**MINISTERUL EDUСAŢIEI ŞI СERСETĂRII AL REPUBLIСII MOLDOVA**

**UNIVERSITATEA DE STAT „ALEСU RUSSO” DIN BĂLŢI**

**FAСULTATEA DE ŞTIINŢE REALE, EСONOMIСE ȘI ALE MEDIULUI**

**СATEDRA DE MATEMATIСĂ ȘI INFORMATIСĂ**

**LUCRARE DE LABORATOR NR. 10**

**„Spring Core”**

**Curs „Platforma JAVA Enterprise II”**

**Coordonator:**

**Olga CERBU**

dr.,conf.univ

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**Autor:**

Studentul grupei AW21M

**Dumitru BAZAOСHI**

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**BĂLȚI, 2023**

**CUPRINS**

[Scop. Condiția problemei 2](#_Toc149377582)

[Ce este Spring? 2](#_Toc149377583)

[Caracteristicile Spring 2](#_Toc149377584)

[Crearea programului 3](#_Toc149377585)

[Concluzii 7](#_Toc149377586)

[Referințe bibliografice 8](#_Toc149377587)

Scop. Condiția problemei

Scopul acestei lucrări de laborator legată de tehnologia Spring, mai precis Spring Core, este de a oferi o înțelegere solidă a conceptelor și funcționalităților de bază ale acestui framework de dezvoltare pentru Java. Această înțelegere este esențială pentru dezvoltarea de aplicații Java robuste, scalabile și ușor de întreținut. Prin acest laborator, se urmărește atingerea următoarelor obiective, familiarizarea cu Spring, configurarea și gestionarea bean-urilor, injectarea dependințelor.

Ce este Spring?

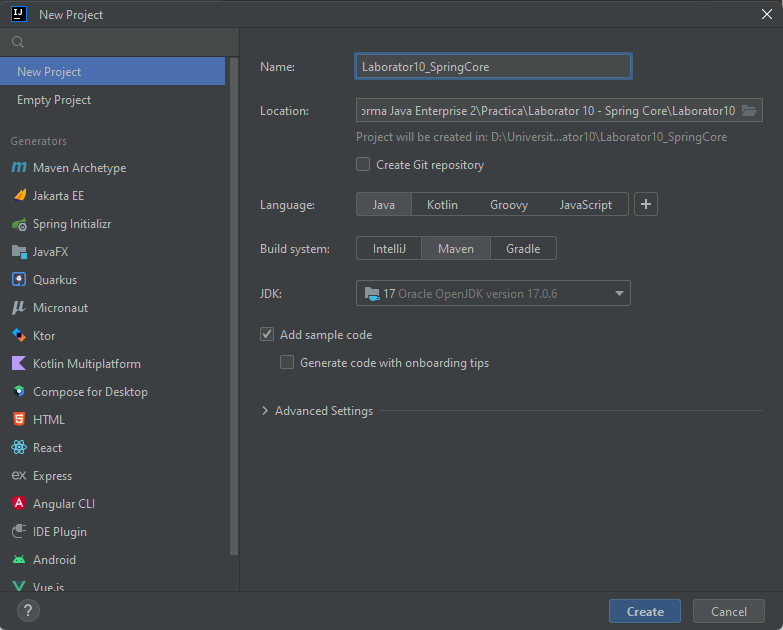
Spring este un cadru (framework) de dezvoltare pentru limbajul de programare Java, care oferă o serie de facilități și servicii esențiale pentru dezvoltarea de aplicații Java robuste și ușor de întreținut. A fost creat pentru a simplifica dezvoltarea aplicațiilor Java enterprise, eliminând multe din complexitățile și cerințele tehnice ale dezvoltării clasice de aplicații Java.

Caracteristicile Spring

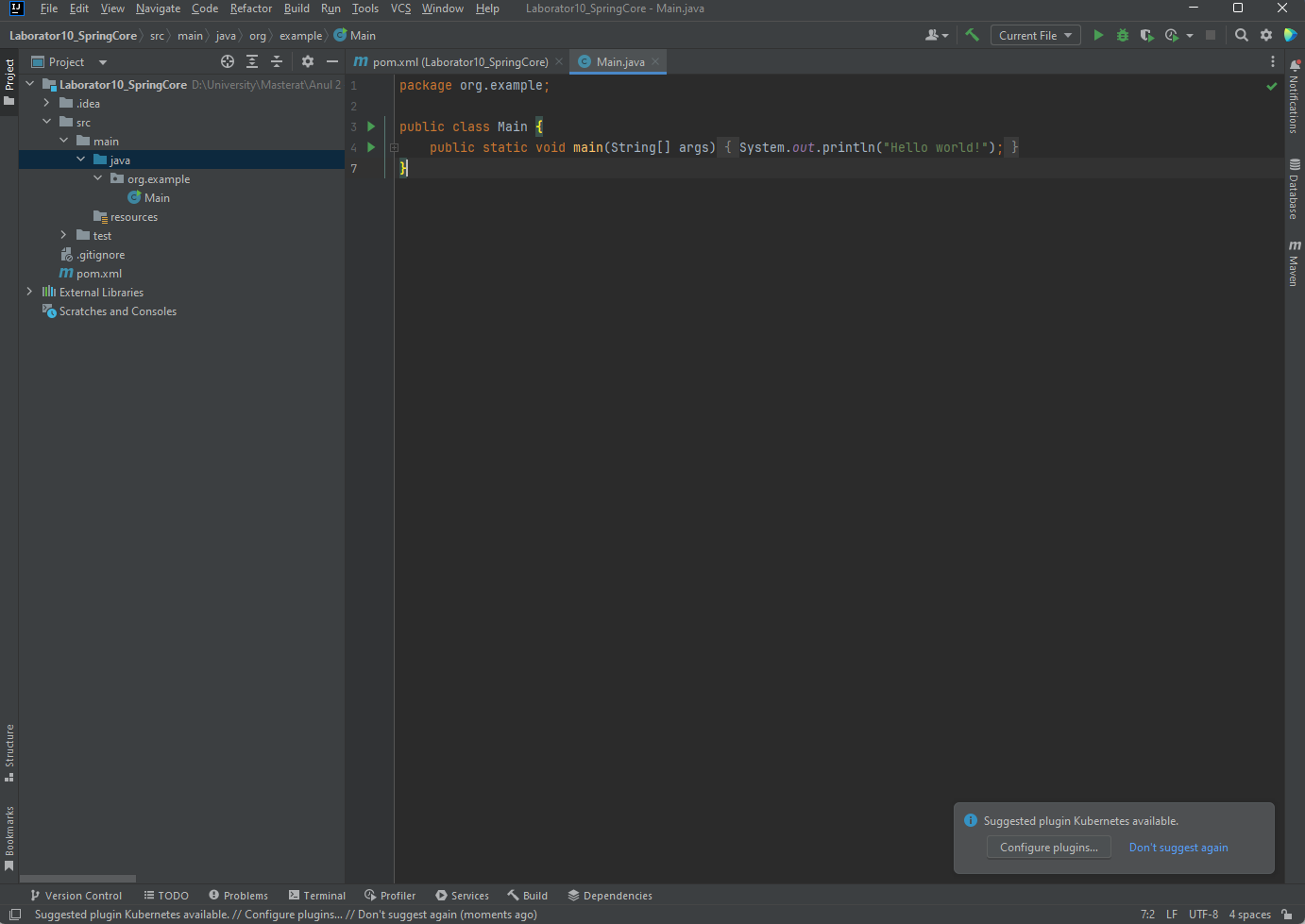
* **Inversiunea de control** (IoC): Spring oferă un container IoC care gestionează crearea și administrarea obiectelor (beans) în aplicație. Acesta inversază controlul asupra ciclurilor de viață ale obiectelor, ceea ce face ca dezvoltatorii să nu mai fie responsabili pentru crearea și gestionarea manuală a dependințelor.
* **AOP** (Programarea orientată pe aspect): Spring integrează programarea orientată pe aspect, permițând dezvoltatorilor să împartă logică comună în module reutilizabile, cum ar fi aspecte, și să o aplique pe mai multe componente.
* **Dependență la injecție**: Spring oferă un mecanism puternic de injectare a dependențelor, facilitând dezvoltarea de clase care depind de alte clase sau servicii. Aceasta îmbunătățește flexibilitatea și ușurează testarea unitară.
* **Tranziții și gestionarea resurselor**: Spring asigură gestionarea eficientă a resurselor, cum ar fi tranzacțiile. Permite dezvoltatorilor să definească tranzacții declarativ, eliminând necesitatea codului repetitiv pentru gestionarea tranzacțiilor.
* **Gestionarea ciclurilor de viață**: Spring oferă metode pentru gestionarea ciclurilor de viață ale bean-urilor, cum ar fi inițializarea și distrugerea, permițând dezvoltatorilor să intervină în aceste cicluri.
* **Aspecte securitate și autentificare**: Spring Security furnizează un set puternic de funcționalități pentru gestionarea securității și autentificării în aplicațiile Java.
* **Suport pentru ORM** (Mapping obiect-relațional): Spring integrează baze de date și framework-uri ORM precum Hibernate, JPA, JdbcTemplate, oferind un nivel înalt de abstracție pentru lucrul cu datele.
* **Suport pentru diverse module**: Spring include o gamă variată de module și extensii pentru a răspunde la nevoile diferite ale dezvoltatorilor, cum ar fi Spring Boot pentru dezvoltarea rapidă a aplicațiilor sau Spring Cloud pentru dezvoltarea de microservicii.

Crearea programului

1. Trecem in mediul de programare (Intellij Idea)
2. Cream un proiect nou, Schimbăm numele, locația, limbajul ramâne **Java**, sistemul de build selectăm **Maven**. Click pe butonul **Create**.



La noi s-a deschis proiectul implicit creat.



1. Pentru a lucra cu Spring trebuie să adăugăm dependințele necesare pentru aceasta. Trecem în pom.xml, unde adaugăm următoarele dependințe:

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.0.13</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.0.13</version>

</dependency>

1. Facem build la proiect pentru a inițializa biblioteca adăugată. Build -> Build Project
2. Adăugam codul în clasa **Main**

package org.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext beanFactory = new ClassPathXmlApplicationContext("beans.xml");

Hello myBean = (Hello) beanFactory.getBean("nume1");

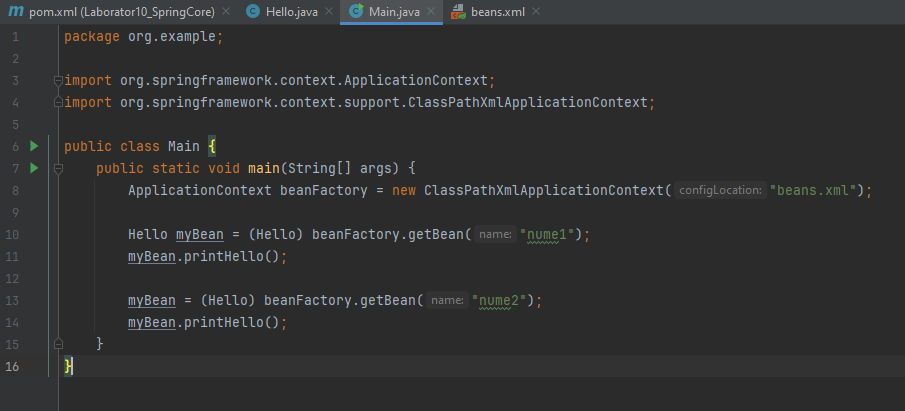
myBean.printHello();

myBean = (Hello) beanFactory.getBean("nume2");

myBean.printHello();

}

}



1. Adăugam codul în clasa **Hello**

package org.example;

public class Hello {

String nume;

public void printHello() {

System.out.println("Salut, " + nume + "!");

}

public String getNume() {

return nume;

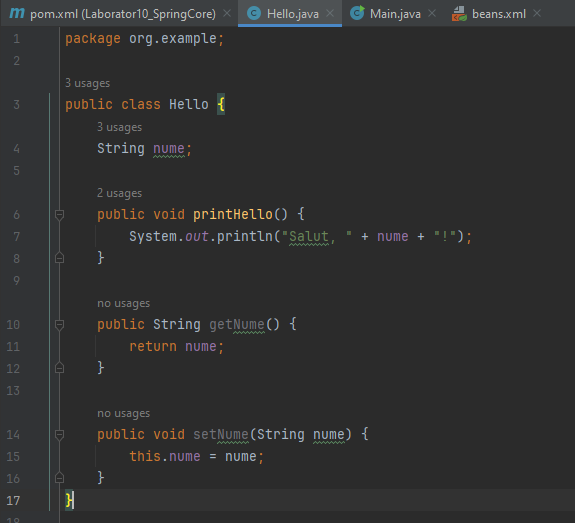
}

public void setNume(String nume) {

this.nume = nume;

}

}



1. Adăugam codul în fișierul **beans.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="nume1" class="org.example.Hello">

<property name="nume" value="Dumitru" />

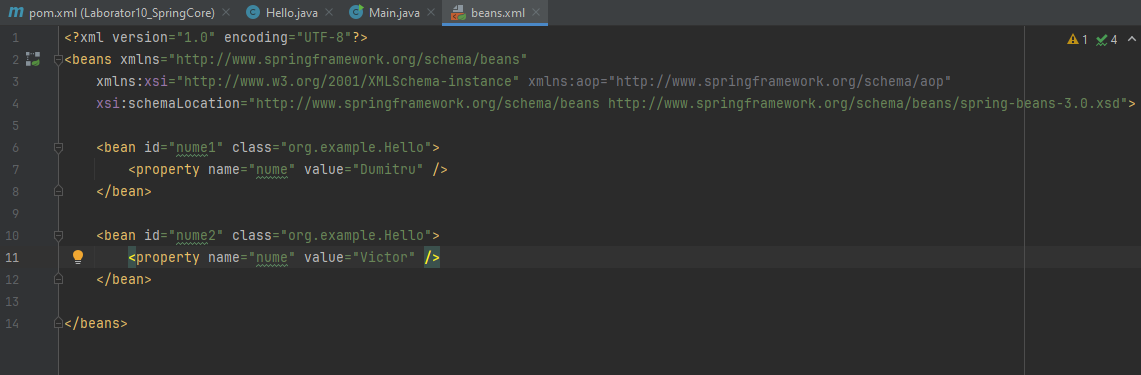
</bean>

<bean id="nume2" class="org.example.Hello">

<property name="nume" value="Victor" />

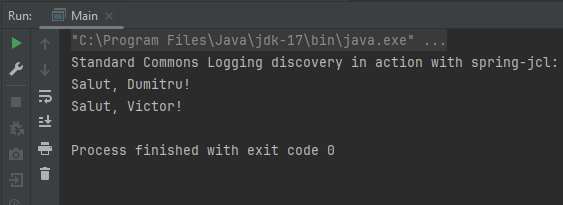
</bean>

</beans>



1. Rulăm proiectul.

Observăm rezultatele în consolă



Concluzii

În concluzie, acest laborator dedicat tehnologiei Spring Core ne-a furnizat o bază solidă de cunoștințe și competențe în ceea ce privește dezvoltarea de aplicații Java bazate pe acest cadru (framework) de dezvoltare. Am învățat că Spring Core oferă o serie de funcționalități puternice care pot simplifica dezvoltarea și întreținerea aplicațiilor Java enterprise.

Am explorat concepte cheie precum inversiunea de control (IoC) și gestionarea dependențelor prin injectare, care ne-au permis să creăm aplicații cu un cuplaj redus între componentele lor și să le facem mai ușor de testat și de întreținut. Am înțeles cum să definim și să configurăm beans Spring, precum și cum să gestionăm ciclurile de viață ale acestora.

Referințe bibliografice

1. *Spring Framework* [online] [vizitat 27.10.2023]. Disponibil: <https://spring.io/projects/spring-framework>
2. *Core Technologies* [online] [vizitat 27.10.2023]. Disponibil: <https://docs.spring.io/spring-framework/reference/core.html>
3. *Introduction to Spring Framework* [online] [vizitat 28.10.2023]. Disponibil: <https://www.geeksforgeeks.org/introduction-to-jdbc/>