**2013/6/3**

**From： 朱越尧**

**关于RestAPI的实现、调用流程**

1.在提供RestAPI的模块中（以VirtualNetwork为例），向IRestApiServer增加RestletRoutable

@init方法：

restApi = context.getServiceImpl(IRestApiService.**class**);

@startup方法：

restApi.addRestletRoutable(**new** VirtualNetworkWebRoutable());

2.其中的VirtualNetworkWebRoutable实现了RestletRoutable接口，该接口在basePath中给出basePath，在getRestLet中给出了不同的URL的服务类

@Override

**public** Restlet getRestlet(Context context) {

Router router = **new** Router(context);

router.attach("/tenants/{tenant}/networks", NetworkResource.**class**); // GET

……

router.attachDefault(NoOp.**class**);

**return** router;

}

@Override

**public** String basePath() {

**return** "/networkService/v1.1";

}

3.上述对应URL的服务类NetworkResource继承了ServerResource类，在每个方法前通过@Get、@Post、@Path等参数对应Http请求

@Get("json")

**public** Collection <VirtualNetwork> retrieve() {

IVirtualNetworkService vns =

(IVirtualNetworkService)getContext().getAttributes().

get(IVirtualNetworkService.**class**.getCanonicalName());

**return** vns.listNetworks();

}

在程序执行中，如果收到REST请求（curl……），RestServer调用RestletRoutable接口（VirtualNetworkWebRoutable），得到对应URL的服务类（此处为NetworkResource.class），然后在该类中，通过@GET等参数调用不同的函数，返回String可以显示给Http请求者，也可以返回一个Collection对象，由RestAPI自动转换成JSON串。