第 X 章 静态地图服务——Google Static Maps API

导读：本章介绍了静态地图 Google Static Maps API 的常用函数、API 调用方式以及使用静态地图 API 的开发示例。

1．Google Static Maps API 概述

Google Maps 提供了强大的地图调用函数，但是对于普通的用户来说，至少需要了解一定的 JavaScript 语法才可以用好 Google Maps 所提供的各种功能。

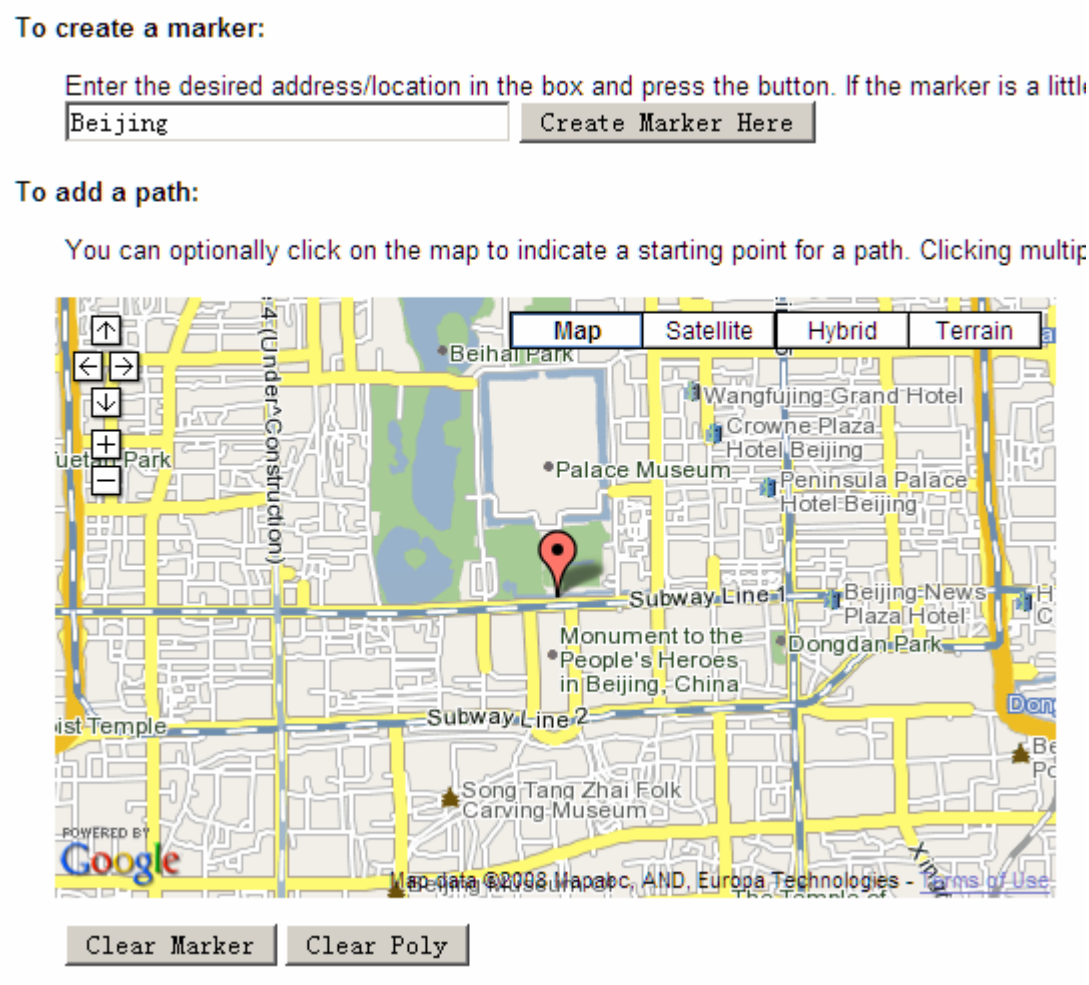
但对于部分用户来说，仅希望在自己的站点嵌入一幅静态的地图示意图即可，而并不需要 Google Maps 提供的缩放、拖拽等 Ajax 功能。那么，对于这些用户，不必编写 JavaScript 代码就可以在站点嵌入静态地图的 Google Static Maps API 便是最好的选择。

静态地图 API 极大地方便了用户调用 Google Maps 提供的地图图片，Google Maps 还为用户提供了一个静态地图的辅助生成器（http://gmaps-samples.googlecode.com/svn/trunk/simplewizard/makestaticmap.html），帮助用户在不使用动态脚本的情况下就可以定制一幅所需的静态地图图片。

在辅助生成器中，可以通过地址译码查找所需的地图位置，定义静态地图图片的缩放尺度、图片大小、地图类型、标识点位置以及标注显示的大小、颜色和显示的字母。设定之后用户定制的静态地图图片会实时刷新，并给出得到图片的预览和图片调用的 URL 地址。用户复制地址后，便可以轻易的将所需地图图片嵌入到页面中显示。

目前静态地图支持 Google Maps 所提供的四种地图类型，包含行政区划图（Map）、卫星影像（Satellite）、混合模式（Hybrid）以及地形地貌图（Terrain）。

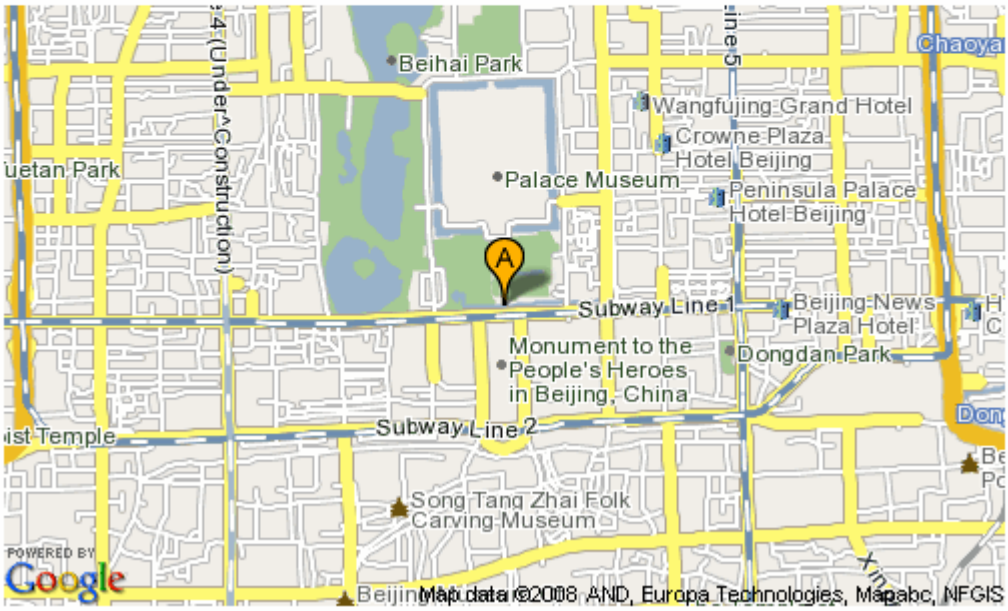
比方说我们在搜索框输入 Beijing 来调用北京的静态地图图片，如果需要设定路径，还可以在地图上通过单击来绘制一条需要显示的路径。



图：查找北京所对应的静态地图图片

接下来设定返回图片的大小为 500*300，标记点使用橙色，显示字母 A，设定好之后，就可以在下方的浏览窗口中得到一幅对应的静态地图图片。

Map	width: 500	height: 300	
Marker	size: normal	Color: orange	Letter: A
Path	Color: brown	Alpha: 50%	Weight: 5



图：Google Maps 静态地图辅助生成器中得到的图片

需要注意的是，调用 Google Maps 静态地图图片同样需要申请地图的 API_Key，申请的地址为：http://code.google.com/apis/maps/signup.html，API_Key 与使用 Google Maps 的授权码是相同的，在填入使用者申请的 API_Key 之后，刚才我们所调用的静态地图图片对应的访问地址如下：

http://maps.google.com/staticmap?center=39.908173,116.397947&markers=39.908173,116.397947,orangea&zoom=13&size=500x300&key=YOUR_KEY_HERE

通过静态地图的调用 API，可以进一步增加电子地图的应用范围，同时也使得获得地图图片的方法变得更为简洁直接。

2．Google Static Maps API 语法格式

2．1 Google Static Maps API 常用参数

Google Static Maps 的图像都是通过特定的 URL 地址返回，在 URl 请求中，用户可以指定请求的位置、地图大小、缩放级别、地图类型以及标记点的设置。

常见的 Google Static Maps 请求地址串可以表示如下：

http://maps.google.com/staticmap?center=经纬度&markers=经纬度,颜色&zoom=缩放级别&size=尺寸&key=申请的 key 值

由于地图图片请求是由标准 HTTP 提交，所有参数都需要使用&字符作为间隔，其中某

些参数是可选的。地址串中所附带的参数及其取值如下：

- Center：如果没有图标出现在地图中央的话，需要用 cent 来定义地图的中心位置，后面所跟的纬度和经度需要使用逗号作为分隔；
- Zoom：用以定义地图的缩放级别，可以通过数字选择 0 至 19 的缩放级别；
- Size：用以定义地图图像的大小，分别用横向和纵向的像素来表示；
- Format（可选）：定义生成的影像格式，默认情况下，静态地图 API 将生成 GIF 格式的图像，其他可选的格式类型为 JPEG 和 PNG 类型。相对来说，GIF 和 PNG 格式包含的图像细节更多，但 JPEG 格式的压缩比更大；
- Maptype（可选）：可以选择的地图类型包含有卫星影像、地形图、混合地图以及适于便携设备的地图，在默认情况下，我们获得的是正常的行政交通图；
- Markers（可选）：用来定义附加在地图图层之上的一个或多个地点标记，标记之间可以使用标记分隔符（ | ）进行划分；
- Path（可选）：通过两点或多点的链接来定义显示在地图之上的道路路径，同样，一系列的点之间也是使用标记(|)来进行分隔；
- Frame（可选）：为返回的地图图像添加一个可以设定颜色的边界；
- Key：填入为站点申请的 Google Maps API 授权码，这个 API Key 与使用 Google Maps 时的授权码相同，可以在 Google 开发者站点免费申请。

在使用 Google Static Maps API 时，我们还需要注意的是，目前 Google Static Maps API 的查询调用限制为每天 1000 次请求，这样的设定是为了限制用户大量获取静态地图的图片而移作他用，如果发现有滥用 API 获取静态图片的站点，Google 可能会暂停对站点地图图片请求的响应。

2．2 Google Static Maps API 的支持服务

我们在使用 Google Static Maps API 来获取地图图片的时候，往往仅知道请求位置的地址，而对其所处的经纬度信息并不了解，这样的话，我们可以通过 Google 提供的地址译码（Geocoding）服务来将用户输入的地址转换成为对应的经纬度坐标。

在 Google Maps 的 API 之中已经包含了地址译码功能，使用者可以在 HTTP 请求中调用 GClientGeocoder 对象来完成地址译码的操作。

在 Google 提供的官方示例（http://code.google.com/apis/maps/documentation/examples/geocoding-simple.html）中，我们可以使用如下的代码段来完成输入地名的地址译码操作，首先在初始化时我们通过 GMap2 类生成一幅新的地图，之后使用 GClientGeocoder 类创建地理译码器的实例，与 Google 服务器建立连接来获得地址译码的转换。目前，中国国内地图的地址解析器支持市、县、区级别的地址，在美国可以支持到街区门牌号的地址转换。

```
function initialize() {
  if (GBrowserIsCompatible()) {
    map = new GMap2(document.getElementById("map_canvas"));
    map.setCenter(new GLatLng(37.4419, -122.1419), 13);
    geocoder = new GClientGeocoder();
  }
}
```

在接下来的 showAddress 函数中，我们取得 geocoder 返回结果对应的经纬度，进行判断之后将得到的经纬度标注在地图之上。

```
function showAddress(address) {
  if (geocoder) {
    geocoder.getLatLng(
      address,
      function(point) {
        if (!point) {
          alert(address + " not found");
        } else {
          map.setCenter(point, 13);
          var marker = new GMarker(point);
          map.addOverlay(marker);
          marker.openInfoWindowHtml(address);
        }
      }
    );
  }
}
```

在从地址到经纬度的译码转换之后，我们就可以根据得到的经纬度设置地图显示的中心位置，以及进行记号点标注等操作。

当然，如果我们不希望在代码中调用 GClientGeocoder 类来实现地址译码的功能，还可以直接通过浏览器 URL 提交的 HTTP 请求来获得地址译码。

在 URL 中实现地址译码仅需要交请求提交到地址：http://maps.google.com/maps/geo?，并其后添加上相对应的浏览器请求参数，常用的请求参数如下：

- q（必需）— 后面跟将要译码的地址字串；
- Key（必需） — 申请的 Google API 授权码；
- output（必需）— 生成输出文件的格式，可选的格式为 json（默认）、xml、kml、csv；
- gl（可选）— 可以用来指定国家代码；

举个例子，我们通过请求得到 Google 总部山景城的经纬度位置如下：

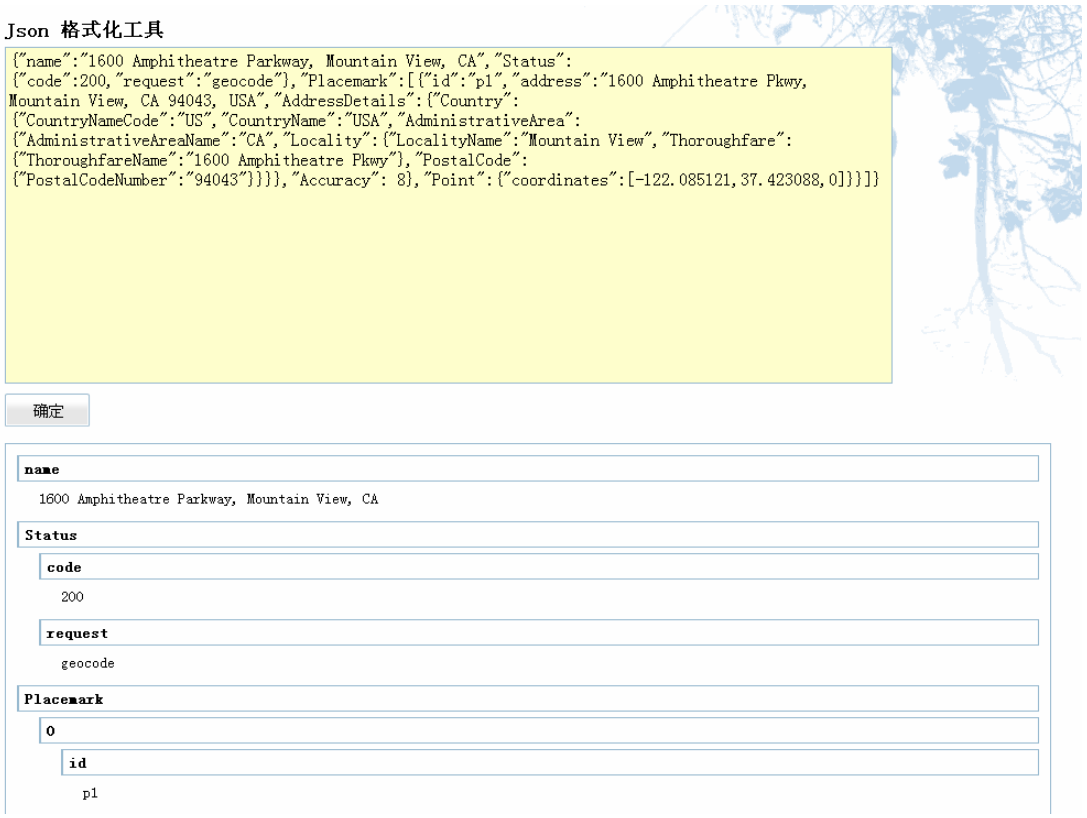
http://maps.google.com/maps/geo?q=1600+Amphitheatre+Parkway,+Mountain+View,+CA&output=xml&key=

现在我们得到的输出格式是 XML 文档的形式，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<kml xmlns="http://earth.google.com/kml/2.0">
  <Response>
    <name>1600 Amphitheatre Parkway, Mountain View, CA</name>
    <Status>
      <code>200</code>
      <request>geocode</request>
    </Status>
<Placemark id="p1">
  <address>1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA</address>
 <AddressDetails Accuracy="8" xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">
<Country>
```

```
  <CountryNameCode>US</CountryNameCode>
<AdministrativeArea>
  <AdministrativeAreaName>CA</AdministrativeAreaName>
<Locality>
  <LocalityName>Mountain View</LocalityName>
<Thoroughfare>
  <ThoroughfareName>1600 Amphitheatre Pkwy</ThoroughfareName>
    </Thoroughfare>
<PostalCode>
  <PostalCodeNumber>94043</PostalCodeNumber>
    </PostalCode>
    </Locality>
    </AdministrativeArea>
    </Country>
    </AddressDetails>
<Point>
  <coordinates>-122.085121,37.423088,0</coordinates>
    </Point>
    </Placemark>
    </Response>
    </kml>
```

如果不指定输出格式，则是按照默认的 JSON 格式将内容进行输出，上述内容的 JSON 格式输出后，我们在 JSON 格式化工具（http://lab.gracecode.com/format_json/）中查看时内容如下：



图：JSON 格式化后查看包含的内容

除此之外，在 Google Maps 提供的服务之中，还包含有 XML 和数据解析服务、使用 Flash 的方式展现街景图的示例，以及 Google Earth 浏览器插件的支持，更多的细节可以参照 Google Maps 提供服务的页面：http://code.google.com/apis/maps/documentation/services.html

如果使用 Ruby 和 Rails 进行地图 Mashup 应用的开发，可以使用名为 StaticGmaps（http://static-gmaps.rubyforge.org/）的 gems 封装，为 Google Static Maps API 提供了 Ruby 的访问接口，帮助用户简化开发过程中的工作量。

3．Google Static Maps API 示例

3．1 开发环境简介

在了解 Goolge Static Maps API 的基本函数的使用之后，下面就让我们着手制作一个可以调用静态地图的小程序，应用所要实现的功能描述如下：

- 首先需要设计一个包含输入框，确认按钮和静态地图图片显示面板的应用程序界面；
- 应用启动后，用户可以在文本框中输入所要查找的地名，点击确定按钮之后，组合静态地图的 URL 请求串，取得请求的静态地图图片，显示在窗体之中，并将图片保存在程序运行的当前位置；

简单的需求描述如上，让我们小步快跑，看看这样嵌入 Goolge Static Maps 的应用是如何实现的。

为了尝试一下新的技术并保持应用的趣味性，在这里我们使用 Python 语言、PyQt 图形库以及 NetBeans IDE 的 NBPython 编程环境来完成这个调用静态地图的小例子。

使用 Python 语言和 Qt 图形库开发的应用具备跨平台的特性，在这里我们使用的是 Windows 开发环境，开发所使用的软件环境搭建如下：

- NBPython Milestone 6
  https://nbpython.dev.java.net/
- Python 2.5.2 Windows installer
  http://www.python.org
- PyQt（PyQt-Py2.5-gpl-4.4.3-1.exe）
  http://www.riverbankcomputing.co.uk/

在开始之前，先简单介绍一下 Qt 以及程序中将使用的 PyQt 图形库。Qt 是一套成熟稳定的 GUI 工具箱，Qt 跨平台的图形库可以运行在 Windows、Linux、Mac OSX 以及多种便携式设备平台之上。Google 著名的 Google Earth 应用的界面也是使用 Qt 图形库进行设计，分别针对 Linux、Windows 以及 Mac 平台发布了不同的版本。

Qt 在设计上具备良好的面向对象结构、以及众多实用的 API。而 PyQt 则为 Python 程序员开发基于 Qt 图形库的跨平台应用搭建了桥梁。

相对于 Python 默认使用的图形库 Tk（由 Tkinter 绑定）来说，PyQt 具备众多的优点，如 Qt 使用信号 / 插槽（signals/slots）的机制在窗口构件（以及其它对象）之间传递事件和消息，让编写事件触发的代码段变得十分轻松。并且 PyQt 包含 300 多个类和近 6000 个的

函数和方法，功能十分强大。

至于 PyQt 的安装，相对来说较为复杂，通常是需要下载源代码之后在本机编译为二进制的代码，编译的过程在 Linux 环境下出错的概率较小，但在 Windows 环境下编译时往往需要多尝试几次才可以编译通过。

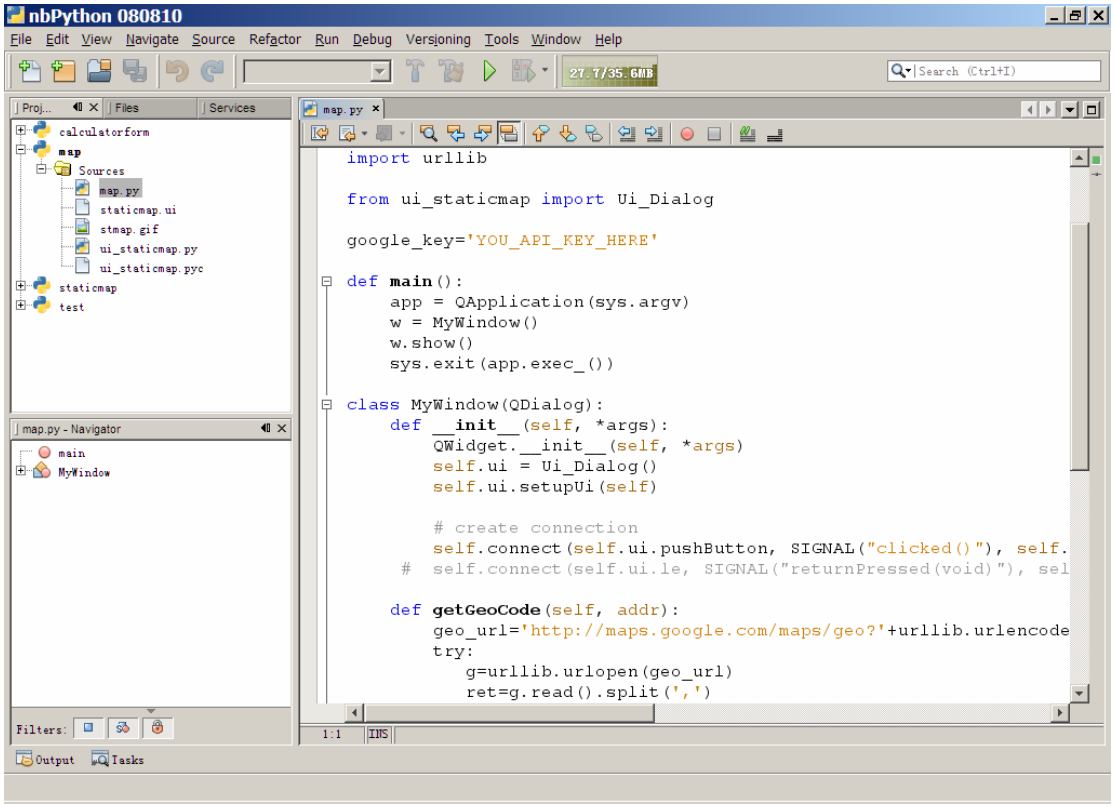
在环境搭建方面不必有过多的担心，在这里我们使用了 PyQt 提供的已经编译好的二进制安装版本 PyQt-Py2.5-gpl-4.4.3-1.exe，除了 Python 的解析器需要单独安装之外，这个版本包含了使用 PyQt 进行应用开发所有需要的要素。主要包含的项如下：

- PyQt 库
- Qt 库
- Qt Designer（界面设计器）
- Qt Assistant（编程助手）
- Qt Linguist（国际化翻译工具）
- pyuic4（ui 界面文件的编译器）
- QScintilla（Scintilla 编辑器类的 Qt 接口）

编程环境使用了 NetBeans 的 Python IDE，NBPython 可以作为一个插件加入到 NetBeans 环境之中，也可以单独下载 46MB 大小的 Netbeans Python 专用 IDE，在这个 IDE 中，计划添加的 Python 支持功能有:

- 语法高亮显示
- 代码补全
- Python 与 Jython 的支持
- PyUnit 单元测试支持
- Python 代码断点调试
- Python 类库管理
- Python 控制台
- Python 脚本执行
- Python 版本管理

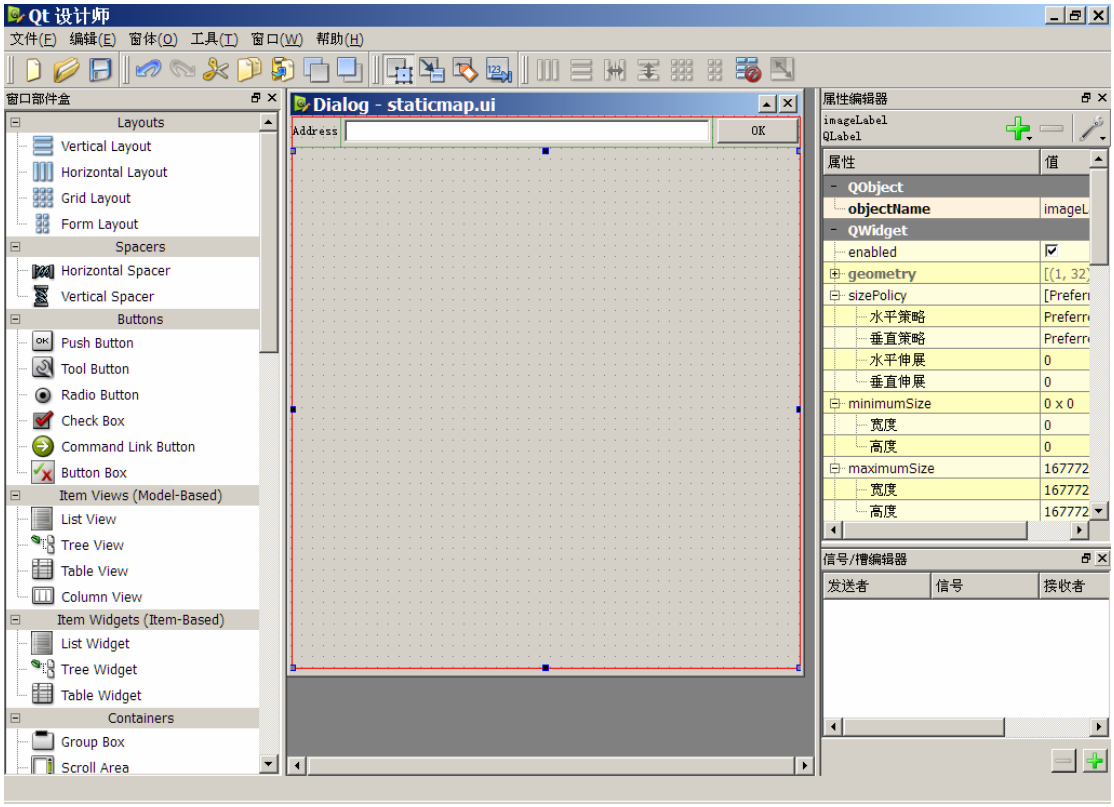
NetBeans 的 Python IDE 同样提供了不同平台下的安装版本，在 Windows 环境下的运行如下图所示。



图：NetBeans 的 Python IDE

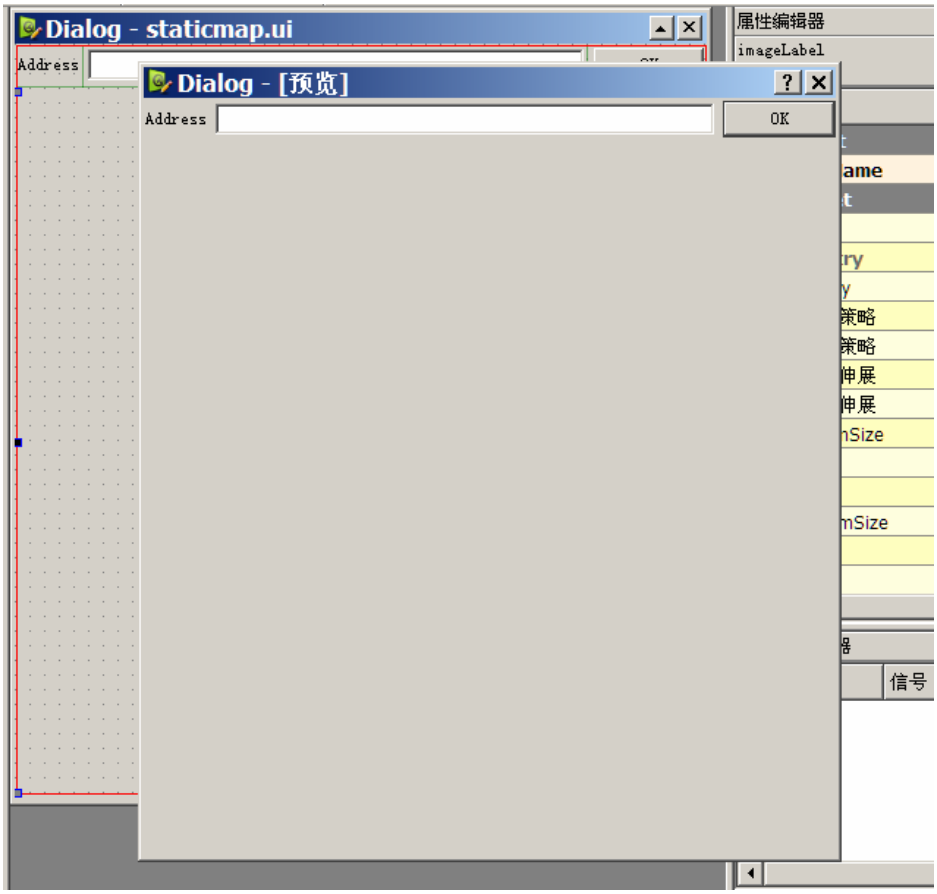
使用 Qt 进行跨平台 GUI 应用开发的优势之一就是方便的 Qt 界面设计器，使用 Qt Designer 可以很快的设计出美观大方的程序窗体布局。

下面我们打开 Qt Designer，新建一个对话框，之后在对话框中添加一个命名为 Address 的 Lable 标签（QLable），一个文本输入框（QLineEdit）以及一个确定按钮（QPushButton），在应用程序的窗口主体，也加入一个 Lable 标签，用于显示从 Google 获得的静态地图。之后使用栅格布局（Grid Layout）调整各个部件的位置。设计好之后的对话框如下图所示。



图：Qt 设计器中进行应用程序窗体设计

在使用 Qt 设计器的同时，我们还可以随时点击“窗体—预览”（Ctrl+R）来预览查看当前窗体的模样。窗体的预览如下图所示。



图：Qt 设计器中预览设计的窗体

保存之后，Qt 设计器为我们生成了一个拓展名为.ui 的文件。在 Qt 中，UI 文件以 XML 的格式记录了 QT Designer 所生成界面的相关内容，包含在 UI 文件中的内容如下：

- Widget 属性，包含的元素（如按钮、文本框、下拉列表等等），Layout 布局方式等相关属性；
- 引用的头文件（使用 C++时会编译生成头文件）；
- 变量；
- 槽（Slot）；
- 函数；

UI 文件保存之后，可以使用 Qt 提供的 UIC（user interface compiler）编译器将 UI 文件内容转换成标准.h 和.cpp 文件，在 PyQt 中，UI 文件将被转换为.py 文件。

UI 文件编译器位于 PyQt 的安装目录（Windows 环境下，通常是 C:\Python25\Lib\site-packages\PyQt4）中，在终端模式的命令行提示符下，使用目录中的 pyuic4.bat 命令即可将 UI 文件编译为 Python 的代码。使用方法如下：

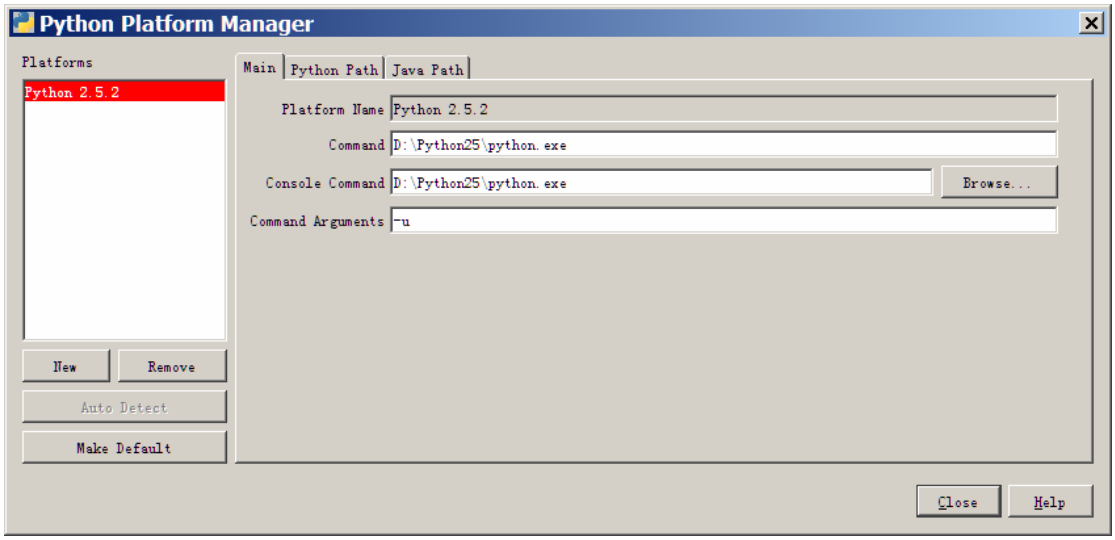
pyuic4.bat UI_FILE -o PY_FILE

（UI_FILE 为 UI 文件的路径，PY_FILE 为计划生成的 Python 文件的路径，例如，可以使用名称 ui_myapp.py 来命名生成的 Python 文件）

除此之外，pyuic4 命令后还可以添加参数来使用，参数-p（--preview）将会生成 UI 文件转换后的预览，而不生成代码文件；参数-x（--execute）生成用来测试和显示类的额外代码；参数-d（--debug）用以显示调试输出。

我们将 UI 生成的 Python 代码命名为 ui_staticmap.py，下面在 NetBeans 中新建一个工程，来为我们设计的窗体添加事件相应代码。

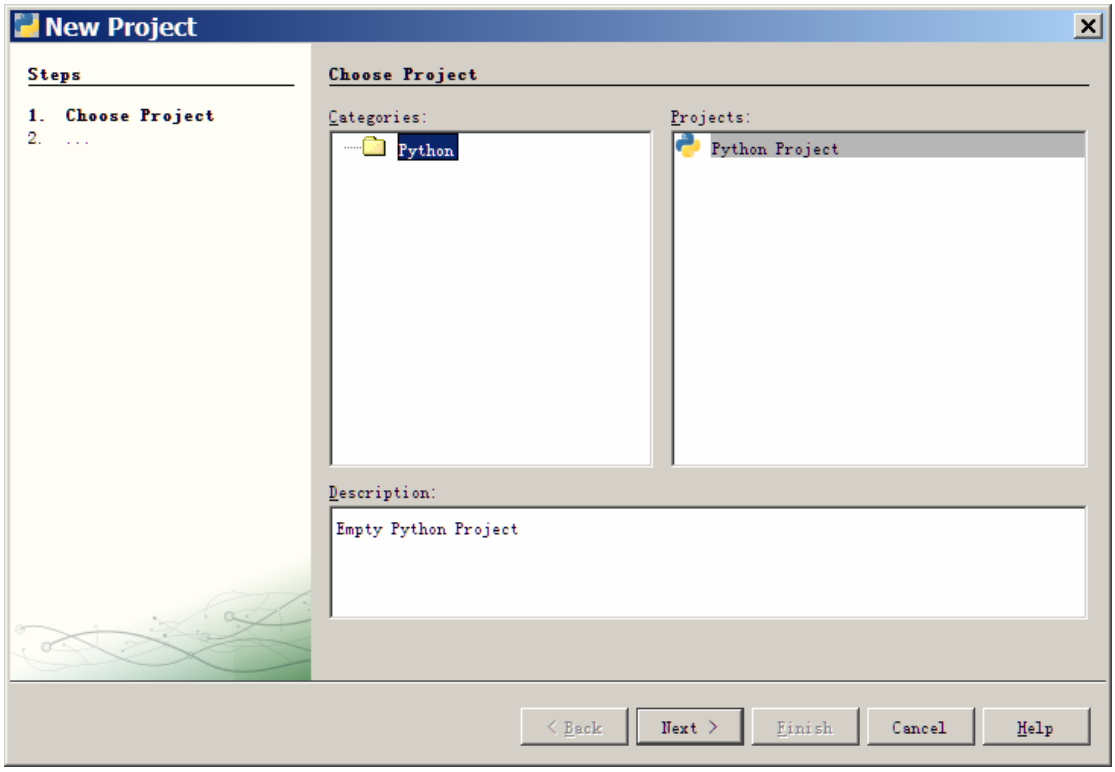
在 NetBeans 的 Python IDE 安装之后，首先需要在菜单 Tools—Python Platforms 中设置 Python 解释器的路径，如下图所示。



图：IDE 中设置 Python 解释器路径

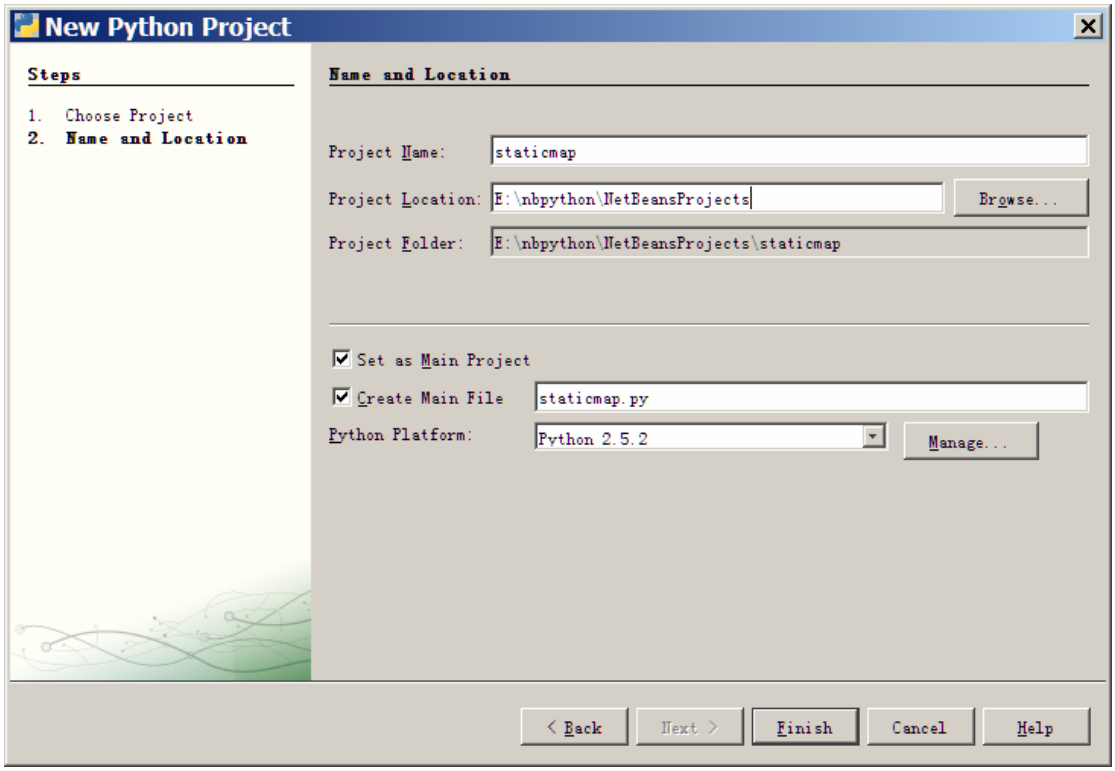
3．2 定制静态地图应用

下面我们在 NetBeans 的 Python IDE 中新建一个 Python 工程，点击创建工程后如下图所示。



图：新建 Python 工程

输入工程的名称为 staticmap，Python IDE 会为我们创建名为 staticmap.py 的代码文件，并将其设定为主文件，如下图所示。



图：设定工程名称和存储位置

下面我们来为生成的主文件 staticmap.py 添加相应的代码，首先要在程序的开始加载所需的模块：

```
from PyQt4.QtCore import *
from PyQt4.QtGui import *
import urllib
```

首先将 QtCore 和 QtGui 所包含的类引入，PyQt4 的基本模块都包含在 QtGui 中。由于静态地图的请求还需要对 URL 地址的解析，所以再引入 urllib 类库处理 HTTP 相关的部分。

主程序部分也不复杂，代码如下：

```
def main():
    app = QApplication(sys.argv)
    w = MyWindow()
    w.show()
    sys.exit(app.exec_())
```

对于 PyQt4 程序来说，首先需要创建一个 application 对象，application 类位于 QtGui 模块中，程序的主体就是这个 QApplication，sys.argv 用于传入命令行参数。

MyWindow 是后面用于定义窗体的类，w.show()的作用是将定义的窗体显示出来。最后，主程序会进入 application 的事件循环。循环不断从窗口接受需要处理的事件，然后将其分发给对应的事件处理方法。

事件循环的终止需要通过调用 exit()方法或者销毁 widget 来结束，调用 sys.exit()方法可以确保程序可以正确的退出，并且在退出时会告知系统。由于 exec 是 python 的一个关键字，

所以在 exec_()方法后会有一个下划线来表示与关键字的区别。

在 MyWindow 类中，我们指定所使用的 QDialog 是刚才在 Qt 设计器中通过 UI 文件生成的 Ui_Dialog（需要与 UI 设计器中指定的名称保持一致），代码如下：

```
class MyWindow(QDialog):
    def __init__(self, *args):
        QDialog.__init__(self, *args)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        # create connection
        self.connect(self.ui.pushButton, SIGNAL("clicked()"), self.run_command)
```

__init__内的代码段定制界面的外观，我们引用了 Qt 设计器中通过 UI 文件编译生成的 Ui_Dialog。当然，如果不是很复杂的界面，也完全可以在函数中手工编写来加入在窗体中要显示的文本框、按钮等部件。

在这里，我们通过信号槽的机制将确认按钮与接下来执行命令的操作 run_command 进行连接，也就是说，当按钮被点击时，触发的事件交由 run_command 函数进行处理。

run_command 函数的代码如下：

```
    def run_command(self):
        addr = str(self.ui.le.text().toUtf8())
        self.getGeoCode(addr)
```

在函数中取得从输入栏获取的地址作为查询条件传递给 getGeoCode 函数，QLineEdit.text()回传的 QString 字符串与 Python 字串不同，可以转成 Unicode 字串后再赋值给字符串变量。

getGeoCode 函数取得坐标的代码段如下：

```
    def getGeoCode(self, addr):
        geo_url='http://maps.google.com/maps/geo?'+urllib.urlencode({'q':addr})+'&output
    =csv&key='+google_key
        try:
            g=urllib.urlopen(geo_url)
            ret=g.read().split(',')
            if(ret[0]!='200'):
                QMessageBox.warning(None, "Error", addr+" not found",
QMessageBox.Yes)
            else:
                self.showMap(ret[2], ret[3])
        except urllib.HTTPError:
            QMessageBox.warning(None, "Error", "Http error", QMessageBox.Yes)
```

因为我们在静态地图请求中，输入项为请求地址的地名，需要通过 Google 提供的地址译码功能将地名对应的位置转换为经纬度。所以我们使用的 URL 字串为“http://maps.google.com/maps/geo?”后面带的参数是查询的地址，output 是输出的格式，目前的输出格式支持 xml、kml、csv、Json 等四种格式，我们在这里使用的是以逗号分隔的 csv 格式，后面再添加上之前申请的 Google Maps API Key 就可以得到解析后的地址坐标。

在 urllib 库的支持下，我们得到请求位置的地址坐标，如果没有找到地址对应的坐标，

则使用 QMessageBox 来弹出一个消息用以提示用户。如果找到地址对应的坐标，就将经纬度坐标作为参数传递给函数 showMap 来显示对应的地图。

函数 showMap 的代码如下：

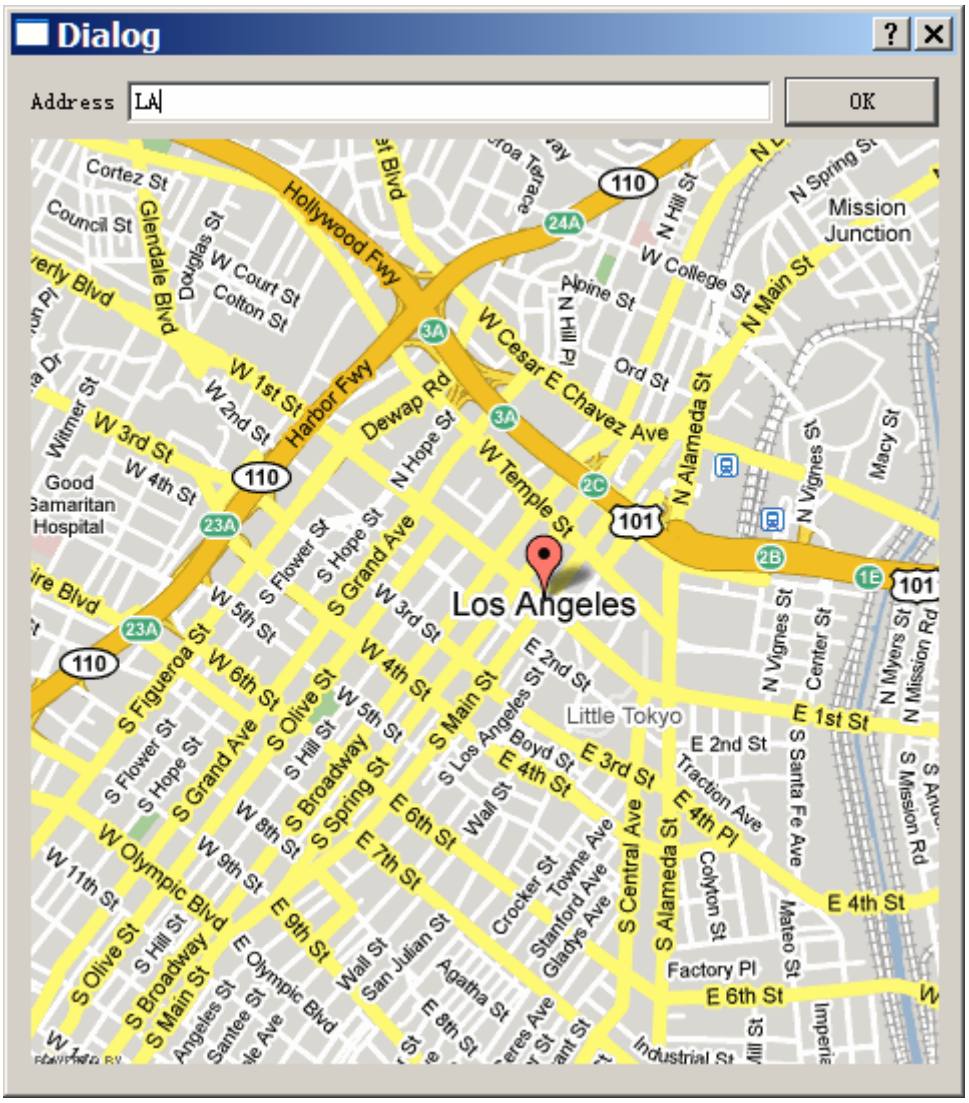
```
def showMap(self, lat,lang):
    stmap_url='http://maps.google.com/staticmap?center='+lat+','+lang+'&markers='+l
at+','+lang+',red&zoom=14&size=512x512&maptype=mobile&key='+google_key
    urllib.urlretrieve(stmap_url, "stmap.gif")
    image = QImage("stmap.gif")
    self.ui.imageLabel.setPixmap(QPixmap.fromImage(image))
    self.ui.imageLabel.adjustSize()
```

在正确取得坐标后，我们将纬度（lat）和经度（lang）传递给 showMap 函数，用以组装请求静态地图的字符串。在这里定义的缩放等级为 14，在 Google Maps 中缩放的范围是 0 到 18，0 级即为整个地球，18 级为显示最为精细的地图，但在美加等国之外，往往无法获得最为精细的地图。

取回地图的图片大小设置为 512x512 像素，同时 512x512 像素也是可以获取的最大的图片。MapType 种类设置为 mobile，比较适合在便携式设备上查看，此外还可以在地图上放置自定义的图标，此处就采用在图片中心显示的默认图标。

得到的静态地图使用 urllib.urlretrieve()函数保存为名为 stmap.gif 的图片，之后将图片填充到 imageLabel 所对应的显示标签之中。这样，取得的地图就可以显示在程序对应的视窗之上。

下面我们在 NetBeans 中点击运行（F6）来启动应用，在弹出的对话框中，输入 Los Angeles 的简称 LA，点击确定之后，就可以在程序中获取到 Google 静态地图返回的地图图片如下所示。



图：在程序中调用 Google 的静态地图

应用程序完整的代码清单参见：

```
import os
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *
import urllib
from ui_staticmap import Ui_Dialog
google_key='YOU_API_KEY_HERE'

def main():
    app = QApplication(sys.argv)
    w = MyWindow()
    w.show()
    sys.exit(app.exec_())
```

```
class MyWindow(QDialog):
    def __init__(self, *args):
        QDialog.__init__(self, *args)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        # create connection
        self.connect(self.ui.pushButton, SIGNAL("clicked()"), self.run_command)

    def getGeoCode(self, addr):
        geo_url='http://maps.google.com/maps/geo?'+urllib.urlencode({'q':addr})+'&output
    =csv&key='+google_key
        try:
            g=urllib.urlopen(geo_url)
            ret=g.read().split(',')
            if(ret[0]!='200'):
                QMessageBox.warning(None, "Error", addr+" not found",
QMessageBox.Yes)
            else:
                self.showMap(ret[2], ret[3])
        except urllib.HTTPError:
            QMessageBox.warning(None, "Error", "Http error", QMessageBox.Yes)

    def showMap(self, lat,lang):
        stmap_url='http://maps.google.com/staticmap?center='+lat+','+lang+'&markers='+l
    at+','+lang+',red&zoom=14&size=512x512&maptype=mobile&key='+google_key
        urllib.urlretrieve(stmap_url, "stmap.gif")
        image = QImage("stmap.gif")
        self.ui.imageLabel.setPixmap(QPixmap.fromImage(image))
        self.ui.imageLabel.adjustSize()

    def run_command(self):
        addr = str(self.ui.le.text().toUtf8())
        self.getGeoCode(addr)

if __name__ == "__main__":
    main()
```

Google 静态地图的功能的发挥这只是一小部分，感兴趣的朋友还可以对程序进一步拓展，为应用加上地图缩放等级的调整，手动位置的调整，甚至可以加入静态地图上进行路径设置等功能。

Google 静态地图的使用，为地图图片的调用提供了简便快捷的途径，更多的应用和说明可以参考 Google 官方提供的 Static Maps API Developer’s Guide（http://code.google.com/apis/maps/documentation/staticmaps/），通过静态地图为自己的站点或

应用添加更多有趣的内容。