新平台使用技术相关说明

**文档历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 更新要点 | 日期 | 作者 |
| V1.0 | 1. 初版 | 2016-01-03 | 陈伟 |
| V1.0 | 1. 添加配置说明 | 2016-01-11 | 陈伟 |
|  | 1. 添加maven三脚架模板 | 2016-01-12 | 陈伟 |

## **整体技术选型**

spring-boot,grpc-java,mybatis

## **细节注意点**

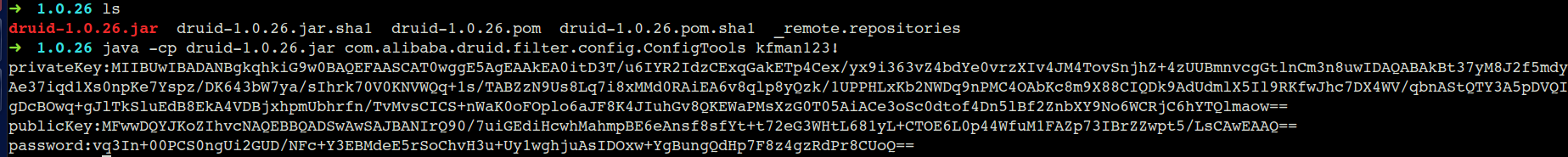
### **①druid加密**

mybatis已经集成在spring-boot-starter-mybatis里面，使用druid作为dataSource,程序配置需要注意druid的加密方法。参考(<https://github.com/alibaba/druid/wiki/使用ConfigFilter>)

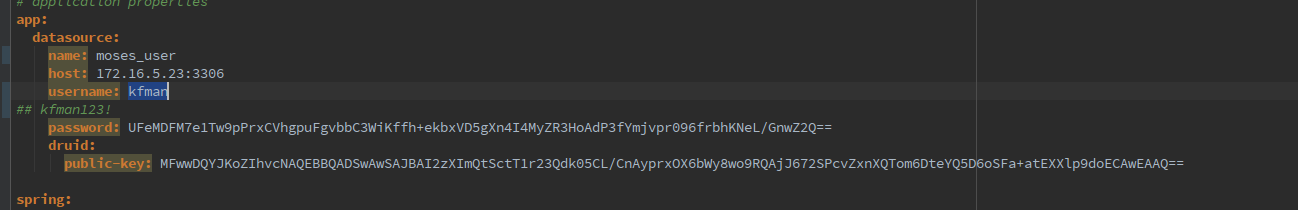
step1:

|  |
| --- |
| 命令行执行(druid的jar包，一般都在本地的maven repository中) |
| java -cp druid-1.0.26.jar com.alibaba.druid.filter.config.ConfigTools you\_password |

输出:



Step2:

 将输出的结果，放入项目resource下面application.yml中对应的字段

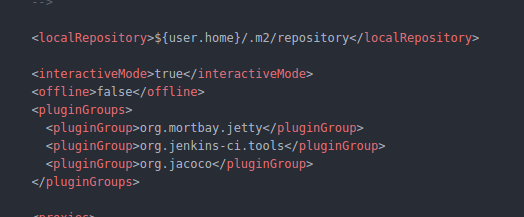
### **②**sonar集成maven．(前提是已经搭建好sonar服务器)

Step1:

配置自己本地的maven的settings.xml，加入

<pluginGroup>org.jacoco</pluginGroup>

|  |
| --- |
| <pluginGroups>  <pluginGroup>org.mortbay.jetty</pluginGroup>  <pluginGroup>org.jenkins-ci.tools</pluginGroup>  　　　　　<pluginGroup>org.jacoco</pluginGroup>  </pluginGroups> |

 如图:

　　　step2:

进入本地maven项目，执行

|  |
| --- |
| mvn jacoco:report sonar:sonar -Dsonar.host.url=http://172.16.45.3:9000 -DskipTests=true -Dbuild=dev |

### **③多profile的maven选择**

由于项目使用了kubernetes,cicd方式运维,所以使用了多profile的maven配置.配置如下:

|  |
| --- |
| <profiles>  <profile>  <id>dev</id>  <activation>  <property>  <name>build</name>  <value>dev</value>  </property>  </activation>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </profile>  <profile>  <id>kubernetes</id>  <activation>  <property>  <name>build</name>  <value>kubernetes</value>  </property>  <activeByDefault>true</activeByDefault>  </activation>  <dependencies>  <!-- spring-cloud kubernetes-->  <dependency>  <groupId>io.fabric8</groupId>  <artifactId>spring-cloud-starter-kubernetes</artifactId>  <version>${spring-cloud-starter-kubernetes.version}</version>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-commons</artifactId>  </dependency>  <!--add spring-cloud-zipkin-->  <dependency>  <groupId>io.fabric8</groupId>  <artifactId>spring-cloud-starter-kubernetes-zipkin</artifactId>  <version>${spring-cloud-starter-kubernetes.version}</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <inherited>true</inherited>  <configuration>  <argLine>-Djava.security.egd=file:/dev/./urandom</argLine>  <excludes>  <exclude>\*\*/\*KT.java</exclude>  </excludes>  </configuration>  </plugin>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  <plugin>  <groupId>io.fabric8</groupId>  <artifactId>fabric8-maven-plugin</artifactId>  <version>${fabric8.maven.plugin.version}</version>  <executions>  <execution>  <goals>  <goal>resource</goal>  <goal>build</goal>  <goal>helm</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build>  </profile>  </profiles> |

本地开发时,通过build参数指定profile(dev),自动化部署时候,就选择默认的kubernetes.

### **④本地开需配置环境变量,便于测试(指定默认spring-boot项目profile 为dev)**

* windows设置环境变量

SPRING\_PROFILES\_ACTIVE=dev

* linux设置环境变量

vim /etc/profile

#add

export SPRING\_PROFILES\_ACTIVE=dev

### **⑤需要暴露的端口,需要配置相关deployment.yml,svc.yml.**

* 如下暴露7575grpc端口

|  |  |
| --- | --- |
| src/main/fabric8/deployment.yml | |
| **spec:**  **replicas: 1**  **template:**  **spec:**  **volumes:**  - **name:** config  **configMap:**  **name:** ${project.artifactId}  **items:**  - **key:** application.properties  **path:** application.properties  **containers:**  - **name:** ${project.artifactId}  **ports:**  - **containerPort:** 8080  **name:** http  **protocol:** TCP  - **containerPort:** 7575  **name:** grpc  **volumeMounts:**  - **name:** config  **mountPath:** /app/config | |
| src/main/fabric8/svc.yml | |
| **metadata:**  **labels:**  **expose:** true  **hystrix.enabled:** true  **spec:**  **ports:**  - **name:** http  **port:** 8080  **protocol:** TCP  **targetPort:** 8080  - **name:** grpc  **port:** 7575  **protocol:** TCP  **targetPort:** 7575 | |

### **⑥使用context-path配置需要注意(主要在有controller的代码)**

* 如下配置.

|  |  |
| --- | --- |
| src/main/resources/application.yml | |
| **server:**  **context-path:** "/moses/user" | |
| src/main/fabric8/deployment.yml | |
| **spec:**  **replicas:** 1  **template:**  **spec:**  **volumes:**  - **name:** config  **configMap:**  **name:** ${project.artifactId}  **items:**  - **key:** application.yml  **path:** application.yml  **containers:**  - **volumeMounts:**  - **name:** config  **mountPath:** /app/config  **livenessProbe:**  **httpGet:**  **path:** /moses/user/health  **port:** 8080  **scheme:** HTTP  **initialDelaySeconds:** 200  **timeoutSeconds:** 10  **readinessProbe:**  **httpGet:**  **path:** /moses/user/health  **port:** 8080  **scheme:** HTTP  **initialDelaySeconds:** 200  **timeoutSeconds:** 10 | |

### **⑦使用模板快速创建项目**

|  |
| --- |
| maven创建自己的原型 |
| cd 原型项目目录  mvn archetype:create-from-project  cd target/generated-sources/archetype/  mvn install  mvn deploy –DskipTests  #可以发现有.m2/archetype-catalog.xml  #创建项目  mvn archetype:generate -DarchetypeCatalog=local |

Idea 使用私服中的archetype

|  |
| --- |
| 参考[(http://plugins.jetbrains.com/idea/plugin/7965-maven-archetype-catalogs)](http://plugins.jetbrains.com/idea/plugin/7965-maven-archetype-catalogs)  使用idea创建maven项目    选择add archetype 添加自己的原型模板.    选择自己的模板    输入项目名称信息      点击finish,看结果    #另外provider的demo也是类似方法 |

### **⑧使用mybatis-generator生成domain,mapper,xml**

|  |
| --- |
| 本项目默认使用 <**parent**>  <**groupId**>cn.vpclub</**groupId**>  <**artifactId**>spring-boot-starters</**artifactId**>  <**version**>1.0.5</**version**>  *<!-- lookup parent from repository -->* </**parent**>  作为顶级依赖,请查阅者自己注意.  Step1. 项目resources中添加,  Step2. 项目配置文件application.yml添加如下配置: |
| **mybatis:  config-location : "classpath:mybatis-config.xml"** *#checkConfig-location : false* **mapper-locations : "classpath:mapper/\*Mapper.xml"**  Step3. 执行**mybatis-generator(插件安装可以使用黄俊提供的破解版)** **.**  [Step4.新生成的mapper接口添加@Mapper](mailto:Step4.新生成的mapper接口添加@Mapper)注解.如下  **import** org.apache.ibatis.annotations.Mapper;  @Mapper **public interface** AppResourceMapper { |