

Spring Data Introduction

Spring Data, Repositories, Services



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>



sli.do
#Java-DB

1. Framework
 - Spring Platform
 - Spring Projects
 - Spring Boot
 - Spring Framework
2. Spring Data Framework
3. Spring Data Repositories
4. Spring Data Query Creation
5. Spring Data Services





Framework

Framework



- Platform for **developing software applications**
- Provides a **foundation** on which **software developers** can **build programs** for a **specific platform**
- Similar **to an API**
 - A Framework **includes an API**
- May include **code libraries**, a **compiler**, and other programs **used in the software development process**



Spring Platform

Spring Platform

- Spring makes **programming Java quicker, easier, and safer for everybody**
- Spring's **focus is on speed, simplicity, and productivity** built by multiple Spring Projects
 - Spring **Boot**
 - Spring **Framework**
 - Spring **Data**



Spring Module

- Spring **Core Container**
 - The base module of Spring and provides Spring containers
- **Aspect-Oriented Programming**
 - Enables implementing cross-cutting concerns
- **Authentication and Authorization**



- **Data Access**
 - Working with **RDBMS** using **JDBC** and **ORM** tools
- **IoC Container**
 - Configuration of application **components and lifecycle management of Java objects**, done mainly via **dependency injection**
- **Testing**
 - Support classes for writing **unit tests and integration tests**



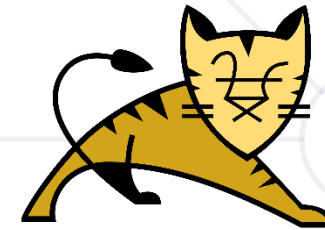


Spring Projects

- Spring **Boot**
 - Makes it easy to **create stand-alone, production-grade Spring based Applications**
- Spring **Framework**
 - Provides a **comprehensive programming and configuration model** for modern Java-based enterprise applications - on any kind of deployment platform

- Spring **Data**
 - Spring Data's mission is to **provide a familiar and consistent, Spring-based programming model for data access** while still retaining the special traits of the underlying data store
- Spring **Cloud**
 - Spring Cloud **provides tools for developers to quickly build** some of the **common patterns in distributed systems**

- **Opinionated view** of building production-ready Spring applications



Tomcat



maven

pom.xml



Auto configuration

Spring Framework

- **Open-Source Application framework** and **inversion of control container** for the Java platform
- Core features can be used by any Java application **extensions** for building web applications **on top of the Java EE**



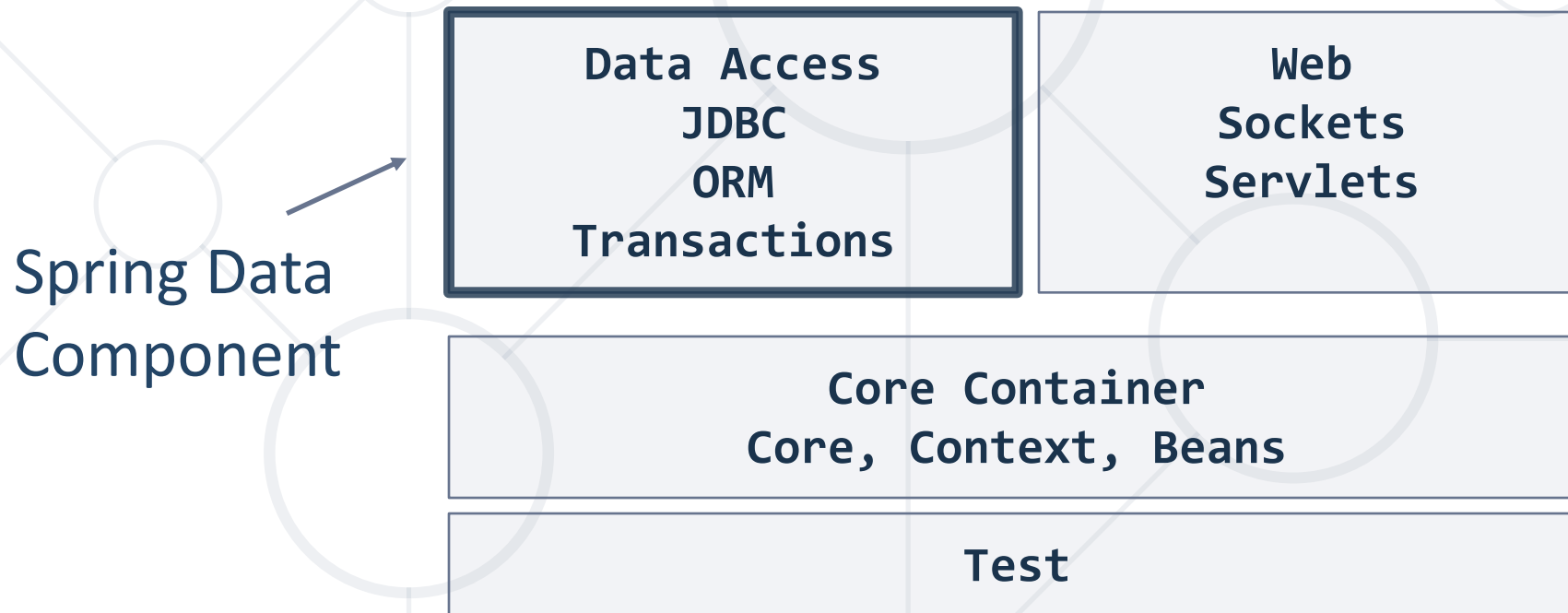


Spring Data Framework

Spring Framework Ecosystem

What is Spring Framework

- Application framework for the Java Platform
 - Technology stack - includes several modules that provide a range of services

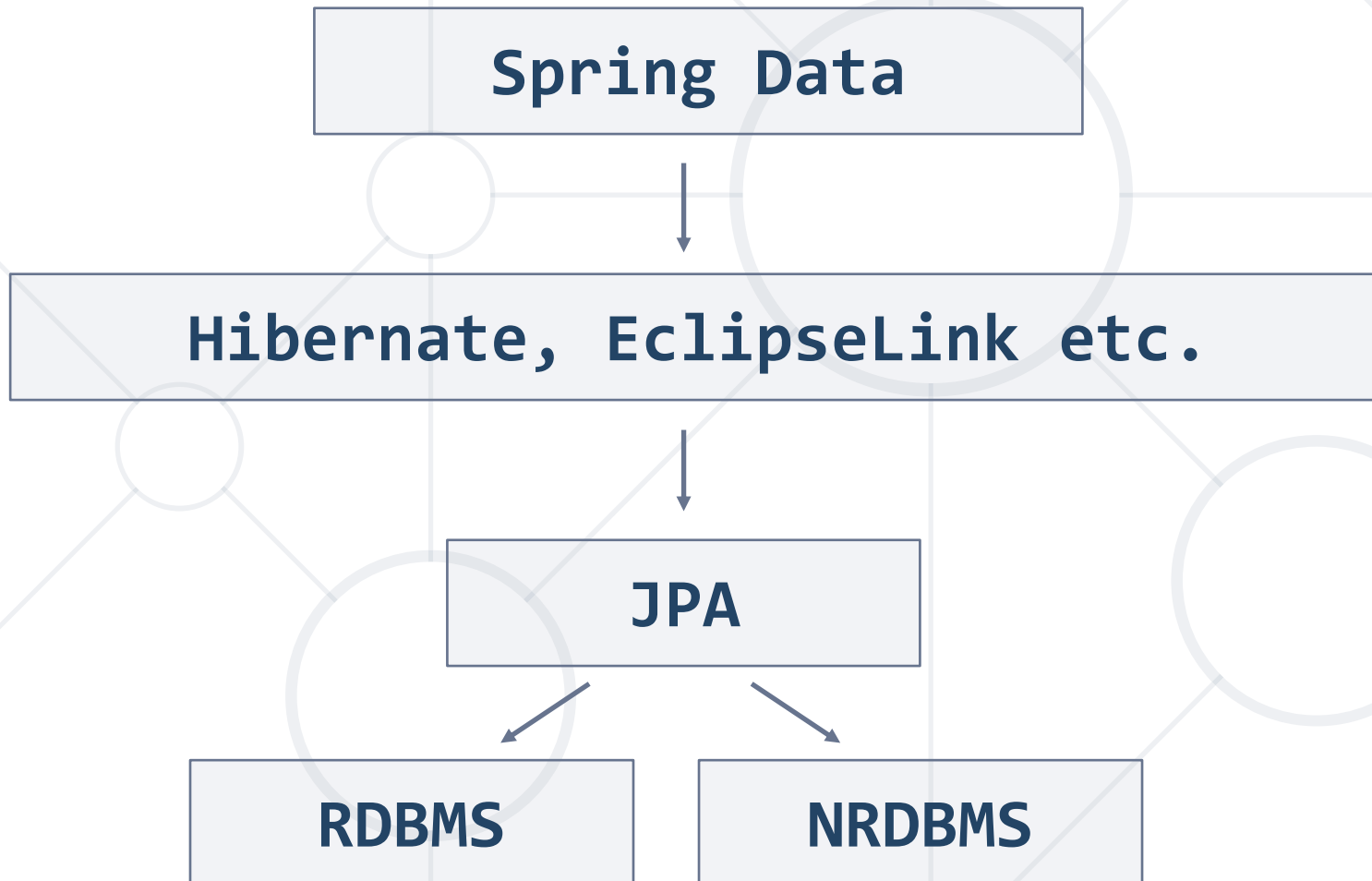


Spring Framework Overview

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**





Extra layer of abstraction
over the used ORM

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

`pom.xml`

```
<parent>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-parent</artifactId>  
</parent>
```

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>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

Spring Data

MySQL Connector

pom.xml

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
      <configuration>
        <source>16</source>
        <target>16</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Java compile
version

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

application.properties

```
#Data Source Properties
spring.datasource.driverClassName =
com.mysql.cj.jdbc.Driver
spring.datasource.url =
jdbc:mysql://localhost:3306/school?useSSL=false
spring.datasource.username = root
spring.datasource.password = 12345
```

Database Connection

```
#JPA Properties
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL8Dialect
spring.jpa.properties.hibernate.format_sql = TRUE
spring.jpa.hibernate.ddl-auto = create-drop
```

JPA properties

application.properties

...

###Logging Levels

Disable the default loggers

logging.level.org = WARN

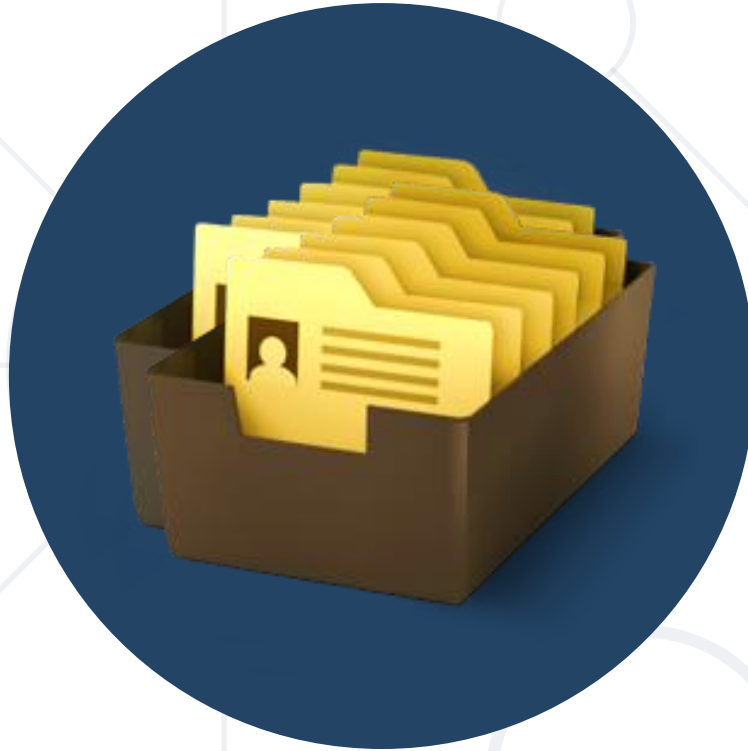
logging.level.blog = WARN

Loggin settings

#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

- 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

- 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



StudentRepository.java

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

Custom method



SQL

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

Query Lookup Strategies

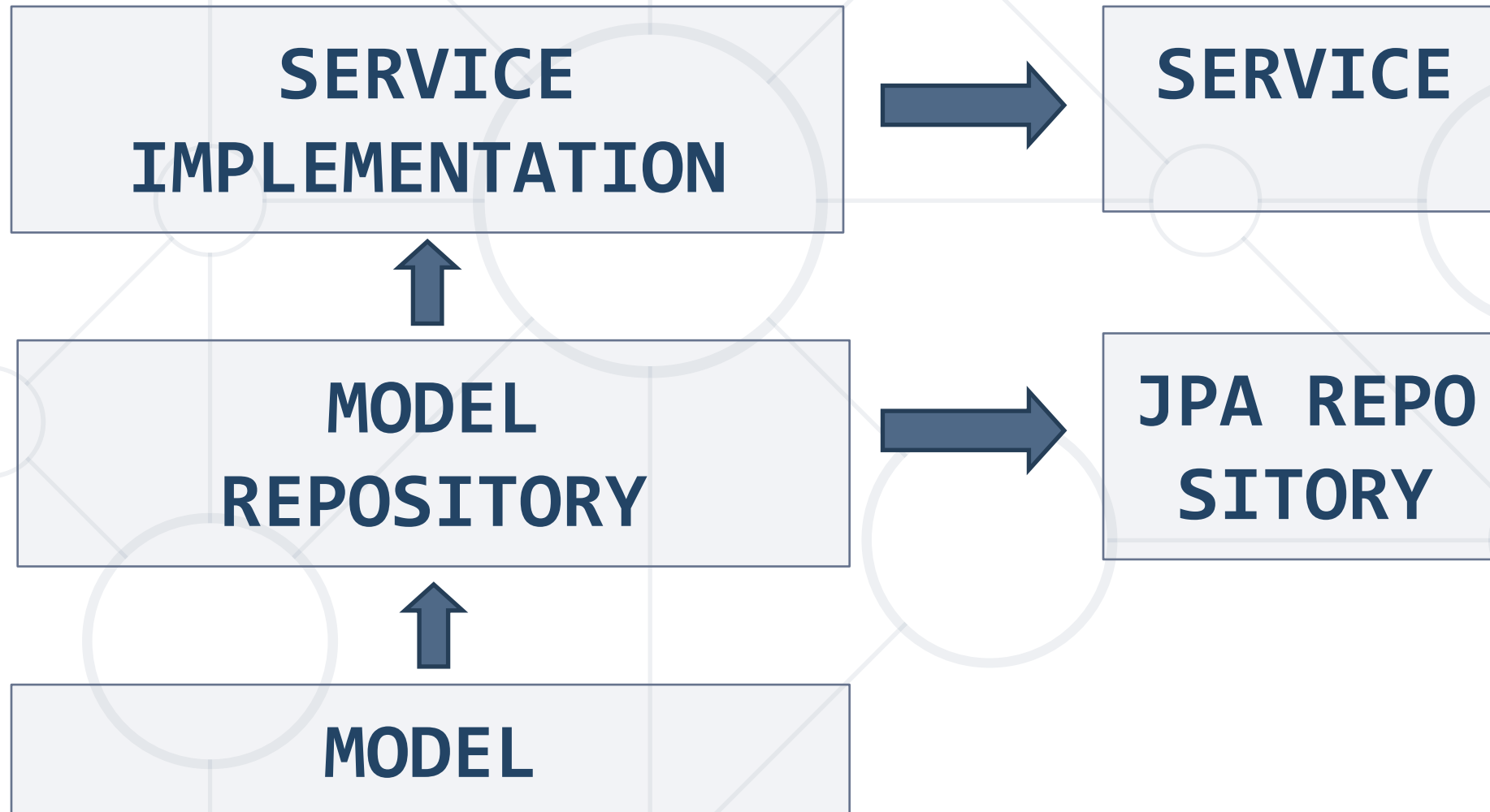
| Keyword | Sample | JPQL |
|------------|-----------------------------------|--|
| And | findByLastnameAndFirstName | ... where x.last_name = ?1 and x.firstname = ?2 |
| Or | findByLastnameOrFirstname | ... where x.lastname = ?1 or x.firstname = ?2 |
| Between | findByStartDateBetween | ... where x.startDate 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 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



StudentService.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

StudentServiceImpl.java

@Service

```
public class StudentServiceImpl implements StudentService {
```

Service Implementation

@Autowired

```
private StudentRepository studentRepository;
```

StudentRepository injection

@Override

```
public void register(Student student) {  
    studentRepository.save(student);  
}
```

Method implementation

@Override

```
public void expel(Student student) {  
    studentRepository.delete(student);  
}  
}
```

MainApplication.java

@SpringBootApplication

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

Spring Boot Entry Point

CommandLineRunner.java

```
@Component
public class ConsoleRunner implements CommandLineRunner {
    @Autowired
    private StudentService studentService;

    @Autowired
    private MajorService majorService;

    @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);
    }
}
```

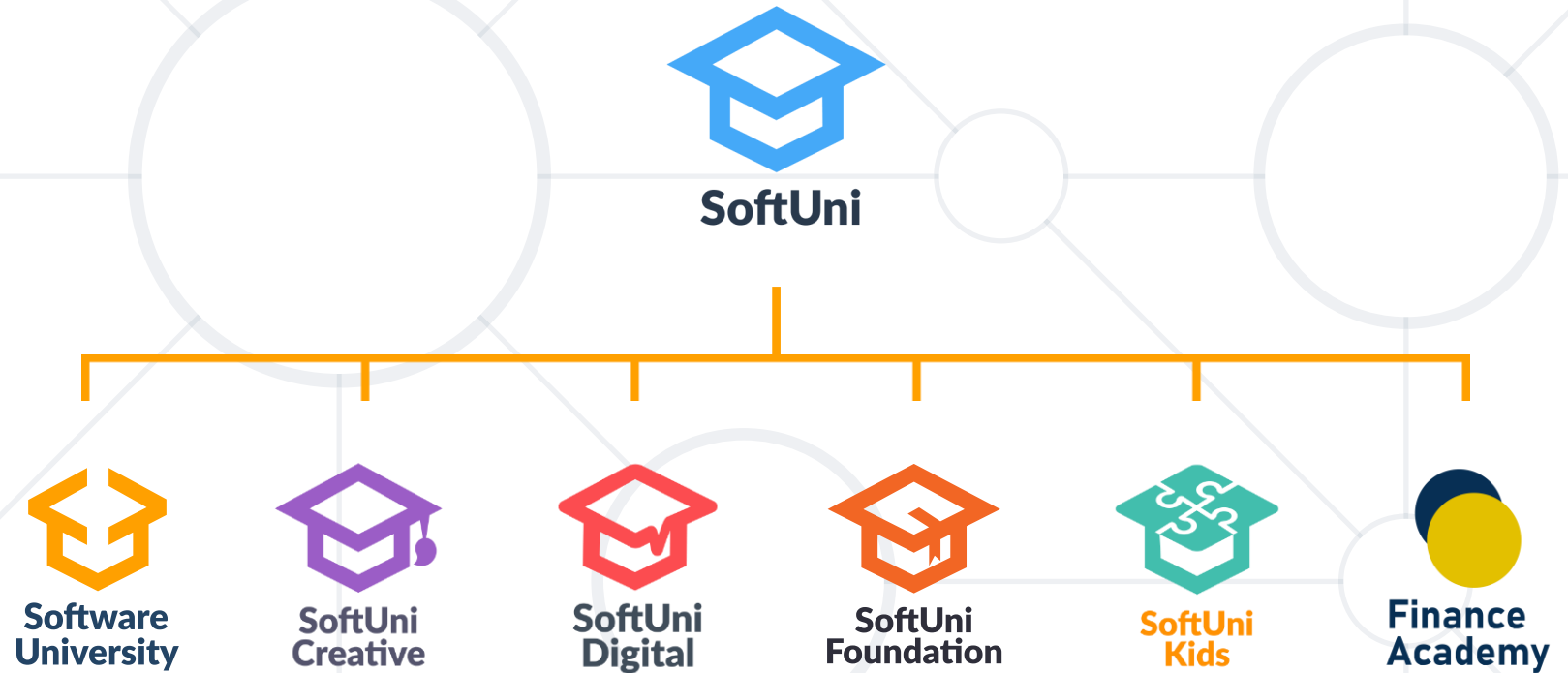
Annotations and actions are highlighted with callouts:

- Component**: Points to `@Component`
- Student service**: Points to `private StudentService studentService;`
- Major service**: Points to `private MajorService majorService;`
- Persist data**: Points to `studentService.register(student);`

- 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 **Services**



Questions?



SoftUni Diamond Partners



- 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



- 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>

