EasyTesting的思考

Lucene

**Lucene** **是一个高效的，基于Java** **的全文检索库。**

**1 创建索引，通过IndexWriter对不同的文件进行索引的创建，并将其保存在索引相关文件存储的位置中。**

**2 通过索引查寻关键字相关文档。**

如何创建索引

### 将原文档传给分词器(Tokenizer)。

### 通过分词结果建立倒排索引：上面经过解析后的对应关系是：“文章号”对“文章中所有关键词”。 这种对应比较符合我们正常的思维习惯，反过来对应的话，就要颠倒思维，所以这种算法叫做倒排！ 倒排索引把这个关系倒过来，变成：“关键词”对“拥有该关键词的所有文章号”。记录关键词在文章中出现次数和出现的位置

输入搜索语句，在倒排索引中找文件

**分词器(Tokenizer)会做以下几件事情(** **此过程称为Tokenize)** **：**

**1.** **将文档分成一个一个单独的单词。**

**2.** **去除标点符号。**

**3.** **去除停词(Stop word)** **。**

Http在测试中很常用。HttpClient的原始用法有些不容易使用，因为如果不是很熟悉就会不知道各种函数以及其用法和调用顺序。在读effective java时，看到builder模式，所以就想到可以用builder模式简化httpClient的开发。比如Headers，有时会忘掉怎么加以及加什么，如果用build模式就可以通过eclipse的提醒功能直接一个一个写，把Header的属性放到一个类中，用static final标识。而且还用了工厂函数优于构造函数的原则，即newInstance

Header[] headers=HttpHeader.newInstance().userAgent("Mozilla/5.0").connection(HttpConstants.KEEP\_ALIVE).contentType(HttpConstants.APPLICATION\_JSON).build();

如果不用这个就得

Headers = new BasicHeader(userAgent, "Mozilla/5.0");

类似：

HttpClient client = EasyHttpClientBuilder.newInstance().configSSL().timeout(timeout).pool(maxTotal, maxPerRoute).build();

这里把比较复杂的ssl也通过构造模式写出来，因为ssl需要

\* 1.重写X509TrustManager和HostnameVerifier方法，使客户端信任服务器证书

\* 2.通过SSLContext设置SSL的算法，证书等上下文

\* 3.通过SSLConnectionSocketFactory（基于SSLContext和TrustManager）产生ssl socket

比较麻烦，这里直接用configSSL函数，直接在RegistryBuilder里注册了Socket，还用了PoolingHttpClientConnectionManager配置maxTotal最大连接数和maxPerRoute路由连接数

Builder模式可以隐藏复杂的方法调用关系，而且可以动态地增减，挑选需要的配置

EasyHttpClient还用了配置类，HttpConfig，它有httpClient，headers，url等配置，这样可以把所有配置都集中到一块来，通过eclipse提示简化操作

response = EasyHttpClient.get(config.headers(headers).url(baseUrl));

还有httpmethod类用了enum，就像effective java里说的enum更优

总的来说分三块，一个是Builder，EasyHttpClientBuilder，负责通过工厂模式创建httpClient；Header等通过build模式创建头。

第二是HttpConfig，存储需要的所有参数和httpClient对象。

最后是EasyHttpClient，负责执行http方法

其中，一共用了effective java里的三个原则，工厂函数优于构造函数；builder模式；enum更优（单例，可读性更好）

对于测试的一些感想：

在执行自动化测试用例执行过程中，可以用测试代码覆盖率工具统计，如java的jacoco，这样方便查看用例还有哪一些路径没有覆盖到

设计用例可以先理解模块的执行步骤比如

之后针对每个入口的参数及其预期结果进行分析，比如入参可以用等价划分，边界值，Mock异常等输入，查看是否符合预期，还可以压力测试，模拟多个用户同时操作的情况

还需要画思维导图来进行发散式测试，主要考虑不同的用户有什么侧重点，他们可能会怎么用这个产品等

如果是互联网产品还需要注意线上日志的监控，比如基于lucene的检索工具就可以用于线上监控，因为都是检测日志，原理一样，不过需要修改一下，比如可以加上多线程，加快处理速度，还可以分别布置在不同的机器上分别负责不同的文件夹，用于应对大规模的日志。把关键字写到一个文件里，定时对关键字进行搜索。

测试基本流程：前期、测试、后期，

保证手段：用例设计，自动化，静态扫描（findbugs），持续集成（jenkins），测试用例的code coverage（Jacoco），线上性能，监控

前期：评审需求，寻找需求漏洞

测试：API功能测试，集成测试，系统测试，性能测试，安全测试，回归测试

后期：收集用户反馈，线上监控等

保证方法：

用例设计：测试的基础。理解需求，熟悉架构，通读源码基础上通过以下几方面考虑：功能性，易用性，性能，安全性，兼容性，稳定性。针对每个分支每个异常进行mock和测试，每一个模块的测试用例都需要覆盖这六个方面

自动化：保证质量的同时有效降低测试的周期。框架是自动化的基础，把最常用，最繁琐的操作封装到框架降低用例编写难度，提高测试的效率。GitHub中EasyTesting就是一个我自己写的开源自动化框架，基于约定由于配置原则。其实自动化框架最好能做到Behavior Driven Development，用例只是一个逻辑的表述，不涉及代码，降低学习成本和编写难度

静态扫描：把隐蔽的非语法错误规避掉。

持续集成：尽早发现集成错误降低风险。每次改动都自动触发自动化用例。

覆盖率：通过工具查看用例的覆盖率（80-90%），哪些路径没有覆盖到，添加用例覆盖路径

性能测试：基于JMeter对系统进行性能测试，还可以做线上的压测

监控：实时搜索日志中的错误异常，及时发现线上问题

可以收集用户操作习惯，统计那些模块哪些功能是高频的，这些模块就必须充分测试，低频的后果不严重的可不用投入太多。