存储数据

存储海量数据

数据持久化的首选方案应该是关系型数据库，关系型数据库的产品很多，包括：Oracle、MySQL、SQLServer、PostgreSQL等。如果要存储海量的低价值数据，文档数据库也是不错的选择，MongoDB是文档数据库中的佼佼者，之前我们已经讲解过MongDB的相关知识，在此不再进行赘述。

数据缓存

通过《网络数据采集和解析》一文，我们已经知道了如何从指定的页面中抓取数据，以及如何保存抓取的结果，但是我们没有考虑过这么一种情况，就是我们可能需要从已经抓取过的页面中提取出更多的数据，重新去下载这些页面对于规模不大的网站倒是问题也不大，但是如果能够把这些页面缓存起来，对应用的性能会有明显的改善。可以使用Redis来提供高速缓存服务，关于Redis的知识，我们在《NoSQL入门》一文中已经做过简要的介绍。

实例 - 缓存知乎发现上的链接和页面代码

```
from hashlib import sha1
from urllib.parse import urljoin

import pickle
import re
import requests
import zlib

from bs4 import BeautifulSoup
from redis import Redis


def main():
    # 指定种子页面
    base_url = 'https://www.zhihu.com/'
    seed_url = urljoin(base_url, 'explore')
    # 创建Redis客户端
    client = Redis(host='1.2.3.4', port=6379, password='1qaz2wsx')
    # 设置用户代理(否则访问会被拒绝)
    headers = {'user-agent': 'Baiduspider'}
    # 通过requests模块发送GET请求并指定用户代理
    resp = requests.get(seed_url, headers=headers)
    # 创建BeautifulSoup对象并指定使用lxml作为解析器
    soup = BeautifulSoup(resp.text, 'lxml')
    href_regex = re.compile(r'^/question')
    # 将URL处理成SHA1摘要(长度固定更简短)
    hasher_proto = sha1()
    # 查找所有href属性以/question打头的a标签
    for a_tag in soup.find_all('a', {'href': href_regex}):
        # 获取a标签的href属性值并组装完整的URL
        href = a_tag.attrs['href']
        full_url = urljoin(base_url, href)
        # 传入URL生成SHA1摘要
        hasher = hasher_proto.copy()
        hasher.update(full_url.encode('utf-8'))
```

```
            field_key = hasher.hexdigest()
            # 如果Redis的键'zhihu'对应的hash数据类型中没有URL的摘要就访问页面并缓存
            if not client.hexists('zhihu', field_key):
                html_page = requests.get(full_url, headers=headers).text
                # 对页面进行序列化和压缩操作
                zipped_page = zlib.compress(pickle.dumps(html_page))
                # 使用hash数据类型保存URL摘要及其对应的页面代码
                client.hset('zhihu', field_key, zipped_page)
        # 显示总共缓存了多少个页面
        print('Total %d question pages found.' % client.hlen('zhihu'))


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