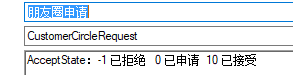
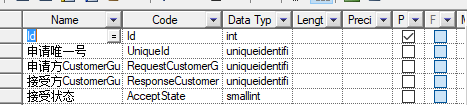
动工——朋友圈

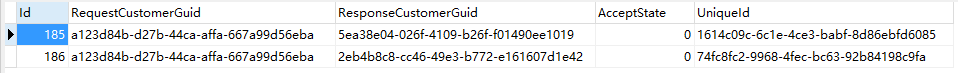
是

crm数据库，四个表：

（1）







A和B，比方花花和阿瓜妞，A，也就是花花请求与B（阿瓜妞）建立朋友关系，发出申请，这时候就多了一条数据：

请求方：花花id 响应方：阿瓜妞的id 接受状态（AcceptState）0

这时候，阿瓜妞那边肯定弹出弹窗，这个是轮询吗，阿瓜妞那边弹出是否接受的弹窗。

我自己觉得，中介者模式，后者说观察者模式，这边点确定了，send事件直接也就触发了另一个用户的Receive方法，也就是弹窗事件。

接下来，阿瓜妞来处理请求，如果接受，AcceptState改为1，如果不同意，AcceptState就是-1.相应地，就是改变了这条数据库的内容。

整理：通知对方，以及弹窗，前端去做。我后台做的就是：

1.请求

A请求B，我写入一条数据。AcceptState为0

2.响应

B是否接受，就是去修改该AcceptState的值，为1或者-1.

3.这个不确定，如果要拉黑了。

是否要将AcceptState由1变成-1.

4.这个感觉是必要的，一旦上面将AcceptState设置为0，那意味着两个建立了朋友关系。那就在我的朋友圈表，MyCustomerCircle中加入两个人的id吗。

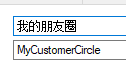
如果是这样，还要加一条IsDeleted，因为拉黑了，就置为true了。

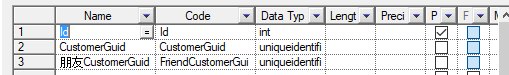
用户表用的是状态字段：

State：-4 已删除 -3 已禁用 -2 已拉黑 -1 已冻结 0 已注册（未激活） 10 已激活（待实名认证）15 实名认证未通过 16 实名认证审核中 17 运营审核通过 20 已实名认证



（2）







A和B建立了朋友关系，A比方是花花，那去查所有花花的朋友，就是CustomerGuid=花花的guid

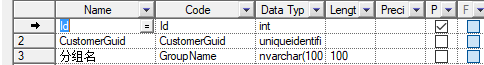
这里如果删除好友的话，要加字段。IsDeleted

用户表用的是状态字段：

State：-4 已删除 -3 已禁用 -2 已拉黑 -1 已冻结 0 已注册（未激活） 10 已激活（待实名认证）15 实名认证未通过 16 实名认证审核中 17 运营审核通过 20 已实名认证

（3）







这里，对应一个的医生Id，也就是CustomerGuid，他可以有n多的分组，比方老师，同学，朋友，家人。等等，他每次添加的时候都加到这个数据库。而当他要添加一个新的朋友时候，输出显示他所有的分组信息。供他选择，然后将新朋友放到该分组当中。

（4）







上面的朋友分组，如家人，朋友，老师，这种，已经建立。这里添加了新朋友，只要选择了对应的标签，就添加到该医生（花花）的分组A（好朋友）中了。

而这里使用了GroupId，通过GroupId与分组表CustomerCircleGroup建立连接join查询，或者说直接以该GroupId去GroupId表中获取了当前的医生id，CustomerGuid，以及分组名GroupName，这样就获得了医生和朋友间的一一对应，并且显示了分组的信息。

比方 花花——恶友——僵僵

要做：

0.

前置页，也就是通讯录页面的首页：

0.1——显示通讯录首页



这里获取所有朋友的信息，信息包括：

朋友列表：包含头像、姓名、科室、所在医院名称（限定最大长度，超出的显示...

0.2 如果有新的朋友，进行处理，同意或者拒绝，分别将数据库的AcceptState进行修改。

这里应该是一个ajax，当显示首页的同时，一个ajax请求，看表CustomerCircleRequest中对应的响应方ResponseCustomerGuid，查出所有响应方是当前登陆用户自己，以及AcceptState为0的所有数据，然后显示在新朋友中。

接下来处理的方式就是修改该条数据中AcceptState是1还是-1.而对应的我的朋友圈中，应该也是要增加这条数据，并且state设置为1，正常。0还是-1是删除状态。

新朋友界面：



（1）

交互1：进入新的朋友入口：通过通讯录

返回列表

GetFriendsList

a：

进入新朋友界面后，就查ResponseCustomerGuid是当前用户，并且AcceptState是0的情况。然后列表。（这是需要回复的）

当点同意之后，就AcceptState置为1.并且在我的朋友圈表MyCustomerCircle增加一条数据。

同意之后要判定对方设定的分组，所以接收这里是看得到对方设置的分组信息的。老师，学生，还是同行。如果对方，设置了同行，那我就默认设置为同行，老师对学生，学生又对家长。

问了李阳，查询新朋友信息后，当点击增加好友，要在三个表中加操作，让在bll层去处理，用事务。

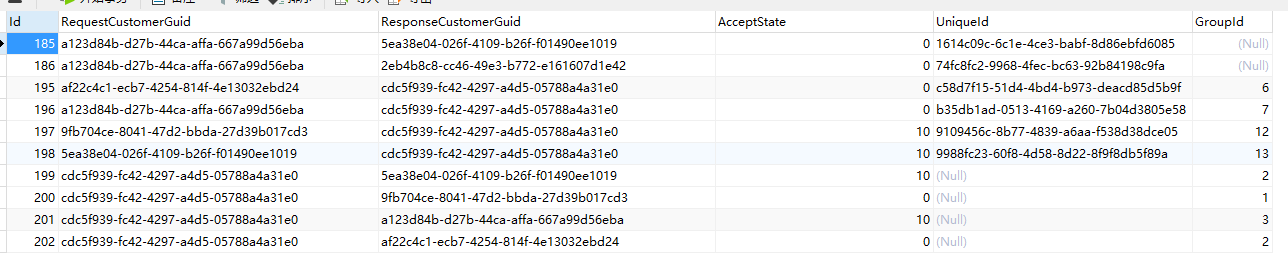
（1）点击添加

ReceiveFriendRequests（requestCustomerGuid，responseCustomerGuid）

ReceiveFriendModelParam

同意之后：

（1）修改request请求表的状态，状态由0改为10

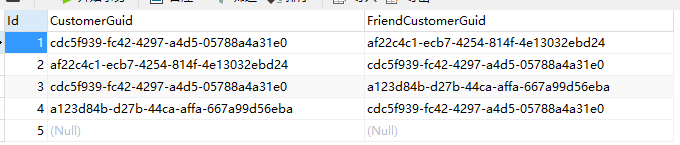


第一步测试成功，状态改为10.



（2）我的好友表中增加好友，且互为好友

MyCustomerCircle

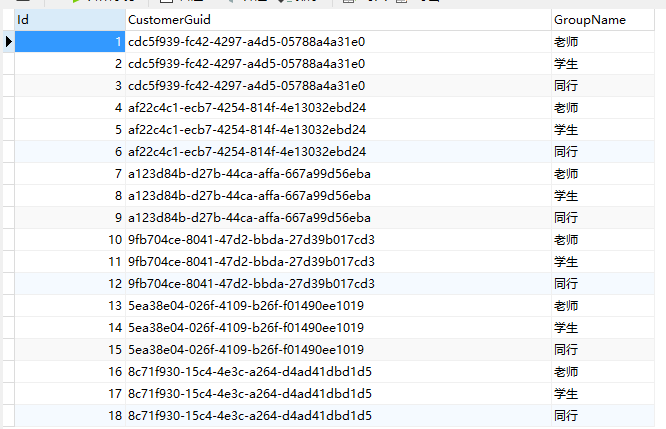


增加两条记录

（3）

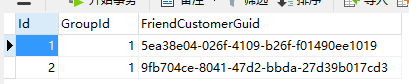
先默认地去增加当前用户的三个标签，因为对方已经默认设置了标签。

CustomerCircleGroup



根据对方设置的标签，比方老师，获取当前登录用户对应的，也就是学生的GroupId，然后增加到CustomerCircleGroupMember表中。

SetLabel



b：

还要查ResponseCustomerGuid是当前用户，并且AcceptState是1的已添加的记录。

这里不用了，这里是上面的操作成功后，比方返回了msg，是成功的ok，就显示已添加。

c：

自己的添加好友申请也显示

查RequestCustomerGuid是当前用户的所有信息，AcceptState为0的显示“等待验证”，AcceptState为1的显示“已添加”。

总结就是这里要进行两次查询，然后将结果整合传给UI端，也就是手机端。

交互2：返回到通讯录

就是跳回通讯录的首页，继续调用通讯录首页的接口

交互3：添加：点击”添加“图标，同意对方添加请求，状态显示为已添加

0.3 对现有的朋友列表进行搜索

前端还是后端去做呢，感觉所有列表都到前端了，前端自己去玩吧。



交互1：在搜索输入框内输入姓名或手机号，进行检索（此处注意IOS自带搜索功能，安卓的需要在页面加一个搜索按钮

do：

根据搜索的结果显示该用户，这里要确定要前端还是后端做。

交互2：选中某位朋友，点击进入朋友详情页

do：根据该朋友的CustomerId去查找详细信息

这里的搜索历史要求：

搜索历史：最多显示9个，按搜索时间最新的显示

应该是在前端的缓存当中。

0.4 添加朋友

跳到下面，详解

0.5 删除好友，

对应的数据库还没有加字段 state

就是当前CustomerId，要删除的好友的CustomerId，然后查到这条数据，将state置为删除

0.6 选中某条朋友记录，点击进入详情页（看到这里，接下来没仔细看了）

获取该朋友CustomerId对应的许多数据吧

1.

分组，加好友，删好友。

2.

添加朋友



加号“+”点了，搜索朋友，这里姓名或者手机，找到该customer。

2.1——搜索用户

交互：在搜索输入框内输入姓名或手机号，进行检索（此处注意IOS自带搜索功能，安卓的需要在页面加一个搜索按钮

根据手机或者姓名去用户表中获取该用户的信息：



搜索的结果这样的标准去显示：

搜索结果列表：包含头像、姓名、科室、所在医院（限定最大长度，超出的显示...）



说是DoctorMemberService中有类似的关联查询，我去找来，改一下。

customer表关联DoctorMember表。查询封装上面的结果集合。

SearchFriends

SearchFriendsModel

SearchFriendsModelParam

建立搜索类

进来的参数，nameOrNum，建了类，回传的话，根据要求去建立新类

头像、姓名、科室、所在医院

可以获取头像guid

Customer表中获取

头像guid，姓名

DoctorMember表中获得

所属医疗机构名称 HospitalName

DoctorOffice表中获得

一级科室名称 Office

返回无分组列表

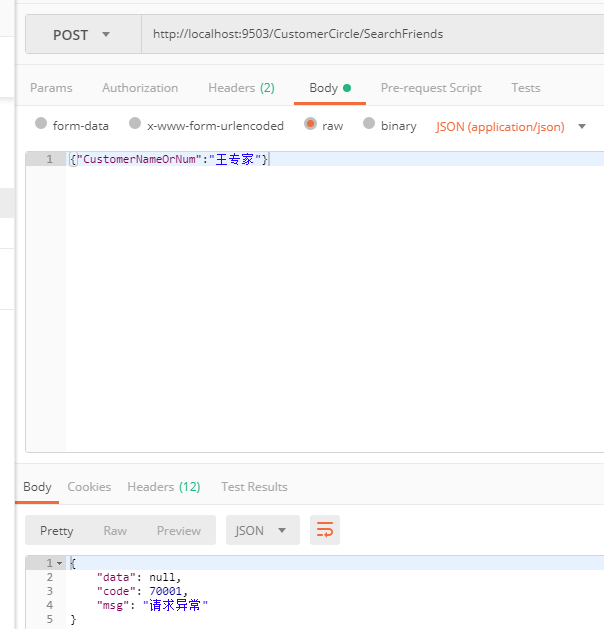
问了李阳，一级科室会有多个的，同一个用户，所以返回list<string>

但查了下，好像是二级科室是多个，一级是1个，等会再确认下

做好了，默认会诊诊断医生A，去查找 王专家

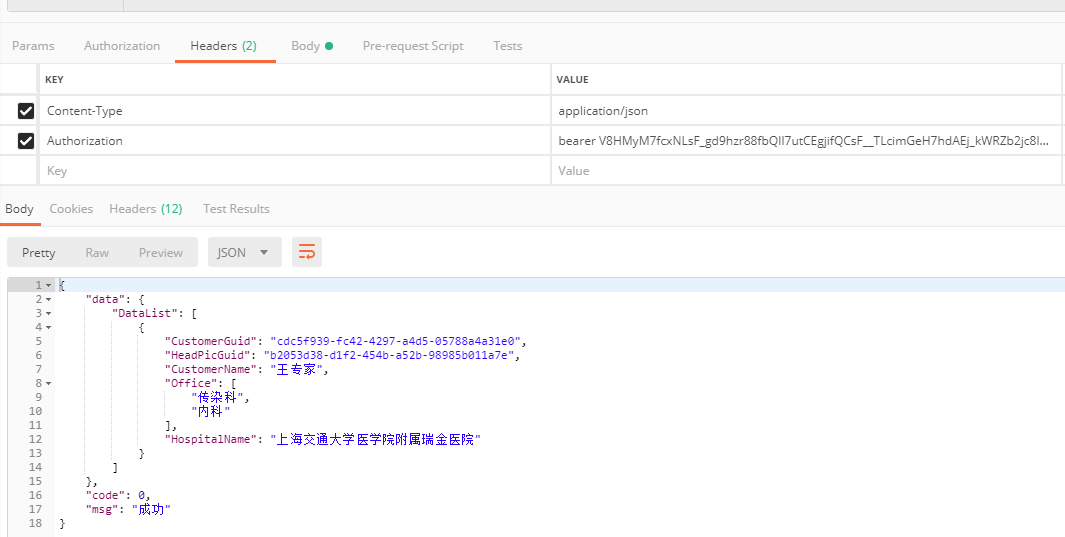
看结果：

请求异常：



换了一个组装方法，还是失败。本来以为是foreach循环中，直接去修改model的值，而引发了错误。

出来了，尼玛。终于知道是哪里出错了。



//根据CustomerGuid集合查询所有医生的机构

foreach (var model in queryCustomer.ToList())

for循环中，ToList就搞定了。原因应该是延时加载吧

里面的friend.HospitalName = queryMember.ToList().FirstOrDefault();

这条一直报错，调到catch住异常。

应该是这里取值，但应该为延时加载，内存中拿不到数据，所以一直报错。

foreach (var model in queryCustomer.ToList())

{

var queryMember = from dmr in \_doctorMemberRepository.Table

where dmr.CustomerGuid == model.CustomerGuid

select dmr.HospitalName;

var queryOffice = from dor in \_doctorOfficeRepository.Table

where dor.CustomerGuid == model.CustomerGuid

select dor.SubjectFirstName;

//model.Office = queryOffice.Distinct().ToList();

NewFriendModel friend = new NewFriendModel();

friend.CustomerGuid = model.CustomerGuid;

friend.CustomerName = model.CustomerName;

friend.HeadPicGuid = model.HeadPicGuid;

friend.HospitalName = queryMember.ToList().FirstOrDefault();

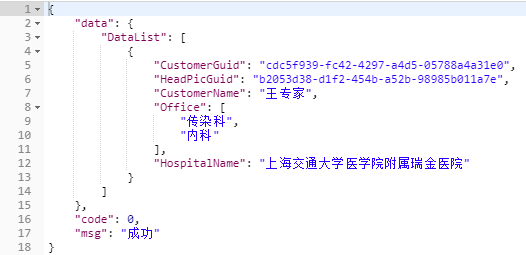
friend.Office = queryOffice.Distinct().ToList();

list.Add(friend);

//model.HospitalName = queryMember.FirstOrDefault();

}

接下来任务就是mapTo一下。

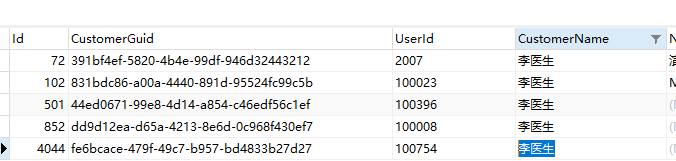


想到，这里guid要的，因为接下来要根据这个guid去添加该好友。

接下来，当用户点击add，就触发这个接口方法：CustomerCircleFriendRequest

也就是发送好友申请请求了。

测试，因为很多李医生，按名字去查询：

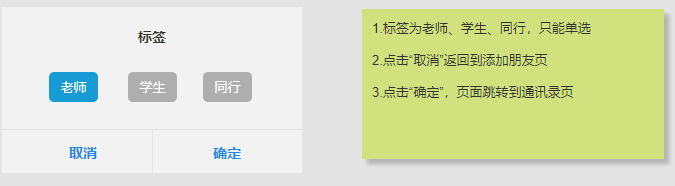


2.2——当点击了添加

交互：点击“添加”按钮，弹出朋友标签分类弹窗

（1）点击添加按钮，post请求返回当前用户的所有分类表

朋友圈分组表中获取该用户的所有分组信息，比方，家人，学生，老师，朋友



（2）单项选择了一个分组后，发送请求加好友的申请

请求参数，当前CustomerId，对方CustomerId，然后在表CustomerCircleRequest中增加这条数据。

FriendRequest

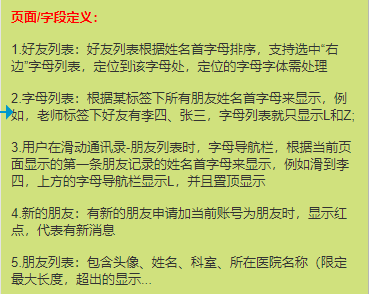
添加好友，dal层用void

bll层用apiModel

3.

逻辑分析:

1.



这里一堆排序，按首字母这些操作，感觉我后台这里只要返回了所有好友信息的list集合，里面带名字了吗？好像没唉，所以要加name，在dal那里，然后给返回到前端，这样拿到了名字前端自己去玩。

还有一种，就是我后台处理好，但这样，前端每点击一次姓名，字母，就要后后台交互一次，感觉费操作，流量了。还是放前端里面好。

2.

朋友列表，以为着每个好友的guid拿到，要去表里查，这个接口应该已经有了。

问题，和李阳确认：

1.

这个分组，老师，学生，同行，是一开始就确认了吗，如果是，就每增加一个医生，就要默认地给他加这个三个分组内容。

2.

3.

开工：

1.先写dal层。操作数据库。

2.再写bll层。写在crm，manager里面。

实现都写在这里面。

但仅限于crm，其他的模块未必这样。

3.wcf实现层，就一句话，因为都写在crm manager里面。

李阳表示，本来dal层的代码是可以写在这里，但因为之前都这么写了，所以现在也这么写。

4.写上契约

5.去web api中添加控制器方法

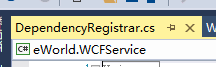
在UserController中写

或者加一个分组的Controller也行。

写接口，HttpPost请求。

6.bll层注入

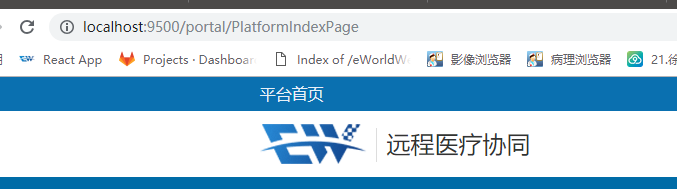
在这里注入，加入这么句话，就出现了。



//朋友圈获取分组

builder.RegisterType<CustomerCircleService>().As<ICustomerCircleService>().InstancePerHttpRequest();

首页终于正常了！开森！



添加，到我这里，customerId已经传过来了，我拿到，访问数据库查询分组信息，默认地给建立老师，学生，朋友，这三个标签。

Tag

List<ServiceCenterManager> GetTagListByCustomerGuid(Guid customerGuid);

List<CustomerCircleGroupModel> GetCustomerCircleGroupListByCustomerGuid(Guid customerGuid);

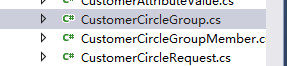
CustomerCircleGroup

遇到：

1.

在eWorld.Entity中，发现数据中的内容，已经建立好了，dbFirst。

我差点去entity中自己建立model类了，好傻。



2.

因为返回的是全部，所以就不需要建dto类了。



3.

ui那边反应过来，先把三个都作为分组，老师，学生，同行。

目前，同行就是显示所有的医生好友，也就是当前的所有好友。

以后可能会加患者，那这样的话，患者+同行才是所有好友了。

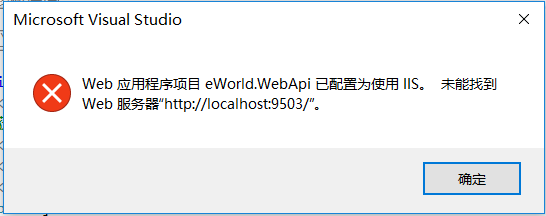
4.

bll层，dal层都加一个注入。

发现：

1.

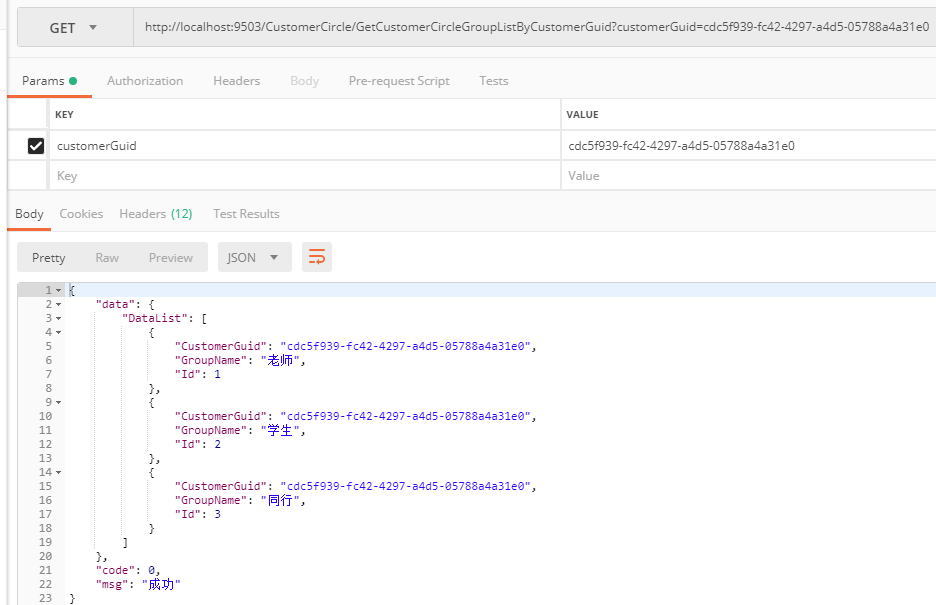
回来，架起来web api



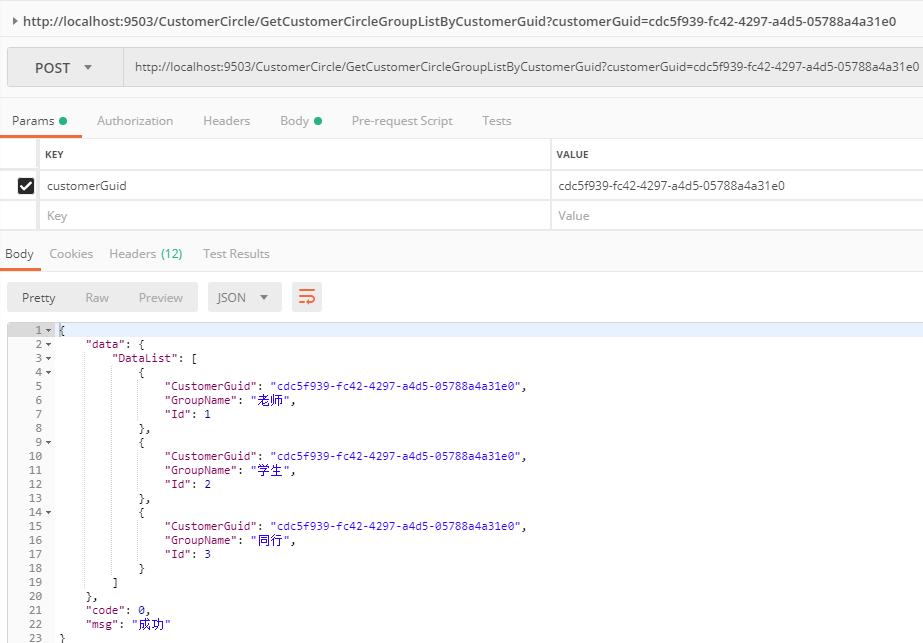
发现不是9500端口尼玛，明天回公司重新用9503端口去获取。

2.

用get测试，搞定了！！！！



哈哈哈。



让我guid不用返回，用outtomap转一下。

又开始学习api：

1.



返回体，现在也是有code，msg，还有泛型T data属性。

还有充分利用http状态码





2.FromUri和FromBody的查找

**FromUri和FromBody用途**

首先，我们来看一段代码：

1 [Route("all01\_2/{id:int?}")] 2 [AcceptVerbs("POST", "GET")] 3 public HttpResponseMessage GetAllStudents01\_2(int id = 0) 4 { 5 var students = db.GetAll(); 6 7 if (id > 0) 8 { 9 students = students.Where(b => b.Id == id).ToList();10 }11 12 return Request.CreateResponse(HttpStatusCode.OK, students);13 }

这里的action传递进来了一个名称id的参数，然后下面做了 id > 0 的判断来获取不同的数据，这种场景我们经常会遇到，并且查询条件的参数不止一个，这样来看我们会不断的在action里面增加传递的参数，格式可能会如此： public HttpResponseMessage GetAllStudents01\_2(int id = 0,参数2,参数3,参数...) ，这样如果需要用到的参数有10个以上，通常看起来不是很方便，这个时候**FromUri和FromBody**的用途就来了；先来看**FromUri**，首先需要我们把上面的那个Action方法的入参格式改成这样：

1 [Route(@"all01\_3/{id:int=0}/{name?}")] 2 [AcceptVerbs("POST", "GET")] 3 public HttpResponseMessage GetAllStudents01\_3([FromUri]MoSearch moSearch) 4 { 5 var students = db.GetAll().AsEnumerable(); 6 7 if (moSearch.Id > 0) 8 { 9 students = students.Where(b => b.Id == moSearch.Id);10 }11 12 if (!string.IsNullOrWhiteSpace(moSearch.Name))13 {14 students = students.Where(b => b.Name.Contains(moSearch.Name));15 }16 17 return Request.CreateResponse(HttpStatusCode.OK, students);18 }

通过实体类 MoSearch 来获取调用端传递的参数，实体类MoSearch定义的属性字段如：

public class MoSearch

{

public int Id { get; set; }

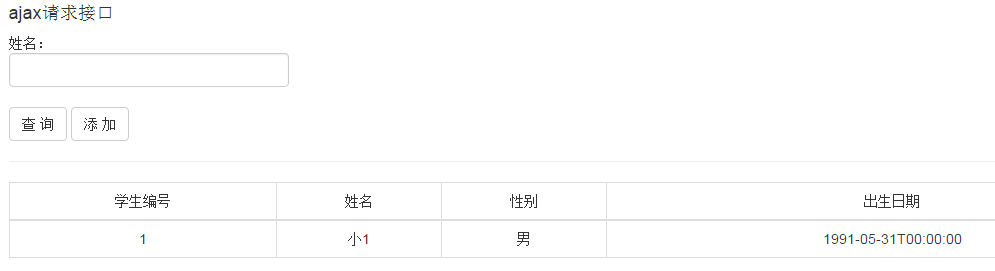
public string Name { get; set; }

}

然后我们改造下查询学生列表的ajax请求方法如：

1 $("#btnSearch").on("click", function () { 2 3 var tabObj = $("#tab tbody"); 4 tabObj.html(‘tr><td colspan="4">加载中...</td></tr>‘); 5 6 7 var url = "http://localhost:1001/s/all01\_3/1"; 8 var txtName = $("input[name=‘txtName‘]").val(); 9 if (txtName.length > 0) {10 url = url + "/" + txtName;11 }12 13 $.get(url, function (data) {14 15 console.log(data);16 17 var tabHtml = [];18 $.each(data, function (i, item) {19 20 tabHtml.push(‘<tr>‘);21 tabHtml.push("<td>" + item.Id + "</td>");22 tabHtml.push("<td>" + item.Name + "</td>");23 tabHtml.push("<td>" + (item.Sex ? "男" : "女") + "</td>");24 tabHtml.push("<td>" + item.Birthday + "</td>");25 tabHtml.push(‘</tr>‘);26 });27 if (tabHtml.length <= 0) { tabHtml.push(‘tr><td colspan="4">暂无数据</td></tr>‘); }28 29 tabObj.html(tabHtml.join(‘‘));30 });31 });

这里Id=1是固定了，直接查询id=1的学生信息，并且有一个可传递的参数学生名称txtName的值，如果用户没有输入学生名称，那么就不会加入到这个get请求中，下面来看下首先不输入学生姓名查询条件的效果：



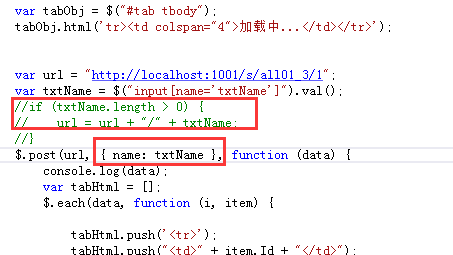
这里能看出来只有学生编号为1的学生被查出来了，下面我们再录入姓名查询条件，能看到结果如：



能看到此时查不到名称为“小2”的学生信息，咋们再改成“小1”试试，



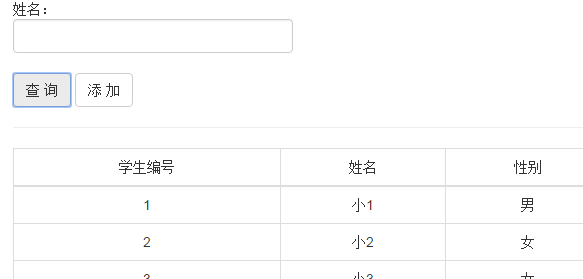
这个时候小1的学生能被查出来，这说明咋们的api的all01\_3能够通过 FromUri]MoSearch moSearch 获取到我们输入的参数；下面我们使用**FromBody**来演示她的效果，首先需要修改js请求webapi格式如：



然后把webapi的all01\_3修改为如下代码：

1 [Route(@"all01\_3/{name?}")] 2 [AcceptVerbs("POST", "GET")] 3 public HttpResponseMessage GetAllStudents01\_3([FromBody]MoSearch moSearch) 4 { 5 var students = db.GetAll().AsEnumerable(); 6 7 if (moSearch.Id > 0) 8 { 9 students = students.Where(b => b.Id == moSearch.Id);10 }11 12 if (!string.IsNullOrWhiteSpace(moSearch.Name))13 {14 students = students.Where(b => b.Name.Contains(moSearch.Name));15 }16 17 return Request.CreateResponse(HttpStatusCode.OK, students);18 }

主要区别吧FromUri换成了FromBody，最后咋们来一起看下没有录入学生名称查询条件的测试效果图：



录入学生查询条件后的效果图：



由上看出能正常查询出来数据，这说明 [FromBody]MoSearch moSearch 获取到了调用客户端的请求参数；好了FromBody和FromUri测试的例子就这些，这两者的区别这里总结下：

FromBody：主要用来获取post方式请求的参数

FromUri：主要获取url地址的请求参数（可以看成get方式）

这两者的效果对比我就不做了，有兴趣的朋友可以自己验证下；本次分享的内容就到这里吧，主要讲解了关于数据模型的一些知识，希望各位喜欢，多多点赞。

3.

这里想到，web api这一层，进来的参数可能很多，用FromUri或者FromBody，需要建立一个对应的参数模型类，而返回的结果，可以需要转化，所以也可能要建立一个output model类。

项目里面的代码就有这样的model类。

[Route("GetSubjectExpertList")]

[HttpPost]

public BaseApiModel<ApiPageListModel<SubjectExpertModel>> GetClinicSubjectExpertList([FromBody]SubjectExpertSearchModel search)

{

search.PageIndex = search.PageIndex - 1;

return \_clinicClient.Invoke(s => s.GetSubjectExpertList(search));

}

我自己也打算用FromBody了，因为请求分组名是guid，但一旦点击了之后，就把name属性的值传进来了。还不如直接就用model模型。

老杨表示这种方法比较麻烦，因为哪怕一个参数，也要写一个model类。

4.

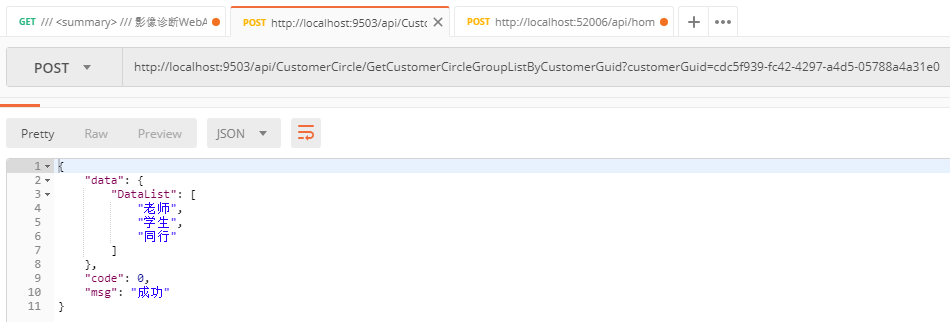
搞了半天，原来错误在于发送jason全部要双引号

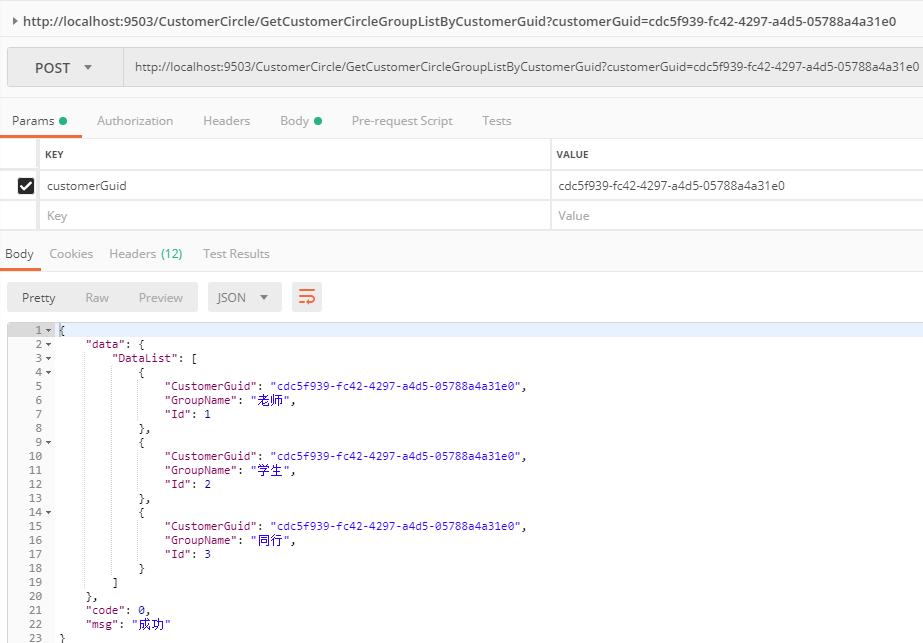


这样就正确了，囧。

现在把list<CustomerCircleGroup>变成了list<string>

结果成了：





一对比发现，不行的。少了字段了。所以还是要改回来，用什么outmap。

李大神的教育：

1.

医生的guid不是传进来的，而是默认当前登录的。

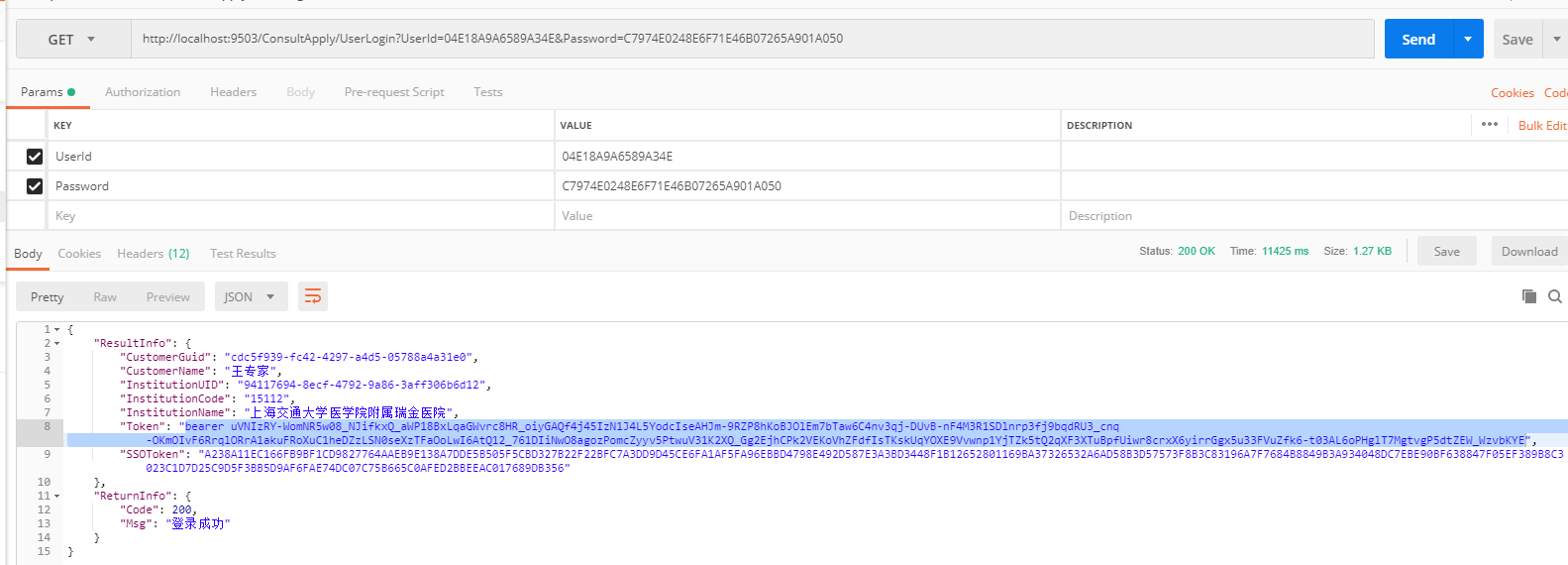
2.

mapto model的用法，以及mapto list<>的方法。

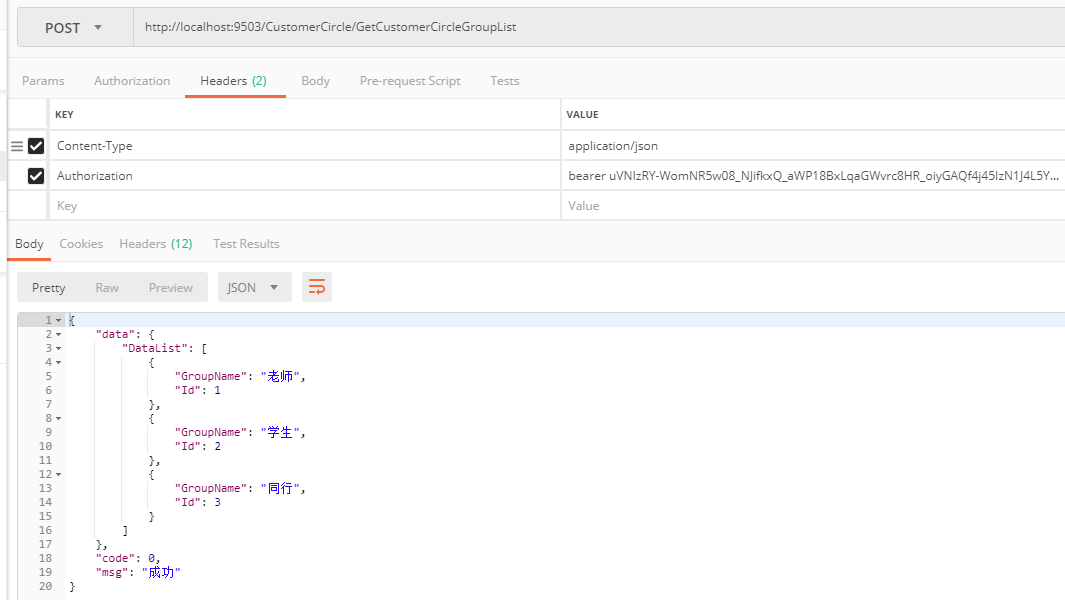
3.

步骤：

（1）先登录获取当前的token



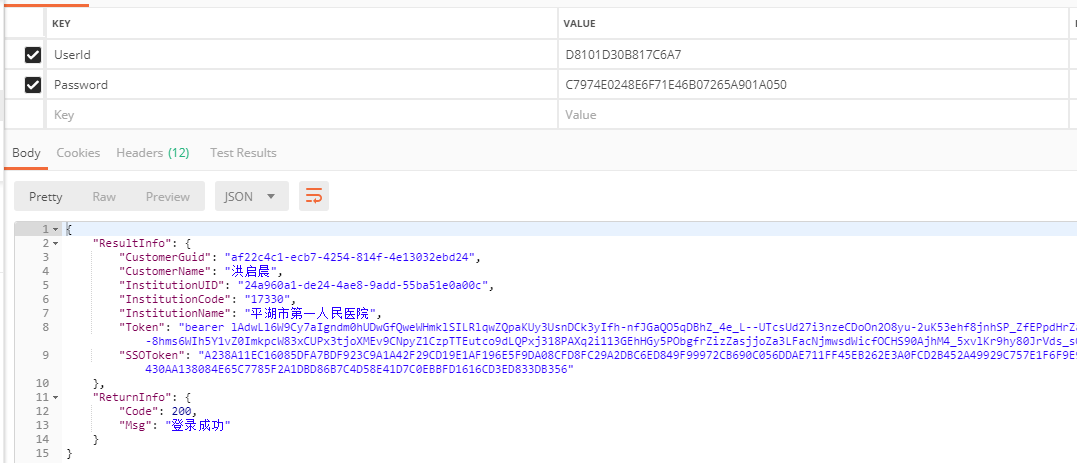
（2）把token复制到当前的接口中，请求分组信息



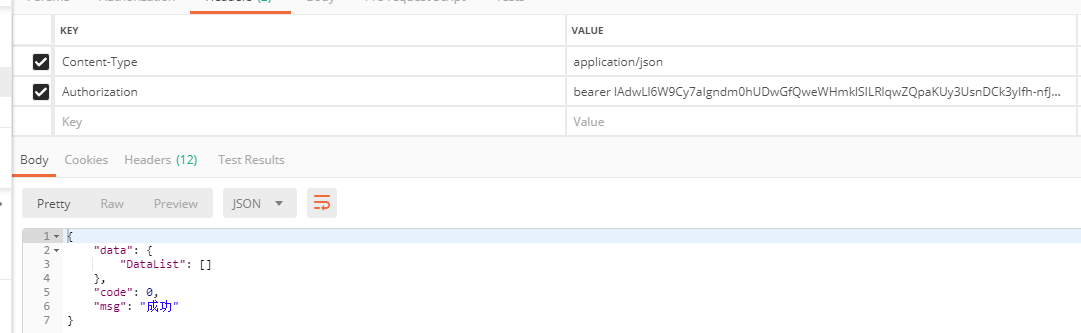
4.

换一个医生测试，看会不会自动生成三个默认分组

（1）登录洪启晨，获取token



获取分组信息

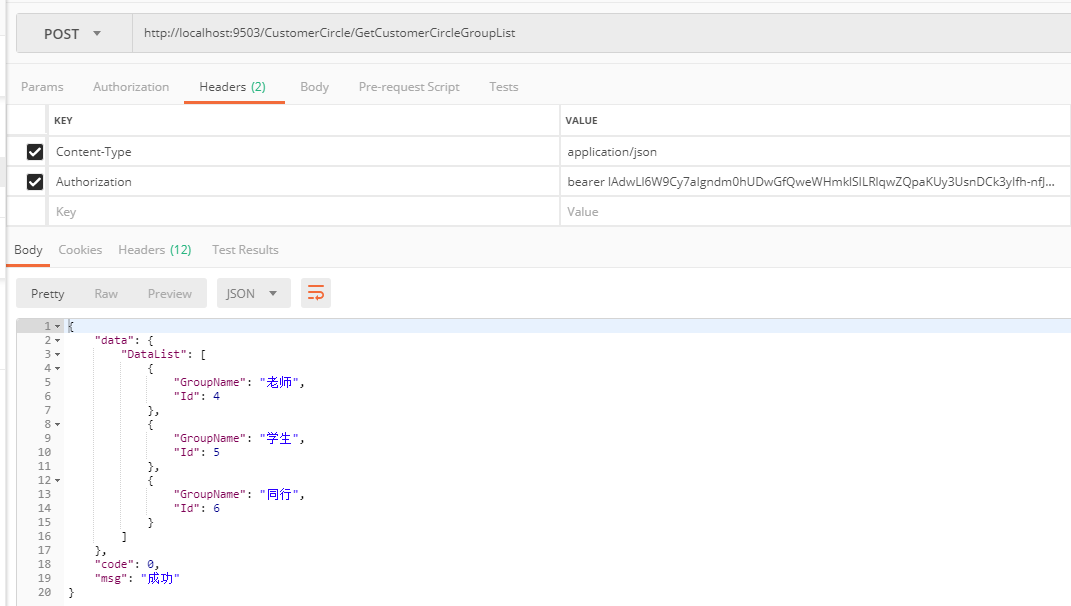


尼玛，没有自动创建啊

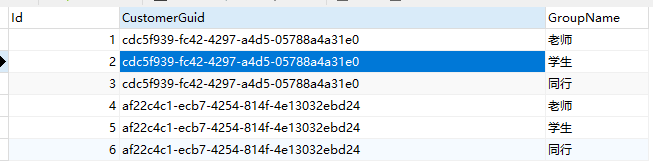
怀疑是延时加载的缘故，所以回去查看。

问了李彪大神，知道了怎么断点调试。把api设置为iis启动，然后断点调试，附加进程。

好了，成功！！！

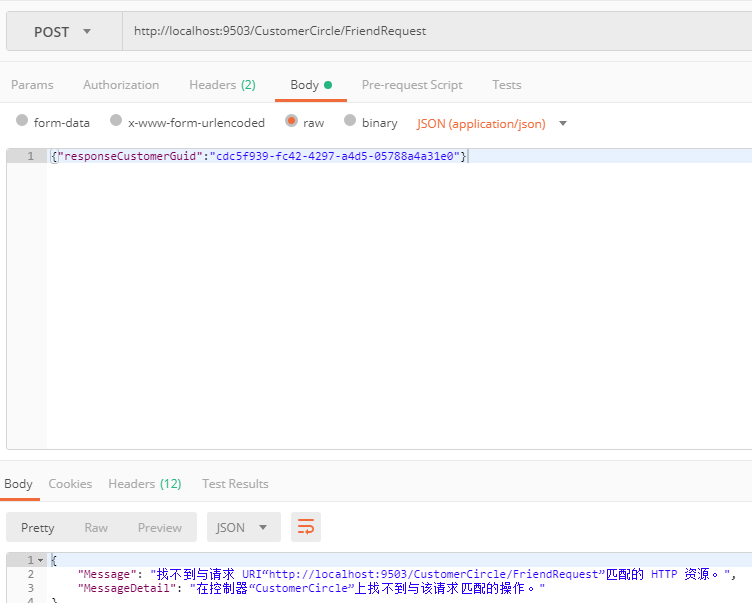


数据库中也进去了：

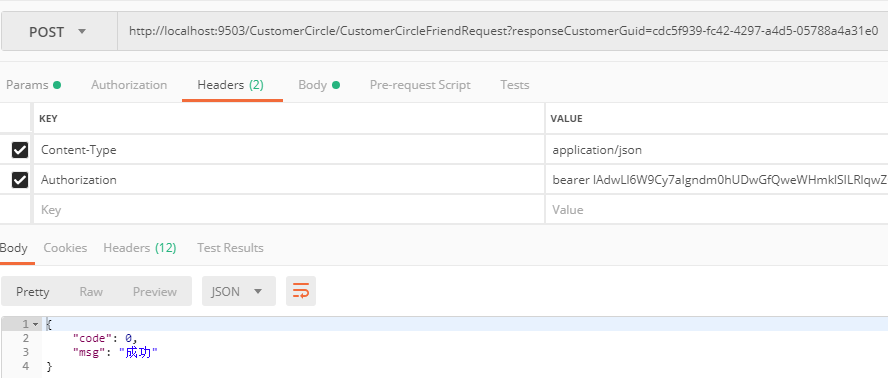


接下来测试，好友申请！

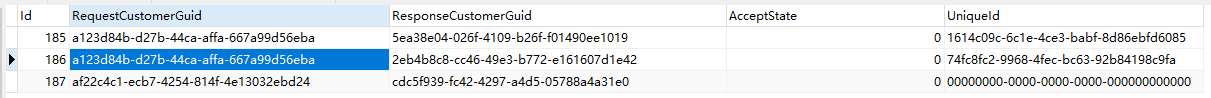
结果：



原来是方法名写得不对，以参数的形式，并且post提交，就ok了。！！

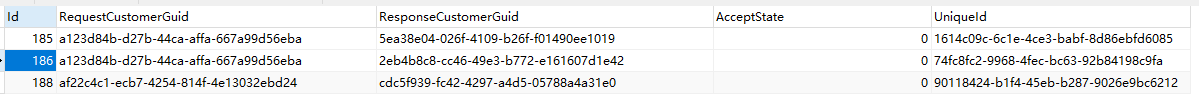


数据库中多了一条数据。



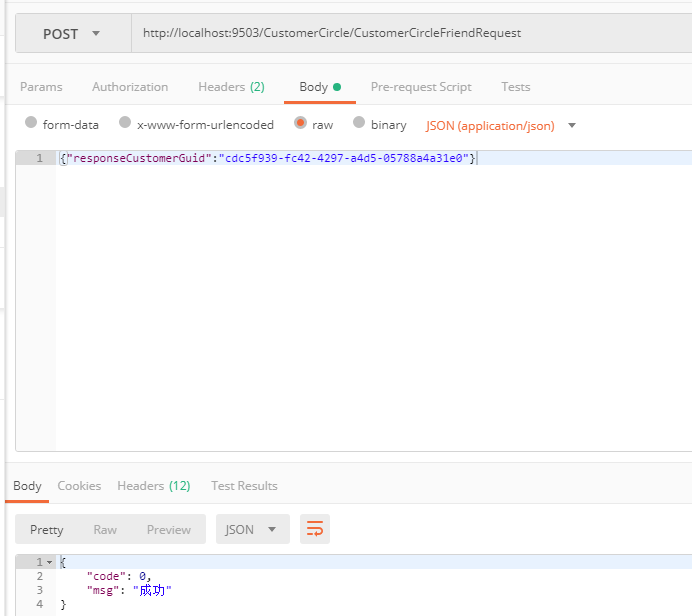
但是唯一guid坑爹了啊。

修改了guid的生成方法后就没有问题了。

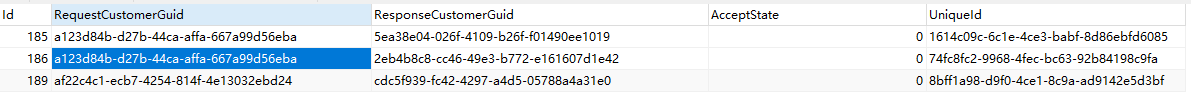


接下来把api这里的参数改为[FromBody]

ok！成功了！



数据库中有了这条记录：

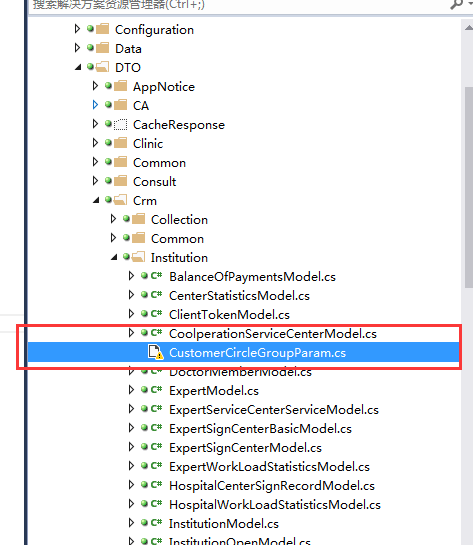


svn提交，下班。

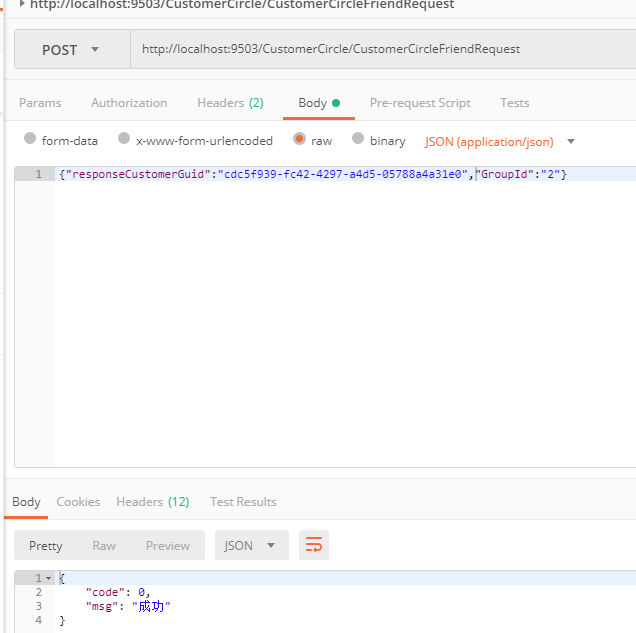


结果提交东西落下了，

喻永红提醒，帮忙，提交了。

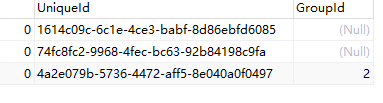


好友申请之后，加入了GroupId字段。



数据库中对应的就是这样的记录，记录了分组信息，当对方同意了之后，就可以获取当前标签名，然后设定对方的分组。





5.

建立分组：

朋友圈默认分组已经建立：

wangzj abc123456 王专家 cdc5f939-fc42-4297-a4d5-05788a4a31e0

04E18A9A6589A34E C7974E0248E6F71E46B07265A901A050

100042 abc123456 洪启晨 af22c4c1-ecb7-4254-814f-4e13032ebd24

D8101D30B817C6A7 C7974E0248E6F71E46B07265A901A050

shenys abc123456 沈医生 a123d84b-d27b-44ca-affa-667a99d56eba

FCE7D9EF8FAAB89F C7974E0248E6F71E46B07265A901A050

xiaozj tomtaw 肖剑 9fb704ce-8041-47d2-bbda-27d39b017cd3

83F2A6DD8490BBFB 122AA5699CCE9444

zhangzj tomtaw 张君1 5ea38e04-026f-4109-b26f-f01490ee1019

FB0A7D3F7AACC610 122AA5699CCE9444

100720 abc123456 会诊诊断医生A 8c71f930-15c4-4e3c-a264-d4ad41dbd1d5

3B8FE393AF88D6A1 C7974E0248E6F71E46B07265A901A050

6.

朋友圈关系默认下

王专家 洪启晨 互为好友

王专家 沈医生 互为好友

7.

老孔在说，那张表它已经设计好了，不需要诊断表，医生回复表，还哪张表，三张表连表查询，现在有一张表，可以搞定。

现在不用连表了。

8.

李阳的指点：

//查询搜索医生列表

//获取医生列表内的CustomerGuid集合

//根据CustomerGuid集合查询所有医生的科室

//组装科室

9.

现在把参数传递都改成model传递，而不是参数。

10.

学习李阳的写法：

crmManager中的具体实现：



#region 根据用户GUID集合查询用户列表

/// <summary>

/// 根据用户GUID集合查询用户列表

/// </summary>

/// <param name="guids"></param>

/// <returns></returns>

public BaseApiModel<ApiListModel<CustomerModel>> GetCustomerByGuids(List<Guid> guids)

{

var result = new BaseApiModel<ApiListModel<CustomerModel>> { code = ResultStatusCode.OK, msg = "成功" };

try

{

if (guids == null || !guids.Any())

{

result.code = ResultStatusCode.LACK\_PARA;

result.msg = "缺少必要参数";

return result;

}

result.data = new ApiListModel<CustomerModel>

{

把转换放到了bll层，而在dal层只是根据guid字符串查出对应的医生数据。所有数据都返回，在bll层才正式

DataList = \_customerService.GetCustomerByGuids(guids).Select(item => new CustomerModel

{

CustomerGuid = item.CustomerGuid,

HeadPicGuid = item.HeadPicGuid,

Sex = item.Sex

}).ToList()

};

}

catch (Exception ex)

{

WriteLog("协同-[crm公共库]-根据用户GUID集合查询用户列表", $"传入参数：{JsonConvert.SerializeObject(guids)},详细错误：{ex}", "", "CrmManage/GetCustomerByGuids", "");

result.code = ResultStatusCode.DEFAULT\_ERROR;

result.msg = "查询异常";

}

return result;

}

#endregion

Dal层是返回数据，不做处理：

public List<Customer> GetCustomerByGuids(List<Guid> guids)

{

if (guids == null || guids.Count <= 0)

return null;

var query = from s in \_customerRepository.Table

where guids.Contains(s.CustomerGuid) && s.State != -5 && s.State >= 10//&& s.Deleted == false && s.Active == true

select s;

return query.ToList();

}

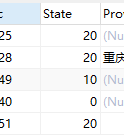
11.

改了查找用户，加了state状态后

查找李医生，就少了一个内容了。



看了state。



12.

看了老马的项目也是，dal层不参与逻辑运算，全是和数据库的交互，所以我之后方法也是全部在bll层调用dal层的方法，然后在bll层去判断，某个字段是否满足条件，然后在这里进行new一个model，进行数据的重新组装，然后返回给webapi。

13.

这个参数验证的方法也是写得碉堡了。

namespace eWorld.Framework.Utils

{

/// <summary>

/// 参数验证扩展

/// </summary>

public class ValidateParaExtend<T>

{

/// <summary>

///

/// </summary>

public bool IsStop { get; set; }

/// <summary>

///

/// </summary>

public string Msg { get; set; }

/// <summary>

///

/// </summary>

public T Model { get; private set; }

/// <summary>

///

/// </summary>

/// <param name="t"></param>

public ValidateParaExtend(T t)

{

Model = t;

}

}

/// <summary>

///

/// </summary>

public static class ParaExtend

{

/// <summary>

///

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="t"></param>

/// <param name="msg"></param>

/// <returns></returns>

public static ValidateParaExtend<T> IsNull<T>(this ValidateParaExtend<T> t, string msg)

{

if (t.Model == null)

{

t.Msg = msg;

t.IsStop = true;

}

return t;

}

/// <summary>

///

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="t"></param>

/// <param name="pre"></param>

/// <param name="msg"></param>

/// <returns></returns>

public static ValidateParaExtend<T> Check<T>(this ValidateParaExtend<T> t, Predicate<T> pre, string msg)

{

if (t.IsStop) return t;

if (pre(t.Model))

{

t.Msg = msg;

t.IsStop = true;

}

return t;

}

}

}

这样调用验证：

/// <summary>

/// 好友申请

/// </summary>

/// <param name="param">好友申请参数</param>

/// <returns></returns>

public BaseApiModel CustomerCircleFriendRequest(FriendRequestParam param)

{

var result = new BaseApiModel { code = ResultStatusCode.OK, msg = "成功" };

try

{

#region 参数校验

var ckResult = new ValidateParaExtend<FriendRequestParam>(param).IsNull("参数不能为空")

.Check(p => p.RequestCustomerGuid == Guid.Empty, "医生唯一号不能为空")

.Check(p => p.ResponseCustomerGuid == Guid.Empty, "对方唯一号不能为空")

.Check(p => p.GroupId < 1, "分组ID错误");

if (ckResult.IsStop)

{

result.code = ResultStatusCode.LACK\_PARA;

result.msg = ckResult.Msg;

return result;

}

14.

GetGroupInfoById

15.

测试，挨个登陆，都加一次

16.



自己请求自己也能，居然，改bug！

17.

李阳这里的参数，传的就是当前的guid，还有分组的GroupId，换我设计，就是传一个字符串，名字是分组名。好low。因为数据库根本没有这个名词的字段，所以，要传的肯定是数据库字段存在的。

那这里怎么区分，app怎么给groupId呢，我好sb，不是有另外一个获取分组信息的接口么，app先调用这个接口，然后再把对应标签的GroupId给传过来。

因为这里要传一个GroupId，所以获取分组信息的时候，发GroupId也发送给app端。

#region 获取通讯录信息

/// <summary>

/// 获取通讯录信息

/// </summary>

/// <param name="param"></param>

/// <returns></returns>

public List<AddressBookModel> GetAddressBookList(AddressBookParam param)

{

var query = from ccgm in \_customerCircleGroupMemberRepository.Table

join c in \_customerRepository.Table on ccgm.FriendCustomerGuid equals c.CustomerGuid

join dm in \_doctorMemberRepository.Table on ccgm.FriendCustomerGuid equals dm.CustomerGuid

where ccgm.GroupId == param.GroupId

select new AddressBookModel

{

FriendCustomerGuid = ccgm.FriendCustomerGuid,

CustomerName = c.CustomerName,

HospitalName = dm.HospitalName

};

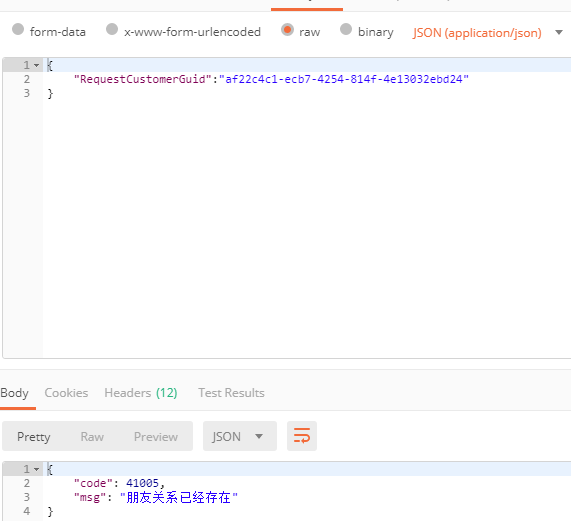
return query.ToList();

}

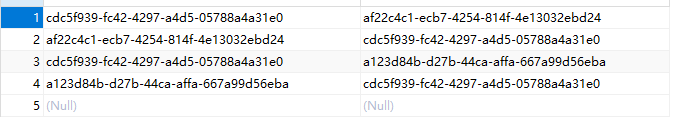
18.

测试 王专家登陆，处理洪启晨的申请，洪启晨设置的是6，同行，那对应的，王专家的同行是3，3中对应的是洪启晨的guid，另外，我的朋友圈表中，二人互为好友。

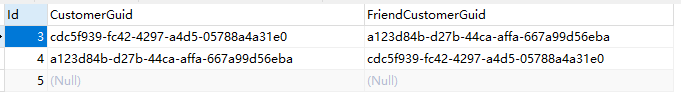
测试！



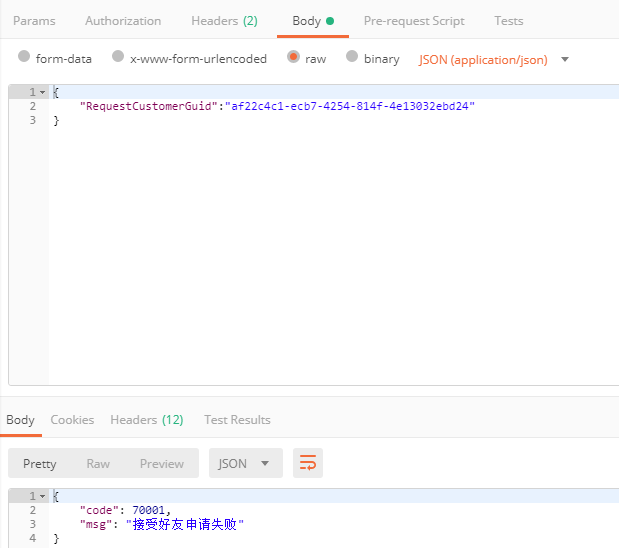
确实啊，朋友关系已经存在了。



删了。



再次申请：



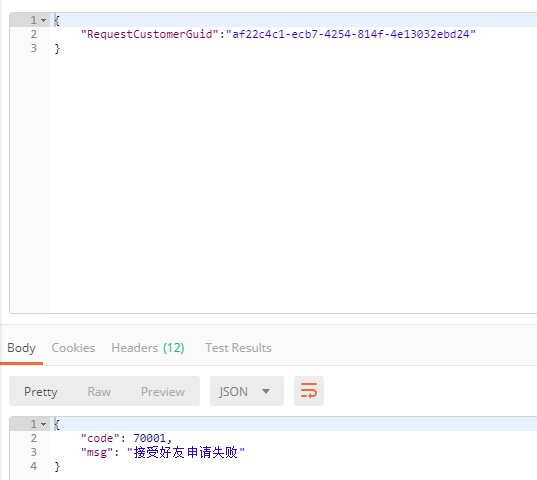
度娘，要开启分布式服务。



开启了。



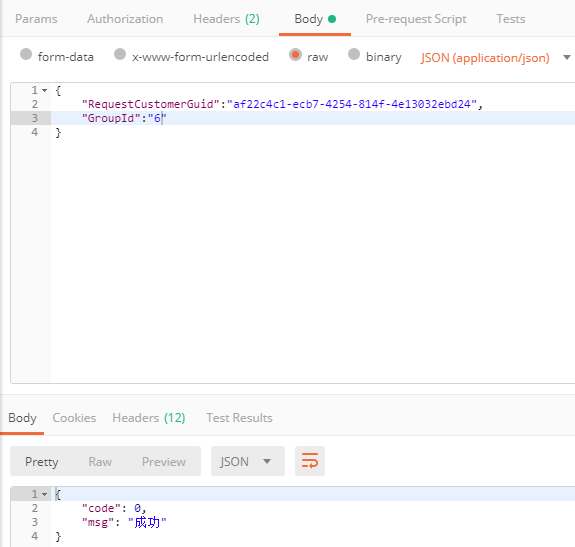
还是失败：



调试发现，这里没有传参，为0，而且check的时候我忘了验证了，因为是后面加的属性。



修改之后，成功！



验证。

请求表中，状态值从0变成了10



朋友圈中加了两条信息，互为好友



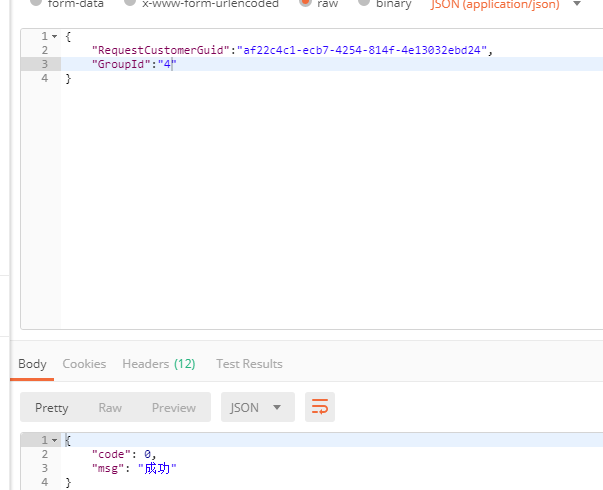
朋友圈分组成员表中增加了洪启晨，并且放在了3，也就是同行的表中。



ok！

接下来删除，然后加到老师表中。

分组id变成4.



请求表，0变10，正确。



我的朋友圈多了两条数据，get。

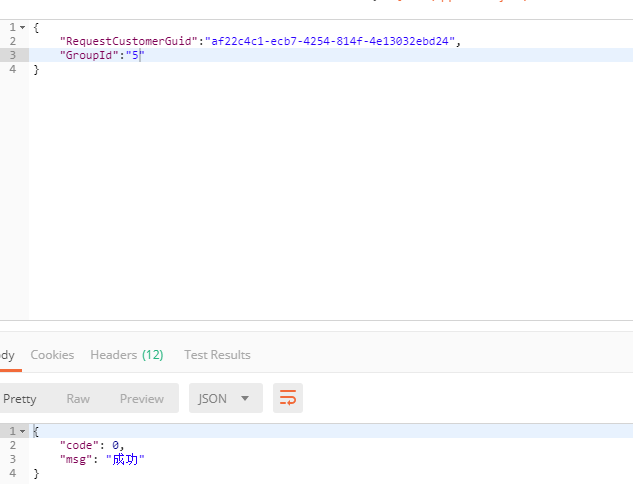


王专家，在2，也就是学生中，找到该条数据。get。



测试最后的一个学生。

成功。

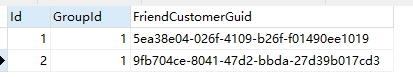


三个成功了，但我发现，groupID，其实在请求表中已经有了，尼玛，那就不需要传了，fuck。

GetCustomerCircleRequest

还有，同意的时候，必须保证AcceptState是0，有没有这个问题。

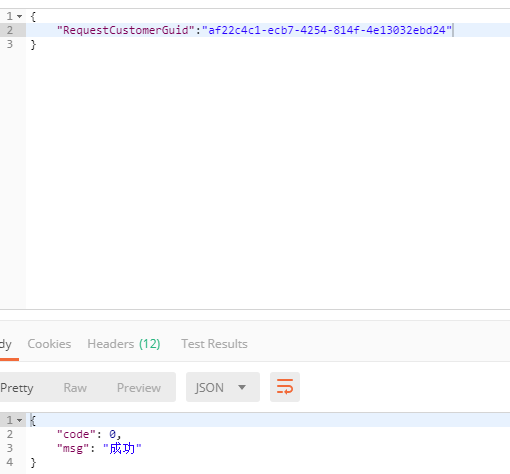
还有bug，插入数据，可能已经有了，就是CustomerCircleGroupMember中，



其实可以不区分，这样就可以在多个标签中了，一个人。

考虑下。这里顺便写进去了。

修改之后测试，因为默认是6，也就是同行。



成功。

查看数据库。

0变成了10 。



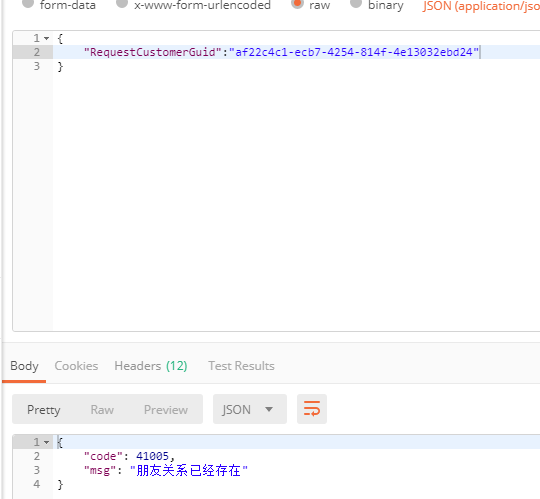
增加了两条数据。



在专家中写入数据。

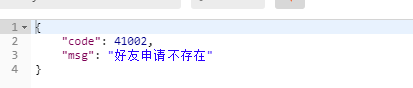


不改状态请求下，看是否是已经存在的报错。



朋友关系已经存在。

然后我删掉两个便宜关系。

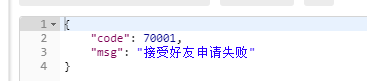


因为AcceptState是10，这样都验证了，暂时能想到的bug都解决了。

随便测试一个AcceptState是错误的。看报不报错。



报错的。

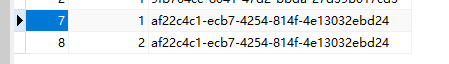


但是这个报错，得修改。让它报不存在的错。

测试学生，和老师。通过。

还有个bug。

王专家同意之后，现在洪启晨在了两个分组中。要通过代码杜绝这种情况。



验证朋友圈分组成员是否已经存在，加了，添加朋友的时候感觉也要加啊。

还有分组信息的添加，得双方都加，现在我只加了接收方。尼玛。

申请方和申请id，

现在的版本定死了，学生对老师，李阳新加了一张表，到时候会放redis里面的，去这张表中找对应关系。

另外，在请求方的“老师”中加数据，接收方的“学生”中加数据，同时在双方的同行中加数据，还没有去实现。

暂时能想到的，也写死的，都实现了。下班！

wangzj abc123456 王专家 cdc5f939-fc42-4297-a4d5-05788a4a31e0

04E18A9A6589A34E C7974E0248E6F71E46B07265A901A050

100042 abc123456 洪启晨 af22c4c1-ecb7-4254-814f-4e13032ebd24

D8101D30B817C6A7 C7974E0248E6F71E46B07265A901A050

19.

周报：

实现用户名或手机号搜索好友的接口

实现获取医生分组标签名的接口

实现好友申请的接口

实现接收好友申请的接口

实现新朋友栏目消息列表的接口

20.

我在接受好友申请的实现中做了 用户是否已经存在于分组当中的判断。

其实只验证上面的，两者是否已经存在朋友圈关系就可以了，因为实际的写入数据库，用的事务，同生共死，如果写入了朋友圈分组信息表，那势必写入了我的朋友圈这个数据库。当然，这是在排除了人为修改朋友圈的前提下。

所以，朋友圈申请的时候，我就没有去验证朋友圈分组信息，我默认是同生共死，没有人为去修改数据库的。

但完美主义，不给自己挖坑的情况下，我还是验证吧。囧= =

//判断好友关系是否已经存在

var myCustomerCircle = \_customerCircleService.GetMyCustomerCircleById(param.RequestCustomerGuid,

param.ResponseCustomerGuid);

if (myCustomerCircle != null)

{

result.code = ResultStatusCode.DATA\_ALREADY\_EXIST;

result.msg = "朋友关系已经存在";

return result;

}

myCustomerCircle = \_customerCircleService.GetMyCustomerCircleById(param.ResponseCustomerGuid,

param.RequestCustomerGuid);

if (myCustomerCircle != null)

{

result.code = ResultStatusCode.DATA\_ALREADY\_EXIST;

result.msg = "朋友关系已经存在";

return result;

}

//验证该条好友申请是否存在

var customerCircle = \_customerCircleService.GetCustomerCircleRequestInfo(param.RequestCustomerGuid,

param.ResponseCustomerGuid, 0);

if (customerCircle == null)

{

result.code = ResultStatusCode.DATA\_DONOT\_EXIST;

result.msg = "好友申请不存在";

return result;

}

var groupId = customerCircle.GroupId.Value;

//验证朋友圈分组成员是否已经存在

//获取所有的分组

var list = \_customerCircleService.GetCustomerCircleGroupListByCustomerGuid(param.ResponseCustomerGuid);

foreach (var customerCircleGroup in list)

{

var groupMember = \_customerCircleService.GetCustomerCircleGroupMemberInfo(customerCircleGroup.Id,

param.RequestCustomerGuid);

if (groupMember != null)

{

result.code = ResultStatusCode.DATA\_ALREADY\_EXIST;

result.msg = "朋友圈分组成员已经存在";

return result;

}

}

//请求方的分组成员验证

list = \_customerCircleService.GetCustomerCircleGroupListByCustomerGuid(param.RequestCustomerGuid);

foreach (var customerCircleGroup in list)

{

var groupMember = \_customerCircleService.GetCustomerCircleGroupMemberInfo(customerCircleGroup.Id,

param.ResponseCustomerGuid);

if (groupMember != null)

{

result.code = ResultStatusCode.DATA\_ALREADY\_EXIST;

result.msg = "朋友圈分组成员已经存在";

return result;

}

}

21.

终于理解彪哥写的方法的妙用了！李阳写的获取通讯录，也是参照的这个方法。

之前我不知道，这里为什么对这么多的参数可能性进行判断，现在明白了，这是对所有可能的查询条件进行封装，这样后续无论是什么条件，这里都可以获取想要的结果。

这就是复用吧，一个方法应变所有可能，而且里面是iQueryable，延时加载，所以最终会在ToList()的时候才生成最终sql语句。

#region 获取医生科室列表

/// <summary>

/// 获取医生科室列表

/// </summary>

/// <returns></returns>

public List<DoctorOffice> GetDoctorOfficeList(DoctorOfficeParam param)

{

var query = from doctorOffice in \_doctorOfficeRepository.Table

select doctorOffice;

if (param.CustomerGuid != Guid.Empty)

{

query = query.Where(p => p.CustomerGuid == param.CustomerGuid);

}

if (param.CustomerGuidList != null && param.CustomerGuidList.Count > 0)

{

query = query.Where(p => param.CustomerGuidList.Contains(p.CustomerGuid));

}

return query.ToList();

}

#endregion

22.

关于现在写死的关系，喻永红说是用枚举，感觉不错的。李阳这边是新写了一个数据库内容，按这个去查找。但是太频繁，每次都要查询。

处理好友删除

DeleteFriend

删除 我的朋友圈 表的互为好友关系。

删除已同意好友申请的状态

删除朋友圈分组信息中的互相关系

1. 己方学生，删除对方 老师（同行要保留吗）
2. 己方老师，删除对方 学生
3. 己方同行，删除对方 同行

今天先模仿一波彪哥和李阳的查询。

实现下，不能查询到自己

23.

修改原来的代码，所有用到标签的地方，都去Commom数据库中的DataDicInfo表中查询数据。

从获取分组开始，字典类型是朋友圈标记的值，

LookupKey=FriendTag

把所有的DicName给罗列，并且都写进标签中。我还是满足都拥有着这三个标签，不读，

获取朋友圈分组信息，更换，搞定。

接下来是同意好友申请。

如果是老师和学生的话，查找

24.

上面没有做判断

//变成事务

using (TransactionScope transaction = new TransactionScope())

{

//更新朋友圈请求的状态

\_customerCircleService.ChangeRequestAcceptState(param.RequestCustomerGuid, param.ResponseCustomerGuid, 10);

//分别建立朋友圈关系

\_customerCircleService.AddMyCustomerCircle(param.RequestCustomerGuid, param.ResponseCustomerGuid);

\_customerCircleService.AddMyCustomerCircle(param.ResponseCustomerGuid, param.RequestCustomerGuid);

//在申请方的朋友圈分组成员中添加一条数据

//还得去查询，是否是老师，学生，不然的话，同行里面也必须加一条数据

CustomerCircleGroupMember entity=new CustomerCircleGroupMember()

{

FriendCustomerGuid=param.ResponseCustomerGuid,

GroupId=groupId

};

\_customerCircleService.AddCustomerCircleGroupMember(entity);

//现在添加

var groupModel = \_customerCircleService.GetGroupModel(new CustomerCircleGroupDetailParam()

{

CustomerGuid = param.RequestCustomerGuid,

GroupId = groupId

});

string groupName = groupModel.GroupName;

//

var entityDic = \_dataDicInfoService.GetDataDicInfoByDicName(groupName);

//var entity =

// GetGroupModel(new CustomerCircleGroupDetailParam()

// {

// CustomerGuid = requestCustomerGuid,

// GroupId = groupId

// });

//string groupName = entity.GroupName;

//根据对方设置的标签反向设置标签

\_customerCircleService.ReverseSetLabel(param.RequestCustomerGuid, param.ResponseCustomerGuid, groupId,entityDic);

transaction.Complete();

}

25.

问了李阳，让我在事务里面只做增删改，别去这些判断

好友删除

事务

（1）

删除 我的朋友圈 表——MyCustomerCircle，双方的数据

（2）

删除 朋友圈分组成员 CustomerCircleGroupMember中，彼此的数据。

这里，如果删除的是学生，对方老师标签中的数据消失。同行的数据保留。

如果删除的是同行，则对方同行的数据删除。学生，老师标签中的数据继续保存？

如果是同行，先把彼此同行的数据删除。

然后查询该好友是否还有其他的分组信息

1. 获取当前用户的所有分组信息，从里面获取所有的GroupId，list<int>
2. 在朋友圈分组成员表中查询，包含contains该GroupId，并且friendGuid是对方Guid的所有数据，count，只要>0，说明有数据，就全部删除。
3. 反向同样获得所有的GroupId的list<int>，然后去朋友圈分组成员中，把包含的所有数据都给删除。

（3）

申请状态要删除吗？

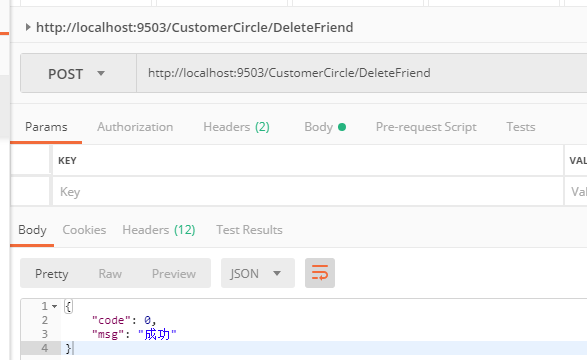
入参：

分组ID，GroupId，对方Guid，被删除BeenDeletedCustomerGuid

DeleteFriend

还要通知对方，删除了。亮哥说是有成熟的接口，可以发消息，发短信，明天问李阳。

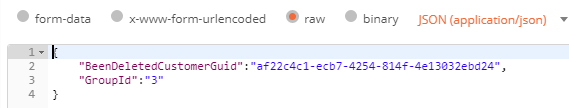
测试，



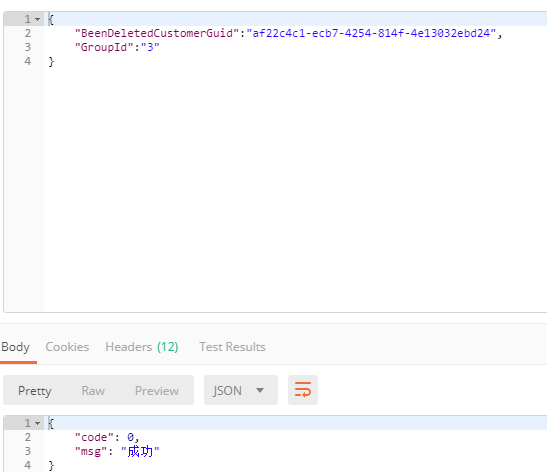
居然一次就通过，别骗我啊！！！

现在验证老师和学生

先测试删除好友，看是否全部删除干净：



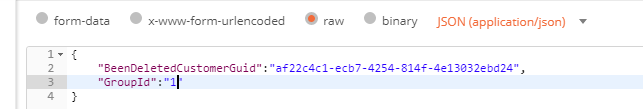
显示成功：



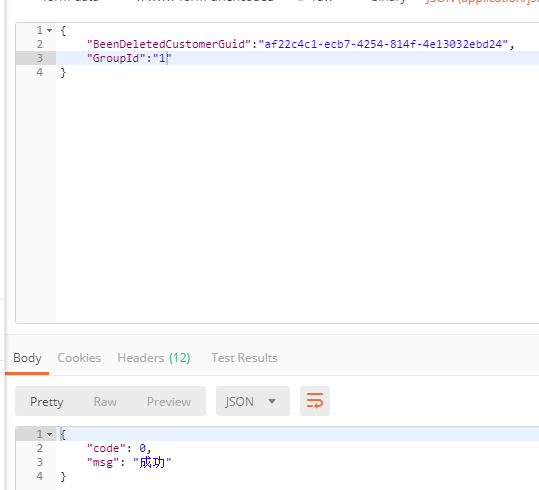
数据库中也干净了。ok。

现在测试删除老师，或者学生。

洪启晨加的是5，也就是学生，并且6，同行中也有。王专家这边好友是1学生，外加3，同行。



删除显示：



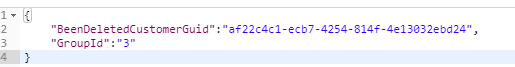
我的朋友圈表，互相好友数据消失。

但朋友圈分组成员表中，彼此的同行数据还是存在的。

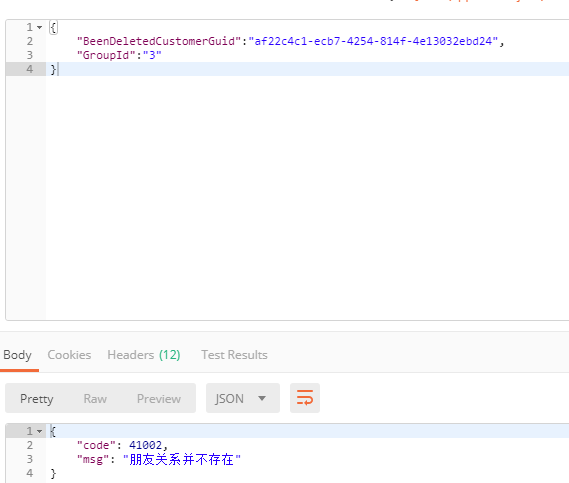


所以，验证是通过的。

接下来删除同行：



但显示朋友关系并不存在。。。。



所以单独删除老师，学生标签的时候，不能删除 我的朋友圈 表中的互为朋友关系数据。

26.

之前写的，现在没用了。就删除了。

接口：

#region 根据对方设置的标签反向设置标签

/// <summary>

/// 根据对方设置的标签反向设置标签

/// </summary>

/// <param name="requestCustomerGuid">请求方</param>

/// <param name="responseCustomerGuid">接收方</param>

/// <param name="groupId">分组ID</param>

void ReverseSetLabel(Guid requestCustomerGuid, Guid responseCustomerGuid, int groupId, DataDicInfoModelParam dataDicInfoModel);

#endregion

dal层的实现：

#region 根据对方设置的标签反向设置标签（第一版，基本没用了，现在）（2018年10月30日13:47:28）

/// <summary>

/// 根据对方设置的标签反向设置标签

/// </summary>

/// <param name="requestCustomerGuid">请求方</param>

/// <param name="responseCustomerGuid">接收方</param>

/// <param name="groupId">分组ID</param>

public void ReverseSetLabel(Guid requestCustomerGuid, Guid responseCustomerGuid, int groupId, DataDicInfoModelParam dataDicInfoModel)

{

var entity =

GetGroupModel(new CustomerCircleGroupDetailParam()

{

CustomerGuid = requestCustomerGuid,

GroupId = groupId

});

string groupName = entity.GroupName;

//首先和默认的 “老师” “学生” “同行” 三个标签进行比较，后续增加自定义标签，再继续添加到预留的if判断中

//获取DataDicInfo表中默认需要加入的标签数据，第一版默认是“老师”，“学生”，“同行”，LookupKe=FriendTag，DicExtend=0

// var listDicName =

//\_dataDicInfoRepository.Table.Where(d => d.LookupKey == "FriendTag" && d.DicExtend == "0").Select(c => c.DicName).ToList();

//var listA = from d in \_dataDicInfoRepository.Table

// where d.DicExtend == "0"

// select d;

//var query =

// (from dd in \_dataDicInfoRepository.Table

// where groupName.Contains(dd.DicName)

// select dd).ToList();

//var listB = from d in \_dataDicInfoRepository.Table

// where d.DicExtend == "0"

// select d;

//var n = query.Count();

//int a = 9;

//var dataDicInfoModel = query.ToList().FirstOrDefault();

// \_dataDicInfoRepository.Table.Where(d => d.DicName == groupName).ToList().FirstOrDefault();

//var dataDicInfo = \_dataDicInfoRepository.Table.Where(d => d.DicName == groupName).FirstOrDefault();

//如果是在默认的三个标签内

if (dataDicInfoModel.DicExtend == "0" && dataDicInfoModel.LookupKey == "FriendTag")

{

//第一版默认是“老师”，“学生”，“同行”，LookupKe=FriendTag，DicExtend=0

//var list = \_dataDicInfoRepository.Table.Where(d => d.LookupKey == "FriendTag" && d.DicExtend == "0");

//如果是老师 或者 学生。

if (!string.IsNullOrEmpty(dataDicInfoModel.DicCode) && !string.IsNullOrEmpty(dataDicInfoModel.DicComment))

{

//在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

//查询 学生 所在分组的GroupId

//----这里是之前的，默认是老师

//找到对应的标签名字

//获取反向标签名

var reverseDicName = \_dataDicInfoRepository.Table.Where(d => d.LookupKey == "FriendTag" && d.DicExtend == "0" && d.DicCode == dataDicInfoModel.DicComment).Select(d => d.DicName).FirstOrDefault();

//list.Where(d => d.DicCode == dataDicInfoModel.DicComment).Select(d => d.DicName).FirstOrDefault();

var responseGroupId = GetGroupModel(responseCustomerGuid, reverseDicName).Id;

//插入该条数据

AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

//并把数据加到同行里面

//获取默认的同行的dicName

var defaultDicName =

\_dataDicInfoRepository.Table

.Where(d => d.LookupKey == "FriendTag" && d.DicExtend == "0" && string.IsNullOrEmpty(d.DicCode) && string.IsNullOrEmpty(d.DicComment))

.Select(d => d.DicName)

.FirstOrDefault();

responseGroupId = GetGroupModel(responseCustomerGuid, defaultDicName).Id;

AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

//同时加到请求方的同行里面

var requestGroupId = GetGroupModel(requestCustomerGuid, defaultDicName).Id;

AddCustomerCircleGroupMember(requestGroupId, responseCustomerGuid);

}

//如果是同行

if (string.IsNullOrEmpty(dataDicInfoModel.DicCode) && string.IsNullOrEmpty(dataDicInfoModel.DicComment))

{

//在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

//查询 学生 所在分组的GroupId

//获取默认的同行的dicName

var defaultDicName =

\_dataDicInfoRepository.Table

.Where(d => d.LookupKey == "FriendTag" && d.DicExtend == "0" && string.IsNullOrEmpty(d.DicCode) && string.IsNullOrEmpty(d.DicComment))

.Select(d => d.DicName)

.FirstOrDefault();

var responseGroupId = GetGroupModel(responseCustomerGuid, defaultDicName).Id;

//插入该条数据

AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

}

////获取DataDicInfo表中默认需要加入的标签数据，第一版默认是“老师”，“学生”，“同行”，LookupKe=FriendTag，DicExtend=0

//var list =\_dataDicInfoRepository.Table.Where(d => d.DicName == groupName).ToList();

//foreach (var dataDicInfo in list)

//{

// //第一版默认是“老师”，“学生”，“同行”，LookupKe=FriendTag，DicExtend=0

// if (dataDicInfo.DicExtend == "0" && dataDicInfo.LookupKey == "FriendTag")

// {

// //如果是老师 或者 学生。

// if (!string.IsNullOrEmpty(dataDicInfo.DicCode) && !string.IsNullOrEmpty(dataDicInfo.DicComment))

// {

// //在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

// //查询 学生 所在分组的GroupId

// //----这里是之前的，默认是老师

// //找到对应的标签名字

// //获取反向标签名

// var reverseDicName =

// list.Where(d => d.DicCode == dataDicInfo.DicComment).Select(d => d.DicName).FirstOrDefault();

// var responseGroupId = GetGroupModel(responseCustomerGuid, reverseDicName).Id;

// //插入该条数据

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

// //并把数据加到同行里面

// //获取默认的同行的dicName

// var defaultDicName =

// list.Where(d => string.IsNullOrEmpty(d.DicCode) && string.IsNullOrEmpty(d.DicComment))

// .Select(d => d.DicName)

// .FirstOrDefault();

// responseGroupId = GetGroupModel(responseCustomerGuid, defaultDicName).Id;

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

// //同时加到请求方的同行里面

// var requestGroupId = GetGroupModel(requestCustomerGuid, defaultDicName).Id;

// AddCustomerCircleGroupMember(requestGroupId, responseCustomerGuid);

// }

// //如果是同行

// if (string.IsNullOrEmpty(dataDicInfo.DicCode) && string.IsNullOrEmpty(dataDicInfo.DicComment))

// {

// //在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

// //查询 学生 所在分组的GroupId

// //获取默认的同行的dicName

// var defaultDicName =

// list.Where(d => string.IsNullOrEmpty(d.DicCode) && string.IsNullOrEmpty(d.DicComment))

// .Select(d => d.DicName)

// .FirstOrDefault();

// var responseGroupId = GetGroupModel(responseCustomerGuid, defaultDicName).Id;

// //插入该条数据

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

// }

// }

}

//如果是在自定义，后来添加的标签中。

//TODO //预留

//if (dataDicInfo.DicExtend == "1或2或3，等自定义标签" && dataDicInfo.LookupKey == "SomeTag，等自定义标签名字")

//{

//}

//之前的

// var entity =

//GetGroupModel(new CustomerCircleGroupDetailParam()

//{

// CustomerGuid = requestCustomerGuid,

// GroupId = groupId

//});

// string groupName = entity.GroupName;

//if (groupName=="老师")

//{

// //在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

// //查询 学生 所在分组的GroupId

// var responseGroupId = GetGroupModel(responseCustomerGuid, "学生").Id;

// //插入该条数据

// AddCustomerCircleGroupMember(responseGroupId,requestCustomerGuid);

// //并把数据加到同行里面

// responseGroupId = GetGroupModel(responseCustomerGuid, "同行").Id;

// AddCustomerCircleGroupMember(responseGroupId,requestCustomerGuid);

// //同时加到请求方的同行里面

// var requestGroupId = GetGroupModel(requestCustomerGuid, "同行").Id;

// AddCustomerCircleGroupMember(requestGroupId, responseCustomerGuid);

//}

//if (groupName == "学生")

//{

// //在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

// //查询 学生 所在分组的GroupId

// var responseGroupId = GetGroupModel(responseCustomerGuid, "老师").Id;

// //插入该条数据

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

// //并把数据加到同行里面

// responseGroupId = GetGroupModel(responseCustomerGuid, "同行").Id;

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

// //同时加到请求方的同行里面

// var requestGroupId = GetGroupModel(requestCustomerGuid, "同行").Id;

// AddCustomerCircleGroupMember(requestGroupId, responseCustomerGuid);

//}

//if (groupName == "同行")

//{

// //在本用户，也就是responseCustomerGuid中增加“学生”，对应该用户

// //查询 学生 所在分组的GroupId

// var responseGroupId = GetGroupModel(responseCustomerGuid, "同行").Id;

// //插入该条数据

// AddCustomerCircleGroupMember(responseGroupId, requestCustomerGuid);

//}

}

#endregion

27.

28.

29.