

CS209A Team Project Report

Author: 12011914 冯晨晨, 12013027 余坤屹

Github repo site: <https://github.com/starquakee/java2-Project> (<https://github.com/starquakee/java2-Project>)

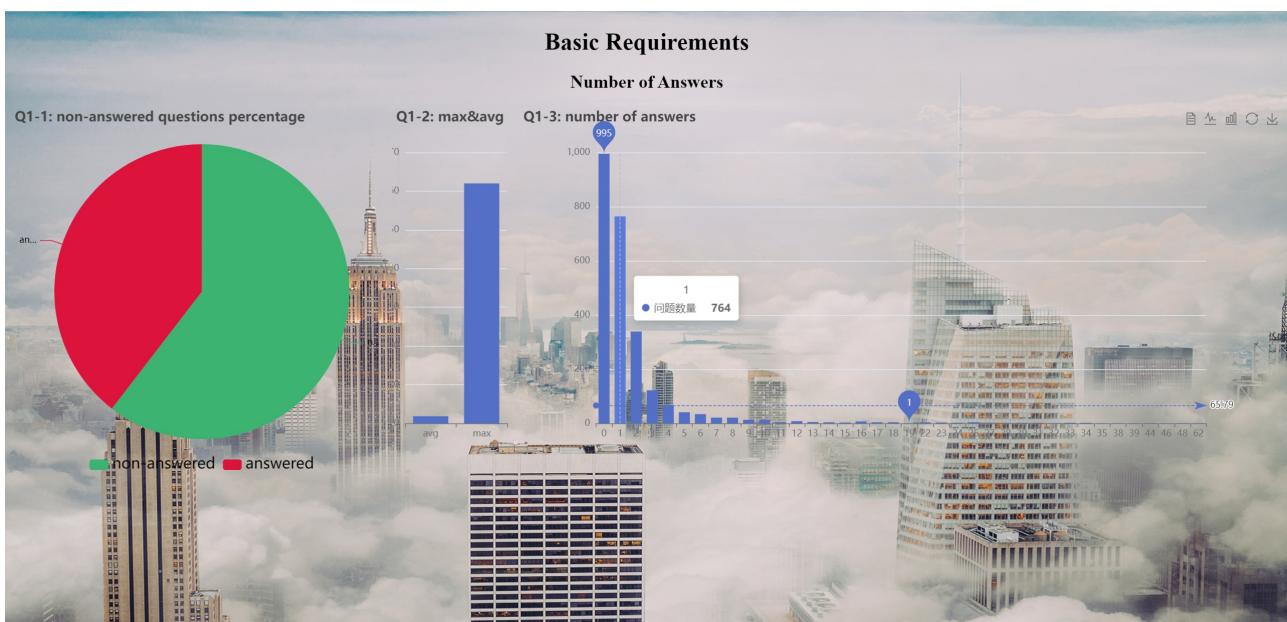
1 前端设计&项目最终呈现效果

主页效果，包含github地址，支持跳转到各个分支问题：



1.1 Number of Answers

展示了项目要求中的数据，当鼠标放置于pie chart和bar chart的item上方时会显示具体的名称和数据：



1.2 Accepted Answers

如图所示：



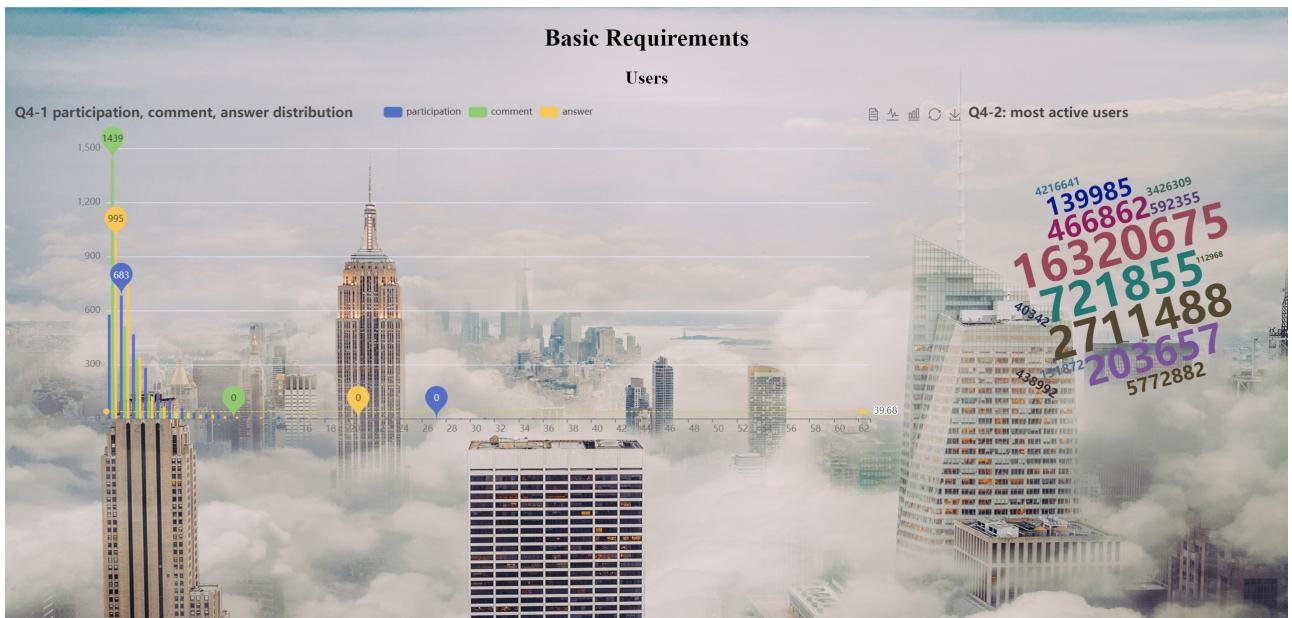
1.3 Tags

需要额外说明的是，tags标签长度原因，导致底部legend展示不全，但是鼠标放置于item上方可以查看tags的名称。tags均只展示了前10名的组合。



1.4 Users

参与度分为总体、回答、评论三个角度统计，最活跃用户展示了userid：



例证userId=16320675的活跃度：

user16320675's user avatar

user16320675 Unregistered

Member for 1 year, 11 months Last seen this week

[Profile](#) [Activity](#)

Stats

135	14k
reputation	reached
96	0
answers	questions

About

I do neither upvote nor downvote any post on this site.
I want to stay as anonymous as possible - I prefer not registering!
My comments will eventually be deleted.
Sorry for not expressing myself clearly, so my comments are incomprehensible.

1.5 Advanced_1: Java APIs

我们统计了问题、回答、评论中的content中出现JAVA official documentation中Class的数量，得出了以下统计结果，可以看到符合常理：



1.6 Advanced_2: Restful APIs

1.5中下方展示的三个Restful APIs是我们提供的三个使用示例，APIs的查询内容如名称所示：

<http://localhost:9090/QuestionDistribution?num=2>

← → ⌂ ⌂ ⓘ <http://localhost:9090/QuestionDistribution?num=2>

339

{0:995,1:764,2:339,3:122,4:67,5:41,6:34,7:21,8:21,9:13,10:12,11:4,12:8,13:5,14:4,15:3,16:7,17:4,18:4,19:1,22:4,23:3,24:2,25:4,27:1,28:1,29:2,31:2,32:1,33:2,34:2,35:1,38:1,39:1,44:1,46:1,48:1,62:1};

<http://localhost:9090/AcceptedTimeDistribution?num=200>

← → ⌂ ⌂ ⓘ http://localhost:9090/AcceptedTimeDistribution?num=200

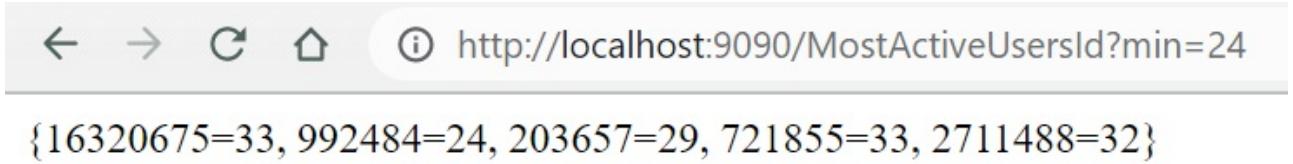
100-1000 : 155

← → ⌂ ⌂ ⓘ http://localhost:9090/AcceptedTimeDistribution?num=77

10-100 : 10

<http://localhost:9090/MostActiveUsersId?min=15>

```
{16320675=33, 992484=24, 16376827=21, 203657=29, 466862=23, 721855=33, 642706=21, 2711488=32, 5772882=17, 522444=23, 139985=20}
```



2 使用爬虫的框架和思路

实现了一个简单的爬虫实现，用于从 Stack Overflow 的 API 中获取关于 Java 标签的问题数据，并将数据保存到文件中。这段代码没有使用任何特定的爬虫框架，而是使用了 Java 的标准库来发送HTTP请求和处理数据。

代码的主要思路如下：

1. 构建目标URL：使用特定的参数构建目标URL，这里使用了 Stack Overflow 的 API，并指定了标签为 "java"，每页100条记录。
2. 发起HTTP请求：使用 URLConnection 类打开连接，并设置请求头中的 "Accept-Charset" 为 "gzip"，以支持GZIP压缩。
3. 获取响应数据：使用 BufferedReader 读取连接中的数据，并使用 GZIPInputStream 对数据进行解压缩。
4. 解析数据：通过逐行读取数据，并将其存储到 StringBuilder 对象 data_in 中。根据数据格式，使用字符串操作提取所需的问题数据部分和相关信息。
5. 数据存储：将获取到的问题数据存储到指定的文件中。使用 FileWriter 打开文件，并将 data_in 中的数据写入文件。
6. 异常处理：在可能发生的IO操作过程中，通过捕获 IOException 并进行处理来避免程序异常退出。
7. 调用入口：在 main 方法中调用 getData 方法来执行数据获取和存储的过程。这里指定了获取25页的问题数据，并将结果保存到名为 "question_2500_upvotes.js" 的文件中。

3 后端设计介绍

以第一段Number API为例，Controller提供了三个方法来处理问题数据：unansweredPercentage、average_maximumNum和distributionNum。下面解释一下后端的设计思路：

1. unansweredPercentage 方法：该方法计算并输出未回答问题的数量和未回答问题的百分比，并将结果保存到文件中。它首先读取保存问题数据的文件，然后解析JSON数据，提取每个问题的回答状态。通过迭代问题列表，统计未回答问题的数量，并计算未回答问题占总问题数量的百分比。最后，将结果保存到名为 "Number_1.js" 的文件中。
2. average_maximumNum 方法：该方法计算并输出问题的平均回答数量和最大回答数量，并将结果保存到文件中。它与前一个方法类似，读取保存问题数据的文件并解析JSON数据。通过迭代问题列表，统计所有问题的回答数量，并计算平均回答数量。同时，跟踪最大回答数量并更新。最后，将平均回答数量和最大回答数量保存到名为 "Number_2.js" 的文件中。
3. distributionNum 方法：该方法计算并输出问题回答数量的分布情况，并将结果保存到文件中。同样，它读取保存问题数据的文件并解析JSON数据。通过迭代问题列表，统计每个回答数量出现的次数，并将结果存储在一个映射（Map）中。最后，将问题回答数量的分布情况保存到名为 "Number_3.js" 的文件中。

这些方法通过解析预先已经存储好的问题数据的JSON格式，提取所需的信息，并进行相应的计算和处理。它们根据问题数据的特定要求，计算相关的统计指标，并将结果以特定的格式保存到文件中。这样设计的后端控制器可以被前端或其他部分调用，以获取并处理问题数据的相关统计信息。其他API部分与上类似。