Spring Data Advanced Querying

Query Methods, JPQL Advanced Repositories, Spring Configuration

SoftUni Team

Technical Trainers

Software University

https://softuni.bg

Table of Contents

1. Retrieving Data by Custom Queries.
2. Java Persistence Query Language.
3. Repository Inheritance.
4. Spring Custom Configuration.

Questions

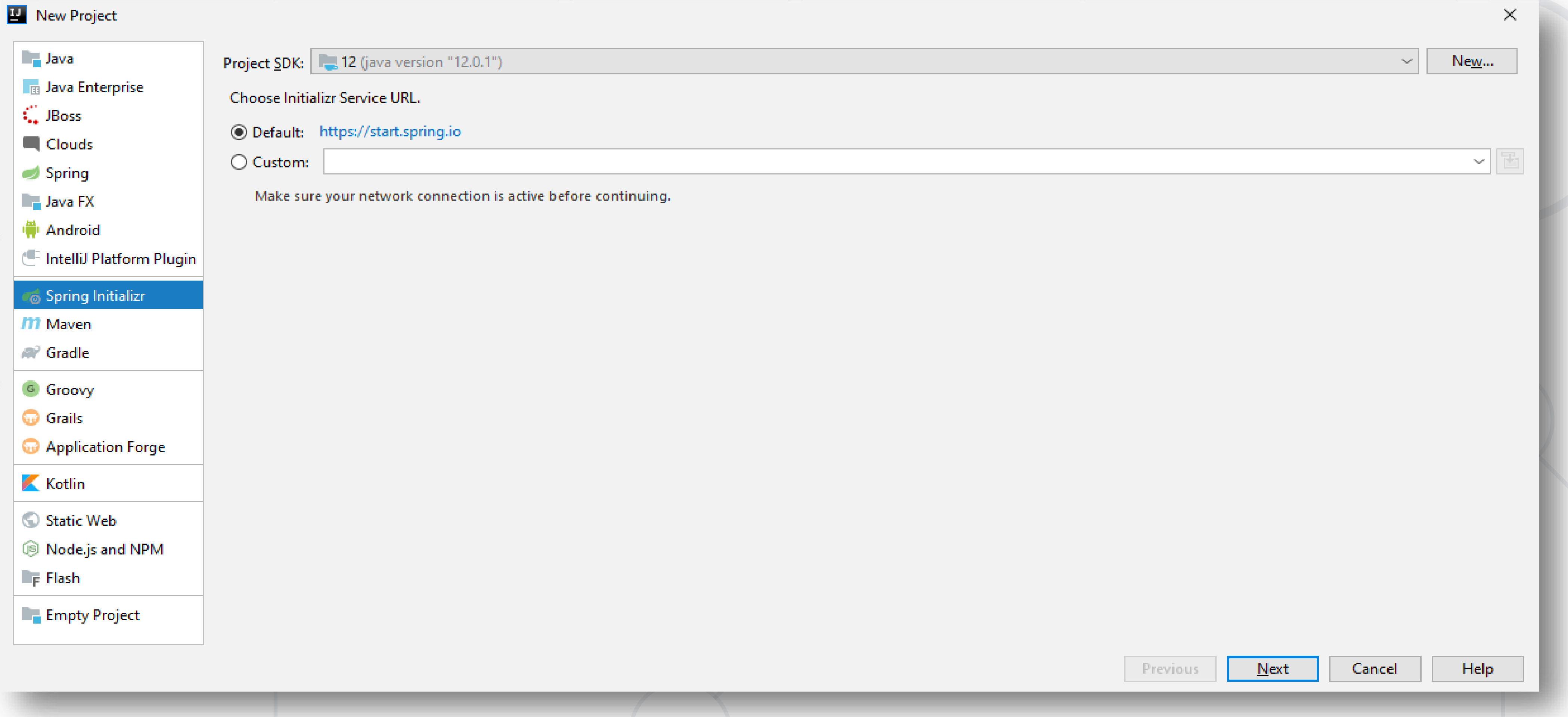
sli.do

#Java-DB

Querying

Retrieving Data by Custom Queries

Spring Project



New Project

Project SDK: 12 (java version "12.0.1")

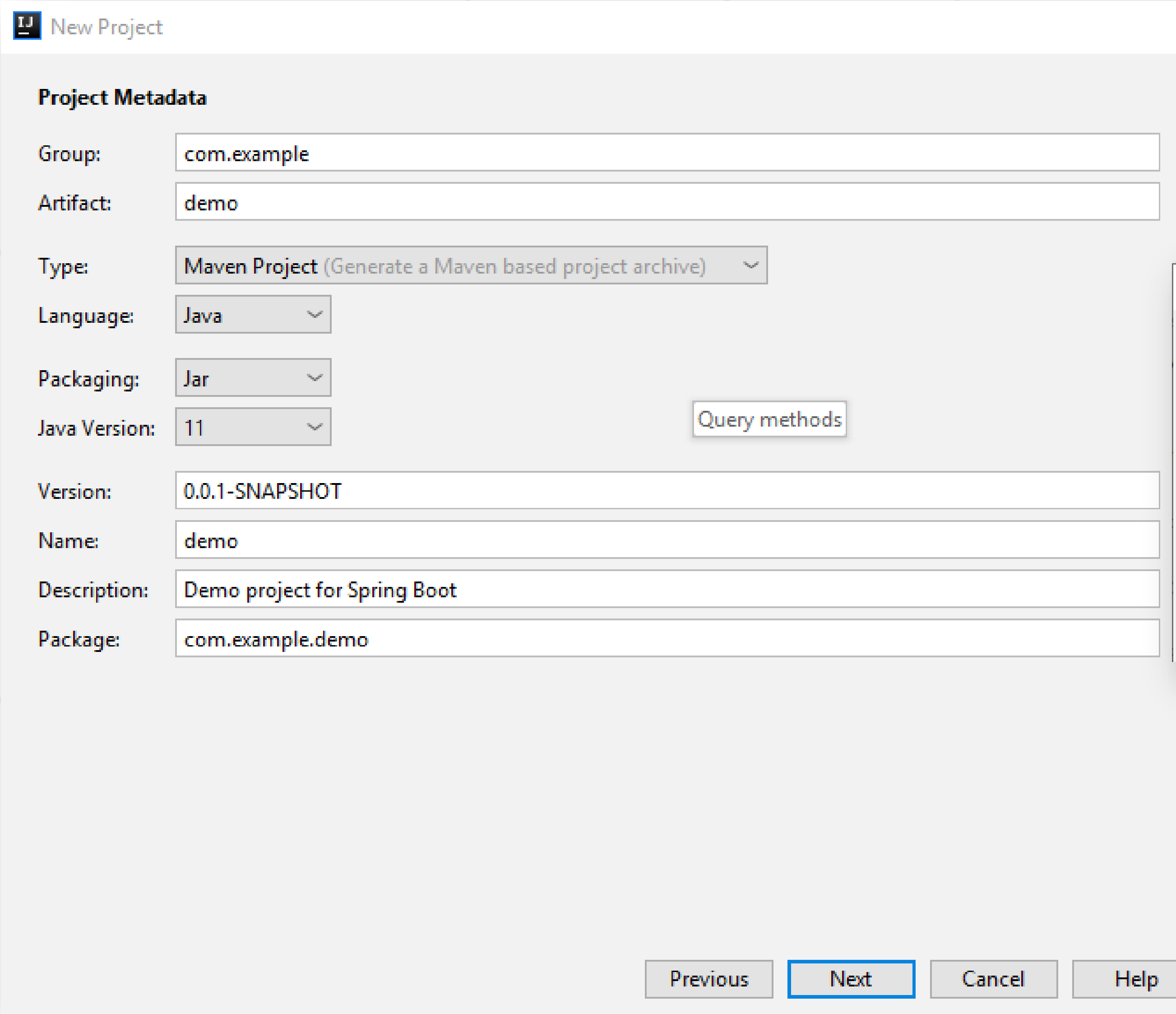
Choose Initializr Service URL.

● Default: https://start.spring.io

○ Custom:

Make sure your network connection is active before continuing.

Spring Project (2)

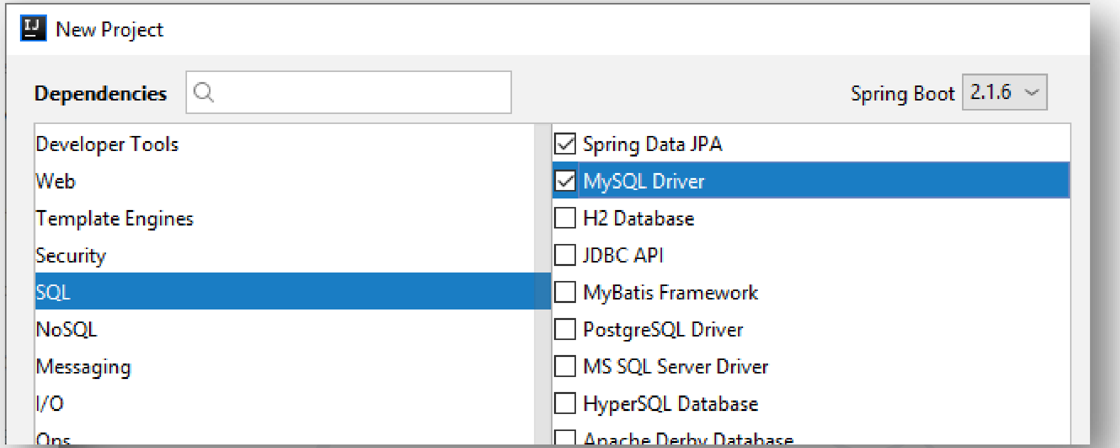


New Project

Project Metadata

Group:	com.example
Artifact:	demo
Type:	Maven Project (Generate a Maven based project archive)
Language:	Java
Packaging:	Jar
Java Version:	11
Version:	0.0.1-SNAPSHOT
Name:	demo
Description:	Demo project for Spring Boot
Package:	com.example.demo

Query methods



New Project

Dependencies Spring Boot 2.1.6

Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops

☑ Spring Data JPA
☑ MySQL Driver
☐ H2 Database
☐ JDBC API
☐ MyBatis Framework
☐ PostgreSQL Driver
☐ MS SQL Server Driver
☐ HyperSQL Database
☐ Apache Derby Database

Query Methods

ShampooRepository.java

```
@Repository
public interface ShampooDao extends JpaRepository
<Shampoo, Long> {

    List<Shampoo> findByBrand(String brand);
}
```

Query method

Paramater

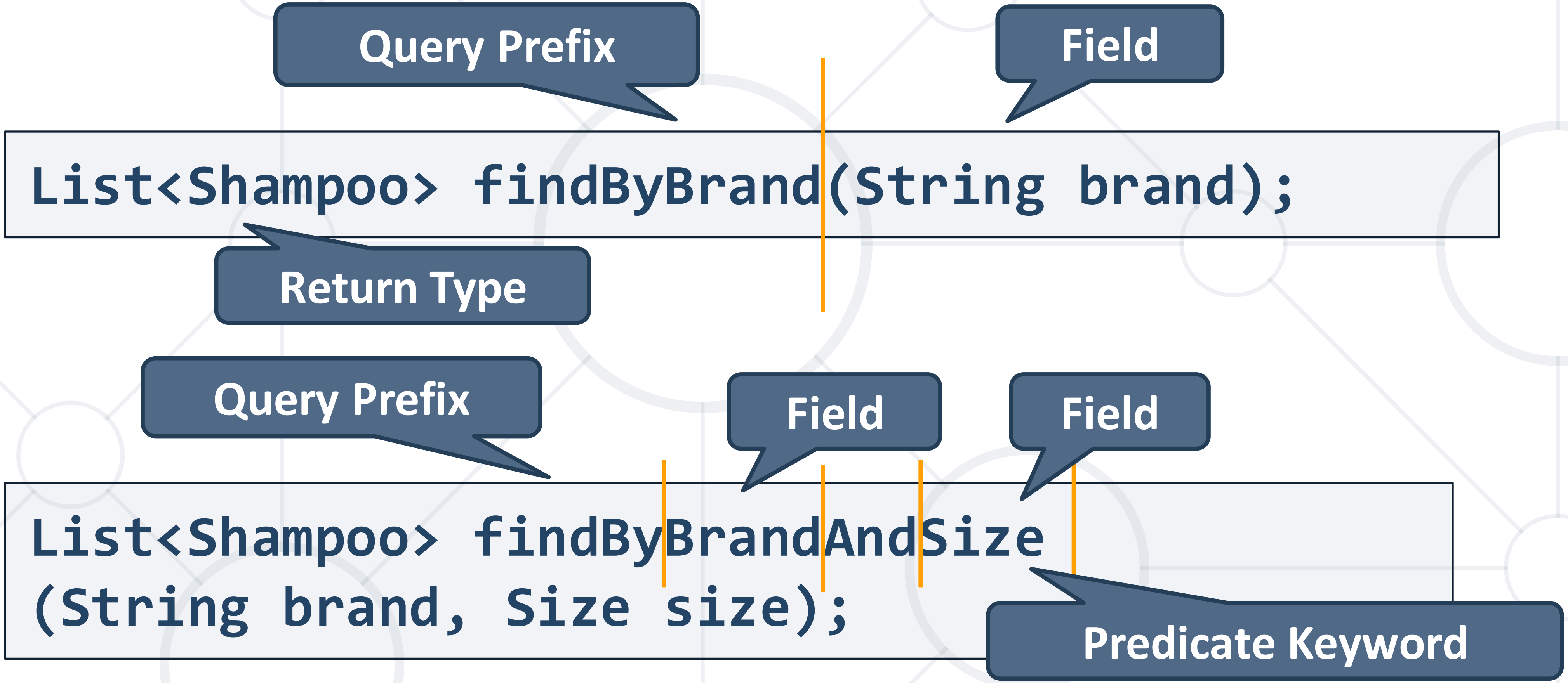
The method translates to the following query:

SQL

```
SELECT *
  FROM shampoos AS s
 WHERE s.brand = ?
```

Paramater

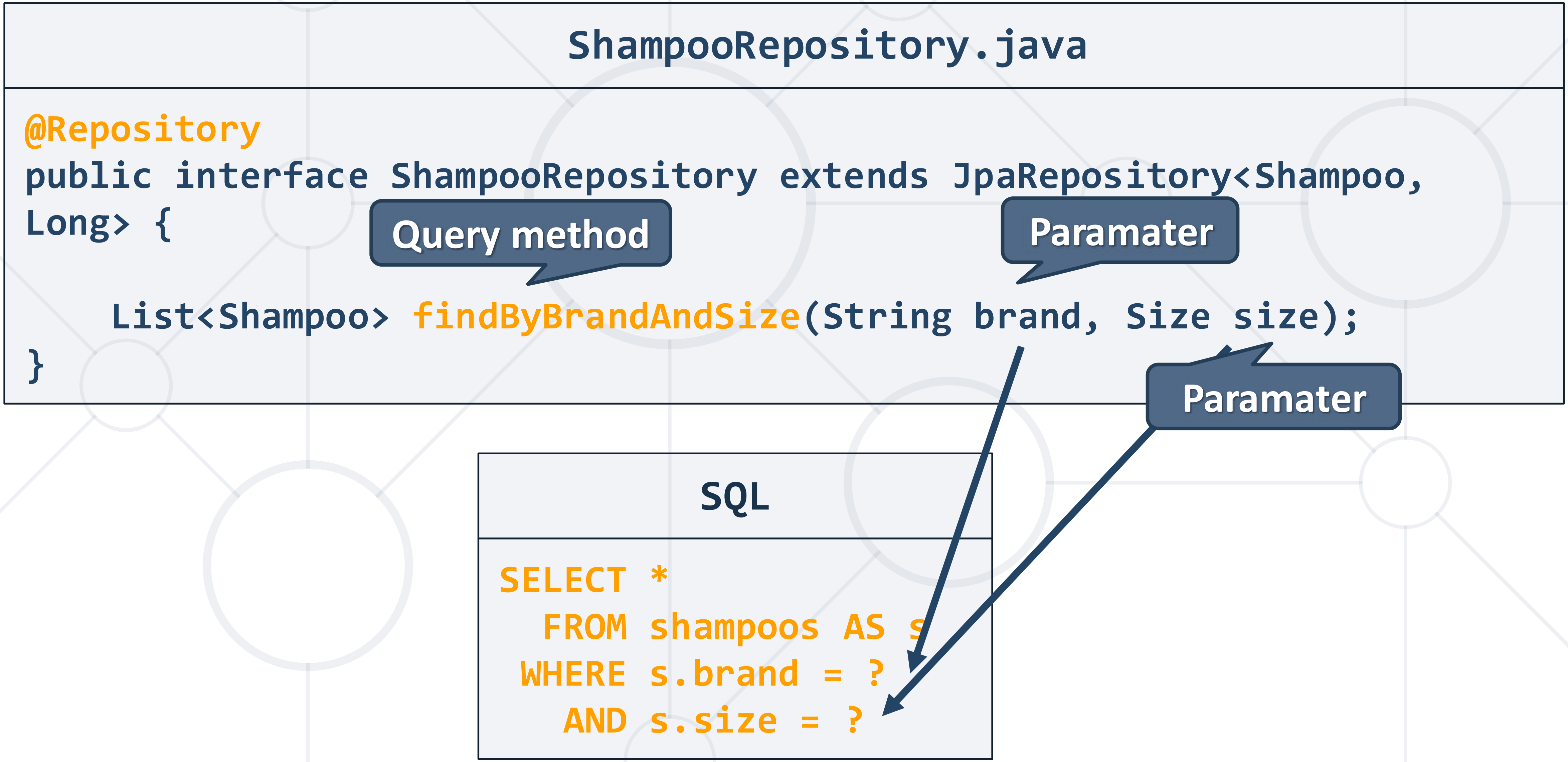
Query Lookup



```
List<Shampoo> findByBrand(String brand);
```

```
List<Shampoo> findByBrandAndSize
(String brand, Size size);
```

Query Methods



ShampooRepository.java

```
@Repository
public interface ShampooRepository extends JpaRepository<Shampoo,
Long> {

    List<Shampoo> findByBrandAndSize(String brand, Size size);
}
```

SQL

```
SELECT *
  FROM shampoos AS s
 WHERE s.brand = ?
   AND s.size = ?
```

Problem: Select Shampoos by Size

- Write a method that selects all shampoos by input size
  - Order the result by shampoo id
- Example input-output:

```
MEDIUM
```

```
Nature Moments Mediterranean Olive Oil & Aloe
Vera MEDIUM 6.50lv.
Volume & Fullness Lavender MEDIUM 5.50lv.
Rose Shine & Hydration MEDIUM 6.50lv.
Color Protection & Radiance MEDIUM 6.75lv.
…
```

Solution: Select Shampoos by Size

ShampooRepository.java

```java
@Repository
public interface ShampooRepository extends JpaRepository<Shampoo, Long> {
    List<Shampoo> getAllBySizeOrderById(Size sizeValue);
}
```

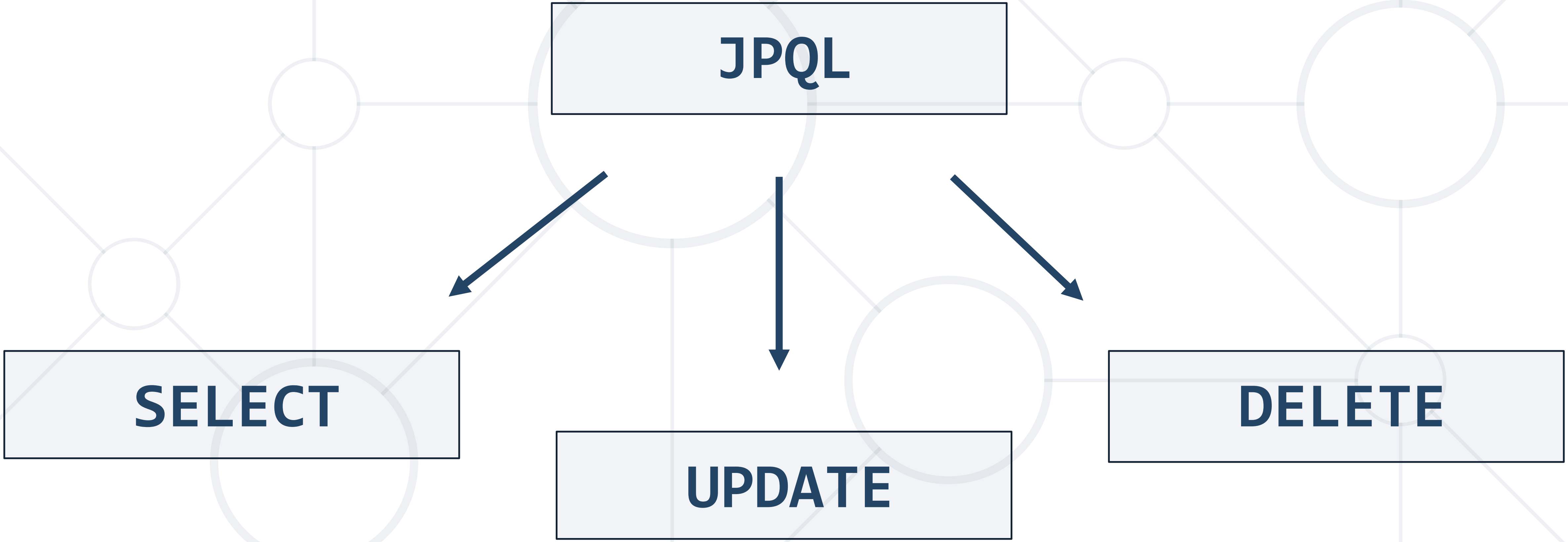
JPQL

Java Persistence Query Language

JPQL

- Object-oriented query language
  - Part of the Java Persistence API
  - Used to make queries against entities stored in a relational database
  - SQL syntax operating with entities, not tables in the data source

JPQL Functionalities



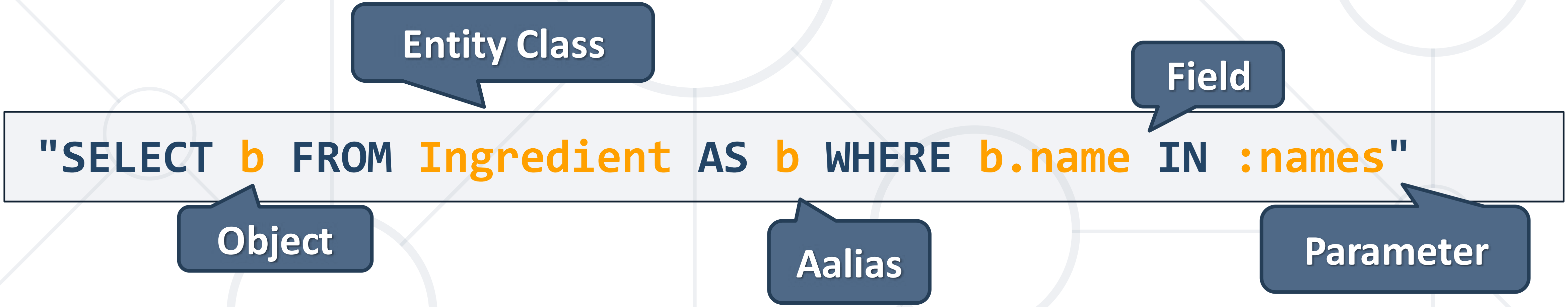
JPQL

SELECT

UPDATE

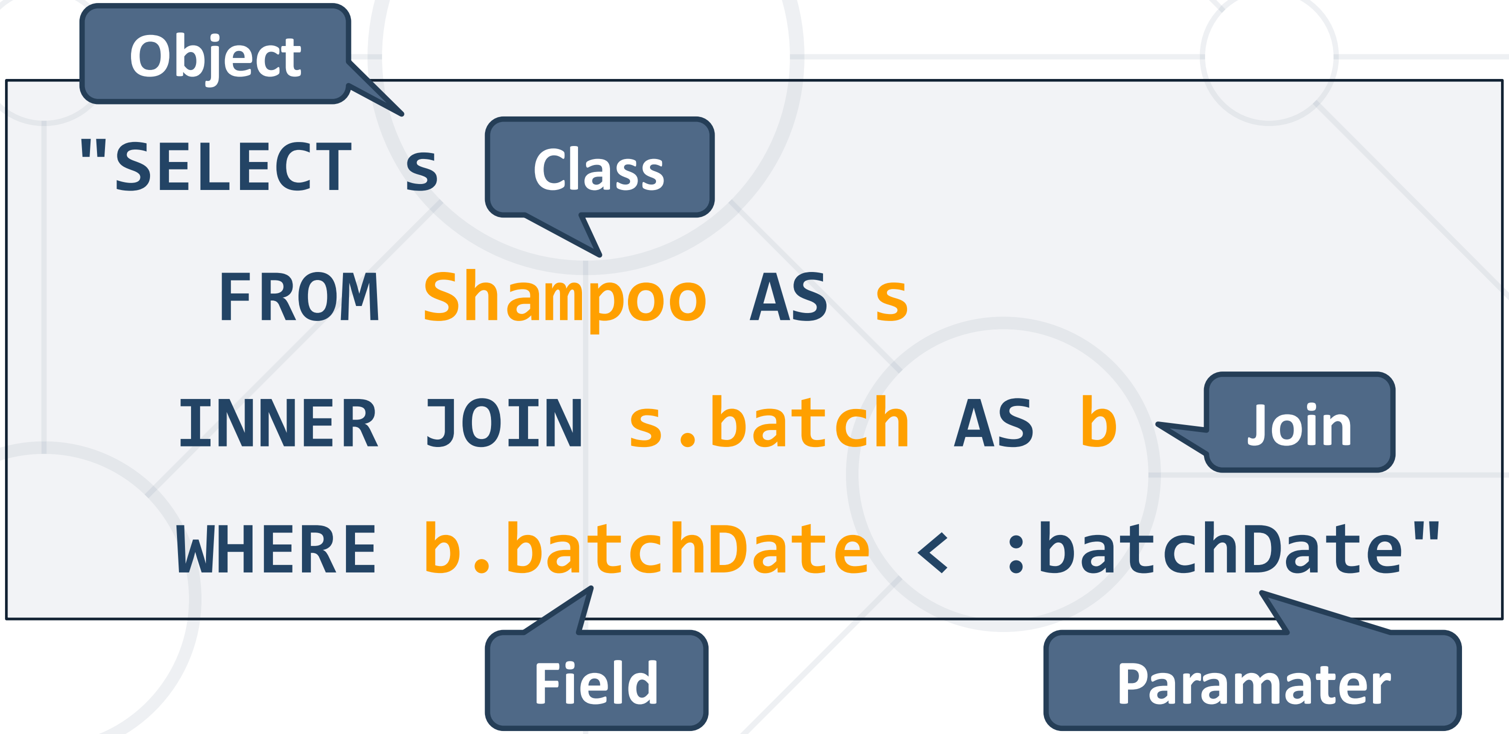
DELETE

JPQL Select Syntax



```
"SELECT b FROM Ingredient AS b WHERE b.name IN :names"
```

JPQL Join Syntax



```
"SELECT s
    FROM Shampoo AS s
   INNER JOIN s.batch AS b
   WHERE b.batchDate < :batchDate"
```

JPQL Syntax

- Update:

```
"UPDATE Ingredient AS b
    SET b.price = b.price*1.10
  WHERE b.name IN :names"
```

Parameter

- Delete:

```
"DELETE FROM Ingredient AS b
  WHERE b.name = :name"
```

Problem: Select Shampoos by Ingredients

- Write a method that selects all shampoos with ingredients in a given list
- Example input-output:

```
Berry
Mineral-Colagen
```

```
Color Protection & Radiance
Fresh it Up!
Nectar Nutrition
Superfruit Nutrition
Color Protection & Radiance
Nectar Nutrition
…
```

Solution: Select Shampoos by Ingredients

ShampooRepository.java

```
@Repository
public interface IngredientRepository extends JpaRepository<Ingredient, Lo
ng>{

     @Query(value = "select s from Shampoo s " +
      "join s.ingredients i where i in :ingredients")
     List<Shampoo> findByIngredientsIn(@Param(value = "ingredients")
                                        Set<Ingredient> ingredients);
}
```

Advanced Repositories

Repository Inheritance

Repository Inheritance

- In bigger applications we have similar entities extending an abstract class
- Their base attributes and actions towards them are the same regardless differences
- We can set up a **base repository** to reduce query and code duplication
- It can be inherited to clear up specifics

Example: Repository Inheritance

Not a repository

IngredientRepository.java

```
@NoRepositoryBean
public interface IngredientRepository<T extends Ingredient>
extends JpaRepository<T, Long>{
        //…
}
```

ChemicalIngredientRepository.java

```
@Repository
public interface ChemicalIngredientRepository extends IngredientRepository
<BasicChemicalIngredient> {
   List<ChemicalIngredient> findByChemicalFormula(String chemicalFormula);
}
```

Example: Repository Inheritance

CustomShampooRepository.java

```java
public interface CustomShampooRepository {

    void create(BasicShampoo basicShampoo);
}
```

CustomShampooRepositoryImpl.java

```java
@Repository
public class CustomShampooDaoImpl implements CustomShampooRepository {
    @PersistenceContext
    private EntityManager entityManager;

    @Transactional
    public void create(BasicShampoo basicShampoo){
        entityManager.persist(basicShampoo);
    }
}
```

Inject Entity Manager

Single Transaction

Spring Custom Configuration

Java-Based Setup

Application Properties

- So far we've configured our project with a spring properties file:

application.properties

```
#Data Source Properties
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/shampoo_company?useSSL=fa
lse&createDatabaseIfNotExist=true
spring.datasource.username = root
spring.datasource.password = 1234
```

Connection properties

Java-Based Configuration

Configuration Class

JavaConfig.java

Repositories Directory

```
@Configuration
@EnableJpaRepositories(basePackages = "com.demo.dao")
@EnableTransactionManagement
@PropertySource(value = "application.properties" )
public class JavaConfig {
       //Add configuration
}
```

Property File

Java-Based Configuration

JavaConfig.java

Data Source Connection

```
    @Autowired
    private Environment environment;

    @Bean
    public DataSource dataSource() {

    DriverManagerDataSource driverManagerDataSource = new DriverManagerDataSource();
driverManagerDataSource.setDriverClassName(environment.getProperty("spring.datasource.driverClassName"));
    driverManagerDataSource.setUrl(environment.getProperty("spring.datasource.url"));
    driverManagerDataSource.setUsername(environment.getProperty("spring.datasource.username"));
    driverManagerDataSource.setPassword(environment.getProperty("spring.datasource.password"));
    return driverManagerDataSource;
}
```

Java-Based Configuration

JavaConfig.java

```
@Bean
public EntityManagerFactory entityManagerFactory() {

    HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
    vendorAdapter.setDatabase(Database.MYSQL);
    vendorAdapter.setGenerateDdl(true);
    vendorAdapter.setShowSql(true);
    LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();
    factory.setJpaVendorAdapter(vendorAdapter);
    factory.setPackagesToScan("com.demo.domain");
    factory.setDataSource(dataSource());
    Properties jpaProperties = new Properties();
    jpaProperties.setProperty("hibernate.hbm2ddl.auto","validate");
    jpaProperties.setProperty("hibernate.format_sql", "true");
    factory.setJpaProperties(jpaProperties);
    factory.afterPropertiesSet();
    return factory.getObject();
}
```

JPA Configuration

Models Package

Java-Based Configuration

JavaConfig.java

```
@Bean
public PlatformTransactionManager transactionManager() {

   JpaTransactionManager txManager = new JpaTransactionManager();

   txManager.setEntityManagerFactory(entityManagerFactory());

   return txManager;
}
```

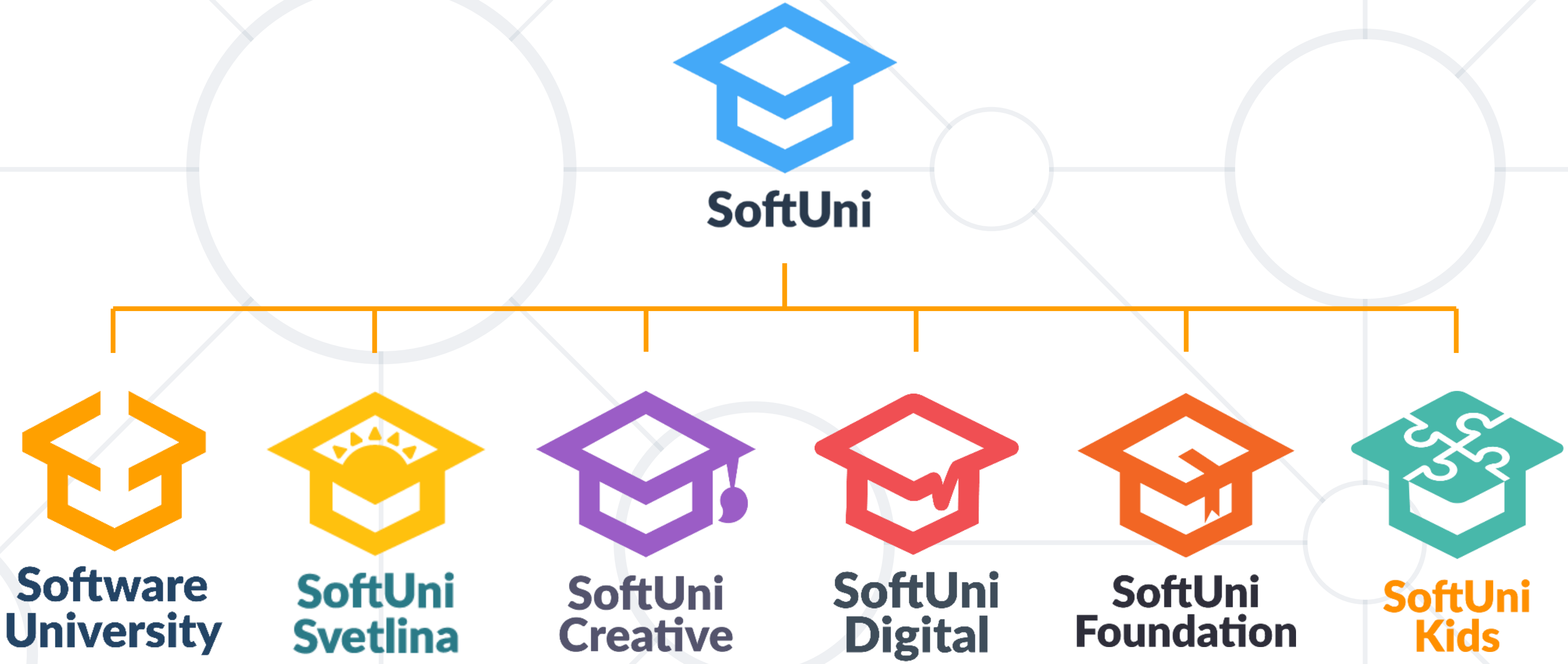
Transaction Manager Configuration

Summary

- Spring Data translates methods to SQL Queries
- We can write custom queries
  - JPQL syntax on entity classes
- Repositories can be inherited
  - Reduces code duplication for inherited entities

Software University

Questions?



SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

STEMO
Computer Systems & Software

SoftUni Organizational Partners

Software University

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

FORUM

License