Criteria Based Authorization Model

——基于条件的权限模型（CBAM）

Author ： Yaohui Wu

Date ： 2014.9.22

Version : 1.0

## 1 综述

本文档只讨论权限中的授权。

以下表格对比了Spring Security 与 Apache Shiro两个权限框架。

|  |  |  |  |
| --- | --- | --- | --- |
| **方面** | **框架** | **Spring Security** | **Apache Shiro** |
| 原理 | | 访问控制表（Access Control List）ACL | 权限表达式  Action : Type: Instance |
| 缺点 | | 1需要列举所有的资源，AC数据量~=角色数X资源数  2 配置较复杂 | 1表达式仅支持通配符，不支持更复杂的条件过滤  2 不提供数据访问操作，需要自己实现 |
| 优点 | | 框架中包含权限数据访问操作接口 | 权限数据量较少 |
| 可优化 | | 否 | 可以做数据存储，访问的优化 |

Table 权限框架对比

通过对比可以看到，ACL模型对每一个资源都要配置相应的权限，粒度太细而且不灵活，以一个10000角色，10万个资源对象的系统为例，权限表数据达到10亿条（至少）；Shiro更进了一步，可以对一定范围的资源进行授权，但是表达式的语义过于简单。

于是，我们需要一种既能够精确定位到某一个资源，又能对符合特定条件资源进行授权的权限模型。

这就是基于条件的权限模型。

## 2 数据结构

### 2.1用户（User）

如果系统中已存在用户表，则可以省略。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **含义** | **是否主键** | **约束** |
| id | varchar(32) | 用户ID | √ |  |
| name | varchar(100) | 用户名 |  |  |
| org\_id | varchar(32) | 机构ID |  |  |

Table User

注：其中红色标注的字段为业务系统特有，非本权限模型定义。

### 2.2动态角色（Dynamic Role）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **含义** | **是否主键** | **约束** |
| id | varchar(32) | 角色ID | √ |  |
| name | varchar(255) | 角色名称 |  |  |
| user\_criteria | text | 属于该角色的用户条件表达式 |  |  |

Table Role

### 2.3权限（Permission）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **含义** | **是否主键** | **约束** |
| id | varchar(32) | 权限ID | √ |  |
| action | varchar(100) | 操作名称 |  |  |
| object\_type | varchar(255) | 对象类型 |  |  |
| criteria | text | 对象条件表达式 |  |  |

Table Permission

### 2.4 授权表（Authorization）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **含义** | **是否主键** | **约束** |
| id | varchar(32) | 授权ID | √ |  |
| role\_id | varchar(32) | 角色ID |  |  |
| permission\_id | varchar(32) | 权限ID |  |  |
| update\_time | datetime | 更新时间 |  |  |
| authorized\_by | varchar(32) | 授权动作的发起者 |  |  |

Table Authorization

## 3 条件表达式

用于对JavaBean（包含getter）进行过滤，表达式中的值必须是标量（简单起见）。

### 3.0 基本语法

包含SQL的where子句的子集。

数据类型：整数(INT)，浮点数(FlOAT)，布尔变量(BOOLEAN)，字符串(STRING，空值（null）；

运算符：>、>=、=、!=、<、<=、in、like；

逻辑运算：and、or；

支持小括号，改变优先级，嵌套。

当表达式为 \* 时，表示匹配所有对象。

### 3.1角色表达式

对用户进行选择过滤，决定符合何种条件的用户属于该角色。表达式中的标示符必须是User对象（JavaBean）中的属性。

示例：

年龄在18周岁以下的用户：age < 18；

机构A中的用户：org\_id = ‘org\_a’；

机构A中的管理员：org\_id = ‘org\_a’ and user\_type = ‘2’ (假设2是管理员类型)；

所有的用户： \*。

### 3.2 权限表达式

Action ： ObjectType ：Criteria

操作 对象类型 对象条件

其中Action可以是create，delete，update，view，当为\*时表示任何操作；

ObjectType是类全名，当为\*时表示任何类型；（类名改变了如何？）

Criteria是条件表达式，当为\*时匹配所有类型制定的对象。

其中的标示符必须ObjectType(JavaBean)存在的属性。

示例：

删除机构A中的所有文档：

delete：com.pekall.mdm.Document:\*

分发机构A中管理员M上传的文档：

deliver:com.pekall.mdm.Document:upload\_by = ‘M’

查看所有资源：

view:\*：\*

## 4 技术细节

## 5 案例

现有一个树形的组织机构，每个机构存在一个或以上的管理员。每个管理员的权限有以下规则：（以机构A的管理员M为例）

1. M可以增加、删除、修改、查看、分发机构A以及A的子孙机构的所有资源对象；
2. M可以查看、分发机构A的祖先机构的所有资源对象。



我们看一下采用CBAM模型时M的权限数据：

### 4.1 用户数据

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **org\_id** |
| user\_01 | 管理员A | org\_a |

Table 管理员M的用户数据

### 4.2 角色数据

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **user\_criteria** |
| role\_01 | 机构A的管理员 | org\_id = ‘org\_01’ |

### 4.3 权限数据

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **action** | **object\_type** | **criteria** |
| permission\_01 | \* | \* | org\_id in (‘org\_a’,’org\_c’,) |
| permission\_02 | view | \* | org\_id in (‘org\_r’) |
| permission\_03 | deliver | \* | org\_id in (‘org\_r’) |

注：红色标注的单元格字符串可能很长，可以进行优化（TODO）。

### 4.4 授权数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **role\_id** | **permission\_id** | **update\_time** | **authorized\_by** |
| 01 | role\_01 | permission\_01 |  |  |
| 02 | role\_01 | permission\_02 |  |  |
| 03 | role\_01 | permission\_02 |  |  |