02 高级装配

C-2 创建: 张林伟, 最后修改: 张林伟 2018-09-20 10:07

环境与 profile

不同环境可能需要不同的bean的配置，解决方案：

1. 使用 Maven 的 profiles，在构建阶段确定要将哪一个配置文件编译到可部署的应用中

点击展开内容

代码块 XML

```xml
<profiles>
    <profile>
        <id>dev</id>
        <build>
            <resources>
                <resource>
                    <directory>src/main/profiles/dev</directory>
                </resource>
                <resource>
                    <directory>src/main/resources</directory>
                </resource>
            </resources>
        </build>
    </profile>
    <profile>
        <id>qa</id>
        <build>
            <resources>
                <resource>
                    <directory>src/main/profiles/qa</directory>
                </resource>
                <resource>
                    <directory>src/main/resources</directory>
                </resource>
            </resources>
        </build>
    </profile>
    <profile>
        <id>local</id>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
        <build>
            <resources>
                <resource>
                    <directory>src/main/profiles/local</directory>
                </resource>
                <resource>
                    <directory>src/main/resources</directory>
                </resource>
            </resources>
        </build>
    </profile>
    <profile>
        <id>staging</id>
        <build>
            <resources>
                <resource>
                    <directory>src/main/profiles/staging</directory>
                </resource>
                <resource>
                    <directory>src/main/resources</directory>
```

```
                    </resource>
                </resources>
            </build>
        </profile>
        <profile>
            <id>online</id>
            <build>
                <resources>
                    <resource>
                        <directory>src/main/profiles/online</directory>
                    </resource>
                    <resource>
                        <directory>src/main/resources</directory>
                    </resource>
                </resources>
            </build>
        </profile>
    </profiles>
```

2. Spring 提供的解决方案：@Profile 注解或者 XML配置 profile

   不同于 Maven 在构建时决策，Spring 是等到运行时决定该创建哪些 bean 和不创建哪些 bean

Spring 解决方案

- 使用 @Profile 注解

  点击展开内容

  代码块 Java

```java
@Configuration
@Profile("prod") //应用于该类下所有bean
public class ProductionProfileConfig {
    @Bean
    public DataSource dataSource() {
        ...
    }
}
```

  代码块 Java

```java
@Configuration
public class ProductionProfileConfig {
    @Bean
    @Profile("dev") //应用于该bean
    public DataSource dataSource() {
        ...
    }
}
```

- 使用 XML 配置

  点击展开内容

  通过<beans>元素的 profile 属性，在 XML 中配置 profile bean

  代码块 XML

```xml
<beans profile="qa">
  ...
</beans>
```

激活 profile

Spring 依赖两个属性：**spring.profiles.active** 和 **spring.profiles.default**。

如果设置了 spring.profiles.active，则按照它的值来确定激活哪个 profile，如果没有则按照 spring.profiles.default 来确定，如果都没有设置，则没有激活的 profile，因此只能创建那行没有定义在 profile 中的bean。

可以设置多个 profile，用逗号隔开。

设置方式：

- 作为 DispatcherServlet 的初始化参数
- 作为 Web 应用的上下文参数
- 作为 JNDI 条目
- 作为环境变量
- 作为 JVM 的系统属性
- 在集成测试类上，使用 @ActiveProfiles注解设置

点击展开内容

代码块 Java

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes={PersistenceTestConfig.class})
@ActiveProfiles("dev")
public class PersistenceTest {
  ...
}
```

条件化的 bean

Spring4 引入的 @Conditional 注解可以帮你条件化地创建 bean。

代码块 Java

```java
@Bean
@Conditional(MagicExistsCondition.class)
public MagicBean magicBean() {
    return new MagicBean();
}

// 条件 MagicExistsCondition 类
public class MagicExistsCondition implements Condition {
    public boolean matches(ConditionContext context, AnnotatedTypeMetadata metadata) {
        Environment env = context.getEnvironment();
        return env.containsProperty("magic");
    }
}

// Condition 接口
@FunctionalInterface
public interface Condition {
    boolean matches(ConditionContext var1, AnnotatedTypeMetadata var2);
}

// ConditionContext 接口
public interface ConditionContext {
    BeanDefinitionRegistry getRegistry();  // 检查 bean 定义
    @Nullable
    ConfigurableListableBeanFactory getBeanFactory();  // 检查 bean 是否存在，甚至探查 bean 的属性
    Environment getEnvironment();  // 检查环境变量是否存在以及值是什么
    ResourceLoader getResourceLoader();  // 读取并探查 ResourceLoader 所加载的资源
    @Nullable
    ClassLoader getClassLoader();  //加载并检查类是否存在
}

// AnnotatedTypeMetadata 接口
```

```java
public interface AnnotatedTypeMetadata {
  boolean isAnnotated(String annotationName);  // 判断带有@Bean注解的方法是不是还有其他特定注解
  @Nullable
  Map<String, Object> getAnnotationAttributes(String annotationName);
  @Nullable
  Map<String, Object> getAnnotationAttributes(String annotationName, boolean classValuesAsString);
  @Nullable
  MultiValueMap<String, Object> getAllAnnotationAttributes(String annotationName);
  @Nullable
  MultiValueMap<String, Object> getAllAnnotationAttributes(String annotationName, boolean classValuesAsString);
}
```

处理自动装配的歧义性

依赖注入