Enum as Bean

In Spring, you can register an enum as a Spring bean by creating a configuration class and defining a method annotated with `@Bean` that returns the enum instance. Here's an example:

Let's say you have an enum called `MyEnum`:

```java
public enum MyEnum {
    VALUE1,
    VALUE2,
    VALUE3
}
```

You can register this enum as a Spring bean like this:

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MyEnumConfig {

    @Bean
    public MyEnum myEnumBean() {
        return MyEnum.VALUE1;
    }
}
```

In this configuration class:

- `MyEnumConfig` is annotated with `@Configuration` to indicate that it contains bean definitions.
- The `myEnumBean()` method is annotated with `@Bean`, which tells Spring to register the returned enum instance as a bean in the Spring application context.

Now, you can inject this enum bean into other Spring components by autowiring it:

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

```
@Component
public class MyComponent {

    private MyEnum myEnum;

    @Autowired
    public MyComponent(MyEnum myEnum) {
        this.myEnum = myEnum;
    }

    // Other methods that use myEnum
}
```

In this example, the `MyComponent` class is a Spring-managed component, and the `MyEnum` bean is injected into it via constructor injection. Spring automatically resolves the `MyEnum` bean from the application context and injects it into the `MyComponent` instance.