CS-321: Server-Side Web Development
Spring 2021
Northeastern Illinois University
Homework #2: Dependency Injection
Suggested Due Date: Thursday, 01/28/21 by 6:00 p.m. CDT
Terminal Due Date: Thursday, 02/11/21 by 6:00 p.m. CDT

Homework Goals:

1. Build a basic Spring project without annotations to demonstrate dependency injection.
2. Build a basic Spring project with annotations to demonstrate dependency injection.
3. Build a basic Spring project with component scanning to demonstrate dependency injection.
4. Deadlines are flexible. The suggested deadline is when I think you should submit your homework by to stay on track. You can submit before then of course! It takes me about 5-7 days to provide feedback.
5. The terminal due date is the last date/time that you can submit the assignment by to receive credit.

Part I: Basic Spring project, no annotations

- Launch IntelliJ and choose New Project. Click Maven on the sidebar and then click Next (do not check the archetype checkbox).
- Create a folder with the name `hw02-p1` (this will also be the name of the project). Expand the Artifact Coordinates and fill in `edu.cs321` for GroupId. Click Finish.
- In the pom.xml file, add the following dependencies section underneath the version tags and click the little `m` icon that appears in the window to have Maven download the dependencies.

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.3</version>
    </dependency>

    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.1</version>
    </dependency>
</dependencies>
```

- In `src` → `main` → `java`, add a package named `edu.cs321`.
- In that package, create a Java class named `BeanB`. `BeanB` should have a default constructor that prints out the text: `Creating BeanB`

- In the same package, create a Java class named `BeanC`. `BeanC` should have a default constructor that prints out the text: `Creating BeanC`
- Then, in the same package, create a Java class named `BeanA` with the following:
  - A private instance variable of type `BeanB` named `beanB`.
  - A private instance variable of type `BeanC` named `beanC`.
  - A default constructor that prints out the text: `Creating BeanA`
  - A setter for the `beanB` instance variable. This setter should also print out the text: `Setting bean reference for BeanB`
  - A setter for the `beanC` instance variable. This setter should also print out the text: `Setting bean reference for BeanC`
- Right click on the `src` → `main` → `resources` folder, and choose add a new XML Configuration File → Spring Config. Name it `beans`. You will see a warning at the top about your application context. But, because we are going to load the application context using Java code, you can ignore this.
- In `src` → `test` → `java`, add a class named `BeanDemo` and a `main` method. Add the following code:

```
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
```

- As comments in this class, answer the following questions:
  1. What does `ClassPathXmlApplicationContext` do?
  2. Where does `ClassPathXmlApplicationContext` expect the XML application context to be located?
- Run the `main` method. Why do you think there is no output?
- Go back to the `beans.xml` file and add the following code inside the `beans` tags (after the opening tag):

```
<bean id="beanA" class="edu.cs321.BeanA"></bean>
<bean id="beanB" class="edu.cs321.BeanB"></bean>
<bean id="beanC" class="edu.cs321.BeanC"></bean>
```

- Run the `main` method in the `BeanDemo` class. Answer the following question as a comment in the class:
  3. What is the role of the `ApplicationContext` (related to beans) and how does it know how to do this?
  4. Is this dependency injection? Why or why not?
- In the `beans.xml` file, modify the first bean declaration (for `BeanA`) so that the following is inside the tags:

```
    <property name="beanB" ref="beanB"></property>
    <property name="beanC" ref="beanC"></property>
```

- Run the `main` method in the `BeanDemo` class. Answer the following question as a comment in the class:

    4. How is this output different?
    5. Is this dependency injection? Why or why not?

Part II: Basic Spring project, with annotations

- As in Part I, create a new project and name it `hw02-p2`. Use `edu.cs321` for GroupId.
- As in Part I, add the dependencies to the pom.xml file, create the three bean classes and the `BeanDemo` class. Also create the `beans.xml` file, but do not add the bean declarations.
- Above each bean class header, add the annotation `@Component`. Above the setters in `BeanA`, add the annotation `@Autowired`.
- Modify the beans.xml file to look like the following:

```
<?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
      https://www.springframework.org/schema/context/spring-context.xsd">

   <context:annotation-config />
</beans>
```

- As comments in the `BeanDemo` class, answer the following questions:

    1. What does the `@Component` annotation do?
    2. What does the `@Autowired` annotation do?
    3. What does `<context:annotation-config />` do and what is needed for it to properly inject resources into classes?

- Run the `main` method in the `BeanDemo` class. What happens?
- In the `beans.xml` file, add the following underneath the annotation-config tag:

```
    <bean id="beanA" class="edu.cs321.BeanA"></bean>
    <bean id="beanB" class="edu.cs321.BeanB"></bean>
    <bean id="beanC" class="edu.cs321.BeanC"></bean>
```

- Run the `main` method in the `BeanDemo` class. What happens? Based on this, revise your answer to #3 if needed.

Part III: Basic Spring project, with component scanning

- As in Part II, create a new project and name it `hw02-p3`. Use `edu.cs321` for GroupId.
- As in Part II, add the dependencies to the pom.xml file, create the three bean classes and the `BeanDemo` class. Also create the `beans.xml` file, but do not add the bean declarations.
- Above each bean class header, add the annotation `@Component`. Above the setters in `BeanA`, add the annotation `@Autowired`.
- Modify the beans.xml file to look like the following:

```
<?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
      https://www.springframework.org/schema/context/spring-context.xsd">

   <context:component-scan base-package="edu.neiu.cs321"/>
</beans>
```

- Run the `main` method in the `BeanDemo` class.
- As comments in the `BeanDemo` class, answer the following questions:
  1. What does `<context:component-scan base-package="edu.neiu.cs321"/>` do?
  2. How is the component-scan related to the annotation-config from Part II?

Submitting your code to D2L:

1. Zip up all 3 project folders (`hw01-p2`, `hw02-p2`, `hw02-p3`). In other words, your zip file should contain 3 folders.
2. Submit this one zip file to D2L.