Practice Activity #5 - Dependency Injection with Annotations

Practice Activity #5 - Dependency Injection with Annotations

1. Define a new implementation for the FortuneService.

- Your fortune service should read the fortunes from a file.
- The fortune service should load the fortunes into an array
- When the getFortune() method is called it would return a random fortune from the array.

2. Inject your new dependency into your Coach implementation

3. Test your application to verify you are getting random fortunes based on your fortunes file.

Compare your code to the solution. The solution is available here:
- http://www.luv2code.com/downloads/udemy-spring-hibernate/solution-practice-activities.zip

---

FAQ: I'm getting null for the fortunes.

Question

When I create an array of fortunes using this

String[] data= {a,b,c,d,e};

The array is always null. Why?

Answer

The root cause is this is a Spring Bean Lifecycle issue.

When Spring creates the beans it follows this general process

1. Construct an instance of class
2. Inject dependencies
3. Set properties etc (@Value)

In your case, when you initialized the array using this code

```
// create an array of strings
private String[] data = { a, b, c, d, e };
```

The Spring Bean lifecycle was at step #1 above. It created an instance ... but during the assigment of the string array, the properties/fields for a, b, c, d, e haven't been set yet using @Value. That doesn't happen later until step #3. So that's why you had null with the field assignment.

When you made mods to your code and moved the assignment into the getFortune() method, then by the time this method is invoked all steps 1-3 are already complete and it works as desired.

For this use case, the recommended solution is to use the @PostConstruct annotation. This is called at the end of the bean lifecycle process. So all of steps 1-3 are already completed and then you can safely perform assignments. This is the best use case for making any custom assignments in your code.

Here's the updated code for your service. Make note of this section:

```
private String[] data;

@PostConstruct
public void setupMyData() {

    data = new String[5];

    data[0] = a;
    data[1] = b;
    data[2] = c;
    data[3] = d;
```

```
        data[4] = e;

    }
```

Let me know if this clears it up for you. :-)

NOTE: JAVA 9 USERS HEADS UP

If you are using Java 9 and want to make use of @PostConstruct annotation, then you'll need to add support for @PostConstruct. See this link for how to set it up.

https://www.udemy.com/spring-hibernate-tutorial/learn/v4/t/lecture/9120288?start=0
