Spring Data Introduction

Spring Data, Repositories, Services

SoftUni Team

Technical Trainers

Table of Content

1. Spring Data Framework

2. Spring Data Repositories

3. Spring Data Query Creation

4. Spring Data Services

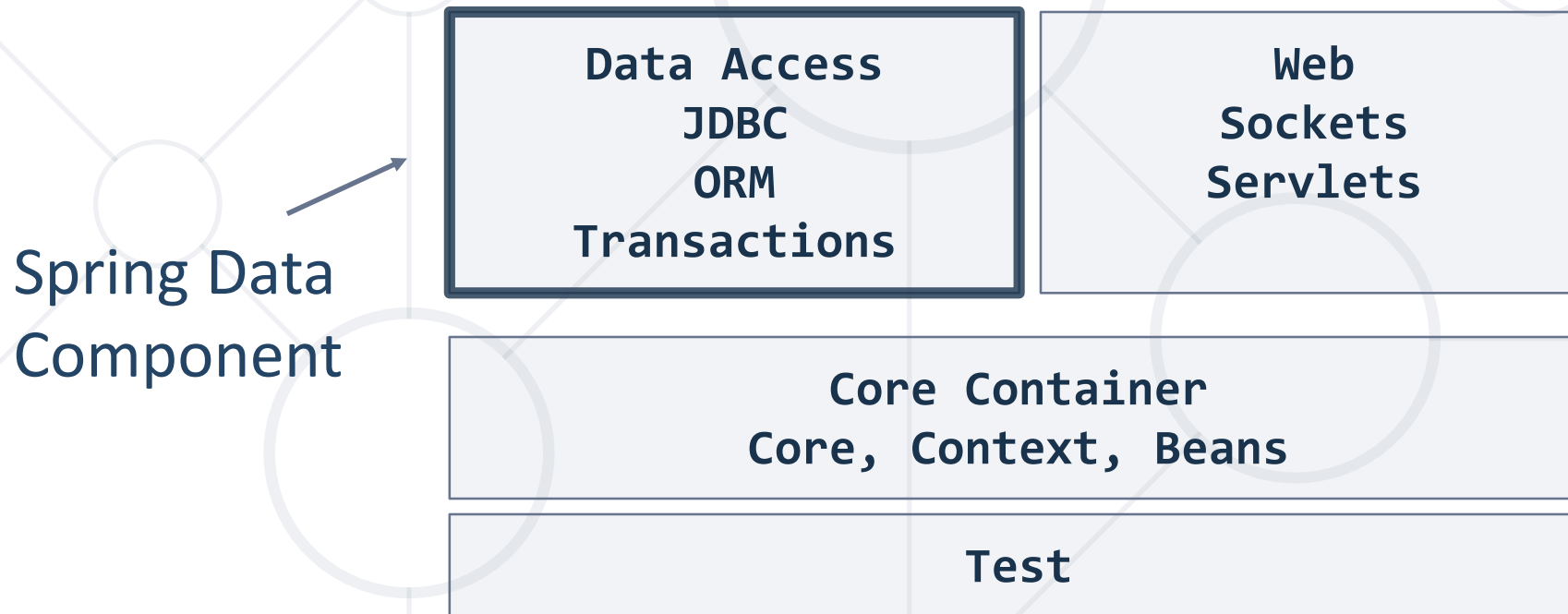**sli.do**

#JavaDb

Spring Data Framework
Spring Framework Ecosystem

What is Spring Framework

- Application framework for the Java Platform

    - Technology stack - includes several modules that provide a range of services

Spring Data
Component

Data Access JDBC ORM Transactions	Web Sockets Servlets

Core Container
Core, Context, Beans

Test

Spring Framework Overview

What is Spring Data

- Library that adds an **extra layer of abstraction** on the top of our JPA provider

- Provides:

  - Dynamic query derivation from repository method names

  - Possibility to integrate custom repositories and many more

- What Spring Data is not:

  - **Spring Data JPA is not a JPA provider**

Spring Data Role

```
Spring Data
```

```
Hibernate, EclipseLink etc.
```

JPA

RDBMS　　　**NRDBMS**
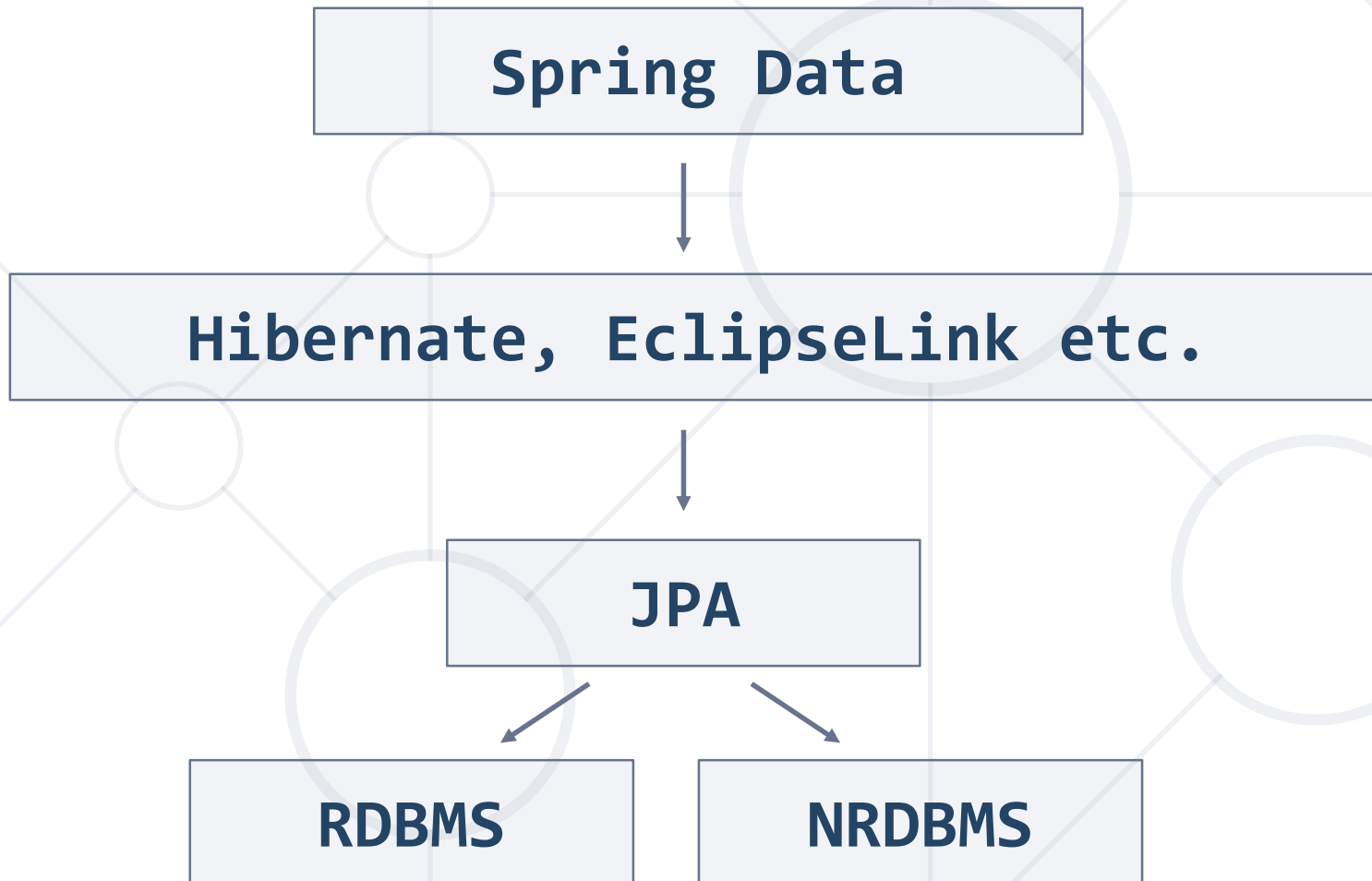
Extra layer of abstraction over the used ORM

Spring Boot – Convention over configuration

- Creates stand-alone Spring applications
    - Provide opinionated 'starter' POMs to simplify your Maven confi guration
- Automatically configure Spring whenever possible
- Absolutely no code generation and no requirement for XML con figuration

Dependencies

SoftUni Foundation

pom.xml
| ```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.1.RELEASE</version>
</parent>
``` |

Dependencies (2)

```xml
                                pom.xml

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

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
</dependencies>
```

Spring Data

MySQL Connector

Build

pom.xml

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Java compile version

Configuration

- Spring boot configurations are held in a **application.properties** file

application.properties

```
#Data Source Properties
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.datasource.url =
jdbc:mysql://localhost:3306/school?useSSL=false
spring.datasource.username = root
spring.datasource.password = 1234


#JPA Properties
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.properties.hibernate.format_sql = TRUE
spring.jpa.hibernate.ddl-auto = create-drop
```

Database Connection

JPA properties

Configuration (2)

```
                    application.properties

…
###Logging Levels
# Disable the default loggers        Loggin settings
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
```
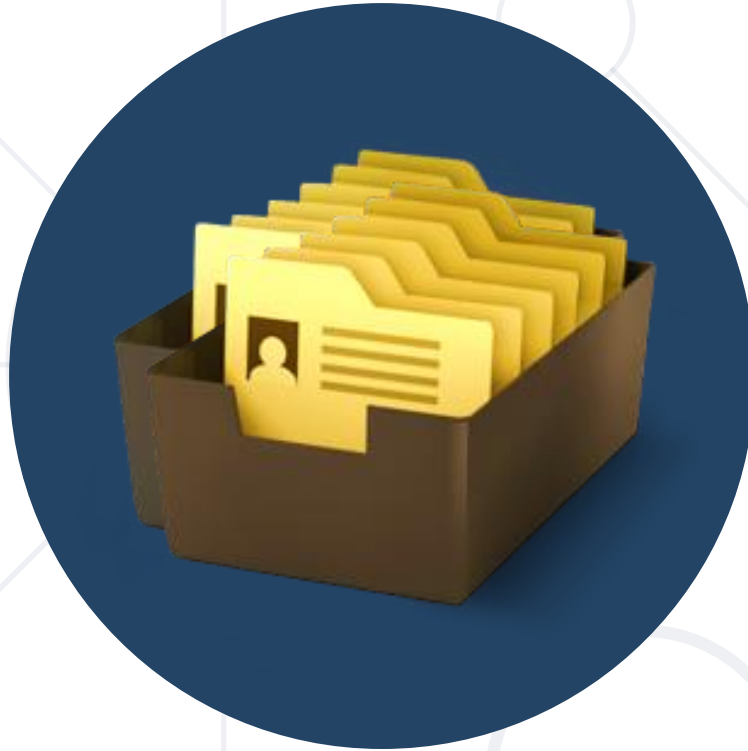
Spring Data Repositories
Spring Framework Ecosystem

Spring Repository

- Abstraction to significantly reduce the amount of boilerplate code required to implement data access layers

  - Perform CRUD Operations

  - Automatically generates JPQL/SQL code

  - Highly customizable

Built-in CRUD Operations

```
JPA REPOSITORY
- <S extends T> S save(S var1);
- <S extends T> Iterable<S>
save(Iterable<S> var1);
- T findOne(ID var1);
- boolean exists(ID var1);
- Iterable<T> findAll();
- long count();
- void delete(ID var1);
void deleteAll();
…
```

Spring Data Query Creation
Building Mechanism

Query Creation

- Queries are created via a query builder mechanism built into Spring Data

    - Strips the prefixes like **find…By**, **read…By**, **query…By** and starts parsing the rest of it

- Spring Data JPA will do a property check

and traverse nested properties

Custom CRUD Operations

StudentRepository.java

```java
@Repository
public interface StudentDao extends CrudRepository<Student,
Long> {
    List<Student> findByMajor(Major major);
}
```

Cutom method

SQL

```sql
SELECT s.*
  FROM students AS s
 INNER JOIN majors AS m
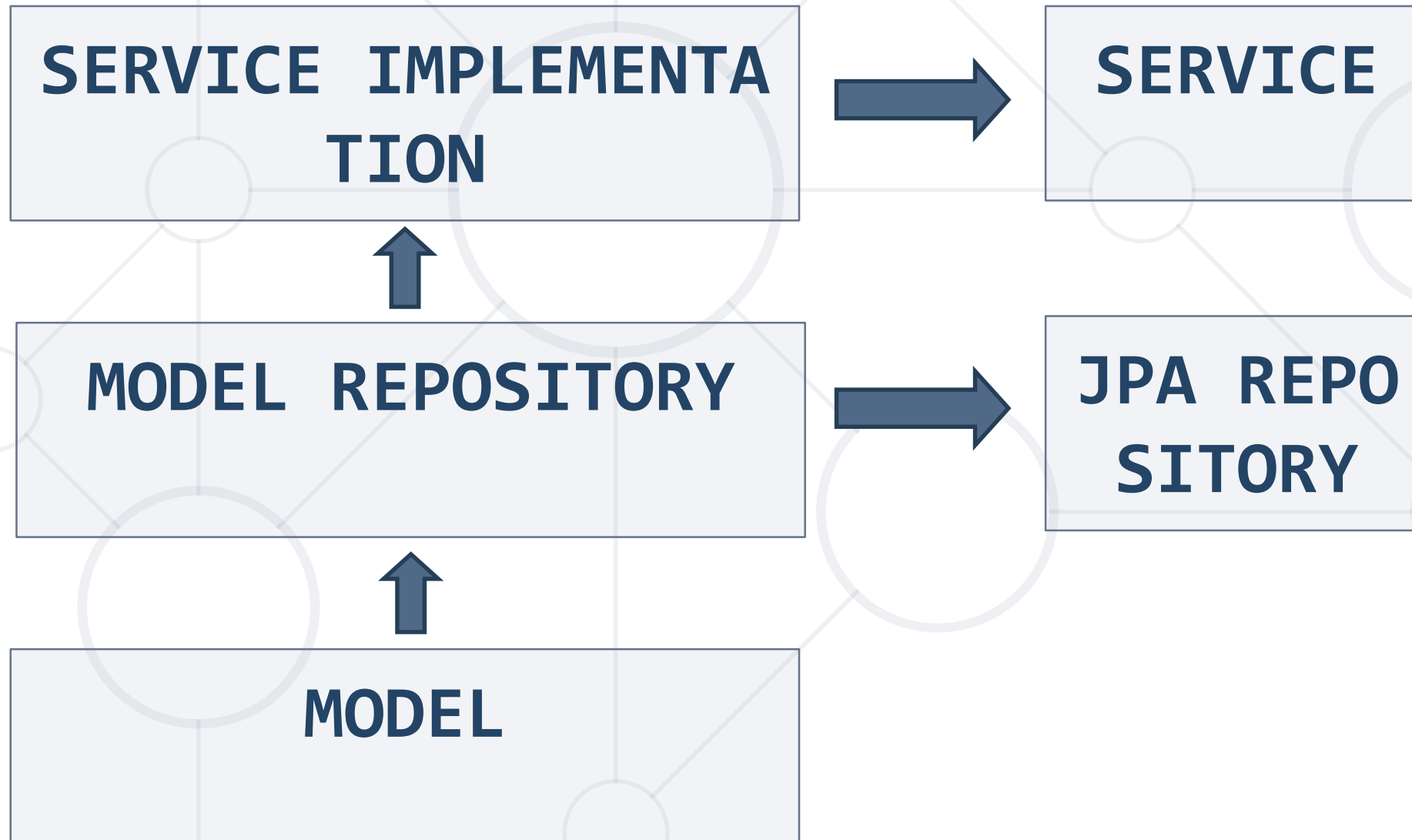    ON s.major_id = m.id
WHERE m.id = ?
```

Query Lookup Strategies

Keyword	Sample	JPQL
And	findByLastnameAndFirstName	… where x.last_name = ?1 and x.firstname = ?2
Or	findByLastnameOrFistname	… where x.lastname = ?1 or x.firstname = ?2
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1

Spring Data Services
Encapsulating Business Logic

Service Pattern

- Service Layer is a design pattern of organizing business logic into layers

  - Service classes are categorized into a particular layer and share functionality

- Main concept is **not exposing details** of internal processes on entities

  - Services **interact closely** with Repositories

Spring Data Architecture

SERVICE IMPLEMENTATION → SERVICE

↑

MODEL REPOSITORY → JPA REPOSITORY

↑

MODEL

Services

StudentService.java

```java
public interface StudentService {

    void register(Student student);

    void expel(Student student);

    void expel(long id);

    Student findStudent(long id);

    List<Student> findSampleByMajor(Major major);
}
```

Business Logic

Services

StudentServiceImpl.java

```java
@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;


    @Override
    public void register(Student student) {
        studentDao.save(student);
    }


    @Override
    public void expel(Student student) {
        studentDao.delete(student);
    }
}
```

Service Implementation

StudentRepository injection

Method implementation

Entry Point

SoftUni Foundation

MainApplication.java

```java
@SpringBootApplication
public class MainApplication {
    public static void main(String[] args) {
        SpringApplication.run(MainApplication.class, args);
    }
}
```

Spring Boot Entry Point

Command Line Runner

CommandLineRunner.java

```java
@Component                          ← Component
public class ConsoleRunner implements CommandLineRunner {
    @Autowired                      ← Student service
    private StudentService studentService;

    @Autowired
    private MajorService majorService;    ← Major service

    @Override
    public void run(String... strings) throws Exception {
        Major major = new Major("Java DB Fundamentals");
        Student student = new Student("John",new Date(), major);
        majorService.create(major);
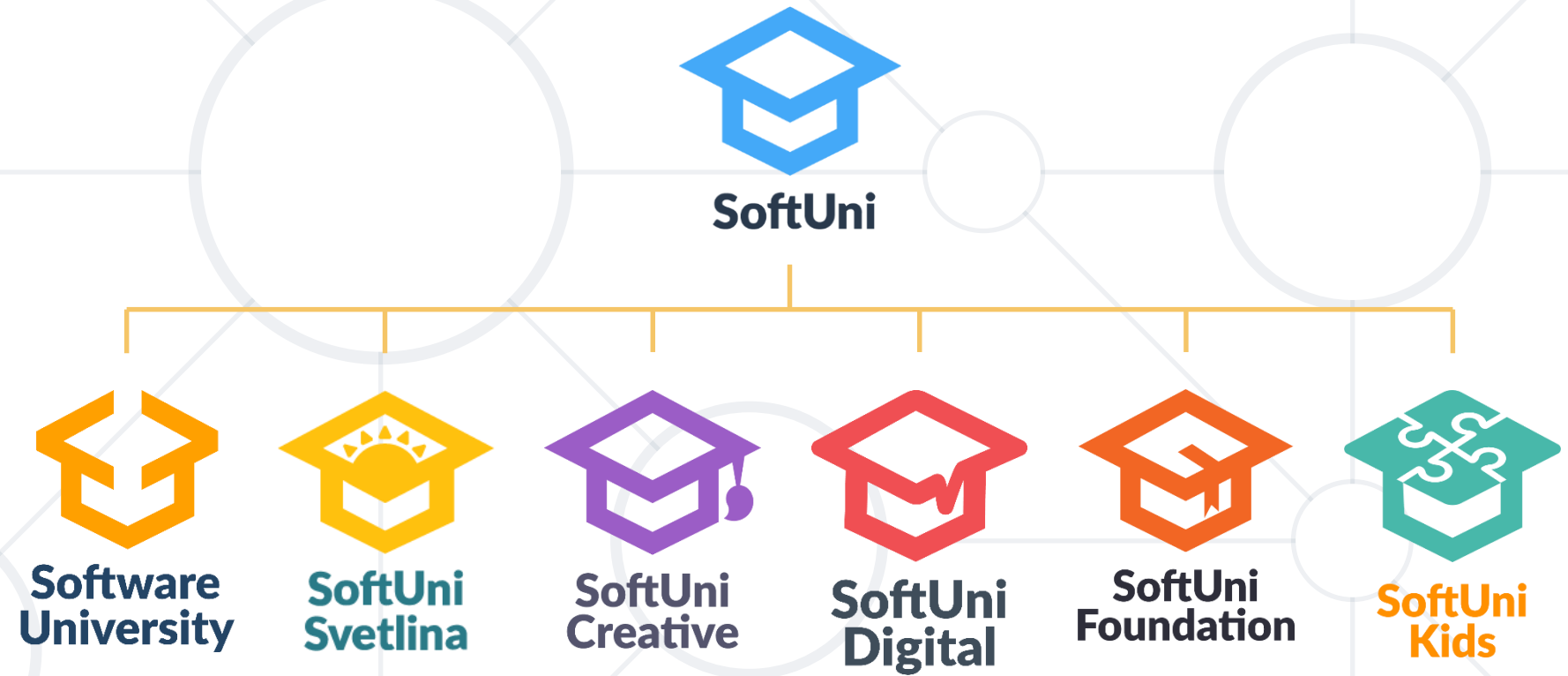        studentService.register(student);    ← Persist data
    }
}
```

Summary

- Spring Data is part of the Spring Framework
  - It is not a JPA Provider, just an abstraction over it
- Spring Data builds queries over conventions
- Main concept of Spring Data are Repositories and Servicesч

Questions?

SoftUni

Software University · SoftUni Svetlina · SoftUni Creative · SoftUni Digital · SoftUni Foundation · SoftUni Kids

https://softuni.bg/trainings/courses

SoftUni Diamond Partners

SoftUni Organizational Partners

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

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