

# 实时规则引擎设计与实现

@Neeao

2013年12月26日

# 目录

- 需求
- 方案
- 实现
- 挑战

# 目录

➤ 需求

➤ 方案

➤ 实现

➤ 挑战

# 一大波扫号的出现

一般处理流程：

- 分析日志
- 制定防御规则
- 找相关Domain开发防御规则

# 防御规则

5分钟内IP登陆超过100次，封IP60分钟

如何实现？

# 硬编码

```
if time<=5m and count(8.8.8.8)>=100:  
    block ip 60m
```

# 硬编码--成本

每天成功登陆:2W UID

账号有钱的UID:10%

UID平均账号余额:100元

每天存在威胁资金量:

$$2W * 10\% * 100 = \mathbf{20W}$$

# 硬编码--成本

开发上线时间:

1 sprint(开发+测试+上线)=14天

$$14 * 20 = \mathbf{280w}$$

如何解决？



# 硬编码--改进版

将参数放入配置项,需要时更新配置数据:

`db_gap_time=5m`

`db_times=100`

`db_block_time=60m`

`if time<=db_gap_time and count(8.8.8.8)>=db_times:`

`block ip db_block_time`

# 硬编码--改进版

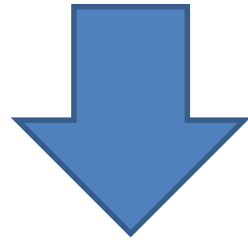
一键秒刷，立即生效

新的问题：

- 新的规则怎么添加？
- 继续重复硬编码的思路？
- 有没有更好的方法？

# 终极大招

将规则逻辑从应用中分离  
应用提交数据至规则逻辑得到结果



# 规则引擎

# 规则引擎

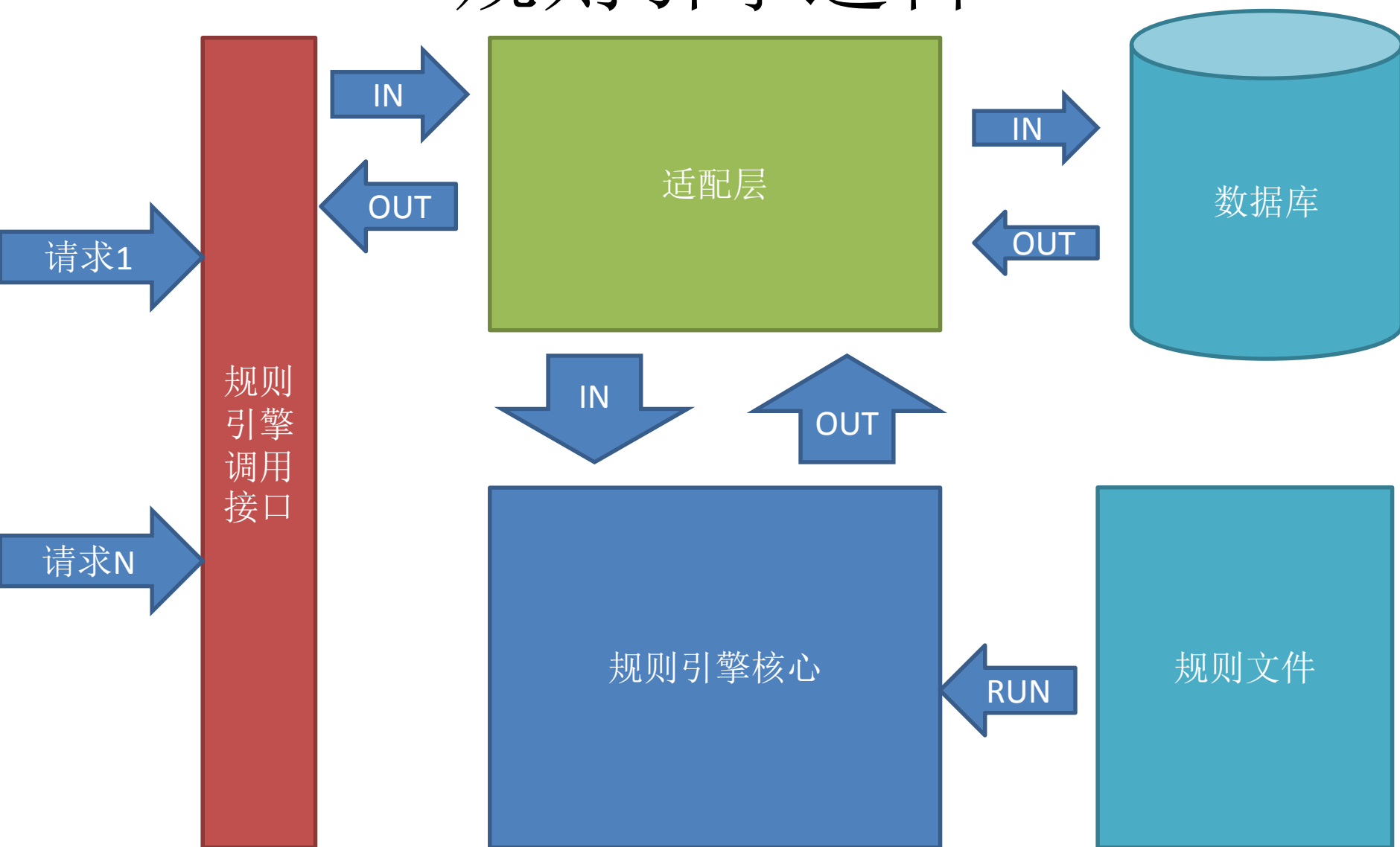
定义：

规则引擎由推理引擎发展而来，是一种嵌入在应用程序中的组件，实现了将业务决策从应用程序代码中分离出来，并使用预定义的语义模块编写业务决策。接受数据输入，解释业务规则，并根据业务规则做出业务决策。

应用背景：

- 提高效率，解决复杂的业务规则
- 规则经常变化，IT系统根据规则快速、低成本变化
- 为了快速、低成本的更新，业务人员应能直接管理IT系统中的规则，不需要程序开发人员参与。

# 规则引擎逻辑



# 目录

➤ 需求

➤ **方案**

➤ 实现

➤ 挑战

# 功能

- 业务逻辑与应用分离
- 方便调用接口
- 性能与硬编码差别不大
- 轻量级，方便开发维护
- 友好的规则编辑界面
- 现有硬编码规则的集成

# 为什么不用开源产品

- 性能原因
- 队列特性、实时性无法满足
- 现有Java规则逻辑的很难复用
- 学习维护成本高



# 规则逻辑脚本语言

## Groovy

优点：

- 基于JVM（Java虚拟机）
- 借鉴Python、Ruby和Smalltalk的许多强大的特性，支持DSL，语法简洁
- 直接调用现有Java的类库，Java无缝集成
- 方便的实例化接口



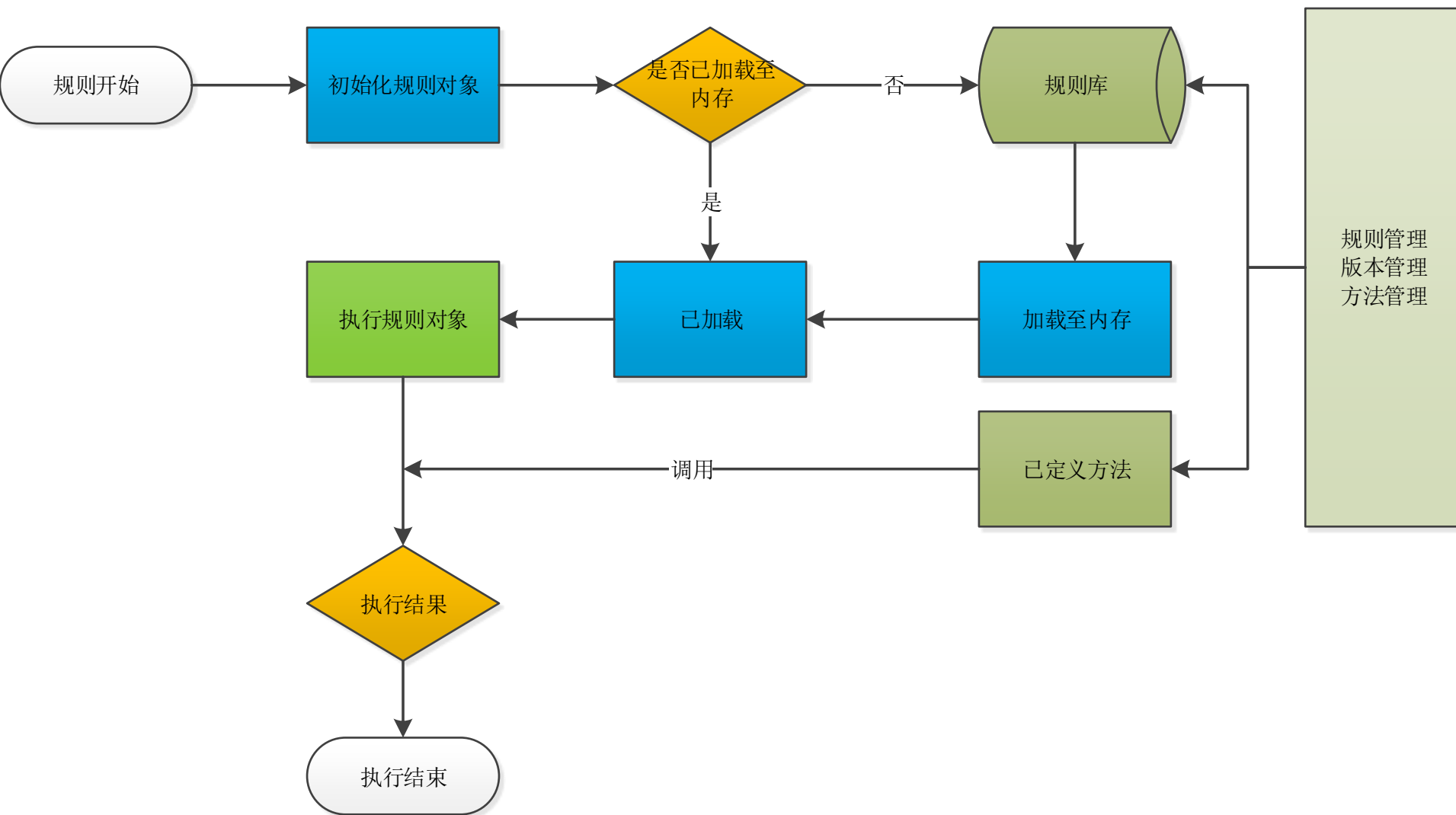
# 性能方面



# 规则管理



# 规则流程



# 目录

- 需求
- 方案
- **实现**
- 挑战

# 实现

- Groovy脚本加载为Java对象
- 接口参数支持多参数、任意类型
- 规则脚本执行流程
- Sandbox
- 规则Web管理及测试
- 规则调用流程

# Groovy脚本加载为Java对象

```
/**
 * 初始化规则对象
 * @param ruleName
 * @return
 * @throws IllegalAccessException
 * @throws InstantiationException
 */
public static Rule initRule(String ruleName, String ruleContent, List<RuleParameter> parameterList) {
    Object ruleObj=null;
    try{
        String content=Sandbox.formatScript(ruleName,ruleContent,parameterList);
        ClassLoader loader=RuleBeanFactory.class.getClassLoader();
        GroovyClassLoader groovyloader = new GroovyClassLoader(loader);
        Class<?> newClazz = groovyloader.parseClass(content);
        ruleObj = newClazz.newInstance();
        return new Rule(ruleContent,ruleObj);
    }catch(Exception e){
        LOG.error(e.getMessage());
    }
    return null;
}
```

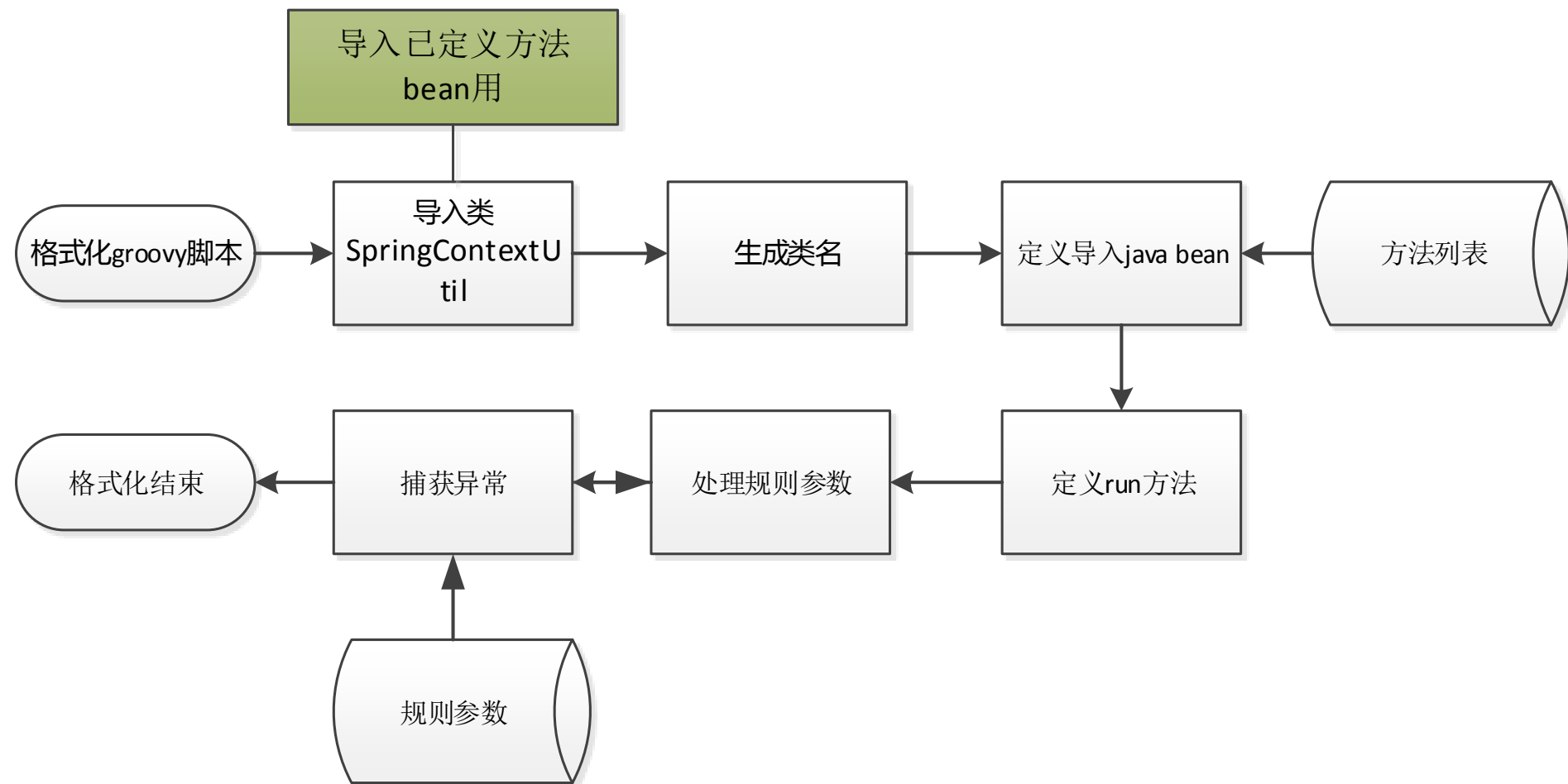
# 接口参数支持多参数、任意类型

```
/**
 * 执行方法
 * @param args
 * @return 返回值统一为整形
 * @throws Exception
 */
public Object runRule(Map<String, Object> args) {
    >> Object returnObj=null;
    >> try {
    >>     >> Method m = ruleObj.getClass().getMethod(Sandbox.groovyMethodName, Map.class);
    >>     >> returnObj=m.invoke(ruleObj, args);
    >> } catch (Exception e) {
    >>     >> LOG.error(e.getMessage());
    >> }
    >> return returnObj;
}
```

输入参数			
 添加  修改  删除  刷新			
参数名称	数据类型	描述	测试值
userIP	String	用户IP地址	<input type="text"/>



# 规则脚本执行流程



# Groovy--SandBox

- 默认导入包定义
- SecureASTCustomizer
  - 闭包: {}
  - 包导入: import
  - 包定义: package
  - 方法定义: public static void main
  - 方法返回值定义: return Integer/Double/String
  - 运算符定义: PLUS/MINUS/MULTIPLY/ DIVIDE
  - <http://groovy.codehaus.org/Advanced+compiler+configuration>
  - <http://groovy-sandbox.kohsuke.org/>

# 规则Web管理及测试

欢迎使用

方法列表 ×

内置方法

+

添加

✎

修改

🗑

删除

🔄

刷新

<input type="checkbox"/>	方法名称	所属Bean	描述
<input type="checkbox"/>	isIPWhiteList	ipPolicy	调用：ipPolicy.isIPWhiteList() /** * 指定IP是否在白名单列表中 * @param ip * @return * true:存在 * false:不存在 */
<input type="checkbox"/>	write	ruleLog	调用： ruleLog.write(String inData,S /** * 记录规则执行日志 * @param inData 许记录的数 * @param remark 简单说明 */
<input type="checkbox"/>	timesCheck	unitTimeToTimes	调用： unitTimeToTimes.timesChec /** * 检查目标字段是否触发规则 * @param target 目标 * @param gapTime 单位时间 * @param maxTimes 最大次

10 ▾

⏮ ⏪






第 1 共1页

⏩ ⏭ 🔄





# 规则Web管理及测试

欢迎使用		规则列表 ×	规则版本:passport登陆反欺诈 ×
规则版本			
 添加  拷贝  刷新			
<input type="checkbox"/>	规则名称	版本号	版本简介
<input type="checkbox"/>	passport登陆反欺诈	2	Default rule (
<input type="checkbox"/>	passport登陆反欺诈	1	IP单位时间内 1.白名单IP不 2.10秒钟登陆

# 规则Web管理及测试

输入参数				功能描述
<div> 添加  修改  删除  刷新</div>				<div> 保存</div>
参数名称	数据类型	描述	测试值	IP单位时间内登陆次数。 1. 白名单IP不受影响; 2. 10秒钟登陆超过2次, 封停1分钟
userIP	String	用户IP地址	<input type="text"/>	

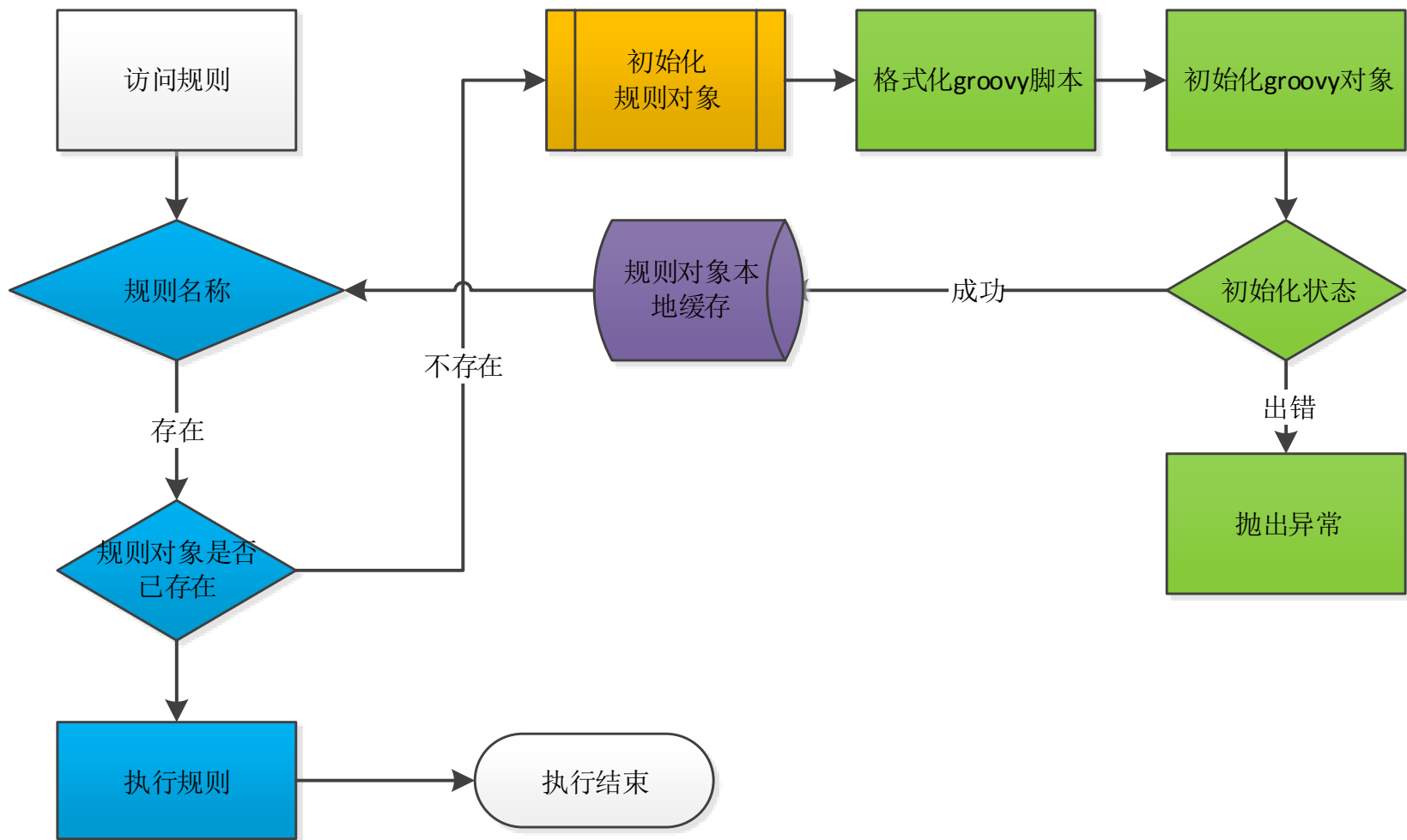
规则脚本

 保存更改  语法检查  执行测试  设置为可用

```
1 use unitTimeToTimes;
2 use ruleLog;
3 use ipPolicy;
4
5 String ip=userIP;
6
7 //白名单IP不做检查
8 if(ipPolicy.isIPWhiteList(ip)){
9     return '0';
}
```

Log

# 规则调用流程



# 目录

- 需求
- 方案
- 实现
- **挑战**

# 挑战

- 应用接入
- 规则接口数据定义
- 更友好的规则编辑界面



Q/A