MongoDB进行操作和管理的交互式环境
- Redis
  - ✓ https://redis.io/download
  - ✓ 请使用默认端口号6379
  - ✓ redis-server，服务端
  - ✓ redis-cli，客户端

如果使用docker（可选）

- MongoDB

（1）创建网络

docker network create mongo-net

（2）启动Server

docker run --name my-mongo --network mongo-net -p 27017:27017 -d mongo:latest

（3）客户端访问

docker run -it --network mongo-net --rm mongo mongosh --host my-mongo

- Redis

（1）创建网络

docker network create redis-net

（2）启动Server

docker run --name my-redis --network redis-net -p 6379:6379 -d redis:latest

（3）客户端访问

docker run -it --network redis-net --rm redis redis-cli -h my-redis

使用客户端程序测试是否安装成功

1、MongoDB命令

show dbs

use mytest

db.createCollection('person')

db.person.insertOne({name:'**********',age:18})

db.person.find().pretty()

show dbs

用你自己名字取代

2、Redis命令

set counter 100

incr counter

get counter

keys *

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
test    0.000GB
> use mytest
switched to db mytest
> db.createCollection('person')
{ "ok" : 1 }
> db.person.insert({name:'**********',age:18})
WriteResult({ "nInserted" : 1 })
> db.person.find().pretty()
{
        "_id" : ObjectId("5e6ca0edfcb9a1d2721cb5ca"),
        "name" : "**********",
        "age" : 18
}
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
mytest  0.000GB
test    0.000GB
>
```

```
127.0.0.1:6379> set counter 100
OK
127.0.0.1:6379> incr counter
(integer) 101
127.0.0.1:6379> get counter
"101"
127.0.0.1:6379> keys *
1) "counter"
127.0.0.1:6379>
```

谢谢观看！