# Lab S1: Spring Framework Basics

## Objectives

In this lab you will use some of the basic features of the Spring Core Framework to deploy a very basic Java application.

The application has the following components:

1. A Consultant type, implemented as an interface. The interface has one method called “getAdvice()”
2. Two subtypes of Consultant, an ITGuru and a PRWhiz implemented as POJOs

## Lab Setup

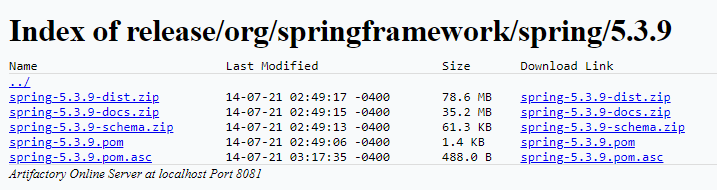
This lab will be set up as an ordinary Java project with the Spring jars added as an external archive. Once the basic project is set up and works, you will “Springify” the project by setting up a Spring context that will manage the Java components

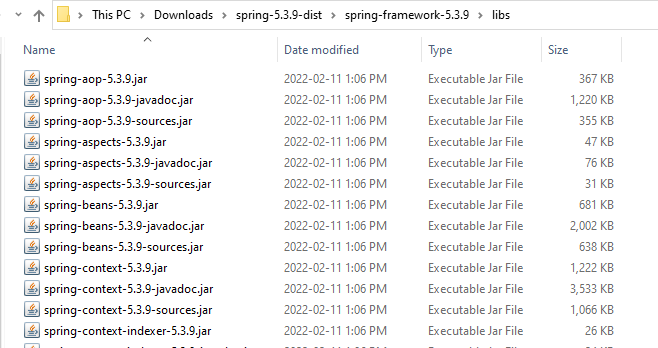
### Step 1: Set up the Java Project

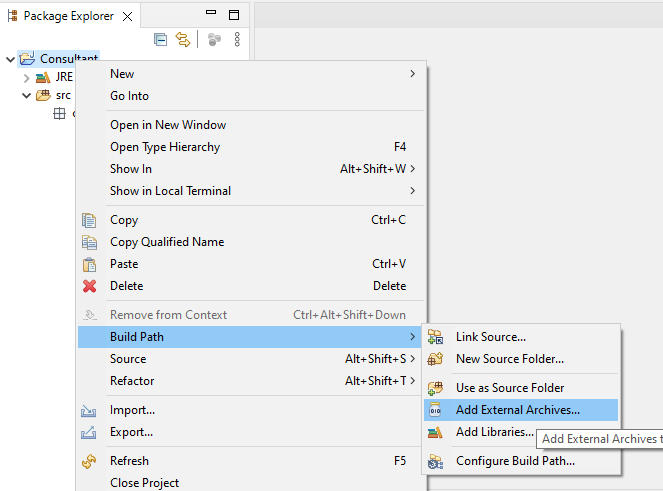
1. Create a Java Project. In the example code, this project is called Consultant\_1.
2. Create a source package called com.consulting.

### Step 2: Add the Spring Jar files

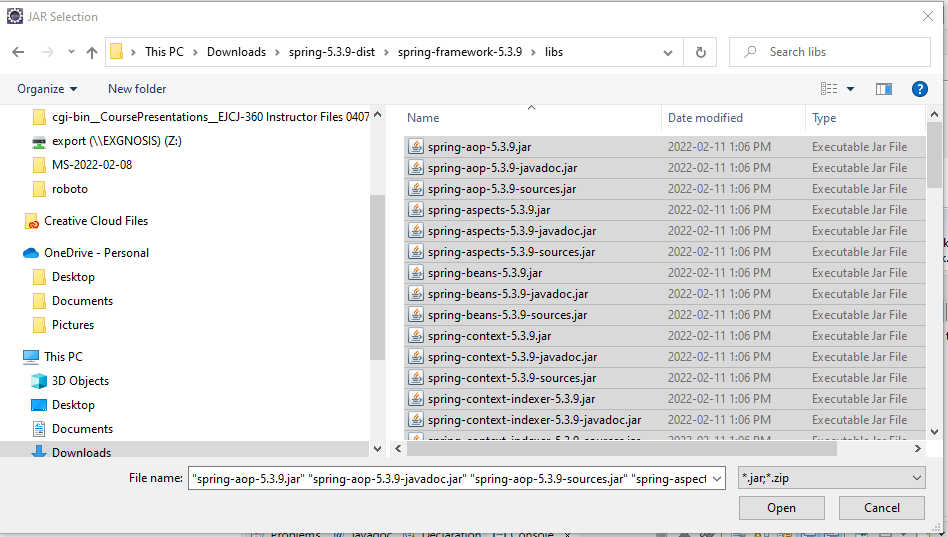
1. Download the latest Spring jars from the link below. At the time this was written, the latest version was 5.3.9 but you may have the option of downloading a more recent version

[](https://repo.spring.io/ui/native/release/org/springframework/spring/5.3.9/)https://repo.spring.io/ui/native/release/org/springframework/spring/5.3.9/

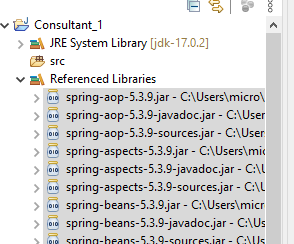
1. Download and unzip the “\*dist.zip” file anywhere, the “Downloads” directory is used in the demo for this lab.
2. You should see something that looks like this”
3. Right click on the project folder and select the “Build Path” option and the “Add external archives” option



1. Using the file browser, go to the directory with the Spring jar files, select them all and then add them using the “Open” option.



1. If successful, you should see the jars appearing under a new folder called “referenced libraries.”



## Coding the Java Components

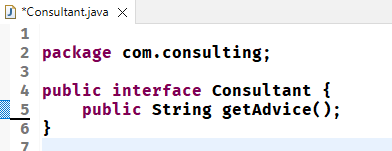
In this section, you will write the code for the application and test the application to ensure it is working.

In the next section, you will add the Spring code to let Spring manage the POJOs for the application.

Following good programming practices, you will write the code interface first then write the implementation classes that instantiate the interface

### Step 1: Create the Consultant interface

1. At this point, you should be able to add the following interface to the project without problem.
2. Ensure that you have already created the package com.consulting



### Step 2: Create the Implementation Classes

1. Graphical user interface, text, application

   Description automatically generatedCreate two implementation classes which are identical except for the message returned from the getAdvice() method

Graphical user interface, text, application

Description automatically generated

### Step 3: Write the App class

1. Create a main app class that instantiates an ITGuru object and a PRWhiz object and invokes the getAdvice() method on both.
2. Run the application and confirm the output is correct.

Graphical user interface, text, application

Description automatically generated

## Adding Spring

### Step1: Add the configuration file

1. An XML file needs to be used to configure the String context when it is created.
2. The file must comply to a specific XML schema.
3. The file you will be using is called config.xml and is shown below
4. The file should be placed in the src directory and directs Spring to scan through the Java files in the package specified for Spring annotations
5. The file is available as part of the lab resources
6. In the samples below, the name of the project has been changes to Consultant\_2

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:context="http://www.springframework.org/schema/context"

    xsi:schemaLocation="http://www.springframework.org/schema/beans

    http://www.springframework.org/schema/beans/spring-beans.xsd

    http://www.springframework.org/schema/context

    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- add entry to enable component scanning -->

    <context:component-scan base-package="com.consuting" />

</beans>

### Step 2: Identify the Spring components

1. Add the @Component directive to the classes that implement the Consultant interface.
2. The name of the component will default to the name of the class for ITGuru but is explicitly given as “SpinDoctor” for PRWhiz.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

### Step 3: Create the Spring context

1. The first thing that the application needs to do is to wrap the client code inside an application context
2. The XML configuration file is read in, and the context object created
3. During the application lifetime, the context manages the POJOs
4. When the application context is closed, all of the existing POJOs are destroyed

Graphical user interface, text, application, Word

Description automatically generated

### Step 4: Create the beans

1. Now use the bean factory methods to get references to the POJOs created by the Spring context
2. Graphical user interface, text

   Description automatically generatedPrint out the results to show that the objects are created correctly