spring学习笔记-TestContext框架

|  |  |
| --- | --- |
|  | 🢂 内容概览 |
|  | Why：此文档用来做什么？它存在的意义是什么？为解决什么问题？   |  | | --- | |  |   What：当前包含了那些内容？   |  | | --- | |  |   How：此文档应如何参考？   |  | | --- | |  |   Who：此文档适用于那些人员阅读参考？   |  | | --- | |  |   Summary：摘要   |  | | --- | |  |   Reference：参考文献   |  | | --- | |  | |

目录

[1 TestContext框架 3](#_Toc463794184)

[1.1 使用定制TestExecutionListeners 3](#_Toc463794185)

[1.2 Context管理 4](#_Toc463794186)

[1.2.1 使用ApplicationContextAware注入context 4](#_Toc463794187)

[1.2.2 使用@Autowired注入context 5](#_Toc463794188)

[1.2.3 使用xml配置context 5](#_Toc463794189)

[1.2.4 使用注解类配置context 6](#_Toc463794190)

[1.2.5 使用initializers配置context 7](#_Toc463794191)

[1.2.6 配置context指定profile 7](#_Toc463794192)

[1.2.6.1 使用xml方式定义profile; 8](#_Toc463794193)

[1.2.6.2 使用基于Java的配置方式定义profile： 9](#_Toc463794194)

[1.2.6.3 使用抽象类避免重复加载 11](#_Toc463794195)

[1.2.7 加载WebAppliactionContext 11](#_Toc463794196)

[1.2.8 Context继承 11](#_Toc463794197)

[1.3 事务管理 12](#_Toc463794198)

[1.3.1 在事务中执行方法 12](#_Toc463794199)

[1.3.2 事务生命周期回调函数 12](#_Toc463794200)

[1.4 执行sql脚本 13](#_Toc463794201)

[1.4.1 以编程的方式执行sql脚本 13](#_Toc463794202)

[1.4.2 使用@Sql执行sql脚本 14](#_Toc463794203)

[1.5 测试支撑类 15](#_Toc463794204)

[1.5.1 SpringJUnit4ClassRunner 15](#_Toc463794205)

[1.5.2 Spring JUnit Rules 15](#_Toc463794206)

[1.5.3 Junit支撑类 16](#_Toc463794207)

[1.5.4 TestNG支撑类 16](#_Toc463794208)

# TestContext框架

1. TestContext类：封装测试上下文环境；
2. TestContextManager类：作为测试框架的入口，执行每个测试方法时，都会创建一个，用来管理TestContext，并通知TestExecutionListener监听器；
3. TestExecutionListener接口：监听TestContext变化；默认监听器如下（如果未指定监听器，则使用默认监听器）：

|  |  |
| --- | --- |
| **实现类** | **描述** |
| ServletTestExecutionListener | 给WebApplicationContext配置Servlet API； |
| DependencyInjectionTestExecutionListener | 为测试实例提供依赖注入； |
| DirtiesContextTestExecutionListener | 处理@DirtiesContext注解； |
| TransactionalTestExecutionListener | 给测试提供事务； |
| SqlScriptsTestExecutionListener | 执行@Sql标记的sql脚本； |

1. ContextLoader接口 & SmartContextLoader接口：用于给测试类加载ApplicationContext；
2. 实现类都在org.springframework.test.context.support包中；

|  |  |
| --- | --- |
| **实现类** | **描述** |
| DelegatingSmartContextLoader |  |
| WebDelegatingSmartContextLoader |  |
| AnnotationConfigContextLoader |  |
| AnnotationConfigWebContextLoader |  |
| GenericGroovyXmlContextLoader |  |
| GenericGroovyXmlWebContextLoader |  |
| GenericXmlContextLoader |  |
| GenericXmlWebContextLoader |  |
| GenericPropertiesContextLoader |  |

## 使用定制TestExecutionListeners

@TestExecutionListeners用于指定测试类需要用到的监听 ；

|  |
| --- |
|  |

为了避免自定义监听器和默认监听器重复，可以指定重复监听器合并：

|  |
| --- |
|  |

## Context管理

1. @ ContextConfiguration注解用来初始化ApplicationContext环境；
2. 当子类和父类都标注@ ContextConfiguration时，子类的配置将会重写父类相同的配置；

|  |
| --- |
|  |

### 使用ApplicationContextAware注入context

测试类默认不会自己获取ApplicationContext对象。可以通过实现ApplicationContextAware接口注入ApplicationContext上下文；如下两个抽象类实现了ApplicationContextAware接口，故测试类集成抽象类后，能够获得ApplicationContex对象；

org.springframework.test.context.junit4.AbstractJUnit4SpringContextTests;

org.springframework.test.context.testng.AbstractTestNGSpringContextTests;

|  |
| --- |
|  |

### 使用@Autowired注入context

依靠DependencyInjectionTestExecutionListener实现；

|  |
| --- |
|  |

注入WebApplicationContext：

|  |
| --- |
|  |

### 使用xml配置context

可以使用相对路径（相对于classpath）或者绝对路径（URL，如：classpath:，file:，http:）;

|  |
| --- |
|  |

或者简写：

|  |
| --- |
|  |

如果省略配置文件地址，则GenericXmlContextLoader和GenericXmlWebContextLoader会使用默认的约定查找配置文件，比如：测试类是com.example.MyTest，则加载配置文件默认为：classpath:com/example/MyTest-context.xml；

|  |
| --- |
|  |

### 使用注解类配置context

测试配置注解类满足如下条件之一，即可：

1. 注解类需要使用@Configuration修饰；
2. 注解类是一个组件（被标注为@Component，@Service，@Repository等）；
3. 通过JSR-330相关注解标注的类；
4. 包含@Bean方法的类

|  |
| --- |
|  |

如果省略classes属性，则AnnotationConfigContextLoader和AnnotationConfigWebContextLoader会使用默认的约定查找配置类，即查找测试类中使用@Configuration标注的静态内部类。

|  |
| --- |
|  |

### 使用initializers配置context

通过@ContextConfiguration的initializers属性指定ApplicationContextInitializer的实现类，用来初始化ConfigurableApplicationContext实例。

|  |
| --- |
|  |

### 配置context指定profile

#### 使用xml方式定义profile;

|  |
| --- |
|  |

|  |
| --- |
|  |

#### 使用基于Java的配置方式定义profile：

|  |
| --- |
|  |

|  |
| --- |
|  |

#### 使用抽象类避免重复加载

|  |
| --- |
|  |

### 加载WebAppliactionContext

|  |
| --- |
|  |

### Context继承

集成测试需要依赖一个ApplicationContext，有时候，ApplicationContext有继承关系，比如，在使用SpringMVC时，有一个通过ContextLoaderListener加载的根WebApplicationContext，还有一个通过DispatcherServlet加载的子WebApplicationContext。

|  |
| --- |
|  |

## 事务管理

### 在事务中执行方法

|  |
| --- |
|  |

### 事务生命周期回调函数

|  |
| --- |
|  |

## 执行sql脚本

### 以编程的方式执行sql脚本

1. org.springframework.jdbc.datasource.init.ScriptUtils：提供处理SQL脚本的静态方法，主要供内部使用；
2. org.springframework.jdbc.datasource.init.ResourceDatabasePopulator：提供简单的面向对象的API；
3. org.springframework.test.context.junit4.AbstractTransactionalJUnit4SpringContextTests；
4. org.springframework.test.context.testng.AbstractTransactionalTestNGSpringContextTests；

|  |
| --- |
|  |

### 使用@Sql执行sql脚本

1. @Sql即可以被标注在类上，也可以被标注在方法上，标注在方法上将覆盖类上相同的脚本；
2. 标注在类上，默认脚本路径：classpath:com/example/MyTest.sql；
3. 标注在方法上，默认脚本路劲：classpath:com/example/MyTest.testMethod.sql；

|  |
| --- |
|  |

执行多个脚本：

|  |
| --- |
|  |

设置在哪个阶段执行：

|  |
| --- |
|  |

执行脚本时，设置事务：

|  |
| --- |
|  |

## 测试支撑类

### SpringJUnit4ClassRunner

1. 实现了JUnit的运行器，能够支持在junit上整合spring的测试工具；
2. 如果运行器使用定制运行器或者其它第三方运行器，则需要使用Spring JUnit Rules方式整合spring的测试工具到junit；

### Spring JUnit Rules

1. org.springframework.test.context.junit4.rules.SpringClassRule：在类级别支持spring testContext框架；
2. org.springframework.test.context.junit4.rules.SpringMethodRule：在实例和方法级别支持spring testContext框架；

|  |
| --- |
|  |

### Junit支撑类

1. AbstractJUnit4SpringContextTests：抽象类，集成spring的TestContext框架和ApplicationContext；
2. AbstractTransactionalJUnit4SpringContextTests：对AbstractJUnit4SpringContextTests的扩展，增加的JDBC操作的API；

### TestNG支撑类

1. AbstractTestNGSpringContextTests；
2. AbstractTransactionalTestNGSpringContextTests；