# 2018-4-16

1. 编写juniperScheduler的grammar文件

|-对固定的字符进行匹配时，要注意匹配的唯一性，不要忽略空格的存在

如：test\_ test 进行匹配时，第一个test的下划线相当于空格，可以一起进行匹配，而不要匹配第二个test

|-词法的匹配不是按照顺序从文本的开始到结束进行匹配的，它有可能匹配到文本中的任意一个符合规则的串

1. 对于类中的属性进行约束的方法

|-@NotNull

@Length(min = 1, max = 63)

@Pattern.List({

@Pattern(regexp = "(^[0-9A-Za-z][0-9A-Za-z\_-]\*$)")

})

**protected** String name;//相当于对name属性进行了三种约束，非空，长度，正则

|-使用了泛型的属性，要加好标记，列出泛型可能对应的类，方便于查找维护

1. 熟悉Juniper防火墙的基本查看，提交，撤销相关的命令

|-https://blog.csdn.net/qxy1242062102/article/details/52972536

# 2018-4-17

1. Java枚举类，关于values()方法的使用，定义一个枚举类时，要定义个属性，并且带一个构造器

|-**public** **enum** Week {

***SUNDAY***("sunday"),

***THURSDAY***("thursday"),

***TUESDAY***("tuesday"),

***WEDNESDAY***("wednesday");

**public** String name;

**private** **Week**(String name) {

**this**.name = name;

}

values()就是将枚举中的每个值的名称拿出来，然后用枚举类中的属性可以获得

1. 注意每天开始写代码的时候更新代码，避免重复工作
2. 完成scheduler相关的测试

# 2018-4-18

1. 开始写代码时，先创建一个分支，然后写代码，写完代码先在本地与主分支进行合并，解决完冲突后，再合并到工作分支，然后提交到远程，创建一个PR，等待合并。
2. 同步不同仓库的代码，首先修改版本号，与所做的代码修改一起提交，然后合并

|-PR合并完成后，gradle build

|- gradle uploadArchives 推送到远程

|- 修改要进行代码同步的仓库，修改build.gradle文件中的版本号，与之前修改的版本号一致

|-刷新gradle项目

1. 当存在一个远程分支的时候，可以直接切换的该远程分支下面，对应的本地仓库也有这个分支，可以在这个分支上创建新的分支，进行开发。

# 2018-4-19

1. Calendar类，simpleDateFormat类了解一下

把driver里debug.10的版本拉下来，然后合并

该driver版本，到debug.11

再测试一下api

提交api里面的修改

1. **public** **static** **void** main(String[] args) {
2. String dStr = "2001.12.12-08:23:21";
3. Date d = **null**;
4. SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy.MM.dd-HH:mm:ss");
5. **try** {
6. d = sdf.parse(dStr);
7. } **catch** (ParseException pe) {
8. System.**out**.println(pe.getMessage());
9. }
10. System.**out**.println(d);
11. System.**out**.println(d.getTime());
12. }

字符串"yyyy-MM-dd hh:mm:ss",其中:

yyyy : 代表年(不去区分大小写) 假设年份为 2015

    "y" , "yyy" , "yyyy" 匹配的都是4位完整的年 如 : "2015"

    "yy" 匹配的是年分的后两位 如 : "15"

    超过4位,会在年份前面加"0"补位 如 "YYYYY"对应"02015"

MM : 代表月(只能使用大写) 假设月份为 9

    "M" 对应 "9"

    "MM" 对应 "09"

    "MMM" 对应 "Sep"

    "MMMM" 对应 "September"

    超出4位,仍然对应 "September"

dd : 代表日(只能使用小写) 假设为13号

    "d" , "dd" 都对应 "13"

    超出2位,会在数字前面加"0"补位. 例如 "dddd" 对应 "0013"

hh : 代表时(区分大小写,大写为24进制计时,小写为12进制计时) 假设为15时

    "H" , "HH" 都对应 "15" , 超出2位,会在数字前面加"0"补位. 例如 "HHHH" 对应 "0015"

    "h" 对应 "3"

"hh" 对应 "03" , 超出2位,会在数字前面加"0"补位. 例如 "hhhh" 对应 "0003"

mm : 代表分(只能使用小写) 假设为32分

    "m" , "mm" 都对应 "32" ,  超出2位,会在数字前面加"0"补位. 例如 "mmmm" 对应 "0032"

ss : 代表秒(只能使用小写) 假设为15秒

    "s" , "ss" 都对应 "15" , 超出2位,会在数字前面加"0"补位. 例如 "ssss" 对应 "0015"

E : 代表星期(只能使用大写) 假设为 Sunday

    "E" , "EE" , "EEE" 都对应 "Sun"

    "EEEE" 对应 "Sunday" , 超出4位 , 仍然对应 "Sunday"

a : 代表上午还是下午,如果是上午就对应 "AM" , 如果是下午就对应 "PM"

Vendor2Vm

/\*if (end != null & start != null) {

schedulerValueList.add(new net.skycloud.nap.driver.model.unified.SchedulerValue.Builder()

.setType(SchedulerType.DATE)

.setValue(new net.skycloud.nap.driver.model.unified.DateValue.Builder()

//YYYY-MM-DD.hh:mm

.setStart(start.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(start.getMonth()) + "-" + start.timeForm(start.getDay()) + "." + start.timeForm(start.getHour()) + ":" + start.timeForm(start.getMinute()))

//YYYY-MM-DD.hh:mm

.setStop(end.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(end.getMonth()) + "-" + end.timeForm(end.getDay()) + "." + end.timeForm(end.getHour()) + ":" + end.timeForm(end.getMinute()))

.build())

.build());

} else if (start != null){

schedulerValueList.add(new net.skycloud.nap.driver.model.unified.SchedulerValue.Builder()

.setType(SchedulerType.DATE)

.setValue(new net.skycloud.nap.driver.model.unified.DateValue.Builder()

.setStart(start.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(start.getMonth()) + "-" + start.timeForm(start.getDay()) + "." + start.timeForm(start.getHour()) + ":" + start.timeForm(start.getMinute()))

.build())

.build());

} else {

schedulerValueList.add(new net.skycloud.nap.driver.model.unified.SchedulerValue.Builder()

.setType(SchedulerType.DATE)

.setValue(new net.skycloud.nap.driver.model.unified.DateValue.Builder()

.setStop(end.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(end.getMonth()) + "-" + end.timeForm(end.getDay()) + "." + end.timeForm(end.getHour()) + ":" + end.timeForm(end.getMinute()))

.build())

.build());

}\*/

/\*if (start != null){

schedulerValueList.add(schedulerValueBuilder

.setType(SchedulerType.DATE)

.setValue(dateValueBuilder

.setStart(start.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(start.getMonth()) + "-" + start.timeForm(start.getDay()) + "." + start.timeForm(start.getHour()) + ":" + start.timeForm(start.getMinute()))

.build())

.build());

} else if (end != null){

schedulerValueList.add(schedulerValueBuilder

.setType(SchedulerType.DATE)

.setValue(dateValueBuilder

.setStop(end.getYear() + "-" + net.skycloud.nap.driver.translator.unified.Utils.month2Num(end.getMonth()) + "-" + end.timeForm(end.getDay()) + "." + end.timeForm(end.getHour()) + ":" + end.timeForm(end.getMinute()))

.build())

.build());

}\*/

Vm2Vendor

/\* timeRangeObject.getElements().add(new TimeRangeElement.Builder()

.setType(TimeRangeElementType.ABSOLUTE)

.setValue(new AbsoluteValue.Builder()

.setStart(new net.skycloud.nap.driver.model.cisco.DateValue.Builder()

.setYear(start.get(Calendar.YEAR))

.setMonth(net.skycloud.nap.driver.translator.unified.Utils.num2Month((start.get(Calendar.MONTH) + 1) + ""))

.setDay(start.get(Calendar.DAY\_OF\_MONTH))

.setHour(start.get(Calendar.HOUR\_OF\_DAY))

.setMinute(start.get(Calendar.MINUTE))

.build())

.setEnd(new net.skycloud.nap.driver.model.cisco.DateValue.Builder()

.setYear(stop.get(Calendar.YEAR))

.setMonth(net.skycloud.nap.driver.translator.unified.Utils.num2Month((stop.get(Calendar.MONTH) + 1) + ""))

.setDay(stop.get(Calendar.DAY\_OF\_MONTH))

.setHour(stop.get(Calendar.HOUR\_OF\_DAY))

.setMinute(stop.get(Calendar.MINUTE))

.build())

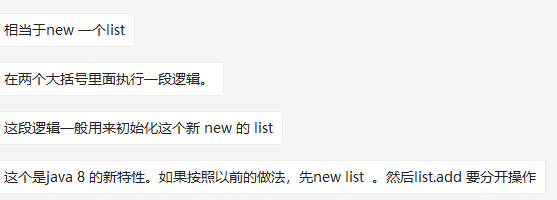
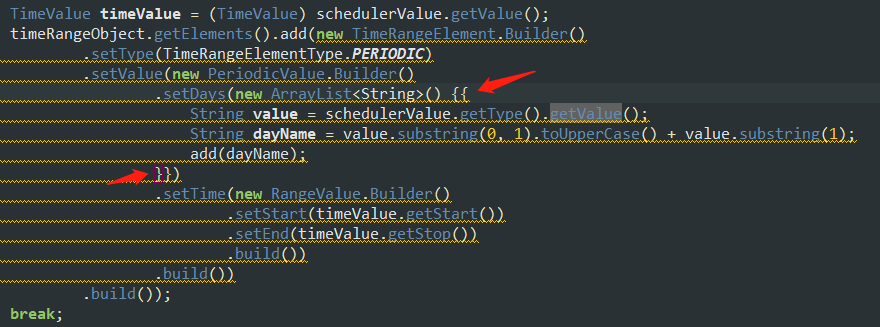
)

.build()

);\*/

1. TODO

|-lambda表达式，



# 2018-4-20

1. Lambda表达式

|- <https://blog.csdn.net/renfufei/article/details/24600507>

|- 用于匿名内部类

|- 用于