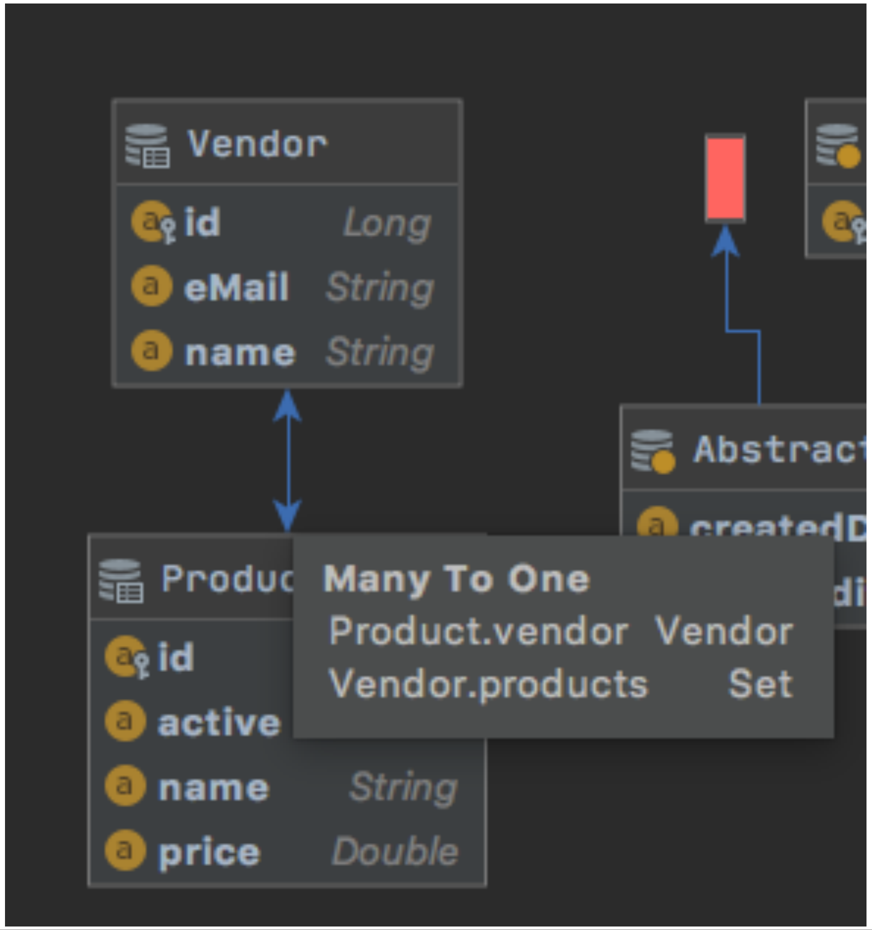
Our goal: one-to-many relationship

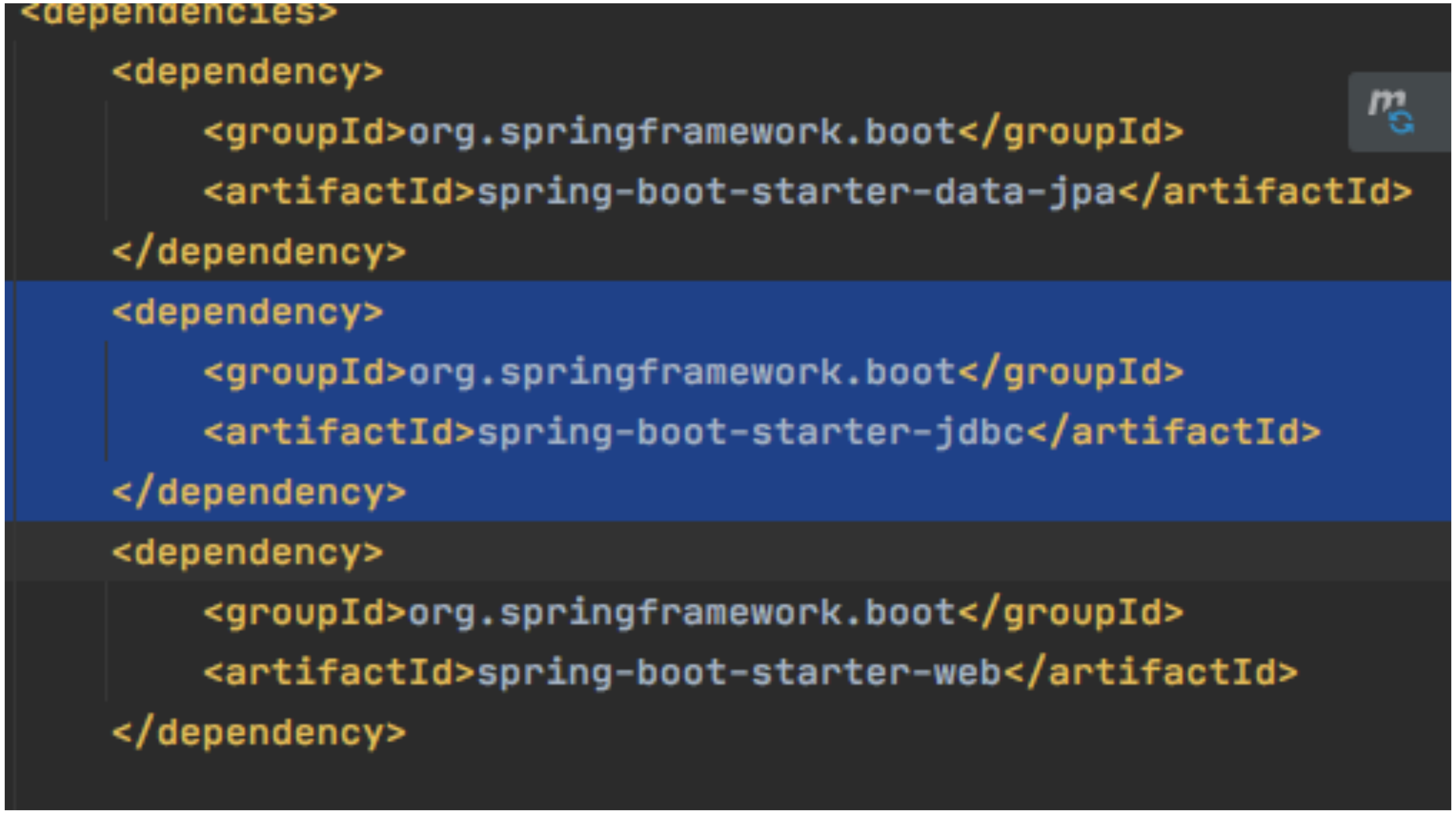


Many To One

Product.vendor Vendor

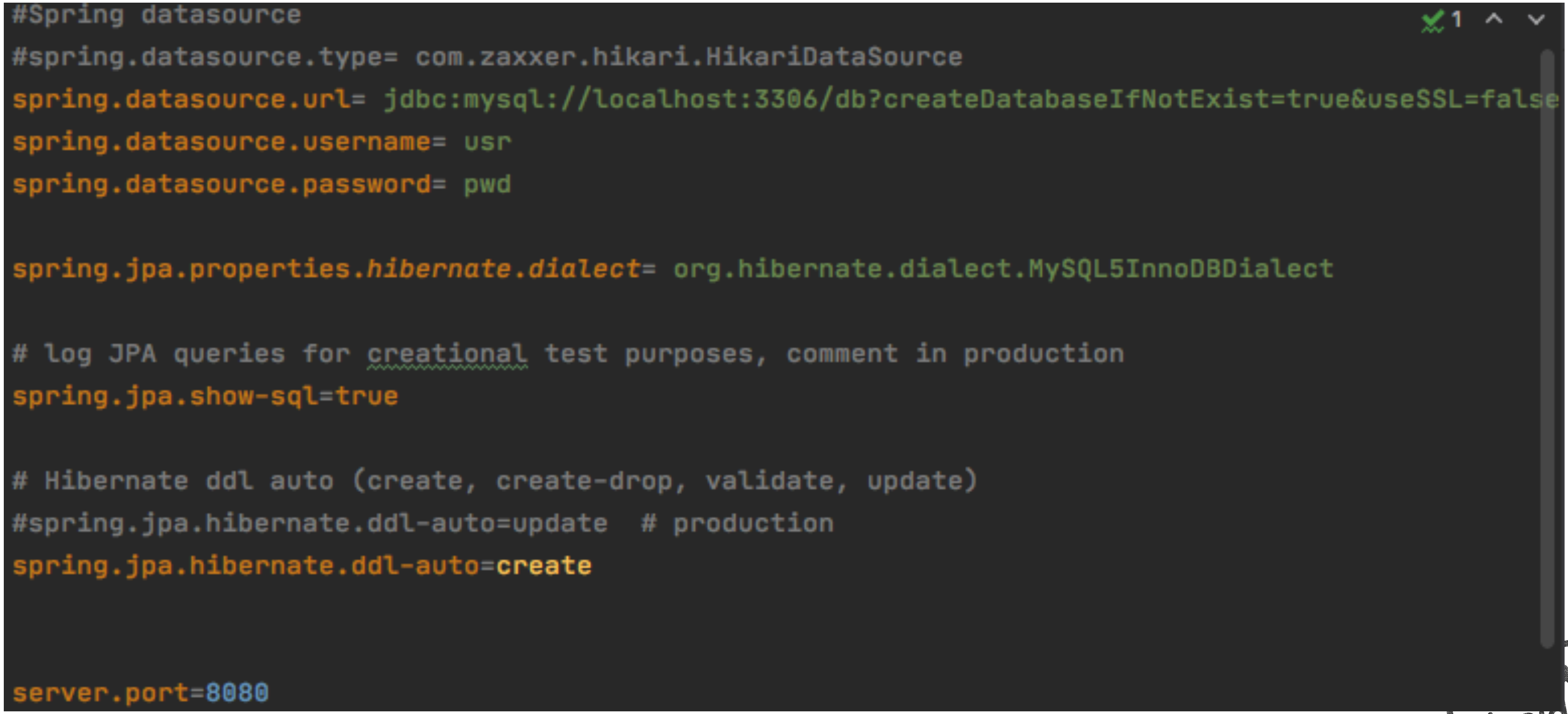
Vendor.products Set

enable Spring Data JDBC in pom.xml



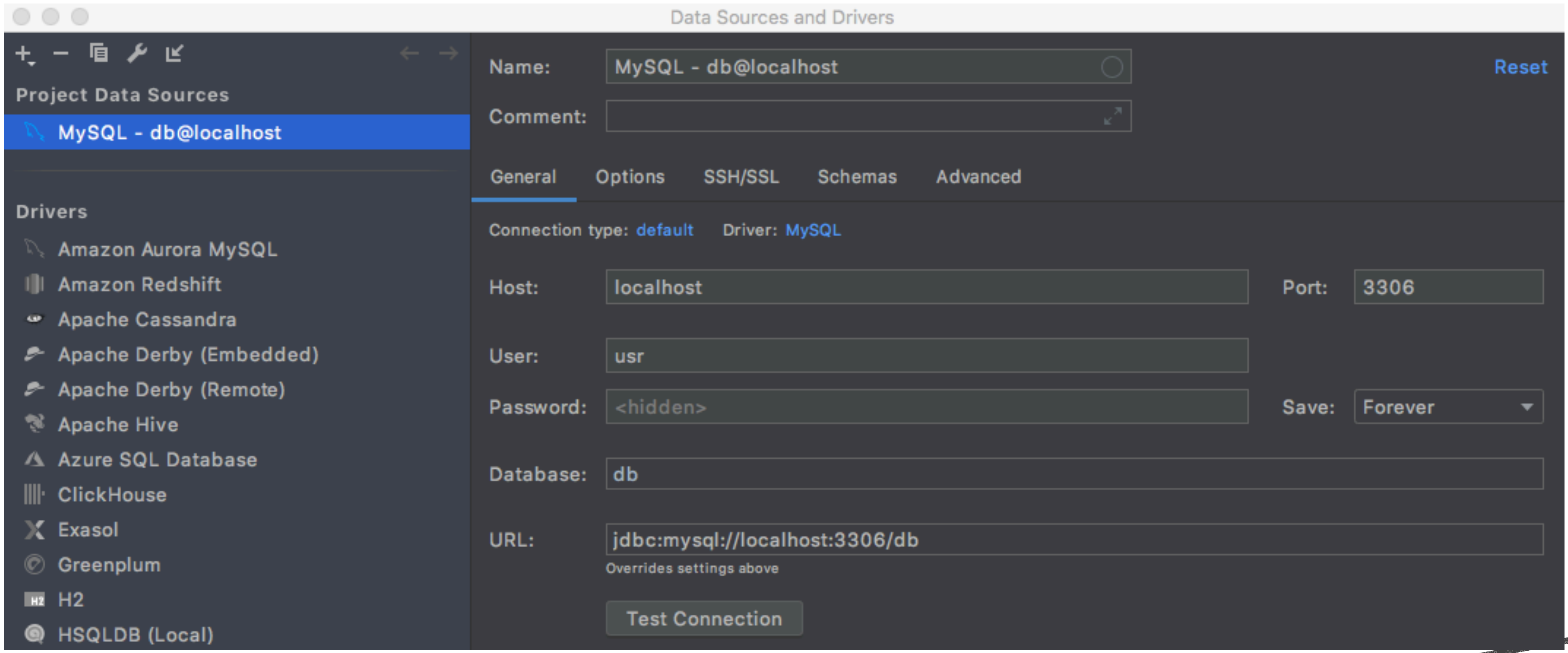
```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>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

Adapt application.properties



```
#Spring datasource
#spring.datasource.type= com.zaxxer.hikari.HikariDataSource
spring.datasource.url= jdbc:mysql://localhost:3306/db?createDatabaseIfNotExist=true&useSSL=false
spring.datasource.username= usr
spring.datasource.password= pwd

spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect

# log JPA queries for creational test purposes, comment in production
spring.jpa.show-sql=true

# Hibernate ddl auto (create, create-drop, validate, update)
#spring.jpa.hibernate.ddl-auto=update  # production
spring.jpa.hibernate.ddl-auto=create



server.port=8080
```

Add in View / Tool Windows / Database


Data Sources and Drivers

Project Data Sources
- MySQL - db@localhost

Drivers
- Amazon Aurora MySQL
- Amazon Redshift
- Apache Cassandra
- Apache Derby (Embedded)
- Apache Derby (Remote)
- Apache Hive
- Azure SQL Database
- ClickHouse
- Exasol
- Greenplum
- H2
- HSQLDB (Local)

Name: MySQL - db@localhost
Comment:

Reset

General | Options | SSH/SSL | Schemas | Advanced

Connection type: default Driver: MySQL

Host: localhost Port: 3306

User: usr

Password: <hidden> Save: Forever

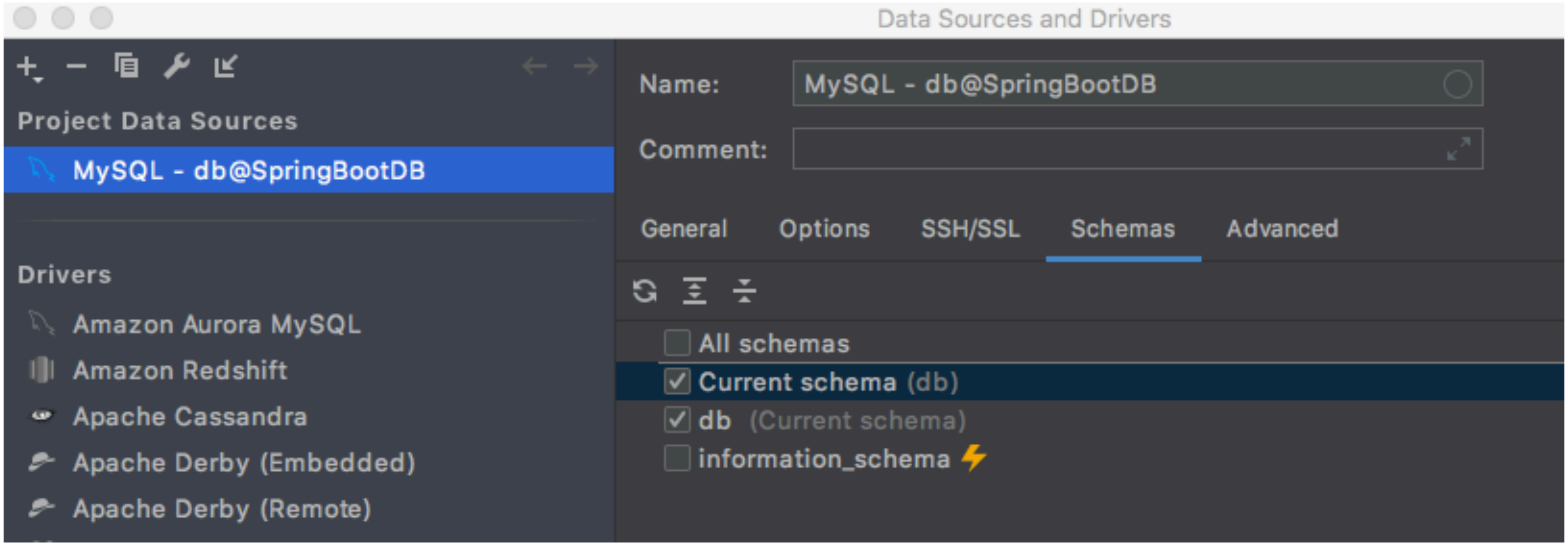
Database: db

URL: jdbc:mysql://localhost:3306/db

Overrides settings above

Test Connection

consider data base schemas



Data Sources and Drivers

Project Data Sources

- MySQL - db@SpringBootDB

Drivers

- Amazon Aurora MySQL
- Amazon Redshift
- Apache Cassandra
- Apache Derby (Embedded)
- Apache Derby (Remote)

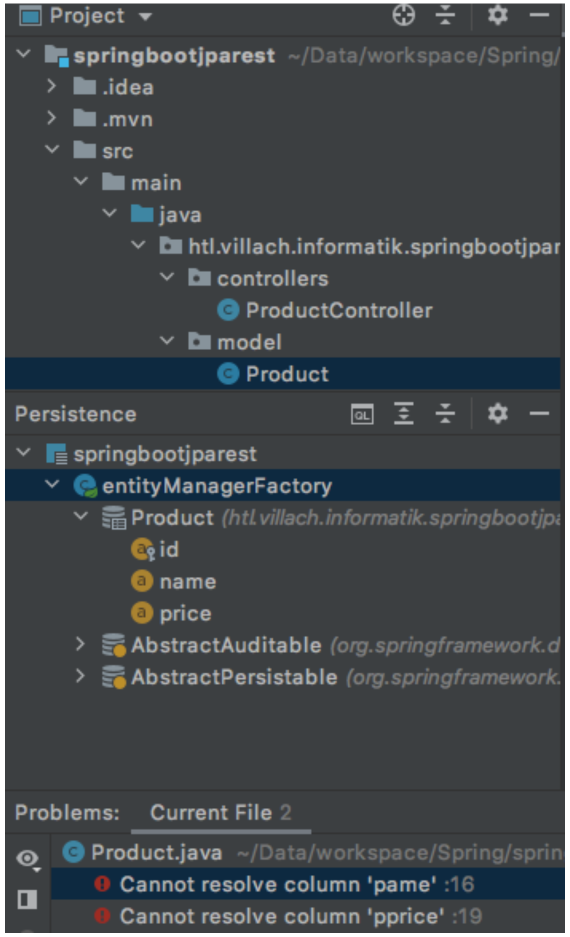
Name: MySQL - db@SpringBootDB

Comment:

General | Options | SSH/SSL | Schemas | Advanced

- ☐ All schemas
- ☑ Current schema (db)
- ☑ db (Current schema)
- ☐ information_schema

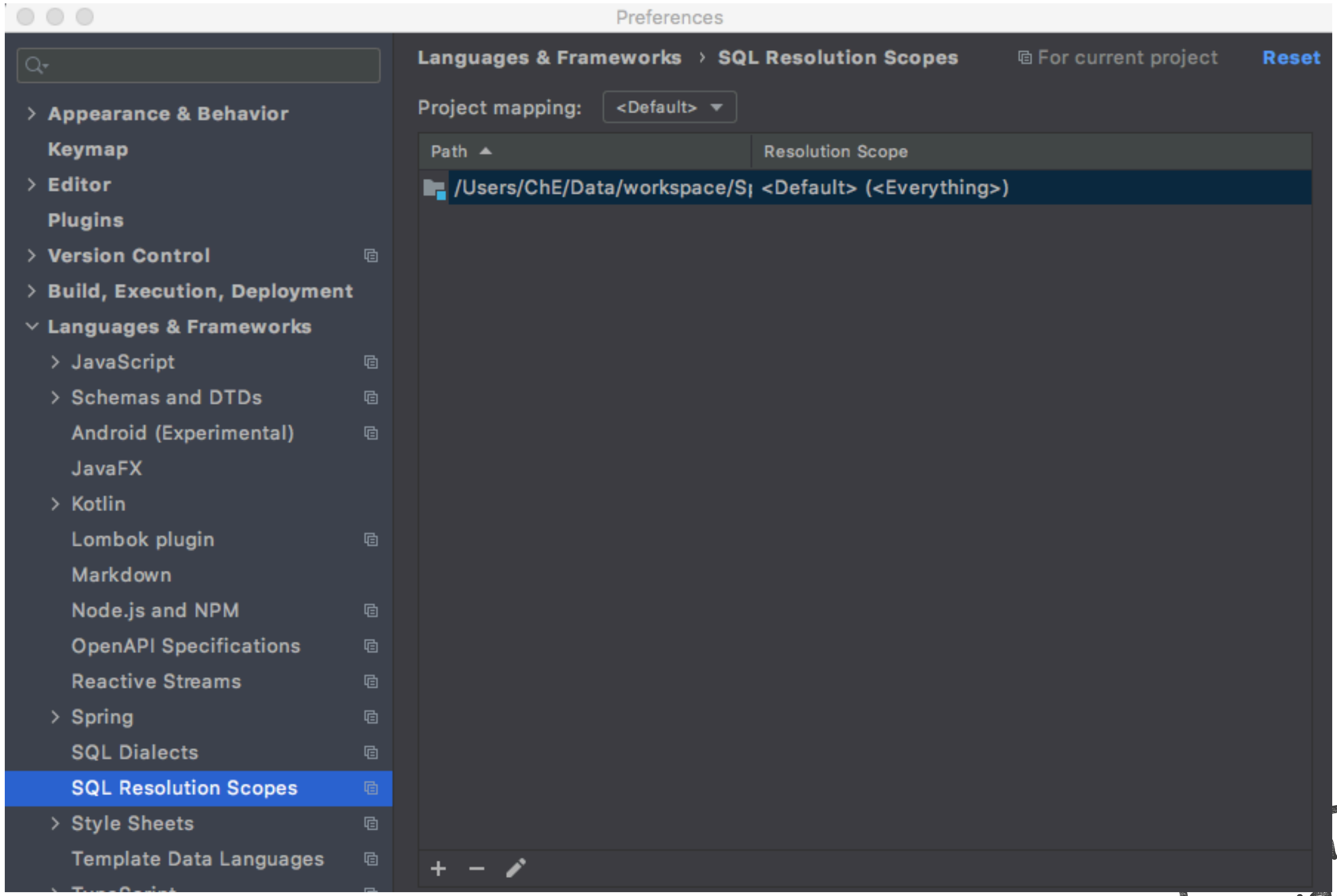
See in View / Tool Windows / Persistence



Project
- springbootjparest ~/Data/workspace/Spring/
  - .idea
  - .mvn
  - src
    - main
      - java
        - htl.villach.informatik.springbootjpar
          - controllers
            - ProductController
          - model
            - Product

Persistence
- springbootjparest
  - entityManagerFactory
    - Product (htl.villach.informatik.springbootjp
      - id
      - name
      - price
    - AbstractAuditable (org.springframework.d
    - AbstractPersistable (org.springframework.

Problems: Current File 2
- Product.java ~/Data/workspace/Spring/sprin
  - Cannot resolve column 'pame' :16
  - Cannot resolve column 'pprice' :19



Preferences

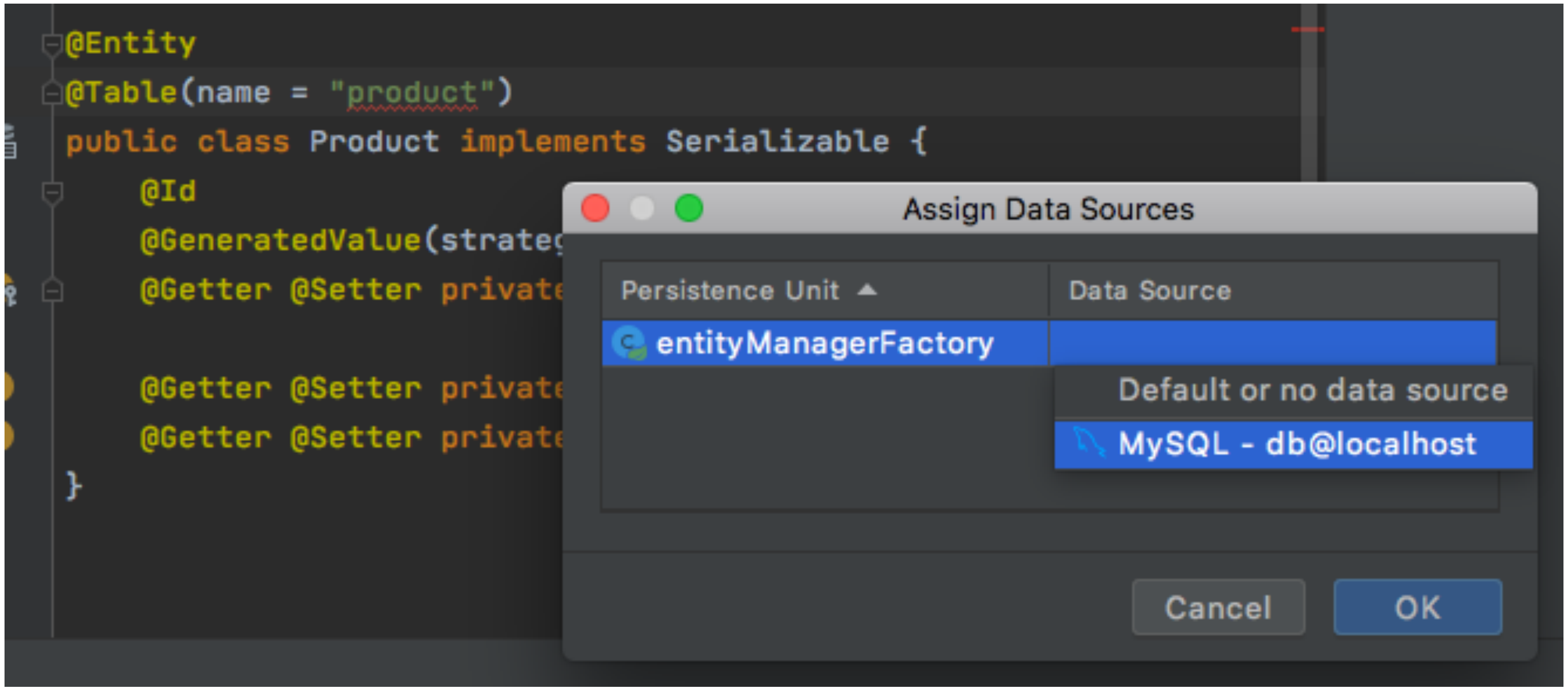
- Appearance & Behavior
- Keymap
- Editor
- Plugins
- Version Control
- Build, Execution, Deployment
- Languages & Frameworks
  - JavaScript
  - Schemas and DTDs
  - Android (Experimental)
  - JavaFX
  - Kotlin
  - Lombok plugin
  - Markdown
  - Node.js and NPM
  - OpenAPI Specifications
  - Reactive Streams
  - Spring
  - SQL Dialects
  - SQL Resolution Scopes
  - Style Sheets
  - Template Data Languages

Languages & Frameworks › SQL Resolution Scopes For current project Reset

Project mapping: <Default>

Path	Resolution Scope
/Users/ChE/Data/workspace/S	<Default> (<Everything>)

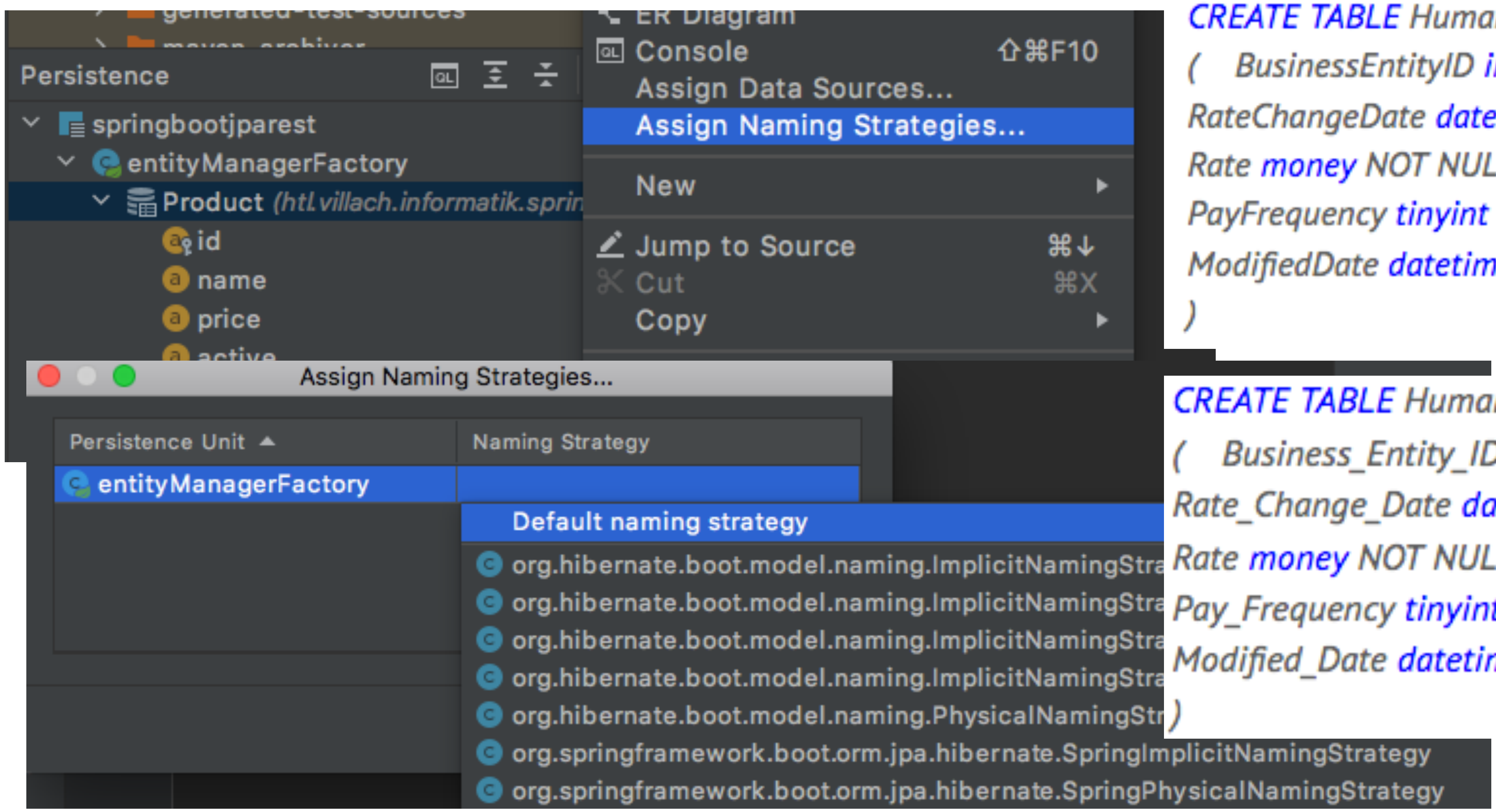
or assign data source to your models



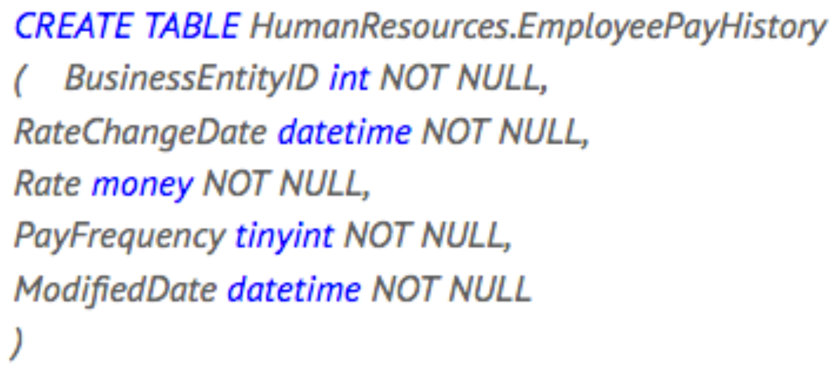
```
@Entity
@Table(name = "product")
public class Product implements Serializable {
    @Id
    @GeneratedValue(strateg
    @Getter @Setter private

    @Getter @Setter private
    @Getter @Setter private
}
```

choose a naming strategy

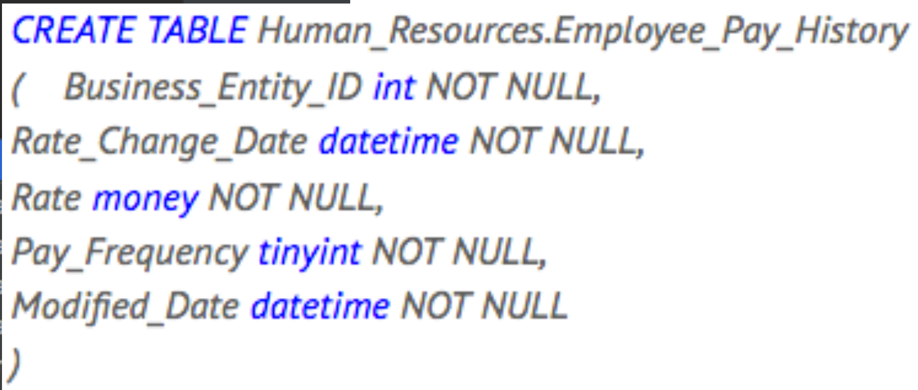
Java uses camel case while SQL uses underscore per default



Persistence
springbootjparest
entityManagerFactory
Product (htl.villach.informatik.sprin
id
name
price
active
ER Diagram
Console ⇧⌘F10
Assign Data Sources...
Assign Naming Strategies...
New
Jump to Source ⌘↓
Cut ⌘X
Copy
Assign Naming Strategies...
Persistence Unit
Naming Strategy
entityManagerFactory
Default naming strategy
org.hibernate.boot.model.naming.ImplicitNamingStra
org.hibernate.boot.model.naming.ImplicitNamingStra
org.hibernate.boot.model.naming.ImplicitNamingStra
org.hibernate.boot.model.naming.ImplicitNamingStra
org.hibernate.boot.model.naming.PhysicalNamingStr
org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy

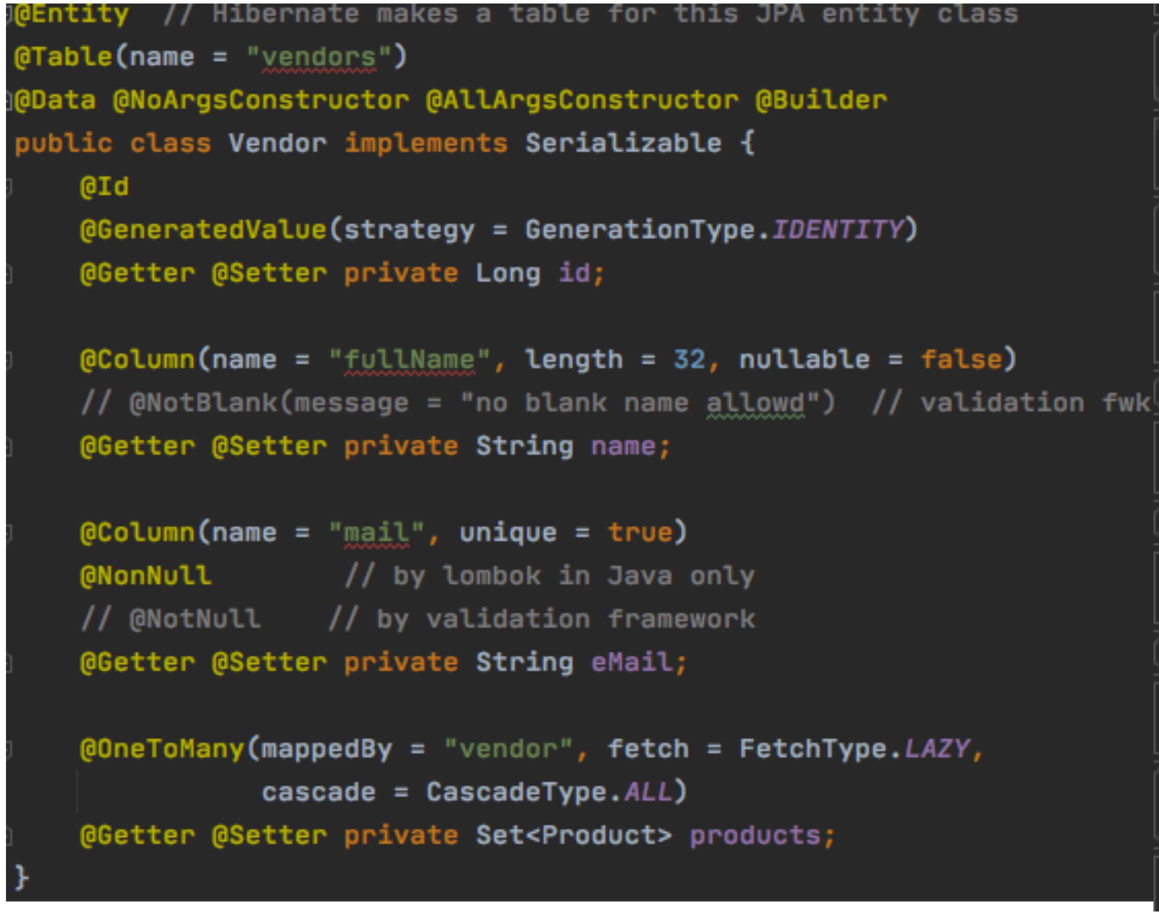


```
CREATE TABLE HumanResources.EmployeePayHistory
(    BusinessEntityID int NOT NULL,
RateChangeDate datetime NOT NULL,
Rate money NOT NULL,
PayFrequency tinyint NOT NULL,
ModifiedDate datetime NOT NULL
)
```

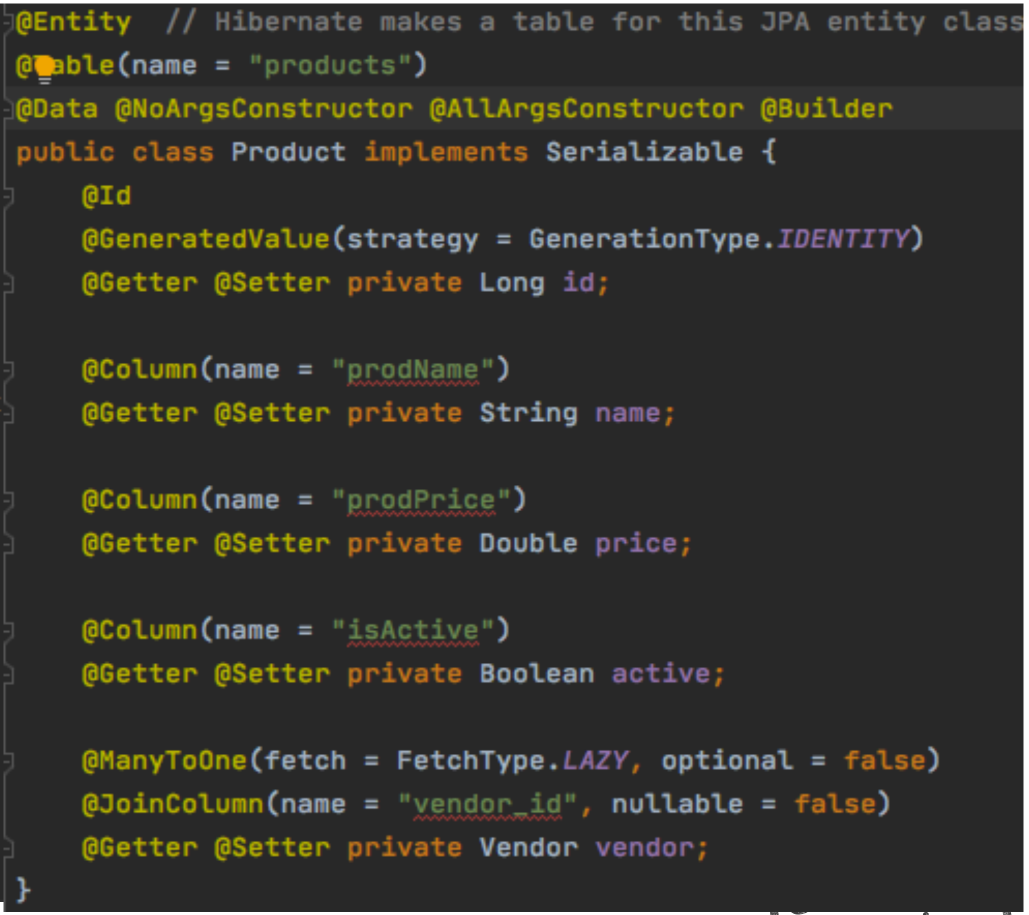


```
CREATE TABLE Human_Resources.Employee_Pay_History
(    Business_Entity_ID int NOT NULL,
Rate_Change_Date datetime NOT NULL,
Rate money NOT NULL,
Pay_Frequency tinyint NOT NULL,
Modified_Date datetime NOT NULL
)
```

One-To-Many Mapping, different ways/places to consider null values

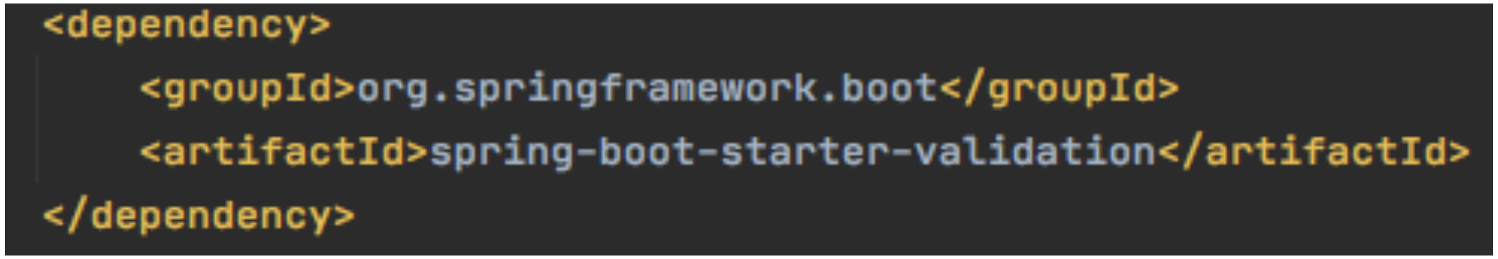


```java
@Entity  // Hibernate makes a table for this JPA entity class
@Table(name = "vendors")
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Vendor implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter @Setter private Long id;

    @Column(name = "fullName", length = 32, nullable = false)
    // @NotBlank(message = "no blank name allowd")  // validation fwk
    @Getter @Setter private String name;

    @Column(name = "mail", unique = true)
    @NonNull          // by lombok in Java only
    // @NotNull    // by validation framework
    @Getter @Setter private String eMail;

    @OneToMany(mappedBy = "vendor", fetch = FetchType.LAZY,
               cascade = CascadeType.ALL)
    @Getter @Setter private Set<Product> products;
}
```



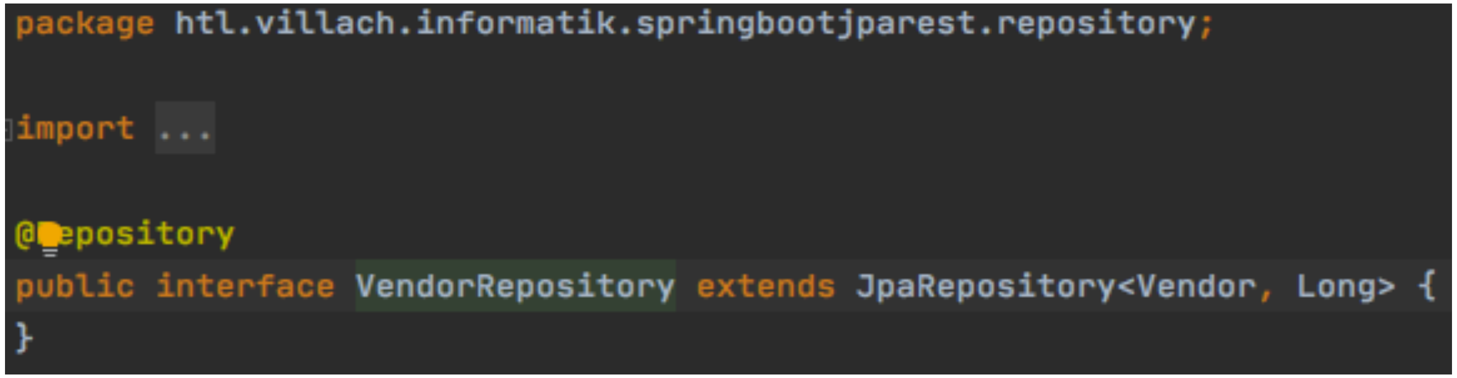
```java
@Entity  // Hibernate makes a table for this JPA entity class
@Table(name = "products")
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Product implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter @Setter private Long id;

    @Column(name = "prodName")
    @Getter @Setter private String name;

    @Column(name = "prodPrice")
    @Getter @Setter private Double price;

    @Column(name = "isActive")
    @Getter @Setter private Boolean active;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "vendor_id", nullable = false)
    @Getter @Setter private Vendor vendor;
}
```

Spring Data Validation for @NotBlank ...

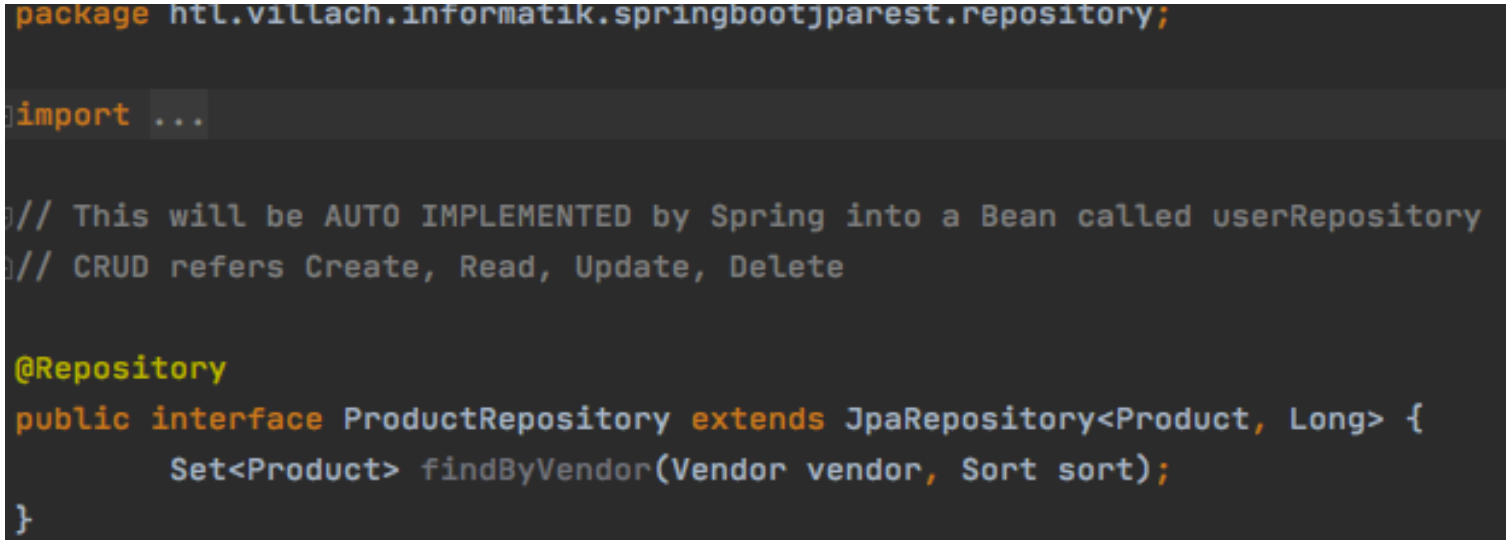


```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

Create/Extend Repositories



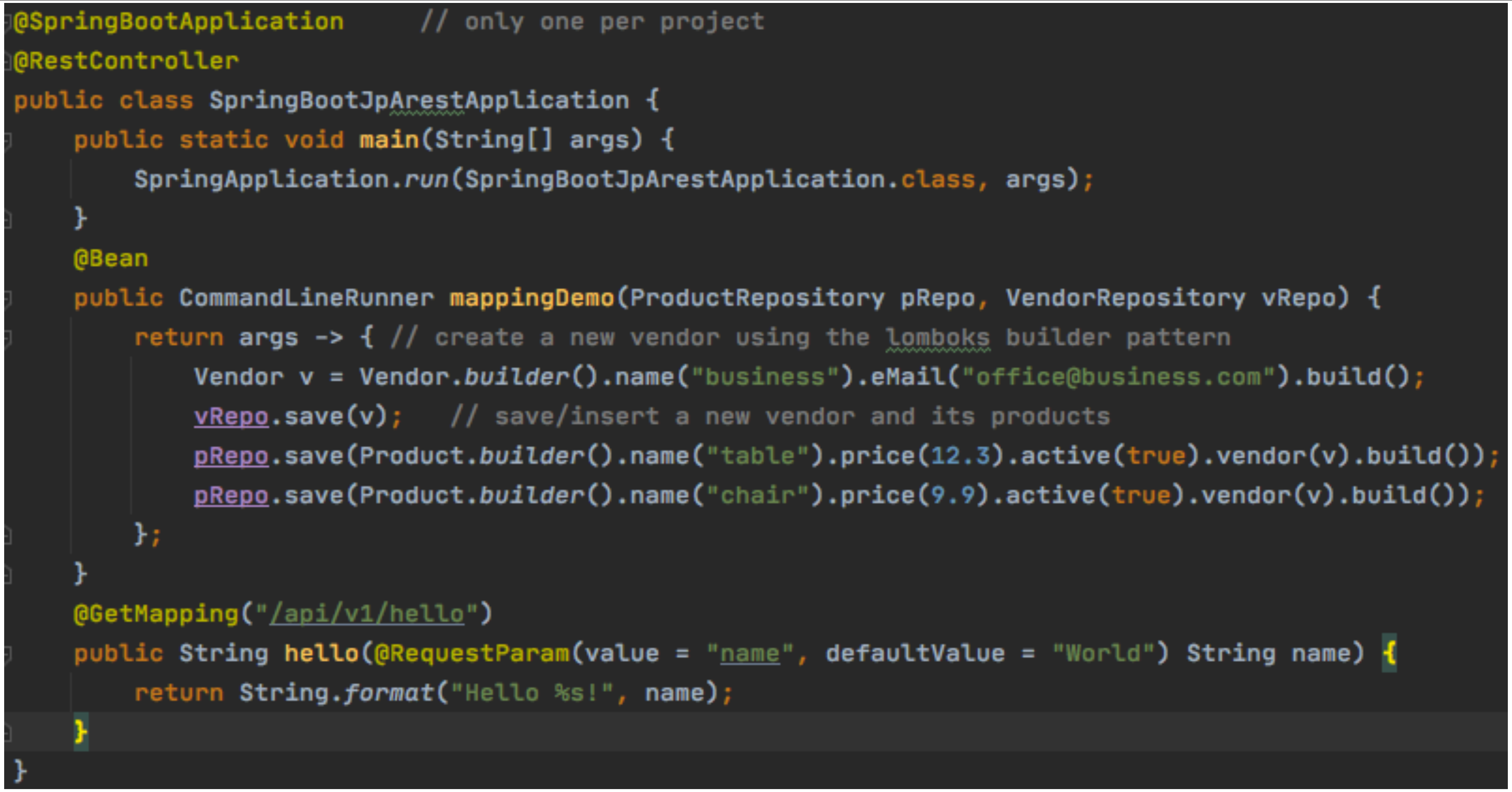
```
package htl.villach.informatik.springbootjparest.repository;

import ...

@Repository
public interface VendorRepository extends JpaRepository<Vendor, Long> {
}
```



```
package htl.villach.informatik.springbootjparest.repository;

import ...

// This will be AUTO IMPLEMENTED by Spring into a Bean called userRepository
// CRUD refers Create, Read, Update, Delete

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
        Set<Product> findByVendor(Vendor vendor, Sort sort);
}
```

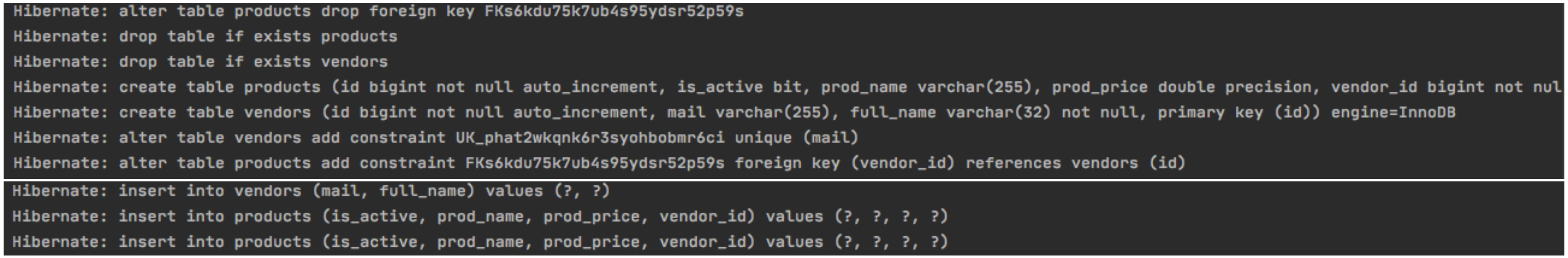
Check for One-To-Many Mapping

JUnit tests for Spring Data and RESTful API will follow soon

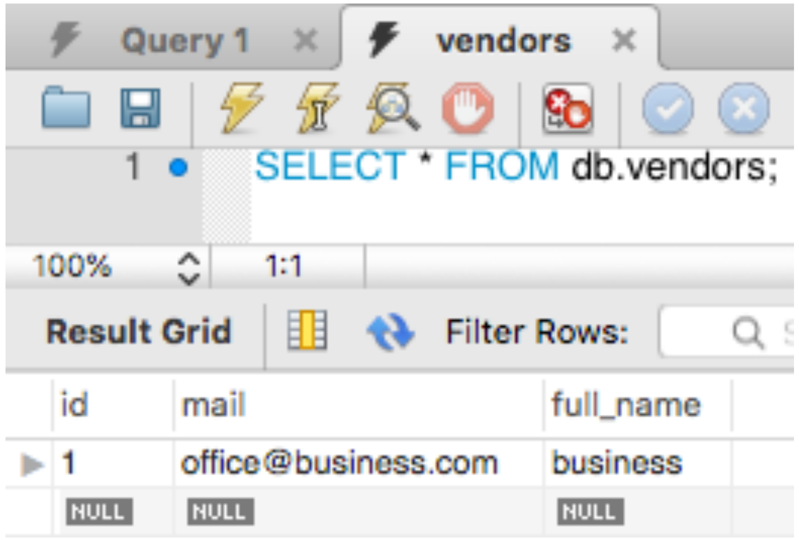


```java
@SpringBootApplication      // only one per project
@RestController
public class SpringBootJpArestApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootJpArestApplication.class, args);
    }
    @Bean
    public CommandLineRunner mappingDemo(ProductRepository pRepo, VendorRepository vRepo) {
        return args -> { // create a new vendor using the lomboks builder pattern
            Vendor v = Vendor.builder().name("business").eMail("office@business.com").build();
            vRepo.save(v);    // save/insert a new vendor and its products
            pRepo.save(Product.builder().name("table").price(12.3).active(true).vendor(v).build());
            pRepo.save(Product.builder().name("chair").price(9.9).active(true).vendor(v).build());
        };
    }
    @GetMapping("/api/v1/hello")
    public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
        return String.format("Hello %s!", name);
    }
}
```

Results of One-To-Many Mapping

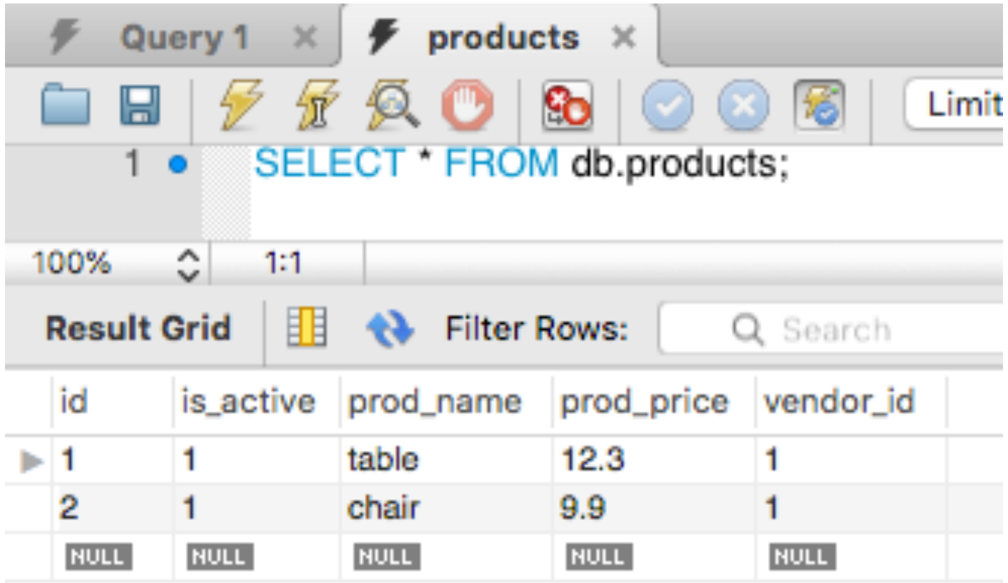


```
Hibernate: alter table products drop foreign key FKs6kdu75k7ub4s95ydsr52p59s
Hibernate: drop table if exists products
Hibernate: drop table if exists vendors
Hibernate: create table products (id bigint not null auto_increment, is_active bit, prod_name varchar(255), prod_price double precision, vendor_id bigint not nul
Hibernate: create table vendors (id bigint not null auto_increment, mail varchar(255), full_name varchar(32) not null, primary key (id)) engine=InnoDB
Hibernate: alter table vendors add constraint UK_phat2wkqnk6r3syohbobmr6ci unique (mail)
Hibernate: alter table products add constraint FKs6kdu75k7ub4s95ydsr52p59s foreign key (vendor_id) references vendors (id)
Hibernate: insert into vendors (mail, full_name) values (?, ?)
Hibernate: insert into products (is_active, prod_name, prod_price, vendor_id) values (?, ?, ?, ?)
Hibernate: insert into products (is_active, prod_name, prod_price, vendor_id) values (?, ?, ?, ?)
```



```
SELECT * FROM db.vendors;
```

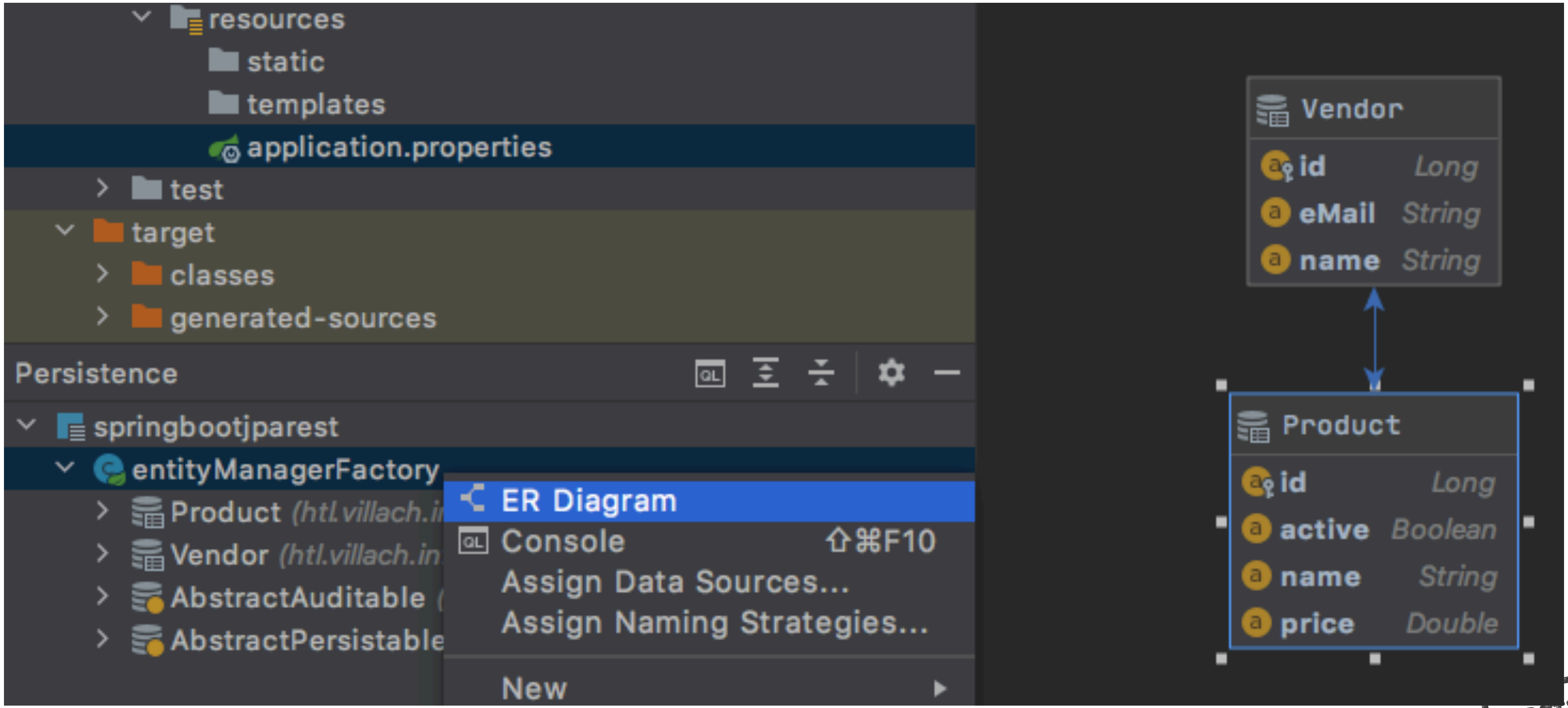
id	mail	full_name
1	office@business.com	business
NULL	NULL	NULL



```
SELECT * FROM db.products;
```

id	is_active	prod_name	prod_price	vendor_id
1	1	table	12.3	1
2	1	chair	9.9	1
NULL	NULL	NULL	NULL	NULL

graphical view on data base schema



resources
static
templates
application.properties
test
target
classes
generated-sources
Persistence
springbootjparest
entityManagerFactory
Product (htl.villach.i
Vendor (htl.villach.in
AbstractAuditable (
AbstractPersistable
ER Diagram
Console ⇧⌘F10
Assign Data Sources...
Assign Naming Strategies...
New
Vendor
id Long
eMail String
name String
Product
id Long
active Boolean
name String
price Double