Jenkinsfile 讲义(第二期)

参数化构建
parameters 语句块，在这个语句块中可以放入我们构建时需要的参数，全局唯一；

string 字符串参数
string(name: 'str', defaultValue: '默认值', description: '这是描述文字', trim: false)

分别解释字段

name: 一般用于存储参数的变量名

defaultValue: 默认值

description: 描述文字

trim: 是否去掉字符串前后的空格，值是布尔值

bool 参数
booleanParam(name: 'bool', defaultValue: false, description: '这是描述文字')

choice 选项参数
choice(name: 'chs', choices: ['ios','android'], description: '这是描述文字')

choice(name: 'chs', choices: 'ios\nandroid', description: '这是描述文字')
choices: 选项 2 种写法

其中的选项 choices，数组结构，默认指向第一个选项选

text 文本参数
text(name: 'tex', defaultValue: '默认值', description: '这是描述文字')

换行使用 \\n 或者 '''...'''

实际使用

```
pipeline{
  agent any
  parameters{
    string(name:'str', defaultValue: 'DEFAULT', description:'这是描述文字'， trim:false)
    booleanParam(name: 'bool', defaultValue: false, description: '这是描述文字')
    choice(name: 'chs', choices: ['选项 1','选项 2'], description: '这是描述文字')
    text(name: 'tex', defaultValue: '默认值', description: '这是描述文字')
  }
  stages{
    stage('打包'){
      steps{
        sh label:'', script: 'echo build'
      }
    }
    stage ('发布'){
      steps{
        sh label:'', script:'echo deploy'
      }
    }
    stage('do sth'){
      steps{
        sh label:'',script:'echo do sth'
      }
    }
  }
  post {
    always {
      sh label: '',script: 'echo always'
    }
    success{
      sh label:'' , script:'echo success'
    }
    failure{
```

```
        sh label:'', script:'echo failure'
      }
      aborted{
        sh label:'', script:'echo aborted'
      }
    }
}
```

参数接收
params.参数名

```
pipeline{
    agent any
    parameters{
      string(name:'str', defaultValue: 'DEFAULT', description:'这是描述文字'， trim:false)
      booleanParam(name: 'bool', defaultValue: false, description: '这是描述文字')
      choice(name: 'chs', choices: ['选项 1','选项 2'], description: '这是描述文字')
      text(name: 'tex', defaultValue: '默认值', description: '这是描述文字')
    }
    stages{
      stage('打包'){
        steps{
          sh label:'', script: "echo ${params.str}"
          sh label:'', script: "echo ${params.bool}"
          sh label:'', script: "echo ${params.chs}"
          sh label:'', script: "echo ${params.tex}"
        }
      }
      stage ('发布'){
        steps{
          echo "deploy"
        }
      }
      stage('do sth'){
        steps{
          echo "do sth"
        }
```

```
    }
  }
  post {
    always {
      sh label:'', script: "echo always"
    }
    success{
      sh label:'', script: "echo success"
    }
    failure{
      sh label:'', script: "echo failure"
    }
    aborted{
      sh label:'', script: "echo aborted"
    }
  }
}
```

options
指令允许从流水线内部配置特定于流水线的选项，全局唯一

disableConcurrentBuilds() 禁止并发

例子

```
pipeline{
  agent any
  parameters{
    string(name:'str', defaultValue: 'DEFAULT', description:'这是描述文字'， trim:false)
    booleanParam(name: 'bool', defaultValue: false, description: '这是描述文字')
    choice(name: 'chs', choices: ['选项 1','选项 2'], description: '这是描述文字')
    text(name: 'tex', defaultValue: '默认值', description: '这是描述文字')
  }
  stages{
    stage('打包'){
      steps{
        sh label:'', script: "echo ${params.str}"
```

```
            sh label:'', script: "echo ${params.bool}"
            sh label:'', script: "echo ${params.chs}"
            sh label:'', script: "echo ${params.tex}"
        }
    }
    stage ('发布'){
        steps{
            sh label:'', script: "echo deploy"
        }
    }
    stage('do sth'){
        steps{
            sh label:'', script: "echo do sth"
        }
    }
  }
  post {
    always {
        sh label:'', script: "echo always"
    }
    success{
        sh label:'', script: "echo success"
    }
    failure{
        sh label:'', script: "echo failure"
    }
    aborted{
        sh label:'', script: "secho aborted"
    }
  }
}
```

sleep:
NANOSECONDS:：纳秒
sleep(time: 5, unit: 'NANOSECONDS')

MICROSECONDS:：微秒
sleep(time: 5, unit: 'MICROSECONDS')

MILLISECONDS：毫秒

sleep(time: 5, unit: 'MILLISECONDS')

SECONDS：秒
sleep(5) 简写
sleep(time:5, unit: 'SECONDS')

MINUTES: 分钟
sleep(time: 5, unit: 'MINUTES')

HOURS: 小时
sleep(time: 5, unit: 'HOURS')

DAYS: 天
sleep()time: 5, unit: 'DAYS')

timestamps() 打印时间戳
```
pipeline{
   agent any
   options{
      timestamps()
   }
   stages{
      stage('test'){
         steps{
             sh label:'', script: "echo one"
            sleep(5)
            sh label:'', script: "echo two"
         }
      }
   }
}
```

timeout(time:超时时间, unit: '单位')

unit:
NANOSECONDS:：纳秒
timeout(time: 3, unit: 'NANOSECONDS')

MICROSECONDS:：微秒
timeout(time: 3, unit: 'MICROSECONDS')

MILLISECONDS：毫秒
timeout(time: 3, unit: 'MILLISECONDS')

SECONDS：秒
timeout(time: 3, unit: 'SECONDS')

MINUTES: 分钟
timeout(3) 默认的是 MINUTES, 可以使用这样的缺省写法
timeout(time:3, unit: 'MINUTES')

HOURS: 小时
timeout(time: 3, unit: 'HOURS')

DAYS: 天
timeout(time: 3, unit: 'DAYS')

例子

```
pipeline{
   agent any
   options{
      timestamps()
      timeout(time: 3, unit: 'SECONDS')
   }
   stages{
      stage('test'){
         steps{
            sleep(20)
         }
      }
   }
}
```

activity
此块的日志中没有活动之后的超时，而不是绝对持续时间。
布尔类型 true or false
timeout(activity: true, time: 5, unit: 'NANOSECONDS')

例子

```
pipeline{
```

```
    agent any
    options{
        timestamps()
        timeout(activity: true, time: 3, unit: 'SECONDS')
    }
    stages{
        stage('test'){
            steps{
                sleep(20)
            }
        }
    }
}
```

environment

environment 指令制定一个 键-值对序列，该序列将被定义为所有步骤的环境变量，或者是特定于阶段的步骤， 这取决于 environment 指令在流水线内的位置。

说白了就是用来存储变量的

environment 的位置取决于他可以被访问到的作用域

```
pipeline{
    agent any
    environment{
        foo = 'today is Friday'
    }
    stages{
        stage('test'){
            steps{
                sh label:'', script: "echo ${foo}"
            }
        }
    }
}
```

结合 groovy 语言的写法，可以这么读取变量

```
pipeline{
    agent any
```

```
    environment{
        foo = 'today is Friday'
    }
    stages{
        stage('test'){
            steps{
                script{
                    println(foo)
                }
            }
        }
    }
}
```

如果放在 stage 中

```
pipeline{
    agent any
    stages{
        stage('test'){
            environment{
                foo = 'today is Friday'
                }
            steps{
                sh label:'', script: "echo in test ${foo}"
            }
        }
        stage('readfoo'){
            steps{
                sh label:'', script: "echo in readfoo ${foo}"
            }
        }
    }
}
```