

实时采集日志并显示到浏览器端

- 1 整体技术路线
- 2 前端页面
- 3 Server 接收浏览器请求并返回数据
- 4 Agent 按行读取日志信息
- 5 log4j 配置系统日志信息
- 6 结果展示

1 整体技术路线

项目使用的框架是 spring+springmvc;

整体技术路线: 在前端轮询向服务端发送 ajax 请求, 服务端 Server 接收到请求之后调用 Agent 读取系统日志, Server 将日志信息以 json 字符串格式返回, ajax 利用回调处理将返回的日志信息显示在前端页面上。

系统日志信息的记录: 采用 log4j 配置文件记录系统运行信息和错误信息。

2 前端页面

2.1 前端页面代码和视图

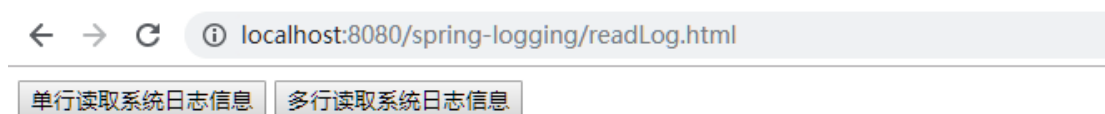
```

<body>
  <!-- 按钮区-->
  <div>
    <input type="button" id="readLine" value='单行读取系统日志信息' />
    <input type="button" id="readLines" value='多行读取系统日志信息' />
  </div>
  <!-- 日志信息显示区 -->
  <div id="log" >

  </div>
</body>

```

前端代码



前端显示

功能说明：点击“单行读取系统日志信息”或“多行读取日志信息”触发 ajax 轮询请求，即可按单行读取日志信息或多行读取日志信息。

2.2 ajax 轮询请求日志信息

点击相应的按钮触发请求事件,利用 **window** 设置循环发送请求,请求间隔设置成 1 秒,也就是说每隔 1 秒读取系统日志一行或者多行,当日志信息读取完成之后结束循环。设置 **ajax** 请求地址 **url**,请求方式 **type** 为 **psot**,请求参数 **data** 为日志行数 **row** 以及每次读 **line** 行,请求返回值类型 **dataType** 设置成 **json** 类型,回调函数 **success** 将返回的日志信息追到到日志信息显示区,并且实现日志的分页显示。代码如下。

```
<script type="text/javascript">
var i=0;
var line=0;
$(function(){
    $("#readLine").click(function(){
        line=1;//每次读取一行日志信息
        c = window.setInterval("getLog()",1000); //间隔1秒去触发ajax
    });
    $("#readLines").click(function(){
        line=2;//每次读取两行日志信息
        c = window.setInterval("getLog()",1000); //间隔1秒去触发ajax
    });
});
function getLog(){
    $.ajax({
        url:"http://localhost:8080/spring-logging/log/logging.do",
        type:"post",
        data:{"row":i,"line":line},//从第i行开始读日志，每次读line行
        dataType:"json",
        success:function(result){
            if(result.status==0){
                $("#log").append("读取日志信息完成!!!");
                window.clearInterval(c); //关闭ajax轮询请求
            }
            if(i%16==0){
                //日志信息分页显示
                $("#log").html("第"+(i+16)/16+"页日志信息....."+"<br />")
            }
            i=i+line;
            $("#log").append(result.data); //追加读取到的日志信息
        },
        error:function(){
            window.clearInterval(c); //获取日志失败，结束请求
        }
    });
};
```

3 Server 接收浏览器请求并返回数据

Server 类负责接收前端请求以及请求参数数，并且调用 **Agent** 的 **readLog** 方法读取日志信息，并将结果存入 **Result** 类并以 json 字符串形式返回。

```
@Controller
@RequestMapping("/log")
public class Server {
    @RequestMapping("/logging.do")
    @ResponseBody //以json对象返回
    public Result<String> showLog(HttpServletRequest request) throws IOException {
        Agent agent=new Agent();
        String row=request.getParameter("row");
        String line=request.getParameter("line");
        //String ua = request.getHeader("User-Agent");//获取浏览器ua信息
        //agent.wiiteLog(ua);//将浏览器信息记录到日志中
        //读取系统日志信息
        Result<String> result=null;
        result=agent.readLog(Integer.parseInt(row),Integer.parseInt(line));
        return result; //将结果返回给前端页面
    }
}
```

4 Agent 按行读取日志信息

Agent 类的 readLog 函数通过 BufferRead 完成按行读取日志信息的功能。

```
public class Agent {
    private static Logger logger = Logger.getLogger(Server.class);

    //读取系统日志
    public static Result<String> readLog(int row,int line) throws IOException {
        Result<String> result=new Result<String>();
        FileReader fr=new FileReader ("G://logs/info.log");
        BufferedReader bufr=new BufferedReader (fr);
        StringBuffer buf=new StringBuffer();
        while(row>0) {
            bufr.readLine();
            row--;
        }
        String str=bufr.readLine();
        if(str==null) {
            result.setStatus(0);
        }else {
            buf.append(str+"<br / >");
            while(line>1) {
                buf.append(bufr.readLine()+"<br / >");
                line--;
            }
            result.setData(buf.toString());
            result.setStatus(1);
        }
        bufr.close();
        return result;
    }
}
```

5 log4j 配置系统日志信息

配置系统日志信息文件 log4j.properties，分别将系统运行 info 日志信息以及 error 日志信息保存到 info.log 以及 error.log 文件中，分别配置日期显示格式以及输出信息类型和换行标志符。

```

1### 设置日志输出级别为info###
2log4j.rootLogger = info,stdout,D,E
3
4### 日志在控制台输出 ###
5log4j.appender.stdout = org.apache.log4j.ConsoleAppender
6log4j.appender.stdout.Target = System.out
7log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
8log4j.appender.stdout.layout.ConversionPattern = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%n
9
10### info日志信息保存到G://logs/error.log ###
11log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
12log4j.appender.D.File = G://logs/info.log
13log4j.appender.D.Append = true
14log4j.appender.D.Threshold = INFO
15log4j.appender.D.layout = org.apache.log4j.PatternLayout
16log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %p ] %m%n
17
18### error日志信息保存到G://logs/error.log ###
19log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
20log4j.appender.E.File = G://logs/error.log
21log4j.appender.E.Append = true
22log4j.appender.E.Threshold = ERROR
23log4j.appender.E.layout = org.apache.log4j.PatternLayout
24log4j.appender.E.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %p ] %m%n

```

writeLog 函数负责记录系统运行时的 info 日志。

```

//记录系统日志
public static void writeLog(String str){
    // 记录info级别的信息
    logger.info("浏览器User Agent信息: "+str);
}

```

6 结果展示

前段获取日志信息图如下：

← → ↻ ⓘ localhost:8080/spring-logging/readLog.html ☆ ⓘ

单行读取系统日志信息 多行读取系统日志信息

第1页日志信息.....

2019-07-11 17:41:45 [INFO] FrameworkServlet 'mvc': initialization started

2019-07-11 17:41:46 [INFO] Refreshing WebApplicationContext for namespace 'mvc-servlet': startup date [Thu Jul 11 17:41:46 CST 2019]; root of context hierarchy

2019-07-11 17:41:46 [INFO] Loading XML bean definitions from file [D:\apache-tomcat-7.0.94\wtpwebapps\spring-logging\WEB-INF\classes\conf\spring-mvc.xml]

2019-07-11 17:41:46 [INFO] Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@5b70a0e6: defining beans [server,org.springframework.context.annotation.internalConfigurationAnnotationProcessor,org.springframework.context.annotation.root of factory hierarchy

2019-07-11 17:41:46 [INFO] Mapped "{[/log/logging.do],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public entity.Result controller.Server.showLog(javax.servlet.http.HttpServletRequest) throws java.io.IOException

2019-07-11 17:41:47 [INFO] FrameworkServlet 'mvc': initialization completed in 1106 ms

读取日志信息完成!!!