# Exercise 1 – Dependency Injection Using Spring Framework

## 1 Overview

In this exercise you will develop a simple application, which injects bean objects into Java collections at runtime using the Spring framework. Collection types demonstrated in this example include List, Set, Map and Properties.

Estimated time to complete this tutorial: 30 minutes.

Note: This tutorial is based on Eclipse IDE for Java EE Developers, version Kepler SR1. If you are using a different version of Eclipse, or a different IDE, the actual screens you see will be different from the screenshots in this tutorial.

## 2 Environment & Tools

* JDK 7  
  <http://www.oracle.com/technetwork/java/javase/downloads/>
* Eclipse IDE (Kepler SR1) for Java EE Developers  
  <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr1>
* Apache Commons Logging  
  <http://commons.apache.org/proper/commons-logging/download_logging.cgi>
* Spring Framework 4.0.3  
  <http://repo.spring.io/release/org/springframework/spring/4.0.3.RELEASE/>

## 3 Tasks

### Task 1: Set up the development environment

#### Step1: Create a new Java project

Open Eclipse, select **File** -> **New** -> **Project** -> **Java Project**. Give your project a name in the pop-up window, and make sure you have selected “Create separate folders for source and class files”.

Then, right click on the source folder (e.g. src), and create a new package, namely napier.spring.exercise.

#### Step 2: Add Spring dependencies to the project

You should have downloaded the Spring Framework 4.0.3. This is typically a zip file, e.g. spring-framework-4.0.3.RELEASE-dist.zip. Unzip the file to find the following jars:

* spring-beans-4.0.3.RELEASE.jar
* spring-context-4.0.3.RELEASE.jar
* spring-core-4.0.3.RELEASE.jar
* spring-expression-4.0.3.RELEASE.jar

In addition, you should have downloaded the Apache Commons Logging library, e.g. commons-logging-1.1.3-bin.zip. Within this file you should be able to find commons-logging-1.1.3.jar.

Add all the five jars into the build path of your project.

### Task 2: Build the Spring Beans

#### Step 1: Create a Cohort Class

Right click on the package napier.spring.exercise and create a new Class, namely Cohort. The content of this class is listed below:

**package** napier.spring.exercise;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Properties;

**import** java.util.Set;

**public** **class** Cohort {

**private** List<Object> list;

**private** Set<Object> set;

**private** Map<Object, Object> map;

**private** Properties props;

**public** List<Object> getList() {

**return** list;

}

**public** **void** setList(List<Object> list) {

**this**.list = list;

}

**public** Set<Object> getSet() {

**return** set;

}

**public** **void** setSet(Set<Object> set) {

**this**.set = set;

}

**public** Map<Object, Object> getMap() {

**return** map;

}

**public** **void** setMap(Map<Object, Object> map) {

**this**.map = map;

}

**public** Properties getProps() {

**return** props;

}

**public** **void** setProps(Properties props) {

**this**.props = props;

}

}

Note: For demonstration purposes, this class contains four collection objects of List, Set, Map and Properties types respectively. Later on, these will be injected by the Spring framework using corresponding setters at runtime.

#### Step 2: Create a Person Class

Create a Person Class with the following source code:

**package** napier.spring.exercise;

**public** **class** Person {

**private** Long id;

**private** String name;

**public** **void** setId(Long id){

**this**.id = id;

}

**public** **void** setName(String name){

**this**.name = name;

}

@Override

**public** String toString() {

**return** "Person:[" + id +"," + name + "]";

}

}

Note: This is a plain Spring Bean, which will be instantiated at runtime, and injected into a cohort object.

### Task 3: Configure Spring Dependency Injection

Right click on the source folder (e.g. src) and create a new XML file, namely applicationContext.xml. The content of this file is listed below:

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-4.0.xsd"*>

<bean id=*"cohortBean"* class=*"napier.spring.exercise.Cohort"*>

<property name=*"list"*>

<list>

<value>listValue1</value>

<ref bean=*"personBean"* />

<bean class=*"napier.spring.exercise.Person"*>

<property name=*"name"* value=*"Bob"* />

<property name=*"id"* value=*"112233"* />

</bean>

</list>

</property>

<property name=*"set"*>

<set>

<value>setValue1</value>

<ref bean=*"personBean"* />

<bean class=*"napier.spring.exercise.Person"*>

<property name=*"name"* value=*"Chris"* />

<property name=*"id"* value=*"778899"* />

</bean>

</set>

</property>

<property name=*"map"*>

<map>

<entry key=*"1"* value=*"mapValue1"* />

<entry key=*"2"* value-ref=*"personBean"* />

<entry key=*"3"*>

<bean class=*"napier.spring.exercise.Person"*>

<property name=*"name"* value=*"David"* />

<property name=*"id"* value=*"987654"* />

</bean>

</entry>

</map>

</property>

<property name=*"props"*>

<props>

<prop key=*"key1"*>1000</prop>

<prop key=*"key2"*>2000</prop>

<prop key=*"key3"*>3000</prop>

</props>

</property>

</bean>

<bean id=*"personBean"* class=*"napier.spring.exercise.Person"*>

<property name=*"name"* value=*"Alice"* />

<property name=*"id"* value=*"123456"* />

</bean>

</beans>

Note:

* This configuration file defines a bean called personBean, of type Person. Its properties, such as name and id, are instantiated via setter injection. This bean shall be created first, because it is referred to by other sections of the configuration file.
* The cohortBean of type Cohort is also instantiated via setter injection. Read the examples carefully to find out how Spring injects List, Set, Map and Properties types.
* For List, Set and Map there are three ways of setting an object value. It could be a literal value like listValue1, setValue1 and mapValue1; or a reference to another Spring Bean, such as the personBean for Alice; or an inline bean like the ones for Bob, Chris and David.

### Task 4: Run the application

Create a Main Class as below:

**package** napier.spring.exercise;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new**

ClassPathXmlApplicationContext("applicationContext.xml");

Cohort c = (Cohort)context.getBean("cohortBean");

System.*out*.println("List: " + c.getList());

System.*out*.println("Set: " + c.getSet());

System.*out*.println("Map: " + c.getMap());

System.*out*.println("Props: " + c.getProps());

}

}

Run the application, and the console output should look like:

List: [listValue1, Person:[123456,Alice], Person:[112233,Bob]]

Set: [setValue1, Person:[123456,Alice], Person:[778899,Chris]]

Map: {1=mapValue1, 2=Person:[123456,Alice], 3=Person:[987654,David]}

Props: {key3=3000, key2=2000, key1=1000}