# EXPLORING SPRING FRAMEWORK BEAN AUTO WIRING - PRIMARY & QUALIFIER

**How can we list all beans managed by Spring Framework?**

- **getBeanDefinitionNames ():** this method will return a string array which contains the names of all beans defined in this registry, or an empty array if none defined.
- **getBeanDefinition ()**: It will return the BeanDefinition for the given name (never null). We should pass the **beanName** as **arguments**.
- **getBeanDefinitionCount ()**: Return the **number of beans** defined in the registry.

**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");

    }

}
```

**App02HelloWorldSpring.java**

```
package com.naveen.learnspringframework;

import java.util.Arrays;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        var context =
         new AnnotationConfigApplicationContext(HelloWorldConfiguration.class);

        Arrays.stream(context.getBeanDefinitionNames())
            .forEach(System.out::println);

    }

}
```

**OUTPUT:**

```
20:46:16.016 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.018 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.020 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.021 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.022 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.037 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.037 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
20:46:16.038 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
helloWorldConfiguration
name
age
person
person2MethodCall
person3Parameter
address2
```

**What if multiple matching beans are available?**

**HelloWorldConfiguration.java**

```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");

    }
}
```

**App02HelloWorldSpring.java**

```
package com.naveen.learnspringframework;

import java.util.Arrays;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        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"));

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

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

}
```

**OUTPUT:**

**Exception** in thread "main"
org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'com.naveen.learnspringframework.Person' available: expected single matching bean but found 3: person,person2MethodCall,person3Parameter

The problem is finding three objects with same class.

**How can we help spring framework to give priority to one of them?**

- In Spring Framework, the **@Primary** annotation is used to indicate that a particular bean should be given preference when there are multiple beans of the same type available to be injected.
- When there are multiple beans of the same type, Spring will try to resolve the ambiguity by looking for a primary bean. If there is a primary bean, Spring will use that bean. If there is no primary bean, Spring will throw an exception.

**HelloWorldConfiguration.java**

```java
package com.naveen.learnspringframework;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

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
    @Primary
    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");

    }
}
```

**App02HelloWorldSpring.java**

```
package com.naveen.learnspringframework;

import java.util.Arrays;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App02HelloWorldSpring {

    public static void main(String[] args) {

        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"));

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

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

    }

}
```

**OUTPUT:**

```
21:27:05.628 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
21:27:05.629 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
21:27:05.631 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
21:27:05.645 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
21:27:05.647 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
21:27:05.647 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
Naveen
22
Address[firstLine=Baker Street, city=London]
Person[name=Hariharan, age=24, address=Address[firstLine=Kochi, city=Kerala]]
Person[name=Naveen, age=22, address=Address[firstLine=Baker Street, city=London]]
Person[name=Naveen, age=22, address=Address[firstLine=Baker Street, city=London]]
Person[name=Hariharan, age=24, address=Address[firstLine=Kochi, city=Kerala]]
Address[firstLine=Baker Street, city=London]
```

- In Spring Framework, the **@Qualifier** annotation is used to resolve ambiguity when there are multiple beans of the same type available to be injected.
- When there are multiple beans of the same type, Spring will try to resolve the ambiguity by using the name of the bean as the qualifier. However, this approach can be problematic if the name of the bean changes, or if there are multiple beans with the same name.
- The **@Qualifier** annotation provides a more flexible way to specify which bean should be injected, by allowing you to specify a custom qualifier. You can use any string as a qualifier and annotate the injection point with the **@Qualifier** annotation to specify which bean to use.