**EXPLORING JAKARTA CDI WITH SPRING FRAMEWORK**

* Jakarta Contexts and Dependency Injection (CDI) is a powerful set of APIs that provide a standard way for Java applications to manage dependencies and context. It is an important part of the Jakarta EE platform and provides a flexible and extensible way to manage the lifecycle of Java objects.
* CDI provides a unified way to manage object dependencies and lifecycle management and allows developers to create more modular and maintainable code.
* CDI is a specification (interface). Spring Framework implements CDI.
* Important Inject API Annotations:
  + Inject (~Autowired in Spring)
  + Named (~Component in Spring)
  + Qualifier
  + Scope
  + Singleton

To make use of CDI we should add dependency in pom.xml.

package com.naveen.learnspringframework.JakartaCDIExample;

import java.util.Arrays;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import jakarta.inject.Inject;

import jakarta.inject.Named;

//instead of @Component, there is an alternative that is @named.

@Named

class BusinessService{

    private DataService dataService;

    //instead of @autowired, there is an alternative that is @inject.

    @Inject

    public void setDataService(DataService dataService) {

        this.dataService = dataService;

        System.out.println("Data service dependency injected");

    }

    public DataService getDataService() {

        return dataService;

    }

}

@Named

class DataService{

}

@Configuration

@ComponentScan

public class JakartaCDIExampleApp {

    public static void main(String[] args) {

        try(var context =

                new AnnotationConfigApplicationContext

                    (JakartaCDIExampleApp.class)){

            Arrays.stream(context.getBeanDefinitionNames())

                .forEach(System.out::println);

            System.out.println(context.getBean(BusinessService.class));

        }

    }

}

OUTPUT:

