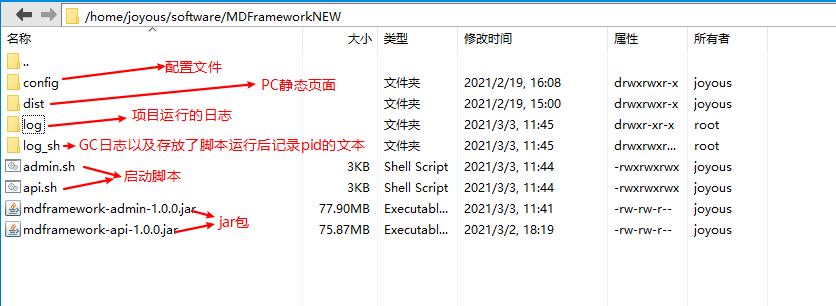
### 1.服务器安装jdk1.8, mysql 5.7, redis 3.2.8，nginx 1.15.0；

1. 服务器部署文件目录结构（以下文件在同级目录的压缩包中）



需要启动的jar包分为admin端（mdframework-admin-1.0.0.jar）和api端（mdframework-api-1.0.0.jar）

1. 项目规范化配置文件

目的：拉取代码对配置文件appcalition.[ yml | properties ]做的修改重新部署时不影响项目

1.在模块的pom文件中的org.apache.maven.plugins插件的configuration标签添加一下<excludes>标签内容；排除自身的配置文件，通过启动脚本[admin.sh](#配置文件名)指定jar包外部的配置文件

<plugin>

    <groupId>org.apache.maven.plugins</groupId>

    <artifactId>maven-jar-plugin</artifactId>

    <configuration>

        <excludes>

            <exclude>application-\*.yml</exclude>

            <exclude>application.yml</exclude>

            <exclude>bootstrap.yml</exclude>

        </excludes>

    </configuration>

</plugin>

2 .针对于日志输出，在项目中有logback-spring.xml和logback.xml来配置，而springboot加载的顺序是

logback.xml--->application.properties--->logback-spring.xml

对于logback.xml输出路径 代码如下

<property name="log.path" value="${user.dir}/logs/admin"/>

通过java的系统属性${user.dir}来拿到程序当前目录，此时无法实现在appcalition修改配置能在logback.xml生效；因为加载顺序的关系，logback.xml是先加载所以拿不到application.properties的值；所以我们采用logback-spring.xml，将logback.xml更名为logback-spring.xml

logback-spring.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<!--    logback.xml-->

<!--    <property name="log.path" value="${user.dir}/logs/admin"/>-->

    <springProperty scope="context" name="logdir" source="logback.logdir"/>

    <springProperty scope="context" name="logname" source="logback.logname"/>

    <property name="console.log.pattern"

              value="%red(%d{yyyy-MM-dd HH:mm:ss}) %green([%thread]) %highlight(%-5level) %boldMagenta(%logger{36}%n) - %msg%n"/>

    <property name="log.pattern" value="%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n"/>

   <!-- 控制台输出 -->

   <appender name="console" class="ch.qos.logback.core.ConsoleAppender">

      <encoder>

         <pattern>${console.log.pattern}</pattern>

            <charset>utf-8</charset>

      </encoder>

   </appender>

   <!-- 系统日志输出 -->

   <appender name="file\_info" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!--       <file>${log.path}/sys-info.log</file>-->

       <file>${logdir}/${logname}/sys-info.log</file>

        <!-- 循环政策：基于时间创建日志文件 -->

      <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <!-- 日志文件名格式 -->

<!--         <fileNamePattern>${log.path}/sys-info.%d{yyyy-MM-dd}.log</fileNamePattern>-->

         <fileNamePattern>${logdir}/${logname}/sys-info.%d{yyyy-MM-dd}.log</fileNamePattern>

         <!-- 日志最大的历史 60天 -->

         <maxHistory>60</maxHistory>

      </rollingPolicy>

      <encoder>

         <pattern>${log.pattern}</pattern>

      </encoder>

      <filter class="ch.qos.logback.classic.filter.LevelFilter">

            <!-- 过滤的级别 -->

            <level>INFO</level>

            <!-- 匹配时的操作：接收（记录） -->

            <onMatch>ACCEPT</onMatch>

            <!-- 不匹配时的操作：拒绝（不记录） -->

            <onMismatch>DENY</onMismatch>

        </filter>

   </appender>

   <appender name="file\_error" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!--       <file>${log.path}/sys-error.log</file>-->

       <file>${logdir}/${logname}/sys-error.log</file>

        <!-- 循环政策：基于时间创建日志文件 -->

        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <!-- 日志文件名格式 -->

<!--            <fileNamePattern>${log.path}/sys-error.%d{yyyy-MM-dd}.log</fileNamePattern>-->

            <fileNamePattern>${logdir}/${logname}/sys-error.%d{yyyy-MM-dd}.log</fileNamePattern>

         <!-- 日志最大的历史 60天 -->

         <maxHistory>60</maxHistory>

        </rollingPolicy>

        <encoder>

            <pattern>${log.pattern}</pattern>

        </encoder>

        <filter class="ch.qos.logback.classic.filter.LevelFilter">

            <!-- 过滤的级别 -->

            <level>ERROR</level>

         <!-- 匹配时的操作：接收（记录） -->

            <onMatch>ACCEPT</onMatch>

         <!-- 不匹配时的操作：拒绝（不记录） -->

            <onMismatch>DENY</onMismatch>

        </filter>

    </appender>

   <!-- 用户访问日志输出  -->

    <appender name="sys-user" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!--      <file>${log.path}/sys-user.log</file>-->

      <file>${logdir}/${logname}/sys-user.log</file>

        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <!-- 按天回滚 daily -->

<!--            <fileNamePattern>${log.path}/sys-user.%d{yyyy-MM-dd}.log</fileNamePattern>-->

            <fileNamePattern>${logdir}/${logname}/sys-user.%d{yyyy-MM-dd}.log</fileNamePattern>

            <!-- 日志最大的历史 60天 -->

            <maxHistory>60</maxHistory>

        </rollingPolicy>

        <encoder>

            <pattern>${log.pattern}</pattern>

        </encoder>

    </appender>

   <!-- 系统模块日志级别控制  -->

   <logger name="com.mdframework" level="info" />

   <!-- Spring日志级别控制  -->

   <logger name="org.springframework" level="warn" />

   <root level="info">

      <appender-ref ref="console" />

   </root>

   <!--系统操作日志-->

    <root level="info">

        <appender-ref ref="file\_info" />

        <appender-ref ref="file\_error" />

    </root>

   <!--系统用户操作日志-->

    <logger name="sys-user" level="info">

        <appender-ref ref="sys-user"/>

    </logger>

</configuration>

source拿取application.properties（jar外部的config下的配置文件）下的

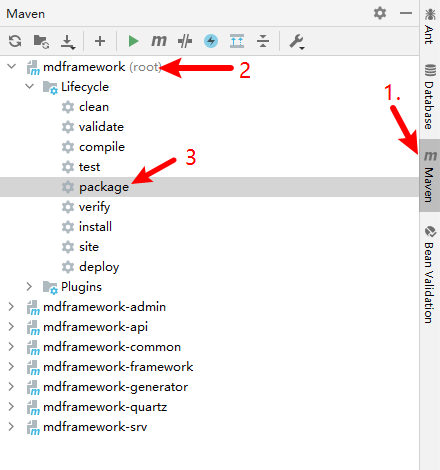
logback:

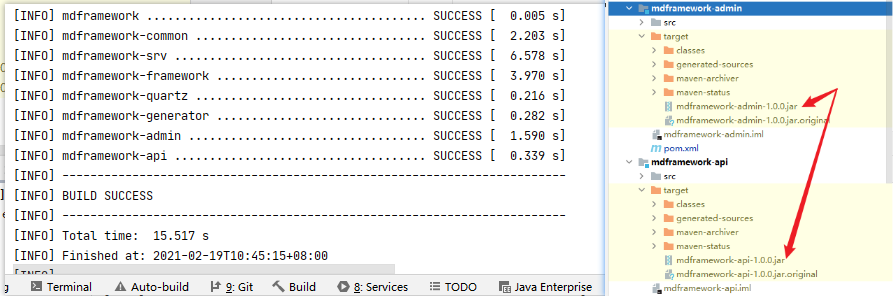
  logdir: log

  logname: admin

此时可以在jar同目录下的config的配置文件application-admin.yml下配置我们的logname文件夹名字了；

1. IDEA打包Jar的方法



通过package打包生成jar包，而在父工程的packaging标签是<packaging>pom</packaging>，意思是使用maven分模块管理，都会有一个父级项目，一般来说所有的父级项目的packaging都为pom，在子模块的pom文件中的packaging标签值默认是jar，如果不做配置，maven会将该项目打成jar包。当在控制台看到【BUILD SUCCESS】则会在左侧的项目栏中看到两个jar包，将jar包移动到（复制）到服务器上的相应位置上。

1. Nginx部署静态前端

Nginx.conf的目录一般是/usr/local/nginx/conf下的nginx.conf；在编辑该文件时保险起见可以先拷贝一份；

Nginx.conf文件(红色根据实际情况修改)：

#橙色魔方官网

server

{

        listen 80;

        server\_name chengse.mdsoftware.cn;

         location /{

         root /usr/local/mdsoftware/wwwroot/ChengSeMoFang/dist;

         index index.html;

         }

}

其中 /usr/local/mdsoftware/wwwroot/ChengSeMoFang/dist;是存放PC静态页面路径；以上的server\_name的值此处是chengse.mdsoftware.cn;（域名）可以解析为[211.149.227.130](https://site.ip138.com/211.149.227.130/" \t "_blank)

当我们访问nginx会帮我们解析域名为[211.149.227.130](https://site.ip138.com/211.149.227.130/" \t "_blank)，而location 的值是 ‘/’ 即<http://chengse.mdsoftware.cn/> 会访问到 root：下的 index.html文件；

当我们修改过nginx.conf文件我们需要进入到/usr/local/nginx/sbin目录下运行

./nginx -s reload



1. 配置文件：application-[admin|yml].properties;

此配置文件为服务器上jar包外部config下的配置文件，

修改以下内容

端口：

server.port=8084

日志目录：

logback.logdir=log

logback.logname=admin

Tomcat临时目录：

server.tomcat.basedir=/home/temp

Mysql数据库地址：

spring.datasource.url=jdbc:mysql://211.149.227.130:3306/md-chengsemofang?useUnicode=true&characterEncoding=UTF-8&useSSL=false&zeroDateTimeBehavior=convertToNull

Mysql数据库用户名：

spring.datasource.username=root

Mysql数据库密码：

spring.datasource.password=abc1no2password3abc

Redis服务器地址：

spring.redis.host=127.0.0.1

Redis端口： (redis默认端口为6379)

spring.redis.port=6379

Redis密码： (默认无密码)

spring.redis.password=

文件上传保存路径：

file-upload-path=/usr/local/mdsoftware/wwwroot/ChengSeMoFang/upload-dir

文件访问前缀：

file-upload-path-url-pre=http://211.149.227.130:8084/uploadfiles/

文件路径：

spring.resources.static-locations:classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/,file:/usr/local/mdsoftware/wwwroot/ChengSeMoFang/upload-dir/

1. 启动脚本xxx.sh (xxx为admin或api)

Admin.sh修改以下变量的值：

SERVICEAPP\_NAME ；SERVICEAPP\_LOGHOME；SERVICEAPP\_LOGHOM；

如果是api.sh则 pid\_admin、SYSTEM\_ENV:admin修改成对应的pid\_api 、SYSTEM\_ENV:api

SERVICEAPP\_NAME为打包后的jar包名称，SERVICEAPP\_LOGHOME则是存放heap dump文件，其中保存的是FullGC后留下的对象信息。

# !/bin/bash

# description:start/stop/check sjyl-admin.jar

SERVICEAPP\_NAME=mdframework-admin-1.0.0.jar

SERVICEAPP\_DEPHOME=/home/joyous/software/MDFrameworkNEW

SERVICEAPP\_LOGHOME=/home/joyous/software/MDFrameworkNEW/log\_sh

SYSTEM\_ENV=admin

# config for 4C8G

JAVA\_OPTS="-Xms256m -Xmx512m  \

-XX:PermSize=128M \

-XX:MaxPermSize=256m \

-Xss1m \

-Xmn70m \

-XX:+AggressiveOpts \

-XX:+UseBiasedLocking \

-XX:+CMSParallelRemarkEnabled \

-XX:+UseConcMarkSweepGC \

-XX:ParallelGCThreads=4 \

-XX:SurvivorRatio=4 \

-XX:TargetSurvivorRatio=85 \

-verbose:gc \

-XX:+PrintGCDetails \

-XX:+PrintGCDateStamps \

-XX:+PrintHeapAtGC \

-Xloggc:$SERVICEAPP\_LOGHOME/adming.log \

-XX:+HeapDumpOnOutOfMemoryError \

-XX:HeapDumpPath=$SERVICEAPP\_LOGHOME/dump.logs "

# -Djava.awt.headless=true

pid\_admin=0

#

checktpid(){

    if [ ! -d ${SERVICEAPP\_LOGHOME} ]; then

     mkdir -p ${SERVICEAPP\_LOGHOME}

     chmod 777 ${SERVICEAPP\_LOGHOME}

    fi

    if [ ! -f ${SERVICEAPP\_LOGHOME}/admin\_pid ]; then

     touch ${SERVICEAPP\_LOGHOME}/admin\_pid

    fi

    admin\_pid=`cat $SERVICEAPP\_LOGHOME/admin\_pid | awk '{print $1}'`

    admin\_pid=`ps -aef | grep ${admin\_pid} | awk '{print $2}' |grep ${admin\_pid}`

    if [ ${admin\_pid} ]; then

            pid\_admin=1

    else

            pid\_admin=0

    fi

}

#

start(){

    checktpid

    if [ $pid\_admin -ne 0 ];then

        echo "WARN:$SERVICEAPP\_NAME already started! Ignoring startup request."

    else

        rm -f $SERVICEAPP\_LOGHOME/admin\_pid

        java -jar -Duser.timezone=GMT+8 $JAVA\_OPTS $SERVICEAPP\_NAME --spring.config.location=config/application-${SYSTEM\_ENV}.yml > /dev/null 2>&1 &

#         java -jar $JAVA\_OPTS $SERVICEAPP\_DEPHOME/$SERVICEAPP\_NAME --spring.profiles.active=$SYSTEM\_ENV > /dev/null 2>&1 &

        echo $! > $SERVICEAPP\_LOGHOME/admin\_pid

        echo "Starting $SERVICEAPP\_NAME admin..."

    fi

}

#

stop(){

    checktpid

    if [ $pid\_admin -ne 0 ];then

        echo "Stoping $SERVICEAPP\_NAME..."

        kill -9 $tpid

        sleep 3s

        checktpid

        if [ $pid\_admin -ne 0 ];then

                echo "Stoping use kill -9..."

            kill -9 $tpid

        else

            echo "$SERVICEAPP\_NAME Stopped!"

        fi

    else

                 echo "WARN:$SERVICEAPP\_NAME is not runing"

    fi

}

#

status(){

    checktpid

    if [ $pid\_admin -eq 0 ];then

        echo "$SERVICEAPP\_NAME is not runing"

    else

        echo "$SERVICEAPP\_NAME is runing"

    fi

}

#

case "$1" in

    'start')

        start

        ;;

    'stop')

        stop

        ;;

    'restart')

        stop

        sleep 5

        start

        ;;

    'status')

        status

        ;;

    \*)

    echo "Usage: $0 {start|stop|restart|status}"

    exit 1

esac

exit 0;

1. 脚本命令

**admin.sh start (后台启动) api.sh start (接口启动)**

**admin.sh stop(后台停止) api.sh stop(接口停止)**

**admin.sh restart (后台重启) api.sh restart (接口重启)**

**admin.sh status (查询后台运行状态) api.sh status (查询接口运行状态)**