Introduction to Spring Data

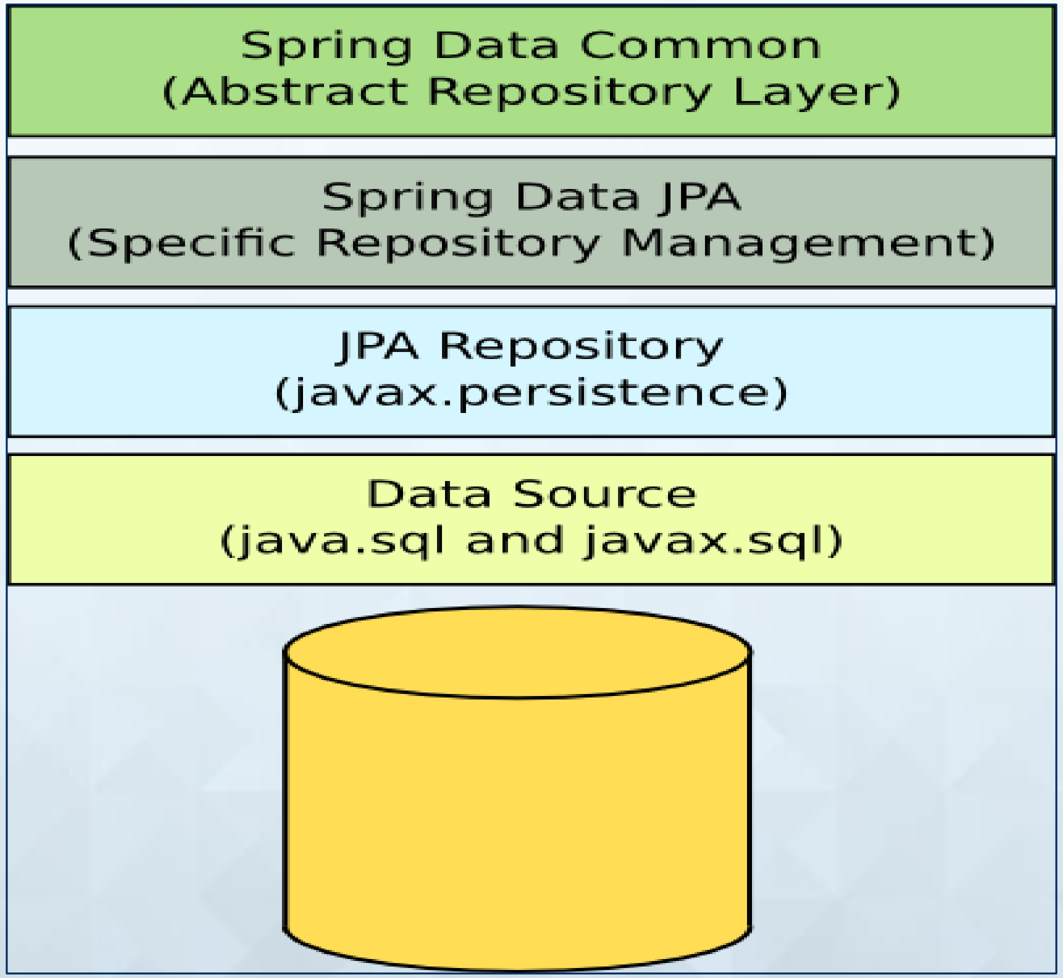
Session Objective

- Spring Data
- Spring Data Modules
- Spring Umbrella
- Spring Data – JDBC
- Spring Data –JPA
- Repositories

Spring Data

- Spring Data’s is used to develop Spring-based program for data access.

- Spring Boot provides an umbrella project to interact with the given database using the sub programs in it.



Spring Data Modules

Few Modules of Spring Data:

- **Spring Data Commons –:**
  Core Spring concepts underpinning every Spring Data module.

- **Spring Data JDBC -:**
  Spring Data repository support for JDBC.

- **Spring Data JPA – :**
  Spring Data repository support for JPA.

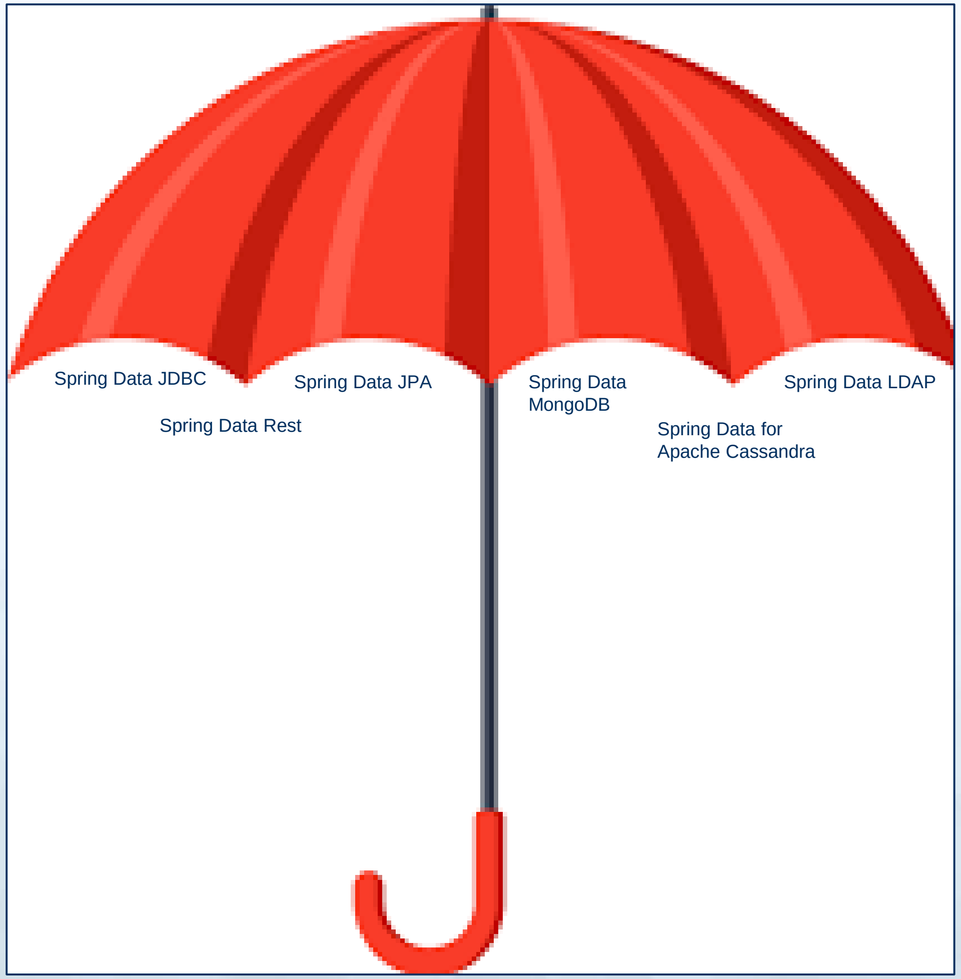
- **Spring Data MongoDB -:**
  Spring based, object-document support and repositories for MongoDB.

- **Spring Data REST - :**
  Exports Spring Data repositories as hypermedia-driven RESTful resources.

Spring Data Commons

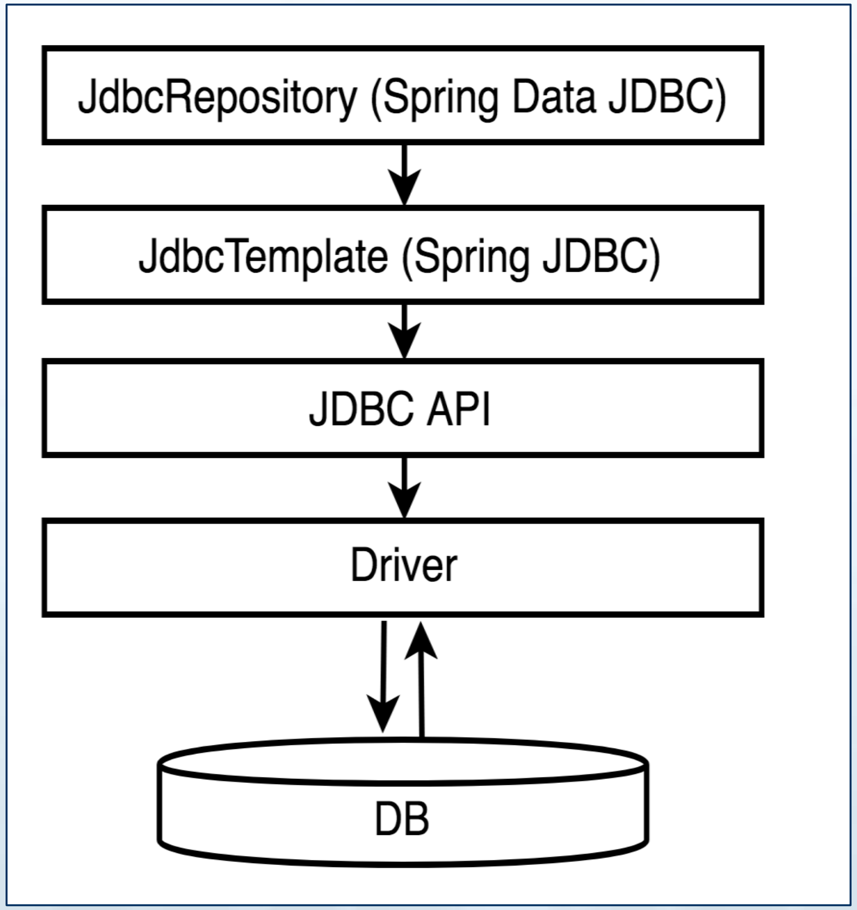
- Spring Data Commons is part of the umbrella Spring Data project it provides shared infrastructure across the Spring Data projects.

- It contains technology neutral repository interfaces as well as a metadata model for persisting Java classes, Powerful Repository and custom object-mapping abstractions.



Spring Data -JDBC

- Spring Data JDBC, is a part of the larger Spring Data family, makes it easy to implement JDBC based repositories.
- This module deals with enhanced support for JDBC based data access layers.
- It makes it easier to build Spring powered applications that use data access technologies.



Spring Data - JDBC

- Spring Data JDBC is simpler conceptually, by embracing the following design decisions:
- No lazy loading or caching is done.
- If you save an entity, it gets saved permanently. There is no dirty tracking.
- Simple model to map entities to tables.

Spring Data JPA

- Sophisticated support to build repositories based on Spring and JPA
- Transparent auditing of domain class
- Pagination support, dynamic query execution, ability to integrate custom data access code
- Validation of @Query annotated queries at bootstrap time
- Support for XML based entity mapping

Spring Data JPA

```
<dependencies>
<dependency>
<groupId>org.springframework.data</groupId>
<artifactId>spring-data-jpa</artifactId>
<version>version-number</version>
</dependency>
</dependencies>
```

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

Repository - JPA-managed entity

The Spring Data allows application programmers to work with data stores using consistent interfaces.

These three core interfaces of Spring Data are:

1) Repository,

2) CrudRepository

3) PagingAndSortingRepository

Repositories

- Application programmers can access data in a consistent way.
- It is easy to switch the underlying storage for a domain entity.
- Specific implementations can provide features specific to data stores.

Repository – Entity class

```
@Entity
@Table(name = "Employee")
public class Emp {
    @Id
    @Column(name = "EmpId")
     private Long id;

    @Column(name = "EmpName")
   private String name;
//setter
//getter
//constuctor
//toString

     }
```

Creating repository for JPA managed class using entity class

Repository – Entity class

Creating repository for JPA managed class using entity class

}

```
public interface EmpRepository extends
CrudRepository<Emp, Long>
 {
   public User findByName(String name); }
```

Invoking the findByName method results in the JPA query

Name must be an attribute on the Emp entity class.

The method name must begin with find, get or read.

Summary

Gives an overview on Spring Data, its modules and repositories.

Gives birds view on Spring Data JDBC, Spring Data JPA

Explore more on spring data

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com