**中期文档**

|  |  |  |  |
| --- | --- | --- | --- |
| 论文（设计）题 目 | 基于水鸟系统的web全栈开发研究 | | |
| Research on web full stack development based on water bird system | | |
| 指导教师 | 王晗 | 职 称 | 讲师(高校) |
| 题目来源 | ■科研 □生产 □模拟 □其它 | | |
| 题目类别 | □论文 □设计 ■论文+设计 | | |
| **一、概述** | | | |
| 已实现功能：  1. 获取最新水鸟新闻  2. 获取所有水鸟简要信息  3. 获取所有水鸟详情信息  4. 获取留言板所有内容  5. 根据水鸟名称搜索  6. 根据地区名字筛选搜索  7. 留言API  8. 每日推送水鸟信息API  9. “你是哪种水鸟”测试  细节优化：   1. 爬取图片时自动备份至云端 2. 爬取新闻时自动过滤掉无图片或无正文的新闻 3. “你是哪种水鸟”测试结果绝非随机生成，每次结果有及其微弱的差别是为了保证与数据库里面水鸟的匹配多样性 4. 采用皮尔森相关系数，能保证其相关度准确性 5. 搜索功能采用模糊搜索，保证最大搜索量 6. 后台Log系统，保证出错时及时找到BUG并修复 7. 整体服务器采用go搭建，Go就是不需要nginx、apache服务器，因为他直接就监听tcp端口了，做了nginx做的事情，然后监听端口函数就是写的逻辑函数了，跟php里面的控制层（controller）函数类似。 8. Go通过简单的几行代码就已经运行起来一个Web服务了，而且这个Web服务内部有支持高并发的特性 | | | |
| **二、实现方法以及存在的问题** | | | |
| **1.获取最新水鸟新闻：**  a.通过内置的http.get获取搜索页面的html文件，并存放在res中  b.如果状态码为200，则从res（文件流形式）读入文档，存放在doc变量中,  c.通过寻找特定的标签，遍历搜索结果，获取新闻题目以及新闻的url  d.通过获取的url再次执行a, b类似的操作  e.在新闻具体的页面寻找特定标签，保存其正文与图片到json中  f.将相应头的代码设置为200，将结构体的内容通过json放置在主体返回给前端  **2．获取水鸟相关信息：**  a.利用go-sql-driver打开数据库  b.通过query函数执行相关SQL函数  c.执行函数后通过scan函数获取值，并存放在事先建立好的结构体中  d.将相应头的代码设置为200，将结构体的内容通过json放置在主体返回给前端  e. 每日推送水鸟信息API,则根据random函数先获取任意2个id，并通过id执行相关SQL函数，其余部分同c, d。  **3.搜索功能：**  a.通过Param获取请求主体内的相关元素，并将其转换为string类型存储  b.利用go-sql-driver打开数据库  c.通过query函数执行相关SQL函数  d.执行函数后通过scan函数获取值，并存放在事先建立好的结构体中  e.将结构体的内容通过json返回给前端  **4.留言API**  a.接受POST请求，获取POST请求中的用户名，留言内容以及联系方式，并将其转换为string类  b.通过time函数获取发送请求时间并存放在变量中  c.利用go-sql-driver打开数据库  d.通过query函数执行相关SQL函数将数据插入数据库中，同时检测error  e.如果插入成功，相应头的代码设置为200，返回给前端  **5. “你是哪种水鸟”测试**  a. 接受POST请求，获取POST请求中的每个input的value  b．遍历答案，首先判断题目类型，然后根据答案的不同，不同的属性变量自增  c. 通过四个维度大于0亦或是小于0判断其类型，并生成其相关类型的描述  d. 打开数据库，通过遍历数据库内每一只水鸟的数据，将两组数据看作（xi,yi）的四个点（因为每一个集合有四个维度），然后通过计算皮尔森相关系数https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D227/sign=2bad72cd59e736d15c138b0aac514ffc/b2de9c82d158ccbfff22892b14d8bc3eb1354100.jpg来检测两组数据相关程度，并得到最相似的水鸟名称以及相似度。  e.根据水鸟名称在数据库中搜索出水鸟相关信息，并将数据以json形式返回给前端  **所遇到问题：**  **Q：**部分图片以base64编码,无法直接获取其url  **A：**获取到src标签内值后通过sniff进行嗅探，确定其为base64编码后，将图片以base64编码并通过阿里云提供的api上传到云端的bucket中并获取其url再返回给服务器，服务器再将数据打包存入json  **Q：**部分图片以url形式存在src标签中，但存在防盗链机制，获取其url无法直接使用。  **A：**获取到src标签内值后通过sniff进行嗅探，确定其为html后，将图片以base64编码读入文件流并通过阿里云提供的api上传到云端的bucket中并获取其url再返回给服务器，服务器再将数据打包存入json  **Q：**部分图片因为其命名方式的问题而导致文件冲突而被覆盖  **A：**采用特殊的命名方式：  前三个数字是从时间戳生成的。  如果时间戳值失去单调性，则第四个数字保留时间唯一性。  第五个数字是提供空间唯一性的IEEE 802节点号。  UID碰撞保证在了极低的情况。  **Q：**同一种水鸟可能分布在多个地区，同一种地区可能存在多种水鸟，并不能简单地一种水鸟对应一个地区属性，这样在数据冗余以及查询方面都有困难  **A：**通过创建水鸟信息表，水鸟地区表，水鸟及地区关联表，并添加联合主键。查询时语句：  select  waterflow\_info.name '水鸟名称',  area.area\_name '所属地区'  from  waterflow\_info,  area,  waterflow\_REF\_area  where  waterflow\_info.id = waterflow\_REF\_area.waterflow\_id  and Area.area\_id = "NM"  and Area.area\_id = waterflow\_REF\_area.area\_id;  达到联合检索的效果，并节约了数据存储  **Q：**留言存在大量数据，水鸟信息依旧同样如此，页面不可能一次性显示所有的数据。  **A：**前端在发送post或者get请求时同时加上page属性，根据page标识符来确认所需要的数据数量以及范围，并添加筛选范围。 | | | |
| **三、参考文献以及相关结构体** | | | |
| 参考文献：  [1]Golang[EB/OL].https://golang.org/.  [2]GoDoc-Air[EB/OL] https://www.godoc.org/github.com/aofei/air  [3]GoDoc- github.com/PuerkitoBio/goquery[EB/OL]https://www.godoc.org/github.com/PuerkitoBio/goquery  [4]GoDoc-github.com/aofei/mimesniffer[EB/OL]https://www.godoc.org/github.com/aofei/mimesniffer  [5]GoDoc-github.com/aofei/sandid[EB/OL] https://www.godoc.org/ github.com/aofei/sandid  [6]GoDoc-github.com/aliyun/aliyun-oss-go-sdk/oss"[EB/OL] https://www.godoc.org/ github.com/ aliyun/aliyun-oss-go-sdk/oss  [7]GoDoc-github.com/go-sql-driver/mysql[EB/OL]https://www.godoc.org/github.com/go-sql-driver/mysql  [8]张峰,谢振华,程江涛,崔高仑,徐衡博.基于向量皮尔森相关系数的组合赋权法[J].火力与指挥控制,2015,40(05):83-86.  结构体：  type NewsMessage struct {  PostTime string `json:"get\_time"`  NewsList []NewsLists `json:"news"`  }  type NewsLists struct {  NewsName string `json:"newsname"`  NewsUrl string `json:"newsurl"`  Newstime string `json:time`  Author string `json:author`  Body string `json:content`  Imageurl string `json:imageurl`  }  type waterflow\_info\_struct struct {  PostTime string `json:"get\_time"`  Waterflow\_List []Flows `json:"waterflow\_info"`  }  type Flows struct {  Uid int `json:"uid"`  Name string `json:"name"`  Order string `json:"order"`  Family string `json:"family"`  Genus string `json:"genus"`  }  type Message\_board\_struct struct {  PostTime string `json:"get\_time"`  Board []Message `json:"Message"`  }  type Message struct {  Uid int `json:"uid"`  Name string `json:"name"`  Comment string `json:"comment"`  Mail string `json:"address"`  SendTime string `json:"sendtime"`  }  type waterflow\_detail\_struct struct {  PostTime string `json:"get\_time"`  Waterflow\_Detail []Flows\_detail `json:"waterflow\_detail"`  }  type Flows\_detail struct {  Uid int `json:"uid"`  Name string `json:"name"`  Latin\_name string `json:"latin\_name"`  Introduce string `json:"introduce"`  Imgurl string `json:"imgurl"`  }  type Test\_result struct {  Ie float64 `json:"内倾/外倾"`  Sn float64 `json:"感觉/直觉"`  Tf float64 `json:"思考/情感"`  Pj float64 `json:"知觉/判断"`  Kind string `json:"类型"`  Introduce string `json:"描述"`  Similar\_name string `json:"相似水鸟"`  Similar string `json:"相似度"`  Similar\_imageurl string `json:"该水鸟图片地址"`  } | | | |
| **四、接下来需要解决的问题** | | | |
| 1. 搜索结果的呈现方式优化 2. 将水鸟测试页面与前端对接 3. 水鸟四维的值商榷以及数据库注入 4. 水鸟测试题目以及测试结果的优化 5. 相似度匹配算法的优化 6. 爬虫函数的优化 | | | |