# Advanced Data Centric Web Applications

## Lab 3 Bean Configuration

* Create a new Maven project as described in Lab 1 and add the latest spring-core, spring-beans and spring-context modules to the project.

Create a class called *Person* with the following attributes:

* String firstName
* String surname
* Address address

Create a class called *Address* with the following attributes:

* String street
* String town
* String county

Create a class called *App* which has a *main* method and uses Spring Dependency Injection to inject the *Address* bean into the *Person* bean.

Both the *Person* and *Address* classes should have *toString* methods.

* Create a new Maven project as described in Lab 1 and add the latest spring-core, spring-beans and spring-context modules to the project.

Create a class called *Engine* with the following attributes:

* double size
* String fuel

Create a class called *Car* with the following attributes:

* String reg
* int doors
* String colour
* Engine engine

Create a class called *App* which has a *main* method and uses Spring Dependency Injection to inject appropriate *Engine* bean into the corresponding *Car* bean as described in the tables below.

Both the *Engine* and *Car* classes should have *toString* methods.

Two *Engine* beans should be created with the following constructor-args:

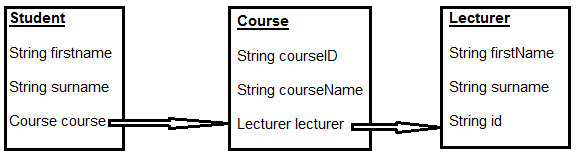
|  |  |  |
| --- | --- | --- |
| Bean ID | size | fuel |
| engine1.2 | 1.2 | Petrol |
| engine1.6 | 1.6 | Diesel |

Three *Car* beans should be created with the following constructor-args:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bean ID | reg | doors | colour | Engine (Bean ID) |
| 2017-G-1 | 2017-G-1 | 5 | Red | engine1.6 |
| 2000-G-456 | 2000-G-456 | 4 | Green | engine1.6 |
| 2010-WH-77 | 2010-WH-77 | 5 | Black | engine1.2 |

* Create a new Maven project as described in Lab 1 and add the latest spring-core, spring-beans and spring-context modules to the project.

The following diagram describes how the objects interact – each student is doing one course, and each course is taught by only one lecturer.



Use Spring Dependency Injection to create two *Student* beans, both of which are doing the same course, taught by the same Lecturer

For example, when the application is run and the beans are set up, the output could look something like as follows:

Student [firstName=John, surname=Smith, course=Course [courseID=SSRAD, courseName=Server Side RAD, lecturer=Lecturer [firstName=Gerard, surname=Harrison, id=G0012345]]]

Student [firstName=Patrick, surname=O'Malley, course=Course [courseID=SSRAD, courseName=Server Side RAD, lecturer=Lecturer [firstName=Gerard, surname=Harrison, id=G0012345]]]

* Create a new Maven project as described in Lab 1 and add the latest spring-core, spring-beans and spring-context modules to the project.

Create a class Employee as follows:

public class Employee {

private String name;

private double salary;

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

public void setName(String name) {

this.name = name;

}

public void setSalary(double salary) {

this.salary = salary;

}

@Override

public String toString() {

return "Employee [name=" + name + ", salary=" + salary + "]";

}

}

Create a Spring bean that uses the 2-argument constructor.

Create a Main Class that uses Spring Dependency Injection to inject the bean into two different variables of type Employee.

Change the attributes of one of the instances using the setter methods and print out both instances. What happens & why?

Now change the bean’s scope to prototype and re-run the application. What happens & why?