

# 多套环境及持续集成方案

---

环境预估及分配

**jenkins**的相关配置

配置**Item**

服务器相关设置

工程代码及**POM**的配置



## 之前共用测试环境现有部署情况

之前测试环境整体与生产环境类似的架构及部署，不同的地方在于`smp-eaec`应用是部署在`Windows`系统 `Weblogic`上的，而生产环境则是部署在`Linux`上的。

## 新测试环境与之前共用环境的部署

根据环境整体情况与人力安排，新的独立测试环境主要对相关的 6 个包进行构建与持续集成，缓存及 MQ 和数据均使用相同的一套环境。

## 环境路径检查

经检查之前共用环境与生产环境下的项目部署的路径情况

### 快速检查

检查当前运行程序，后根据其 PID 查看


```
ps -ef |grep java  
  
ll /proc/PID
```

对比后发现，生产环境与之前共用环境的部署也都不一致，为达到后续统一化，新的测试环境的部署路径需要与生产一致，但生产环境的部署也有点太随意化了，有部署应用还是考虑到整体的统一，均部署到`/opt/pruduct/projectName`内，`projectName`为项目名称。



# 多用户多环境的jenkins 的配置

## jenkins多用户的配置

- 1. 安装插件Role-based Authorization Strategy
- 2. 在jenkins的[全局安全配置](#)选择[启用安全](#)并在授权策略里选中Role-Based Strategy。
- 3. 新建一个用户，用于分配指定项目的权限
- 4. 进入jenkins系统设置页面，点击进入Manage and Assign Roles页面设置,此选择必须装有上面插件后才能显示.
- 5. 点击进入Manage Roles配置,进行如下配置

Manage and Assign Roles



Global roles

Role	Overall		Credentials					Agent							Job							
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ys	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>


Role to add

Add


Project roles

Role	Pattern	Credentials					Job							Run		SCM				
		Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag	
	smp.*	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

- 6. 点击进入Assign Roles页面,进入如下配置

Assign Roles


Global roles

User/group	ys
	<input checked="" type="checkbox"/>
ystest	<input type="checkbox"/>
Anonymous	<input type="checkbox"/>

User/group to add

Add

Item roles

User/group	ys
	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>

此插件的匹配规则是按项目名进行正则匹配的,在命令item的时候注意项目名的命名.

## jenkins多环境的配置

- 1. 安装插件SSH Slaves plugin用户远程上传文件到指定服务器.
- 2. 配置构建需要的服务器,进入jenkins的系统配置选项页面,找到Publish over SSH选项增加一台远程服务器,进行如下配置

SSH Server

Name

smp\_service\_testevn\_172.30.9.118

?

Hostname

172.30.9.118

?

Username

ysdduser

?

Remote Directory

/opt/product/

?

增加

高级...

Test Configuration

删除

高级...

## 注意事项

1. 密码或免验证密钥在高级里设置,
2. 远程目录要特别注意一下,要有相应权限,如果没有权限时先进入家目录再做其它,如果目录有相应权限,则SSH到目标服务器里后继其它操作则是直接在远程目录内操作

## jenkins 多编译环境设置

项目一多,不可避免会遇到多JDK版本的项目,如果用Jenkins编译,则需要设置相应的JDK 进入Jenkins设置--> **全局工具配置**内(根据Jenkins版本不同可能处于不同配置页面),选中JDK增加一个对应版本的JDK.

JDK

JDK 安装

JDK

别名

jdk7

JAVA\_HOME

/usr/local/jdk1.7.0\_79

自动安装

删除 JDK

JDK

别名

jdk8

JAVA\_HOME

/usr/local/jdk/jdk1.8.0\_66

自动安装

删除 JDK

新增 JDK

系统下JDK 安装列表

# 新建及配置Item

## 新建Item

点击Jenkins中的**新建Item**项,在新弹出的页面内输入一个任务名字,注意与前面你想的权限控制的正则匹配,在下面快速选择默认配置项选中**构建一个maven项目**,然后点击确定,进入配置项

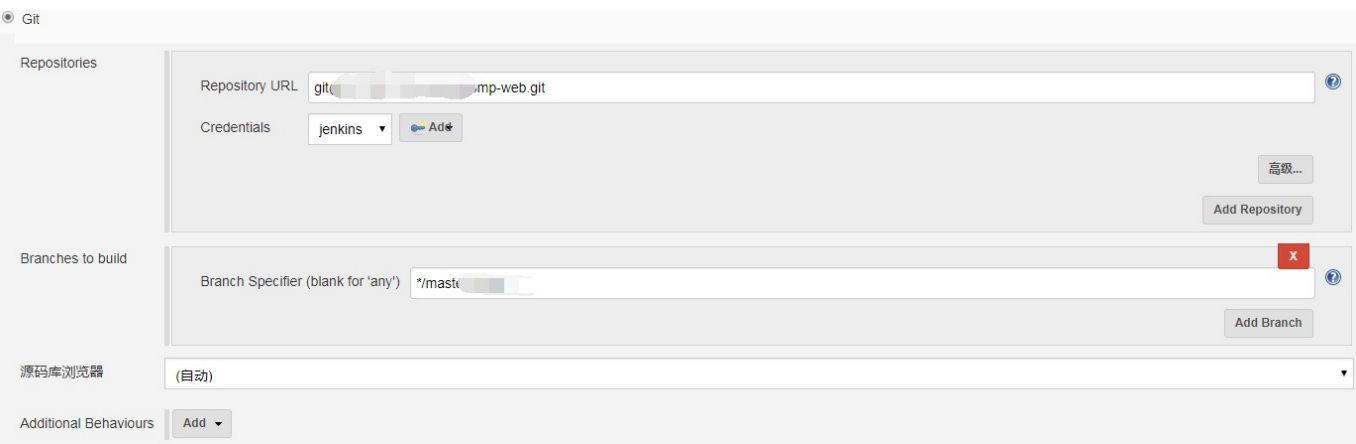


如果你有相似项目,可以用快速复制方式来创建.

## 配置

**General**中选择构建时所用的JDK版本;

**源码管理**选中你的源码管理方式,这里使用的是git,填写git地址与构建分支



**构建环境**中配置你需要远程发部的服务器信息

## 构建环境

☐ Send files or execute commands over SSH before the build starts  
☒ Send files or execute commands over SSH after the build runs

SSH Publishers

SSH Server

Name: smp\_web\_testevn\_

Transfers

Transfer Set

Source files: target/\*.\*war

Remove prefix: target

Remote directory: smp-web-8002/webapps

Exec command: 

```
cd /opt/product/smp-web-8002/webapps
cp smp-web.war smp-web.war.bak
mv smp-web-0.0.1-SNAPSHOT.war smp-web.war
/opt/product/server.sh restart smp-web-8002
```

Either Source files, Exec command or both must be supplied  
All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables

Add Transfer Set

Add Server

注意:这里的行程路径与开始配SSH全局设置的远程是相互影响的,全局为工作空间,这里的路径是指工作空间下的相对路径,这点包括Jenkins自身都是类似的机制,需要拷贝文件的路径也是类似,在Jenkins的工作空间的基础上的相对路径.

Build配置如下图,maven命令

```
clean install -Dmaven.test.skip -Ptest
```

-D后为跳出测试的周期,-P后为不同环境的环境参数,注意两者之间都没有空格

Build

Root POM: pom.xml

Goals and options: clean install -Dmaven.test.skip -Ptest

MAVEN\_OPTS:

☐ Incremental build - only build changed modules  
☐ Disable automatic artifact archiving  
☐ Disable automatic site documentation artifact archiving  
☐ Disable automatic fingerprinting of consumed and produced artifacts  
☒ Enable triggering of downstream projects  
☒ Block downstream trigger when building

☐ Build modules in parallel  
☐ Use private Maven repository  
☐ Resolve Dependencies during Pom parsing  
☐ Run Headless  
☐ Process Plugins during Pom parsing  
☒ 使用自定义的工作空间

目录: smp-parent/smp-web/

注意在特殊项目时,比如多模块的Maven项目,子模块需要构建在父工程的目录下,需要使用自定义的工作空间,这里也是指相对路径,只写父项目的路径下的工程名就好,但必须与pom.xml文件的名字一样,不然容易报错.

上下游项目的配置

项目之间有依赖的,必选是上游所依赖的项目构建完成时才能构建当前项目,这时需要配置上下游项目的依赖 在构建触发器内选中Build after other projects are built,选择填写对应的上游项目,这样构建上游的项目后自动构建下游项目

构建触发器

☐ Build whenever a SNAPSHOT dependency is built

☐ 触发远程构建 (例如,使用脚本)

☒ Build after other projects are built

Projects to watch

smp-common,

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Build periodically

☐ Poll SCM

?

?

?

?

?

整个项目的有比较完整的上下游关系时,只需要构建最上游的项目,则会自动构其下游的项目

## Maven project smp-api

 工作区

 最新修改

上级项目

smp-common

下级项目

smp-cart

smp-order

smp-outside

smp-quartz

smp-sms

smp-web



# 配置服务器设置

Jenkins完成构建工作并把相关文件上传的服务器后,服务器还需要做相关配置来完成整个发布工作其大概步骤包括以下内容,备份原有资料-->对当前版本文件进行相关操作-->重启对应的应用,整个smp项目有发布了Weblogic,Tomcat,JAR三种形式,这里逐一说明.

## 备份原有项目文件及操作当前文件

如果是正式环境,最好是使用shell脚本备份成以日期模式的文件夹,防止意外发生,测试环境因频繁构建与测试,可以直接cp备份文件

```
cd /opt/weblogic/webapp/ea_domain/
rm -rf servers/eaec
cp -rf eaec/ servers/
cd /opt/weblogic/webapp/ea_domain/eaec/
/opt/weblogic/java/jdk1.8.0_144/bin/jar -xvf eaec-1.0-SNAPSHOT.war
rm -rf eaec-1.0-SNAPSHOT.war
cd ..
cp servers/domain_bak/weblogic.xml eaec/WEB-INF/
/etc/init.d/weblogic restart
```

这段shell命令在Jenkins插件的SSH远程服务器里的Exec command设置就是指构建完成后上传文件前执行相关shell命令,主要包括内容有删除原有备份,备份当前项目内容,解压war包到指定项目文件夹,删除对应war包,拷贝当前服务器一些特殊的配置文件,配合服务器中的脚本完成比较繁杂的任务,这里是重启Weblogic

## 重启对应的应用

一般来说,重启应用在Jenkins中比较难完全,因为首先要关闭正在运行的对应项目,最好是事先写一个shell脚本放在指定目录,用Jenkins调用相关脚本来完成这一操作.

一个重启应用脚本的示例

```
#!/bin/sh
#kill smp-server pid
if [ $# -ne 2 ]
then
    echo "Usage: server.sh stop/start/restart APP_NAME"
    exit 1
fi
APP_NAME=$2
curBasePath=`dirname $0`
if [ "stop" = "$1" ]
then
    pidlist=`ps -ef|grep $APP_NAME | grep -v "grep" | grep -v "server.sh " |
    awk '{print $2}'`
```

```

#echo "smp-server Id list :$pidlist"
if [ "$pidlist" = "" ]
then
    echo "no smp-server pid alive"
else
    for pid in ${pidlist}
    {
        kill -9 $pid
        echo "KILL $pid:"
        echo "service stop success"
    }
fi
sleep 2
`rm -rf $curBasePath/$APP_NAME/webapps/ROOT`
elif [ "start" = "$1" ]
then
    nohup $curBasePath/$APP_NAME/startup.sh &> /dev/null
elif [ "restart" = "$1" ]
then
    pidlist=`ps -ef|grep $APP_NAME | grep -v "grep" | grep -v "server.sh " |
awk '{print $2}' `
    #echo "smp-server Id list :$pidlist"
    if [ "$pidlist" = "" ]
    then
        echo "no smp-server pid alive"
    else
        for pid in ${pidlist}
        {
            kill -9 $pid
            echo "KILL $pid:"
            echo "service stop success"
        }
    fi
    sleep 2
    `rm -rf $curBasePath/$APP_NAME/webapps/ROOT`
    nohup $curBasePath/$APP_NAME/startup.sh &> /dev/null
fi
echo "done."
exit 0

```

其对应的`startup.sh`,这里主要是用来启动三个jar包的相关应用

```

cd /opt/product/smp-service/smp-order
exec nohup java -jar smp-order-0.0.1-SNAPSHOT.jar &

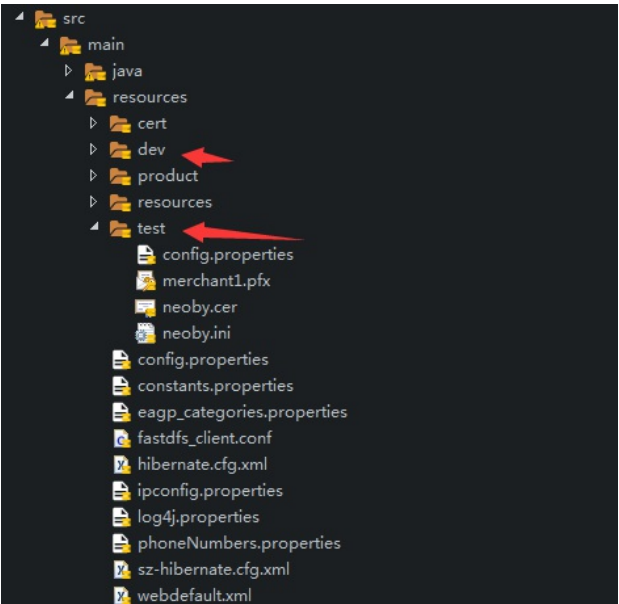
```

# 工程代码及POM的配置

因原项目是单环境,在项目的工程结构和POM文件中均没有相关配置,有些配置项还是写死在代码中的,因为这里需要把相关配置移到配置文件中,并在POM中构建相应的资源

## 工程结构的改变

在原工程目录的resources目录下添加对应环境的文件夹,把对应需要的资源文件放在对应的文件夹内,如下所示,



在test文件下放在测试环境对应的配置文件和相关证书

## POM文件修改

### 添加profile属性

在POM.XML文件下的profiles节点下添加以下代码,代表三个环境的变量

```
<profile>
    <!-- 本地开发环境 -->
    <id>dev</id>
    <properties>
        <profiles.active>dev</profiles.active>
    </properties>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
</profile>

<profile>
    <!-- 测试环境 -->
    <id>test</id>
    <properties>
        <profiles.active>test</profiles.active>
```

```

<maven.resources.overwrite>true</maven.resources.overwrite>
    </properties>
</profile>
<profile>
    <!-- 生产环境 -->
    <id>product</id>
    <properties>
        <profiles.active>product</profiles.active>

    <maven.resources.overwrite>true</maven.resources.overwrite>
        </properties>
    </profile>

```

`activeByDefault`标签为默认值,`maven.resources.overwrite`标签`true`值是代表覆盖原文件

拷贝对应环境的resources资源文件

在POM.XML文件下的`resources`节点下添加以下代码,代表对资源文件的操作,每个`resource`节点都是一个单独的操作,因原POM使用的一个ANT插件用来拷贝证书到特定的目录,这里单独配了一个.

```

    <resource>
        <directory>src/main/resources</directory>
        <includes>
            <include>**/*.properties</include>
            <include>**/*.xml</include>
            <include>**/*.conf</include>
        </includes>
    </resource>
    <!-- 此处功能为原ant插件的功能，拷贝证书到指定路径 -->
    <resource>

        <targetPath>${project.build.directory}/${project.build.finalName}</targetPath>

        <directory>src/main/resources/${profiles.active}</directory>
        <excludes>
            <exclude>**/*.properties</exclude>
            <exclude>**/*.xml</exclude>
            <exclude>**/*.conf</exclude>
        </excludes>
    </resource>
    <!-- 拷贝指定环境的配置文件到替换resources里的配置文件 -->
    <resource>

        <directory>src/main/resources/${profiles.active}</directory>
        <includes>
            <include>**/*.properties</include>
            <include>**/*.xml</include>
            <include>**/*.conf</include>
        </includes>
    </resource>

```

```
</resource>
```

注意拷贝路径以及include和exclude的合理使用

## 修改工程代码配合后续测试操作

在测试配置文件添加环境信息

在test/config.properties内添加变量

```
env=test
```

在环境公共类中添加对环境的判断

原工程开发在异常时返回了true值,感觉比较奇葩,但getProperty方法做了异常处理返回空字符串,暂时没理它.

```
public static boolean isTest() {
    try {
        return
"test".equalsIgnoreCase(CONFIG_CACHE.getProperty("env"));
    } catch (Exception e) {
        return false;
    }
}
```

修改测试环境的特殊需求

比如测试环境不想要验证码,特别是我做的性能测试,对验证码什么的最不需要了,在原来的验证Action中修改相关代码,用一个固定的验证码方便后续测试

```
// 如果当前环境是测试环境,则验证码为固定值"0000"
if (ConfigUtils.isTest()) {
    for (int i = 0; i < 4; i++) {
        validateCode += "0";
        g.setColor(new Color(20 + random.nextInt(110),
20 + random.nextInt(110), 20 + random.nextInt(110)));
        g.drawString("0", 13 * i + 15, 18);
    }
} else {

    for (int i = 0; i < 4; i++) {
        int a = random.nextInt(codeList.length() - 1);
```

```
        String rand = codeList.substring(a, a + 1);
        validateCode += rand;
        g.setColor(new Color(20 + random.nextInt(110),
20 + random.nextInt(110), 20 + random.nextInt(110)));

        g.drawString(rand, 13 * i + 15, 18);
    }
}
```

到此,环境及自动构建工作可以暂时告一段落了,后续改动只涉及细节调整.