



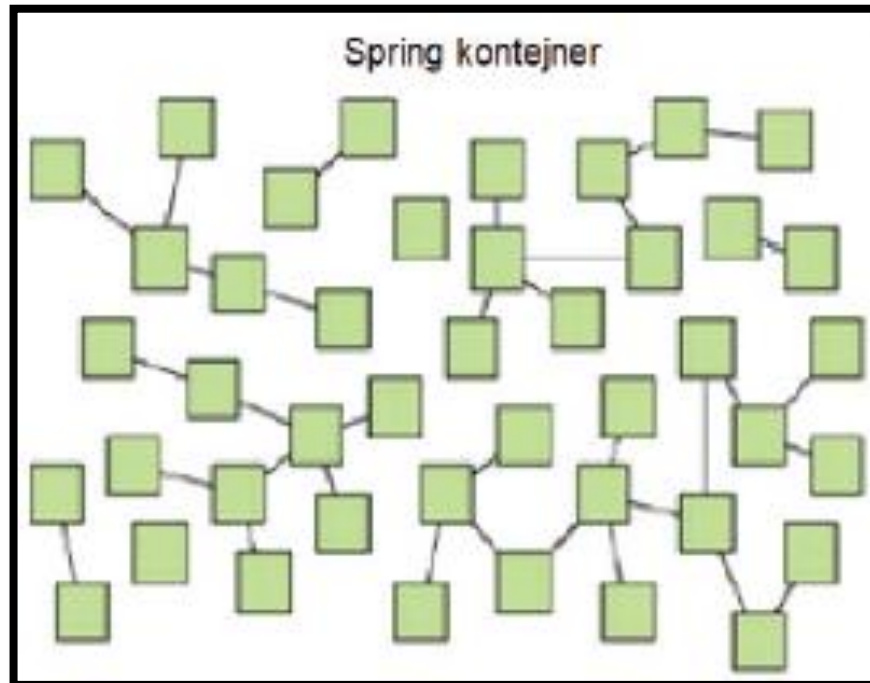
# Konfiguracija zrna u Spring okruženju

# Današnje teme

- Teme koje pokriva ova prezentacija jesu:
  - Konfiguracija zrna u Spring okruženju,
  - Zrna i njihov životni ciklus,
  - JavaSE problem,
  - Deklaracija zrna,
  - Osnovna konfiguracija zrna,
  - ...

# Konfiguracija zrna u Spring okruženju

- Objekti aplikacije žive unutar Spring kontejnera
- Kontejner je zadužen za kreiranje objekata
- Kontejner predstavlja jezgro Spring okruženja



# Konfiguracija zrna u Spring okruženju

- Kontejneri se označavaju terminom IoC (*Inversion of Control*) kontejneri
- Tipovi kontejnera:
  - **Fabrike zrna** - najjednostavniji kontejneri koji pružaju osnovnu podršku za DI i definisani su interfejsom `org.springframework.beans.factory.BeanFactory`
  - **Konteksti aplikacije** – nadograđuju fabrike zrna, definisani su interfejsom `org.springframework.context.ApplicationContext`. Znatno su kompleksniji u odnosu na fabrike zrna

# Konfiguracija zrna u Spring okruženju

- Spring nudi nekoliko različitih oblika konteksta aplikacije:
  - **AnnotationConfigApplicationContext** – učitava kontekst aplikacije iz jedne ili više Java konfiguracionih klasa
  - **AnnotationConfigWebApplicationContext** – učitava kontekst veb aplikacije iz jedne ili više Java konfiguracionih klasa
  - **ClassPathXmlApplicationContext** – učitava kontekst iz jednog ili više XML fajlova koji se nalaze na klasnoj putanji projekta
  - **FileSystemXmlApplicationContext** – učitava kontekst iz jednog ili više XML fajlova koji se nalaze u sistemu fajlova
  - **XmlWebApplicationContext** – učitava kontekst iz jednoig ili više XML fajlova koje sadrži veb aplikacija

# Konfiguracija zrna u Spring okruženju

- Učitavanje konteksta iz sistema fajlova, prosleđivanje putanje:

```
ApplicationContext context = new  
FileSystemXmlApplicationContext("c:/vitez.xml");
```

- Ukoliko se koristi kontekst aplikacije:

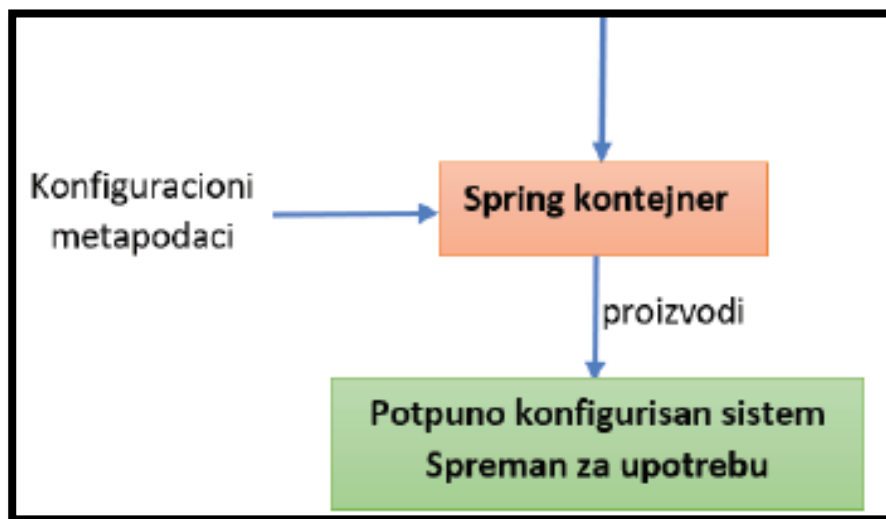
```
ApplicationContext context = new  
ClassPathXmlApplicationContext("vitez.xml");
```

- Pozivanje metode `getBean()`:

```
ClassPathXmlApplicationContext context =  
    new ClassPathXmlApplicationContext(  
        "vitez.xml");  
Vitez vitez = context.getBean(Vitez.class);
```

# Zrna i njihov životni ciklus

- **Zrna** – objekti koji čine aplikaciju i kojima upravlja Spring IoC kontejner. Objekat koga instancira u kojim upravlja kontejner.
- Objekti deklarišu svoje zavisnosti na jedan od sledećih načina:
  - Kroz argumente **konstruktora**
  - Kroz argumente **factory**
  - Kao **property** kojem se instanca objekta setuje nakon konstrukcije



# Zrna i njihov životni ciklus

- Koristimo ključnu reč **new** za instanciranje objekta – zrna
- Kada se ukine poslednja referenca na objekat na **hipu** (mesto u memoriji gde se smeštaju objekti) zrno postaje označeno za sakupljanje đubreta i nakon nekog vremena se uklanja

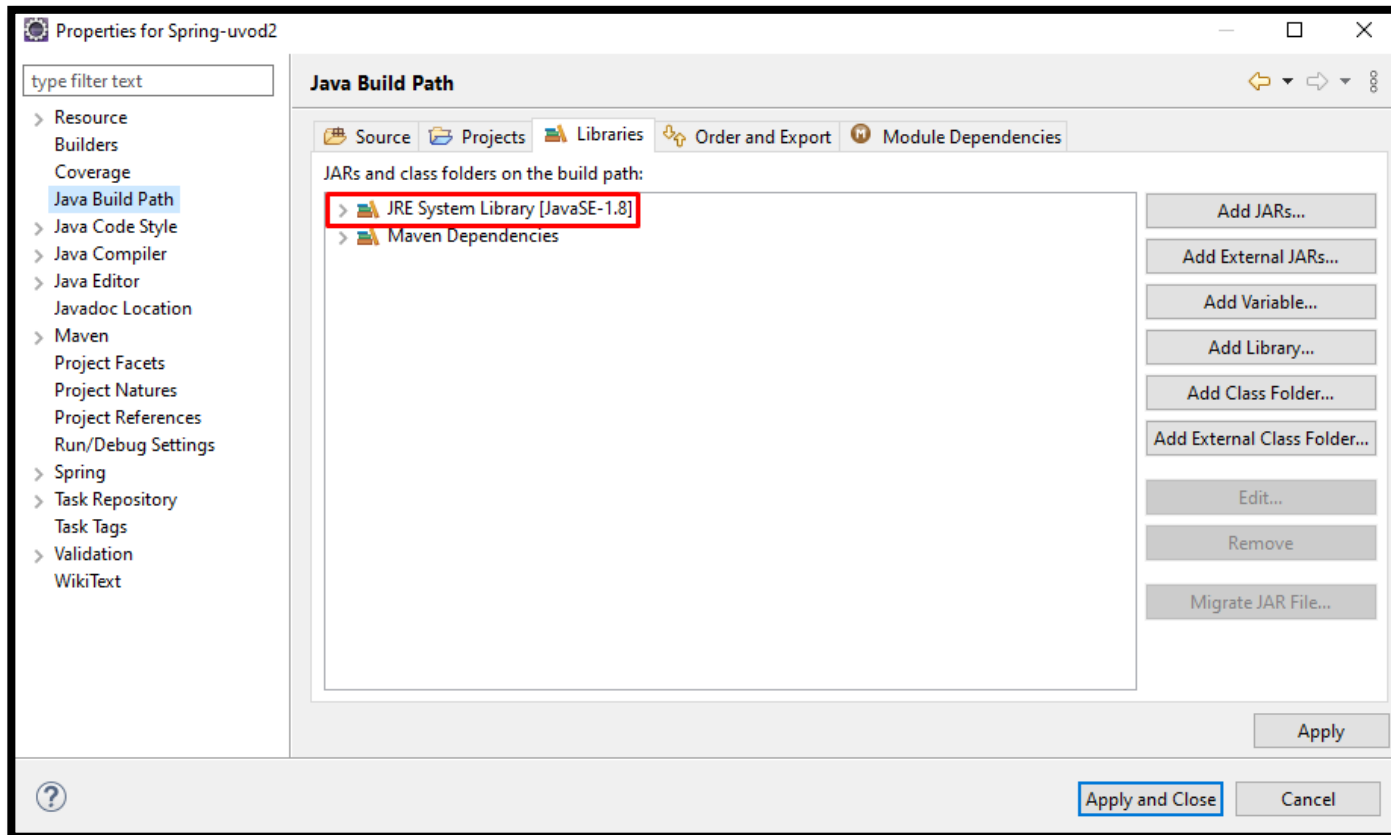


# Deklaracija zrna

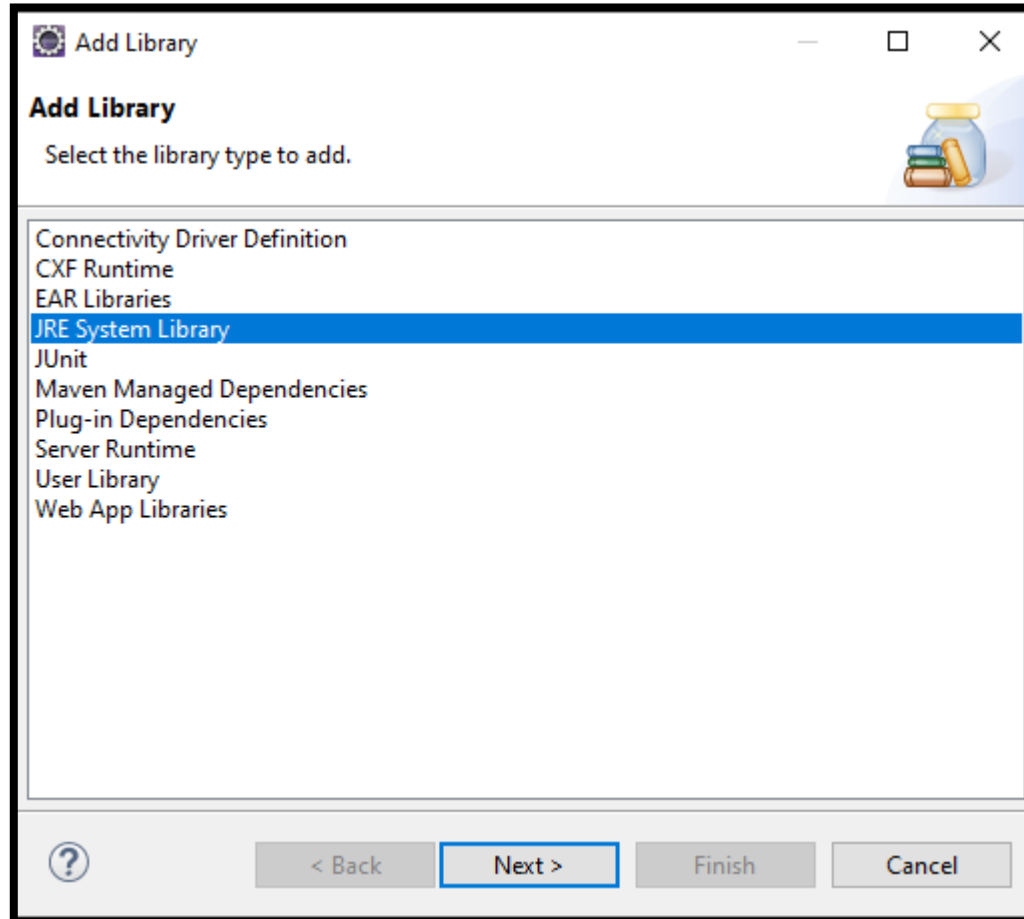
- Od objekata se ne očekuje da traže ili kreiraju druge objekte koji su im potrebni kako bi obavljali svoju funkciju, oni im daju reference na objekte sa kojima saraduju unutar kontejnera
- **Ožičavanje(wiring)** – kreiranje veza između objekata aplikacije, predstavlja srž injekcije zavisnosti
- Spring je baziran na kontejneru – ukoliko se ne konfiguriše ispravno, to je samo prazan kontejner koji ne služi svrsi
- Unutar **<beans>** elementa se smešta kompletna Spring konfiguracija

# JavaSE problem

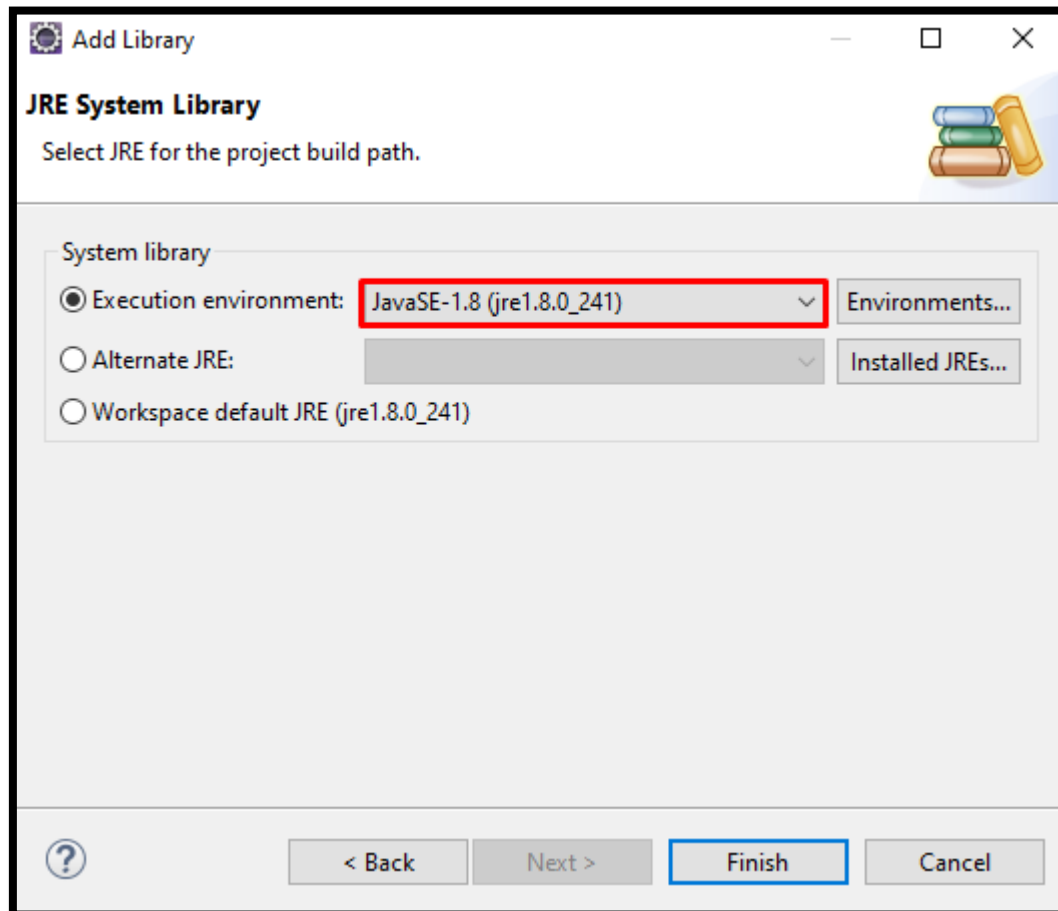
- Ukoliko budete imali problema sa JavomSE potrebno je da ispratite sledeće korake



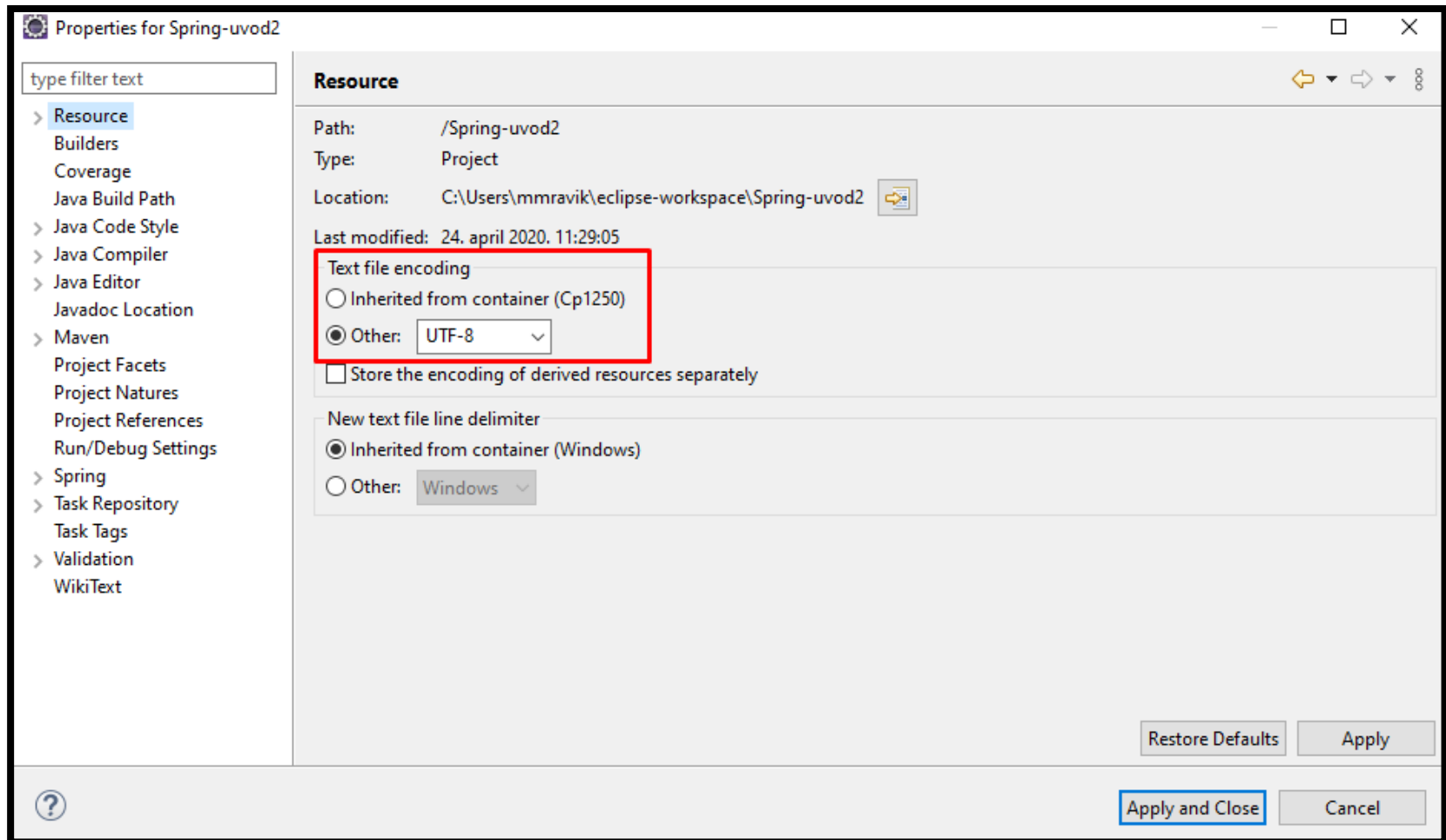
# JavaSE problem



# JavaSE problem



# JavaSE problem



# Osnovna konfiguracija zrna



## Spring Tools 3 Add-On for Spring Tools 4 3.9.13.CI

Spring Tools 3 Add-On for Spring Tools 4 Attention: This add-on pack provides additional components from the previous Spring Tools 3 generation to be installed... [more info](#)

by [Pivotal](#), EPL

[J2EE spring](#) [Spring IDE](#) [Cloud jee ...](#)

★ 227

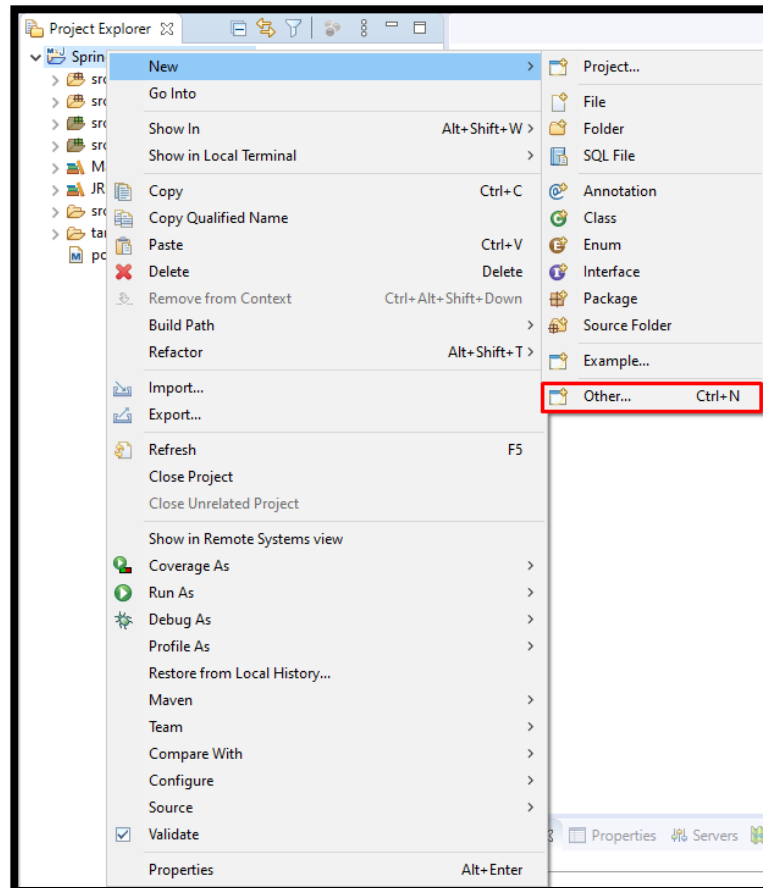


Installs: **303K** (10,170 last month)

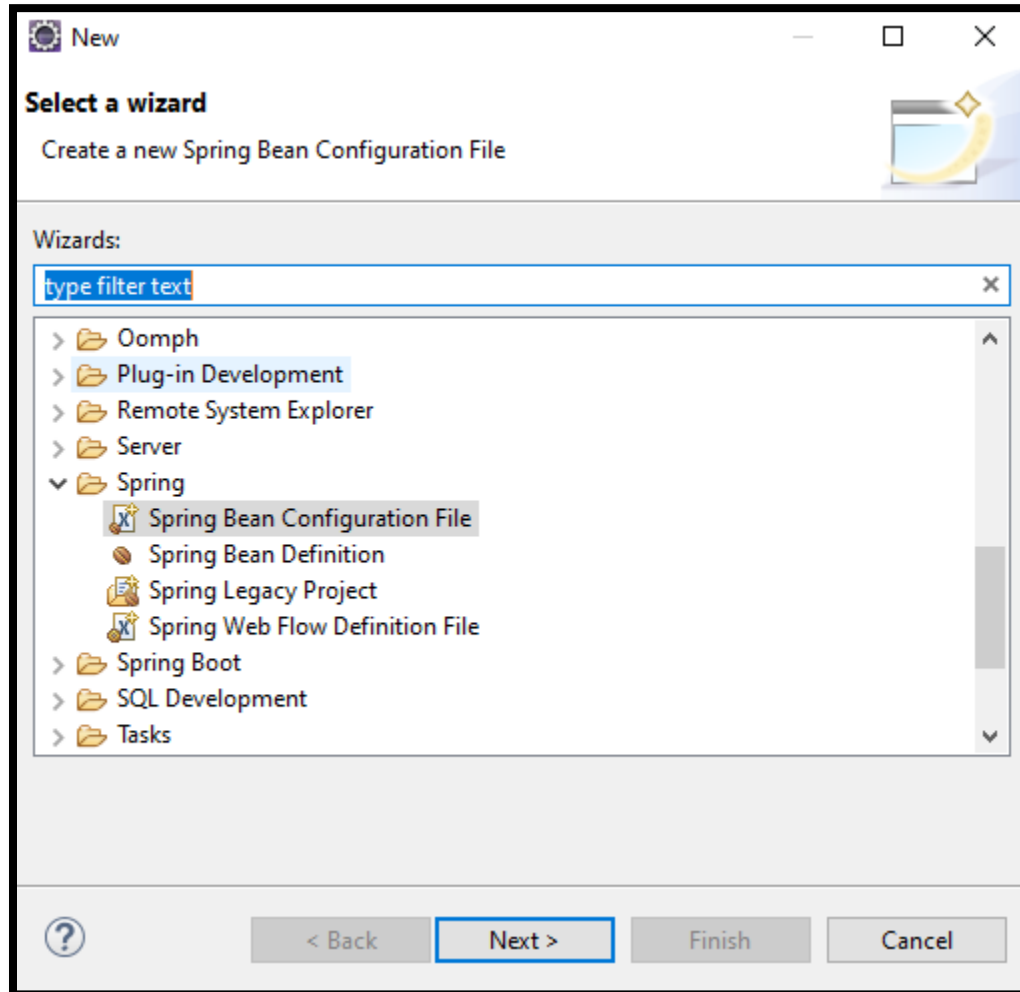
Installed

# Osnovna konfiguracija zrna

- Prvenstveno je potrebno dodati konfiguracioni faj zrna

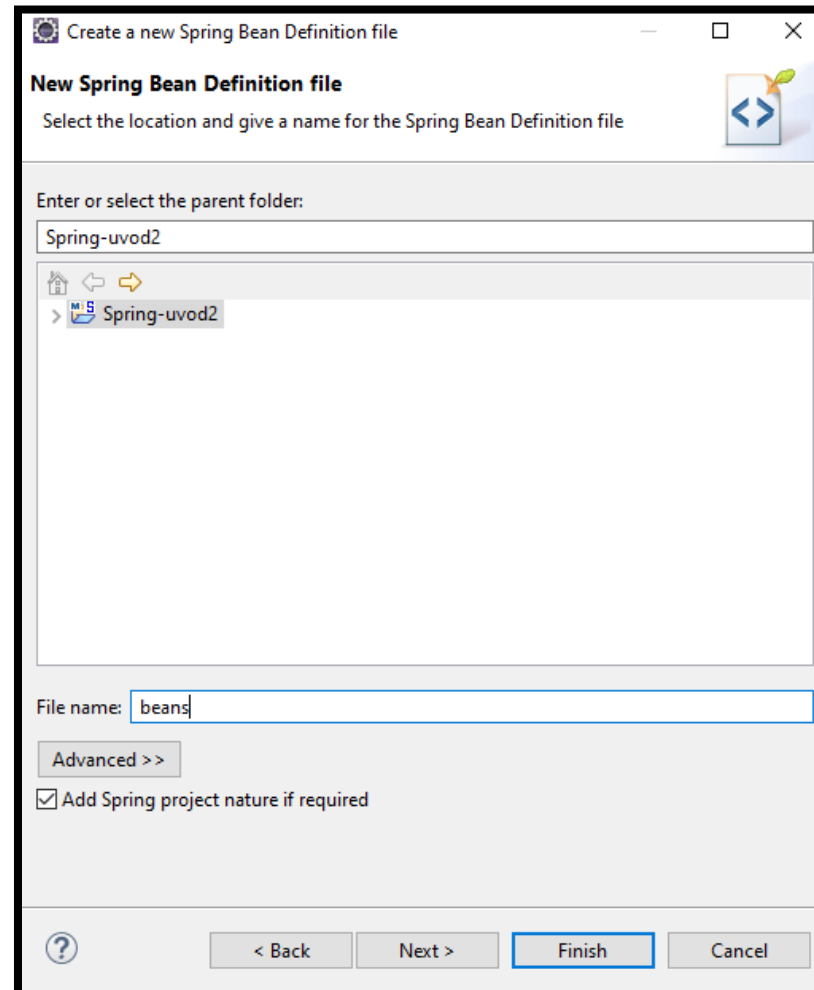


# Osnovna konfiguracija zrna

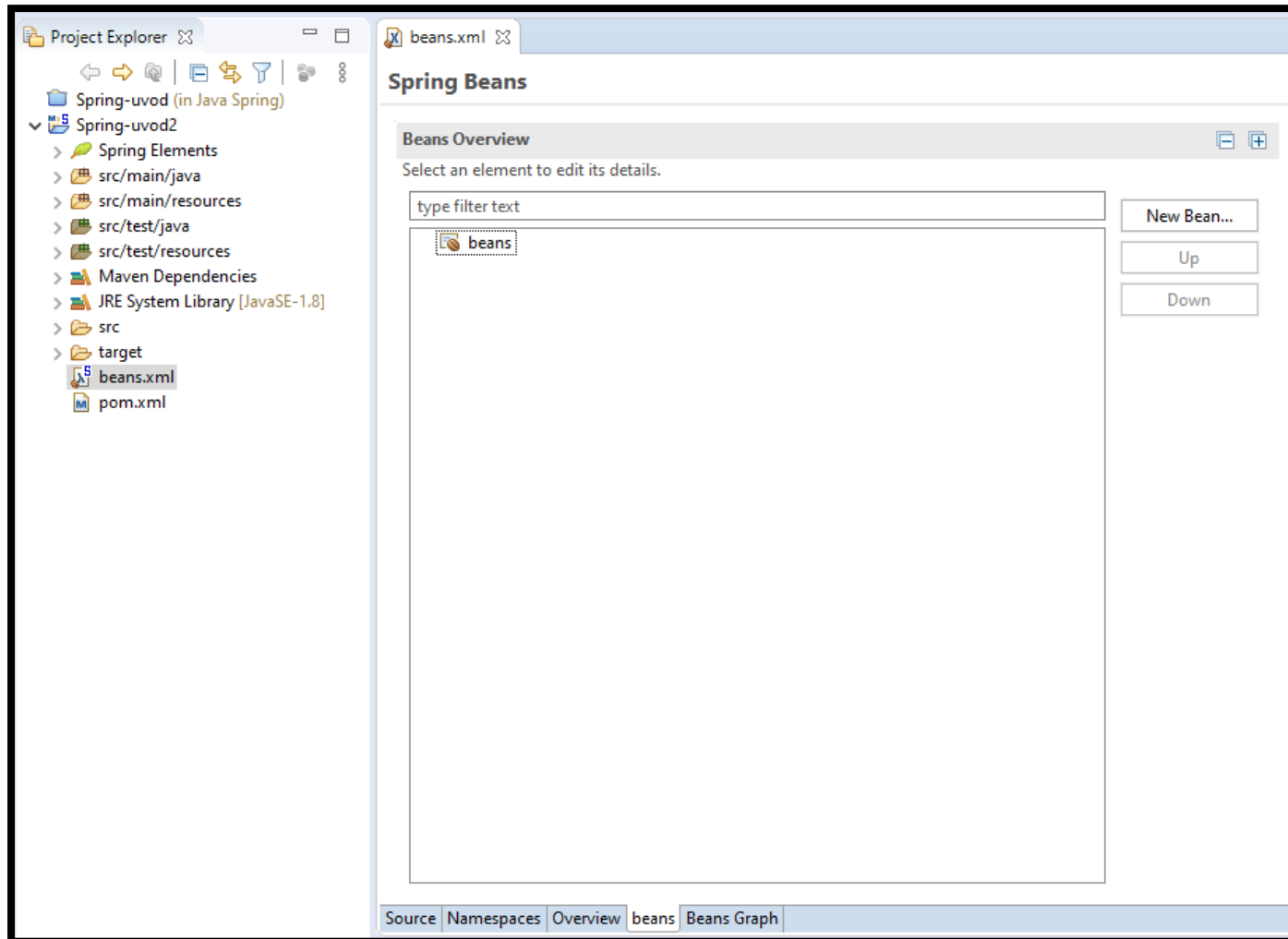




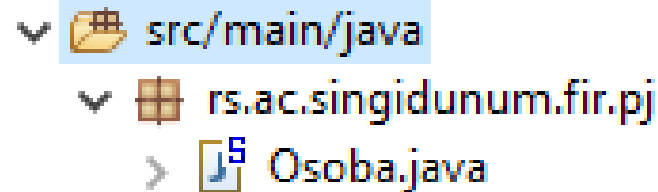
# Osnovna konfiguracija zrna



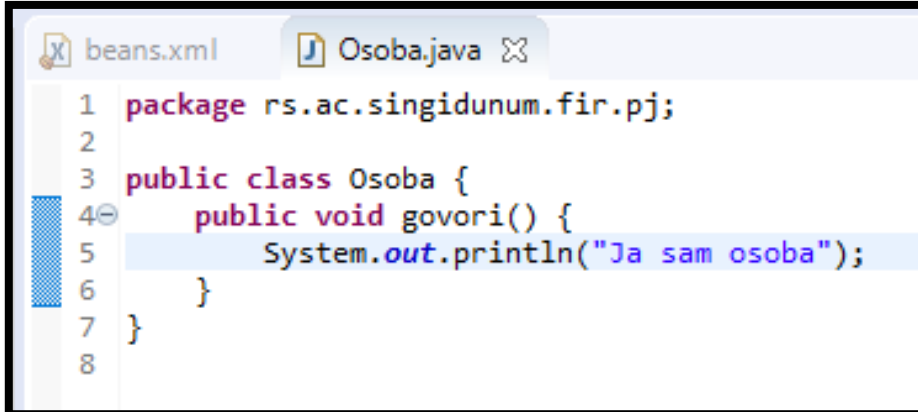
# Osnovna konfiguracija zrna



# Osnovna konfiguracija zrna

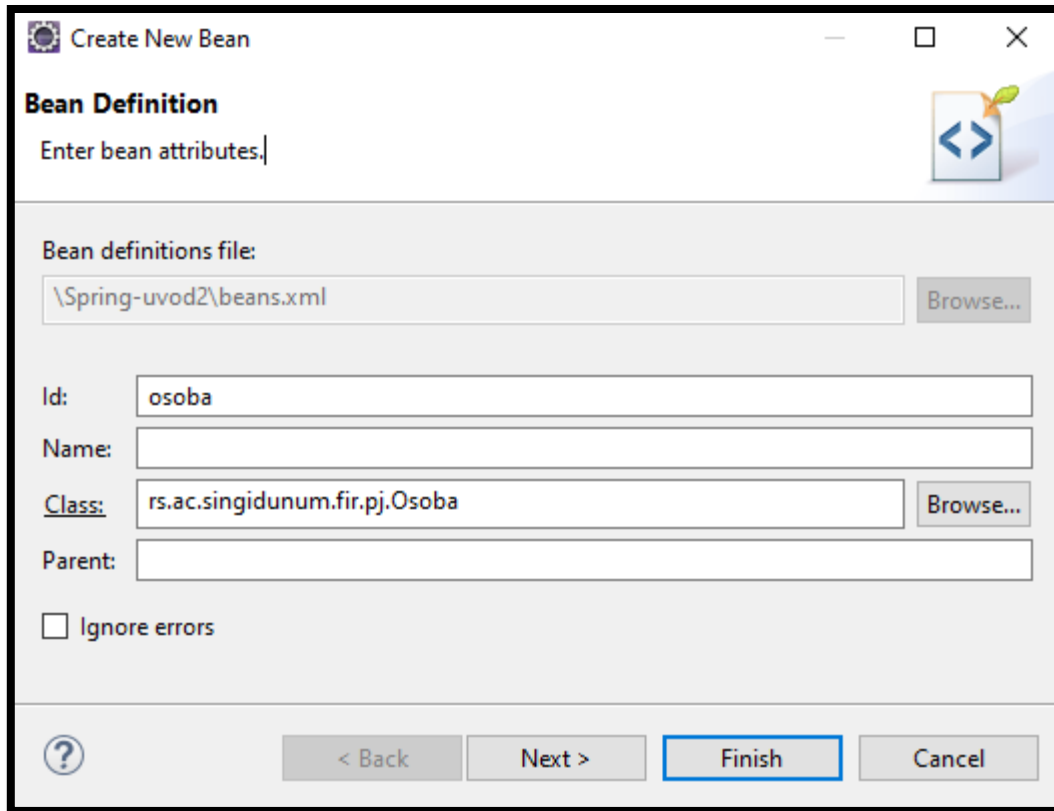


```
▼ src/main/java
  ▼ rs.ac.singidunum.fir.pj
    > Osoba.java
```



```
beans.xml  Osoba.java ✕
1 package rs.ac.singidunum.fir.pj;
2
3 public class Osoba {
4     public void govori() {
5         System.out.println("Ja sam osoba");
6     }
7 }
8
```

# Osnovna konfiguracija zrna



**Create New Bean**

**Bean Definition**  
Enter bean attributes.

Bean definitions file:  
\\Spring-uvod2\\beans.xml

Id:

Name:

Class:

Parent:

Ignore errors

# Osnovna konfiguracija zrna

The screenshot displays the Spring Beans configuration tool. The top window title is "beans.xml" and "Osoba.java". The main area is titled "Spring Beans" and is divided into two main sections: "Beans Overview" and "Element Details".

**Beans Overview:** This section contains a search box labeled "type filter text" and a tree view under the heading "beans". A single entry, "osoba [rs.ac.singidunum.fir.pj.Osoba]", is selected. To the right of the tree are three buttons: "New Bean...", "Up", and "Down".

**Element Details:** This section is titled "Set the properties of the selected element. Required fields are denoted by \*\*\*". It contains a list of properties for the selected bean:

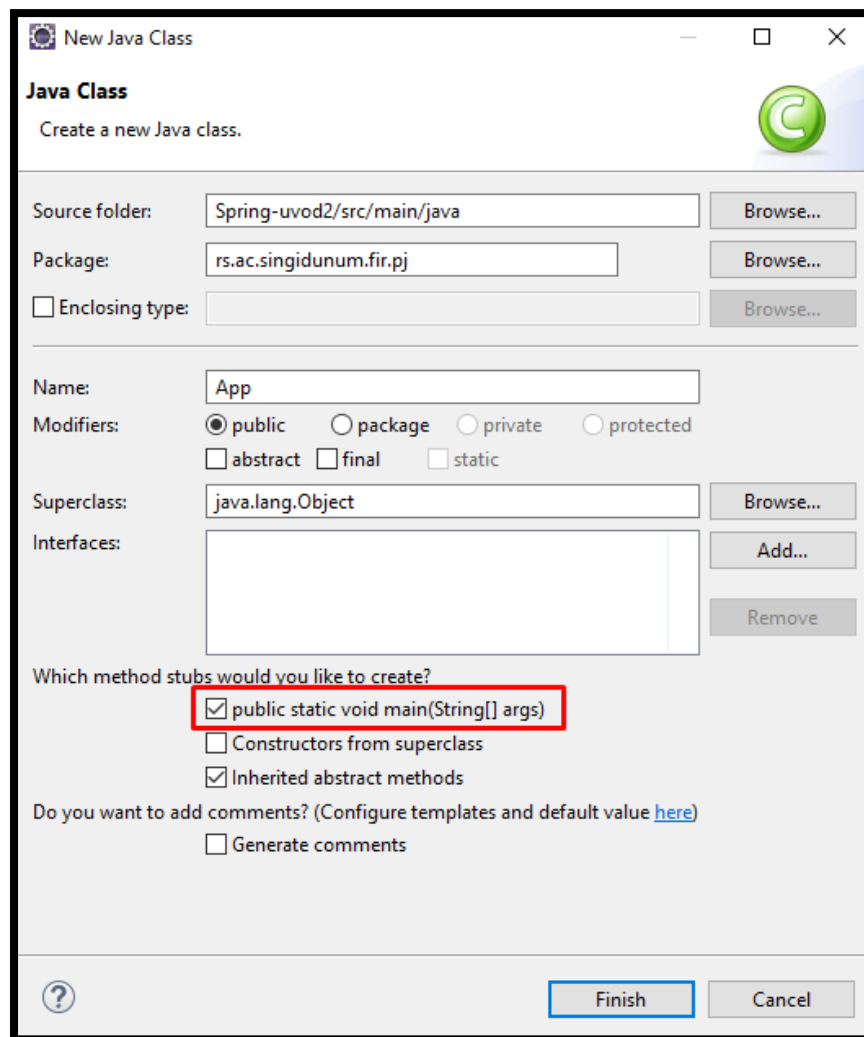
- id:** osoba
- abstract:** (dropdown menu)
- autowire:** (dropdown menu)
- autowire-candidate:** (dropdown menu)
- class:** rs.ac.singidunum.fir.pj.Osoba (with a "Browse..." button)
- depends-on:** (text input)
- destroy-method:** (text input)
- factory-bean:** (text input)
- factory-method:** (text input)
- init-method:** (text input)
- lazy-init:** (dropdown menu)
- name:** (text input)
- parent:** (text input)
- primary:** (dropdown menu)
- scope:** (dropdown menu)

**Documentation:** This section provides a brief description: "Element : bean. Defines a single (usually named) bean. A bean definition may contain nested tags for constructor arguments, property values, lookup methods, and replaced methods. Mixing constructor injection and setter injection on the same bean is explicitly supported."

# Osnovna konfiguracija zrna

```
*beans.xml  Osoba.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
5
6
7     <bean id="osoba" class="rs.ac.singidunum.fir.pj.Osoba"></bean>
8 </beans>
```

# Osnovna konfiguracija zrna



**New Java Class**

Java Class

Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers:  public  package  private  protected  
 abstract  final  static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

# Osnovna konfiguracija zrna

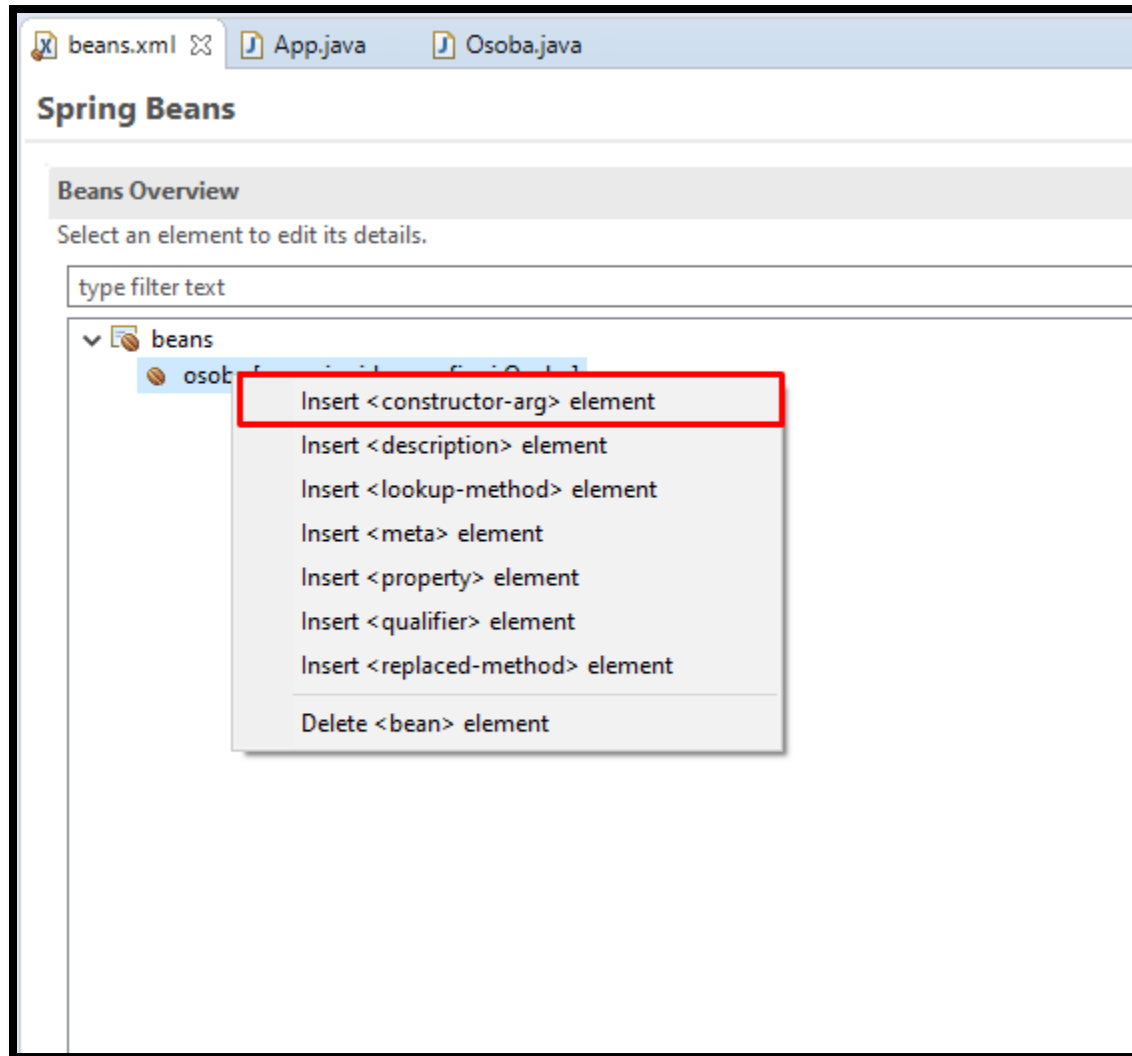
```
beans.xml  Osoba.java  App.java ✕
1 package rs.ac.singidunum.fir.pj;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.FileSystemXmlApplicationContext;
5
6 public class App {
7
8     public static void main(String[] args) {
9         // Standardni Java nacin
10        // Osoba osoba = new Osoba();
11        // osoba.govori();
12
13        //Spring nacin
14        //ApplicationContext - kontejner, klasa koja instancira za nas
15        ApplicationContext context = new FileSystemXmlApplicationContext("C:\\Users\\mmravik\\eclipse-workspace\\Spring-uvod2\\beans.xml");
16        Osoba osoba = (Osoba) context.getBean("osoba");
17        osoba.govori();
18        //Na kraju je potrebno zatvoriti kontekst aplikacije
19        ((FileSystemXmlApplicationContext)context).close();
20    }
21
22 }
```



# Osnovna konfiguracija zrna

```
beans.xml  App.java  Osoba.java ✕
1 package rs.ac.singidunum.fir.pj;
2
3 public class Osoba {
4
5     private long jmbg;
6     private String ime;
7
8     public Osoba() {
9     }
10
11     public Osoba(long jmbg, String ime) {
12         super();
13         this.jmbg = jmbg;
14         this.ime = ime;
15     }
16
17     public void govori() {
18         System.out.println("Ja sam " + this.ime + " i imam jmbg " + this.jmbg);
19     }
20 }
```

# Osnovna konfiguracija zrna



# Osnovna konfiguracija zrna

**Element Details**

Set the properties of the selected element. Required fields are denoted by "\*\*".

**index:**

**name:** ime

**ref:**

**type:** String

**value:** Milos Mravik

**Element Details**

Set the properties of the selected element. Required fields are denoted by "\*\*".

**index:**

**name:** jmbg

**ref:**

**type:** long

**value:** 1234567891029

# Osnovna konfiguracija zrna

- Ožičavanje parametara konstruktora klase Osoba

```
beans.xml App.java Osoba.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
5
6
7 <bean id="osoba" class="rs.ac.singidunum.fir.pj.Osoba">
8     <constructor-arg name="ime" type="String"
9         value="Milos Mravik">
10    </constructor-arg>
11    <constructor-arg name="jmbg" type="Long"
12        value="1234567891029">
13    </constructor-arg>
14 </bean>
15 </beans>
```

**Hvala na pažnji!**  
**Pitanja?**