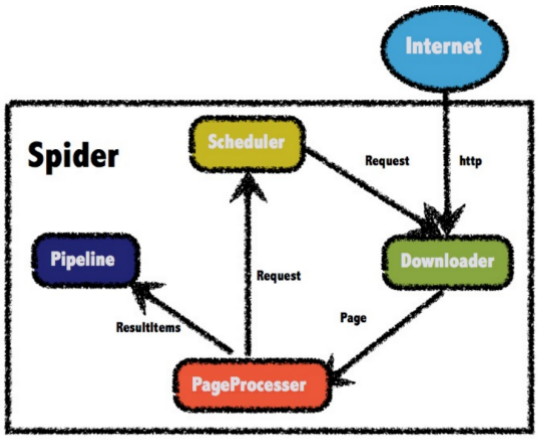
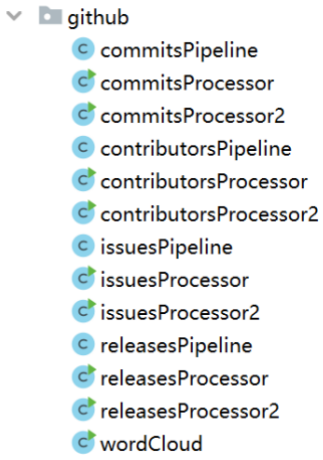
CS209 Final Project

组员：12011904周子懿　12012221李建霖

数据爬取

- 爬虫框架：WebMagic





- 简要流程：
  - 首先自定义Site，伪造Domain以及UserAgent信息，在头部添加Token以实现爬虫过程的稳定性。

```
@Override
public Site getSite() {
    return Site.me()
            .setDomain("blog.sina.com.cn")
            .addHeader("Authorization", "Bearer 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")
            .setUserAgent(
                    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/537.31 (KHTML, like G
            .setCharset("UTF-8");
            }
```

  - Processor通过设置好的Site访问指定url，将爬取到的数据通过JsonPathSelector获取想要的键值List。

```
@Override
public void process(Page page) {
    String str = page.getHtml().regex("\\[.*\\]").toString();
    List<String> id = new JsonPathSelector( jsonPathStr: "$.*.id").selectList(str);
    List<String> name = new JsonPathSelector( jsonPathStr: "$.*.login").selectList(str);
    List<String> num = new JsonPathSelector( jsonPathStr: "$.*.contributions").selectList(str);

    List<Developer> list = new ArrayList<>();
    for(int i = 0; i < id.size(); i++){
        Developer developer = new Developer();
        developer.setId(Long.parseLong(id.get(i)));
        developer.setName(name.get(i));
        developer.setCommit_num(Integer.parseInt(num.get(i)));
        developer.setRepoName("apollo");
        list.add(developer);
    }

    page.putField( key: "developers", list);
}
```

- 处理后的数据通过Pipeline处理，在Pipeline中自定义数据库的导入方法，重写process方法实现数据导入。
- 使用 **JDBC、Druid数据库连接池、数据库批处理机制**构建数据库导入方法。

```
14 usages
public class Druid {

    10 usages
    public static DruidDataSource dataSource;

    //1.初始化Druid连接池
    static {...}

    //2.获取连接
    5 usages
    public static Connection getConnection() {...}

    5 usages
    public static void closeAll(Connection connection) {...}

    @Test
    public void DruidTest() {...}


}
```

- 通过对不同对象类型自定义Pipeline，以及使用数据库连接池，将对应数据导入数据库中。

```
@Override
public void process(ResultItems resultItems, Task task) {
    balanceChangeLock.lock();
    try {
        openDB( dbname: "cs209_project");
        List<Commit> commits = resultItems.get("commits");
        commits.stream().forEach(o -> load_Githubrepo(o));
        try {
            githubRepo_stmt.executeBatch();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        commit();
        closeDB();
    } finally {
        balanceChangeLock.unlock();
    }

}
```

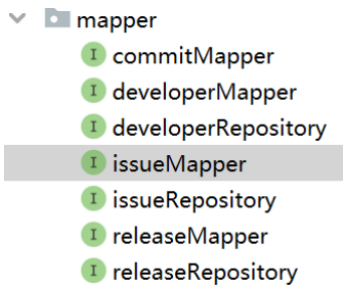
后端

- 项目框架：SpringBoot
- 数据持久层框架：Mybatis-plus

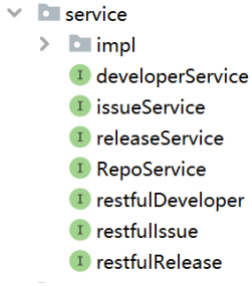
1. 实体类
    - 每个实体类中均包含项目要求的所有信息。
    - 以Commit为例：包含联合主键id、repoName以及时间time。


```
public class Commit {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    2 usages
    private Date time;
    2 usages
    private String repoName;
```

2. Mapper类



3. Service类以及ServiceImpl实现类：具体方法的实现



   - 以issueService为例，具体实现了open、close数量统计、解决时间的均值方差计算以及description的关键词提取等功能。

```
4 usages  1 implementation
public interface issueService extends IService<Issue> {

    2 usages  1 implementation
    public List<Long> getIssueStatus(String name);

    2 usages  1 implementation
    public List<Double> getValues(String name);

    1 usage  1 implementation
    public List<String> getDescriptionKey(String name);

}
```

4. Controller类：前后端接口

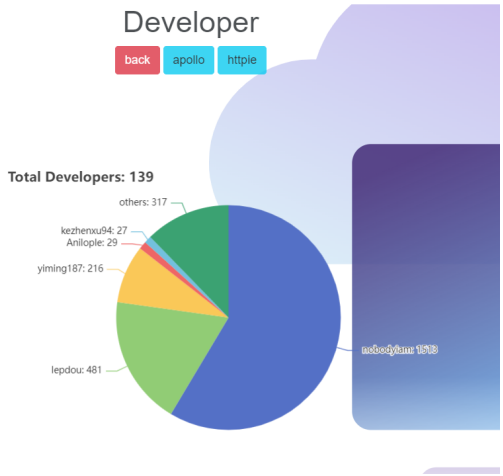
   - 以ReleaseController为例，前端通过指定url调用getTotal方法，之后通过Service中的具体方法，将所需数据以JsonResult格式返回给前端。

```
@GetMapping(value = "/release/total")
public JsonResult getTotal(@RequestParam(name = "name", required = true) String name) {
  List<Long> data = releaseService.getTotal(name);
  System.out.println(data);
  return JsonResult.buildSuccess(data);
}
```

前端

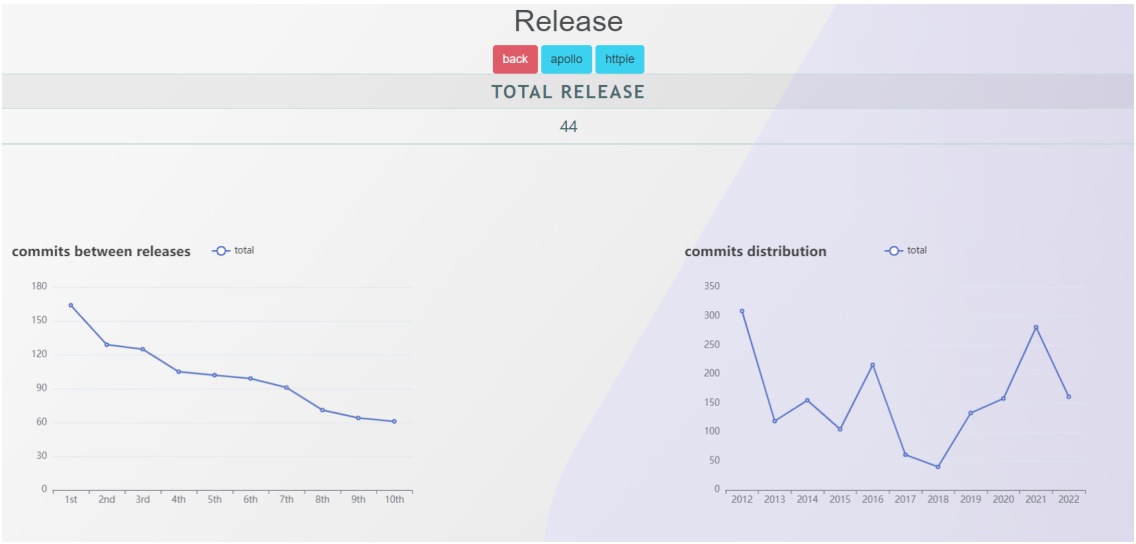
1. 使用Thymeleaf模板引擎 动态渲染网页内容
   - 与Controller进行交互，通过model.addattribute()实现前后端传输数据交互
   - 与JQuery、Bootstrap进行交互，实现前端页面美化及动态效果
2. 前端UI框架
   - 使用Echarts实现数据可视化
3. 前端效果展示

- Developer



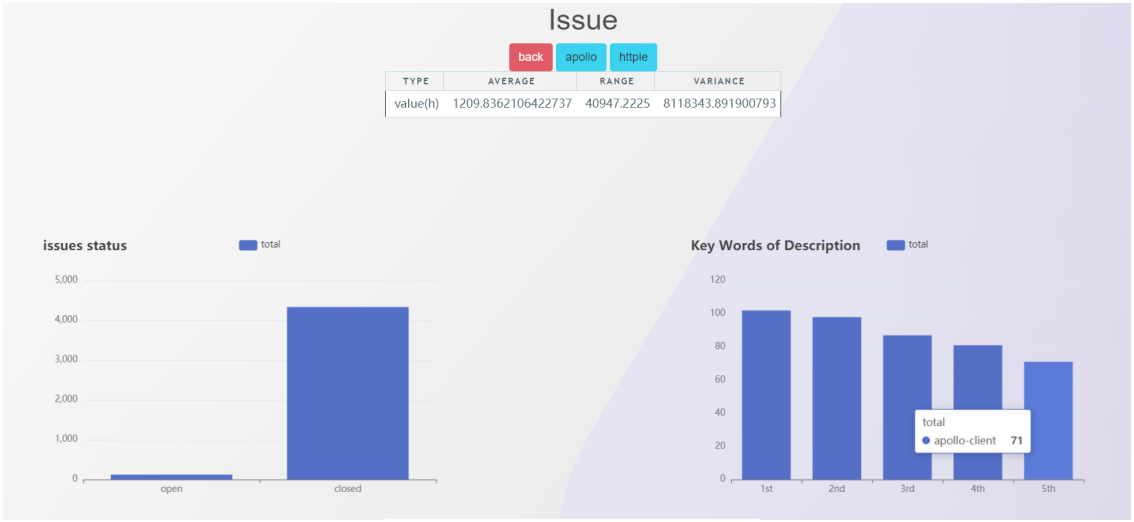
可通过按钮切换展示的Repository，首先展示项目中developer总数，再通过饼状图展示commit数量最高的几位developer信息，美观且明了。

- Release



首先展示release的总数，接着通过折线图分别展示每两个release之间的commit数量以及commit的分布情况。

- Issue



上方的图表展示的issue解决时间的平均值，极值差，方差等信息。左侧图表展示了不同issue状态的数量，分别为open和closed。右侧图表展示了issue描述里排名前五的关键词。将鼠标放置在柱状图上，即可观察到key words的具体内容。

此图为根据issue所有的title信息分析出来的词云，通过**kennycason**库得出的可视化结果。

REST services

结构：

- Repository：提供JPA封装好的接口
- Domain：数据库映射的实体类
- Service：实现CRUD逻辑
- Controller：将对象返回

通过Hibernate获取数据库中Developer、Release、Issue的对应信息，并通过RestController返回作为RESTful API endpoint。因为爬虫已将json信息进行处理并将对应属性存入数据库，通过RestController返回的信息都是直接可用信息。