

# Spring Data Advanced Querying

Query Methods, JPQL Advanced Repositories, Spring Configuration



**SoftUni Team**  
Technical Trainers



**SoftUni**  
Foundation



**Software University**

<http://softuni.bg>

# Table of Content

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



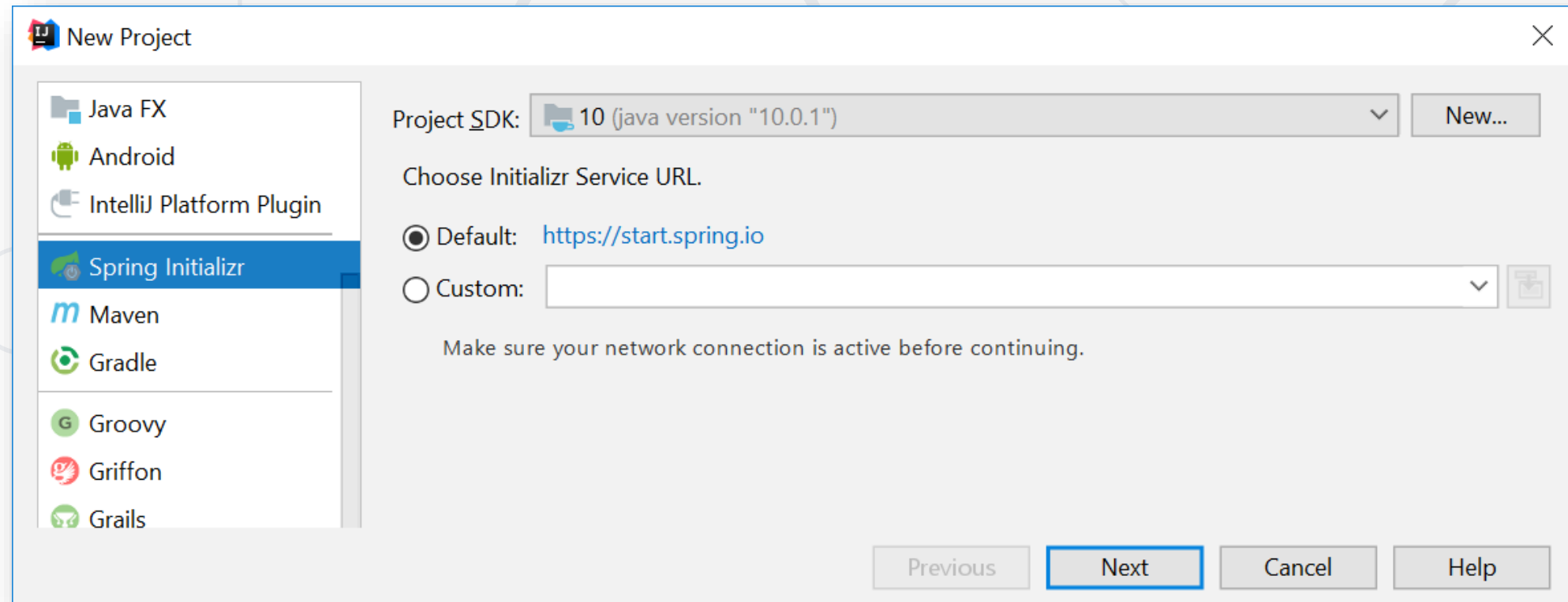
sli.do

**#JavaDb**



# Querying

## Retrieving Data by Custom Queries



# Spring Project (2)

**New Project**

Group:

Artifact:

Type:

Packaging:

Java Version:

Language:

Version:

Name:

Description:

Package:

**New Project**

Dependencies

Spring Boot

Core	<input type="checkbox"/> JPA
Web	<input type="checkbox"/> JOOQ
Template Engines	<input type="checkbox"/> MyBatis
SQL	<input type="checkbox"/> JDBC
NoSQL	<input type="checkbox"/> H2
Cloud Core	<input type="checkbox"/> HSQLDB
Cloud Config	<input type="checkbox"/> Apache Derby
Cloud Discovery	<input checked="" type="checkbox"/> MySQL

## ShampooRepository.java

**@Repository**

```
public interface ShampooDao extends JpaRepository  
<Shampoo, Long> {  
  
    List<Shampoo> findByBrand(String brand);  
  
}
```

Query method

Parameter

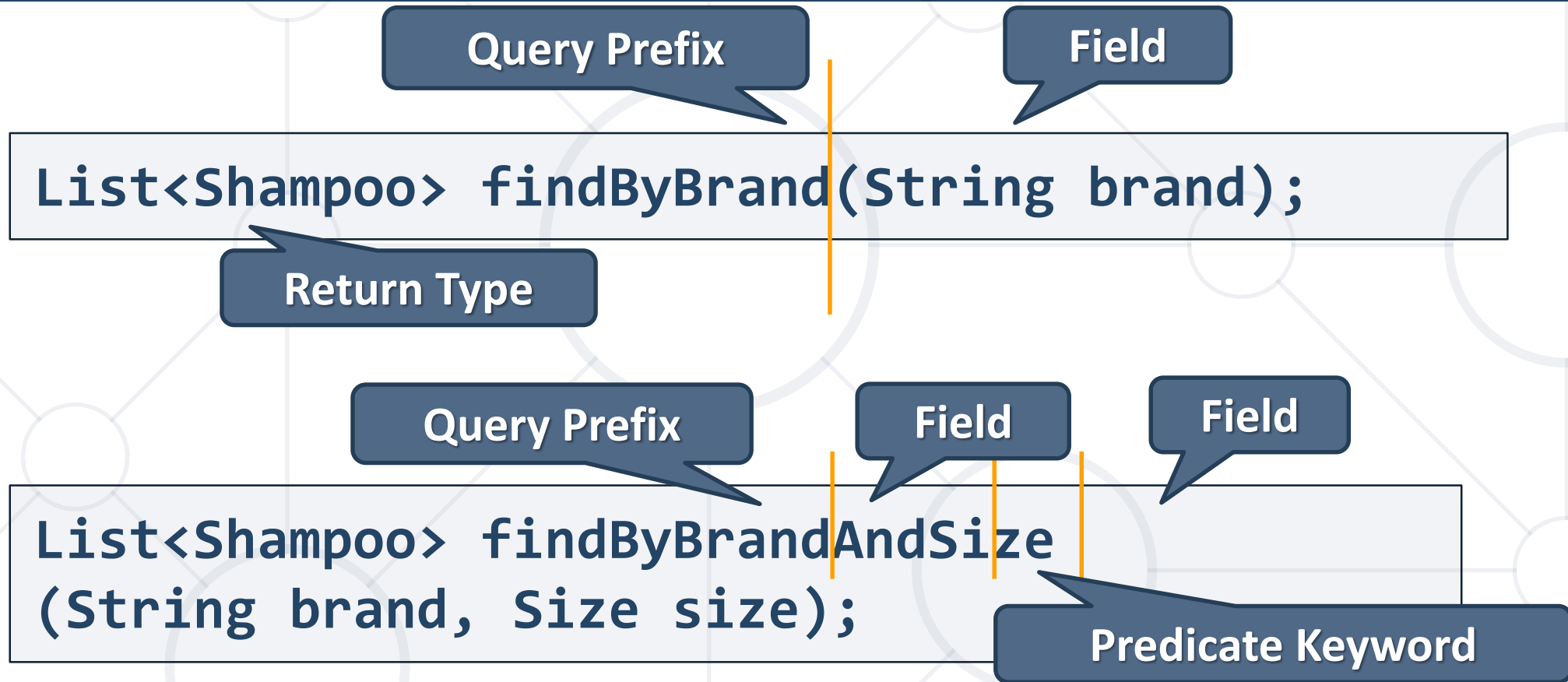
The method translates  
to the following query:



## SQL

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

Parameter





## ShampooRepository.java

**@Repository**

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

Query method

Paramater

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

```
}
```

Paramater

### 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

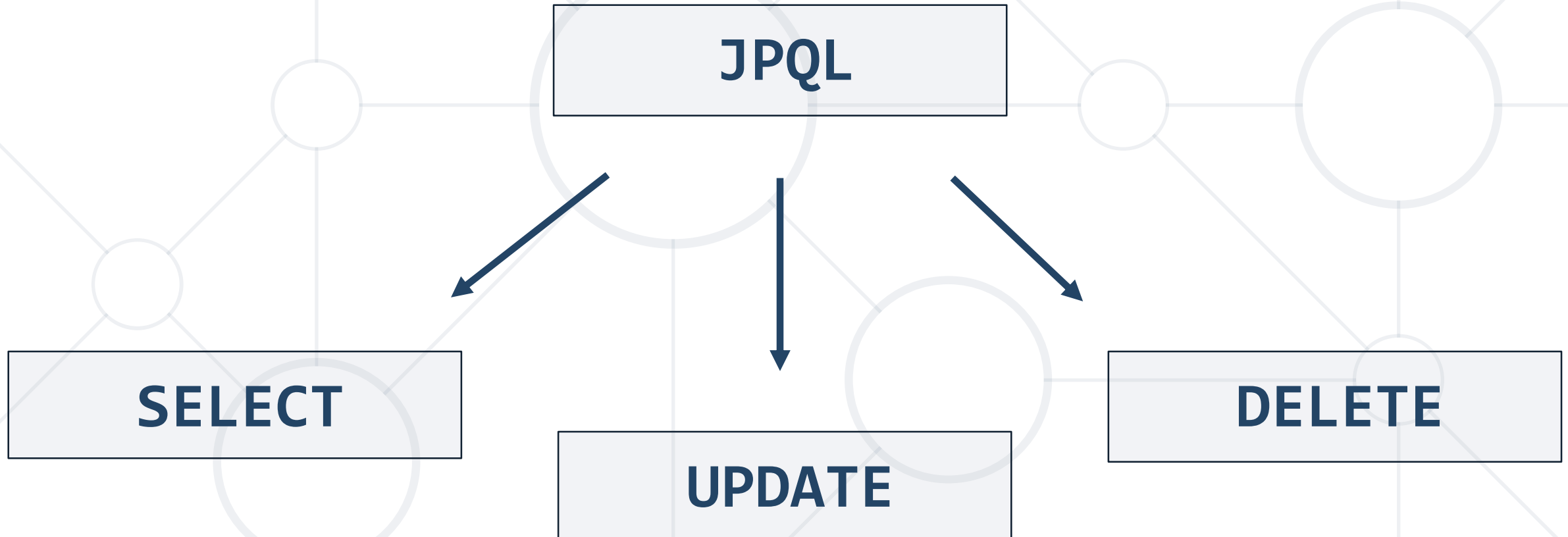
```
@Repository
public interface ShampooRepository extends JpaRepository<Shampoo, Long> {
    List<Shampoo> getAllBySizeOrderBySize(sizeValue);
}
```



# JPQL

## Java Persistence Query Language

- **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



Entity Class

Field

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

Object

Aalias

Parameter

Object

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

Class

Join

Field

Parameter



## ■ Update:

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

Parameter

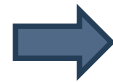
## ■ Delete:

```
"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, Long>{
```

```
    @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

- 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

```
public interface CustomShampooRepository {  
    void create(BasicShampoo basicShampoo);  
}
```

## CustomShampooRepositoryImpl.java

```
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**



- 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/neck_and_elbow?useSSL=false&createDatabaseIfNotExist=true  
spring.datasource.username = root  
spring.datasource.password = 1234
```

Connection properties

Configuration  
Class

JavaConfig.java

Repositories  
Directory

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

Property File

## JavaConfig.java

**@Autowired**

private Environment environment;

**@Bean**

public DataSource dataSource() {

**Data Source Connection**

```
    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;
}
```

## 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.neckandelbows.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

## JavaConfig.java

### Transaction Manager Configuration

**@Bean**

```
public PlatformTransactionManager transactionManager() {  
  
    JpaTransactionManager txManager = new JpaTransactionManager();  
  
    txManager.setEntityManagerFactory(entityManagerFactory());  
  
    return txManager;  
}
```

## JavaConfig.java

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

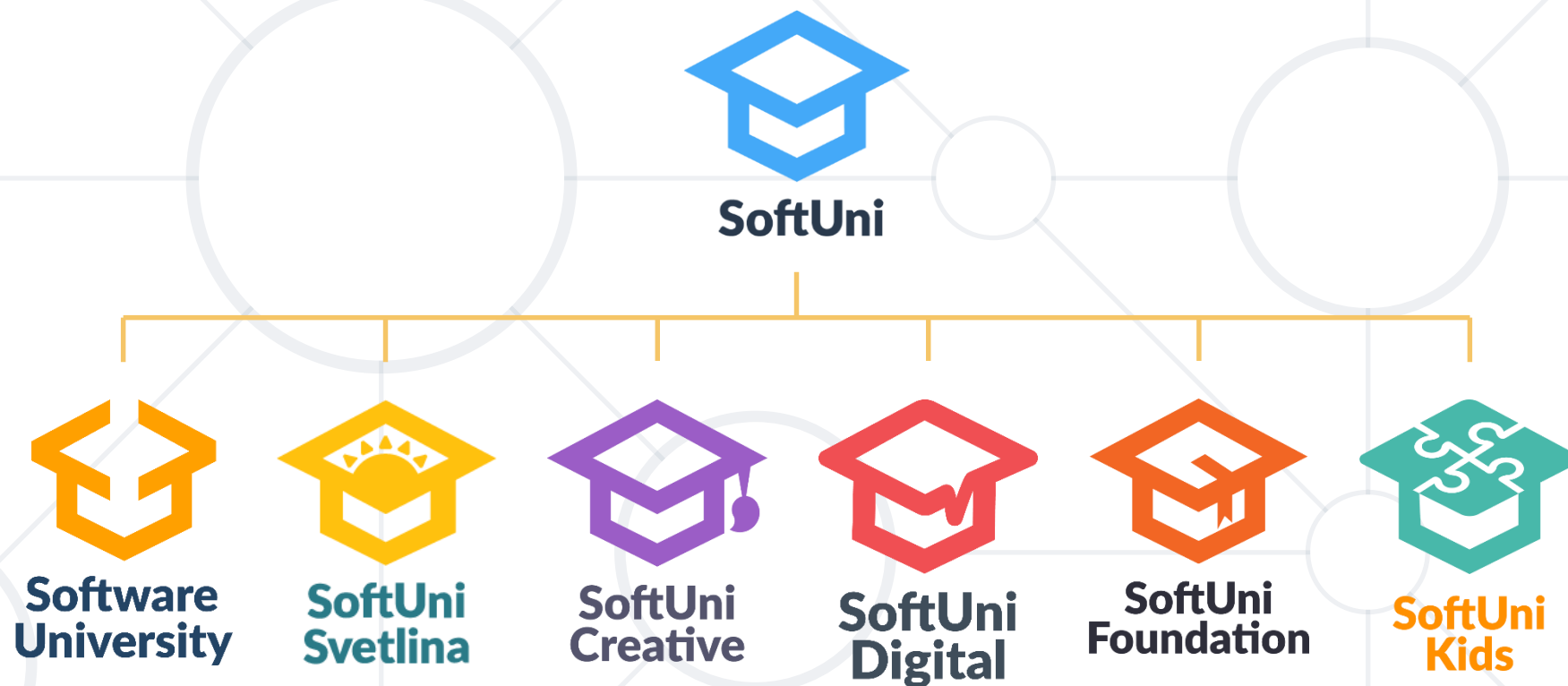
    @Bean
    public CustomShampooDaoImpl shampooDaoImpl(){
        return new CustomShampooDaoImpl();
    }
}
```

Bean Definition

- 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



# Questions?





# SoftUni Diamond Partners



**XS**software



**SBTech**



telenor



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето утре*

**SUPER  
HOSTING  
.BG**

**INDEAVR**

*Serving the high achievers*



**INFRAGISTICS®**

**LIEBHERR**



æternity



**codexio**

# SoftUni Organizational Partners



OneBit  
SOFTWARE



 codexio

# Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - [softuni.bg](http://softuni.bg)
- Software University Foundation
  - <http://softuni.foundation/>
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

