# Advanced Data Centric Web Applications

## Lab 4 DI and Autowiring

### Part 1 DI

* 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 *Module* with the following attributes:

* String moduleName
* int credits
* boolean allCA

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

* String name
* List<Module> modules

Create a number of *Module* beans, and create two *Student* beans, each of which has a list of different modules, using Spring Dependency Injection.

* 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 name
* int age

*Person* should have a private constructor using the above attributes.

In addition, *Person* should have a method called *getInstance* which should be defined as a factory-method when creating the *Person* bean.

The *getInstance* method should prompt the user to enter the person’s name and age and use this information to create the *Person* bean.

* Update the project used in part 2 of the lab as follows:

The *Person* class has an extra attribute

* String occupation

The *Person* constructor now takes 3 parameters instead of 2. (The extra one is occupation).

When defining the bean the default occupation should be *student.*

The *getInstance* method should prompt the user to enter the person’s name, age and occupation and use this information to create the *Person* bean.

However, if the user doesn’t enter any occupation – the default occupation defined in the bean should be used.

Example 1 – No occupation entered:

Enter your name:

John

Enter your age:

21

Enter your occupation:

Person [name=John, age=21, occupation=Student]

Example 1 – Occupation entered:

Enter your name:

Bill

Enter your age:

30

Enter your occupation:

Garda

Person [name=Bill, age=30, occupation=Garda]

### Part2 Autowiring

* 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 *Car* with the following attributes:

* String reg
* double engineSize

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

* String street
* String town
* String county

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

* String name
* Address address
* Car car
* a constructor that takes only 1 parameter: name
* setter methods for address and car

Create Spring beans for each class and set the *Person*’s *autowire* attribute to **no.**

Create 2 *Car* beans, 3 *Address* beans and 2 *Person* beans and manually wire them up.

* Update the project just created so that the *Person* bean’s autowiring is **byName**.

Now update the project again and see what happens - and why - in the following situations:

* + Change the *id* of the *Car* bean to “carxyz”.
  + Change the *id* of the *Car* bean to “address”.
  + Change the setter for the *Person’s* car attribute from
    - setCar(Car car)

to

* + - setCarAsdf(Car car)
  + Add *property* elements to the *Person* bean.
* Update the project created in Part 2 question 1 so that the *Person* bean’s autowiring is **byType**.

Now update the project again and see what happens - and why - in the following situations:

* + Change the *id* of the *Car* bean to “carxyz”.
  + Remove the *Car*bean.
  + Create two *Car beans.*
  + Change the setter for the *Person’s* car attribute from
    - setCar(Car car)

to

* + - setCarAsdf(Car car)
  + Add three new methods to *Person*

**public** **void** setCarA(Car car) {

**this**.car = **new** Car("AA-A-AAAA", 1.0);

}

**public** **void** setCarB(Car car) {

**this**.car = car;

}

**public** **void** setCarC(Car car) {

**this**.car = **new** Car("CC-C-CCCC", 1.0);

}

* + Add *property* elements to the *Person* bean.
* Update the project created in Part 2 question 1 so that the *Person* bean’s autowiring is **constructor**.

The constructor should be updated to have 3 arguments:

* name
* address
* car

Now update the project again and see what happens - and why - in the following situations:

* + Remove the *Car* bean.
  + Create two *Car* beans one called *car* and one called *carA*.
  + Add *property* elements to the *Person* bean.