1. **概述**
2. **实验环境**

后端使用Python的Django 的1.10.2版本作为Web Server；

前端使用Chrome等浏览器实现浏览，网址为<http://osvt.net:3000/static/>

1. **工具实现技术**

本系统在实现上，通过Python的Django提供Restful的API接口，前端通过AJAX来通过API接口来获得所需的信息（主要为JSON），并通过HTML5、JQuery、Bootstrap等技术来展示。

## 3.1 后端

后端采用Python的Django框架为前端提供API。

Django为前端提供的API接口包括（举例）：

1. Get, http://osvt.net:3000/tenants

介绍：Get方法，可以获得所有的“租户”，每个租户有一个ID和一个Name。

1. Get, http://osvt.net:3000/tenants/<TENANT\_ID>

介绍：Get方法，获得TENANT\_ID对应的metadata.json

1. Get, http://osvt.net:3000/tenants/<TENANT\_ID>/policies/<POLICY\_NAME>

介绍：Get方法，获得TENANT\_ID的某个Policy

1. Post, http://osvt.net:3000/tenants/<TENANT\_ID>

介绍：Post方法，为一个TENANT设置metadata.json

1. Post, http://osvt.net:3000/tenants/<TENANT\_ID>/policies/<POLICY\_NAME>

介绍：Post方法，为一个TENANT设置某个Policy

1. Get, http://osvt.net:3000/tenants/<TENANT\_ID>/users

介绍：Get方法，获得某个TENANT的所有User

1. Get, http://osvt.net:3000/tenants/<TENANT\_ID>/users/<USER\_NAME>/commands/<COMMAND>

介绍：Get方法，以属于<TENANT\_ID>的<USER\_NAME>的身份执行命令<COMMAND>并获得执行结果。

## 3.2 前端

在前端上，本系统主要采用了HTML5、JQuery、JQuery-Confirm来实现与API的通信与内容的生成，采用了Bootstrap来作为CSS框架，并且采用了zTree这个JS库来实现Metadata的树状策略结构的展示。为了美观，还引入了第三方库，dropdown-enhancement。

为了提升安全性，本系统支持采用登录的方式来以某个租户的身份登录。

**4. 工具的展示效果和功能测试结果**

## 4.1租户登录/用户列表页

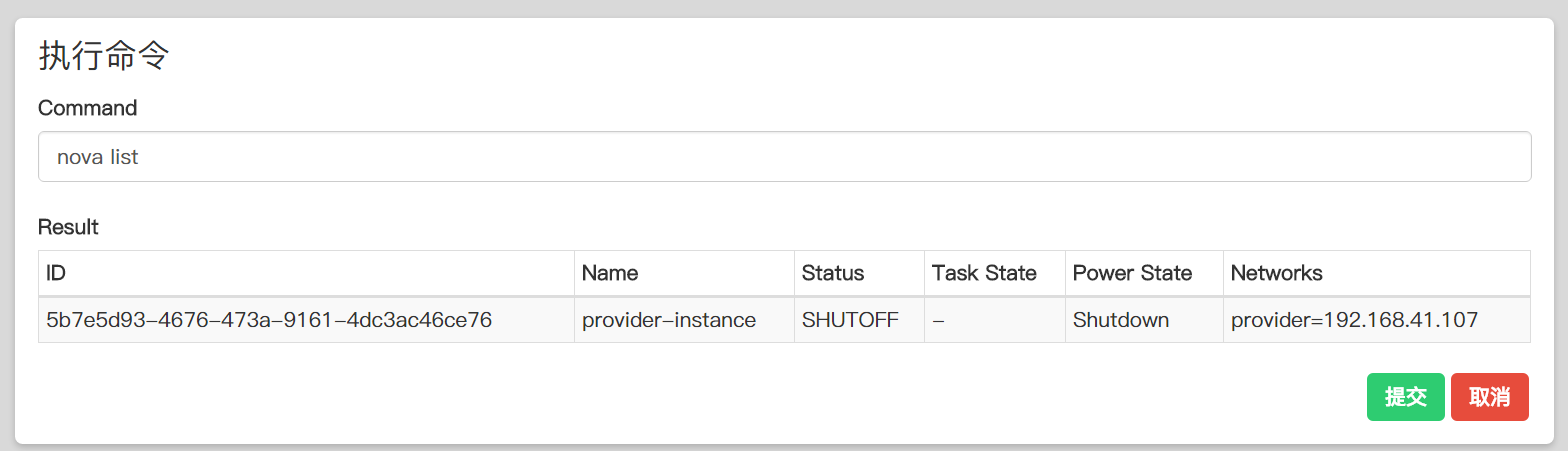
在本模块中，用户首先面对的是租户登录的页面，用户可以通过用户名密码登录，如果错误会有错误提示，成功的话，用户可以选择1）进入用户列表，或2）进入Metadata策略树编辑页面。



比如此时登录到租户“企业A”中，可以查看此租户下的用户列表。



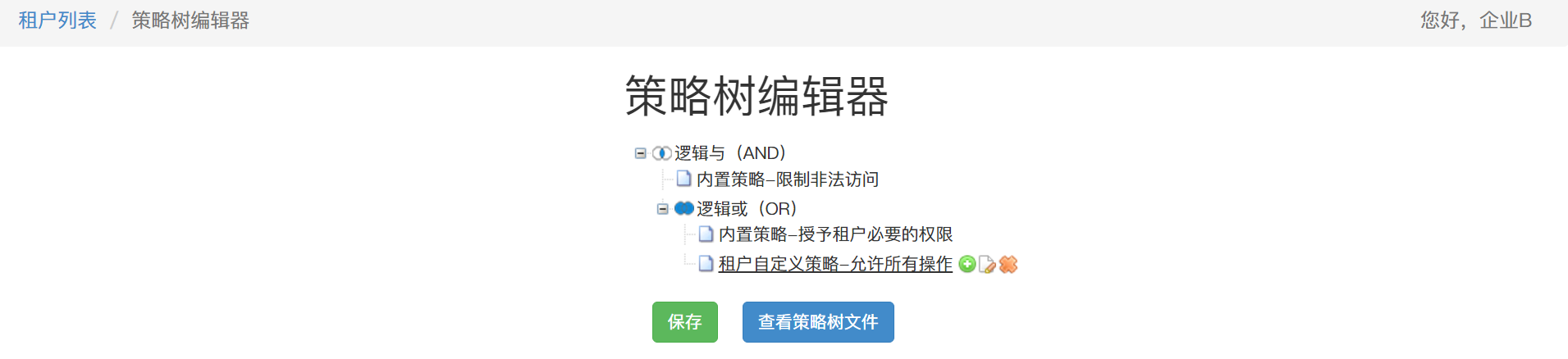
点击每个用户，可以以此用户身份执行命令并获得回显，比如下图中执行的是“Nova list”命令。



## Metadata策略树页面

首先程序从自身url中获取user\_id，并用AJAX获取到Metadata.json内容后，首先把自带的metadata.json格式递归转为zTree控件要求的格式（最后保存时类似），并显示，如下图（对于AND类策略与OR类策略图标不同）。父节点起到的必然是逻辑上的与，或逻辑上的或的功能。

此外，在页面顶部，会显示当前的租户身份，以及用户可以通过点击方便地后退到上一个页面。



对于每个节点有增加按钮，按钮，和删除按钮。在增加节点时，如果它是父节点的第一个子，首先修改父节点的type为op-and，然后把父节点content的\*.json赋给这个子节点，然后把父节点的content清空并换成子节点的name；如果它如果不是父节点的第一个子，就把子节点的name加到父节点的content末尾。删除节点时类似。

双击任意节点或点击编辑按钮，会弹出修改的对话框。

如果type是op-and、op-or、all-pass、all-forbid，即父节点，就禁止修改content防止造成不必要的麻烦；或者如果用户不是管理员，而且这个策略是“内置”的，就会被禁止修改如下图。



如果是子节点，而且不是内置策略，用户就可以编辑其策略，如下图。



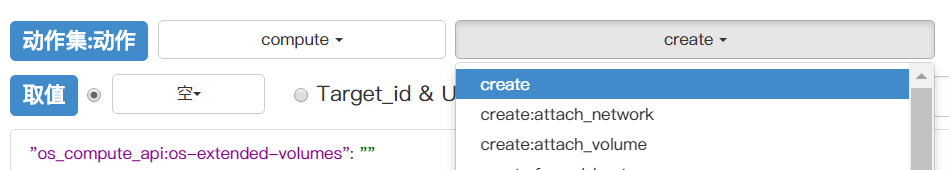
叶节点范例

## 4.3 Policy编辑器



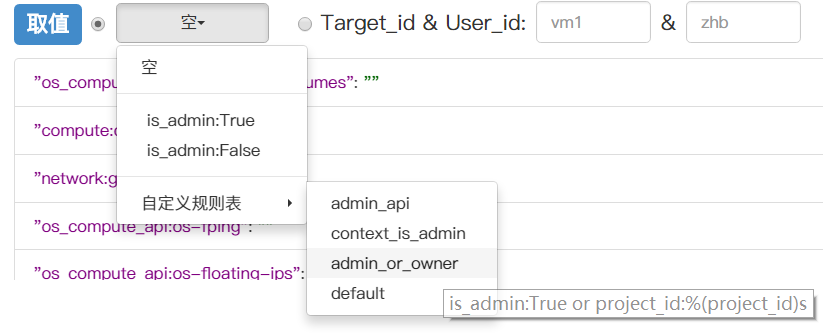
首先，程序通过AJAX获取到Policy.json文件，解析并展示在上下两个列表中，并可以通过点击右侧的垃圾桶图标实现删除。用户可以通过上下两个操作的部分来自行添加规则。

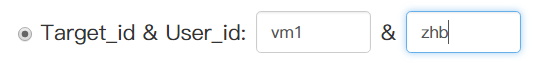
在“动作集”菜单加载后，“动作”的下拉菜单也会随之改变，如下图。

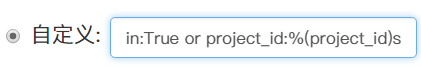


此外，在添加的时候，如果有同名的策略/规则，会问是否要覆盖，覆盖的话就刷新列表。

在设置“取值”时，可以从下拉菜单中选择，也可以设置Target\_ID与User\_ID，还可以自行设置公式，如下图。







在“自定义规则表”的部分，用户可以生成新的规则，然后这个规则不仅会添加到下面的列表中，也会同步更新到“取值”旁边的下拉菜单中。

最后保存的时候，程序会合并两个列表的数据，然后Post给服务器。