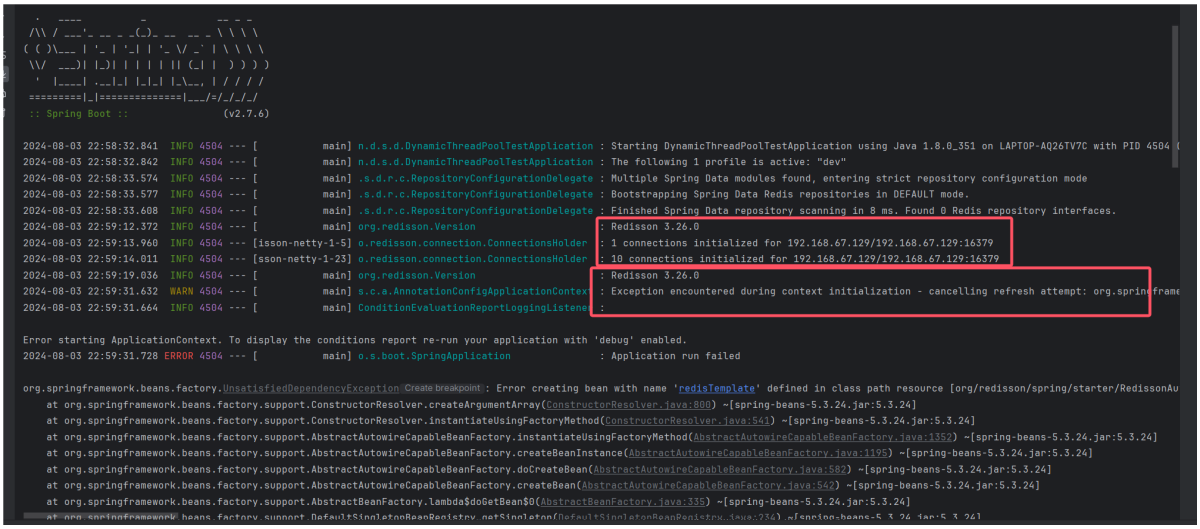
BUG描述

做完 `第4节：线程池数据上报（Redis 注册中心）` 后，运行项目报错如下

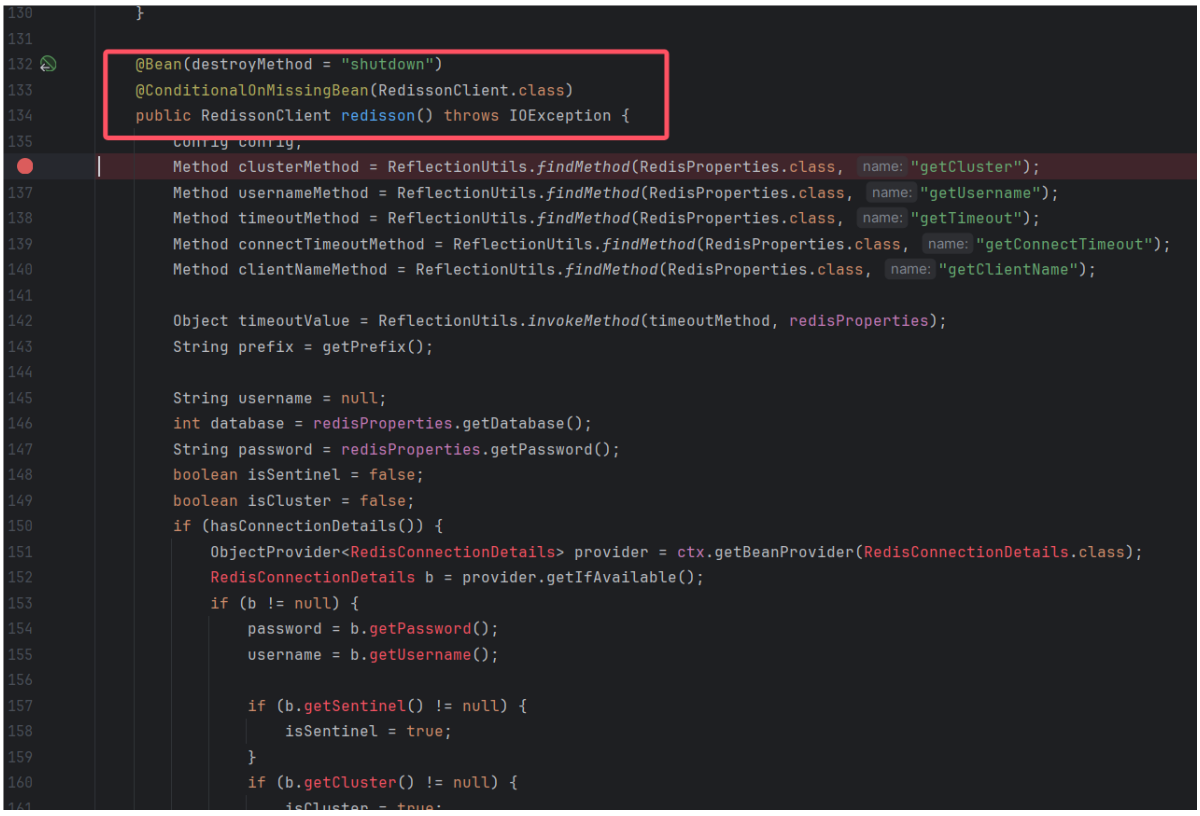


```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.6)

2024-08-03 22:58:32.841  INFO 4504 --- [           main] n.d.s.d.DynamicThreadPoolTestApplication : Starting DynamicThreadPoolTestApplication using Java 1.8.0_351 on LAPTOP-AQ26TV7C with PID 4504 (
2024-08-03 22:58:32.842  INFO 4504 --- [           main] n.d.s.d.DynamicThreadPoolTestApplication : The following 1 profile is active: "dev"
2024-08-03 22:58:33.574  INFO 4504 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode
2024-08-03 22:58:33.577  INFO 4504 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Redis repositories in DEFAULT mode.
2024-08-03 22:58:33.608  INFO 4504 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 8 ms. Found 0 Redis repository interfaces.
2024-08-03 22:59:12.372  INFO 4504 --- [           main] org.redisson.Version                     : Redisson 3.26.0
2024-08-03 22:59:13.960  INFO 4504 --- [sson-netty-1-5] o.redisson.connection.ConnectionsHolder  : 1 connections initialized for 192.168.67.129/192.168.67.129:16379
2024-08-03 22:59:14.011  INFO 4504 --- [sson-netty-1-23] o.redisson.connection.ConnectionsHolder : 10 connections initialized for 192.168.67.129/192.168.67.129:16379
2024-08-03 22:59:19.036  INFO 4504 --- [           main] org.redisson.Version                     : Redisson 3.26.0
2024-08-03 22:59:31.632  WARN 4504 --- [           main] s.c.a.AnnotationConfigApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframe
2024-08-03 22:59:31.664  INFO 4504 --- [           main] ConditionEvaluationReportLoggingListener :

Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.
2024-08-03 22:59:31.728 ERROR 4504 --- [           main] o.s.boot.SpringApplication               : Application run failed

org.springframework.beans.factory.UnsatisfiedDependencyException Create breakpoint : Error creating bean with name 'redisTemplate' defined in class path resource [org/redisson/spring/starter/RedissonAu
    at org.springframework.beans.factory.support.ConstructorResolver.createArgumentArray(ConstructorResolver.java:800) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.ConstructorResolver.instantiateUsingFactoryMethod(ConstructorResolver.java:541) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateUsingFactoryMethod(AbstractAutowireCapableBeanFactory.java:1352) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBeanInstance(AbstractAutowireCapableBeanFactory.java:1195) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:582) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:542) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:335) ~[spring-beans-5.3.24.jar:5.3.24]
    at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:234) ~[spring-beans-5.3.24.jar:5.3.24]
```

我排查后发现，redissonClient 这个Bean注入了`两次`， 如上图所示

一次是我们在自己的项目中 定义的，另一个是官方的

官方定义 redissonClient bean是这样的 的方法是这样的



```
    }

    @Bean(destroyMethod = "shutdown")
    @ConditionalOnMissingBean(RedissonClient.class)
    public RedissonClient redisson() throws IOException {
        Config config;
        Method clusterMethod = ReflectionUtils.findMethod(RedisProperties.class, name: "getCluster");
        Method usernameMethod = ReflectionUtils.findMethod(RedisProperties.class, name: "getUsername");
        Method timeoutMethod = ReflectionUtils.findMethod(RedisProperties.class, name: "getTimeout");
        Method connectTimeoutMethod = ReflectionUtils.findMethod(RedisProperties.class, name: "getConnectTimeout");
        Method clientNameMethod = ReflectionUtils.findMethod(RedisProperties.class, name: "getClientName");

        Object timeoutValue = ReflectionUtils.invokeMethod(timeoutMethod, redisProperties);
        String prefix = getPrefix();

        String username = null;
        int database = redisProperties.getDatabase();
        String password = redisProperties.getPassword();
        boolean isSentinel = false;
        boolean isCluster = false;
        if (hasConnectionDetails()) {
            ObjectProvider<RedisConnectionDetails> provider = ctx.getBeanProvider(RedisConnectionDetails.class);
            RedisConnectionDetails b = provider.getIfAvailable();
            if (b != null) {
                password = b.getPassword();
                username = b.getUsername();

                if (b.getSentinel() != null) {
                    isSentinel = true;
                }
                if (b.getCluster() != null) {
                    isCluster = true;
```

官方定义的方法上有个注解是 `@ConditionalOnMissingBean(RedissonClient.class)`

我的理解是，只有不存在 RedissonClient.class 组件的时候才会启用这个Bean

但是，上面我们已经已经注册了RedissonClient了，这里却还是会执行

我一开始觉得可能是Bean的加载顺序问题，但是我在下断点debug时，也确实是先走的我自己定义 RedissonClient 的方法

所以不知道怎么解决了

附加信息

DynamicThreadPoolAutoConfig.java (自动装配的入口类)

```java
package site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.config;

import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang.StringUtils;
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.Config;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableScheduling;
import site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.config.properties.DynamicThreadPoolAutoProperties;
import site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.config.properties.DynamicThreadPoolRegistryRedisAutoProperties;
import site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.registry.IRegistry;
import site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.registry.redis.RedisRegistry;
import site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.service.impl.DynamicThreadPoolService;

import java.util.Map;
import java.util.concurrent.ThreadPoolExecutor;

/**
 * 自动配置入口
 */
@Slf4j
@Configuration
@EnableConfigurationProperties({
        DynamicThreadPoolRegistryRedisAutoProperties.class,
})
@EnableScheduling
public class DynamicThreadPoolAutoConfig {

```

```
    @Bean
    public RedissonClient redissonClient(DynamicThreadPoolRegistryRedisAutoProperties properties) {
        Config config = new Config();
        config.setCodec(JsonJacksonCodec.INSTANCE);
        config.useSingleServer()
                .setAddress(String.format("redis://%s:%d", properties.getHost(), properties.getPort()))
                .setPassword(properties.getPassword())
                .setDatabase(properties.getDatabase())
                .setConnectionPoolSize(properties.getConnectionPoolSize())
                .setConnectionMinimumIdleSize(properties.getConnectionMinimumIdleSize())
                .setIdleConnectionTimeout(properties.getIdleConnectionTimeout())
                .setConnectTimeout(properties.getConnectTimeout())
                .setRetryAttempts(properties.getRetryAttempts())
                .setRetryInterval(properties.getRetryInterval())
                .setKeepAlive(properties.getKeepAlive());
        return Redisson.create(config);
    }


    @Bean
    public IRegistry redisRegistry(RedissonClient redissonClient) {
        return new RedisRegistry(redissonClient);
    }

    @Bean
    public DynamicThreadPoolService dynamicThreadPoolService(
            ApplicationContext applicationContext,
            Map<String, ThreadPoolExecutor> threadPoolExecutorMap,
            RedissonClient redissonClient
    ) {
        String applicationName = applicationContext.getEnvironment().getProperty("spring.application.name");
        if (StringUtils.isBlank(applicationName)) {
            log.warn("动态线程池启动提示。SpringBoot 应用未配置应用名(spring.application.name)");
        }

        return new DynamicThreadPoolService(applicationName, threadPoolExecutorMap);
    }
}

```

DynamicThreadPoolRegistryRedisAutoProperties.java (配置类)

```
package site.notcoder.dtp.sdk.dynamicthreadpoolspringbootstarter.config.properties;

import lombok.AllArgsConstructor;
```

```java
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.boot.context.properties.ConfigurationProperties;

/**
 * 线程池注册中心 Redis 配置
 */

@Data
@NoArgsConstructor
@AllArgsConstructor
@ConfigurationProperties("dynamic-thread-pool.registry.redis")
public class DynamicThreadPoolRegistryRedisAutoProperties {

    /** Redis地址 */
    private String host;
    /** Redis端口，默认6379 */
    private Integer port = 6379;
    /** Redis数据库，默认是0 */
    private Integer database = 0;
    /** Redis 密码 */
    private String password;
    /** 设置连接池的大小，默认为64 */
    private int connectionPoolSize = 64;
    /** 设置连接池的最小空闲连接数，默认为10 */
    private int connectionMinimumIdleSize = 10;
    /** 设置连接的最大空闲时间（单位：毫秒），超过该时间的空闲连接将被关闭，默认为10000 */
    private int idleConnectionTimeout = 10000;
    /** 设置连接超时时间（单位：毫秒），默认为10000 */
    private int connectTimeout = 10000;
    /** 设置连接重试次数，默认为3 */
    private int retryAttempts = 3;
    /** 设置连接重试的间隔时间（单位：毫秒），默认为1000 */
    private int retryInterval = 1000;
    /** 设置定期检查连接是否可用的时间间隔（单位：毫秒），默认为0，表示不进行定期检查 */
    private int pingInterval = 0;
    /** 设置是否保持长连接，默认为true */
    private Boolean keepAlive = true;
}

```

application-dev.yml (dynamic-thread-pool-tset)

```yaml
server:
  port: 8091

# 线程池配置
thread:
  pool:
    executor:
      config:
        core-pool-size: 20
        max-pool-size: 50
        keep-alive-time: 5000
```

```
        block-queue-size: 5000
        policy: CallerRunsPolicy

# 动态线程池配置
dynamic-thread-pool:
  registry:
    redis:
      host: 192.168.67.129
      port: 16379
```