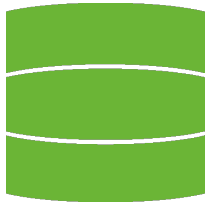
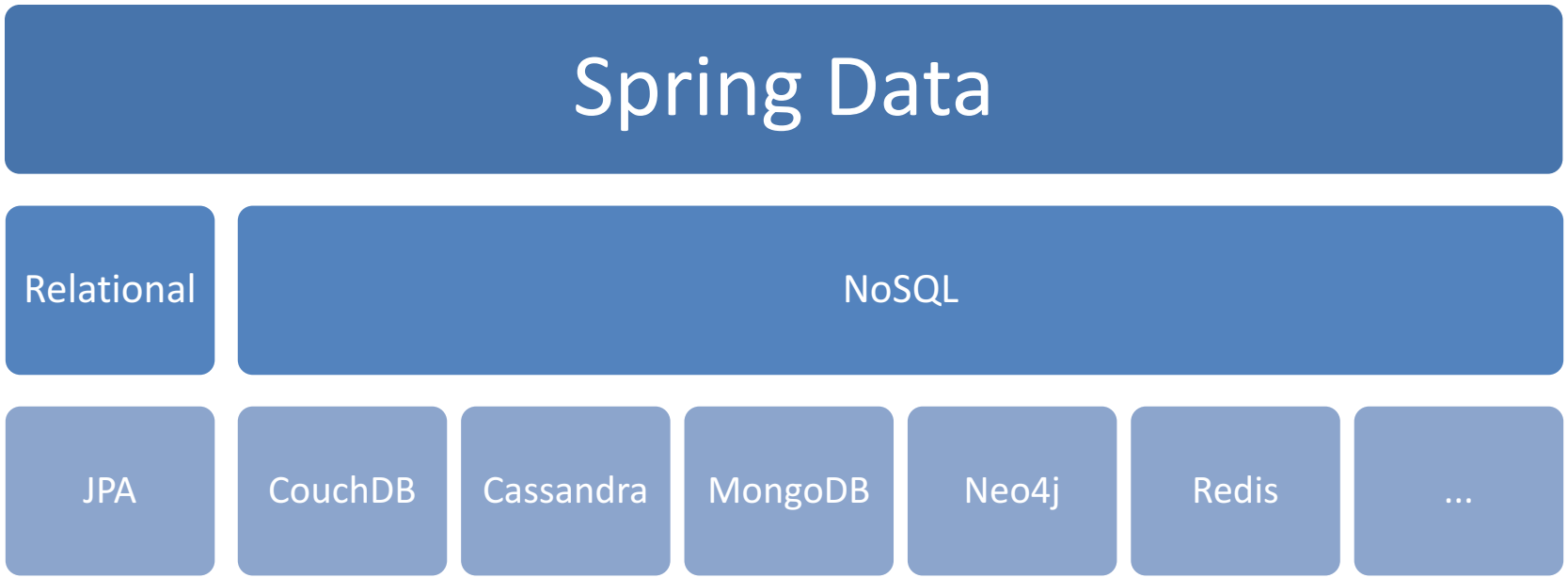
Spring Data JPA

What is Spring Data JPA?

What is Spring Data?



- Umbrella project for data access technologies
- Reduces boiler plate code for data access
- For relational and NoSQL databases



Spring Data JPA Setup

Maven Dependencies

- Adding Spring Data JPA Starter to your project and getting all the necessary dependencies for Transactions, AOP, JDBC, JPA, Hibernate and others.

- Adding H2 (or HSQLDB or Derby) as in-memory DB for faster development

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

Setup

- Starter configures infrastructre Spring beans automatically
  - DataSource
  - LocalContainerEntityMangerFactory
  - PlatformTransactionManager
  - Scans for @Entity -> JPA Mapping

Entities & Repositories

Creating JPA Entities

```
@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;

    protected Employee() {}

    public Employee(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return String.format(
            "Employee[id=%d, firstName='%s', lastName='%s']",
        id,firstName,lastName);
    }
}
```

JPA Repository

- Just an interface, no implementation needed
- Spring Data proxy with implementation
- Crud, paging and sorthing methods provided
- Dynamic finders and @Query support

```
public interface EmployeeRepository extends
    JpaRepository<Employee,Long>{

    List<Employee> findByLastName(String lastName);

    @Query("select e from Employee e where e.lastName = ?")
    List<Employee> findQueryByLastName(String name);
}
```

Repository and Repository Hierarchy

- Repository marker interface
  - Repository<T, ID extends Serializable>

- Repository Interface Hierarchy :
  - JpaRepository<T,ID>
    - PagingAndSortingRepository<T,ID>
      - CrudRepository<T,ID>
        - Repository<T,ID>

Proxy Instance

- Scanning for interfaces which extends Repository<T, ID>

- Methods from interfaces auto generated
  - CRUD methods
  - Paging and Sorting methods
  - Dynamic finders & @Query

Test Run

```
@SpringBootApplication
public class DemoApplication {

    …

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

    @Bean
    public CommandLineRunner demo(EmployeeRepository repository) {
        return(args) ->{
            repository.save(new Employee("Carmen","Bianchi"));
            Employee bianchi = repository.findByLastName("Bianchi");
            log.info(bianchi.toString());
        };
    }
}
```

Important notes

Important notes (1)

- To override default configuration you might use
  - @EnableJpaRepositories
  - @EntityScan
- Stored data at startup deleted? Disable drop & delete

```
spring.jpa.hibernate.ddl-auto=none
```

- Want to initialize your in-memory DB with data?
  - Add a import.sql and data.sql into your resource folder
- Want a more sophistcated database migration tool?
  - Look at Flyway or Liquibase integration

Important notes (2)

- Spring Data is highly customizable. Read the documentation.
- Dynamic finder methods names might get very long
  - Use @Query and JPQL Syntax and provide a nice name
- Use native queries if needed
  - @Query(value = "select * from Employee where firstName=?1" , nativeQuery = true)
- CRUD methods on repository instances are transactional
  - Use @Transactional in repository to override default