**命名规则：**

* package：
  + 全部小写
  + com.geek.geekspider.clawer
  + com.geek.geekspider.extracter
  + …
* class：
  + 首字母大写，动宾结构/名词
  + com.geek.geekspider.extracter：RunClawer.java/WebClawer.java
* function/variable：
  + 首字母小写
* 具体细节请参考**/references/JAVA语言编程规范(华为).pdf**

**项目模块划分&功能实现**

整个项目分为两个工程，具体如下：

* **工程名：GeekSpider（java project）**
  + 页面抓取管理（geekspider.clawer）：
    - 设定抓取策略，访问、采集网页
    - 实现定期更新（e.g.网站添加了新的url或者原有url网页内容被修改过）
  + 页面信息过滤与抽取（geekspider.extracter）：
    - 选择抽取策略，提取关键信息，e.g.时间、标题、关键词、热度（评论数？）
    - 尽量提高抽取算法鲁棒性，而不是只针对固定几种网页模板
  + 页面信息数据库（com.geekspider.util [**公共方法，util->database**]）：
    - 抽取后的页面信息存储于webDB（基于mysql）
* **工程名：GeekSearch（dynamic web project）**
  + 建立索引（geeksearch.indexer）
    - 根据webDB（Read-Only here），建立词典（第三方词典/开源工具）：dicDB
    - term->webIDList,location，建立倒排索引：indexDB
    - 词典&索引压缩
    - 先学习lucene，后期优化尝试多种索引建立的实现方法，e.g. MapReduce等
  + 查询(geeksearch.query)
    - 查询语句分析
      * 可利用好的开源中文分词器
        + 需对现有中文分词器进行比较，选出优秀的一个
        + 将其嵌入lucene框架
    - 搜索策略
      * 合并算法等
  + 结果处理（geeksearch.sort）
    - 对相似新闻进行聚类
      * 聚类算法
      * UI上如何显示
    - 排序：
      * 默认按照相关度排序
      * User可以自己定义排序标准，e.g.相关度(location)、热度(评论数)、时间等
    - 返回结果（）：
      * 传递给web端
  + **Web UI设计与交互**
    - 搜索页：
      * 可配置项，高级搜索（结果排序选项）等
      * 查询自动补齐
      * 传递搜索词到query中（转换为string）
    - 结果页（参考百度）：
      * 从query获得检索结果
      * Snippet生成
      * 结果预览（快照）
      * 相关搜索推荐（纠错）
  + 系统测试
    - 召回率
    - 精准度
    - 响应速度