<beans>的profile属性

## profile配置

我们可以把开发环境和生产环境的配置分别配置到单独的文件中，然后通过配置beans元素的profile属性来标识当前文件是哪个环境的配置，举个例子：

|  |
| --- |
| <beans profile=*"development"*  xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xsi:schemaLocation=*"..."*>    <jdbc:embedded-database id=*"dataSource"*>  <jdbc:script location=*"classpath:com/bank/config/sql/schema.sql"*/>  <jdbc:script location=*"classpath:com/bank/config/sql/test-data.sql"*/>  </jdbc:embedded-database>  </beans> |
| <beans profile=*"production"*  xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:jee=*"http://www.springframework.org/schema/jee"*  xsi:schemaLocation=*"..."*>    <jee:jndi-lookup id=*"dataSource"* jndi-name=*"java:comp/env/jdbc/datasource"*/>  </beans> |

另外，由于beans元素可以嵌套，我们也可以在一个配置文件中使用两个beans子元素来区分开发环境和生产环境的配置：

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xmlns:jee=*"http://www.springframework.org/schema/jee"*  xsi:schemaLocation=*"..."*>    <beans profile=*"development"*>  <jdbc:embedded-database id=*"dataSource"*>  <jdbc:script location=*"classpath:com/bank/config/sql/schema.sql"*/>  <jdbc:script location=*"classpath:com/bank/config/sql/test-data.sql"*/>  </jdbc:embedded-database>  </beans>    <beans profile=*"production"*>  <jee:jndi-lookup id=*"dataSource"* jndi-name=*"java:comp/env/jdbc/datasource"*/>  </beans>  </beans> |

## 如何使用特定的 profile 呢？

Spring 在启动的过程中，会去寻找 “**spring.profiles.active**” 的属性值，并根据这个属性值来使用相应的profile。那怎么配置这个值呢？

Spring 会在这几个地方寻找 spring.profiles.active 的属性值：操作系统环境变量、JVM 系统变量、web.xml 中定义的参数、JNDI。最简单的方式莫过于在程序启动的时候指定：“-Dspring.profiles.active="profile1,profile2”（profile可以激活多个）。

当然，我们也可以通过代码的形式从 Environment 中设置 profile：

|  |
| --- |
| AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();  ctx.getEnvironment().setActiveProfiles("development");  ctx.register(SomeConfig.class, StandaloneDataConfig.class, JndiDataConfig.class);  ctx.refresh(); *// 重启* |

如果是 Spring Boot 的话更简单，我们一般会创建 application.properties、application-dev.properties、application-prod.properties 等文件，其中 application.properties 配置各个环境通用的配置，application-{profile}.properties 中配置特定环境的配置，然后在启动的时候指定 profile：“java -Dspring.profiles.active=prod -jar JavaDoop.jar”。

如果是单元测试中使用的话，在测试类中使用 @ActiveProfiles 指定，这里就不展开了。