**IMPLEMENTING AUTO WIRING IN SPRING FRAMEWORK JAVA CONFIGURATION FILE**

So far, we have observed **the default name of the bean is the name of the method**.

What if we want to **change** this?

Using the **@Bean** annotation with a custom name: we can specify a custom name for a bean using the **name** attribute of the **@Bean** annotation.

**HelloWorldConfiguration.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*

 \* Let's say a spring can managing an object of a custom class

 \* \*/

record Address(String firstLine, String city) {};

record Person(String name, int age) {};

@Configuration

public class HelloWorldConfiguration {

    @Bean

    //Indicates that a method produces a bean to be managed by the Spring container.

    public String name() {

        return "Naveen";

    }

    @Bean

    public int age() {

        return 20;

    }

    @Bean

    public Person person() {

        return new Person("Navaneetha krishnan", 20);

    }

    @Bean(name = "address2")

    public Address address() {

        return new Address("Baker Street", "London");

    }

}

**App02HelloWorldSpring.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        // Launch a Spring Context.

        var context =

         new AnnotationConfigApplicationContext(HelloWorldConfiguration.class);

        System.out.println(context.getBean("name"));

        System.out.println(context.getBean("age"));

        System.out.println(context.getBean("person"));

        System.out.println(context.getBean("address2"));

    }

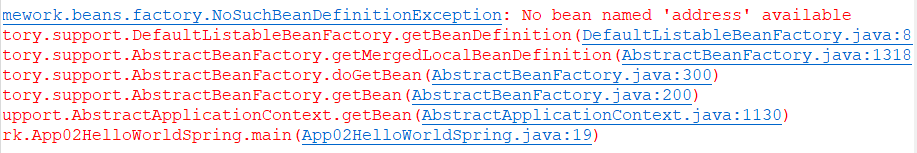
}

OUTPUT:

Text

Description automatically generated

If we mention “address” as a bean name as previous, it will give an exception. Because we have changed that bean name.



There are alternative ways to get a bean.

**HelloWorldConfiguration.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*

 \* Let's say a spring can managing an object of a custom class

 \* \*/

record Address(String firstLine, String city) {};

record Person(String name, int age) {};

@Configuration

public class HelloWorldConfiguration {

    @Bean

    //Indicates that a method produces a bean to be managed by the Spring container.

    public String name() {

        return "Naveen";

    }

    @Bean

    public int age() {

        return 20;

    }

    @Bean

    public Person person() {

        return new Person("Navaneetha krishnan", 20);

    }

    @Bean(name = "address2")

    public Address address() {

        return new Address("Baker Street", "London");

    }

//  @Bean(name = "address1")

//  public Address address1() {

//      return new Address("Baker Street", "Puducherry");

//

//  }

}

**App02HelloWorldSpring.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        // Launch a Spring Context.

        var context =

         new AnnotationConfigApplicationContext(HelloWorldConfiguration.class);

        System.out.println(context.getBean("name"));

        System.out.println(context.getBean("age"));

        System.out.println(context.getBean("person"));

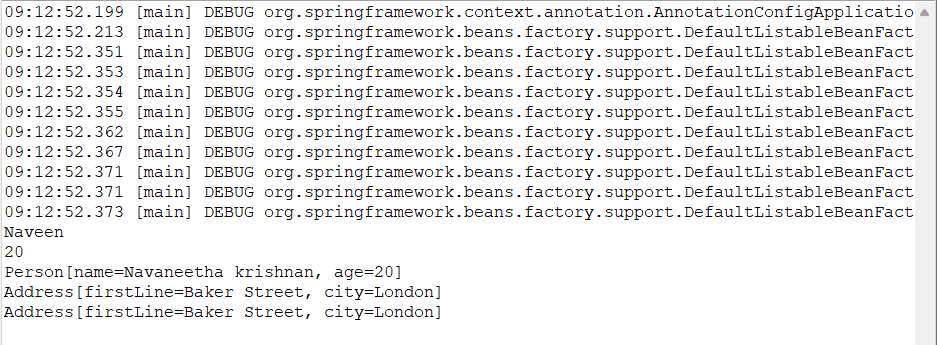
        System.out.println(context.getBean("address2"));

        System.out.println(context.getBean(Address.class));

    }

}

OUTPUT:



Exception: If we have more than one bean having the same return type “Address”.



We can create beans and reuse existing beans which are already managed by Spring framework. In this example, we can link some of the beans.

**HelloWorldConfiguration.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

record Address(String firstLine, String city) {};

record Person(String name, int age, Address address) {};

@Configuration

public class HelloWorldConfiguration {

    @Bean

    public String name() {

        return "Naveen";

    }

    @Bean

    public int age() {

        return 22;

    }

    @Bean

    public Person person() {

        return new Person("Hariharan", 24, new Address("Kochi", "Kerala"));

    }

    @Bean

    public Person person2MethodCall() {

        return new Person(name(), age(), address());

    }

    @Bean

    public Person person3Parameter(String name, int age, Address address2) {

        return new Person(name, age, address2);

    }

    @Bean(name = "address2")

    public Address address() {

        return new Address("Baker Street", "London");

    }

}

In this above example, we are using name bean, age bean and address bean to create the new bean named “person2MethodCall ()”.

**App02HelloWorldSpring.java**

package com.naveen.learnspringframework;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        // Launch a Spring Context.

        var context =

         new AnnotationConfigApplicationContext(HelloWorldConfiguration.class);

        System.out.println(context.getBean("name"));

        System.out.println(context.getBean("age"));

        System.out.println(context.getBean("address2"));

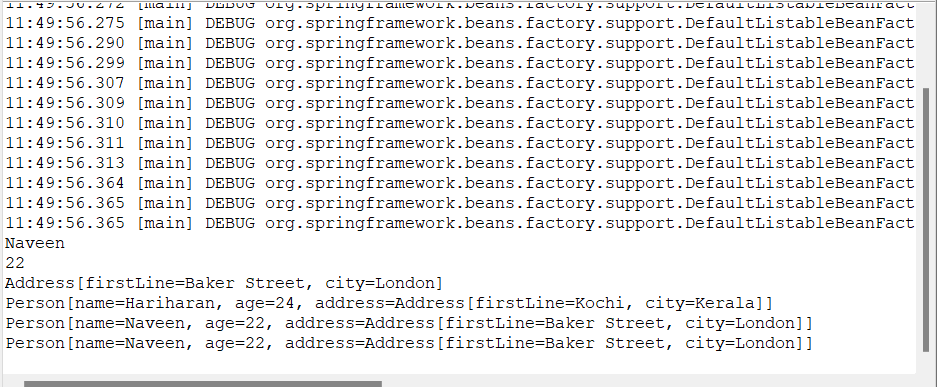
        System.out.println(context.getBean("person"));

        System.out.println(context.getBean("person2MethodCall"));

        System.out.println(context.getBean("person3Parameter"));

    }

}

**OUTPUT:**

We can reuse beans to create a new bean using method calls or using existing beans as a method parameter.