java spring создание prototype scoup

###### **ChatGPT сказал:**

ChatGPT

В Spring контекст предоставляет разные типы жизненных циклов для бинов, среди которых *singleton* и *prototype* — наиболее часто используемые. Если бин настроен как *singleton*, то Spring создаёт только один экземпляр на всё приложение. А вот если вам нужен новый экземпляр бина при каждом его запросе, стоит использовать *prototype* scope.

### **Настройка Prototype Scope**

Чтобы указать, что бин должен быть с жизненным циклом *prototype*, вам нужно отметить его аннотацией @Scope("prototype"). Обычно эта аннотация используется вместе с @Component или @Bean. Пример:

java

Копировать код

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

@Component

@Scope("prototype")

public class PrototypeBean {

public PrototypeBean() {

System.out.println("Создание нового экземпляра PrototypeBean");

}

// Дополнительные методы и логика

}

### **Как работает Prototype Scope**

В отличие от *singleton*, бин с *prototype* создается каждый раз при запросе. Когда вы запрашиваете этот бин у контекста Spring, он создает новый экземпляр:

java

Копировать код

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class MyService {

@Autowired

private PrototypeBean prototypeBean;

public void performTask() {

System.out.println(prototypeBean);

// Выполнение логики с использованием PrototypeBean

}

}

Если вы вызовете метод performTask() несколько раз, то увидите, что каждый раз будет создаваться новый экземпляр PrototypeBean.

### **Примечания**

1. **Контекст управляет только созданием:** В prototype scope Spring создаёт бин, но не управляет его дальнейшим жизненным циклом. Вы должны самостоятельно позаботиться об уничтожении этих объектов.
2. **Использование с другими бинами:** Если бин с singleton scope зависит от prototype-бина, может возникнуть ситуация, когда prototype-бин будет создан только один раз. Чтобы этого избежать, можно использовать аннотацию @Lookup, которая заставит Spring создавать новый экземпляр при каждом запросе prototype-бина.

4o

##### **Вы сказали:**

подробнее

###### **ChatGPT сказал:**

ChatGPT

В Spring фреймворке у каждого бина есть так называемый *scope*, который определяет его жизненный цикл и область действия. Два наиболее часто используемых типа области действия — это *singleton* и *prototype*. В то время как singleton создает один экземпляр на все приложение, prototype создает новый экземпляр каждый раз, когда он запрашивается из контейнера Spring. Рассмотрим подробнее, как создать и использовать prototype scope в Java Spring, включая более сложные случаи использования.

### **Как работает Prototype Scope**

Когда бин объявлен с prototype scope, Spring создаёт новый экземпляр бина каждый раз, когда он запрашивается у контейнера. Это означает, что prototype-бин **не** является кэшируемым и не управляется Spring после его создания (то есть Spring не отслеживает и не управляет его уничтожением).

#### **Пример создания Prototype Scope**

Для настройки prototype-бина нужно использовать аннотацию @Scope("prototype") совместно с аннотацией @Component, @Service, @Controller или @Bean. Вот пример создания prototype-бина:

java

Копировать код

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

@Component

@Scope("prototype")

public class PrototypeBean {

public PrototypeBean() {

System.out.println("Создан новый экземпляр PrototypeBean");

}

public void doSomething() {

System.out.println("Выполняется работа с PrototypeBean");

}

}

#### **Запрос нового экземпляра**

Если prototype-бин внедрён в другой бин с singleton scope, то вы можете столкнуться с проблемой, так как Spring внедрит только один экземпляр prototype-бина. Вот пример того, как можно его использовать:

java

Копировать код

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class SingletonService {

@Autowired

private PrototypeBean prototypeBean;

public void usePrototype() {

prototypeBean.doSomething();

}

}

В данном случае, если вызвать метод usePrototype() несколько раз, будет использован один и тот же экземпляр PrototypeBean. Это происходит, потому что SingletonService создается один раз, и prototype-бин инициализируется один раз при создании SingletonService.

### **Решение проблемы зависимости Singleton от Prototype**

Чтобы каждый раз получать новый экземпляр PrototypeBean, можно использовать следующие подходы:

#### **1. Использование @Lookup**

Аннотация @Lookup заставляет Spring выполнять поиск prototype-бина каждый раз, когда он нужен. Вы добавляете эту аннотацию к методу, который возвращает prototype-бин.

java

Копировать код

import org.springframework.beans.factory.annotation.Lookup;

import org.springframework.stereotype.Service;

@Service

public class SingletonService {

@Lookup

public PrototypeBean getPrototypeBean() {

// Spring сам позаботится о возвращении нового экземпляра PrototypeBean

return null; // Этот метод не будет вызван напрямую, Spring подменит его реализацию.

}

public void usePrototype() {

PrototypeBean prototypeBean = getPrototypeBean();

prototypeBean.doSomething();

}

}

Теперь при каждом вызове метода usePrototype() создаётся новый экземпляр PrototypeBean.

#### **2. Использование ObjectFactory**

Другой способ получить новый экземпляр prototype-бина — использовать ObjectFactory. Это удобный интерфейс, который предоставляет метод getObject() для создания нового экземпляра:

java

Копировать код

import org.springframework.beans.factory.ObjectFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class SingletonService {

@Autowired

private ObjectFactory<PrototypeBean> prototypeBeanFactory;

public void usePrototype() {

PrototypeBean prototypeBean = prototypeBeanFactory.getObject();

prototypeBean.doSomething();

}

}

В этом случае Spring создаст новый экземпляр при каждом вызове метода getObject().

#### **3. Использование Provider (Javax или Spring)**

Если вы работаете с Javax (например, в контексте Spring Boot), можно также использовать Provider, который предоставляет функциональность, аналогичную ObjectFactory.

java

Копировать код

import org.springframework.stereotype.Service;

import javax.inject.Provider;

import org.springframework.beans.factory.annotation.Autowired;

@Service

public class SingletonService {

@Autowired

private Provider<PrototypeBean> prototypeBeanProvider;

public void usePrototype() {

PrototypeBean prototypeBean = prototypeBeanProvider.get();

prototypeBean.doSomething();

}

}

### **Примечания**

* **Управление жизненным циклом:** В отличие от singleton-бинов, Spring не управляет жизненным циклом prototype-бинов после их создания. Это значит, что Spring не будет выполнять их destroy метод (если он есть). Если вам нужно очистить ресурсы после использования prototype-бина, это необходимо делать вручную.
* **Performance:** Учитывайте, что каждый раз при запросе нового экземпляра prototype-бина будет создаваться новый объект, что может сказываться на производительности. Использование prototype-бинов целесообразно в тех случаях, когда необходима их уникальность, например, при работе с транзакциями или специфичными пользователями.
* **Совместное использование с другими типами scope:** Иногда может возникнуть необходимость использовать бины с различными scope в одном и том же классе. В таких случаях использование @Lookup, ObjectFactory или Provider может помочь создать более гибкую конфигурацию, где каждый вызов создаёт новый экземпляр бина с нужным scope.

Использование prototype scope в Spring позволяет создавать объекты по требованию и избегать повторного использования объектов в рамках одного singleton, предоставляя более гибкую архитектуру для приложений с динамическими требованиями.

4o

ChatGPT может допускать ошибки. Рекомендуем проверять важную информацию.