手动注册bean doRegisterBean

\*/

<T> void doRegisterBean(Class<T> **annotatedClass**, *@Nullable* Supplier<T> **instanceSupplier**, *@Nullable* String **name**,

*@Nullable* Class<? extends Annotation>[] **qualifiers**, BeanDefinitionCustomizer... **definitionCustomizers**) {

AnnotatedGenericBeanDefinition abd = new AnnotatedGenericBeanDefinition(**annotatedClass**);

if (this.conditionEvaluator.shouldSkip(abd.getMetadata())) {

return;

}

abd.setInstanceSupplier(**instanceSupplier**);

ScopeMetadata scopeMetadata = this.scopeMetadataResolver.resolveScopeMetadata(abd);

abd.setScope(scopeMetadata.getScopeName());

String beanName = (**name** != null ? **name** : this.beanNameGenerator.generateBeanName(abd, this.registry));

AnnotationConfigUtils.*processCommonDefinitionAnnotations*(abd);

if (**qualifiers** != null) {

for (Class<? extends Annotation> qualifier : **qualifiers**) {

if (*Primary*.class == qualifier) {

abd.setPrimary(true);

}

else if (*Lazy*.class == qualifier) {

abd.setLazyInit(true);

}

else {

abd.addQualifier(new AutowireCandidateQualifier(qualifier));

}

}

}

for (BeanDefinitionCustomizer customizer : **definitionCustomizers**) {

customizer.customize(abd);

}

BeanDefinitionHolder definitionHolder = new BeanDefinitionHolder(abd, beanName);

definitionHolder = AnnotationConfigUtils.*applyScopedProxyMode*(scopeMetadata, definitionHolder, this.registry);

BeanDefinitionReaderUtils.*registerBeanDefinition*(definitionHolder, this.registry);

}