风控分析文档

文档作者: 付乐乐

创建日期: 2020-02-10

更新日期:

文档编码：

当前版本： 1.0



## 文档控制

**变更记录**

485

| 日期 | 作者 | 版本 | 变更参考 |
| --- | --- | --- | --- |
|  |  |  |  |
| 2020/01/11 | 付乐乐 | 1.0 | 原始文档 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**审核**

| 姓名 | 职位 |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**分发**

| 拷贝编号 | 姓名 | 位置/岗位 |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**备注:**

出于文档管理的目的，如果您收到了本文档的电子版本，请打印出来并在封面的相应位置写上您的名字。

出于文档管理的目的，如果您收到了本文档的纸介质版本，请在封面写上您的名字。

目录

文档控制 2

1. 概述 4

1.1. 概述 4

1.2. 文档目的 4

1.3. 审批控制 4

2. 系统分析 5

地址参数 5

login场景 5

代码分析 7

3. 未结事项与己结事项 10

3.1. 未结事项 10

3.2. 己结事项 10

## 概述

### 概述

### 文档目的

本文档资料的目的是帮助本人回顾和巩固所学知识。

### 审批控制

|  |  |  |
| --- | --- | --- |
| **部门** | **审批人** | **签名及日期** |
|  |  |  |
|  |  |  |
|  |  |  |

## 系统分析

#### 地址参数

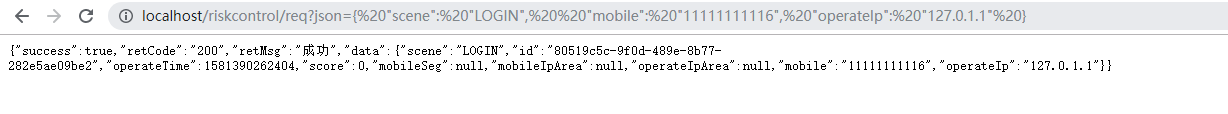
[http://localhost/riskcontrol/req?json=](http://localhost/riskcontrol/req?json=%7b%20%22scene%22:%20%22LOGIN%22,%20%22operateTime%22:%20%222020-02-10%22,%20%22mobile%22:%20%2211111111112%22,%20%22operateIp%22:%20%22127.0.1.1%22%20%7d){"scene": "LOGIN","mobile": "11111111111","operateIp": "127.0.0.1"}

scene：场景

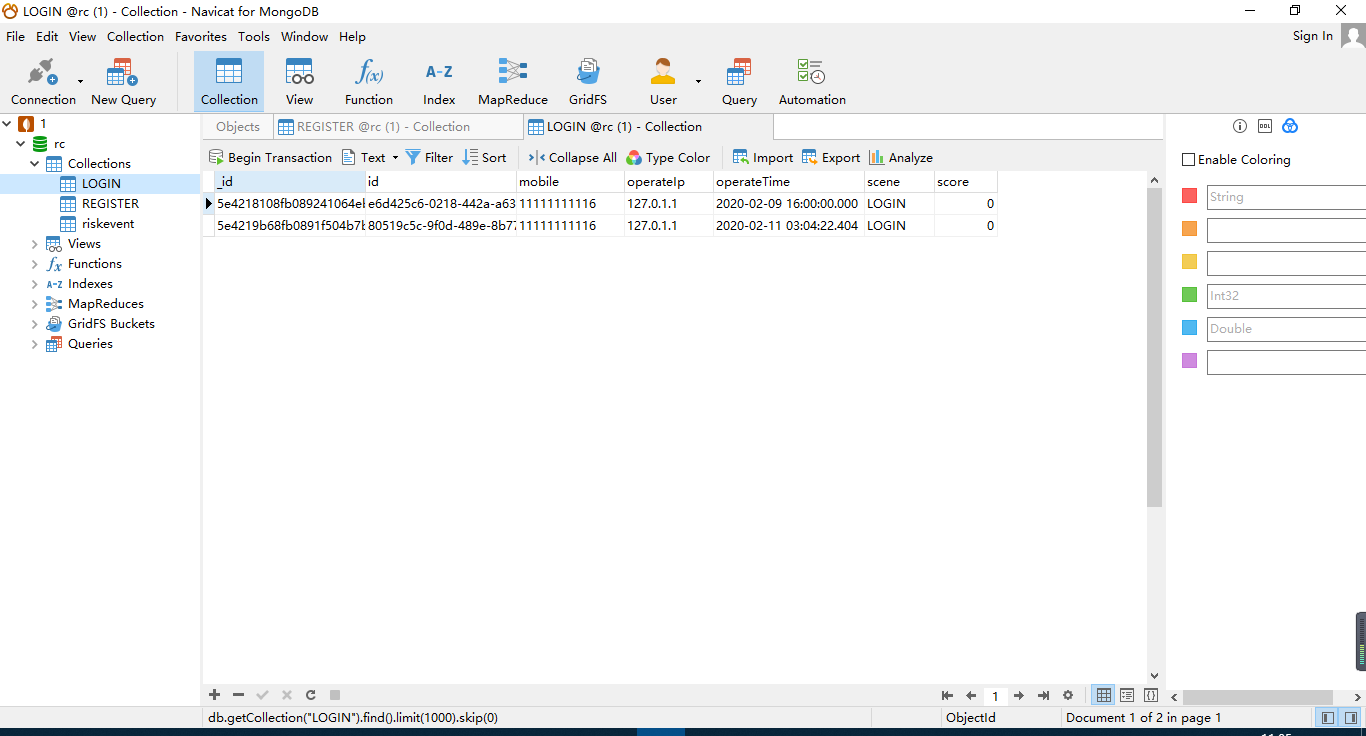
mobile: 电话

operateIp: IP地址

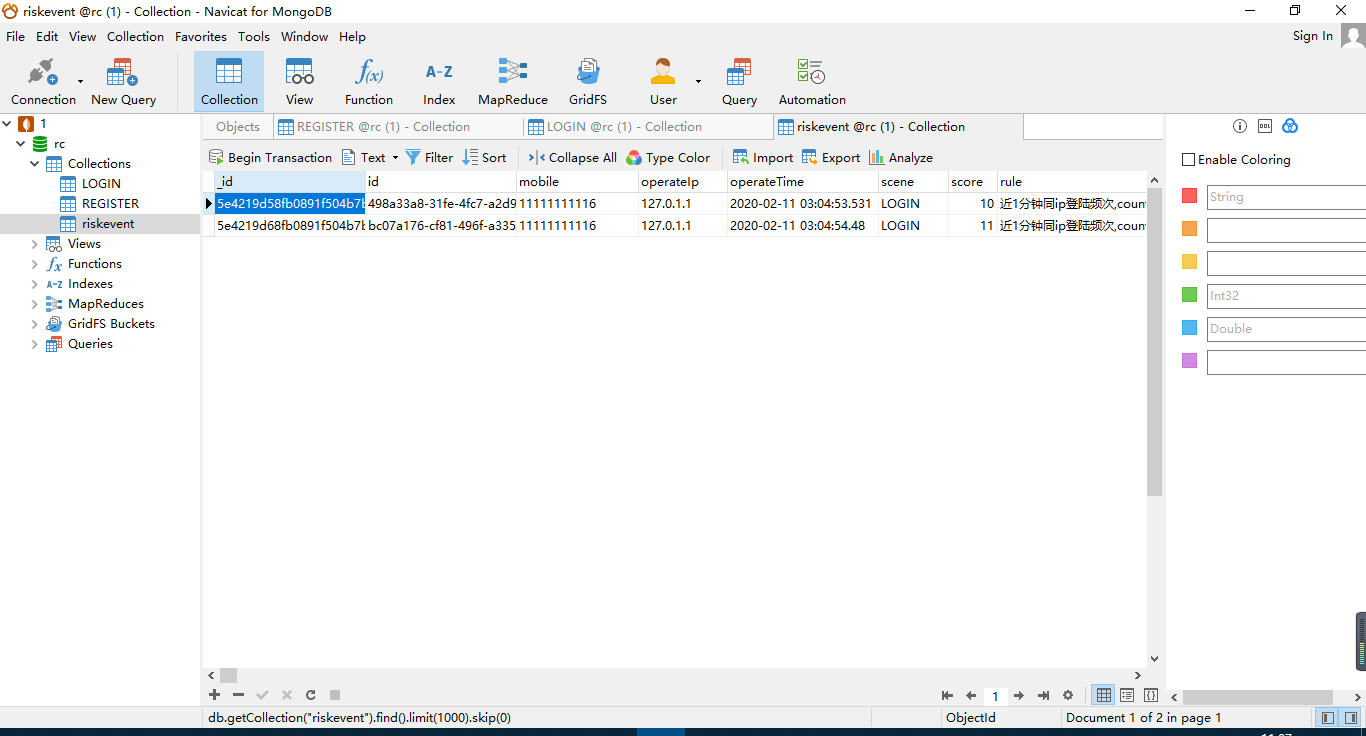
#### login场景



Mongodb

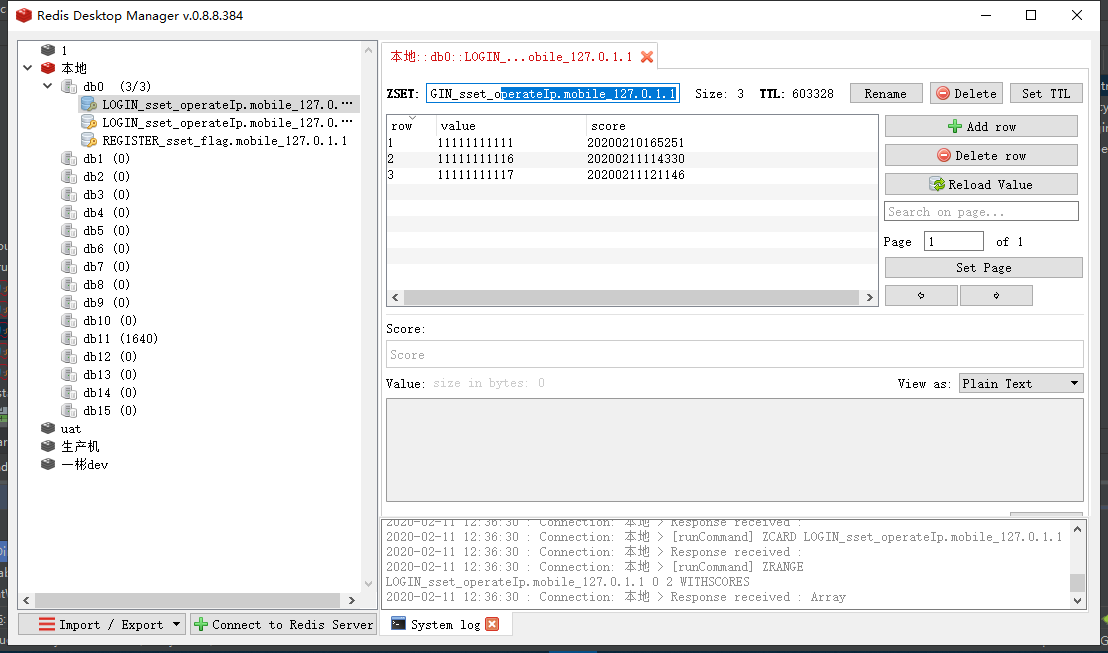


LOGIN负责记录登录事件发生的次数，计算某时间内登录频次的时候从这里取出，计算。



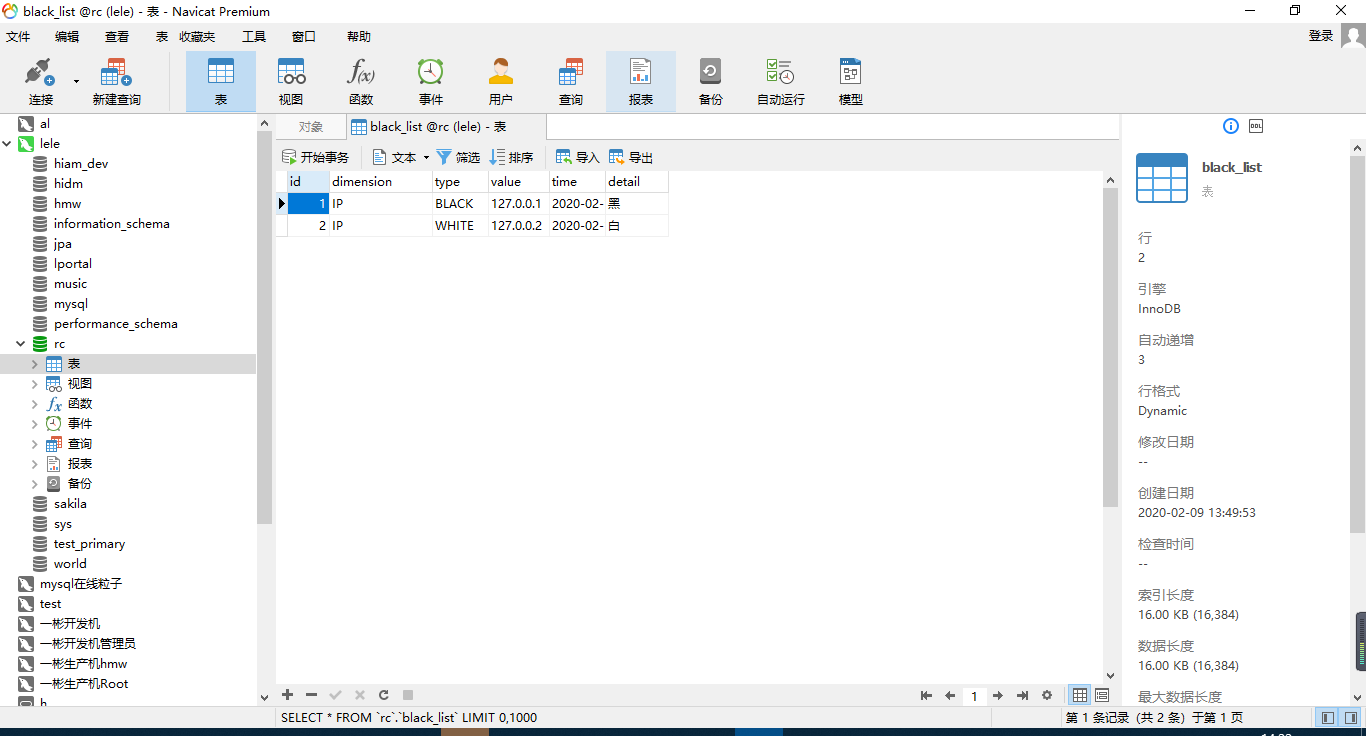
Riskevent表负责插入风险事件，所有分值不为1的事件都会记录。

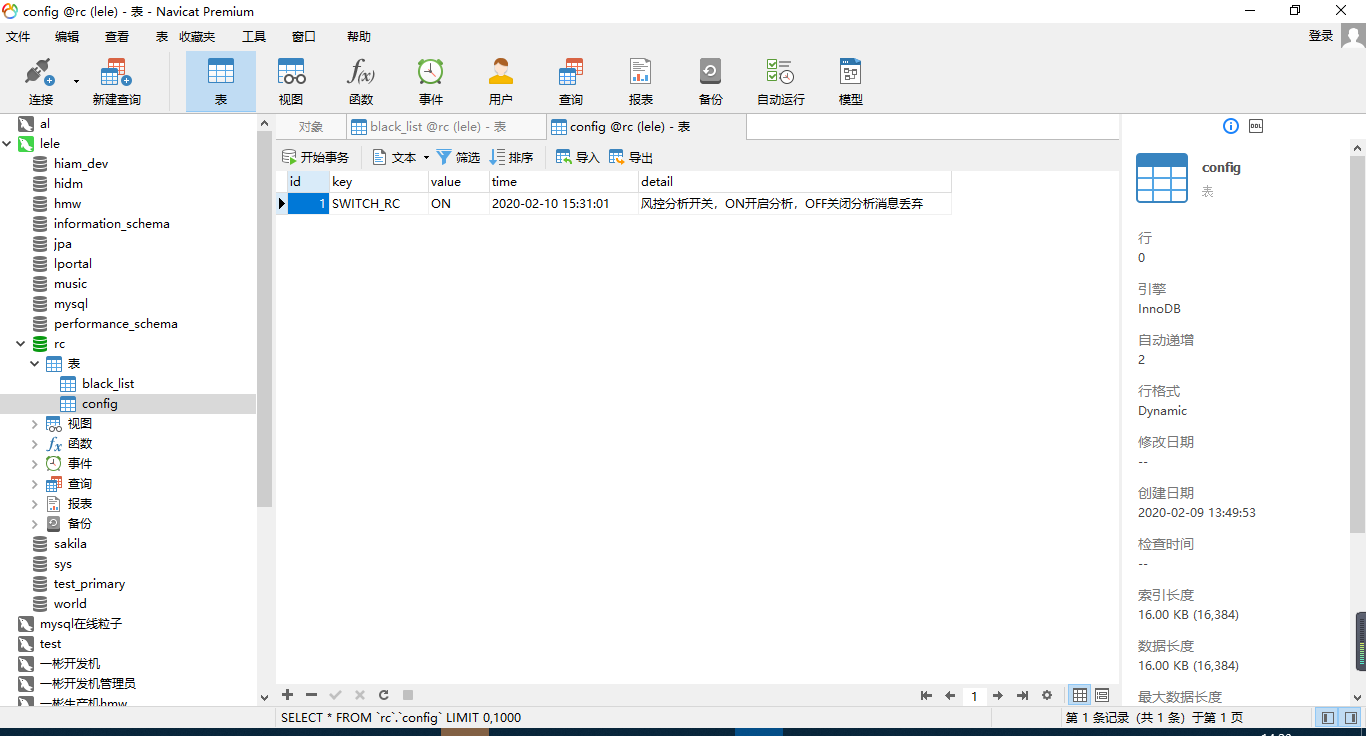
Redis



Redis根据发生事件实体类中所有字段和设定的某一个字段的值作为key，根据设定的另一维度的值作为value，操作时间作为score。只有key或者value不同时才插入新数据，计算某段时间同一ip有多少不同的手机号登录就是从redis中取得数据。

Mysql



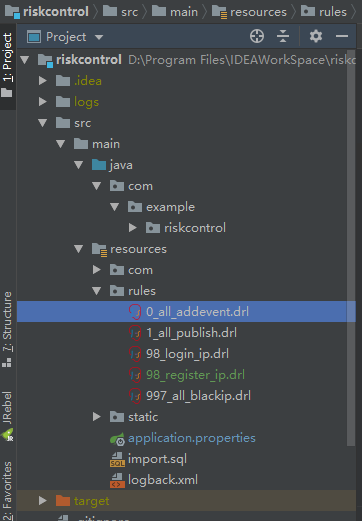


Black\_list:黑白名单表；

Config：风控分析开关，on代表会分析并打分，off是关闭分析，分数为0。

#### 代码分析

1. Rules文件



Rules文件夹中放的是规则文件，采用java语言编写，例

package rules;  
  
import com.example.riskcontrol.model.Event  
import com.example.riskcontrol.service.DimensionService  
  
global DimensionService dimensionService  
  
rule "0\_all\_addevent"  
 salience 0  
 lock-on-active true  
 when  
 event:Event()  
 then  
 dimensionService.insertEvent(event);  
  
end

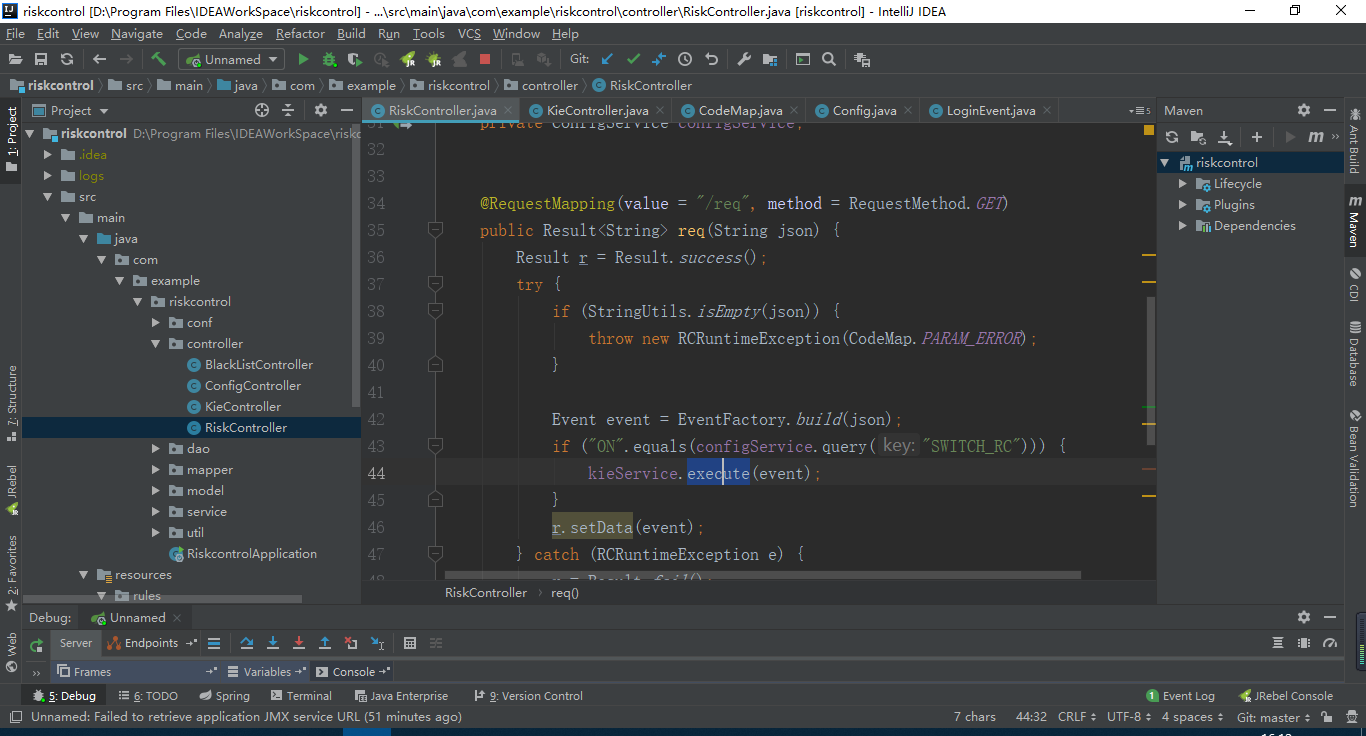
salience代表该规则的执行顺序，数字越大，执行越早。

lock-on-active true：表示该规则只执行一次

when event:Event()：表示当前规则对那个事件生效，比如LoginEvent()就是对登录事件生效，该事件是自己定义的实体类。

Then 后面跟的是需要执行的操作，是自己编写的逻辑。

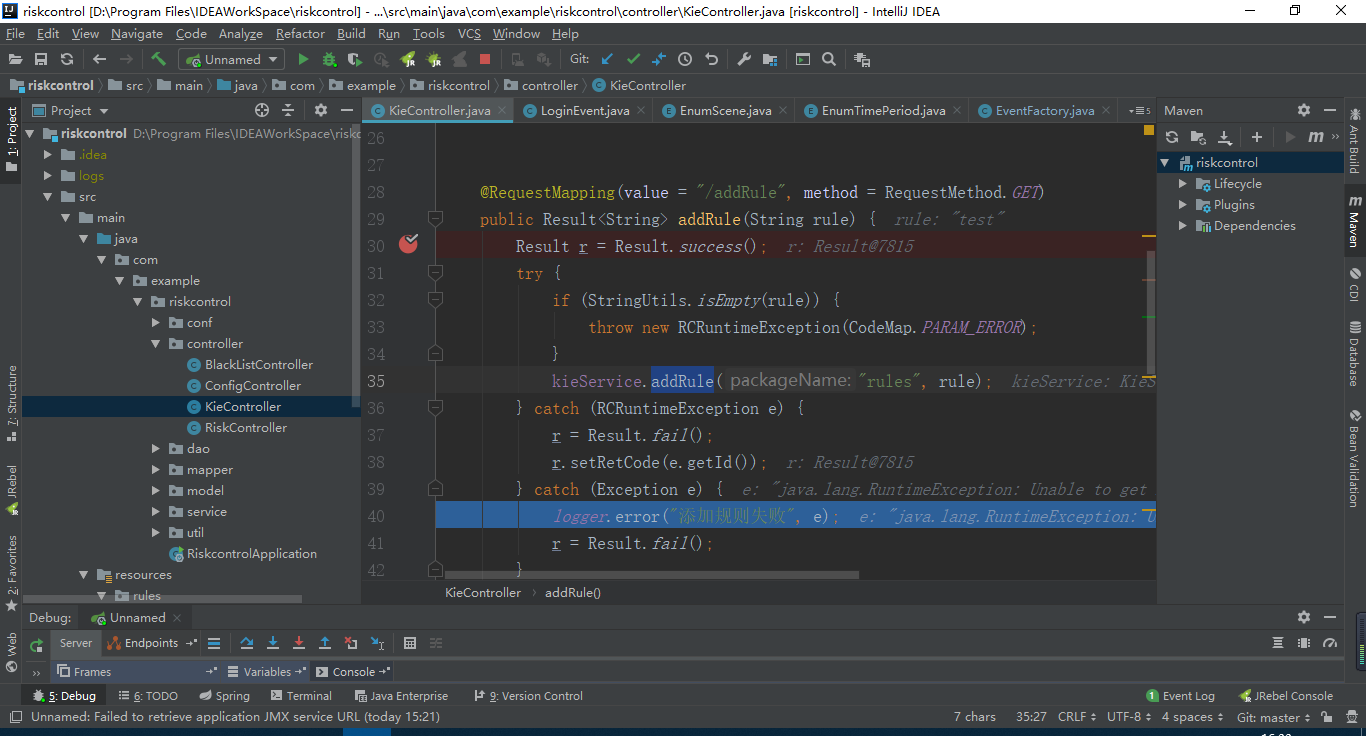
1. controller文件



RiskController是分析事件的入口，判断是否开启事件分析，

Event event = EventFactory.build(json)：按传入的事件类型创建事件。

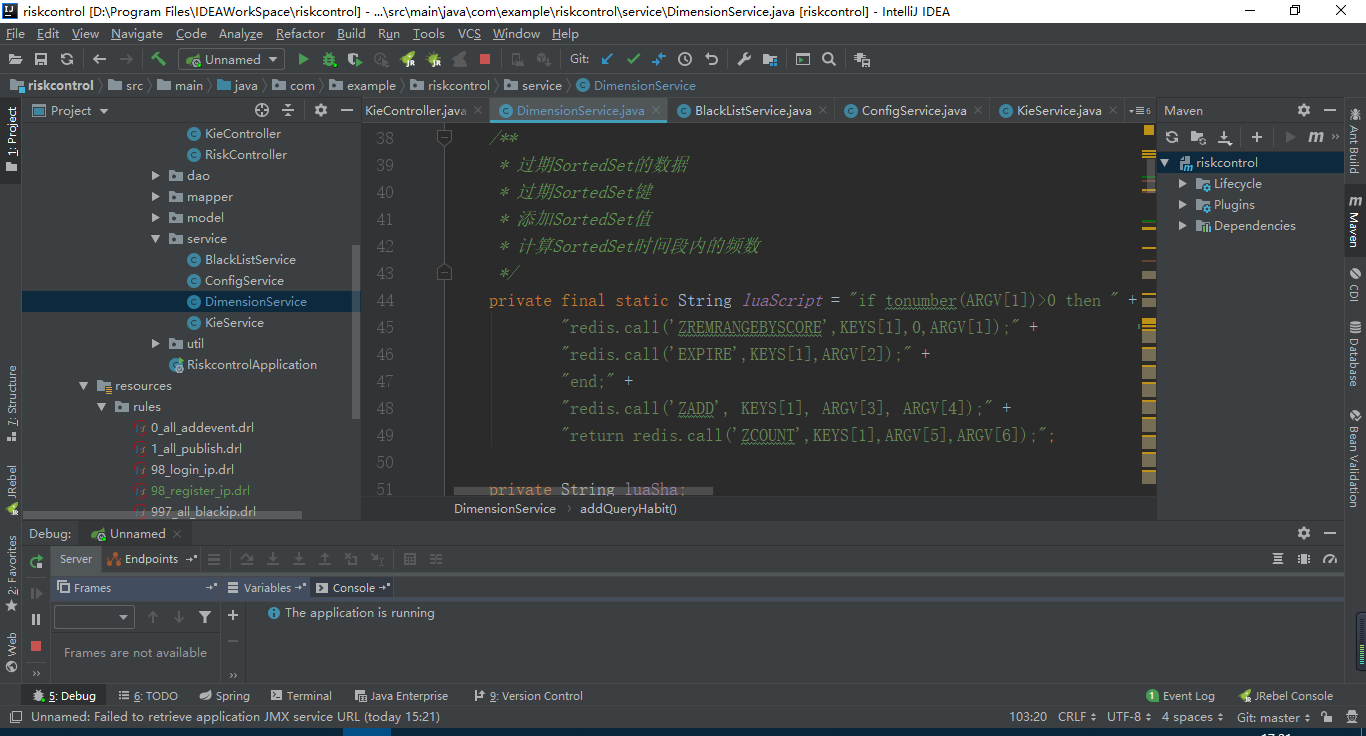
kieService.execute(event);表示执行规则文件。



KieController：规则控制器，主要是规则的增删。

首先在rules文件夹中创建规则文件，然后根据方法地址传参即可对规则进行增删，删掉的规则将不再执行，但是文件还在，要是需要把删掉的规则重新加上调用addRules即可。例：<http://localhost/kie/addRule?rule=98_register_ip>

1. service文件



DimensionService：该service主要是对数据进行处理，数据的增删改查以及分析包括redis以及mongodb的数据。

KieService：该service是对规则集进行的处理，增删改查以及执行规则集。

#### 总结

该代码的开放性很大，从什么维度进行风险分析、用什么规则分析都可以自定义实现，多维度分析时只需要确定维度之间的优先级，然后按不同维度创建不同的规则文件，并按优先级排序执行即可，最终结果按照打分来判断是否有风险。不同的场景也可以创建不同的实体类来区分规则文件。

## 未结事项与己结事项

### 未结事项

| 序号 | 事项 | 解决方案 | 责任人 | 计划完成日期 | 实际完成日期 |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### 己结事项

| 序号 | 事项 | 解决方案 | 责任人 | 计划完成日期 | 实际完成日期 |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |