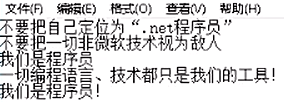
web api

是

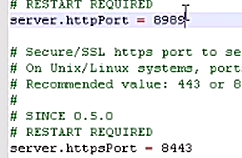
Git:

05 Git - Git简介及github初始化,搭建自己的Git服务器,配置Git本地工具,Git日常开发操作:

老师说，公司人多，不想买GitHub的帐号，然后就自己搭一个服务器，我在想，帐号不是随便注册的吗，还是其他意思？那种月付的帐号？



https需要买证书的。



添加用户，看有多少个成员，给不同成员的不同的权限。

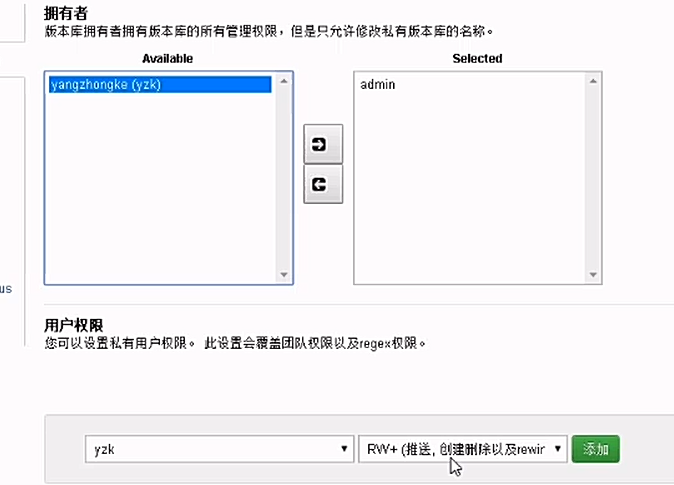
使用gitblit创建自己的服务器。



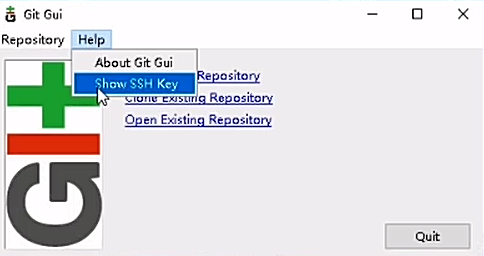
创建版本库：

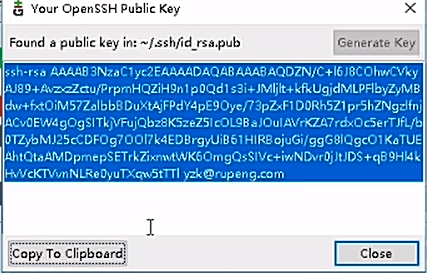


给用户添加权限：



SSH 公钥





将代码提交到服务器：

无论是新建一个存储库，将新建项目提交到服务器；还是下载已有的存储库，后续的操  
作都是一样的。新建存储库也需要先建服务器端存储库，再下载到本地（初始文件），再上  
传。

04. WEB API-1-什么是WebAPI,什么是Restful,简单的WEBAPI,WebAPI参数,返回值,通过自定义路由处理多版本

05. WEB API-2-01实现多版本,自定义filter

05. WEB API-2-02异常处理,安全性,JWT,传输安全保障

06. Web API-3-01-UserCenter,自定义Filter校验Sign,实现接口的AppSecret及Sign验证,封装OpenAPI的SDK

06. Web API-3-02 .net core下的web api

Web.API:

两个不同的系统进行通信，也可以走http协议。比方我们公司调的别的公司的接口，用的也是web API。

连邮箱发送，短信发送等平台的接口，都是http结构。

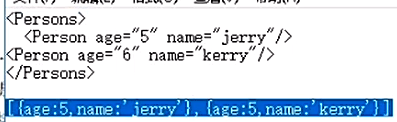
还有手机上装的app，和服务器间通信，大部分走的都是http协议。（我目前做的项目就是这样）

用以前的一般处理程序ashx或者mvc，都可以实现系统间的接口，或者说app和服务器间的接口。

而用Web API的原因是因为它更方便。

wcf，其他语言想访问我们写的服务器，比较麻烦，而WebService渐渐退出历史舞台，因为用xml，传输量太大。

json和xml的简洁程度对比：



因为app如果太耗流量，也许就被用户给卸载掉了。

以后在公司自然会当家作主，所有选的技术一定要是sb都会用的技术。



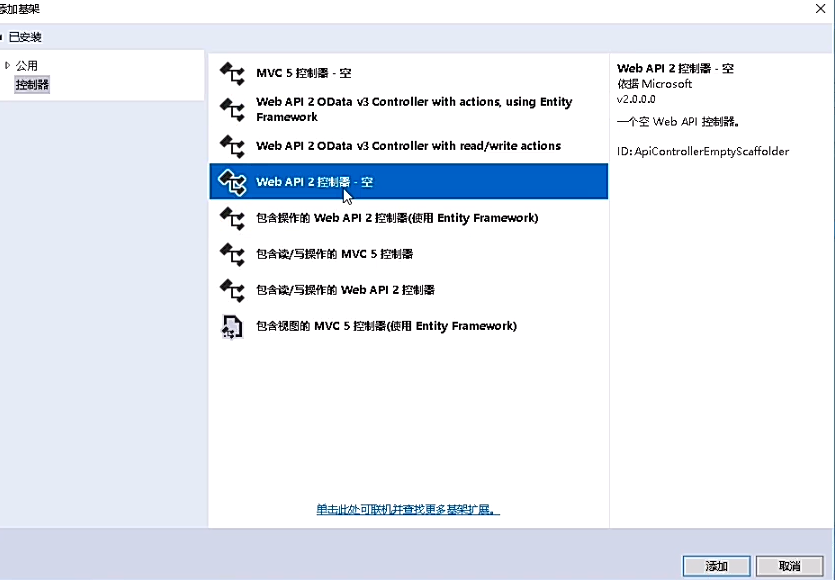
返回体，现在也是有code，msg，还有泛型T data属性。

还有充分利用http状态码





新建Web API项目：



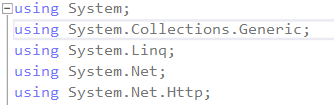
这个类默认继承自：ApiController

public class PersonController : ApiController

{

}

老师把没用的都给删掉了。



我自己也测试：

新建了三个方法：

public class PersonController : ApiController

{

public string[] Get()

{

return new[] {"ssss", "aaa", "gggg"};

}

public string Get(int id)

{

return "花花->" + id;

}

public string Get(string str)

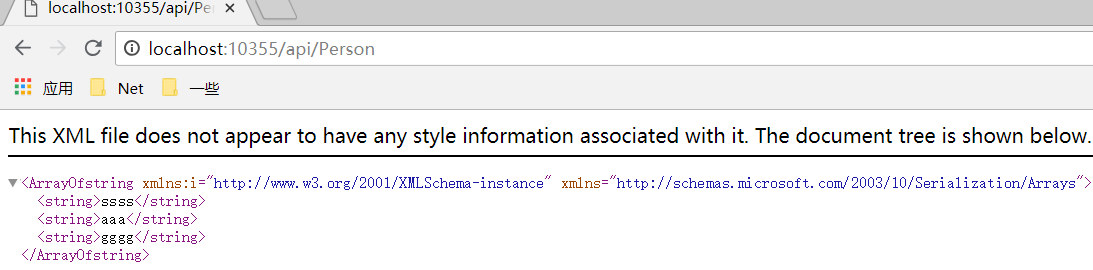
{

return "木森->" + str;

}

}

用浏览器访问：



感觉是xml文件啊。

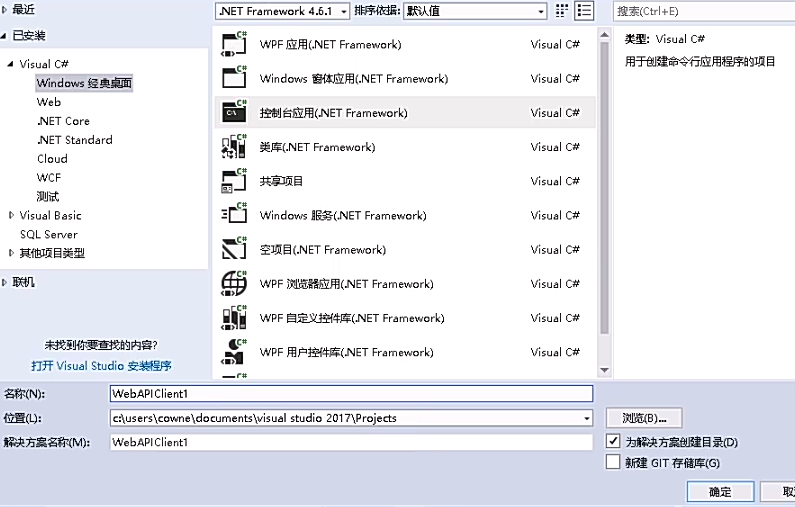


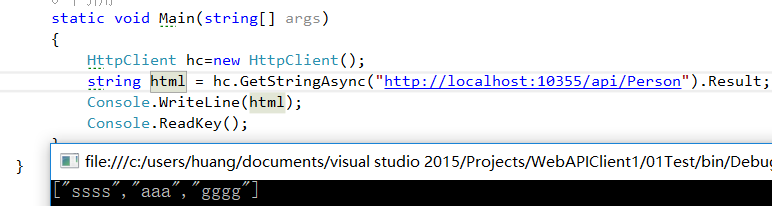


多个参数就错误了。



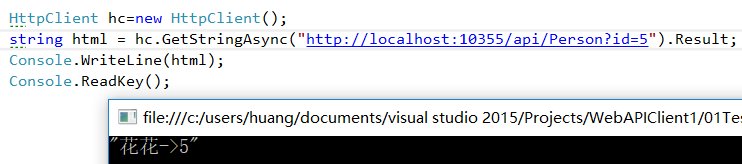
另开一个vs，用控制台去访问。

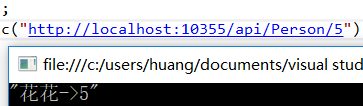




这里没有用异步，而直接返回了结果。await

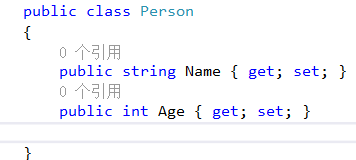
其他的结果也都是：





也可以返回自定义类的对象。

新建了Person类。



方法：

public Person Get(int id)

{

return new Person() {Name="花花",Age=id};

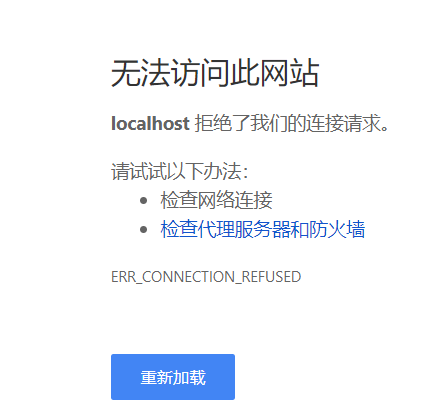
}

结果：

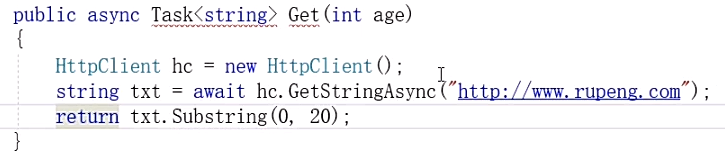


发现客户端根本不需要引用Person类，因为所有的数据都拆成了json格式。

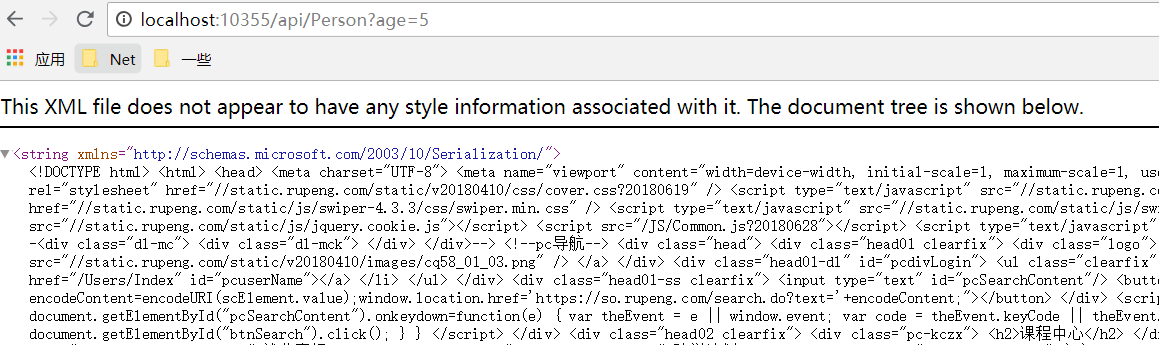
Web API那边一停止调试，就访问不到了。



异步调用：



用异步，调用rupeng网，一大堆返回的xml



设置为本地后：

public async Task<string> GetByAge(int age)

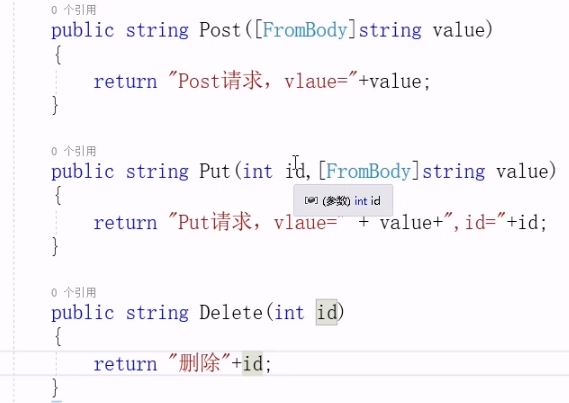
{

HttpClient hc = new HttpClient();

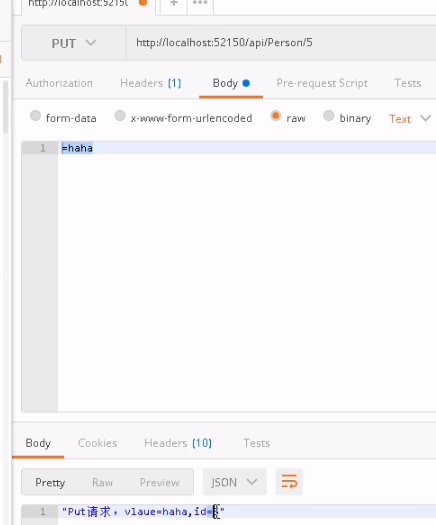
string resp = await hc.GetStringAsync("http://localhost:10355/api/Person");

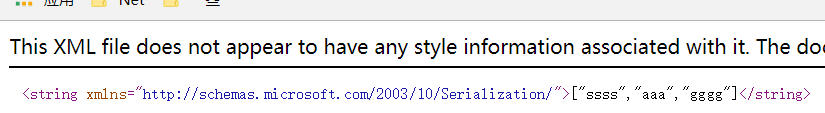
return resp;

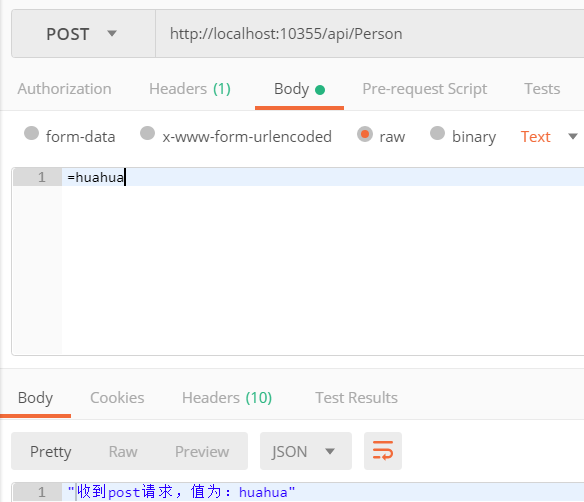
}

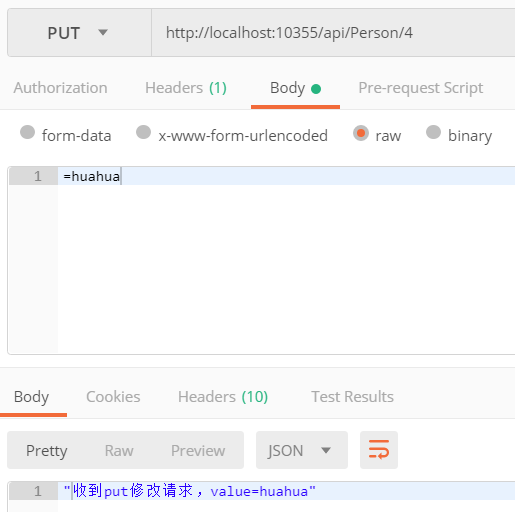


post请求体中，直接=huahua，就可以了，而如果是put，后面加个id





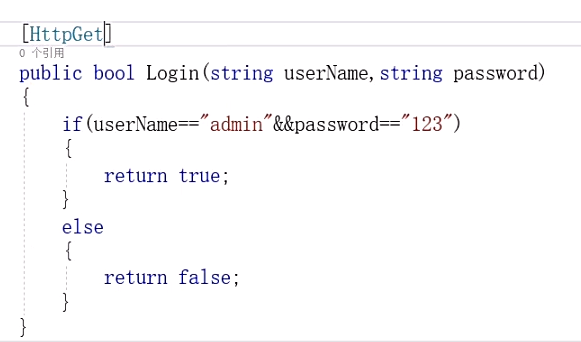




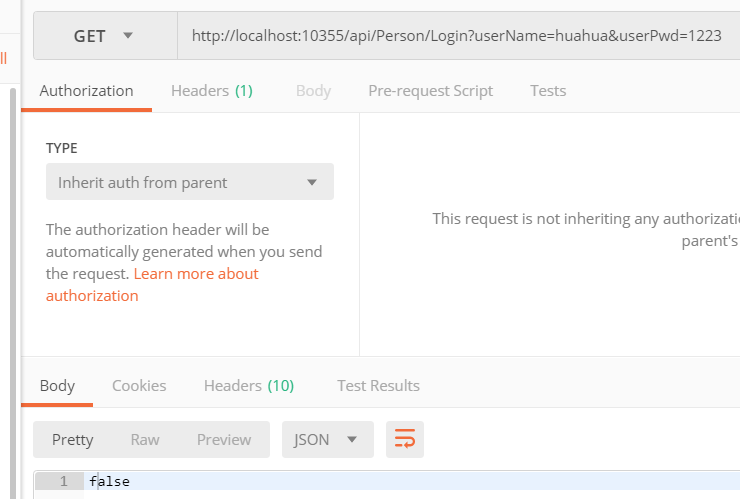
也可以修改路由规则：



如果是get请求，要标上[HttpGet]

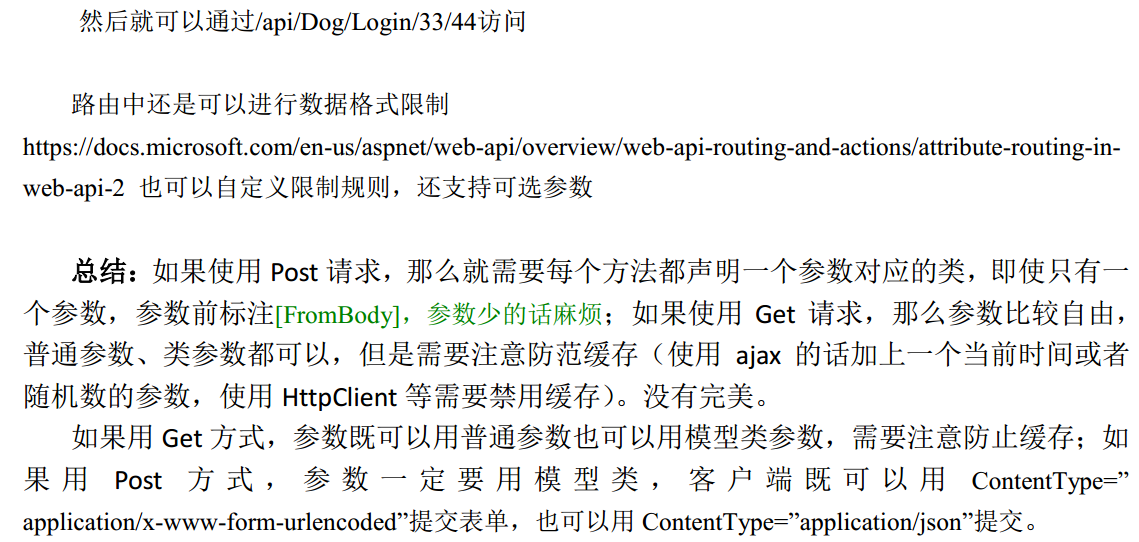


结果：









如果用模型对象，则必须标注[FromUri]。也是和上面的效果一样。（查询的时候，条件很多，就可以用这个）

新建了一个类：

public class LoginModel

{

public string UserName { get; set; }

public string UserPwd { get; set; }

}

Action方法：

[HttpGet]

public bool Login2([FromUri] LoginModel model)

{

if (model.UserName=="huahua"&&model.UserPwd=="123")

{

return true;

}

else

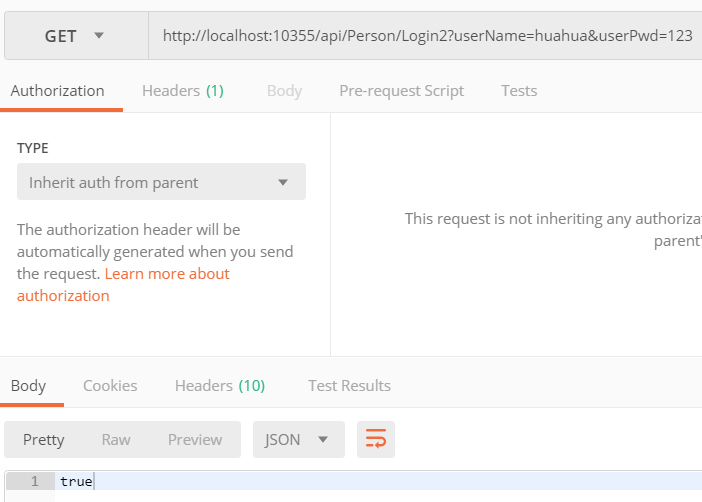
{

return false;

}

}

用的同样的请求：





Post请求：

[HttpPost]

public bool Login3([FromBody] LoginModel model)

{

if (model.UserName == "huahua" && model.UserPwd == "123")

{

return true;

}

else

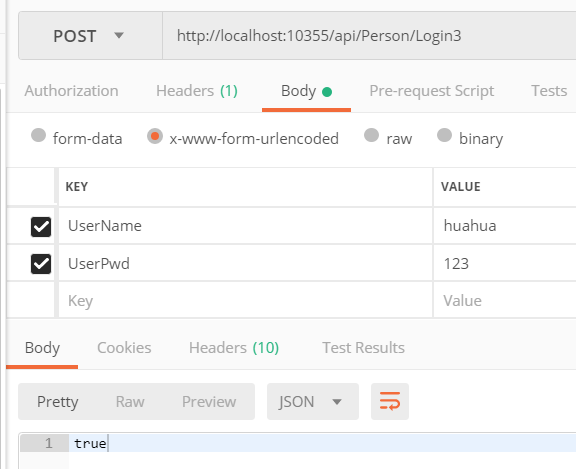
{

return false;

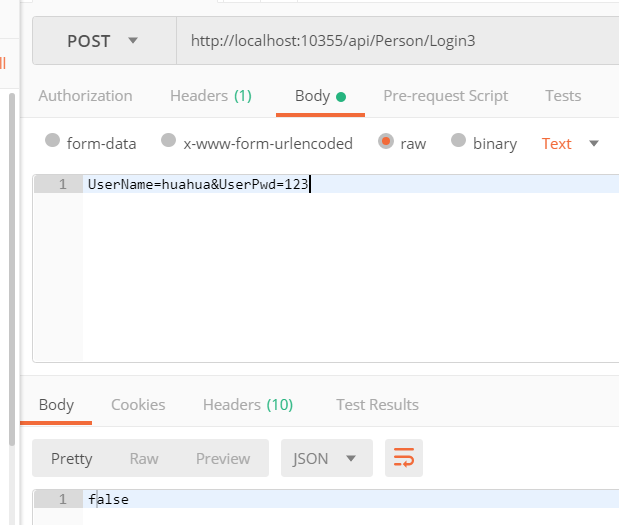
}

}

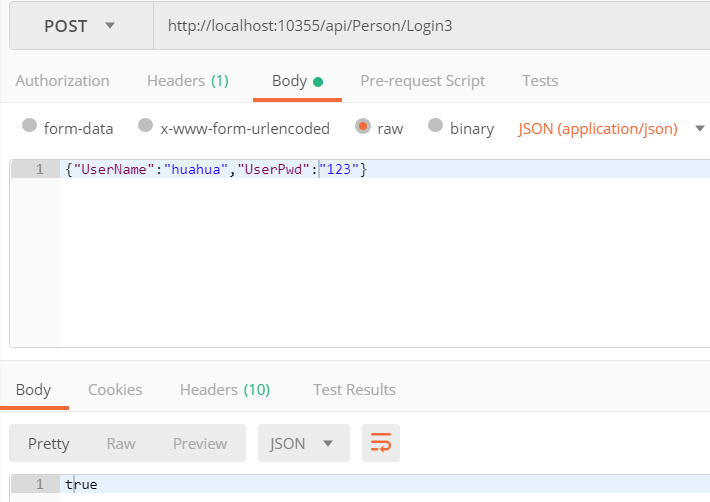
请求结果：



用Raw也可以：



如果传的是json类型的，就要将数据表示为json格式：





我想，这就是就近原则吧，和标签，特性一样。

测试：

public class HuaHuaController : ApiController

{

[Route("GetGua")]

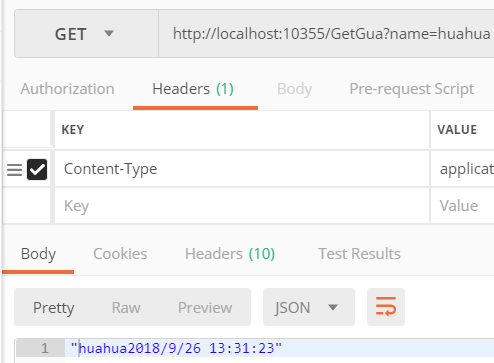
public string Get(string name)

{

return name + DateTime.Now.ToString();

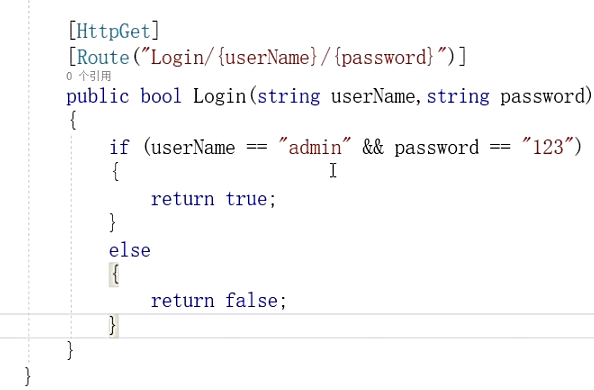
}

}

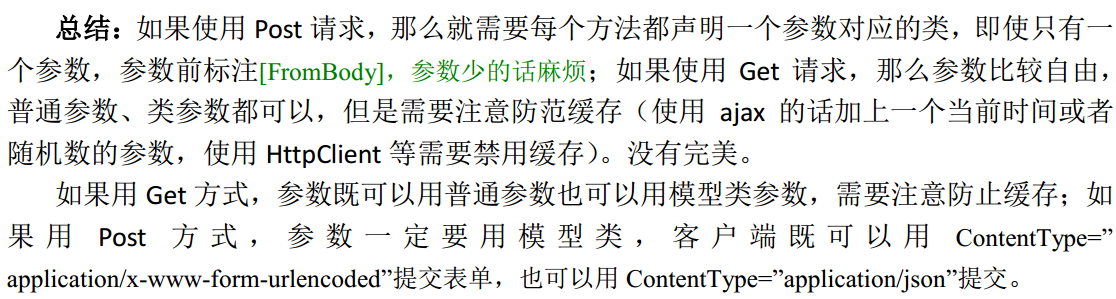




从url捕获参数(无论get、 post等都可以)：



总结：老师推荐用get，如果要用post，就按他上课讲的做。



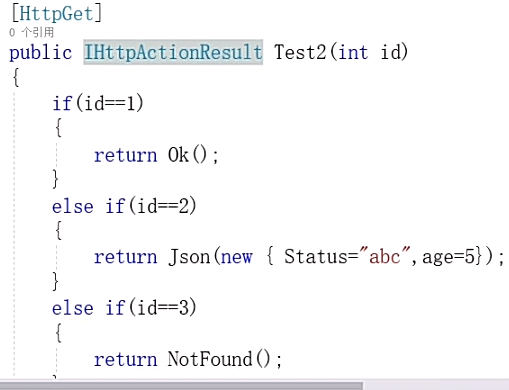
7.Web API返回值类型：

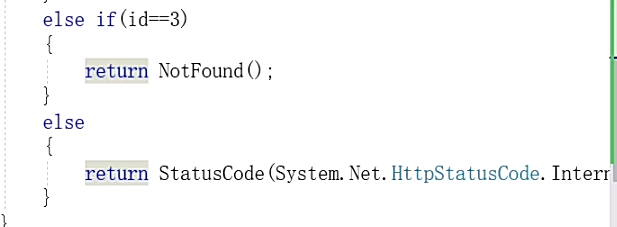


返回值为空的情况下，返回状态为204：

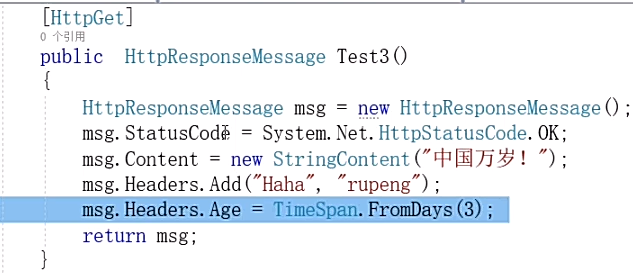


Action 的返回值可以为 IHttpActionResult 类型， 可以调用 ApiController 中的 Ok()、  
NotFound()、 Json()、 Content()、 Redirect()等方法。 如果需要用 Http 状态码来表示处理结果，那么可以用这种方法。





HttpResponseMessage 可以做更精细化的返回内容控制， 比如返回二进制文件、 设置返回报文头



再演示使用 StreamContent 返回文件（ 接口尽量不要直接返回二进制流）。

不推荐直接返回文件，因为流量太大，会卡。一般都是返回一个路径。客户端程序再根据路径去下载。

演示：

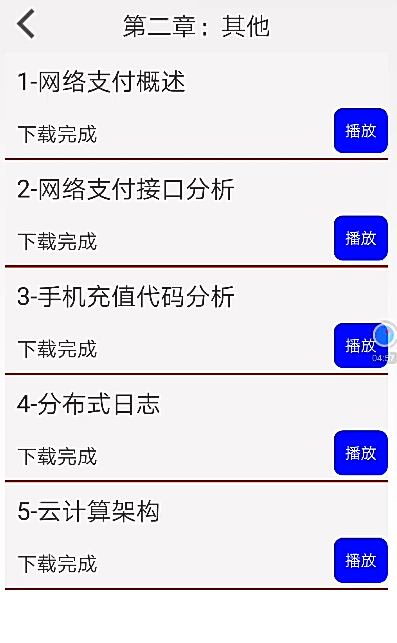


8.Web API多版本管理。

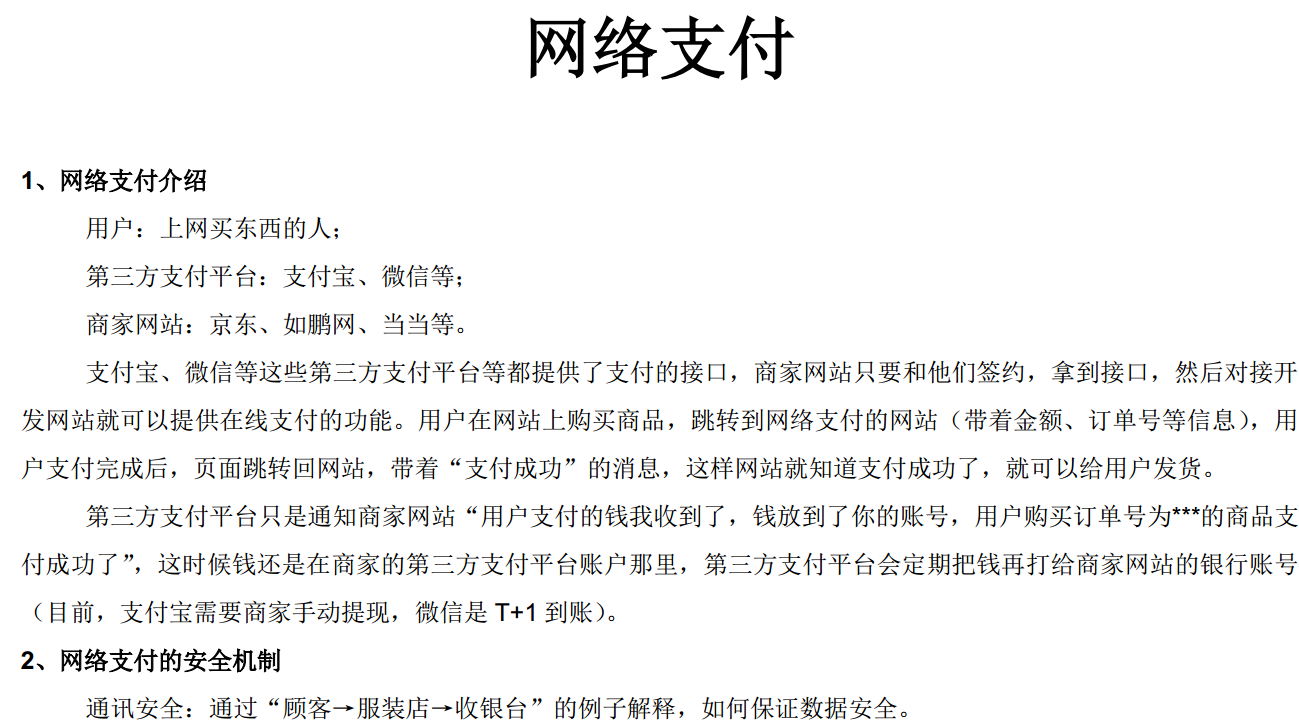




网络支付：



1.网络支付概述



比方对方在生鲜网上买东西，下单。前提我们已经和支付宝签约，获得了接口。然后用户下单，点击支付，跳到支付宝页面，我们把信息，单号啊，价格啊，这些发给支付宝了，支付宝审核，用户结算，支付宝给我们回调的网站发信息，说这个订单号的订单已经付款，支付宝已经已经收到钱了。然后我们发货。体现的话，就去我们在支付宝开通的账户里体现。

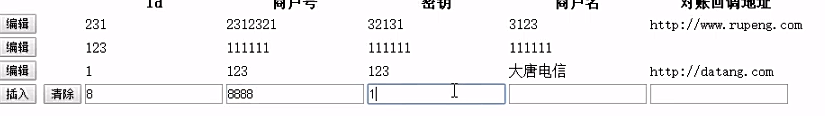
类似商场买衣服，各个品牌的服务员给我们开好小票，比方大衣十斤，我们去收银台排队，付钱，拿了盖了章的小票回去给服务员看，check了，我们拿货。而我们付的钱，商场也会打到商户的卡中，也可能统一结算。然后商场收服务费，支付宝也收我们的服务费，比方花呗等等。

漏洞就是比方我们伪造了商场的章，盖个章，那就拿到货了。其实没交钱。可是商场有监控啊，要去警局了呀。

相比而言，网上就没有监控了，所以各种漏洞都要给填了。

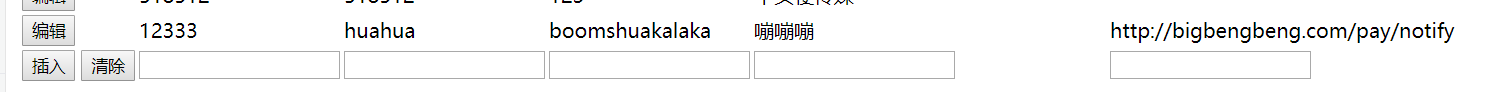
2.网络支付接口分析：

签约了，支付宝工作人员给我们开一个账户。

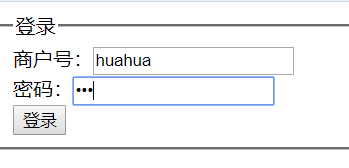


密钥就相当于我们的secretkey。对数据进行加密用的。

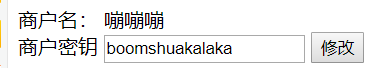
我自己测试，再如鹏网上加了条信息。

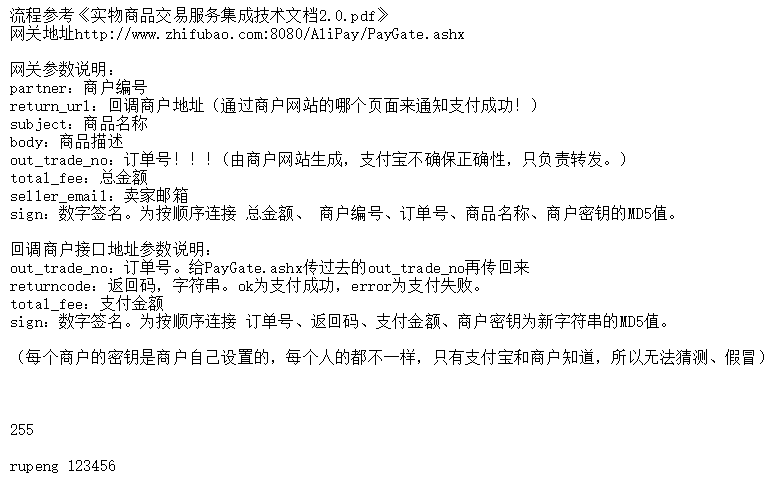


商户登陆：



可以修改密钥。



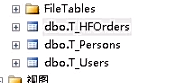


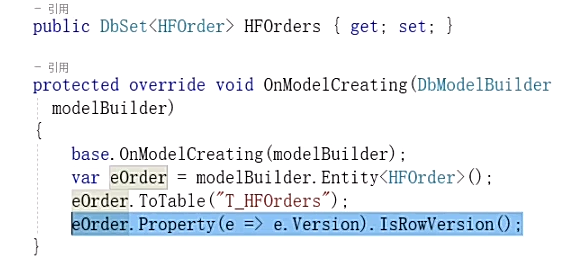
md5加密：

sign：数字签名。为按顺序连接 订单号、返回码、支付金额、商户密钥为新字符串的MD5值。

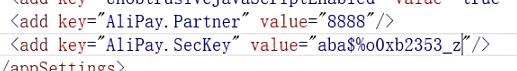
两边都进行md5加密，看结果是否一样。如果不一样，数据就被篡改过了。

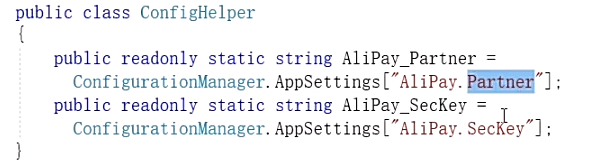
老师做的手机充值demo，然而没有源码。



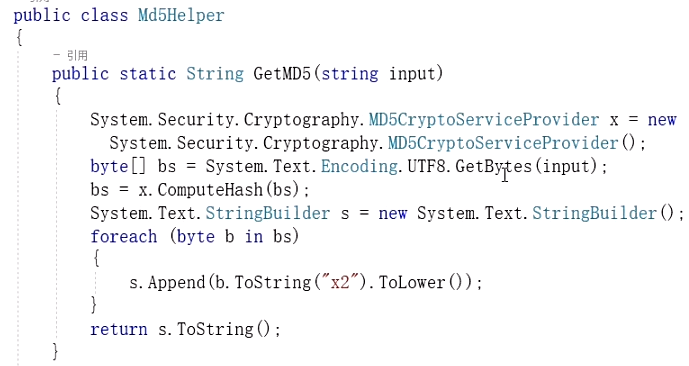


读取商户编号和密钥。



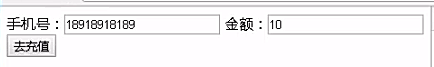


md5算法：



比较麻烦的，比方我这边和支付宝那边有一个字母大小写不一样，一直出错。支付宝那边是不忽略大小写的。

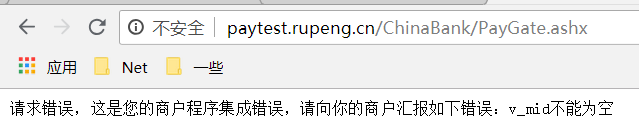
开始测试：



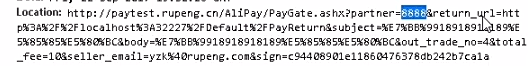
一点充值，跳去支付宝页面充值。这里用如鹏模拟。



我自己测试了下，因为东西都没有传过去，所以错误。

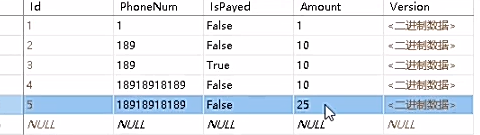


我们请求的时候就把所有的订单，商品，金额信息给发过去了。



这里要登陆支付宝账号密码，不过一般用的勤，都是默认登陆的，也就直接支付。

同时我们数据库中也有了这个订单，状态为未支付。

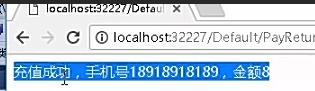


当那边已支付的信息传过来，再把状态改为支付。

支付成功。



跳转商户页面。



支付宝把订单号，商品，价格，这些信息回传给商户。



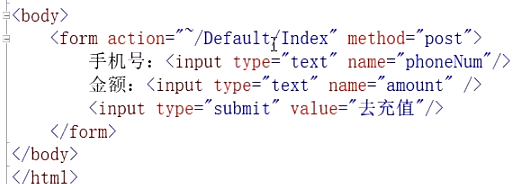
我们网站本身也要对这些信息进行防篡改的校验，校验成功后比方才发货，才正式给用户充值，该订单状态也true，也就是已支付。

老师的实现。

创建订单：

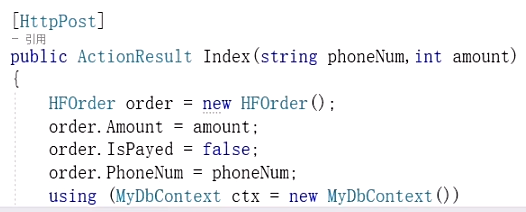


view前台：



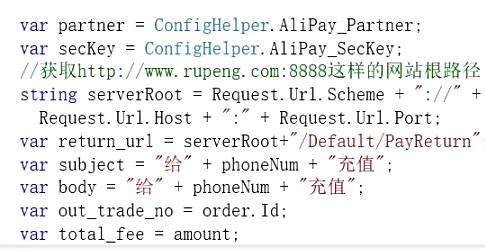
提交之后的处理:

创建订单，插入到数据库，

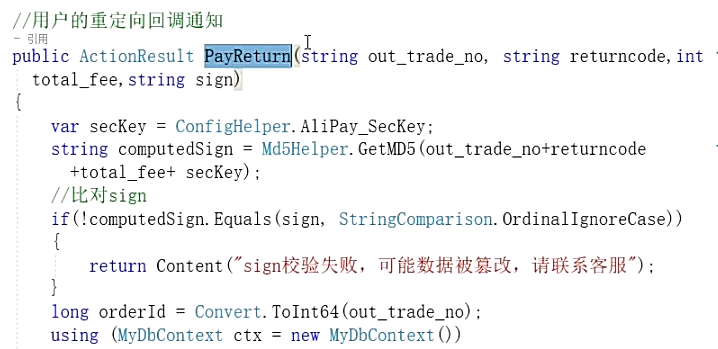


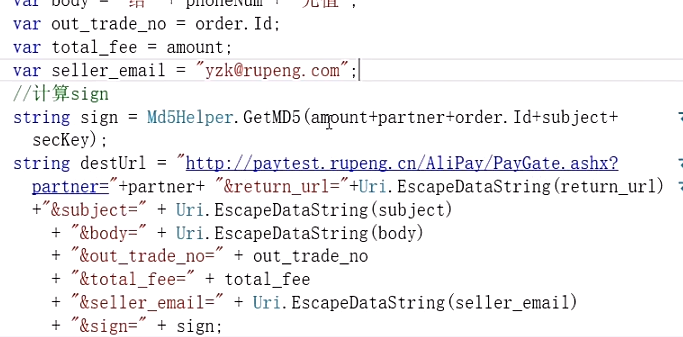


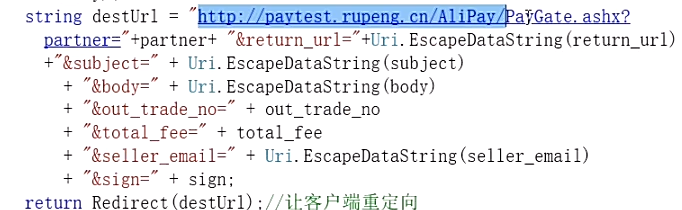
读取支付宝发过来的订单号，商户编号，以及只有我们双方才知道的secretKey。



用户的重定向回调通知：





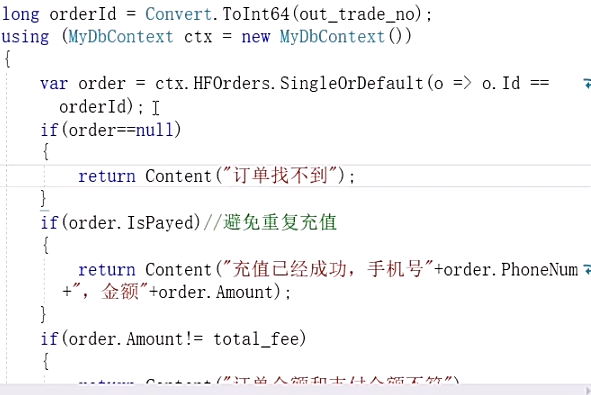


用户收到这个通知后重定向到支付宝的支付页面进行支付。

用户完成支付后，支付宝会让用户重定向到商户这边。return\_url我们已经发给支付宝了。



接下来去数据库中查询订单编号。检查订单是否已经确认过了，有就结束，（不然在充值确认页面，用户一直刷新，每刷新一次就充值一次了。老师表示有学生去游戏网站就没有注意这个点，被用户这样刷走了很多钱）没有就充值，将状态改为已充值。我在想，如果我们又是第三方的，那这里要调用移动的api，然后给用户重置，因为我们在移动那边已经有账户，里面有钱的，扣去，我们赚个差价，这样子。薄利多销。



还有种情况，用户支付完立马关了网页，东西还没有回传到我们网站。这时候用户来骂，钱付了，东西怎么还没到。

这里支付宝有多次对账的机制，会多次在后台向我们的回调网站发送已支付的信息。所以这个环节是支付宝和商家的信息传递，即便用户关闭了，也不要紧。

你只要支付了，就没你什么事了，接下来就是支付宝和商家的事了。你只要坐等充值到账就可以了。

如果多次还不行，就要发消息给工作人员了，工作人员打电话和支付宝确认，确认完毕，手动将后台的支付状态改为true。

3分布式日志：



比方有五百台服务器，把所有的日志集中到一台日志服务器上进行处理，查询，等等。当然，如果日志量大，就要多台日志服务器了。

自己做，用MongoDb也是可以的，因为支持全文检索。

但如果要报表显示，昨天多少，今天多少，这样。就比较麻烦了。

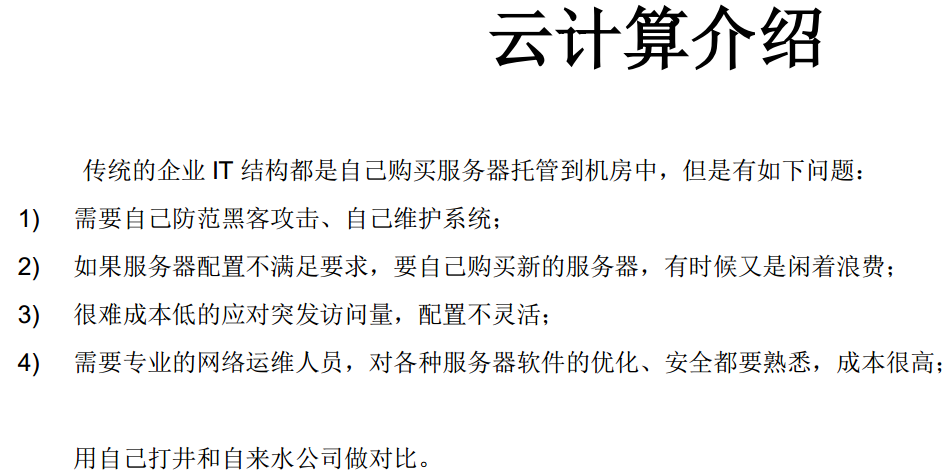
ExceptionLess有他的服务器，我们可以创建账户，直接往他的服务器上扔。

当然，他这个服务是收费的。

但是很良心啊，提供客户端的下载，这样我们下载了，配置在我们自己的服务器上，就没有任何的限制了。

具体的配置什么的，暂时跳过。TODO：mark一下！

5.云计算架构：



自己维护起来太麻烦，成本也高，直接花钱用别人成熟的服务，简单粗暴。

比方明天有讲座，要很大访问量，或者类似查询中签的网站，短期会有很大访问量，但平时基本没什么访问量，所以为了这个而多买带宽，还是一年，太浪费了，还不如去阿里云买，直接调，随叫随到，来得方便。

比方明天下午三点到八点，给我100m带宽，平时给1m就可以了，用阿里云很方便。

老师说起如鹏，比方那三天上课，学生超多，那调一下带宽就可以了，平时不上课，不用的。

因为是在线看视频，在线上课么。带宽需求量肯定是大的。

老师买的，两个如鹏的机房，配置在华北地区。



创建实例去购买：

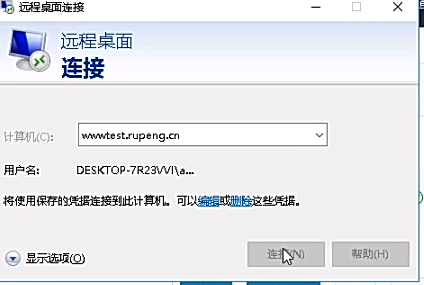


选好配置后，付钱就可以了。



比方几分钟后，就收到邮件，说我们的服务器已经创建成功。

这个服务器，我们是可以通过远程桌面去连接的。



临时需求，加配置就可以了。

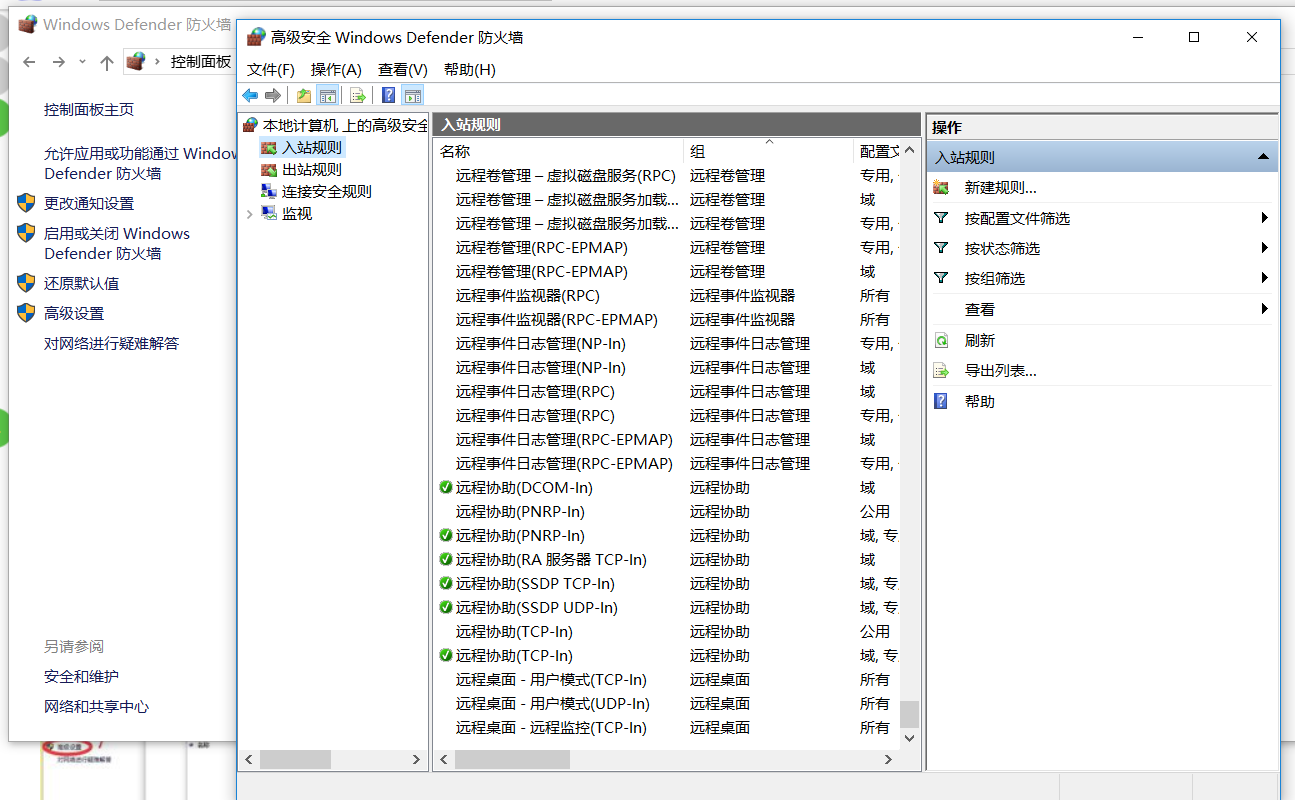


几分钟的事情，操作系统就变成4核16g了。

如果用了微软的Azure，会发现比阿里云强太多。

平时我们要用8888端口，只要再防火墙上把8888端口给打开就行了。

顺便按操作调了下防火墙，比方打开防火墙的某个端口。



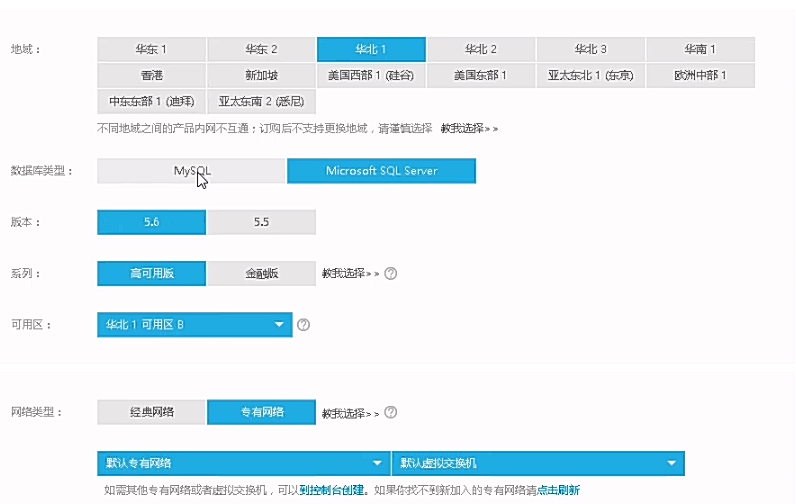
比方8888端口，在防火墙这里已经打开了。还要在阿里云的安全组中进行配置。



需要两个地方同时开放了8888端口，才行。

RDS。云数据库。

老师说不推荐我们自己买数据库，安装，去维护。比较推荐用阿里云的云数据库。



！！！注意点，服务器机房和数据库机房一定要在一个机房，否则跨机房调用，效率特别地低！！！！

比方服务器在华东2，那数据库也在华东2，不要选择其他的。

购买后就有了数据库了。



用他的，他会定期备份，也不怕我们一不小心把数据给删了。

就之前同学有把公司数据库删了的，打电话给老师，老师让看下是不是阿里云的，是，有备份没，有的，立马还原。

另外还可以搭建集群的。全部用他的就可以了。

需要搜索，也有现成的，直接买就是了。当然，也得跟服务器在一个机房。



"uk":2409958245

shareid":509546340

http://pan.baidu.com/share/link?shareid=509546340&uk=2409958245

尝试着看别人的分享，没办法看，没有权限的。

想到，这里的权限设计，只有本人能看。

