Memcache管理系统说明文档

系统简介：一个用于配置管理，配置变更及自动化运维memcahce的系统。（半成品）

代码地址：<https://github.com/abeginner/McAdmin>

本文档包括功能说明和系统架构说明两部分。

1. 功能说明：
2. 登录

Url：<http://192.168.134.129/mcadmin/user/login>



输入用户名，密码及验证码后登录系统。

1. 注册

Url：<http://192.168.134.129/mcadmin/user/register>



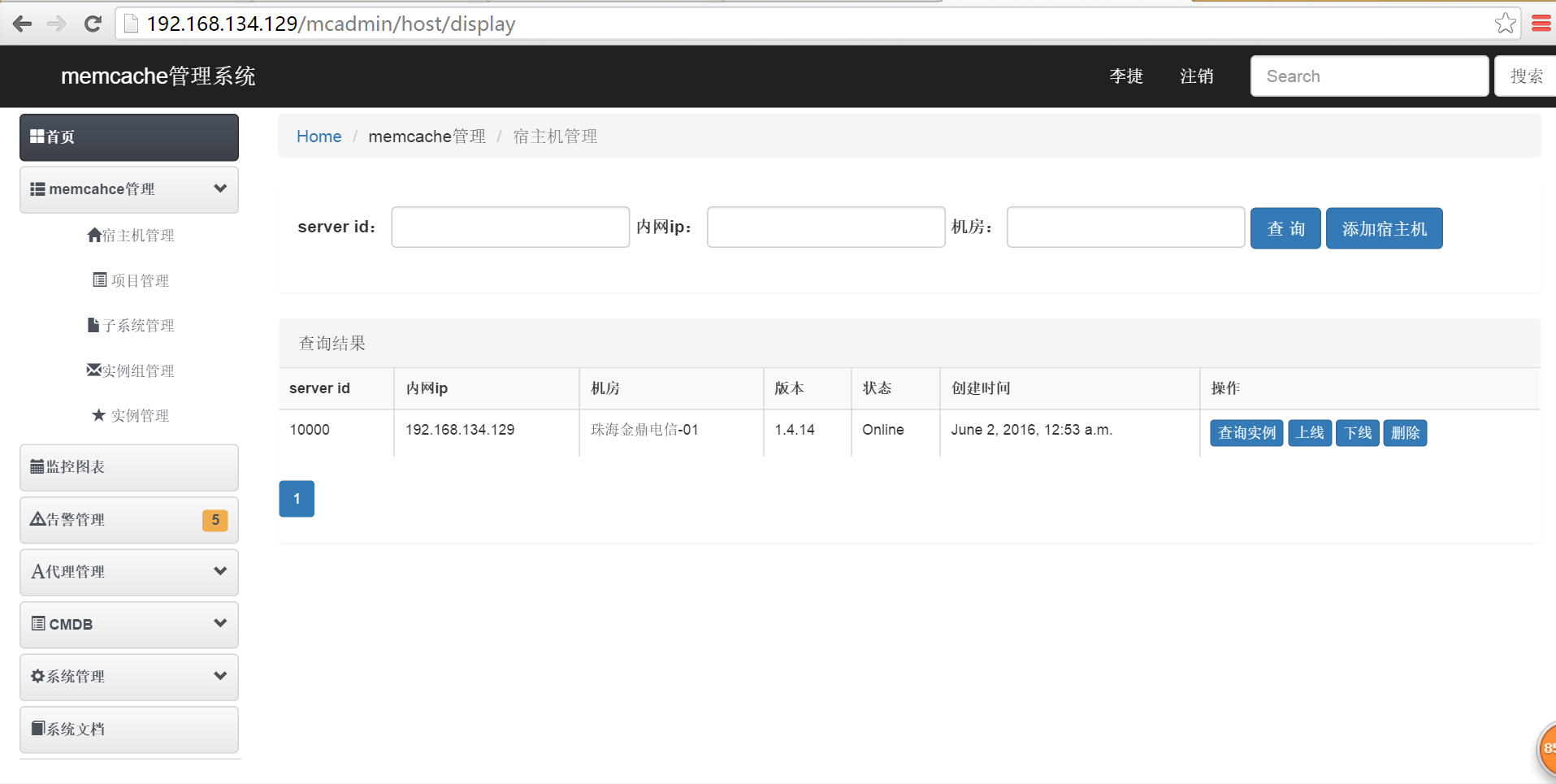
填写用户基础信息注册账户。

（作为一个运维系统注册页面不应该开放的，由于还没做权限系统，所以暂时开放注册页面）

1. 宿主机管理

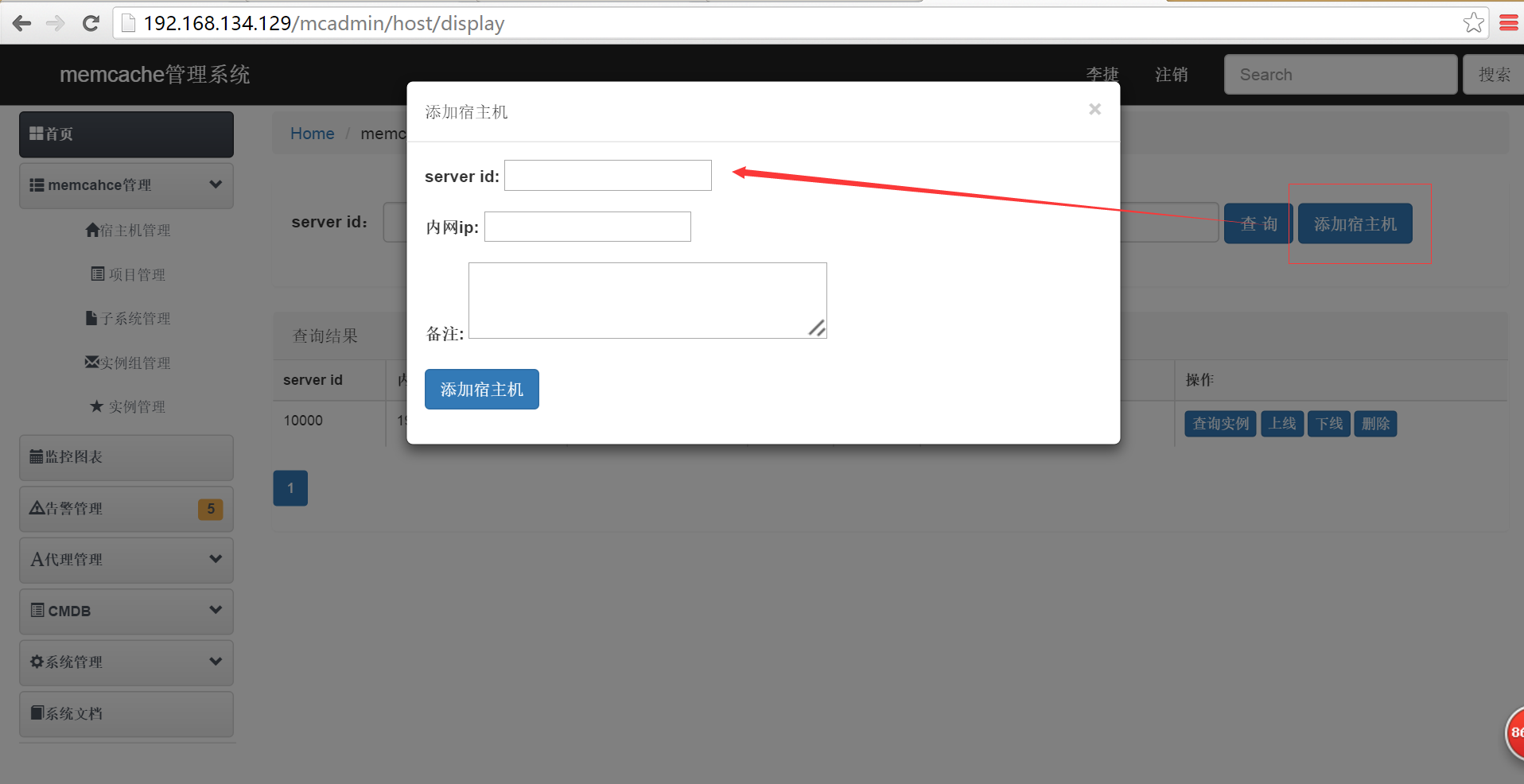
Url：<http://192.168.134.129/mcadmin/host/display>

用于memcache实例运行与物理服务器上面，该视图用于管理安装memcache实例的物理服务器，包括memcache软件的自动安装，上线及下线功能。

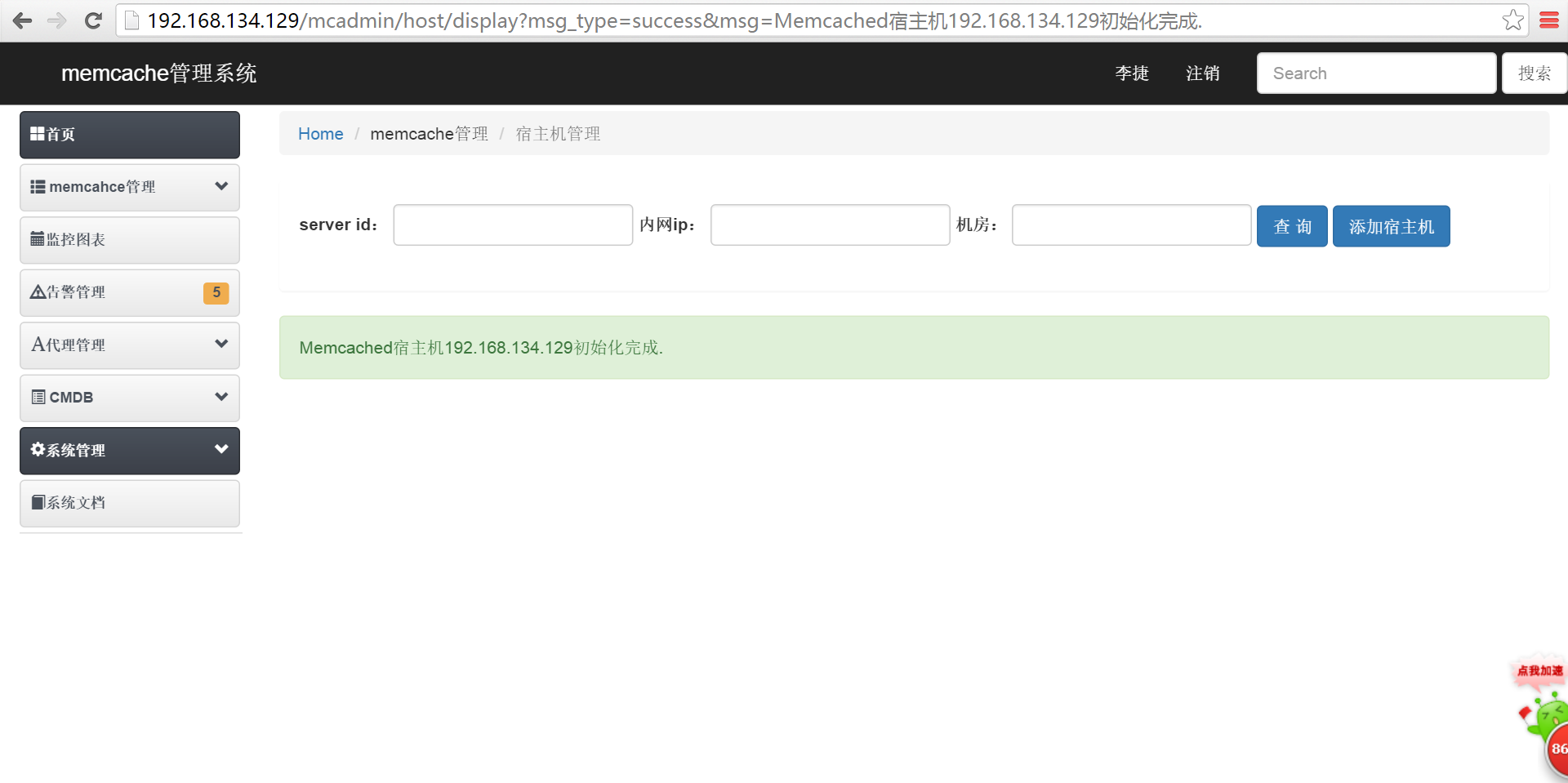


1. 添加宿主机操作

点击添加宿主机按钮就会弹出添加宿主机页面，填写memcache宿主机的server id或内网ip，系统就会往agent的restful接口发送初始化memcache宿主机器的请求，然后agent调用初始化脚本，初始化脚本调用ansible的python api执行playbooks yml脚本，进行初始化操作。



宿主机初始化环境安装完成：



1. 服务器上线操作

点击上线按钮，把ready状态的宿主机状态修改为online状态，只有online状态的宿主机才能在上面部署实例。

1. 服务器下线操作

点击上线按钮，把online状态的宿主机状态修改为ready状态，只有没有活动实例的宿主机才能被修改为ready状态。

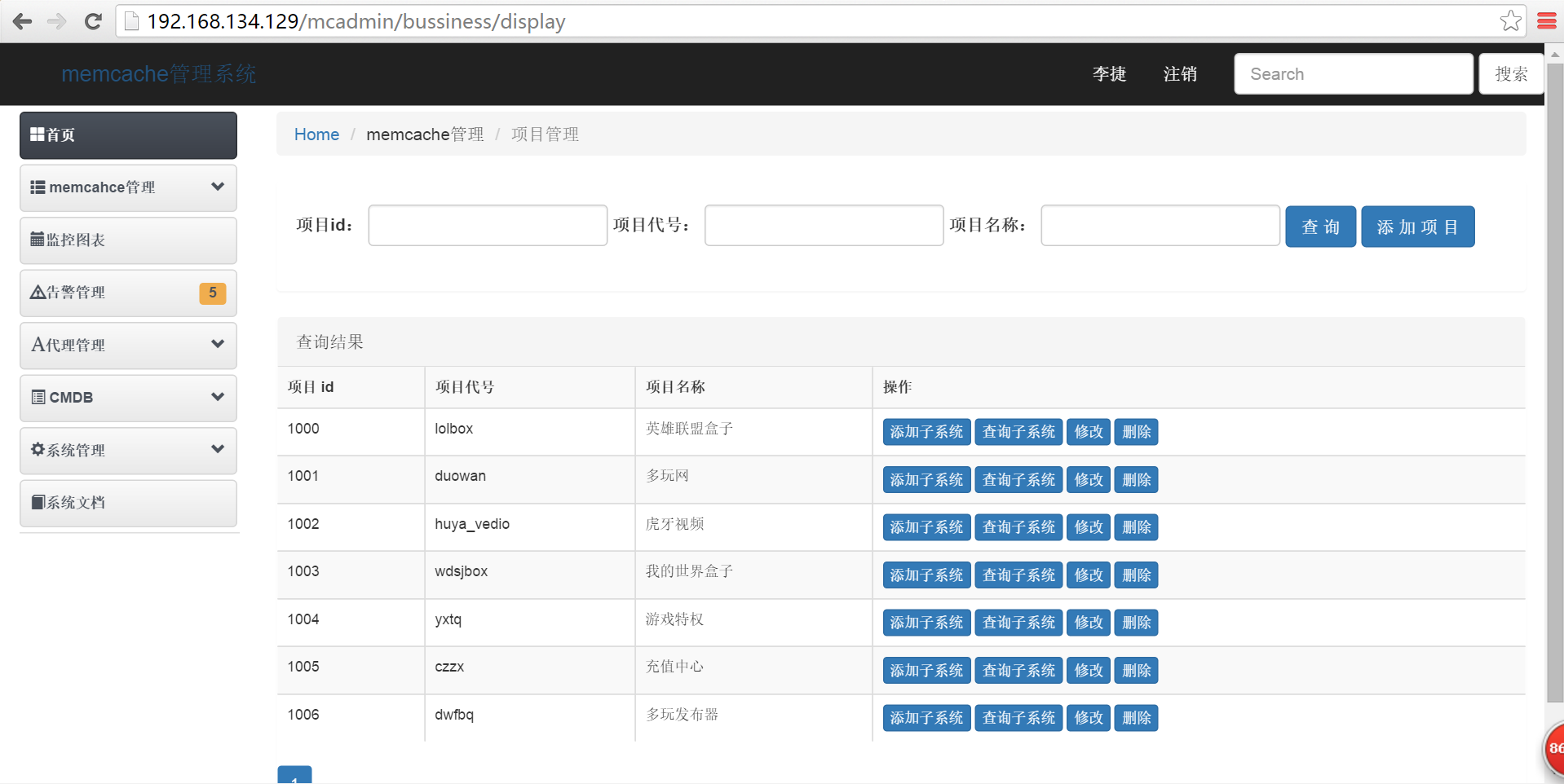
d.删除宿主机

点击删除按钮，删除memcache宿主机，只有ready状态的宿主机才允许被删除，online状态不允许删除操作。

1. 项目管理

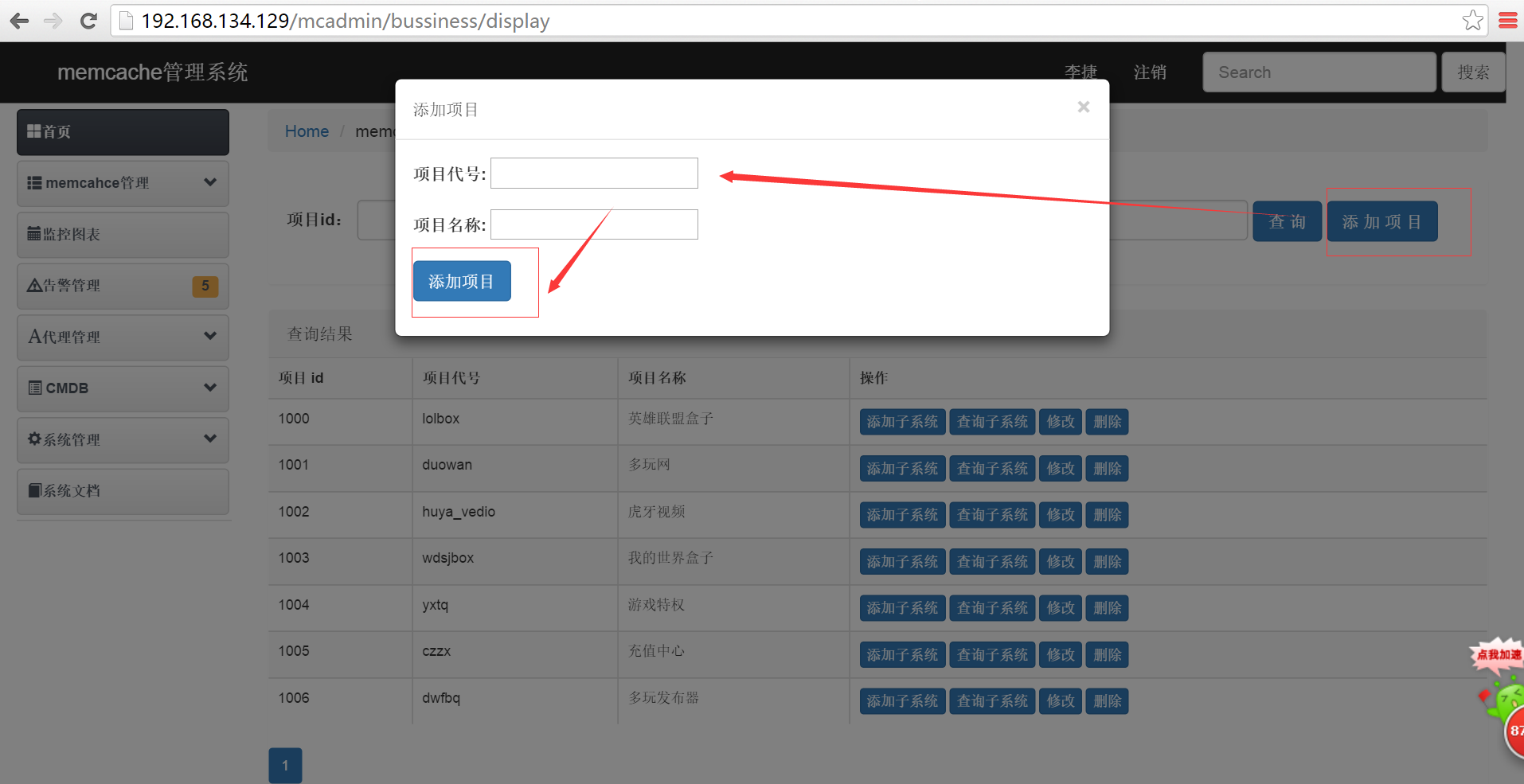
Url：<http://192.168.134.129/mcadmin/bussiness/display>

用于添加和删除项目模块，项目模块下挂子系统模块，子系统模块下挂实例组，实例组下挂具体的memcache实例。

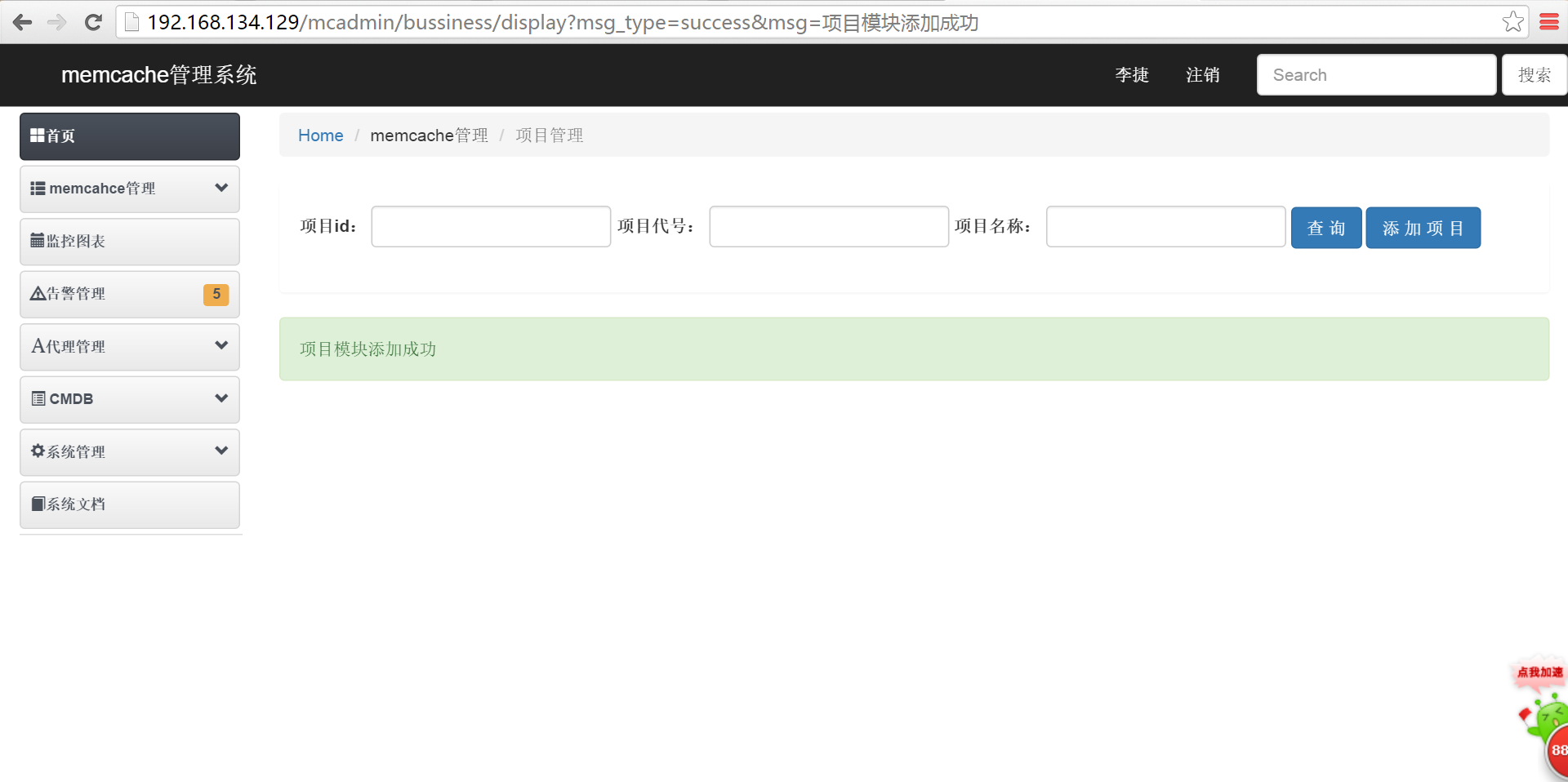


1. 添加项目：

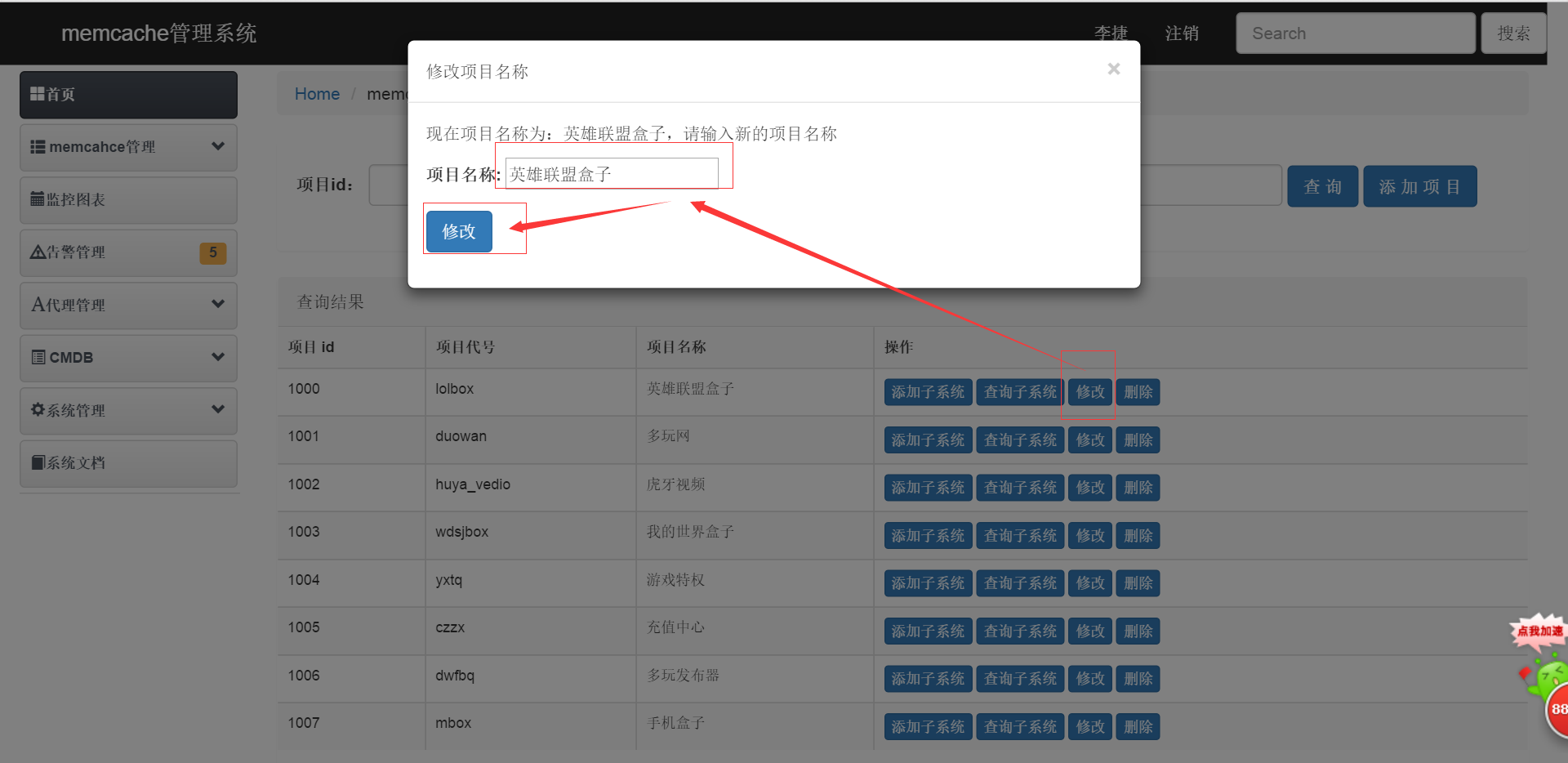
点击添加项目按钮，然后在弹出框里面填写项目代号和项目名称，然后点击添加项目按钮



添加完成：

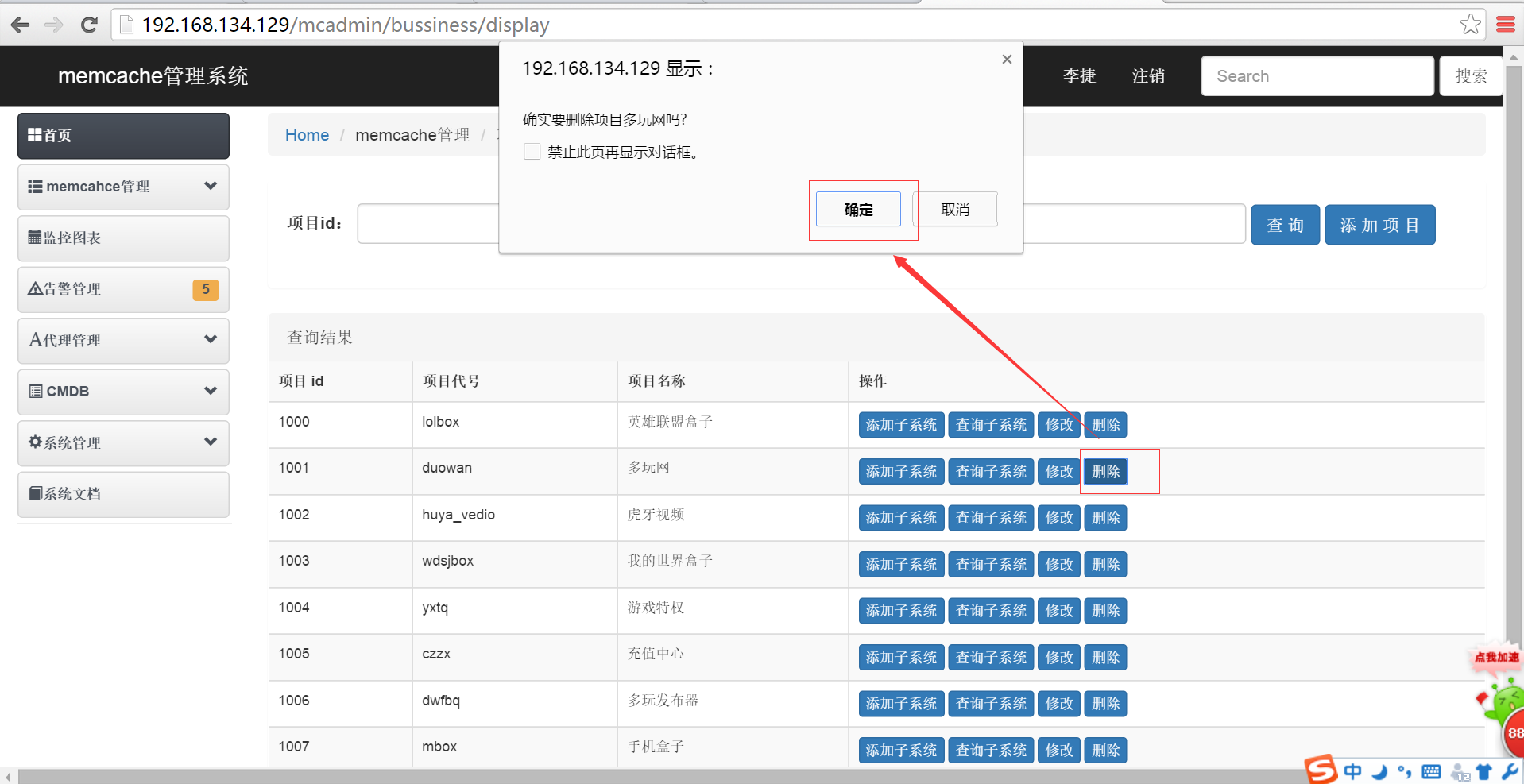


1. 点击修改按钮，修改项目名称，项目代码不能修改



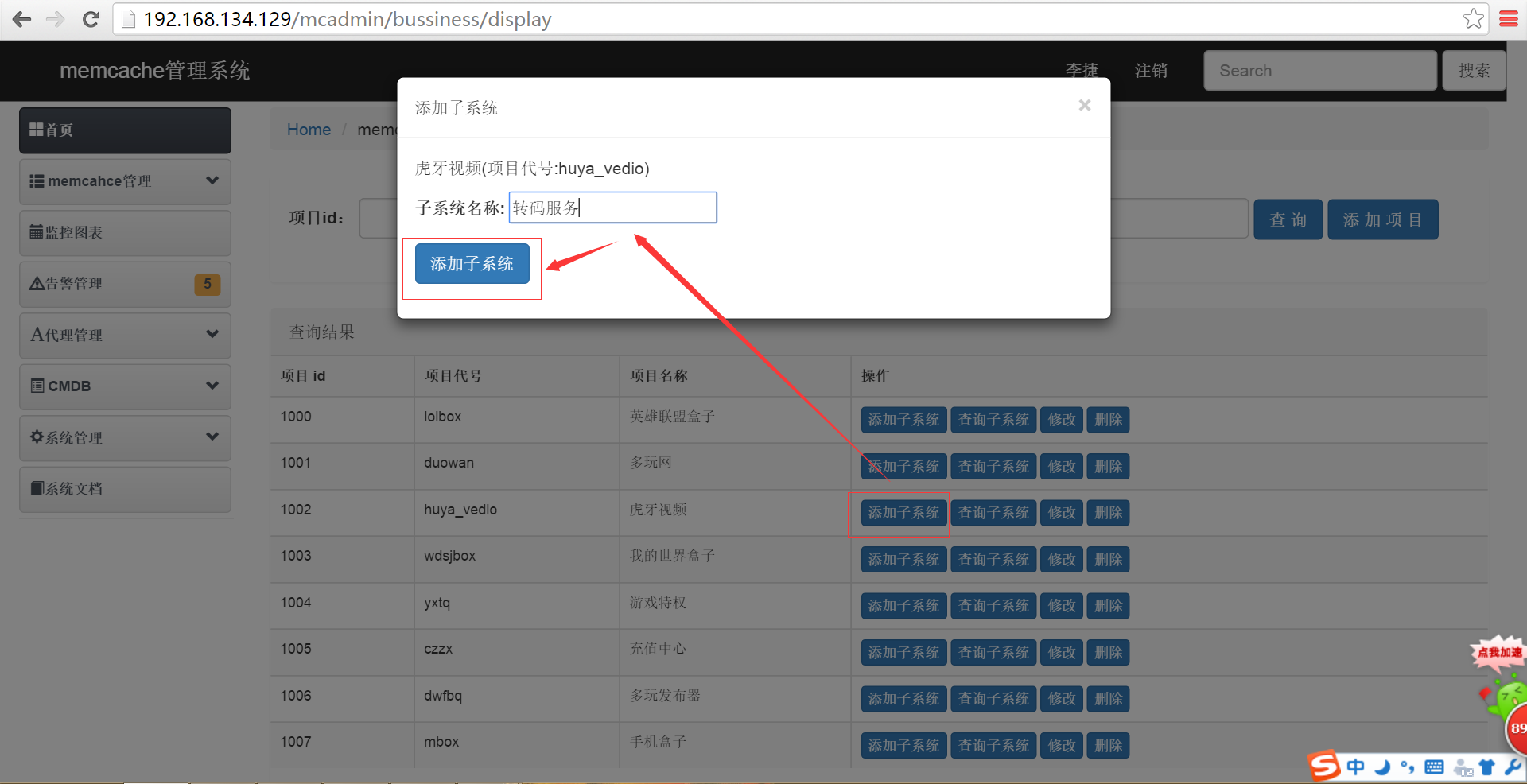
1. 删除项目

点击删除按钮删除项目，只有在项目没有下挂的子系统模块的时候才能删除项目



1. 添加子系统模块

在项目上点击添加子系统按钮，添加子系统模块



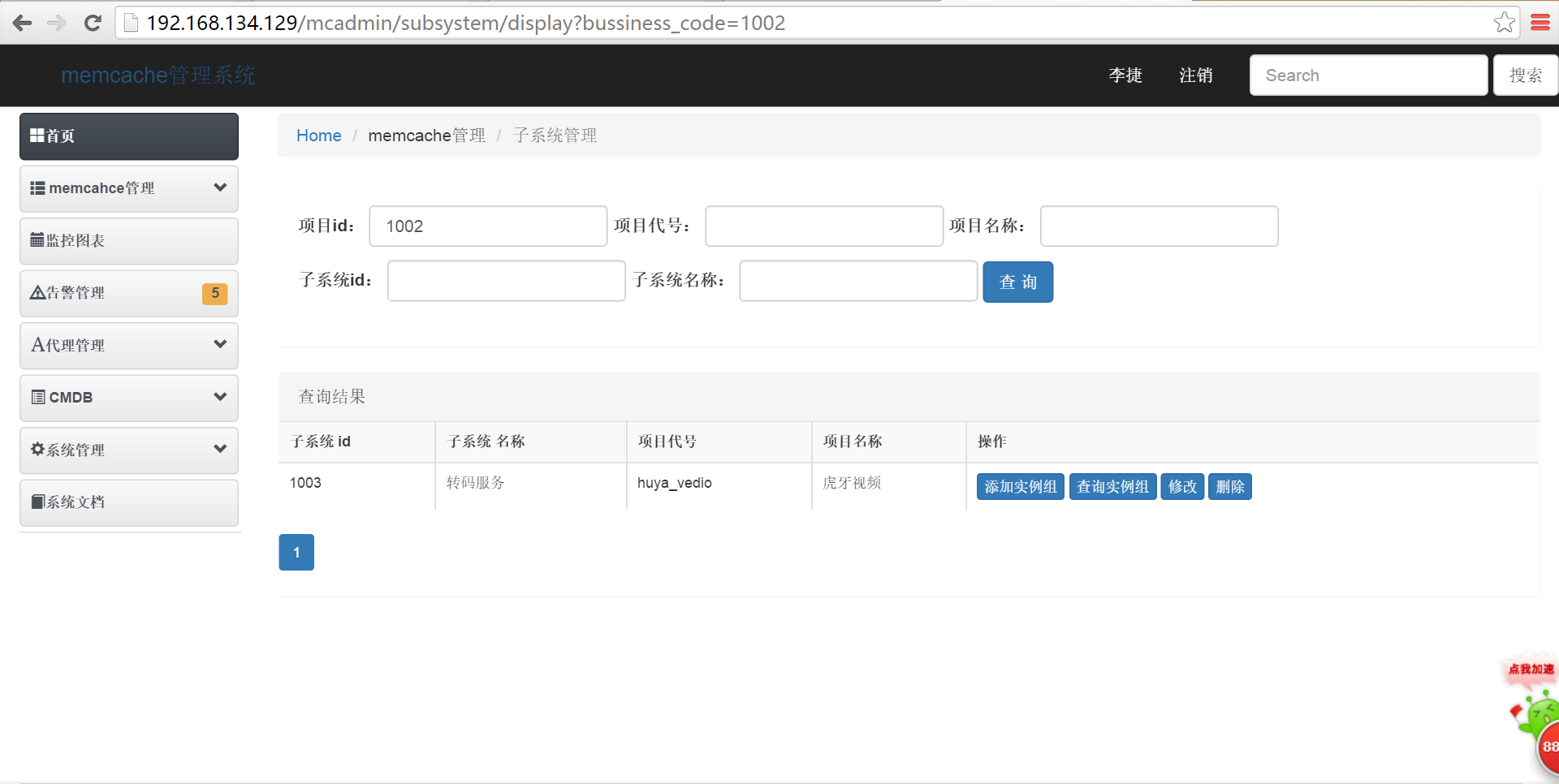
添加成功



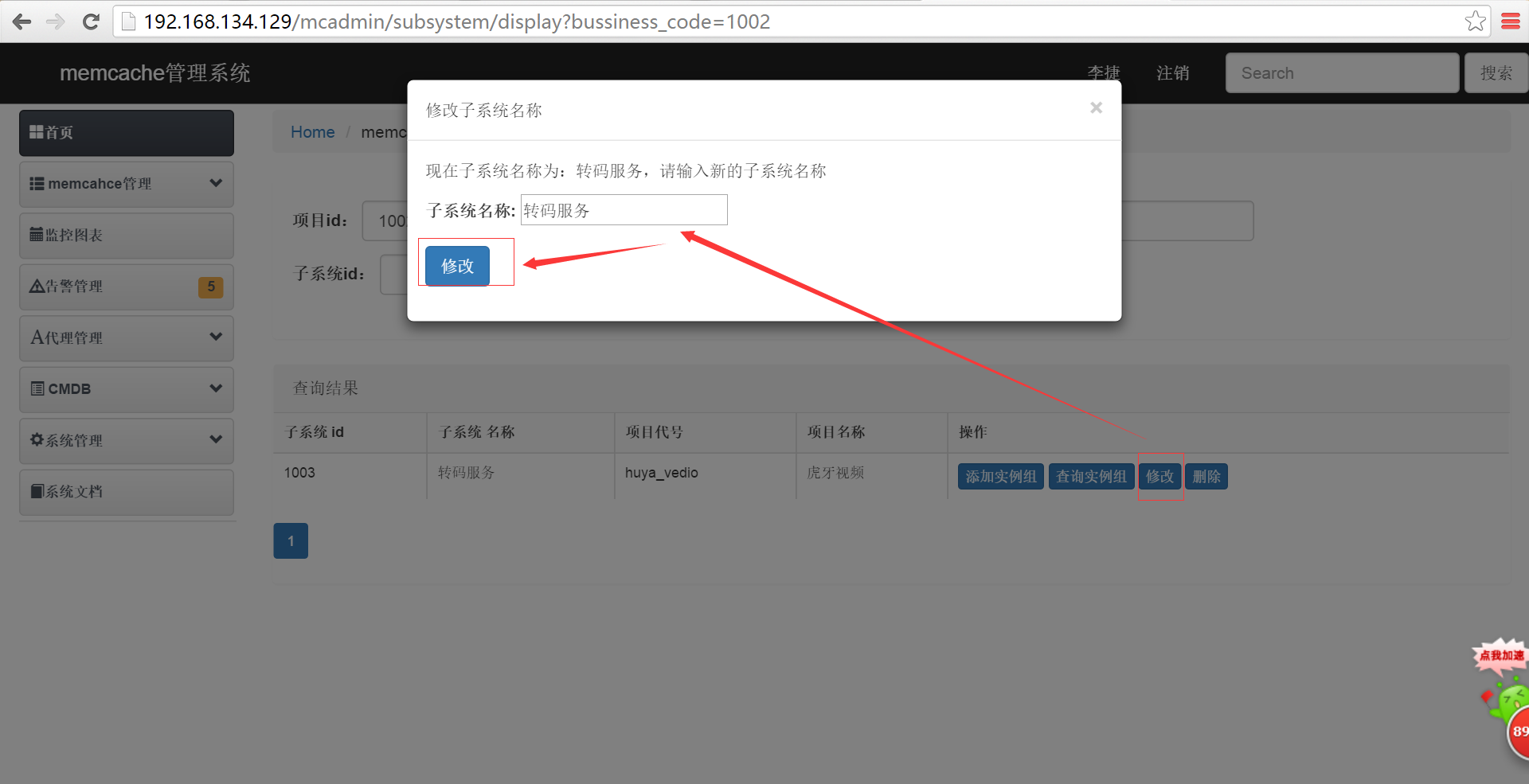
1. 子系统管理

Url:http://192.168.134.129/mcadmin/subsystem/display

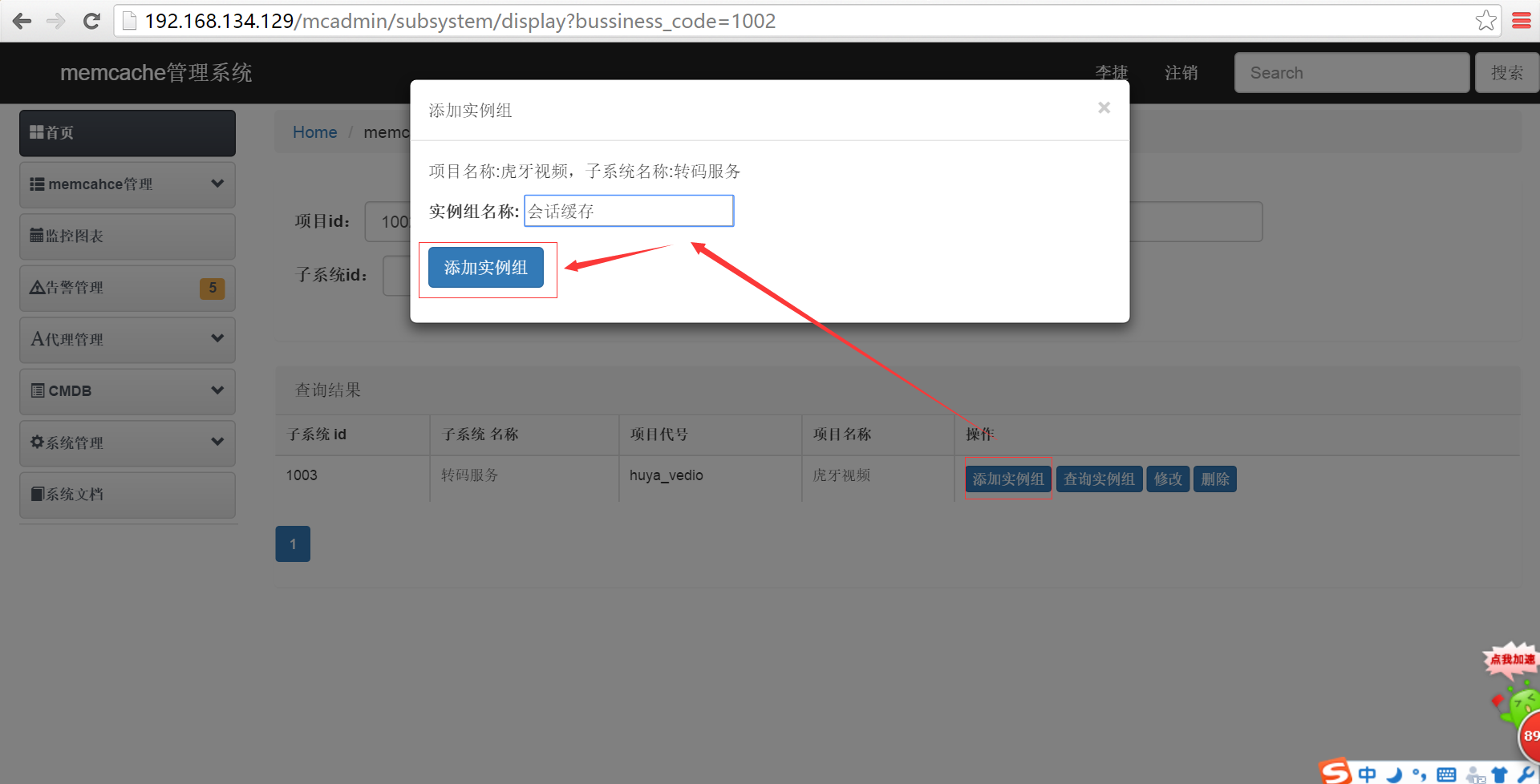
修改，删除和对子系统添加实例



1. 修改子系统名称

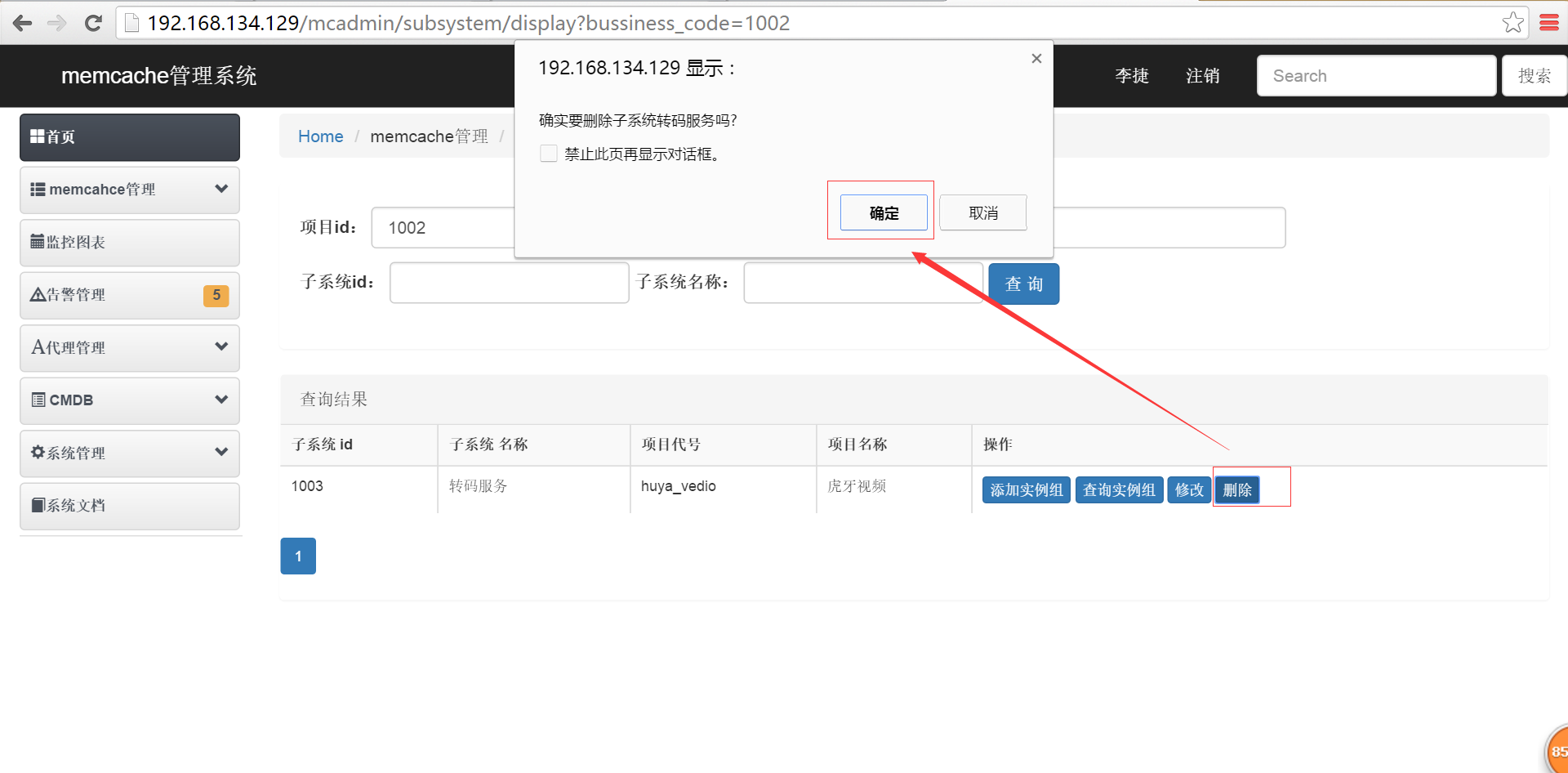


1. 在子系统下面添加实例



c.删除子系统

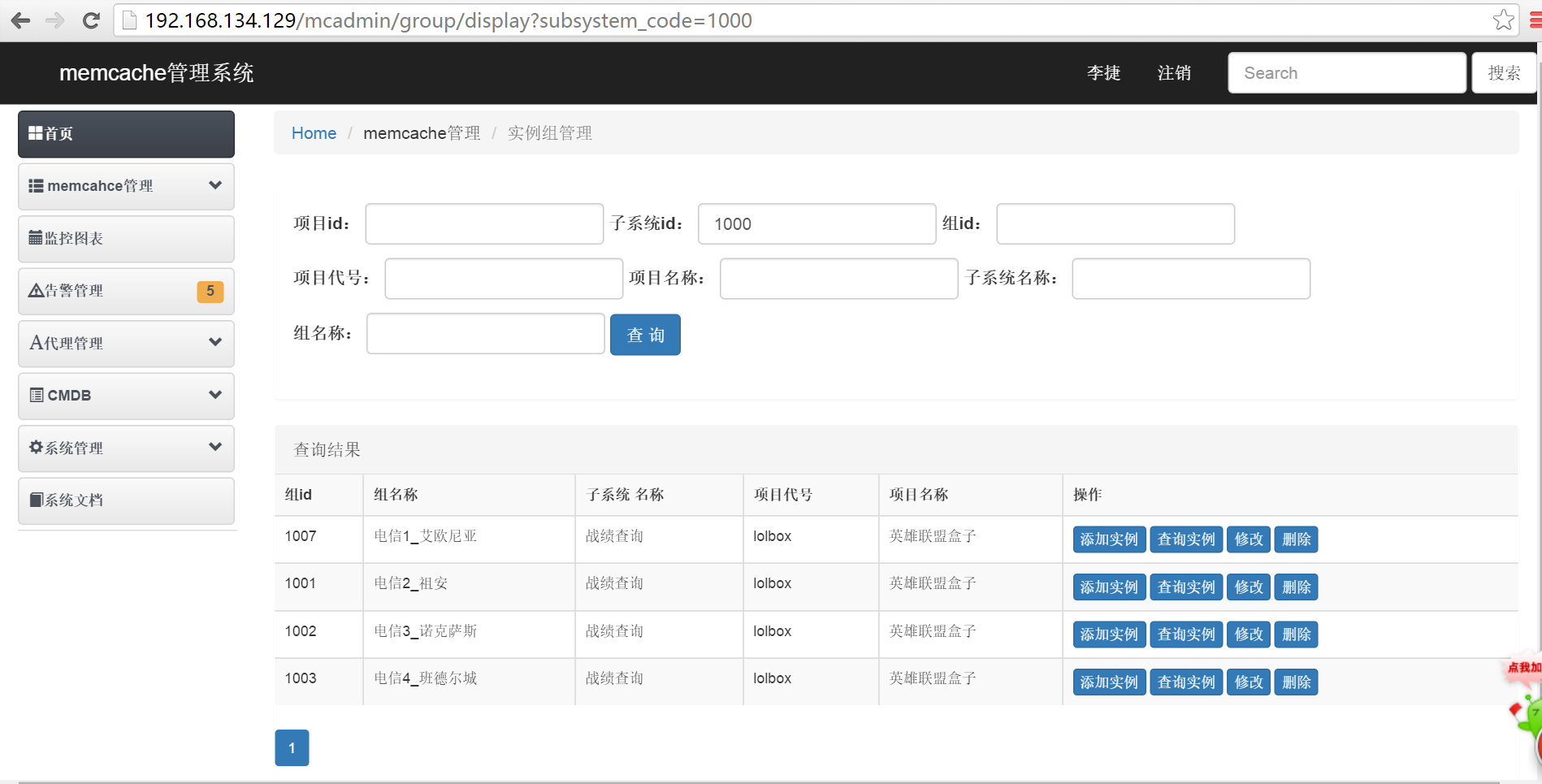
只有在该子系统模块下面没有实例组才能删除该业务模块



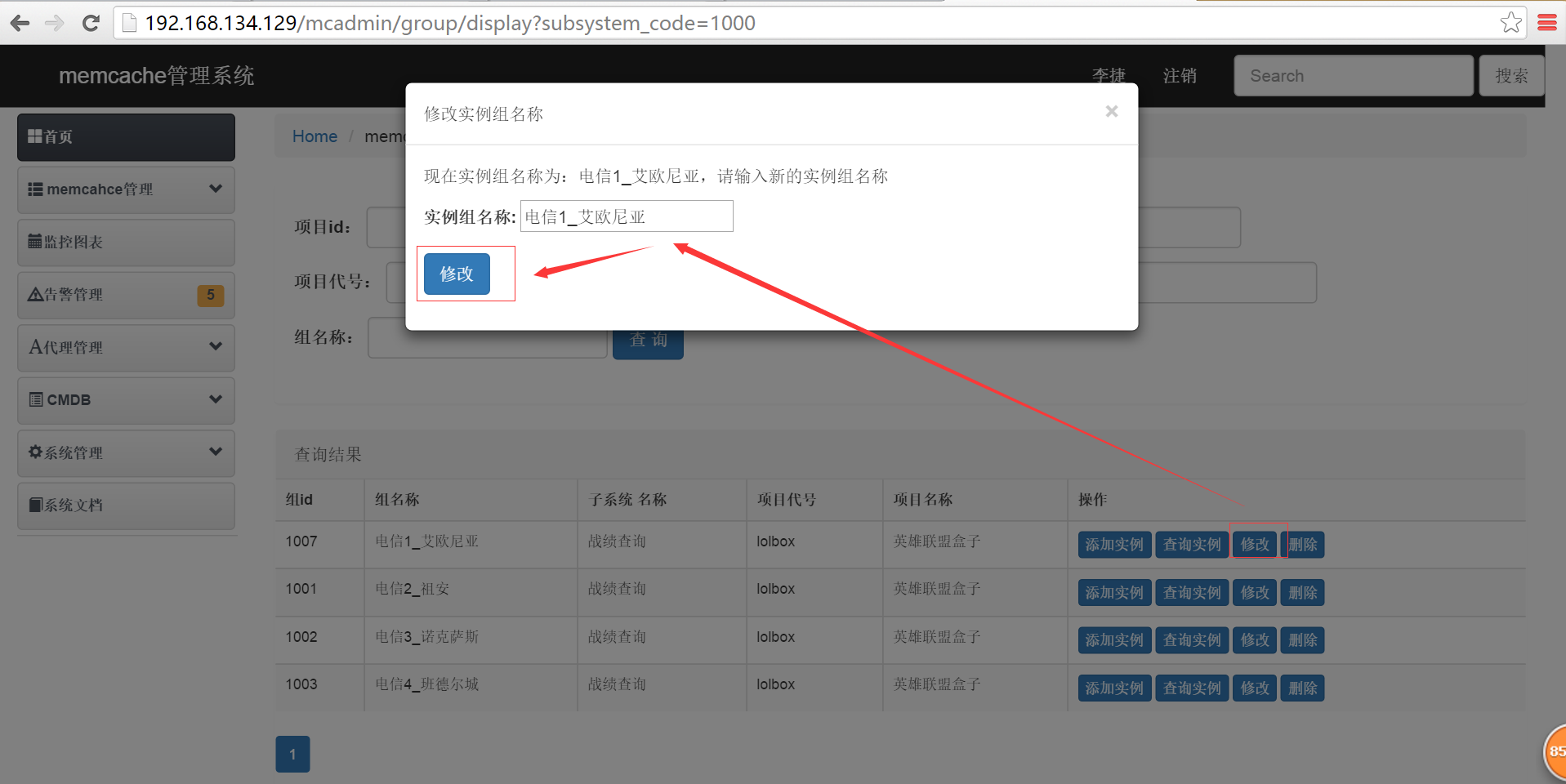
1. 实例组管理

Url:http://192.168.134.129/mcadmin/group/display

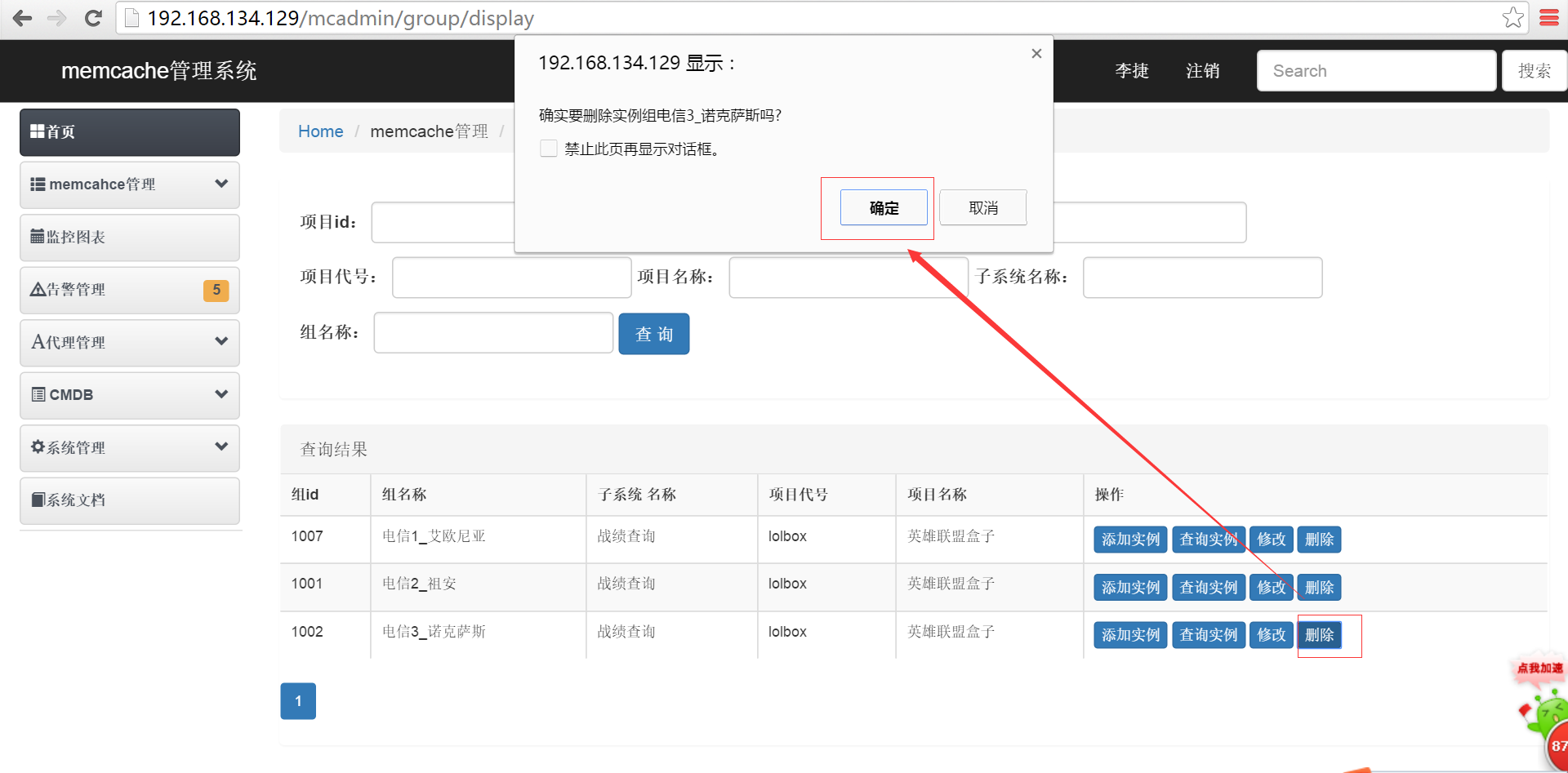
对实例组进行改名，删除和添加实例操作



1. 对实例组点击修改按钮，修改实例组名称

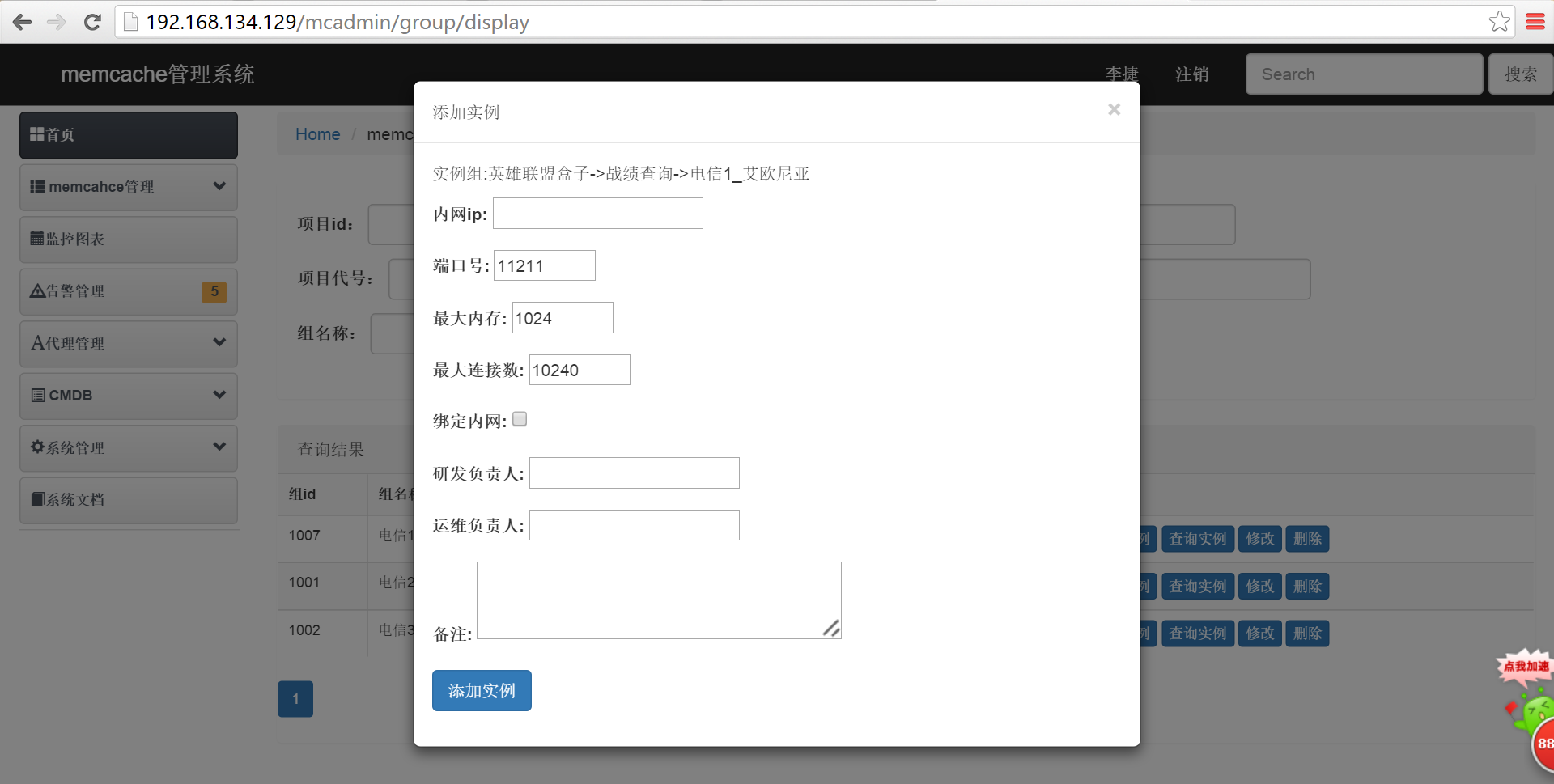


1. 点击删除按钮，删除实例组，只有在该实例组没有下挂实例的情况下，才能删除实例组

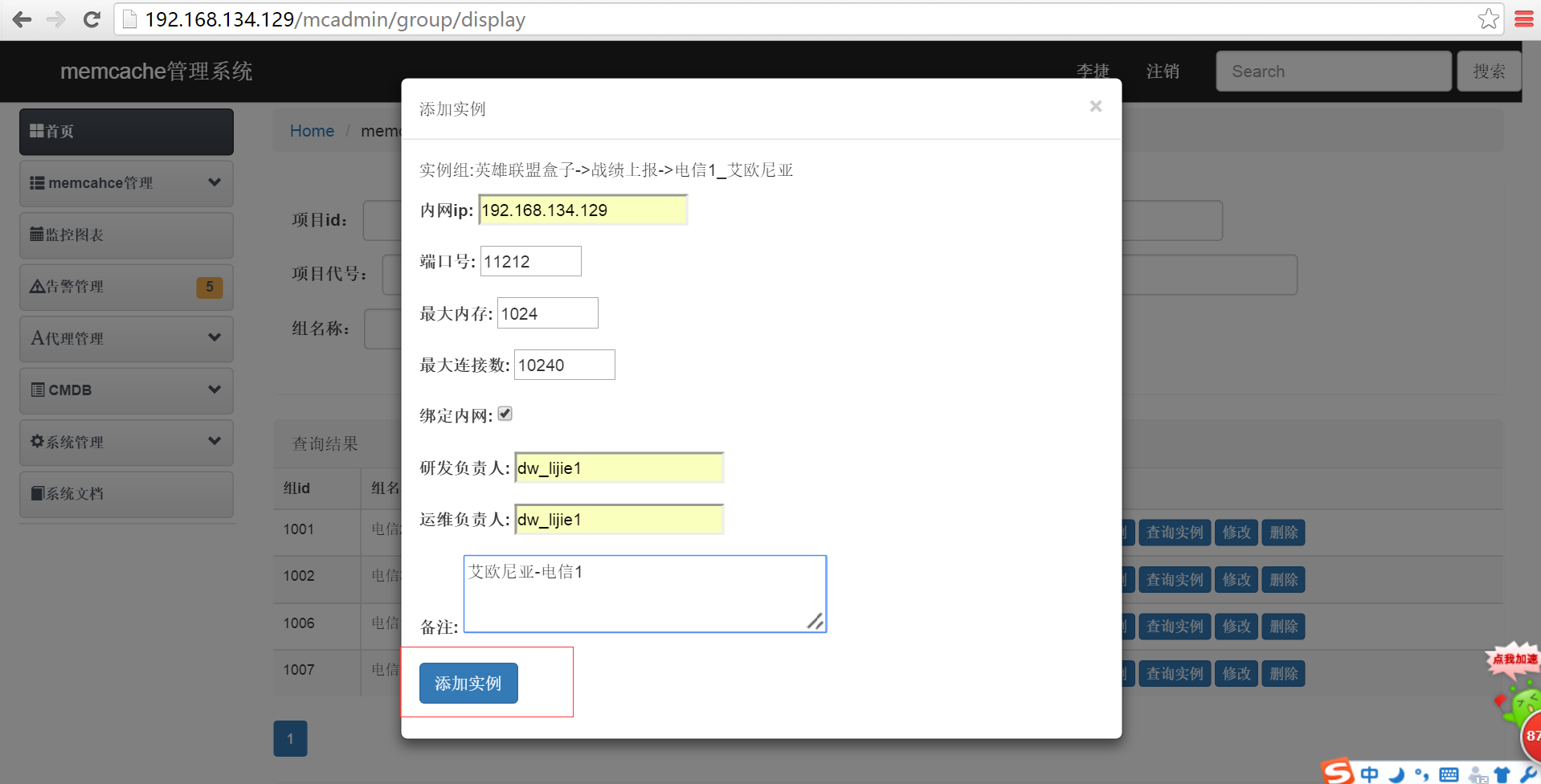


1. 添加实例

点击添加实例



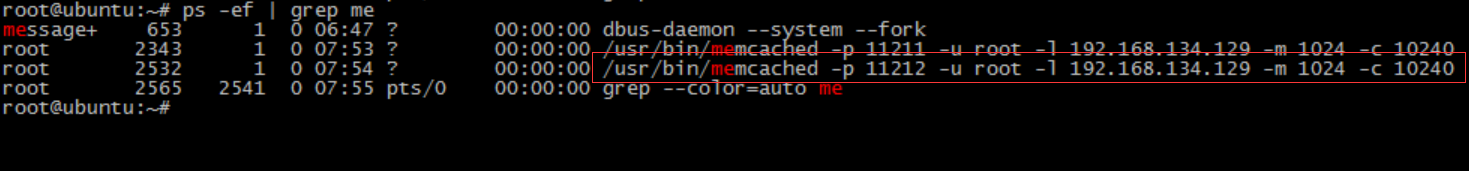
填写配置信息，点击添加实例



添加完成



可以看到，该memcache实例已经通过ansible部署到服务器上了，并且已经启动：



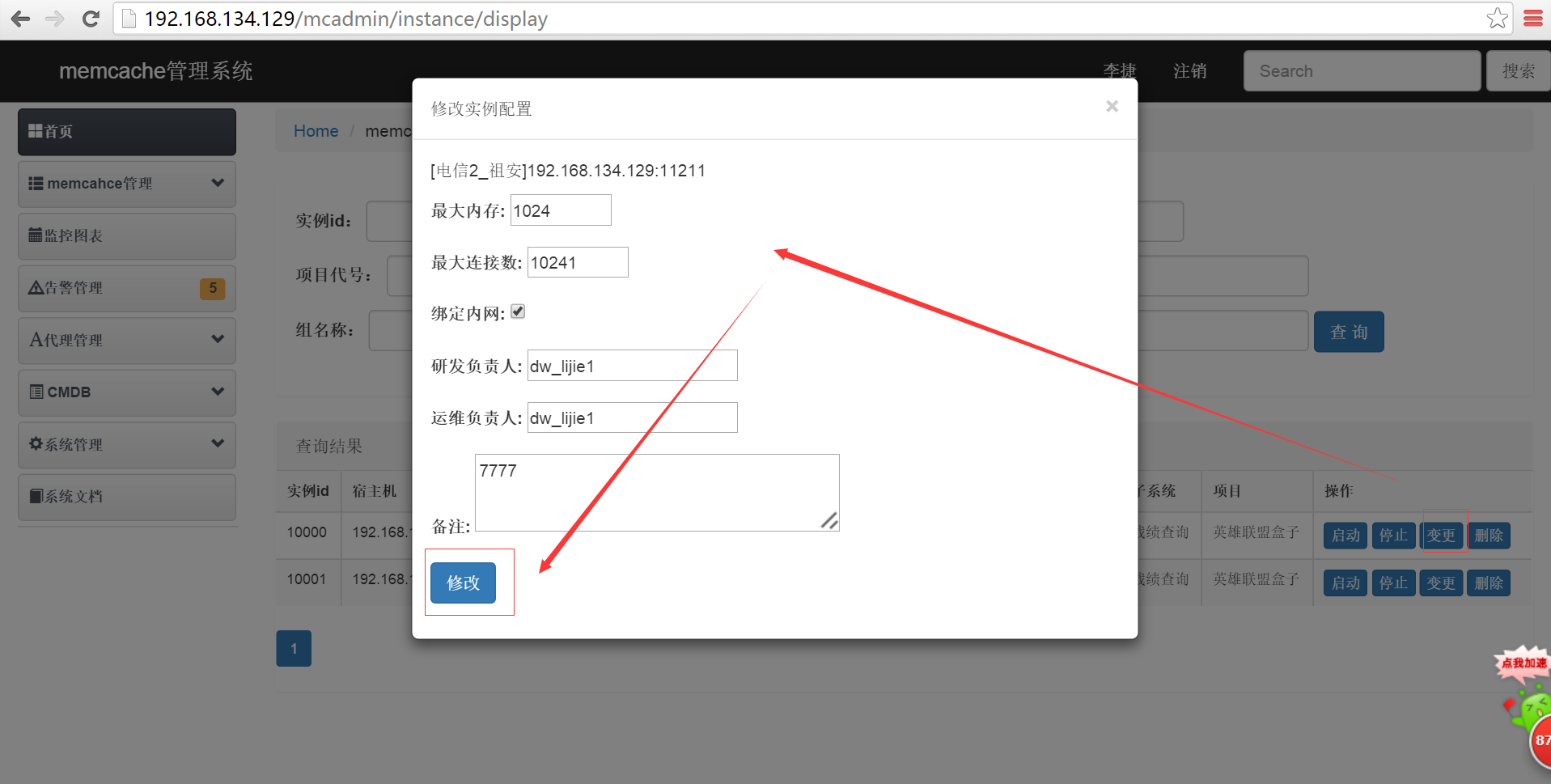
1. 实例管理

Url：<http://192.168.134.129/mcadmin/instance/display>

该视图可以对实例进行配置变更，启动，停止和删除操作



1. 启动实例，把实例从ready状态变更为online状态，使用ansible对服务器发送service memcached 端口 start 命令
2. 停止实例，把实例从online状态变更为ready状态，使用ansible对服务器发送service memcached 端口 stop 命令
3. 变更实例，在运行中的实例不能进行配置变更，只有把实例停止了才能进行配置变更



1. 删除实例：删除该实例，运行中的实例不能进行删除操作，必须把实例停止后才能删除。
2. 监控视图，memcache状态收集统计图表，还没实现，后续再实现。
3. 告警管理：memcache告警管理，还没实现
4. Agent：agent管理，还没实现。
5. Cmdb：cmdb数据增删查改，cmdb
6. 系统架构：

系统数据流如下：

[控制台]（django） ---- http 协议----> restful api [agent](eventlet + webob + routes + [paste.deploy](http://blog.csdn.net/u012977347/article/details/50465918" \t "http://write.blog.csdn.net/_blank)) ----> python脚本 ---> ansible python api----> playbooks ----- ssh协议 -----> [宿主机]

代码文件说明：

1. 控制台代码介绍

McAdmin/Mcadmin：控制台代码，由django框架写成。

McAdmin/urls.py #主url配置文件

McAdmin/settings.py #setting配置文件

McAdmin/mcadmin/backends/base\_backends.py #cmdb backend的基类，用于和cmdb接口对接，读取cmdb里面的服务器信息，默认使用系统自带的cmdb。

McAdmin/mcadmin/backends/\_\_init\_\_.py #CmdbBackend继承自base\_backends.py，重写该类的方法按照给定格式返回数据可以使得本系统和其他cmdb对接

McAdmin/mcadmin/mcadmin\_host\_views.py #宿主机管理视图

McAdmin/mcadmin/mcadmin\_bussiness\_views.py #项目，子系统和实例组管理视图

McAdmin/mcadmin/mcadmin\_forms.py #所有的表单定义文件

McAdmin/mcadmin/cmdb\_urls.py #cmdb的url配置文件

McAdmin/mcadmin/cmdb\_api\_views.py #cmdb的api视图

McAdmin/mcadmin/models.py #所有的数据库表定义

McAdmin/mcadmin/contrib.py #公用函数及类

McAdmin/mcadmin/mcadmin\_urls.py #mamcache管理系统路由表

McAdmin/mcadmin/fsms.py #定义宿主机及实例状态变化的有限状态机

McAdmin/mcadmin/mcadmin\_instances\_views.py #实例管理视图

McAdmin/mcadmin/user\_views.py #验证码，登录，注销及注册视图

1. angent代码说明

agent是一个自写的小型框架，使用eventlet + webob + routes + [paste.deploy](http://blog.csdn.net/u012977347/article/details/50465918" \t "http://write.blog.csdn.net/_blank) + oslo.config编写。

Agent的作用是把shell脚本或者python脚本映射为restfull接口，使用url + http动词访问接口即等于调用相应的shell脚本或者python脚本。

Public/agent/server.py 里面定义了两个类：

Server类：1）使用oslo.config读取配置，2）使用eventlet.wsgi启动wsgi server

application类：注册controller列表及对http请求路由到相应的controller

Public/agent/controller/base.py #所有controller的基类，定义controller必须继承此类。

一个controller即一个资源，也就是一个url，例如：

[http://192.168.134.129:8090/mcadmin/memcache\_instance](http://192.168.134.129/mcadmin/memcache_instance)

url的格式：<http://[地址]:[端口]/[application]/[resource]>

Application即应用的名字，如mcadmin，每个agent是唯一的，在agent.cfg文件中配置。

Resource，资源名，其实就是Public/agent/controller/目录下对应的python脚本的名字，如Public/agent/controller/memcache\_instance.py，其资源名为memcache\_instance，对应的url为[http://192.168.134.129:8090/mcadmin/memcache\_instance](http://192.168.134.129/mcadmin/memcache_instance)

如何定义一个资源：

定义一个shell脚本资源：

1.Public/agent/controller/helloworld.py定义如下：

#-\*-coding:utf-8-\*-

from base import BaseController

class helloworld(BaseController):

def index(self): #index函数对应的http动词是get

self.parameter\_keys = ['name'] #请求数组的参数顺序，这样的意思是get请请求的name参数会作为$1传参给shell脚本。

self.action = 'shell' #脚本类型是shell

self.script = 'hello.sh' #脚本是Public/agent/controller/scripts目录下面的hello.sh

def get\_resources():

return helloworld() #get\_resources()函数返回controller类的实例，该实例是可调用对象。

2.hello.py脚本的定义

#!/bin/bash

echo "hello " $1

1. 执行结果



把name=lijie使用get方法请求url <http://192.168.134.129:8090/mcadmin/hello>

返回结果为：hello lijie

也就是说执行了shell脚本：/bin/bash hello.sh lijie

Python脚本的定义同理：

1.定义controller：

memcache\_host.pyc memcache\_instance\_manage\_single.pyc memcache\_instance.pyc

root@ubuntu:~/www/McAdmin/Public/agent/controller/scripts# cat ../memcache\_instance.py

#-\*-coding:utf-8-\*-

from base import BaseController

class memcache\_instance(BaseController):

def create(self):

print 'memcache\_instance create() is called'

self.action = 'py'

self.script = 'memcache\_instance\_create.py'

self.parameter\_keys = ['host', 'port', 'max\_memory', 'max\_connection', 'is\_bind']

def show(self):

print 'memcache\_instance show() is called'

"""

delete the memcache instance

"""

self.action = 'py'

self.script = 'memcache\_instance\_del.py'

self.parameter\_keys = ['host', 'port' ]

def get\_resources():

return memcache\_instance()

1. 和shell脚本不同的是，其对应的python脚本必须定义一个get\_body(parameter)方法，其返回的内容为最终请求返回的json序列的内容：

def get\_body(parameter):

pb = PlayBook('playbooks/memcacheinstance\_create.yml', parameter)

return pb()

所有restfull请求的Content-Type字段必须为：”application/json"