Google Style Search

花儿：陈琳
组长：王帅
码农：陈杰

2015/10/24

目录

1. 要求

Google-style search box. Key requirements:

- Google-style intelligent search, after typing 3 letters, prompt and narrow down matched list (note – UI friendly so don't overflow the screen) whilst allowing user to continue type and refine search condition.
- You can use any demo or sample DB as the repository of match criteria… eg book names, or anything that you've gathered. But must have at least 1mio records to be used to demo this search functionality.
- You can use a database of your choice. MS SQL or Oracle platform can be arranged in advance with SCB competition organiser. If you require SCB to provide development DB, pls bring DML and DDL to prepare the search data for your demo.
- No other UI element is required. Key is the actual search box and responsiveness of the intelli-search.

2. 设计思路

本题是个 web 项目，前端监听用户输入，采用 ajax 向业务服务器发送异步请求，业务服务器接受请求后，通过 client 向搜索服务器请求检索结果并返回前端，然后前端进行提示展示。其设计架构图如图 1 所示。

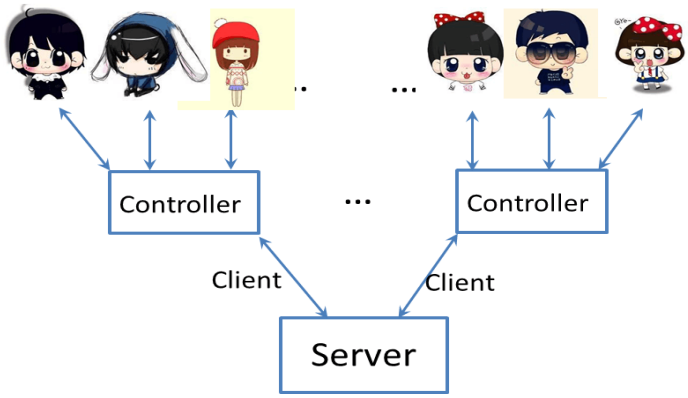

图 1. 系统整体架构图

此架构的优点：

- 实现了业务与服务的分离，可以根据需要扩展多台业务服务器来接受用户请求。
- 多台业务服务器可以共用同一台搜索服务器。

3. 详细实现

3.1. 视图层实现

页面布局模仿谷歌搜索引擎主页面，使用了 Html 5 和 Bootstrap 框架设计与实现页面，利用 jquery 和 jquery autocomplete 插件进行异步调用并完成搜索提示。在页面中，对提示框的总高度进行限制，以避免提示框超出页面范围。当返回结果高度大于设置高度时，提示框自动进行滚动显示，并显示提示项序号。

3.2. 业务逻辑层实现

业务逻辑层采用 spring mvc 框架，并采用 Annotation 方式配置进行。当控制器接受到用户请求时，通过 client 利用 thrift 协议请求搜索服务器，获取查询结果，然后以 Json 格式返回给前端。

3.3. 服务层实现

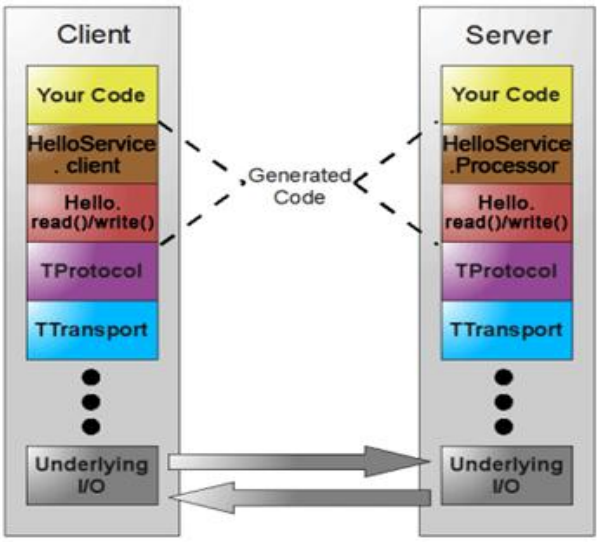

图 2. Thrift 架构图

服务层基于 thrift 框架实现。Apache Thrift 是由 Facebook 开发的远程服务调用框架，它采用接口描述语言定义并创建服务，支持可扩展的跨语言服务开发，所包含的代码生成引擎可以在多种语言中，如 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk 等创建高效的、无缝的服务，其传

输数据采用二进制格式，相对 XML 和 JSON 体积更小，对于高并发、大数据量和多语言的环境更有优势。其架构如图 2 所示.

3.4. 基于 Trie 树的查询算法实现

查询算法采用 Trie 树实现。Trie 树，又称单词查找树或键树，被广泛应用于词频统计以及前缀匹配等。由于能够最大限度地减少字符串的比较，因此经常被搜索引擎系统用于文本词频统计。Trie 树具有如下性质：

- ✧ 根节点不包含字符，除根节点外的每一个子节点都包含一个字符；
- ✧ 将根节点到某一节点路径上的字符连接起来，就是该节点对应的字符串；
- ✧ 每个节点的所有子节点包含的字符都不相同。

例如，我们有 and, as, at, cn, com 这些关键词，按照 Trie 树存储后，如图 3 所示。

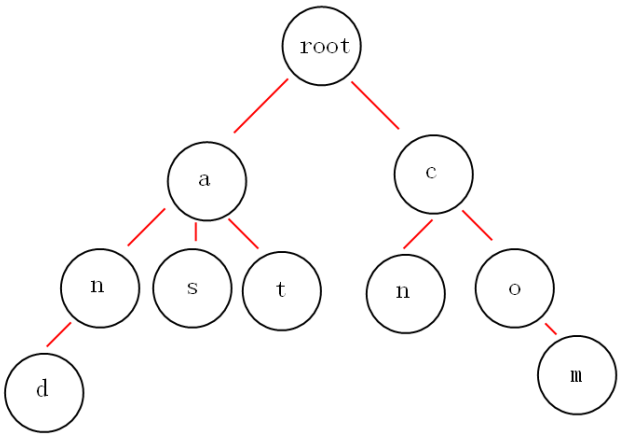

图 3 Trie 树结构

本算法中采用的 Trie 树节点结构如下：

```
class Node {
    Map<String, Node>    son;
    boolean isEnd;
    String value;
    public Node() {
        super();
         son=new HashMap<String, Node>();
    }
}
```

3.4.1. 创建 Trie 树

1) 创建 Trie 树的根结点为 root，value 为空；节点 p=root，i=0。
2) 从单词集中取 s=str[i]，j=0；
3) 取单词 s 的第 j 个字符置为 t，判断 p.son 中是否包含 key 为 t 的子节点；若存在，则将 p 指向该子节点；否则，在 p 的 son 中加入新创建的 value 为 t 的新节点 temp，并将 p 指向 temp;
4) 若已达到单词结尾，将该节点的 isEnd 属性设置为 true；否则，j++，执行步骤 3)；
5) 若单词集未遍历结束，i++，执行步骤 2；否则，程序结束。

3.4.2. Trie 树查找

1) 从根结点开始搜索，创建节点 p=root，i=0；
2) 取得关键词的第 i 个字符置为 t；
3) 判断节点 p 的 son 节点中是否包含 key 为 t 的节点，若存在，置 temp=son.get(t)，将 p 指向 temp；否则，查询失败，返回 false；
4) 若未到达单词结尾，i++，执行步骤 2)；
5) 获取最后一个字母所在节点为新的根节点，以该节点展开广度优先遍历，对每个节点 isEnd 属性为 true 的节点进行字母拼接并存储，直至达到返回结果长度或者遍历结束。

4. 效果

4.1. 系统初始页面

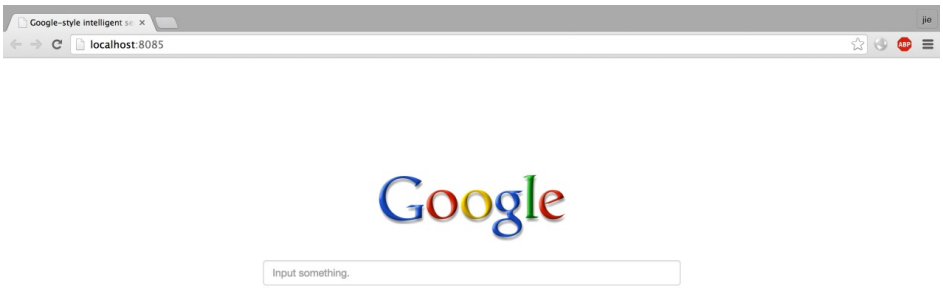

图 4. 系统初始页面

4.2. 搜索提示页面

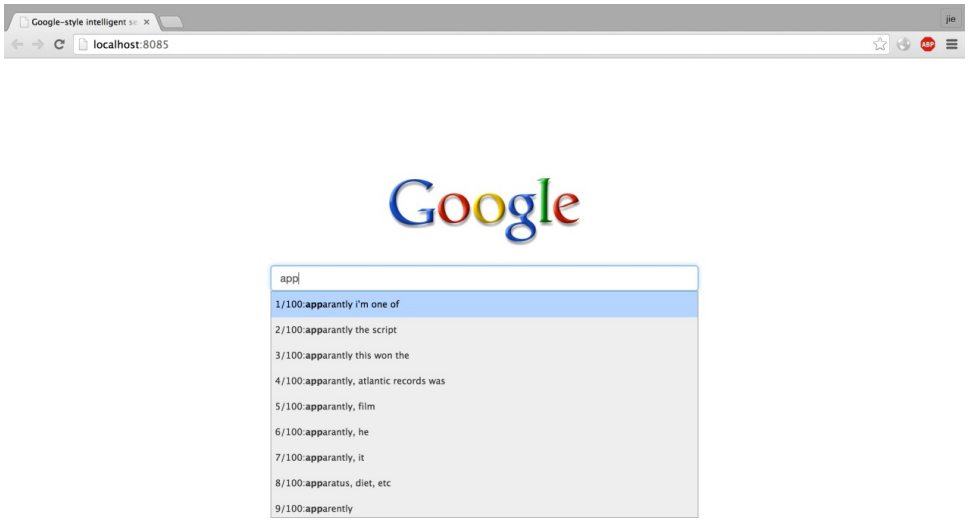

图 5. 搜索 app 提示效果

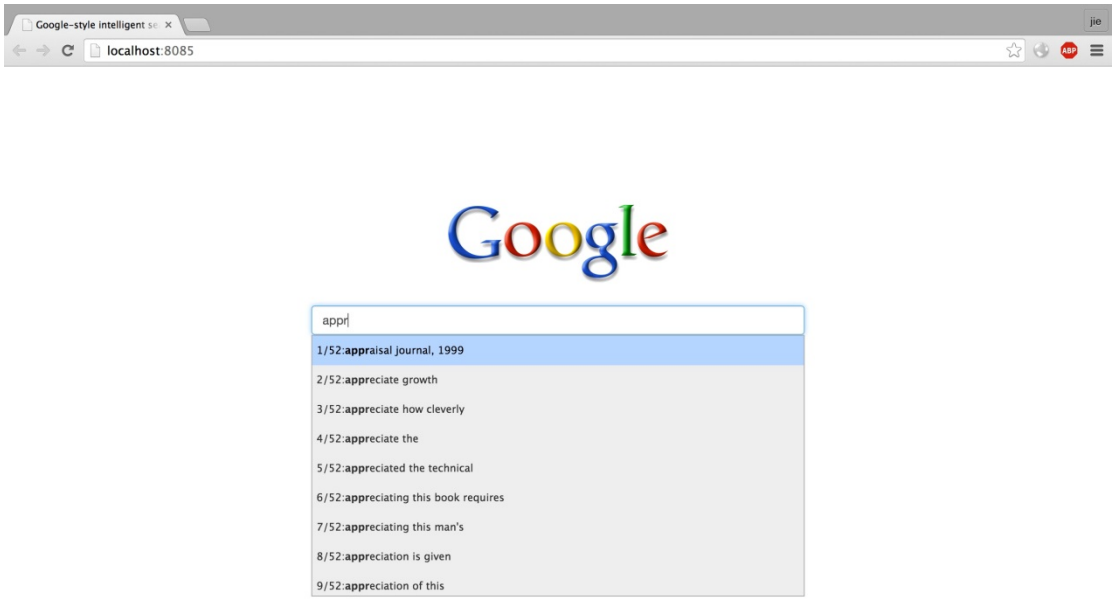

图 6. 搜索 appr 提示效果

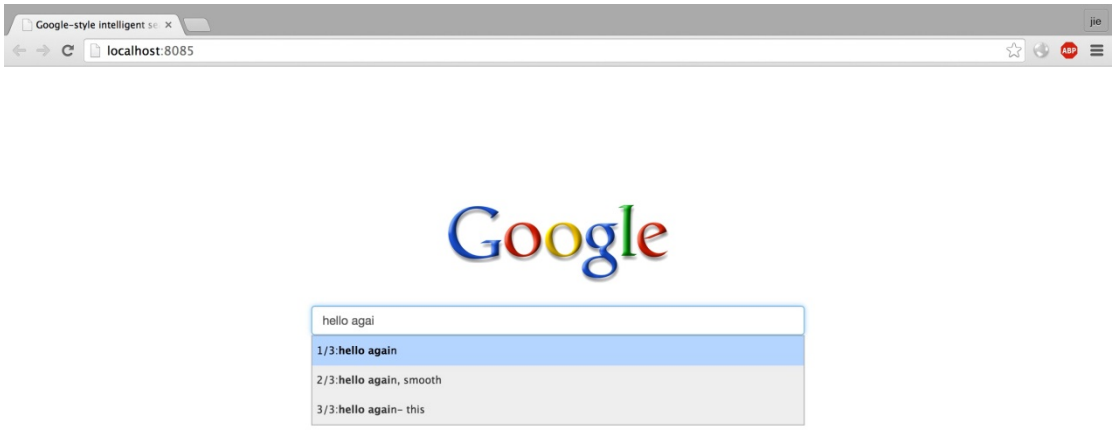

图 7. 搜索带空格字符串”hello agai”提示效果

图 8. 中文搜索提示效果

5. 运行环境及部署

5.1. 使用语言：Java

5.2. 系统环境

1) JDK: **JDK1.7+**
2) Maven: maven3.2.0+
3) Thrift: thrift0.9.2
4) Tomcat: tomcat7.x

5.3. 部署

1) 编译代码

a. 在项目根目录下执行 ./thrift_gen.sh，用于生成 thrift 代码和基础类文件

b. 编译 clint 项目。进入 hackathon-client 目录，执行

mvn clean install

c. 编译 server 项目。进入 hackathon-server 目录，执行

mvn clean assembly:assembly

在 target 下生成

hackathon-server-0.1-SNAPSHOT-jar-with-dependencies.jar 文件

d. 编译 web 项目。进入 hackathon-web 目录，执行

mvn clean package

在 target 目录下生成 hackathon-web-0.1.war 文件。

2) 运行项目

a. 运行 server，在项目根目录下执行命令

```
java -jar
hackathon-server/target/hackathon-server-0.1-SNAPSHOT-jar-w
ith-dependencies.jar -f <data file absloate path>
```

以开发环境为例：数据文件为 sentences 文件，执行命令：

```
java -jar
hackathon-server/target/hackathon-server-0.1-SNAPSHOT-jar-w
ith-dependencies.jar -f
/Users/omar/workspace/zada-hackathon/sentences
```

b. 部署 web 项目。将 hackathon-web-0.1.war 拷贝到 tomcat 文件夹中 webapps 目录中，并重命名为 ROOT.war。

c. 启动 web 项目。在 tomcat 文件夹下 bin 目录中执行 startup.sh

6. 项目亮点

- 界面简洁大方，功能齐全。
- 支持英文单词和短语查询。
- **支持中文词语和句子查询。**
- **web 服务易于扩展到多台机器。**
- **支持数据文件导入，方便测试程序。**
- **响应速度小于 10ms。**
- **词语扩展服务可嵌入到其他 JAVA 项目，易复用。**
- **采用微服务架构。**

7. 参考资料

[1] JQuery 官网 http://jquery.com/
[2] JQuery autocomplete 插件使用
http://www.cnblogs.com/hyl8218/archive/2010/03/19/1688828.html
[3] thrift http://thrift.apache.org/
[4] maven http://maven.apache.org/
[5] tomcat http://tomcat.apache.org/