Spring Data Advanced Querying

Query Methods, JPQL Advanced Repositories, Spring Configuration

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions

Software University

sli.do

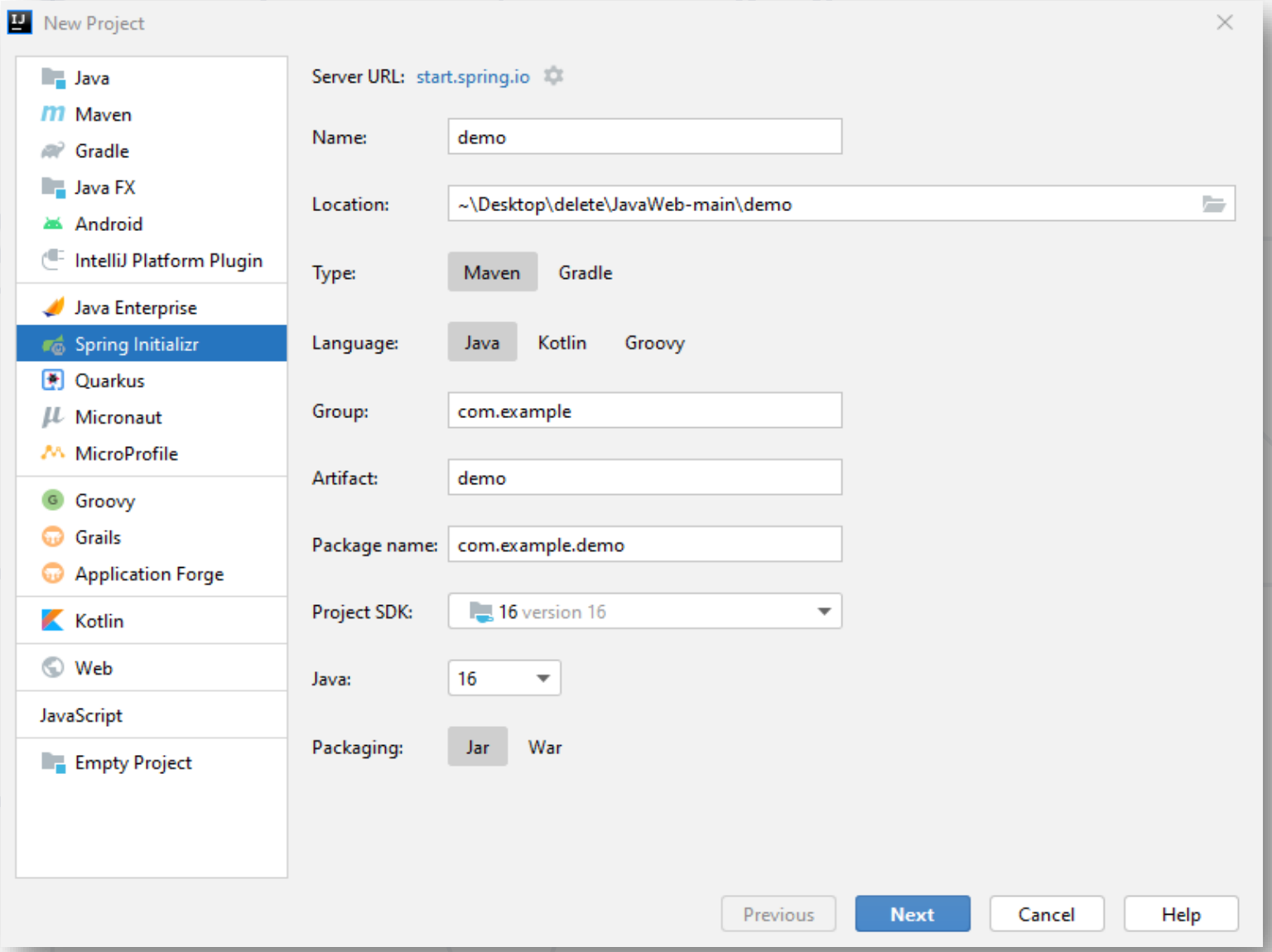
#Java-DB

Table of Contents

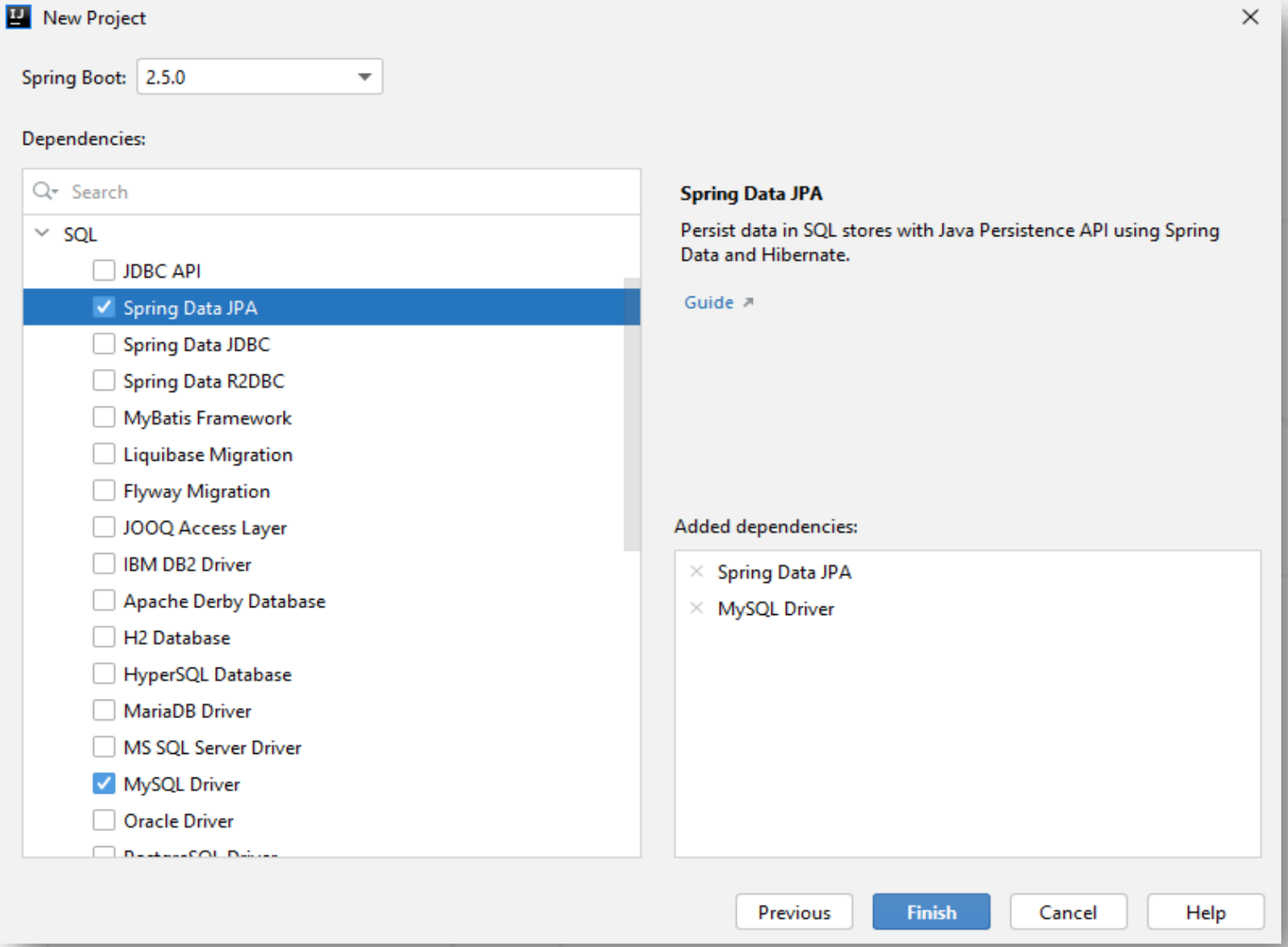
Retrieving Data by Custom Queries

Querying

Spring Project



Spring Project



application.properties – simple example

Software University

application.properties

```
#Data Source Properties
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/1working?useSSL=false&createDatabaseIfNotExist=
true
spring.datasource.username=root
spring.datasource.password=12345
#JPA Properties
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
spring.jpa.properties.hibernate.format_sql = TRUE
spring.jpa.hibernate.ddl-auto = update
spring.jpa.open-in-view=false
###Logging Levels
# Disable the default loggers
logging.level.org = WARN
logging.level.blog = WARN
#Show SQL executed with parameter bindings
logging.level.org.hibernate.SQL = DEBUG
logging.level.org.hibernate.type.descriptor = TRACE
```

Query Methods

ShampooRepository.java

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

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

Query method

Paramater

SQL

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

Paramater

Query Lookup

Query Prefix

Field

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

Return Type

Query Prefix

Field

Field

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

Predicate Keyword

Query Methods

ShampooRepository.java

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

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

Query method

Paramater

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> getAllBySizeOrderById(Size sizeValue);
}
```

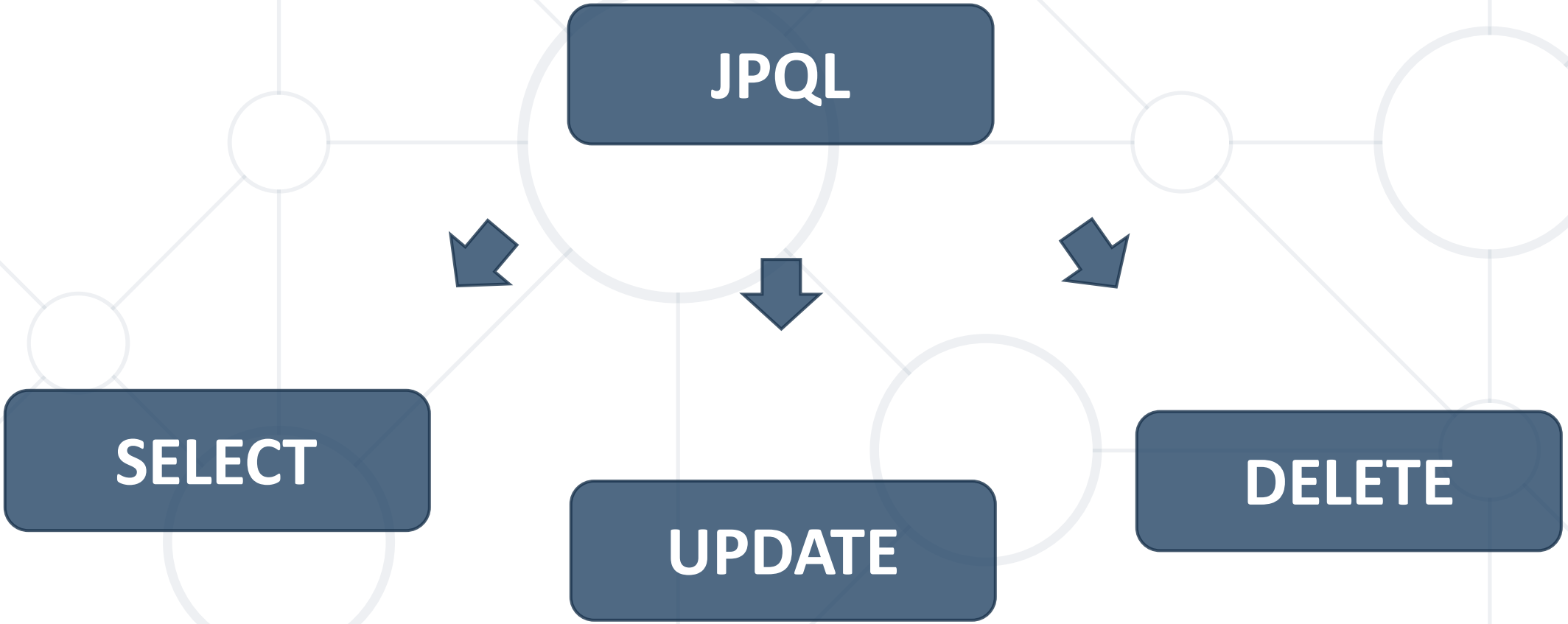
Java Persistence Query Language

JPQL

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 Syntax

Entity Class

Field

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

Object

Alias

Parameter

JPQL Join Syntax

Object

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

Class

Join

Field

Paramater

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 the 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);
}
```

Repository Inheritance

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 of their 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

Software University

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

```
@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=false&
createDatabaseIfNotExist=true
spring.datasource.username = root
spring.datasource.password = 1234
```

Connection properties

Java-Based Configuration

Configuration Class

JavaConfig.java

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

Repositories Directory

Property File

Java-Based Configuration

JavaConfig.java

```
    @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;
}
```

Data Source Connection

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

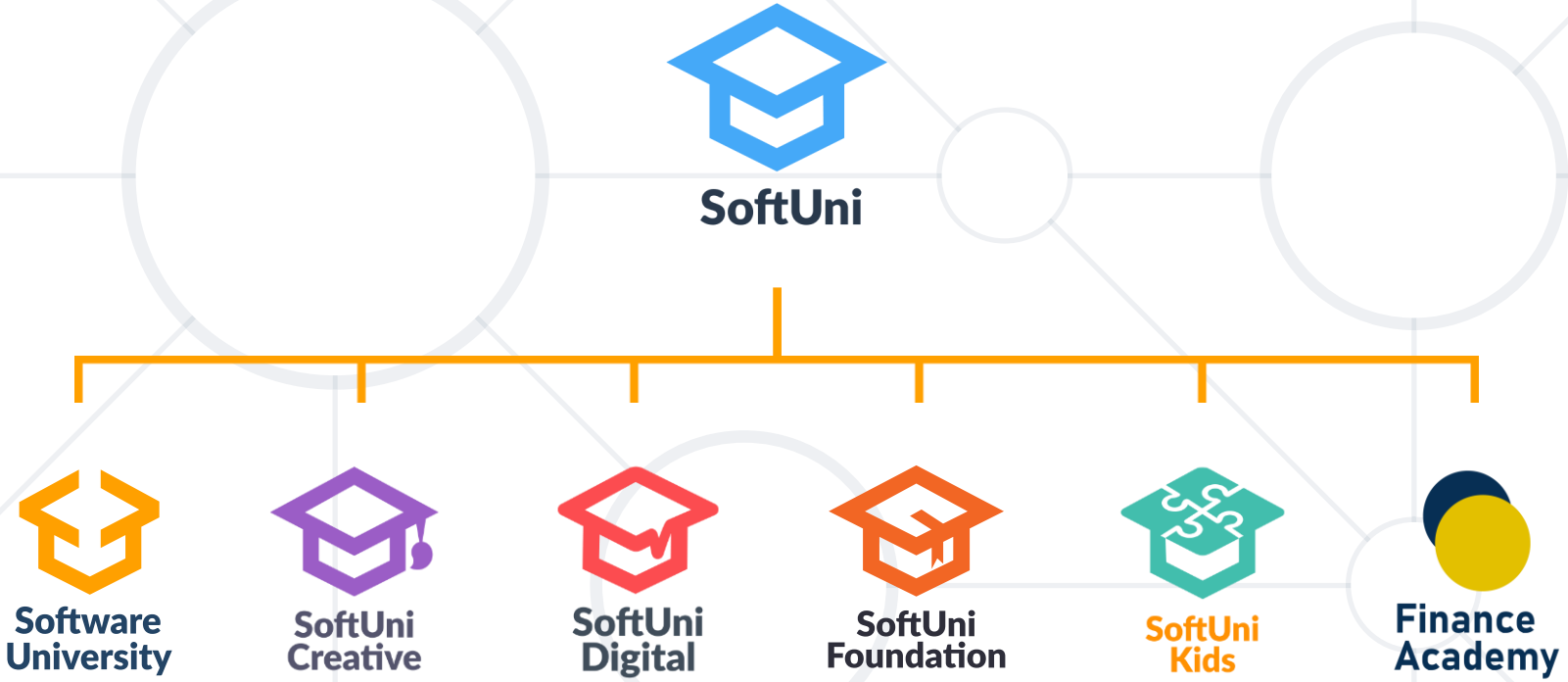
Transaction Manager Configuration

```
@Bean
public PlatformTransactionManager transactionManager() {

    JpaTransactionManager txManager = new JpaTransactionManager();

    txManager.setEntityManagerFactory(entityManagerFactory());

    return txManager;
}
```

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

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank
Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR
Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

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

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg