Java Web Application Framework Populariy Analysis in Recent Years.

Intro

Topic

信息化世界的技术总是瞬息万变。作为学习者，我们总得时刻提醒自己注意周边环境的变化：我们正在学的技术是否已经过时？我们正在用的框架是否已经有了更优化的版本？在茫茫的技术海洋中，有哪些才是真正有价值的，值得我们用心领悟的？

基于以上考量，我们试着瞄准 Java 网络开发框架进行了一次探究。为了分析一项技术的生命力，我们试着从开源社区入手，希望通过考察其被用来进行创造的活跃度、以及人们对相关产品的关注度，来实现我们的最终分析。

于是，本项目针对近年来在 Java Web Application 开发中较为流行的框架，通过爬取开源网站 Github 上的相关仓库信息（仓库stars数量、forks数量、各时间段仓库创建数等），对各个框架在社区中的流行度与活跃度进行了分析比较。经由数据库进行存储和筛选，本项目以多样化的图表的形式呈现了数据分析结果，并提供将各个框架的开源仓库整合在一起的数据库图形接口，以便用户进行整体浏览和检索。

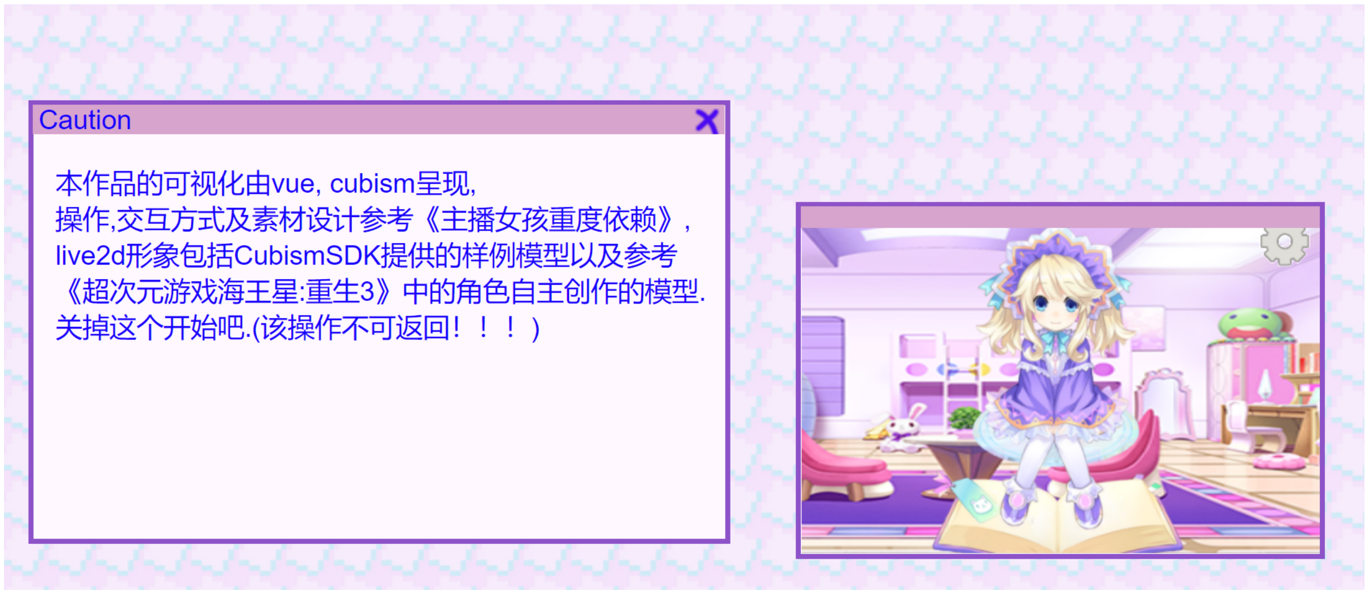
本项目分析的 Java 框架包括：

- **Spring**: Enterprise-level Java application framework
- **Struts**: another MVC framework for enterprise-level Java applications
- **Apache Spark**: Micro framework for web apps and REST APIs
- **GWT(Google Web Toolkit)**: client-side Java apps deployed as JavaScript
- **Dropwizard**: a high-performance but straightforward Java framework for rapid development of RESTful web services.
- **Blade**: Simple application framework with a minimal footprint
- **Vert.x**: Polyglot event-driven application framework for the Java Virtual Machine

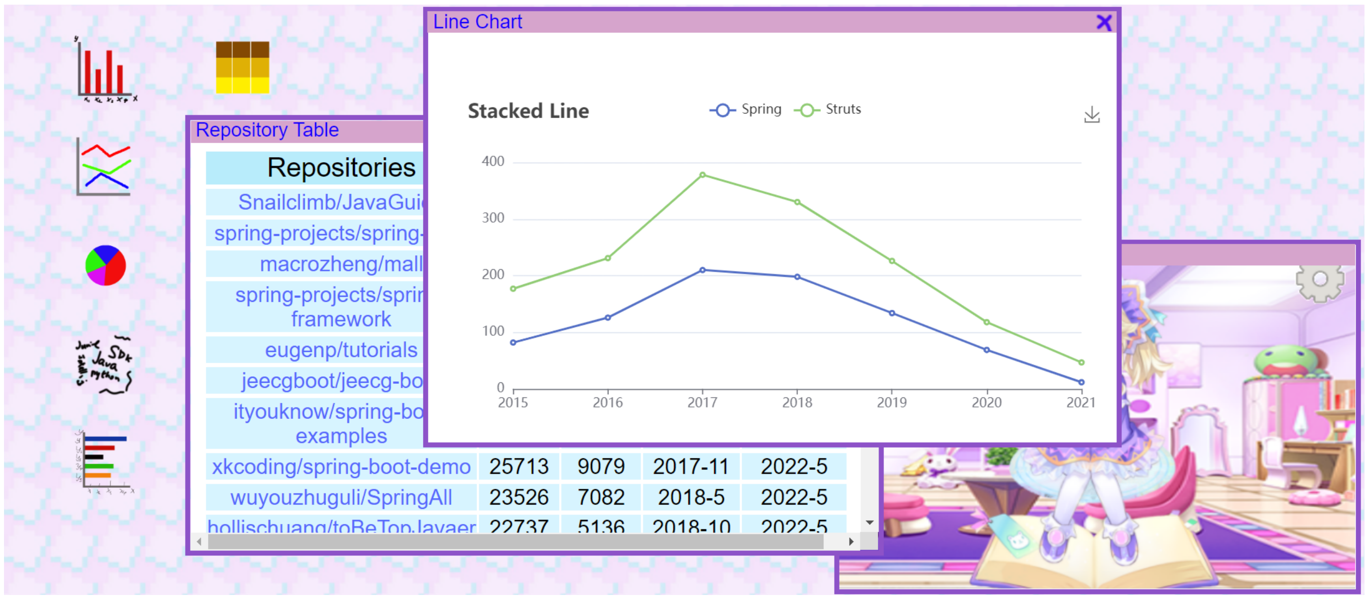
Ref: https://raygun.com/blog/popular-java-frameworks/

我们主要针对了可以进行完整 **Web Application** 开发的框架进行分析，其他如 `Hibernate`, `MyBatis` 等以 ORM 为核心或主要为数据库操作提供便利的框架，或如 `JSF`, `PrimeFace` 等便利 UI 设计的框架，本项目没有涉及。

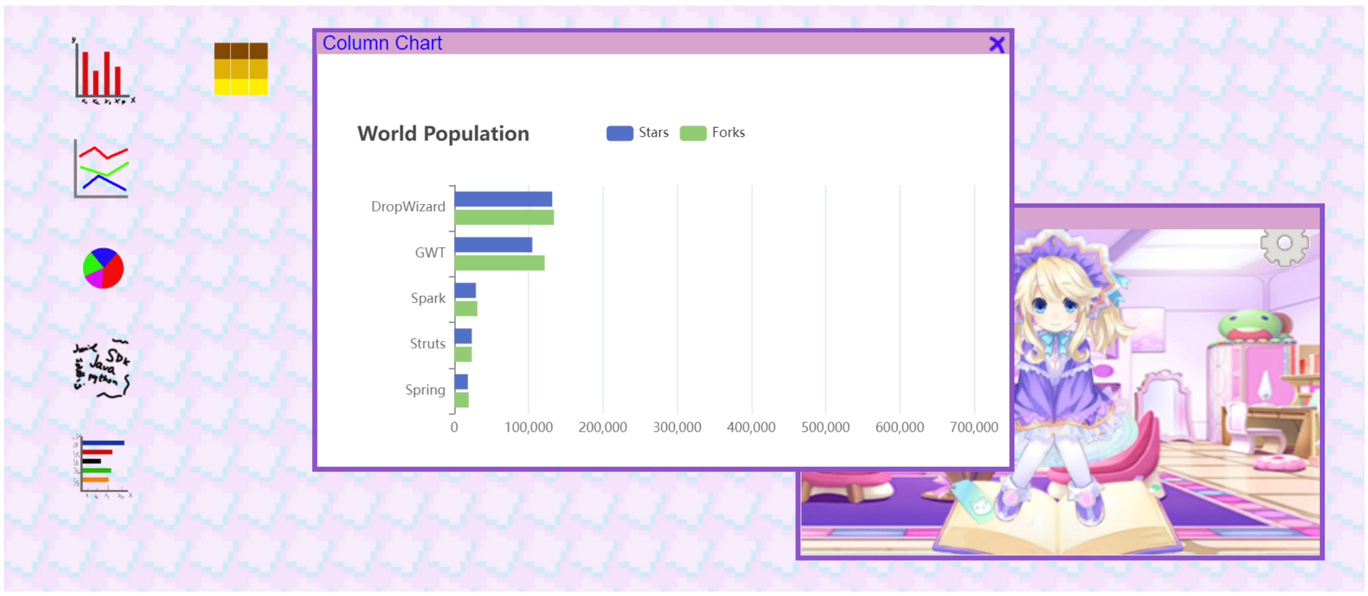
效果呈现

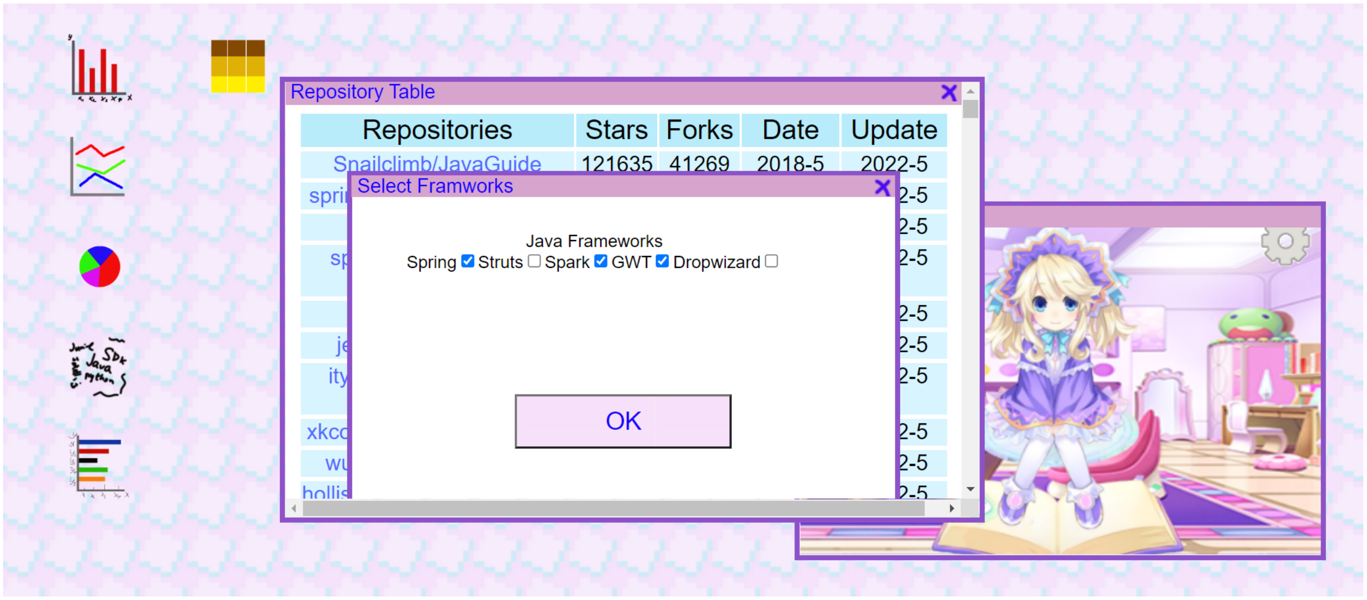


本作品的可视化由vue, cubism呈现,
操作,交互方式及素材设计参考《主播女孩重度依赖》,
live2d形象包括CubismSDK提供的样例模型以及参考
《超次元游戏海王星:重生3》中的角色自主创作的模型.
关掉这个开始吧.(该操作不可返回! ! !)



Repositories				
Snailclimb/JavaGui				
spring-projects/spring-				
macrozheng/mall				
spring-projects/sprir framework				
eugenp/tutorials				
jeecgboot/jeecg-bo				
ityouknow/spring-bo examples				
xkcoding/spring-boot-demo	25713	9079	2017-11	2022-5
wuyouzhuguli/SpringAll	23526	7082	2018-5	2022-5
hollischuang/toBeTopJavaer	22737	5136	2018-10	2022-5





Repositories	Stars	Forks	Date	Update
Snailclimb/JavaGuide	121635	41269	2018-5	2022-5

Java Frameworks

Spring ☑ Struts ☐ Spark ☑ GWT ☑ Dropwizard ☐

项目结构

- Frontend: Vue, ...
- Backend: SpringMVC + Spring + MyBatis (with SpringBoot)
- DataBase: MariaDB (on Linux)
- DataSource: `api.github.com`
- WebScraper & DataProcess: `Java`

团队分工

- 12011411 吴笑丰：后端、数据库设计、部分数据爬取
- 11911109 张倚凡：前端
- 12012428 沈徐檑 ：数据爬取与处理

总体分析

- 总流行度：柱状图、词云
  - 数据：`[[framework1,framework2,...], [num1, num2, num3...]]`, `[{frame1:num1}, {frame2:num2},...]`
- 十年活跃度变化：动态增长图 / 折线图
  - 动态图须传数据：`[["popularity", "framework", "date"],[data],[],[]...]`
  - 折线图须传数据：`name:{framework1, framework2, ...,}, x:{2012, 2013, 2014, ...}, y:{data1, data2, ...} * frameworks`

Feature：可以选择具体框架图标进入图表中

总流行度 = stars数 + forks数

每年活跃度 = 创建仓库数/commit数（月为基本单位）

数量级的差异：spring 和 sparks 框架与其他框架不在一个数量级上，另外两者同样相差较远。因而我们在爬取了两者的高stars数仓库后对整体的数据量进行了控制。由于其他框架的仓库数大概在1w条上下，Spring仓库的数目在60w左右，sparks在10w左右，我们对两者分别取了5w和2w条左右的数据。

整合数据库

将所有框架的仓库整合在一个数据库中，可以点击访问，并进行查询或通过stars、forks、创建时间排序。

Backend

后端部分使用 Maven 进行项目管理，并采用了 SpringMVC + Spring + MyBatis 的SSM整体框架，分别实现了表现层、业务逻辑层、数据访问层。

文件结构

```
.
├── backend.iml
├── HELP.md
├── mvnw
├── mvnw.cmd
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   └── edu
│   │   │       └── sustech
│   │   │           └── backend
│   │   │               ├── scraper
│   │   │               │   ├── DataBaseController.java
│   │   │               │   ├── gitHub.java
│   │   │               │   ├── github_scraper.java
│   │   │               │   ├── jsonObj
│   │   │               │   │   └── JRepo.java
│   │   │               │   └── scrap.java
│   │   │               └── ssm
│   │   │                   ├── controller
│   │   │                   │   └── mainController.java
│   │   │                   ├── dao
│   │   │                   │   └── GithubDao.java
│   │   │                   ├── pojo
│   │   │                   │   ├── Commit.java
│   │   │                   │   ├── frameData.java
│   │   │                   │   └── Repo.java
│   │   │                   ├── sendData
│   │   │                   │   ├── cloudData.java
│   │   │                   │   ├── columnData.java
│   │   │                   │   ├── dynamicData.java
│   │   │                   │   ├── lineChartData.java
│   │   │                   │   ├── pieData.java
│   │   │                   │   └── tableData.java
│   │   │                   ├── services
│   │   │                   │   ├── GitHubService.java
│   │   │                   │   └── impl
│   │   │                   │       └── GitHubServiceImpl.java
│   │   │                   └── SsmApplication.java
│   │   └── resources
│   │       ├── application.yml
│   │       ├── config.properties
│   │       ├── static
│   │       └── templates
│   └── test
│       └── java
│           └── edu
│               └── sustech
│                   └── backend
│                       └── BackendApplicationTests.java
└── target
    ├── classes
    │   ├── application.yml
    │   ├── config.properties
    │   └── edu
    │       └── sustech
    │           └── backend
    │               ├── scraper
    │               │   ├── DataBaseController.class
    │               │   ├── gitHub$Info.class
    │               │   ├── gitHub.class
    │               │   ├── github_scraper.class
    │               │   ├── jsonObj
    |                   |    |    └── JRepo.class
    |                   |    └── scrap.class
    |                   └── ssm
    |                        ├── controller
    |                        |    └── mainController.class
    |                        ├── dao
    |                        |    └── GithubDao.class
    |                        ├── pojo
    |                        |    ├── Commit.class
    |                        |    ├── frameData.class
    |                        |    └── Repo.class
    |                        ├── sendData
    |                        |    ├── columnData.class
    |                        |    ├── lineChartData.class
    |                        |    └── tableData.class
    |                        ├── services
    |                        |    ├── GitHubService.class
    |                        |    └── impl
    |                        |         └── GitHubServiceImpl.class
    |                        └── SsmApplication.class
```

Important Classes, Methods and Fields

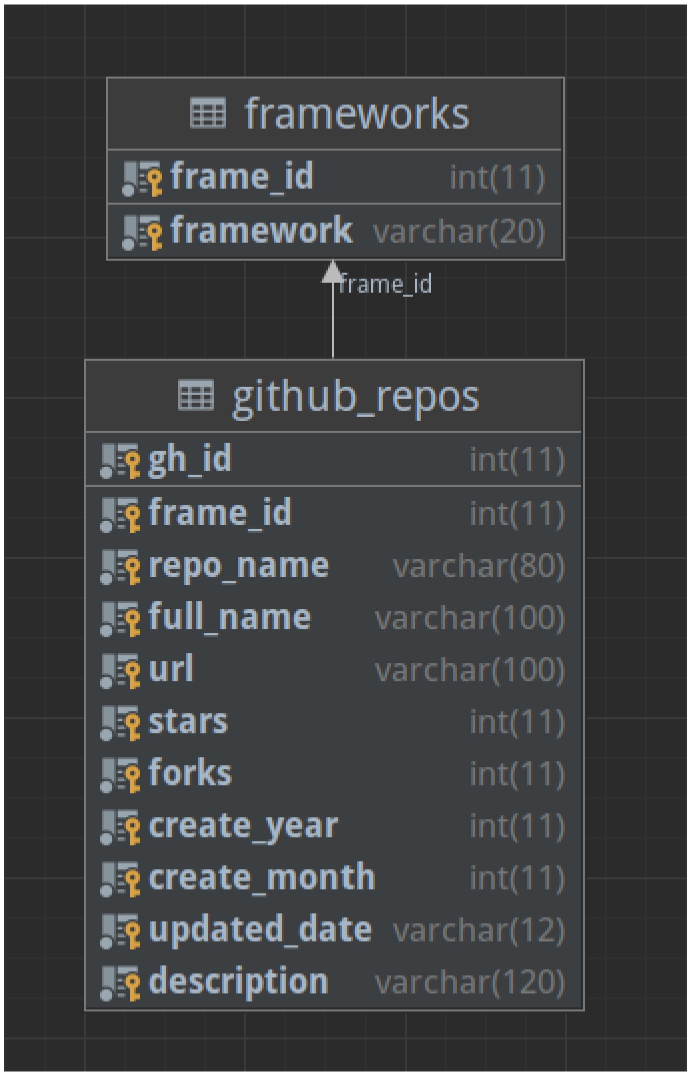
MainController

```
@CrossOrigin
@RestController
@RequestMapping("/data")
public class mainController {

  @Autowired
  private GitHubService gitHubService;

  @GetMapping("/{id}")
  public String getById(@PathVariable Integer id) {
    System.out.println(id);
    return "hello!";
  }

  @GetMapping("/line")
  public String getLineChart() {
    System.out.println("line");
    return gitHubService.sendLineChart();
  }

  @GetMapping("/table")
  public String getTable() {
    System.out.println("table");
    return gitHubService.sendTable();
  }

  @GetMapping("/col")
  public String getCol() {
    System.out.println("col");
    return gitHubService.sendColumn();
  }
}
```

Dao

```
@Mapper
public interface GithubDao {

    @Select("select * from github_repos")
    public List<Repo> getAllRepos();

    @Select("select * from github_repos where create_year=#{year} and 
frame_id=#{frame}")
    public List<Repo> getReposByYearAndFrame(Integer year, Integer frame);

    @Select("select * from github_repos where create_year=#{year} ")
    public List<Repo> getReposByYear(Integer year);

    @Select("select * from github_repos where frame_id=#{frame} ")
    public List<Repo> getRepoByFrame(Integer frmae);

}
```

数据库设计



与爬虫的配合：

```
public class DataBaseController {
    private DataSource dataSource;
    private int repoCnt = 0;
    private Map<String ,Integer> frameworkMap;

    //...

    public void insertRepo(JRepo repo) {
        try (Connection connec = dataSource.getConnection()) {

            String sql = "insert into github_repos (repo_name, full_name,
url, stars, forks, create_year, create_month, updated_date, description,
frame_id) " +
                    "values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

            PreparedStatement pstm = connec.prepareStatement(sql);
            pstm.setString(1, repo.getName());
            pstm.setString(2, repo.getFull_name());
            pstm.setString(3, repo.getHtml_url());
            pstm.setInt(4, repo.getStargazers_count());
            pstm.setInt(5, repo.getForks_count());
            String create_date = repo.getCreated_at();
            String update_date = repo.getUpdated_at();
            pstm.setInt(6,
Integer.parseInt(dateConvert(create_date).substring(0,4)));
            pstm.setInt(7,
Integer.parseInt(dateConvert(create_date).substring(5)));
            pstm.setString(8, dateConvert(update_date));
            String description = repo.getDescription();
            if (description != null && description.length() >= 100)
                description = description.substring(0,100);
            pstm.setString(9, description);
            pstm.setInt(10, frameworkMap.get(repo.getFramework()));

            pstm.execute();
            repoCnt++;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
}
```

scrapper

```
public class github_scraper {

    //...

  public void repoScrapeByTime(String framework) {

    int cnt = 0;
    int total_cnt = 1;
    int limit = (framework.equals("Spring") ? 50000 : 20000);
    String created = "2012-01-01T00:00:00Z";
    String last = null;
    URL url = null;
    do {
      for (int i = 1; i <= 10; i++) {
        String s = String.format("https://api.github.com/search/repositories"
+
                "?q=" + framework + "+language:java+created:>" + created +
                "&sort=created&order=asc&per_page=100&page=" + i);
        try {
          url = new URL(s);
          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
          conn.setRequestMethod("GET");

          byte[] encodedAuth =
Base64.encodeBase64(token.getBytes(StandardCharsets.UTF_8));
          String authHeaderValue = "Basic " + new String(encodedAuth);
          conn.setRequestProperty("Authorization", authHeaderValue);

          conn.connect();

          StringBuilder content = new StringBuilder("");
          BufferedReader bi = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
          String line = null;
          while ((line = bi.readLine()) != null) {
            content.append(line);
          }

          JsonObject result =
JsonParser.parseString(content.toString()).getAsJsonObject();
          total_cnt = result.get("total_count").getAsInt();

          int page_nums = 0;
          for (JsonElement item : result.get("items").getAsJsonArray()) {
            JRepo repo = gson.fromJson(item, JRepo.class);
            repo.setFramework(framework);
            controller.insertRepo(repo);
            last = repo.getCreated_at();
            cnt++;
            page_nums++;
          }
          controller.printCnt();
          if (page_nums < 100 || i == 10) {
            created = last;
            break;
          }

        } catch (Exception e) {
          e.printStackTrace();
        }
      }
      // scrap by time
      if (framework.equals("Spring")) created = timeAdd(created, 1);
      if (framework.equals("Spark")) created = timeAdd(created, 2);
```

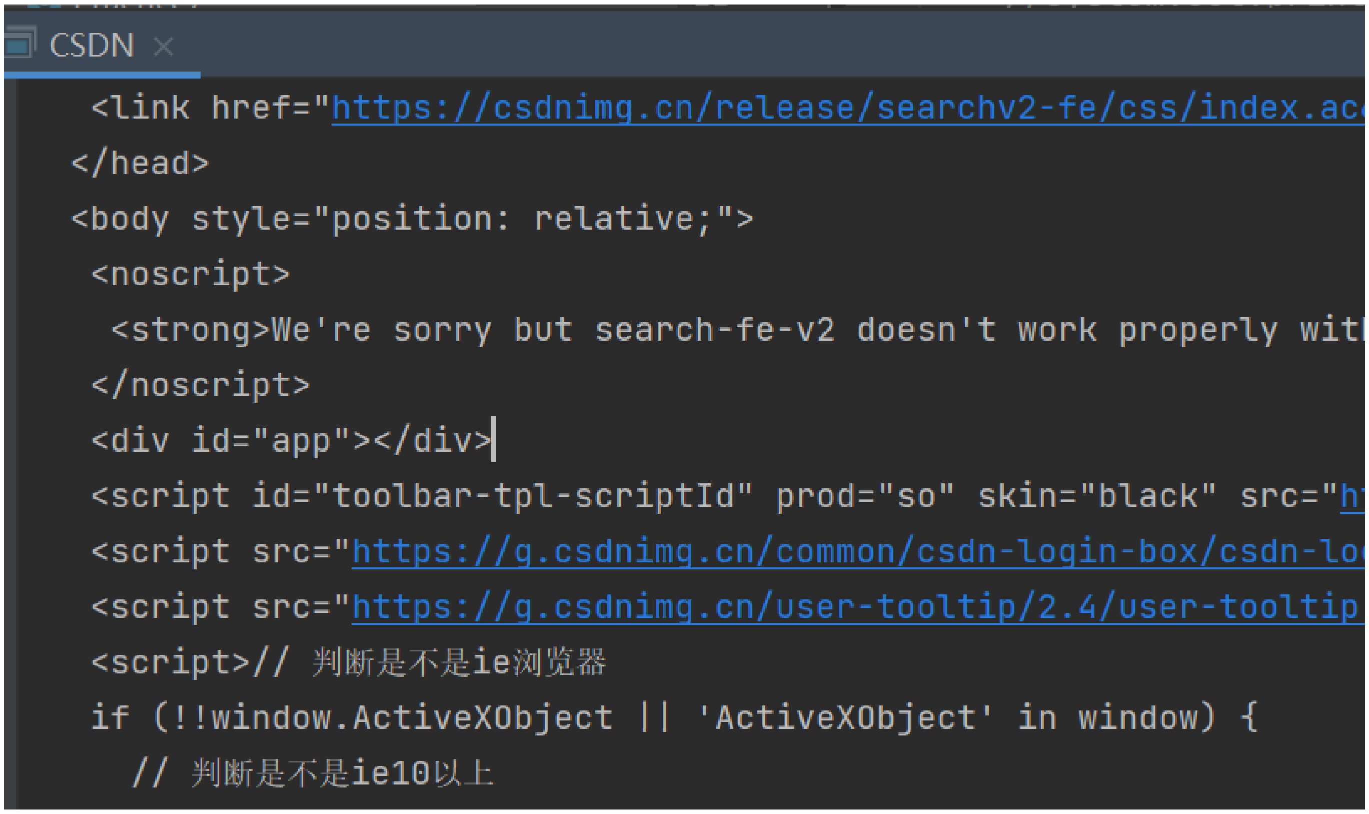
```
        } while ((cnt < total_cnt) && (cnt < limit) && (timeCheck(created)));
      }
    }
```

旧思路

github search 访问限制

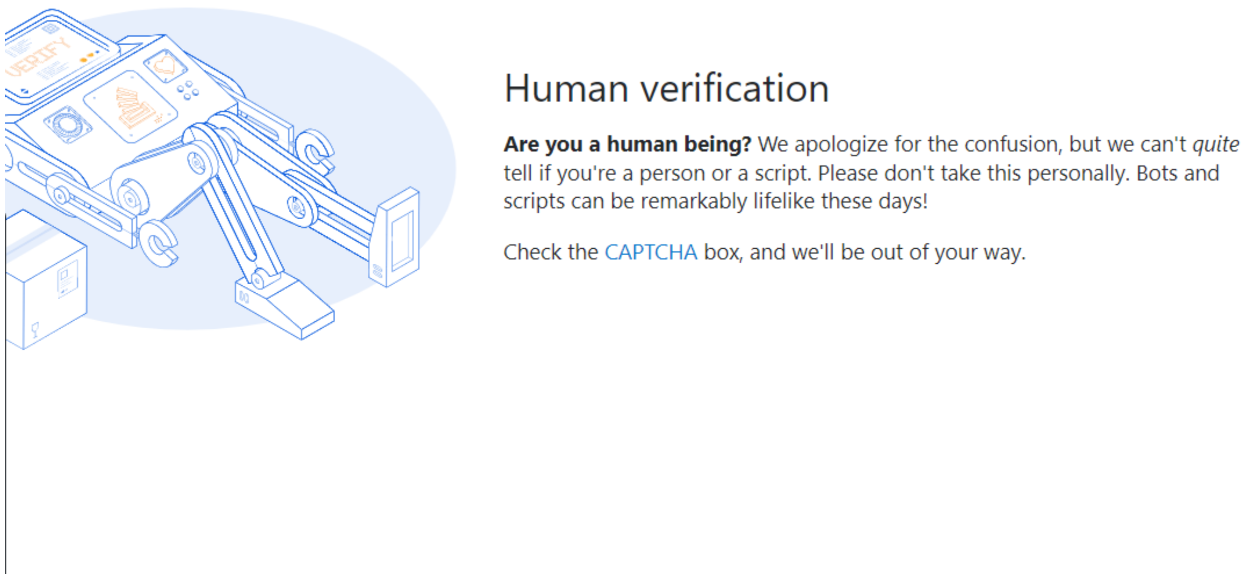
https://docs.github.com/en/rest/search#about-the-search-api

原本的打算是分别挖掘CSDN、知乎、GitHub和Stack Overflow等平台的关于Java Spring的相关数据。可是遇到了jsoup无法执行JavaScript，我们所访问的网页并不包含我们所需要的div数据，所以CSDN和知乎的数据无法爬取。



```
    <link href="https://csdnimg.cn/release/searchv2-fe/css/index.ac
  </head>
  <body style="position: relative;">
    <noscript>
      <strong>We're sorry but search-fe-v2 doesn't work properly wit
    </noscript>
    <div id="app"></div>
    <script id="toolbar-tpl-scriptId" prod="so" skin="black" src="h
    <script src="https://g.csdnimg.cn/common/csdn-login-box/csdn-lo
    <script src="https://g.csdnimg.cn/user-tooltip/2.4/user-tooltip
    <script>// 判断是不是ie浏览器
    if (!!window.ActiveXObject || 'ActiveXObject' in window) {
      // 判断是不是ie10以上
```

可以看到这里的document中不存在想要的#app的子元素。
另一方面，stack overflow有着captcha（全自动区分计算机和人类的图林测试），脚本依然无法访问。

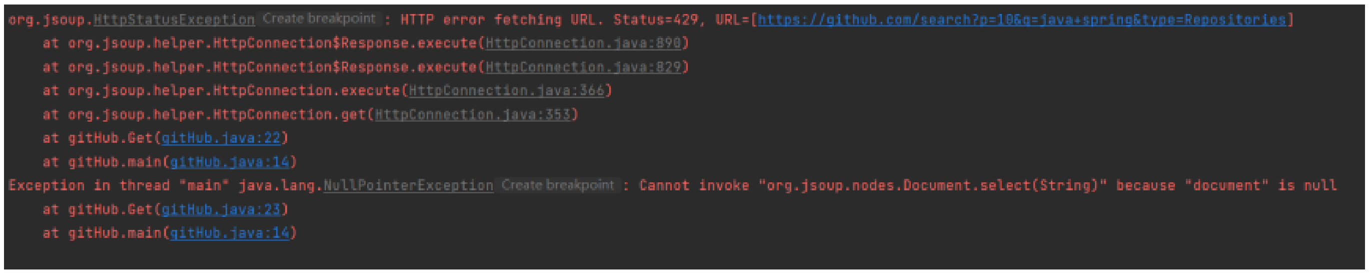


Human verification

Are you a human being? We apologize for the confusion, but we can't *quite* tell if you're a person or a script. Please don't take this personally. Bots and scripts can be remarkably lifelike these days!

Check the CAPTCHA box, and we'll be out of your way.

所以，我们爬取了再GitHub上的关于Java Spring的相关数据。在GitHub主页搜索“Java Spring”词条，可以搜索到100页每页10份的相关结果。我们爬取这些搜索结果的被浏览量和最后更新日期，从而得到1000份相关数据。
由于有100页的搜索结果，爬取过程需要访问近似网页100次。为了保证爬虫的礼貌性，我们设置了每次访问后100毫秒的延迟时间。但是这在访问第十次网页的时候还是跳出了HTTP 429的请求过多错误。



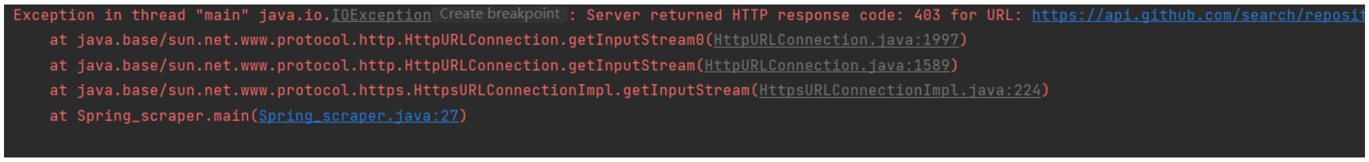
```
org.jsoup.HttpStatusException Create breakpoint : HTTP error fetching URL. Status=429, URL=[https://github.com/search?p=10&q=java+spring&type=Repositories]
    at org.jsoup.helper.HttpConnection$Response.execute(HttpConnection.java:890)
    at org.jsoup.helper.HttpConnection$Response.execute(HttpConnection.java:829)
    at org.jsoup.helper.HttpConnection.execute(HttpConnection.java:366)
    at org.jsoup.helper.HttpConnection.get(HttpConnection.java:353)
    at gitHub.Get(gitHub.java:22)
    at gitHub.main(gitHub.java:14)
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "org.jsoup.nodes.Document.select(String)" because "document" is null
    at gitHub.Get(gitHub.java:23)
    at gitHub.main(gitHub.java:14)
```

发现短时间内只能连续访问九次，所以我们又让脚本在每访问九次之后做一个10000毫秒的延迟，也就是十秒。发现还是无法第十次访问网页。所以我们将延迟扩大至60秒，发现能够正常运行，直至第四十六次访问网页，再一次跳出了HTTP 429的错误。为了能够稳定获取数据，我们最后将延迟时间调整至120秒即120000毫秒之后，脚本方能正常运行，得到1000条目标数据。整个脚本运行时间大概在23分钟左右。

我们分别爬取了在GitHub上搜索词条“Java Spring”、“Spring boot”、“Spring MVC”、“Spring Cloud”、“Spring Data”、“Spring Security”和“Spring Batch”。每个词条分别获取1000条相关数据，来存入后端的数据库。

之后，我们尝试通过GET的请求方法，可以一次获取所有的网页信息。并通过设置per_page=100(最多为100)，让我们可以在一次访问中获取100条目标信息。相比之前的每次访

问中获取10条数据，效率提高了很多。但是该方法在短时间内多次对网页进行访问，还是会有HTTP 403禁止访问的报错现象。



```
Exception in thread "main" java.io.IOException Create breakpoint : Server returned HTTP response code: 403 for URL: https://api.github.com/search/reposit
    at java.base/sun.net.www.protocol.http.HttpURLConnection.getInputStream0(HttpURLConnection.java:1997)
    at java.base/sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1589)
    at java.base/sun.net.www.protocol.https.HttpsURLConnectionImpl.getInputStream(HttpsURLConnectionImpl.java:224)
    at Spring_scraper.main(Spring_scraper.java:27)
```

获取的所有数据通过Json格式传输至数据库。

Frontend

The frontend is build based on Vue framework.

The frontend resource is packed onto an ISS server, thus the website can be browsed on the browser directly and there is no need of running the whole Vue project.

The interactive mode, some picture resources and all of bgm resources refers to the game
Needy Girl Overdose.

The live2d character called Histoire is created artificially by Cubism, who is a character of the game *Hyperdimention Neptunia: Rebirth 3*.
The basic motion such as breathing and looking at the mouse is involved.
There is also a demo character model of official.

The implementation of all kinds of charts refers to the echart API.

acknowedgement

Official documents including echarts, Cubism, Vue, Spring, MyBatis and other CSDN articles.