

# 如何开发一个简单的智能对话查询工具

本文将通过微信小程序介绍如何制作一个智能对话APP。当然，你可以下载源码，修改为其他类型的应用程序，比如Android,MFC,Phthoy,ios等等。

我们经常在电影中看到机器人对答如流，随着越来越多自然语言开放平台的出现，IT爱好者制作一个自己的APP或者小玩具等逐渐可以变为现实。

自然语言对话即应用程序能够对用户输入的语音或者文字做出准确的回应，就像他在和人说话一样。

比如，在微信公众号中，经常要求用户通过输入1、2或者其他关键字来获取相应的服务，而对于句子却无法正确理解。比如，你输入“中秋活动”，这个几个字如果符合关键字的要求，那就会弹出相应的服务。但如果你输入的是“我想参加今年的中秋活动”，“参加中秋活动”等可能就无法进入活动网页了。

再比如，很多智能玩具，你只能跟它进行简单的对话，因为它也是只能抓取简单的关键字。

实际上，我们希望的是做到像许多语音助手一样，能够正确识别用户说的话，并做出对应回复，甚至希望能够理解上下文。

比如，用户问：今天上海的天气

应用回答：今天上海的天气为。。。。

如果用户接着问：那北京呢

你的应用能直接回答：今天北京的天气。。。。

而不需要输入完整的句子“今天北京的天气”就可以得到正确的回复。


这里我选择了[欧拉蜜人工智能开放平台来完成我的快递小助手。

下面从功能讲起，告诉你如何完成智能对话工具的开发工作。

## 一、功能展示

如果希望直接体验程序的功能，请使用微信扫描或者长按下面的二维码，这样更直观。



 微信扫一扫，使用小程序

代码下载：[智能查询工具源码](#)

欧拉蜜语法文件下载：[身份证](#)、[快递](#)、[词典语法文件下载](#)

首页

# GitChat

点击首页中的任何选项和图片，都会进入相应的查询界面：



## 帮助页面

帮助页面提供各种功能的简单介绍和自然语言理解、技术交流的联系方式。



## 子页面

每个子页面里都提供例句和切换例句的功能，可以先点击例句试试看。每个子页面也都有输出结果显示，如果内容过多，需要触摸滚动显示。

< 返回

欧拉蜜快递查询



▶ 3333481970033

◀ 运营商--申通快递 运单号--  
3333481970033

→【签收】已签收, 签收人是:【邮件签收章】

...2017-07-08 21:59:53

→【派件】【上海南汇公司】的【王正阳手机(13482647336)】正在派件, 扫描员是【南汇周浦分公司】

...2017-07-08 13:37:16

→【到件】快件到达【上海南汇公司】, 上一站是【】, 扫描员是【南汇周浦分公司】

请输入查询语句

查询

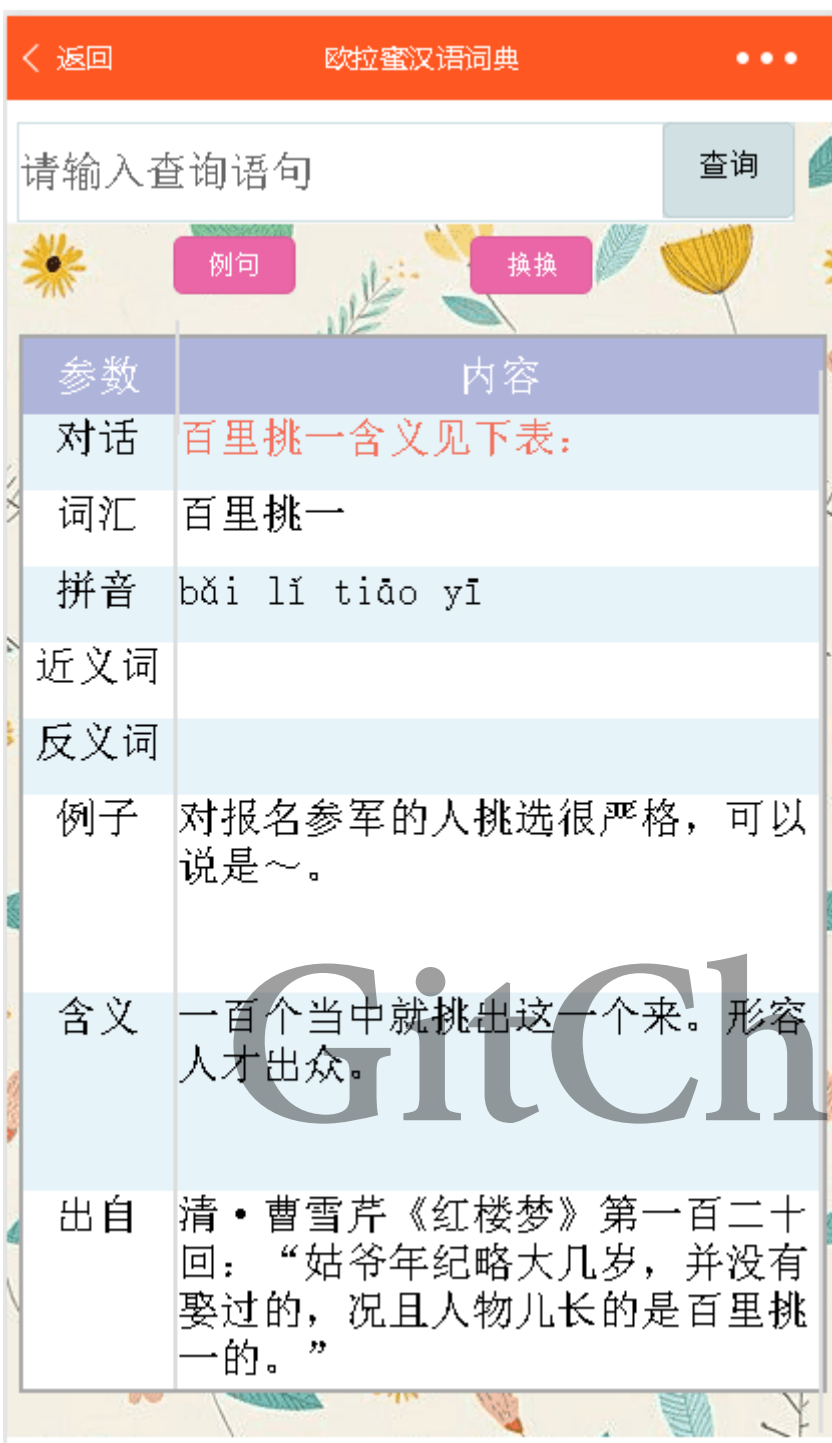
例句

换换



## 词典查询

词库大约有50万数据，支持近义词、反义词、出处、含义单独查询。



## 身份证查询

身份证前6位仅能查到行政位置信息，身份证号码可以查到除了姓名之外的信息。

< 返回

欧拉蜜身份证查询

...

参数	内容
结果	
省	
市	河南
县	焦作
性别	
出生年月	
地址	河南省 焦作市 孟县

： 410826

请输入查询语句

查询

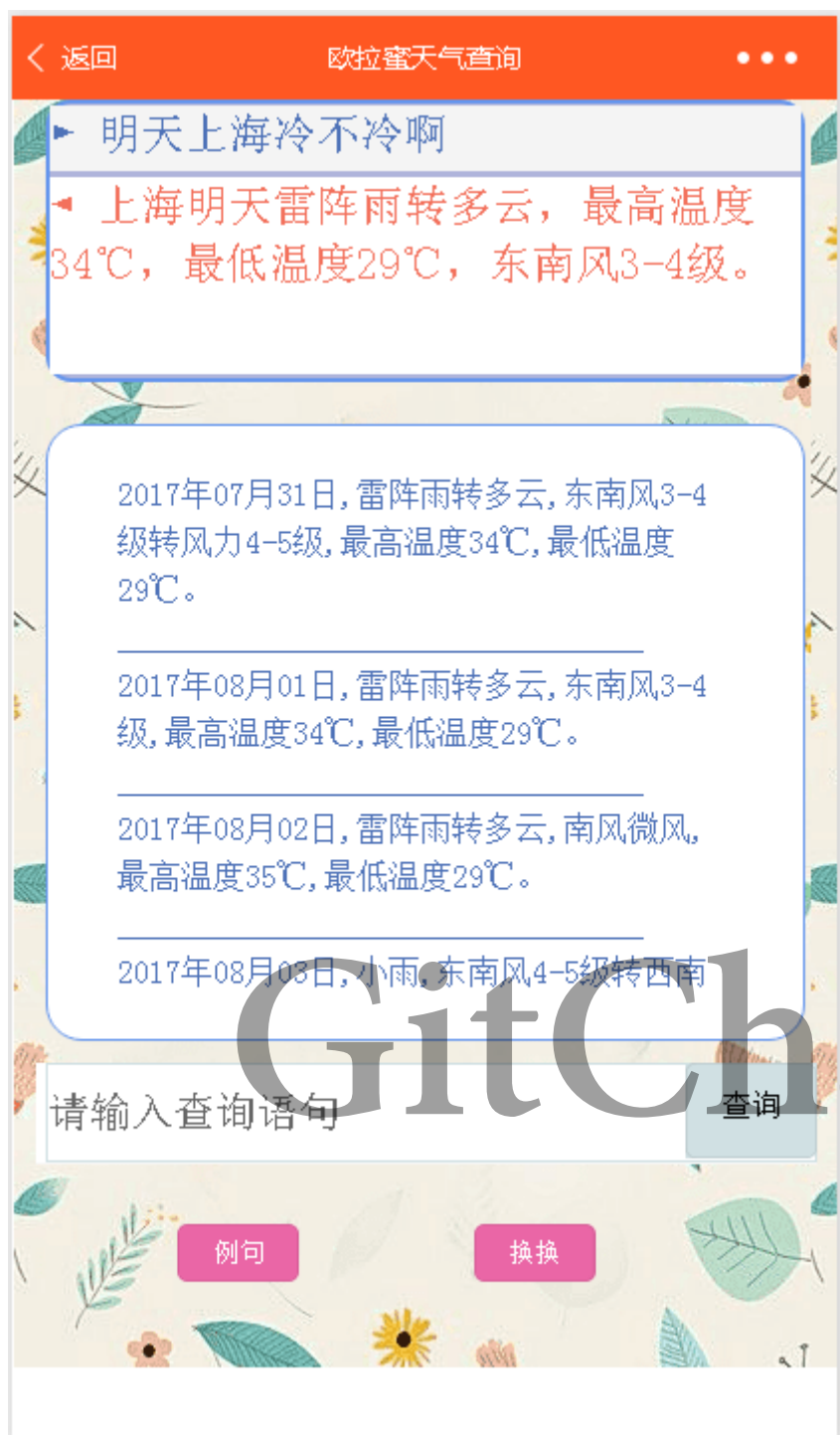
例句

换换

## 天气查询

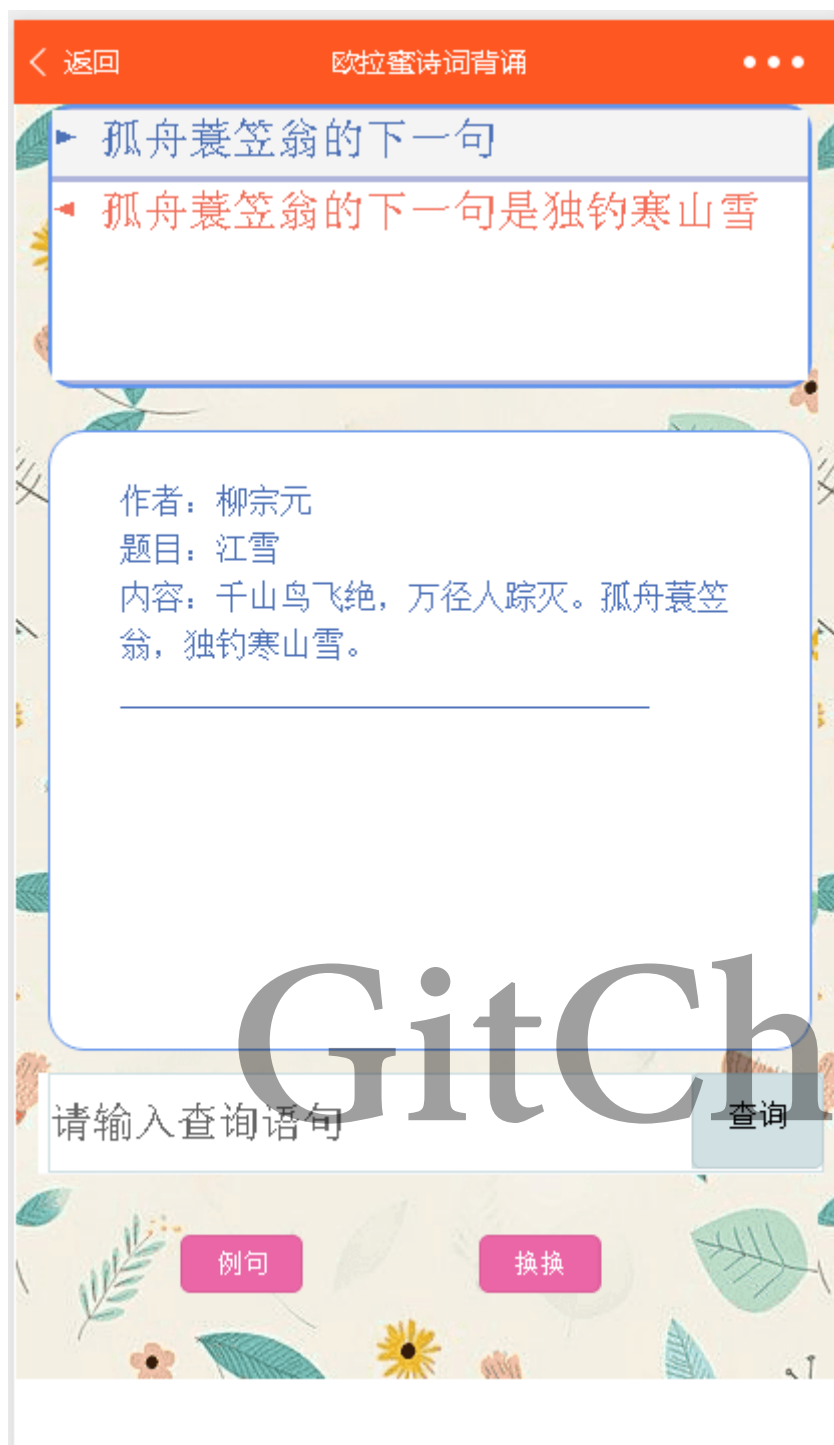
天气支持今天前后五天的查询，也支持温度、风力、风向、指数的查询，口语化做的比较好，比如“明天冷不冷”，“明天上海会下雨吗”。





## 诗歌背诵

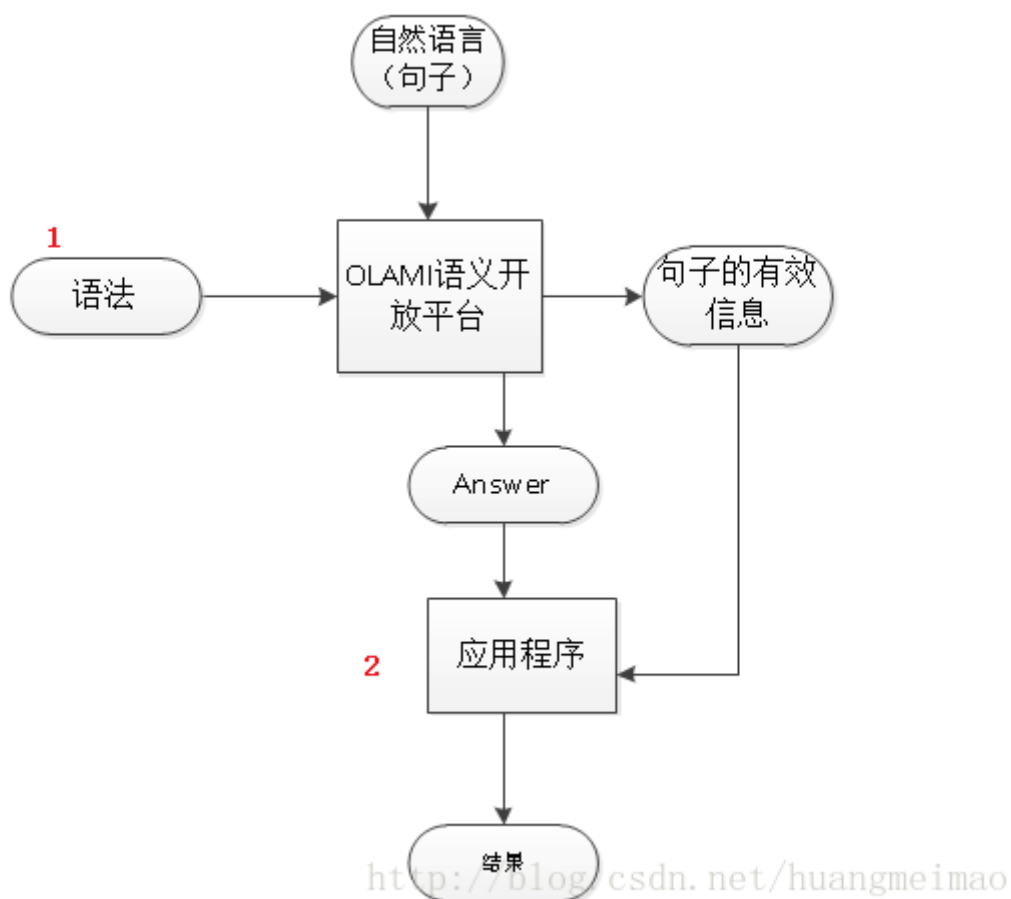
诗歌支持诗歌名称查询，另外支持作者作品、诗词上下句、诗词出处等的查询。



其他功能不再一一展示，大家可以扫描微信小程序体验或者直接下载代码测试。

## 二、智能查询工具工作流程

下图简单描述了OLAMI语言理解开放平台的工作过程，我们要做的工作其实就是图中红色的标志1和标志2，即写语法和写自己的应用程序。而你的应用程序用户只需要输入想说的话，即自然语言句子，就可以得到你的应用程序处理的结果。



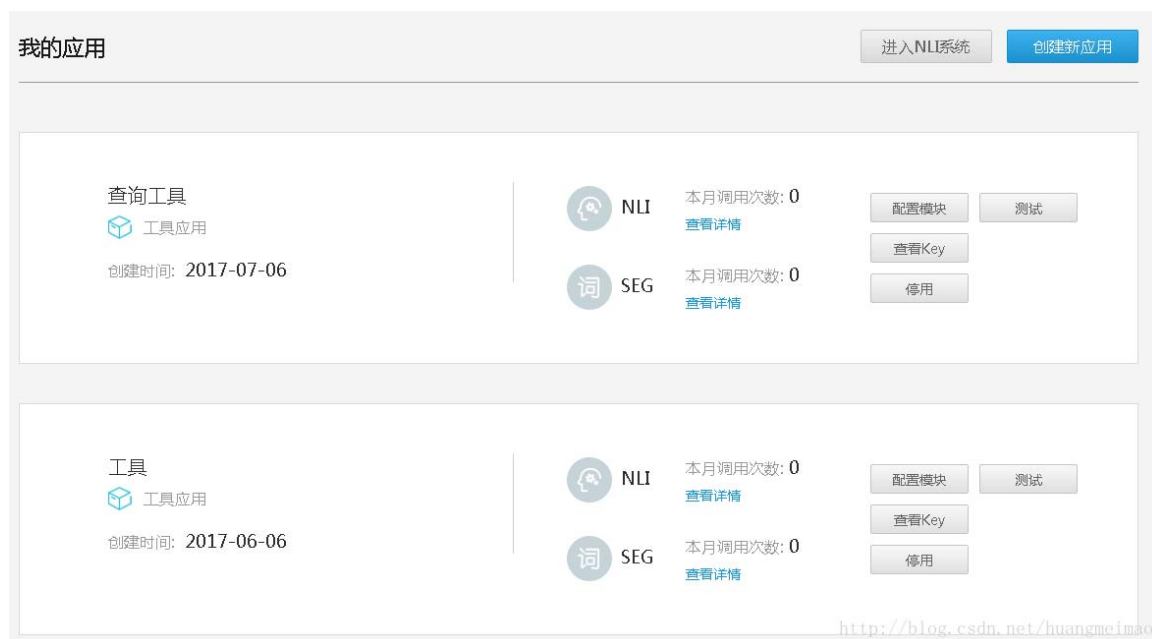
### 三、了解几个关键词的含义

1. 语法，我的理解就是描述自然语言句子的一套规则，看官方文档也确实是这样。比如：“帮我查个快递”，你用一堆符号描述出来，这个组合起来的符号就叫语法。
2. 自然语言，也叫语料。这个不用多说，我们平时说的话是什么，那就是什么。
3. 语料的有效信息，即输入的句子中包含的关键信息，你的应用程序获取到这些这些关键信息之后，可以做相应的处理。比如：‘请你帮我查个快递’，其实这句话隐藏的含义是“查快递”，其他的词汇信息不需要关心。
4. Answer:这个在OLAMI平台中指你写的语法的默认回复，不需要应用程序做深度处理的，这个暂时可以忽略。
5. 应用程序：这就是你自己的APP了。
6. 结果：就是用户输入一句话，你的APP回复结果。比如，用户输入‘查快递1234’，你应该回复‘1234这个快递的物流信息’。再比如“今天的天气”，你应该列出今天的天气状况，而不是昨天的天气，也不是今天的日期。

### 四、写语法

调用API接口之前，首先要写自己的语法，也就是你要支持哪些句子，当然你也可以直接使用OLAMI平台提供的内置语法，如果它的语法符合你的要求的话。在这里，我说说怎么写自己的语法。

- 首先在官网上注册，然后进入开发系统中“我的应用”。



- 确定自己的模块

在正式写语法之前，首先得弄明白怎么写语法，写什么样的语法。

第一步得选择你的应用程序支持的模块，比如是查天气、播放音乐、还是智能家控制。

这个模块，其实可以称之为领域，因为你不可能把人类所有的语言都涵盖，你的APP一般都是针对某个或者某几个领域。我选择的是快递APP。所以我要支持的都是快递的说法。

确定好模块之后，你要思考一下你的应用程序希望从用户的说法中得到哪些有用信息，以及用户会有哪些操作。比如快递查询，我需要有“运单号”，“快递公司名称”这两个有效信息，另外，我的应用程序仅提供快递的查询业务。

我大致知道如果用户说“帮我查一下圆通快递12344”的时候，包含的有效信息很全，我可以直接输出物流信息给他。但如果用户仅仅说“我想查快递”，这时他仅表达了想查的意愿，我的应用程序应该提示用户输入快递单号。

在OLAMI的语法中，使用“**Slot**”来抓取有效信息，它就像一个函数的参数，它的内容由用户决定。比如运单号和快递公司名称，每个用户的内容都是不同的。

因此进入OLAMI的NLI系统之后，我首先“**新增**”一个模块，名字为“**expressage**”，然后进入这个模块开始写语法。



<http://blog.csdn.net/huangmeimao>

- 确定有效信息Slot

就如上面所述，我需要“运单号”和“快递公司名称”这两个关键信息，因此，我定义了“expnumber”和“expname”这两个Slot。



我选择的类型均为'ext',因为用户有可能会直接输入运单号，而运单号的格式无法确定，所以我选择ext来抓取。

快递公司的名称也是有限的，其实选择internal格式的就可以了，但是我选择后面通过ext赋值的方式给slot赋值，这样有利于语法维护。（很拗口是吧？可以暂时不管）

- 确认APP的功能

你的APP准备提供什么功能呢？查询？显示？打开？关闭？OLAMI语法需要用Modifier来描述操作信息。其实也就是一个标志来告诉应用程序这句话的意图是什么。

比如“打开灯”，意图是打开，可以定义一个modifier “open”。

“打开空调”的意图同样是打开，你只要用沿用已经定义的“open”即可。

我的快递APP很简单，我就是提供查询功能，因此我定义一个global modifier ‘query’。

- 了解Grammar,Rule,Template.

其实刚开始只要知道Grammar和Rule就可以了。

Rule即同义词汇的集合，词汇之间用'|'隔开，表示或的关系。

Grammar即描述你要匹配的句子语法。

比如你希望匹配句子“查询快递”，可以将“查询”的同义词定义一个Rule，“快递”同义词汇定义一个Rule，我建议名称能使用中文就使用中文，这样看起来比较直观。下表中是可以匹配“查询快递”这句话的Grammar相关定义。

名称	类型	内容
查询_动词	Rule	查询 查一下 查查 查
快递_名词	Rule	快递 速递 物流 速运 快运
expname	Ext Slot	
expnumber	Ext Slot	
查快递1	Grammar	<查询_动词><快递_名词><{@=query}>
查快递_名称运单号1	Grammar	<查询_动词><expname><快递_名词><{@=query_name_num}>

你在写Grammar之前要确保Grammar中需要的Rule,Slot,Template已经定义好，并且想好自己的操作modifier。

每写好一个Grammar可以通过“例句测试”检查你要支持的句子是不是被当前的Grammar匹配，这个Grammar希望支持的句子都包含进去了，你再提交，然后发布。

发布之后你才能通过API接口进行访问。

## 五、开发自己的APP

- 你需要从语义理解API接口获取什么信息？

https的返回，比如status就不再介绍了。

说白了，开发平台解析你的语法之后，就是会告诉你这句话中的Slot和modifier，以及你的模块名称。

Slot根据你选择的类型不同，你获取的内容不同。比如ext类型的，你可以拿到slot的名称和slot的值。

Datetime类型，即你的Slot是时间，你还可以拿到时间的毫秒数，起始时间等。

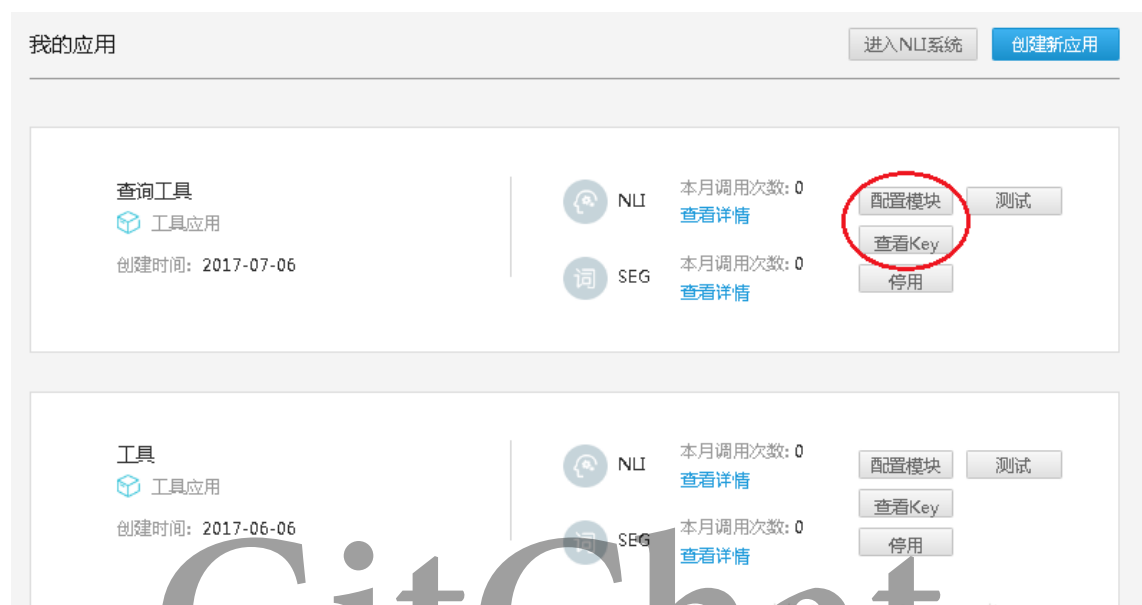
Number类型，表示你只会抓取数字，会得到数字的计算值等。

Modifier就是你自己定义的要支持的操作，只要按照他们规定的格式命名就好。

- 创建应用

应用可以包含多个模块，具体包含哪些模块也是由你自己决定。OLAMI默认支持了“聊天”，“百科”，“查询日期”等等模块。如果你不需要可以去掉。[内置模块的说明见](#)

点击下图中的配置模块添加自己写的模块和你希望添加的内置模块。比如对话系统模块中的nonsense就是聊天用的。你可以点配置模块右边的“测试”，输入要查询的句子就可以看到结果。



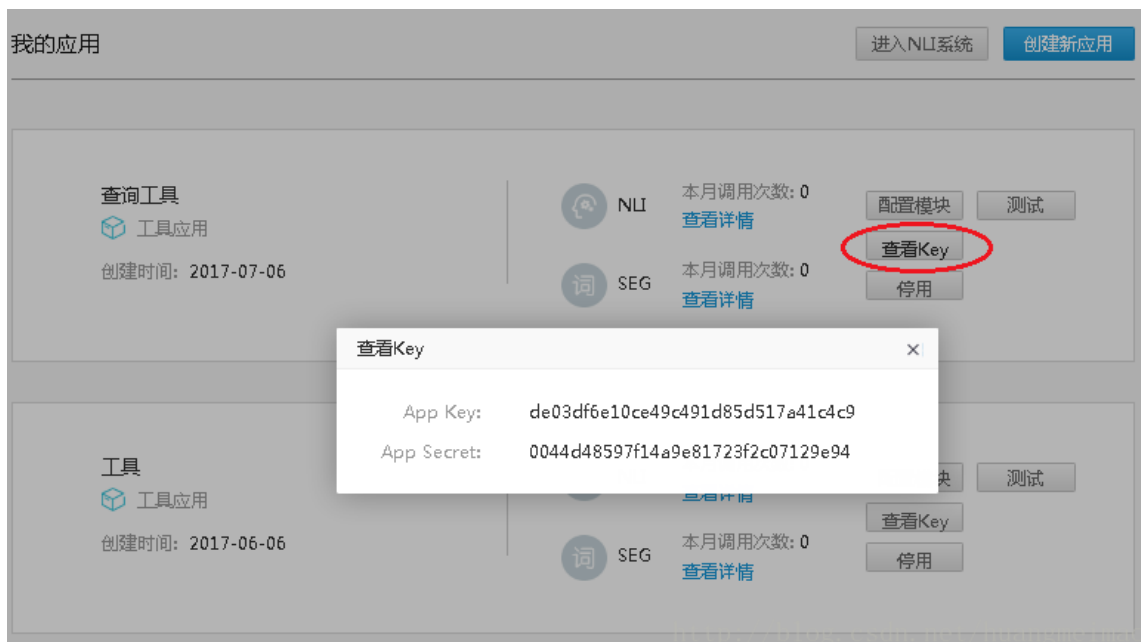
如果测试结果能正确返回，就表明API接口也可以获取同样的结果。

比如我在输入框中“查快递”就可以看到JSON格式的输，如下图显示：



- 查看应用的key

这个key就是你访问API 接口的钥匙，在你的应用中点击”查看key”就可以看到了。



- 访问自然语言解析API接口

API接口是https协议，[相关说明](#)，我就不再赘述了。

我使用的是小程序访问，相关代码如下，你代码下载包的input.js里可以看到：

```
function parseCorpus(corpus, object) {
    var usekey = Appkey;
    var usesecret = Appsecret;
    if (object.data.dialogtype == chat_type){
        usekey = ChatAppkey;
        usesecret = ChatAppsecret;
    }

    //获取sign的MD5值
    object.setData({
        text: '请稍后.....'
    })
    var timestamp = new Date().getTime();

    var originalSign = usesecret + "api=" + api + "appkey=" +
    usekey + "timestamp=" + timestamp + usesecret;
    var sign = MD5.md5(originalSign);

    var rqdata = { "data": { "input_type": 1, "text": corpus },
    "data_type": "stt" };
    console.log(JSON.stringify(rqdata))
    console.log('\r\n')
    wx.request({
        url: requestUrl,
```



```

data: {
  appkey: usekey,
  api: api,
  timestamp: timestamp,
  sign: sign,
  rq: JSON.stringify(rqdata),
  cusid: userId,
  changebuttoncolor: "#d0e0e3"
},
header: {
  'content-type': 'application/x-www-form-urlencoded'
},
method: 'POST',
success: function (result) {
  var data = result.data.data;
  if (result.data != null &&
result.data.status!=null&&result.data.status=='ok'){
    HandleOLAMiresponseData(data.nli[0], corpus, object);
    console.log('欧拉蜜有效数据', result.data);
  }else{
    console.log('欧拉蜜返回失败', result.data.status);
    object.setData({
      text: API_data_error
    })
  }
},
fail: function ({errMsg}) {
  console.log('request fail', errMsg)
  object.setData({
    text: API_data_error
  })
}
})

};

```

- 根据获取到的Semantics内容提供服务

我的处理逻辑是根据不同的modifier进行相应的操作，不同的操作下又要检查slot，代码如下：

```

function HandleOLAMiresponseData(data, corpus, object)
{
  var textData=''; //text文本框要show的内容
  var semantics = data.semantic;
  if (semantics == null || semantics.length==0){
    if (data.desc_obj.result != null &&
data.desc_obj.result.length != 0 && data.desc_obj.status==0) {

```

```

        if (data.type == 'joke' || data.type ==
'cooking'){
            textData = data.data_obj[0].content;
        }else
            textData = data.desc_obj.result;

    }else
        textData='抱歉，我还理解不了你说的话。';
    object.setData({
        expresshead: '',
        text: textData
    })

}else {

    for (var i = 0; i < semantics.length; i++) {
        var tempSem = data.semantic[0];
        if (tempSem.app == expressAPPname) { //仅处理快递

            //处理modifier
            var mods = tempSem.modifier;
            if (mods.indexOf("query") > -1)
                expAppinfo.OPT = OPT_QUERY;
            else if (mods.indexOf("query_num") > -1)
                expAppinfo.OPT = OPT_QUERY_NUM;
            else if (mods.indexOf("query_name") > -1)
                expAppinfo.OPT = OPT_QUERY_NAME;
            else if (mods.indexOf("query_name_num") > -1)
                expAppinfo.OPT = OPT_QUERY_NUM_NAME;

            //获取slots,即快递公司名称和运单号
            var slots = tempSem.slots;
            if (slots != null) {
                for (var j = 0; j < slots.length; j++) {
                    var tempslot = slots[j];
                    if (tempslot.name == expNumSlotName) { //

                        var numslot = new APPSlot();
                        numslot.name = expNumSlotName;
                        numslot.value = tempslot.value;
                        expAppinfo.numSlot = numslot;
                    } else if (tempslot.name ==
expNameSlotName) { //快递名称
                        var nameslot = new APPSlot();
                        nameslot.name = expNumSlotName;
                        nameslot.value = tempslot.value;
                        expAppinfo.nameSlot = nameslot;
                    }

                }

            }

        }
    }
}

```

模块的语义

运单号

```

//handle Operations
switch (expAppinfo.OPT) {

    case OPT_QUERY:
        textData = '请提供您的运单编号。';
        object.setData({
            expresshead: '',
            text: textData
        })
        break;
    case OPT_QUERY_NUM:
        //检测是否存在快递名称
        if (expAppinfo.nameSlot != null &&
expAppinfo.numSlot != null){ //采用快递编号+快递公司方式查询
            //获取快递公司名称
            var expname =
getExpCode(expAppinfo.nameSlot.value);
            var expcode = expCodes[expname];

            queryExpress.queryExpress(expname,expcode,
expAppinfo.numSlot.value, object);
        } else if (expAppinfo.numSlot != null){
            //用运单编号查询

            queryExpress.queryEXPbyNum(expAppinfo.numSlot.value, object);
        }
        resetExpInfo(object);
        break;
    case OPT_QUERY_NAME:
        textData = '请提供您的运单编号。';
        object.setData({
            expresshead: '',
            text: textData
        })
        break;
    case OPT_QUERY_NUM_NAME:
        if (expAppinfo.nameSlot != null &&
expAppinfo.numSlot != null) { //采用快递编号+快递公司方式查询
            //获取快递公司名称
            var expname =
getExpCode(expAppinfo.nameSlot.value);
            var expcode = expCodes[expname];

            queryExpress.queryExpress(expname,expcode,
expAppinfo.numSlot.value, object);
        }
        resetExpInfo(object);
        break;
}

break;

```

```
}  
}  
}
```

```
}
```

至此，OLAMI API 接口的基本调用工作已经完成，至于你要添加语言识别，语法完善，模块添加等就看自己的需求了。

最后说一下语法文件.osl下载之后如何导入。你创建好模块之后，直接选择**上传OSL文件**即可。

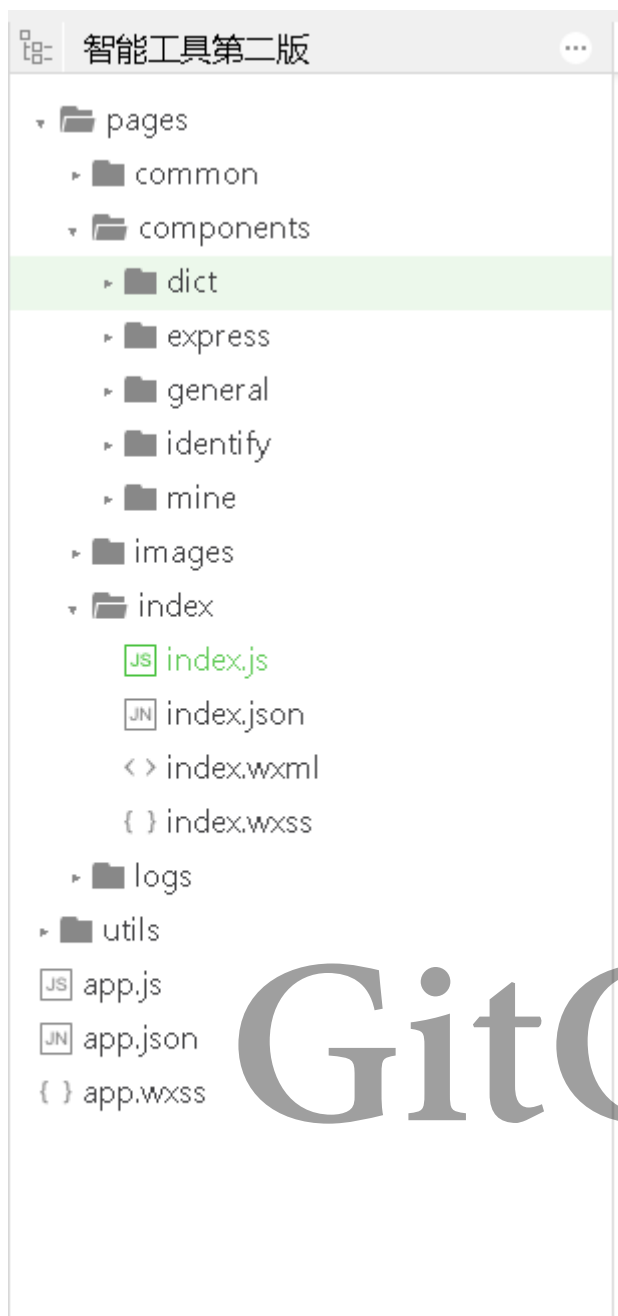


<http://blog.csdn.net/huangmeimao>

# GitChat

## 六、微信小程序查询工具代码解析

### 代码结构



图中app.json定义了所有的页面，以及标题栏和导航栏的样式，包括“首页”和“帮助”两个tabBar。

index表示“首页”page

express表示快递查询页面

identify表示身份证查询页面

dict表示词典查询页面

mine表示帮助页面

general表示其他页面，这些页面同意使用欧拉蜜官网提供的内置语法模块，所以使用统一页面代码。

但需要根据首页不同的选择传入对应的参数。

## 代码中用到的API接口

由于微信小程序仅支持https访问，因此接口必须支持https访问。

所有自然语言的解析，包括输入框中输入的语句和例句，均调用欧拉蜜人工智能开放平台中的自然语言语义理解API接口。

### 快递查询接口

这里使用的是 快递鸟即时查询接口，免费使用，请自行到官网（<http://www.kdniao.com/>）申请APPkey和BusinessID, 请填入util的queryExpress.js中对应的位置：

```
... app.json × index.js × index.wxmi × index.wxss × util.js × app.wxss × app.js >
1 //快递鸟电商ID，登录用户账号查看
2 //http://kdniao.com/
3 var CusBase64 = require('base64.js');
4 var MD5 = require('MD5.js');
5 const EBusinessID='XXXXXX';
6 const AppKey = 'XXXXXXXXXXXX';
7
8 const ReqURL = "https://api.kdniao.com/Ebusiness/EbusinessOrderHandle.aspx";
9 const errorinfo='服务器有点忙，请您稍后再试';
10
```

### 词典API

极速数据的汉语词典，申请获取的APPkey填入dict.js的相应代码：

```
function handleIDinfos(id, object,opcode) {
  //清空显示的信息
  for (var i = 0; i < idinfolist.length;i++)
    idinfolist[i].text = '';
  var requestUrl = 'https://api.jisuapi.com/cidian/word';
  var userkey = 'XXXXXXXXXXXX';
  wx.request({
    url: requestUrl,
    data: {
      appkey: userkey,
      word: id,
    },
  },
```

### 身份证查询API

极速数据的身份证查询API，申请APPkey之后填入 identify.js 相应代码：

```
function handleIDinfos(id, object) {  
    //清空显示的信息  
    for (var i = 0; i < idinfolist.length;i++)  
        idinfolist[i].text = '';  
    var requestUrl = 'https://api.jisuapi.com/idcard/query';  
    var userkey = 'XXXXXXXX';  
    wx.request({  
        url: requestUrl,  
        data: {  
            appkey: userkey,  
            idcard: id,  
        },  
        header: {  
            'content-type': 'application/x-www-form-urlencoded'  
        },  
        method: 'POST',  
        success: function (result) {
```

天气、诗歌、计算、菜谱、笑话等其他模块的输出数据均由欧拉蜜的自然语言理解接口提供结果，相关代码在 general.js 的 function parseCorpus(corpus,object) {}函数处理。

# GitChat