Spring Fundamentals with XML Configurations

Task 1:

Create a Maven project and add required dependency of `spring-context 5.1.4.RELEASE`

Create a `Main` class in package `com.stackroute` and two Spring Beans – `Movie`, and `Actor` in package `com.stackroute.domain`.

`Actor` has two `String` properties, `name` and `gender`, and an `age` property of type `int`.

An `Actor` can be initialized with the three properties via the corresponding setter methods. Use property based injection in the bean definition file (beans.xml)

`Movie` "*has a*" `Actor` that can be initialized via the corresponding setter method. Use property based object injection in the bean definition file (beans.xml)

The `Main` class looks up `Movie` bean via three ways to print out actor information:

1. Using `XmlBeanFactory`
2. Using Spring 3.2 `BeanDefinitionRegistry` and `BeanDefinitionReader`
3. Using `ApplicationContext`

Create a `spring-xml-demo` repo and push the code to master branch.

Task 2:

From the master branch of `spring-xml-demo` repo create a `constructor-injection` branch.

Add constructor to the Actor class to initialize `with name` and `gender`, and `age`

Create three beans of type Actor in the bean definition file.

Use constructor-based injection in the bean definition file (`beans.xml`) to inject property values in each of the three beans via name, index, and type respectively.

For the `Movie` bean, use constructor based object injection in the bean definition file (`beans.xml`) to inject an `Actor` bean.

In the `Main` class, look up `Movie` bean using `ApplicationContext` and print out Author information.

Use the same `ApplicationContext` to again look up the same `Movie` bean.

Print out the equality result of the two `Movie` beans.

```
System.out.println(beanA==beanB);
```

Change the scope of the `Movie` bean in `beans.xml` to `prototype` and run the application again.

Note the output.

Replace `id` of the `Movie` bean with `name` having two values, like this:

`<bean name="MovieA, MovieB" ……..>`

Update the code in Main to get the Movie bean by its two different name.

Push the code to `constructor-injection` branch.

Task 3:

From the `constructor-injection` branch of `spring-xml-demo` repo create a `autowire-xml` branch.

For the `Movie` bean, delete the constructor based object injection in the bean definition file (`beans.xml`) that injects an `Actor` bean.

Use autowire byName in the Movie bean to inject an Actor bean.

Run the application.

Create another Movie bean and try autowire byType.

Run the application and note the exception thrown.

Fix the Movie bean by removing autowire byType and using constructor injection instead.

Push the code to `autowire-xml` branch.

Task 4:

From the `autowire-xml` branch of `spring-xml-demo` repo create an `aware-interface` branch.

Implement `ApplicationContextAware`, `BeanFactoryAware`, `BeanNameAware` in the `Movie` class and print out their results.

Push the code to `aware-interface` branch.

Task 5:

From the `aware-interface` branch of `spring-xml-demo` repo create a `bean-lifecycle` branch.

Add a `BeanLifecycleDemoBean` class in `com.stackroute.demo` that implements `InitializingBean` and DisposableBean.

Override the required methods to print out messages.

Define `BeanLifecycleDemoBean` as a bean in `beans.xml`.

Run the application and observe the result.

Add two methods `customInit()` and customD`estroy()` to the `BeanLifecycleDemoBean` class and print out custom messages.

In the `BeanLifecycleDemoBean` bean definition, in `beans.xml`, set the `customInit()` and customD`estroy()` methods to be called.

Run the application.

Push the code to `bean-lifecycle` branch.

Task 6:

From the `bean-lifecycle` branch of `spring-xml-demo` repo create a `bean-post-processor` branch.

Add a `BeanPostProcessorDemoBean` class in `com.stackroute.demo` that implements `BeanPostProcessor`

Override the required methods to print out messages.

Define `BeanLifecycleDemoBean` as a bean in `beans.xml`.

Run the application and observe the result.

Push the code to `bean-post-processor` branch.