思路&编码实现讲解

伍文彬 / 2021年2月15日

题目分析

针对此道题，可以分析得到：需要在 Tomcat 启动完成之后需要维护一个单例池（类似维护一个 IOC 容器），并且我们所有的 Bean对象 维护、管理、使用都在这个池子里完成。

那么如何在启动完成之后，就能够形成一个包含所有需要维护的对象的单例池呢?

根据学过的知识知道，javax.servlet 包下有个监听器类：ServletContextListener 可以实现针对容器启动、销毁的动作的监听，基于此，我可以在监听容器启动处的方法完成：以上提到的对所有需要注入的对象 完成创建以及后续的使用和维护管理。

实现思路

1. 实现一个我们自己的容器监听器，我把它取名为：MyWebApplication，并继承 Tomcat 的 ServletContextListener 接口，来完成对 Bean 的解析和 Bean 池的维护。
2. 在我们自己实现的容器监听器 MyWebApplication 中维护一个对象：BeanFactory，其中包含这么几个对象和方法：
    - singletonBeanPool（暂定为 Map 类型）：负责对所有的对象进行使用管理维护的对象池
    - getBean0(String name)：暴露给外部使用的原始接口方法，获取池中的对象，以 name 为参数
    - getBean0(Class<?> classType)：暴露给外部使用的原始接口方法，获取池中的对象，以 class 对象为参数
    - containsBean：暴露给外部使用的原始接口方法，判断对象池中是否已包含该对象
    - 等等...
3. 到这里为止，还缺少了启动时完成对 bean 对象的解析以及管理维护的操作，所以还需要在 BeanFactory 中完成对象注入到对象池中的操作，那么为了防止 BeanFactory 代码过于臃肿，我在此处又定义了一个抽象类 AbstractBeanFactory 完成一些基础通用操作，那么这里我就把 BeanFactory 定义为一个顶级父类接口，用于规范一些通用的对外方法即可，具体操作如下几点：
    - 简单的实现接口的一些基础方法：getBean0、containsBean等（此处是指直接从池中拿取，同步锁判断put等，不做额外操作）
    - 在 AbstractBeanFactory 中定义好 singletonBeanPool 对象，作为即将被解析的 bean 对象的存储介质
    - 定义好存储介质（对象单例池）之后，初始化解析 bean 对象，并把得到的对象放入至 singletonBeanPool （对象单例池）中
4. 在以上 3 点中完成了绝大部分的操作，但是还有一个操作：对象增强的操作没有实现（AOP操作，事务控制等），那么基于类似的原因（防止代码过于臃肿，也同时让不同层级的代码各司其职），我又定一个给全局使用的 MyWebApplicationContext 上下文对象，并继承之前的 AbstractBeanFactory（为了继承所有父类和接口的功能和属性），同时在本对象中定一个暴露给全局使用的增强 getBean 方法，完成对对象的从池中的拿取等操作，并自动判断是否需要被代理增强（如事务增强或者其他 AOP 逻辑等），具体的操作：

    - 定义全新的 getBean 方法，实现自动判断的增强代理逻辑
    - 要实现以上的方法，那么我们就要完成一个代理工厂：ProxyFactory（此处就拿事务增强来举例，具体看代码）
5. 那么以上都完成之后，我们就可以在实现层（web / servlet层）调用我们池中的对象和方法来完成相关操作了

编码准备

1. 首先就需要定义一些属性配置注解，如下：
    - 标识配置类对象，使用 @Configuration 注解（由 listener 启动时的全局参数告知容器），用于解析其他的配置信息
    - 数据库链接属性：作业示例中放到配置文件中（使用 @PropertySource 注解标识文件名，并由程序启动时读取文件并注入）
    - 标识哪些 bean 对象需要被维护到对象池 singletonBeanPool 中，使用 @Component 注解
    - 标识配置类中的需要被注入的方法对象，使用 @Bean 注解
    - 标识配置文件中的属性，以便直接从池中拿取，这里和 bean 对象处理方式不太一样，所有的配置都是存在一个 Propertie 对象中的，再把 Properties 对象方法对象池中来管理维护，池中配置属性使用 @PropValue 来标识
    - 标识需要扫描的包路径，使用 @ComponentScan 注解
    - 标识是否需要开启事务，使用 @EnableTransactionManagement 注解
    - 标识对象中需要注入的属性，使用 @Autowired 注解
2. 配置类对象：MyWebConfig

    > 同时在 web.xml 中配置，告知容器的启动参数
    >
    > 以及配置下 listener 为我们自定义的 MyWebApplication

3. 其他一些类（直接引用课程中老师授课的源码），如下：
    - ProxyFactory（代理类工厂，针对事务控制的开启/关闭做了一些判断改造，大体上还是差不多，具体看代码）
    - TransferServlet（获取 service 方式改为我们自定义的 getBean，其余相同）
    - TransactionManager 改为接口+默认实现类（DefaultTransactionManager）的模式
    - TestException：为了测试回滚的自定义异常类