# Lab 3: Introduction to Dependency Injection

In this lab, you will work with the basics of Spring dependency injection.

## Objectives:

- To work with dependency injection

### Part 1: Dependency Injection

In this part, you will work with constructor injection and setter injection.

## Steps:

_1. This lab depends on successfully completing the basic parts of the last lab - it doesn't depend on the "experiments". If you did not finish the basic part of the last lab, you should either finish it or ask the instructor to help you get the last lab's solution.

_2. Follow this procedure to make a copy of your lab02 project:

   a. Choose *File* - *Close All* to close all open editors.

   b. In the Project Explorer, right-click on the *lab02* project and choose *Copy*.

   c. Right-click the blank, white area of the Project Explorer and choose *Paste*.

   In the *Copy Project* dialog, enter **lab03** and press OK.

_3. In the same fashion as in the first lab, create a new class in the *com.oaktreeair.ffprogram* package named **AddressInfo**.

   Complete the class so it has the following properties, complete with get/set methods:

```
private String street;
private String city;
private String state;
private String zip;
private String country;
```

   Then write a constructor in the new class that accepts parameters corresponding to each of the fields.

_4. In the *spring.xml* file, use *constructor injection* to define an AddressInfo bean with values of your choice. Assign the new bean an ID of **addrInfo01**.

_5. Modify the *Flier* interface and the *FlierImpl* class to define a property of type AddressInfo named **homeAddress**. Be sure to define a get/set method pair for the new property:

```
public void setHomeAddress(AddressInfo inf);
public AddressInfo getHomeAddress();
```

_6. In the *spring.xml* file, use *setter injection* to inject the *addrInfo01* bean reference into the new property.

_7. In the FrequentFlierProgram.java file, add code to the *main* method to retrieve the home address from the flier and display its properties to the console.

_8. Run the FrequentFlierProgram.java class as a Java Application. You should see the home address.

## Part 2: The @Required Annotation

In this part, you will work with required properties.

### Steps:

_1. The ContactInfo bean you created in an earlier lab has four properties: emailAddress, homePhone, mobilePhone and smsNumber. Let's make the first two required properties:

a. Open ContactInfo.java into the editor. Find the *setEmailAddress* method, and immediately above it, annotate it as @Required:

```
@Required
public void setEmailAddress(String emailAddress)
{
    this.emailAddress = emailAddress;
}
```

Use *Source - Organize Imports* so that RAD imports org.springframework.beans.factory.annotation.Required.

b. Repeat for the *setHomePhone* method.

c. Edit *spring2.xml* (where the contact01 bean is configured), and comment out the setter-injection lines for the emailAddress and homePhone properties.

d. Above the "contact01" bean definition, add the following to enable annotation processing:

```
<bean class=
"org.springframework.beans.
factory.annotation.RequiredAnnotationBeanPostProcessor"/>
```

**Note:** The second line in the code above (the fully-qualified class name) is split so it fits on the printed page, but you must enter it unbroken.

e. Try running FrequentFlierProgram.java again - You should get a BeanCreationException. Uncomment the required property configuration before continuing.

## Part 3: Experimenting

In this part, you can experiment with your application. The following steps list things you can try.

### Steps:

_1. Try changing the "flier01" bean's ID configuration to **"abc"** instead of a number. Run the main program and see what happens. Be sure to change it back.

_2. In the ContactInfo class, write a constructor that accepts values corresponding to all of the fields. Then, without changing any configuration, run the main program. What happens, and why? Fix the problem either by deleting the new constructor, writing a zero-argument constructor or by updating the configuration.