



Zadaci 2

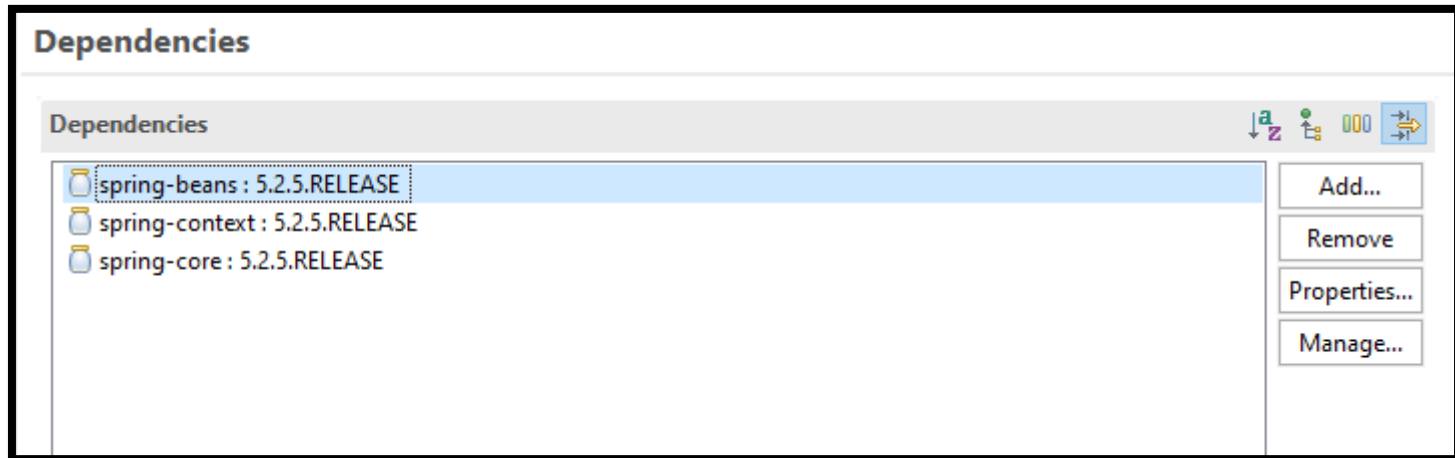
Zadatak 1

Zadatak 1: Napisati Spring projekat upotrebom Maven-a koji sadrži klasu Auto koja sadrži polja marka i tip, klasu Osoba koja sadrži polja ime, prezime i jmbg i klasu SaobracajnaDozvola koji ima polja auto tipa Auto i polje vlasnik tipa Osoba. Klase sadrže konstruktore, set i toString() metode. Kreirati po jedan objekat svake klase (zrna i koristiti injekciju kroz konstruktor kako bi prosledili vrednosti polja klasa). Izvršiti automatsko ožičavanje:

- ◆ Po imenu
- ◆ Po tipu
- ◆ Po konstruktoru

U glavnoj klasi odštampati podatke jedne saobraćajne dozvole.

Zadatak 1

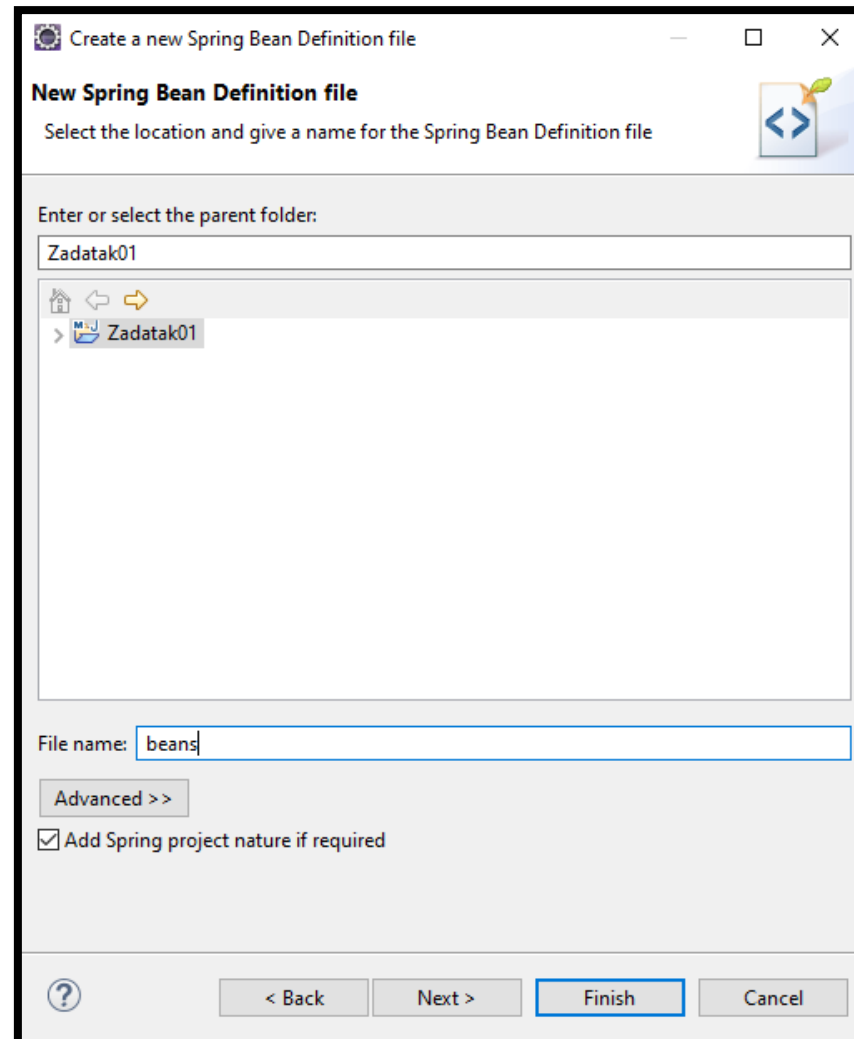


The screenshot shows the 'Dependencies' view in an IDE. The title bar reads 'Dependencies'. Below the title bar, there is a toolbar with icons for sorting (a-z, z-a), a tree view icon, a refresh icon, and a search icon. The main area contains a list of dependencies:

- spring-beans : 5.2.5.RELEASE
- spring-context : 5.2.5.RELEASE
- spring-core : 5.2.5.RELEASE

To the right of the list, there are four buttons: 'Add...', 'Remove', 'Properties...', and 'Manage...'.

Zadatak 1



Zadatak 1

```
Zadatak01/pom.xml  beans.xml  Osoba.java  ⌵
1  package rs.ac.singidunum.fir.pj;
2
3  public class Osoba {
4      private String ime, prezime;
5      private long jmbg;
6
7      public Osoba(String ime, String prezime, long jmbg) {
8          this.ime = ime;
9          this.prezime = prezime;
10         this.jmbg = jmbg;
11     }
12
13     public void setIme(String ime) {
14         this.ime = ime;
15     }
16
17     public void setPrezime(String prezime) {
18         this.prezime = prezime;
19     }
20
21     public void setJmbg(long jmbg) {
22         this.jmbg = jmbg;
23     }
24
25     public String toString() {
26         return ime + " " + prezime + " " + jmbg;
27     }
28 }
```

Zadatak 1

```
Zadatak01/pom.xml  beans.xml  Osoba.java  Auto.java ✕
1 package rs.ac.singidunum.fir.pj;
2
3 public class Auto {
4     private String marka, tip;
5
6     public Auto(String marka, String tip) {
7         this.marka = marka;
8         this.tip = tip;
9     }
10
11     public void setMarka(String marka) {
12         this.marka = marka;
13     }
14
15     public void setTip(String tip) {
16         this.tip = tip;
17     }
18
19     public String toString() {
20         return "Auto [marka=" + marka + ", tip=" + tip + "];";
21     }
22 }
```

Zadatak 1

```
Zadatak01/pom.xml  beans.xml  Osoba.java  Auto.java  SaobracajnaDozvola.java  ⌵
1 package rs.ac.singidunum.fir.pj;
2
3 public class SaobracajnaDozvola {
4     private Osoba vlasnik;
5     private Auto auto;
6
7     public void setVlasnik(Osoba vlasnik) {
8         this.vlasnik = vlasnik;
9     }
10
11    public void setAuto(Auto auto) {
12        this.auto = auto;
13    }
14
15    @Override
16    public String toString() {
17        return "SaobracajnaDozvola [vlasnik=" + vlasnik + ", auto=" + auto + "];"
18    }
19 }
```

Zadatak 1

The screenshot displays the Spring Beans IDE interface. On the left, the 'Beans Overview' tab is active, showing a tree view of beans. The tree is organized as follows:

- beans
 - auto
 - [String] "Hyundai" (selected)
 - [String] "Tucson"
 - vlasnik
 - [String] "Milos"
 - [String] "Mravik"
 - [Long] "1234567891239"
 - saobracajnaDozvola

Below the tree is a search box containing 'type filter text'. To the right of the tree are three buttons: 'New Bean...', 'Up', and 'Down'. On the far right, the 'Element Details' panel is visible, showing the following properties for the selected bean:

- index:
- name: marka
- ref:
- type: String
- value: Hyundai

Below the details panel is a 'Documentation' section with the text: 'Element : constructor-arg Bean definitions can specify... correspond to either a specific... single generic argument... constructor-arg elements... instance factory methods... Content Model : (description... [Click for additional documentation](#)

Zadatak 1

```
Zadatak01/pom.xml  beans.xml  Osoba.java  Auto.java  SaobracajnaDozvola.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         ApplicationContext context = new FileSystemXmlApplicationContext("beans.xml");
10
11         SaobracajnaDozvola dozvola = (SaobracajnaDozvola) context.getBean("saobracajnaDozvola");
12         System.out.println(dozvola.toString());
13
14         ((FileSystemXmlApplicationContext) context).close();
15     }
16 }
```

Zadatak 1

Overview

General Overview [-] [+]

Select an element to edit its details.

type filter text

- beans
 - auto
 - vlasnik
 - saobracajnaDozvola

New Bean...
Up
Down

Element Details

Set the properties of the selected element. Required f

id: saobracajnaDozvola

abstract:

autowire: **byName**

autowire-candidate:

class: rs.ac.singidunum.fir.pj.Saobra

depends-on:

Overview

General Overview [-] [+]

Select an element to edit its details.

- beans
 - auto
 - vlasnik
 - saobracajnaDozvola

New Bean...
Up
Down

Element Details

Set the properties of the selected element

id: saobracajnaD

abstract:

autowire: **byType**

autowire-candidate:

class: rs.ac.singidun

Zadatak 1

```
Zadatak01/pom.xml beans.xml Osoba.java Auto.java SaobracajnaDozvola.java App.java
1 package rs.ac.singidunum.fir.pj;
2
3 public class SaobracajnaDozvola {
4     private Osoba vlasnik;
5     private Auto auto;
6
7     public void setVlasnik(Osoba vlasnik) {
8         this.vlasnik = vlasnik;
9     }
10
11    public void setAuto(Auto auto) {
12        this.auto = auto;
13    }
14
15    @Override
16    public String toString() {
17        return "SaobracajnaDozvola [vlasnik=" + vlasnik + ", auto=" + auto + "];"
18    }
19
20    public SaobracajnaDozvola(Osoba vlasnik, Auto auto) {
21        this.vlasnik = vlasnik;
22        this.auto = auto;
23    }
24 }
```

Zadatak 1

The screenshot shows the IDE's Overview window. On the left, a tree view shows a project with a 'beans' folder containing 'auto', 'vlasnik', and 'saobracajnaDozvola'. The 'saobracajnaDozvola' bean is selected. In the center, there are buttons for 'New Bean...', 'Up', and 'Down'. On the right, the 'Element Details' panel shows the 'id' as 'saobracajnaDozvola' and the 'autowire' property set to 'constructor', which is highlighted with a red box.

The screenshot shows the IDE's Console window. The output text is: `<terminated> App (6) [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (08.05.2020. 12:48:24 - 12:48:25)` followed by `SaobracajnaDozvola [vlasnik=Milos Mravik 1234567891239, auto=Auto [marka=Hyundai, tip=Tucson]]`.

Zadatak 2

Zadatak 2: Izmeniti prehodni zadatak tako da se automatsko ožičavanje vrši pomoću anotacije.

Zadatak 2

General Overview

Select an element to edit its details.

type filter text

- beans
 - auto
 - vlasnik
 - saobracajnaDozvola

New Bean...
Up
Down

Element Details

Set the properties of the selected element. Required fields are denoted with a red asterisk.

id:	saobracajnaDozvola
abstract:	
autowire:	
autowire-candidate:	
class:	rs.ac.singidunum.fir.pj.SaobracajnaDozvola
depends-on:	

Configure Namespaces

Select XSD namespaces to use in the configuration file


- aop - <http://www.springframework.org/schema/aop>
- beans - <http://www.springframework.org/schema/beans>
- c - <http://www.springframework.org/schema/c>
- cache - <http://www.springframework.org/schema/cache>
- context - <http://www.springframework.org/schema/context>
- jee - <http://www.springframework.org/schema/jee>
- lang - <http://www.springframework.org/schema/lang>
- p - <http://www.springframework.org/schema/p>
- task - <http://www.springframework.org/schema/task>
- util - <http://www.springframework.org/schema/util>

Zadatak 2

Context Overview

Select an element to edit its details.

type filter text

▼  beans

 annotation-config

Zadatak 2

```
Zadatak01/pom.xml | beans.xml | Osoba.java | Auto.java | SaobracajnaDozvola.java | App.java
1 package rs.ac.singidunum.fir.pj;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 public class SaobracajnaDozvola {
6     // @Autowired
7     private Osoba vlasnik;
8     // @Autowired
9     private Auto auto;
10
11     @Autowired
12     public void setVlasnik(Osoba vlasnik) {
13         this.vlasnik = vlasnik;
14     }
15
16     @Autowired
17     public void setAuto(Auto auto) {
18         this.auto = auto;
19     }
20
21     @Override
22     public String toString() {
23         return "SaobracajnaDozvola [vlasnik=" + vlasnik + ", auto=" + auto + "];"
24     }
25     /* @Autowired
26     public SaobracajnaDozvola(Osoba vlasnik, Auto auto) {
27         this.vlasnik = vlasnik;
28         this.auto = auto;
29     }
30     */
31 }
32
33 //SAOBRAKAJNA DOZVOLA NE SME DA IMA KONSTRUKTOR AKO RADIMO OZICAVANJE PO TIPU I IMENU,
34 //KADA RADIMO PO KONSTRUKTORU ONDA MORA DA IMA KONSTRUKTOR
```


Zadatak 3

Zadatak 3: Napisati Spring projekat upotrebom Maven-a koji sadrži funkcionalni interfejs Posao koji ima metod uradiPosao(). Klase Menadzer i Radnik implementiraju interefejs Posao tako što menadzer ispisuje poruku „Plan rada je napravljen“ dok radnik ispisuje poruku „Dodeljen posao je izvršen“. Klasa Firma ima dva polje tipa Posao, menadzer i radnik, i metode napraviPlan() i sprovediPlan(). Prva metoda poziva metodu uradiPosao polja menadzer, a druga metoda poziv uradiPosao polja radnik. Kreirati po jedan objekat svake klase (zrna) i izvršiti automatsko ožičavanje.

Hvala na pažnji!
Pitanja?