**1**[**Maven中的-D（Properties属性）和-P（Profiles配置文件）**](http://www.cnblogs.com/EasonJim/p/6828748.html)

**-D代表（Properties属性）**

使用命令行设置属性**-D**的正确方法是：

mvn -DpropertyName=propertyValue clean package

* 如果**propertyName**不存在**pom.xml**，它将被设置。
* 如果**propertyName**已经存在**pom.xml**，其值将被作为参数传递的值覆盖**-D**。

要发送**多个变量**，请使用多个空格分隔符加**-D**：

mvn -DpropA=valueA -DpropB=valueB -DpropC=valueC clean package

例：

如果你的pom.xml如下：

<properties>

<theme>myDefaultTheme</theme>

</properties>

那么在这个执行过程中**mvn -Dtheme=halloween clean package**会覆盖**theme**的值，具有如下效果：

<properties>

<theme>halloween</theme>

</properties>

**2@AssertJ流式断言（可以执行完所有用例一次返回报告）**

Pom依赖：

1. <dependency>
2. <groupId>org.assertj</groupId>
3. <artifactId>assertj-core</artifactId>
4. *<!-- use 2.4.1 for Java 7 projects -->*
5. <version>2.4.1</version>
6. *<!-- <version>3.4.1</version> for java 8-->*
7. <scope>test</scope>
8. </dependency>

在test测试类引入import static org.assertj.core.api.Assertions.\*;

AseertJ: 号称流式断言。什么是流式，常见的断言器一条断言语句只能对实际值断言一个校验点，而AseertJ支持一条断言语句对实际值同时断言多个校验点，这样使得断言的语句更加简洁适合阅读。AseertJ还支持一次性执行所有断言，然后收集所有失败的断言一起反馈。当然除此之外AseertJ还有很多其他特性，可以参考官方文档慢慢挖掘。下面将举例说明一下AseertJ的优势：

@Test  
 public void testAssertJ(){  
  
 //断言集合，执行所有断言后，失败断言统一反馈。  
 SoftAssertions.assertSoftly(softly -> {  
 softly.assertThat(user.getAddress().equals("宁波"));  
 softly.assertThat(user.getMobile().equals("13868000000"));  
 softly.assertThat(user.getUserName().equals("测试"));  
 });  
  
 //流式断言  
 Assertions.assertThat(user.getMobile())  
 .isNotNull()  
 .startsWith("136")  
 .hasSize(11);

添加错误提示信息： as

Assertions.*assertThat*(2).as("检验是否相等").isEqualTo(3);

**2testng**

引用以下依赖

1<!--<dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>6.11</version>  
 <scope>test</scope>  
</dependency>-->  
<!--<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 <exclusions>  
 <exclusion>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-logging</artifactId>  
 </exclusion>  
 </exclusions>  
</dependency>-->

}

2新建测试类继承AbstractTestNGSpringContextTests

@SpringBootTest(classes = { MotanClient.class, RobotTaskComponent.class})  
public class TestClient extends AbstractTestNGSpringContextTests{  
 private static final Logger *logger* = LoggerFactory.*getLogger*(TestClient.class);