**🧱 Prototype**

When a bean has a prototype scope, it means th**at a new instance of the bean is created each time it is requested**. Unlike the singleton scope, where a single instance of the bean is created and reused, in the prototype scope a new instance of the bean is created on each request.

@Component  
@Scope("prototype")  
public class PrototypeBean {  
   
}

Here are some key features of the prototype scope in Spring:

* **New instance on each request**: Each time a bean with “prototype” scope is requested, a new instance of the bean is created. This means that each time the bean is injected or fetched, a new instance will be fetched.
* **No state sharing**: Bean instances with prototype scope do not share their state with other bean instances. Each instance has its own independent state and any modification to one bean instance will not affect other instances.
* **Unmanaged lifecycle**: Unlike other scopes such as singleton or session, bean instances with “prototype” scope are not managed completely by the Spring container. This means that Spring does not take care of destroying prototype bean instances when they are no longer in use. It is the responsibility of the developer to manage the lifetime of prototype bean instances and release resources appropriately.
* **Greater configuration flexibility**: The prototype scope is more flexible in terms of configuration, as it allows multiple instances of the same bean to be created with different configurations or parameters at runtime.
* **Higher resource consumption**: Because a new instance of the bean is created for each request, the use of prototype scoped beans may result in higher resource consumption compared to other scopes such as singleton or request, as more instances are created and destroyed.
* **Suitable for objects with changing state**: The prototype scope is suitable for beans whose state changes frequently and there is no need to share state between different instances.

It is important to note that prototype scoping in Spring can be useful in situations where you need a **new instance of the bean on every request**, and you want to have **more flexibility in state and concurrency management**. However, you should also be aware of the**increased computational cost associated** with creating new instances on each request.

*In contrast to the other scopes, Spring does not manage the complete lifecycle of a prototype bean. The container instantiates, configures, and otherwise assembles a prototype object and hands it to the client, with no further record of that prototype instance. Thus, although initialization lifecycle callback methods are called on all objects regardless of scope, in the case of prototypes, configured destruction lifecycle callbacks are not called. The client code must clean up prototype-scoped objects and release expensive resources that the prototype beans hold. To get the Spring container to release resources held by prototype-scoped beans, try using a custom bean post-processor, which holds a reference to beans that need to be cleaned up. (*[*https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-factory-scopes-prototype*](https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-factory-scopes-prototype)*)*

As a rule, you should use the prototype scope for all stateful beans and the singleton scope for stateless beans.