Spring JPA Integration

Basic Spring 5.0

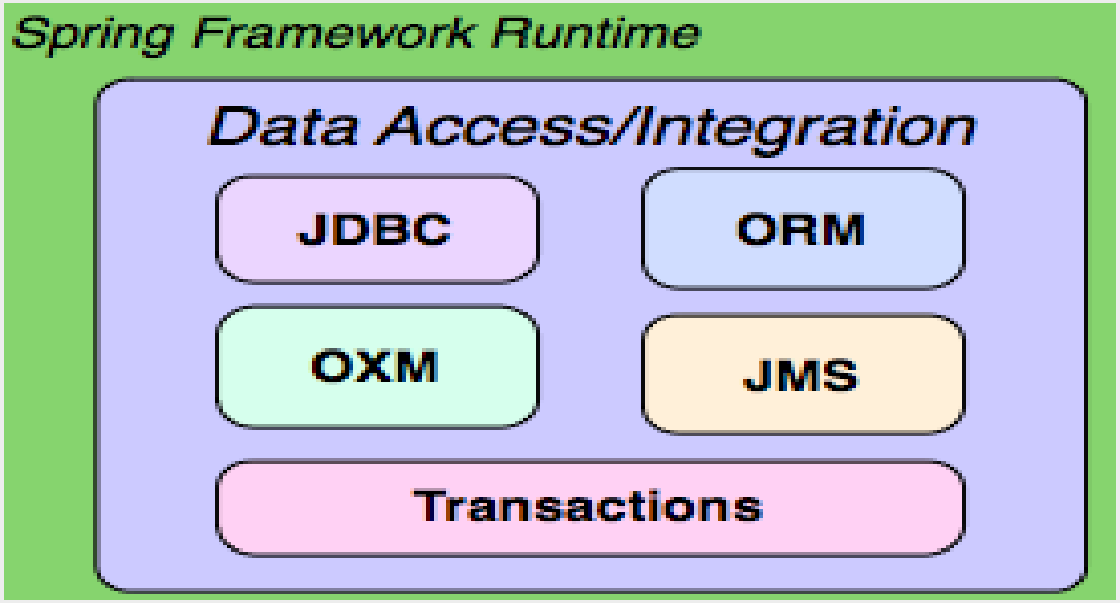
Capgemini

Lesson Objectives

- After completing this lesson, participants will be able to understand:
- Spring support for JPA
- Implementing Spring JPA integration
- Spring Data JPA

4.1 : Spring Support for JPA -Spring JPA Integration : Overview

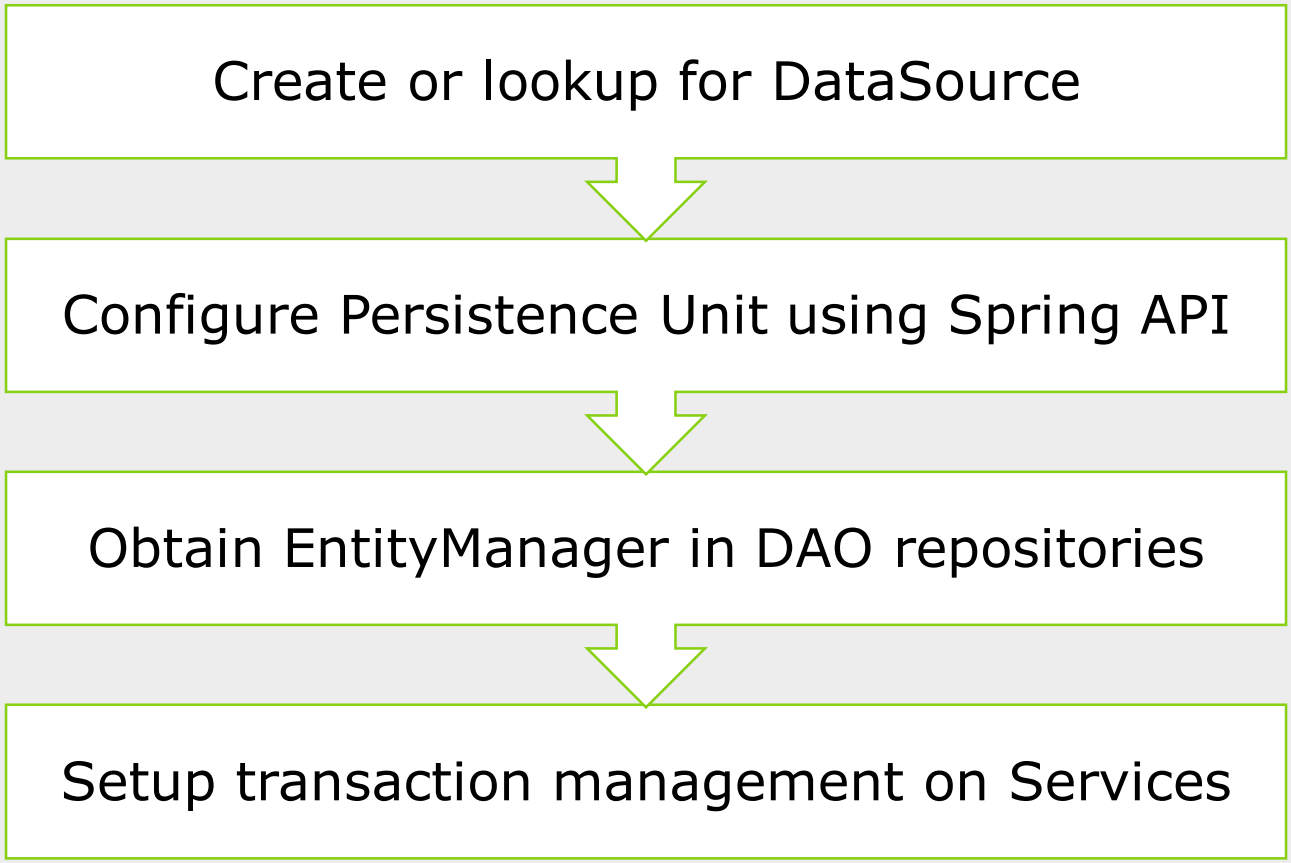
- The Spring Framework supports integration with Hibernate, Java Persistence API (JPA) for
  - resource management,
  - data access object (DAO) implementations, and
  - transaction strategies



4.1 : Spring Support for JPA -Why JPA with Spring?

- Spring benefits to JPA:
- Easy and quick persistence configuration
- Automatic EntityManager management
- Simple testing
- Rich exception hierarchy with common data access exceptions
- Integrated and automated transaction management

4.2 : Implementing Spring JPA Integration -Steps for Spring JPA Integration



4.2 : Spring JPA Integration Steps -Creating or looking up for DataSource

- Two ways to acquire DataSource
- Using simple DriverManagerDataSource

```
<bean name="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
    <property name="username" value="hr"/>
    <property name="password" value="hr"/>
</bean>
```

```
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="java:jdbc/DatabaseName" />
</bean>
```

4.2 : Spring JPA Integration Steps -Spring JPA Configuration

- Used in place of persistence.xml
- Used for Configuration of:
  - EntityManagerFactory
    - JPA Vendor
    - JPA Properties
  - TransactionManager
    - JPA specification defines two kinds of entity managers:
  - Application-managed
    - LocalEntityManagerFactoryBean
  - Container-managed
    - LocalContainerEntityManagerFactoryBean

4.2 : Spring JPA Integration Steps – EntityManagerFactory configuration

- Container-managed EntityManagerFactory
- Configured using LocalContainerEntityManagerFactoryBean
- No need of persistence.xml

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="dataSource" ref="dataSource"/>
      <property name="packagesToScan" value="com.cg.entities"/>
      <property name="persistenceProviderClass" value="org.hibernate.jpa.HibernatePersistenceProvider"/>
      <property name="jpaPropertyMap">
          <map>
              <entry key="hibernate.dialect" value="org.hibernate.dialect.Oracle10gDialect"/>
              <entry key="hibernate.hbm2ddl.auto" value="update"/>
          </map>
      </property>
  </bean>
```

4.2 : Spring JPA Integration Steps – TransactionManager configuration

- Spring managed transaction maangement:
- Configured using JpaTransactionManager

```
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" />
```

4.2 : Spring JPA Integration Steps -Obtaining EntityManager

- JPA provides two annotations to inject EntityManager
- @PersistenceUnit – injects EntityManagerFactory
- @PersistenceContext – injects EntityManager

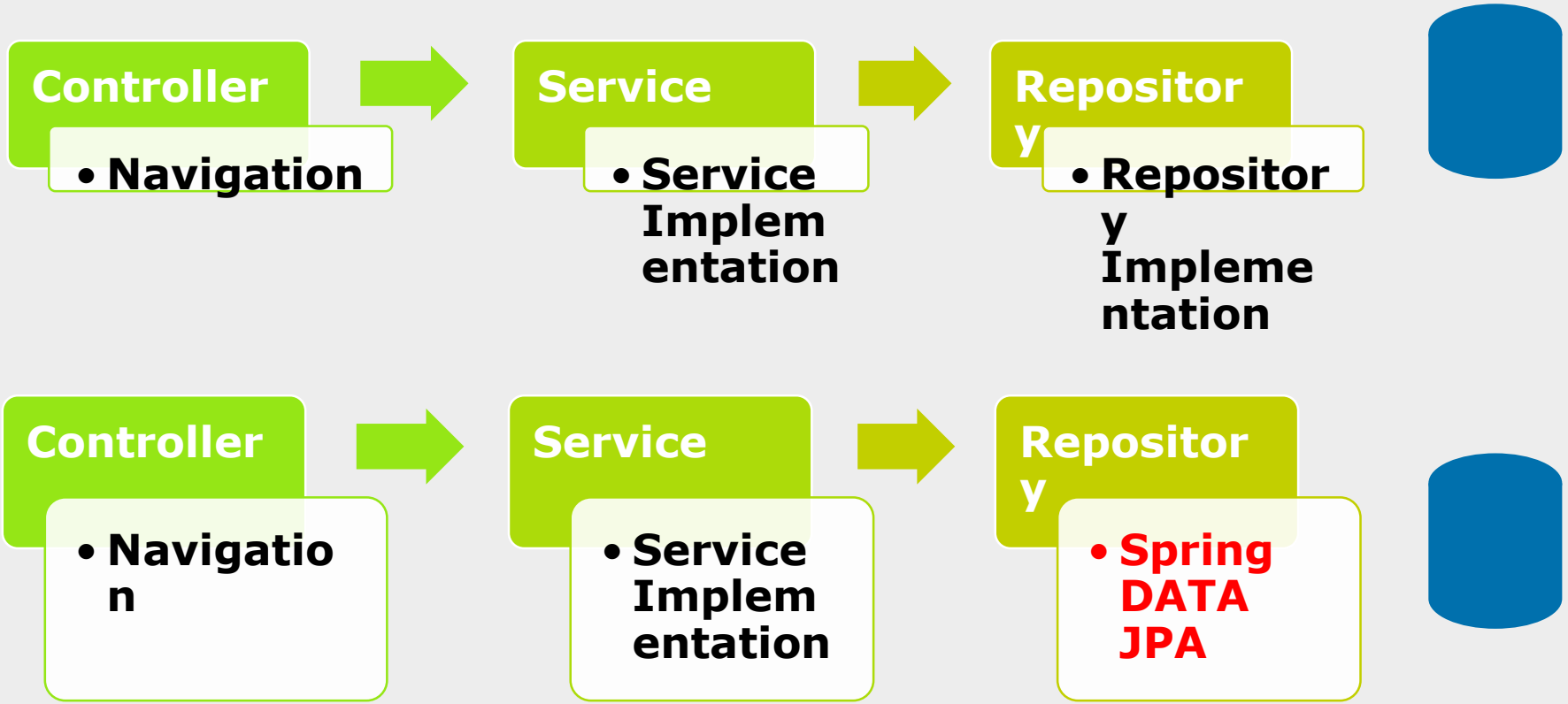
```
@Repository
public class EmployeeRepositoryImpl {

    @PersistenceContext
    private EntityManager entityManager;

    public Employee save(Employee employee) {

        entityManager.persist(employee);
        entityManager.flush();

        return employee;
    }
}
```

4.2 : Spring JPA Integration -Demo

- JPASpringMVC

4.3 : Spring Data JPA - Spring Data JPA

- Traditional Spring JPA Integration
- Spring Data JPA: No need to write repository code



4.3 : Spring DATA JPA -Working with Spring Data JPA

- To use repository from Spring Data, create an interface to extend following interface, which specifies no methods to implement:
- org.springframework.data.repository.Repository<T, ID extends Serializable>
- The generic type parameters:
- T : Type of domain entity class
- ID: ID type of the domain entity class

```
@Entity
public class Student implements Serializable {
    @Id
    private Long studentId;
    private String name;
    //getters and setters
}
```

```
public interface StudentRepository extends
        JpaRepository<Student,Long> {

}
```

4.3 : Spring Data JPA -Demo

- JPASpringDataMVC

Lab

- Lab 2

Summary

- In this lesson, you have learnt:
- Spring support for JPA
- How to integrate Spring and JPA
- How to work with Spring Data repositories

Review Question

- Question 1 Which one of the following is valid type of entity manager in JPA?
  - Option 1: Container managed
  - Option 2: JVM Managed
  - Option 3: Server Managed
- Question 2 LocalEntityManagerFactoryBean doesn't allow you to use a Spring managed DataSource instance.
  - True/False