《JavaEE 平台技术》实验报告

实验名称:	实验一：SpringMVC 合法性检查的效率
实验日期:	2021 年 10 月 7 日星期四
实验地点:	宿舍
提交日期:	2021 年 10 月 8 日星期五

组号:	1-07
组名:	这队更是重量级
专业年级:	软件工程 2019 级
学年学期:	2021-2022 学年上学期

一、 实验目的

1、掌握 SpringMVC 合法性检查的实现方式

2、验证 SpringMVC 中的合法性检查的效率
3、掌握使用 JMeter 测试 RESTful API 的方法

二、 实验环境

1、服务器 A: Ubuntu 18.04 服务器 2 核 4G 内存虚拟机一台，图形界面，安装 JDK 11， Maven、git
2、服务器 B：Ubuntu 18.04 服务器 2 核 2G 内存虚拟机一台，命令行界面，安装 JDK 11，Maven、git，JMeter 5.4.1

三、 实验内容及要求

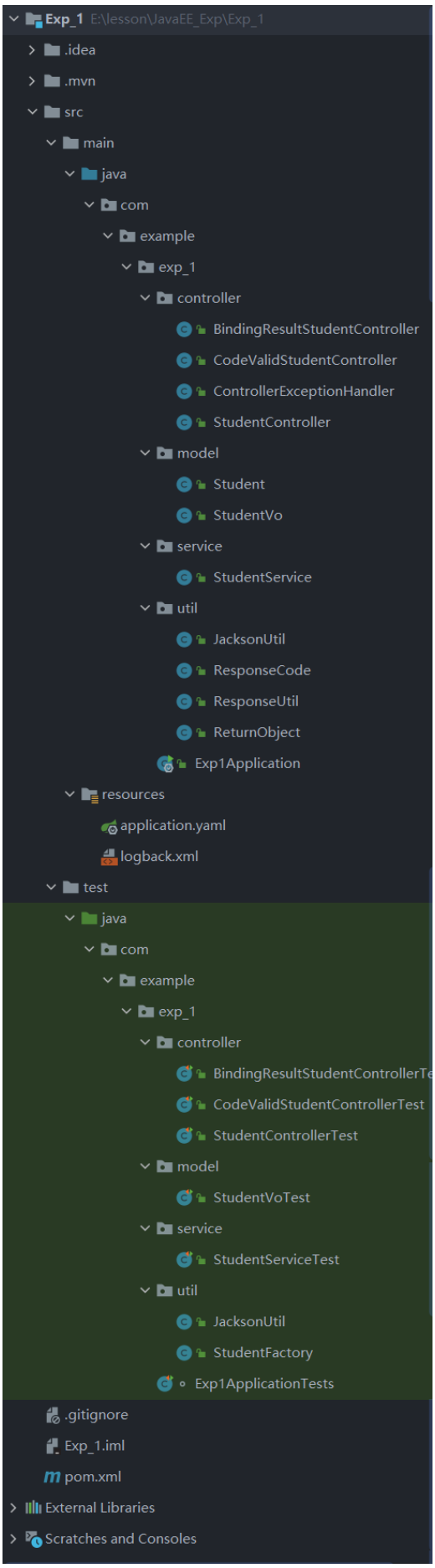
在 SpringMVC 中，合法性检查可以通过抛出异常或 BindingResult 方式的实现。在阿里规约里规定了慎用 Exception，因为在 Java 中异常是十分低效的。设计一个实验对比通过异常、BindingResult 和完全手写合法性检查三种方式的运行效率

四、 Git 地址

https://github.com/529106896/HeavyTeam.git
本次实验位于 HeavyTeam/Experiment/Exp_1 目录下

五、 实验报告

1. 实验项目结构及相关说明

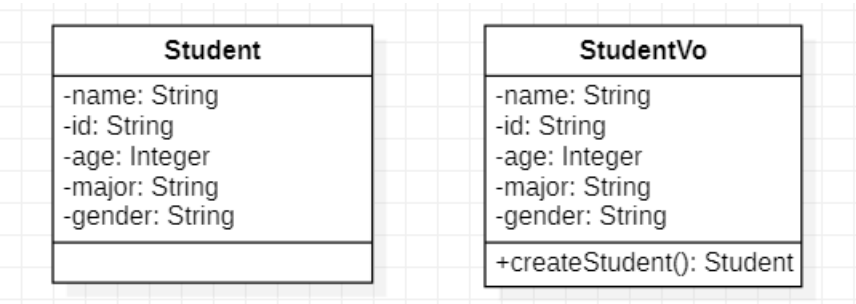

Exp_1 E:\lesson\JavaEE_Exp\Exp_1
.idea
.mvn
src
main
java
com
example
exp_1
controller
BindingResultStudentController
CodeValidStudentController
ControllerExceptionHandler
StudentController
model
Student
StudentVo
service
StudentService
util
JacksonUtil
ResponseCode
ResponseUtil
ReturnObject
Exp1Application
resources
application.yaml
logback.xml
test
java
com
example
exp_1
controller
BindingResultStudentControllerTe
CodeValidStudentControllerTest
StudentControllerTest
model
StudentVoTest
service
StudentServiceTest
util
JacksonUtil
StudentFactory
Exp1ApplicationTests
.gitignore
Exp_1.iml
pom.xml
External Libraries
Scratches and Consoles

说明：

因为实验只要求进行合法性检测，所以我们不需要进行过多分层。

本次实验中，我们设计了一个对象模型——Student 类，以及用于接收前端数据的 StudentVo 类

在 StudentVo 类中，我们设置 name 属性为@NotBlank，之后我们的实验也主要围绕 name 属性进行异常检测效率的对比



在 Service 层，我们只涉及了三个方法：根据姓名查找学生、根据姓名创建学生、根据前端返回数据组建的 VO 对象创建学生，其中第三个为我们主要用于测试的方法

```java
@Service
public class StudentService {

    public Student searchByName(String name){
        Student student = createStudent(name);
        return student;
    }

    private Student createStudent(String name){
        Student s = new Student();
        s.setName(name);
        s.setId("11920192203642");
        s.setGender("男");
        s.setAge(20);
        s.setMajor("软件工程");
        return s;
    }

    /**
     * 新增学生
     * @param studentVo 新学生信息
     * @return 新学生
     */
    public Student createStudent(StudentVo studentVo)
    {
        Student student = studentVo.createStudent();
        student.setMajor("软件工程");
        return student;
    }
}
```

在 Controller 层，我们定义了三个 Controller 和一个 ControllerExceptionHandler，三个 Controller 主要的方法均为根据前端返回的 json 字符串创建学生，发现异常数据则分别抛出异常并交由 ControllerExceptionHandler 处理、手动处理以及利用 BindingResult 自动处理

三个 Controller 对应的 url 分别为：post /student、post /codevalid/student、post /bindingresult/student

在 util 包中，我们设计了四个实用工具类，分别负责对象与 JSON 字符串的转换、返回的操作码、响应操作结果、组建返回对象

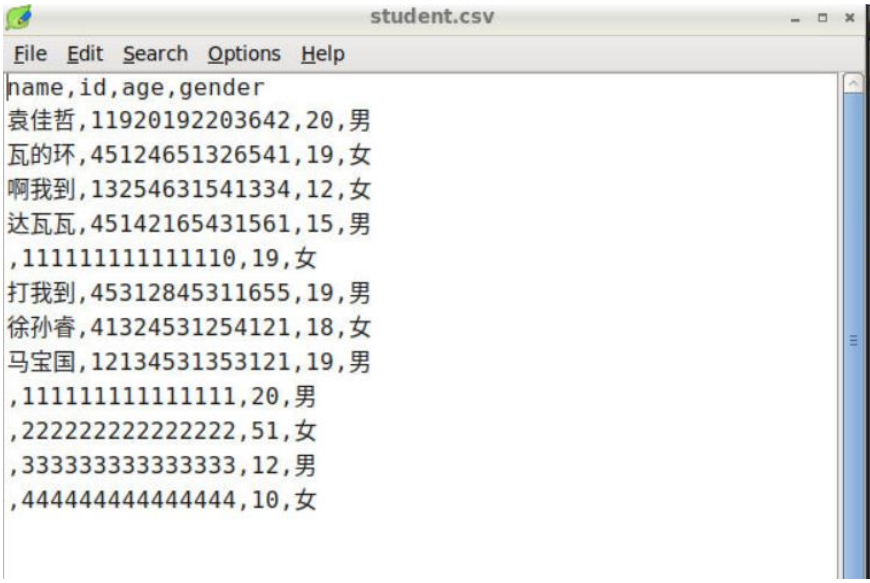
2. 实验设计原理：

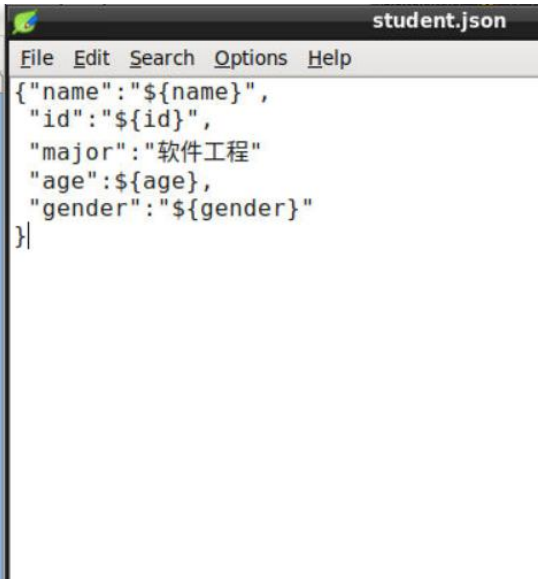
在普通的 StudentController 中，我们在 createStudent 的参数前加上@Validated，如果前端接收到一个 name 属性为空的 json，根据@NotBlank 注解的作用，则会抛出 MethodArgumentNotValidException，这个 Exception 会在 ControllerExceptionHandler 进行后续处理

在 CodeValidStudentController 中，我们不使用@Validated，而是进行手动合法性检查，异常数据条件为 studentVo.getName() == null || studentVo.getName().equals("") || studentVo.getName().trim().length() == 0

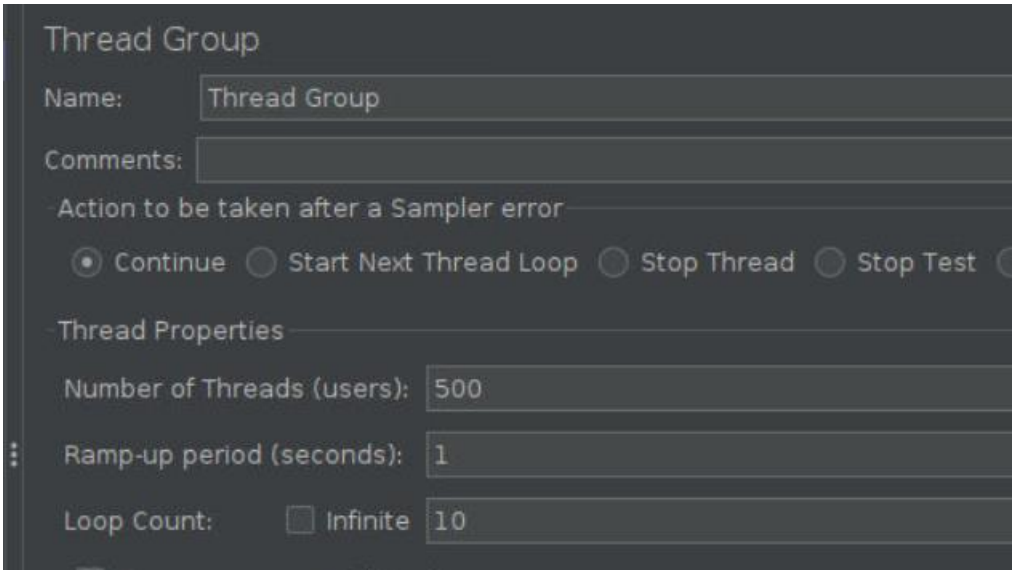
在 BindingResultStudentController 中，我们使用@Validated 和 BindingResult 进行自动合法性检查与处理

3. 测试数据与方法





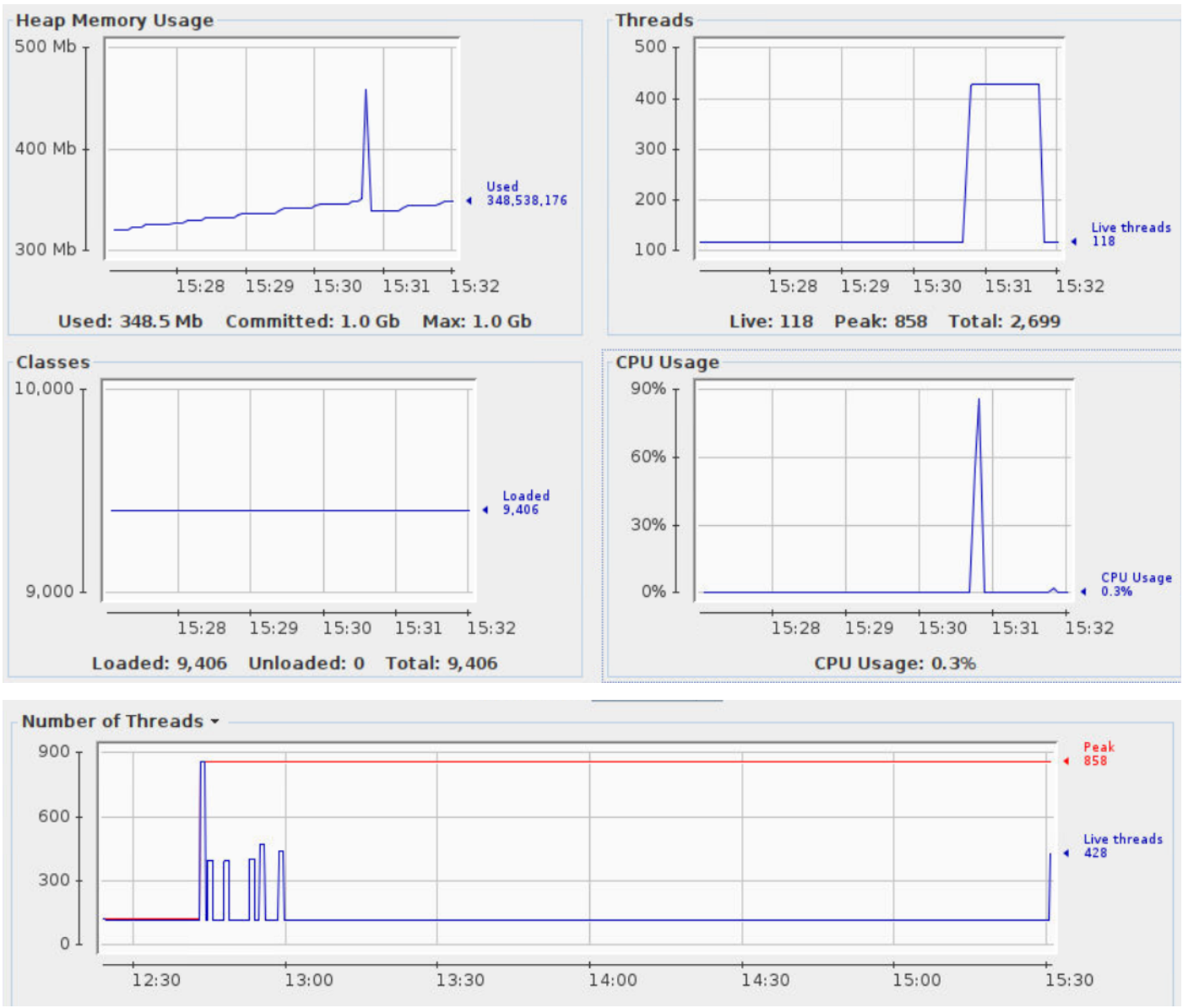
为方便使用 jmeter 进行性能测试，我们使用 json 与 csv 进行测试数据的编写，设计 12 个测试数据，其中 5 个为异常的没有 name 属性的数据，7 个为正常数据（错误数据率约为 41.67%，之后会有测试结果进行对比）



在设计线程组时，我们设置每隔 1 秒发出 500 次请求，循环 10 次，这样总计会发出 5000 次请求，共计建立 5000 个线程，以尽量确保能最大限度观察三种合法性检查的性能

六、 实验结果

1. 测试过程



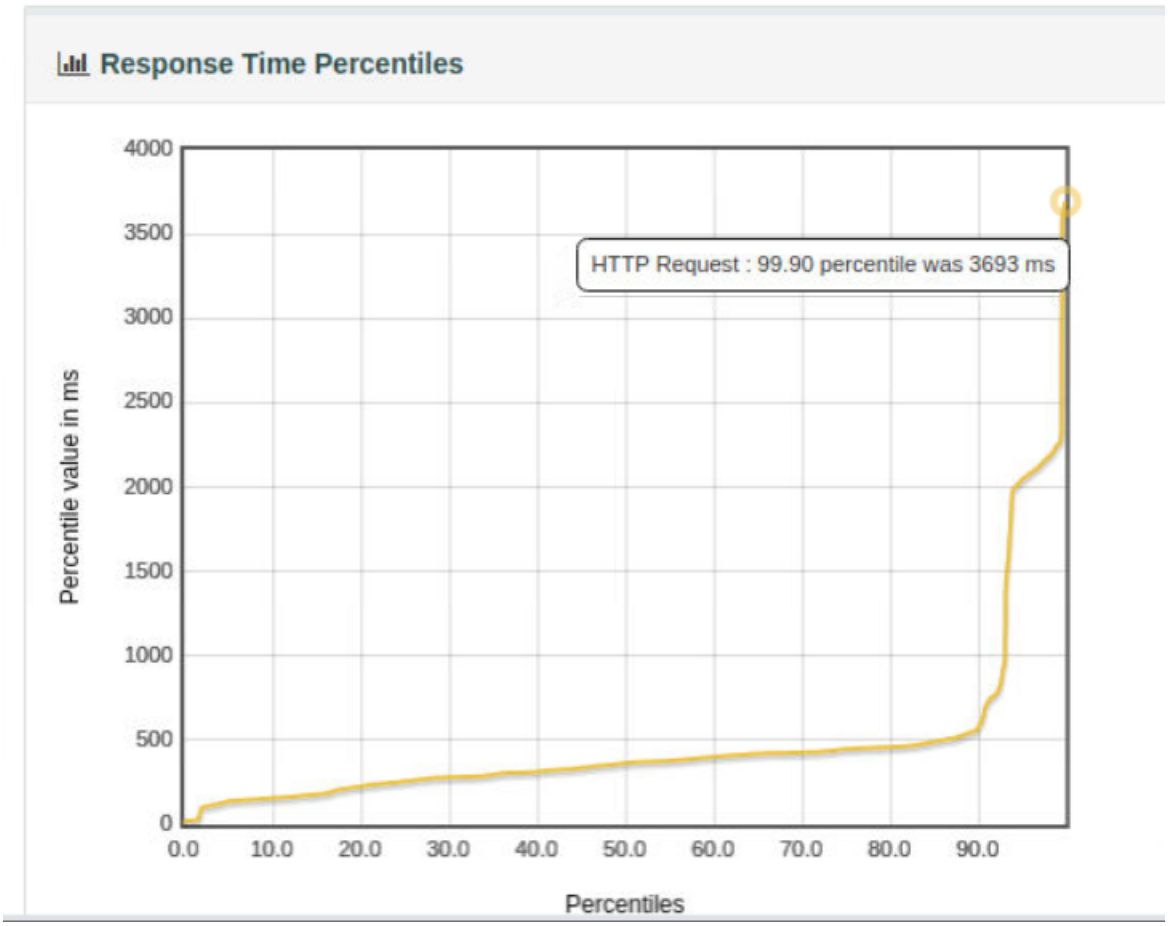
为使实验结果差异尽量明显，我们设计了较高的线程数，尽量在测试时让系统处于高负荷状态

2. 测试结果

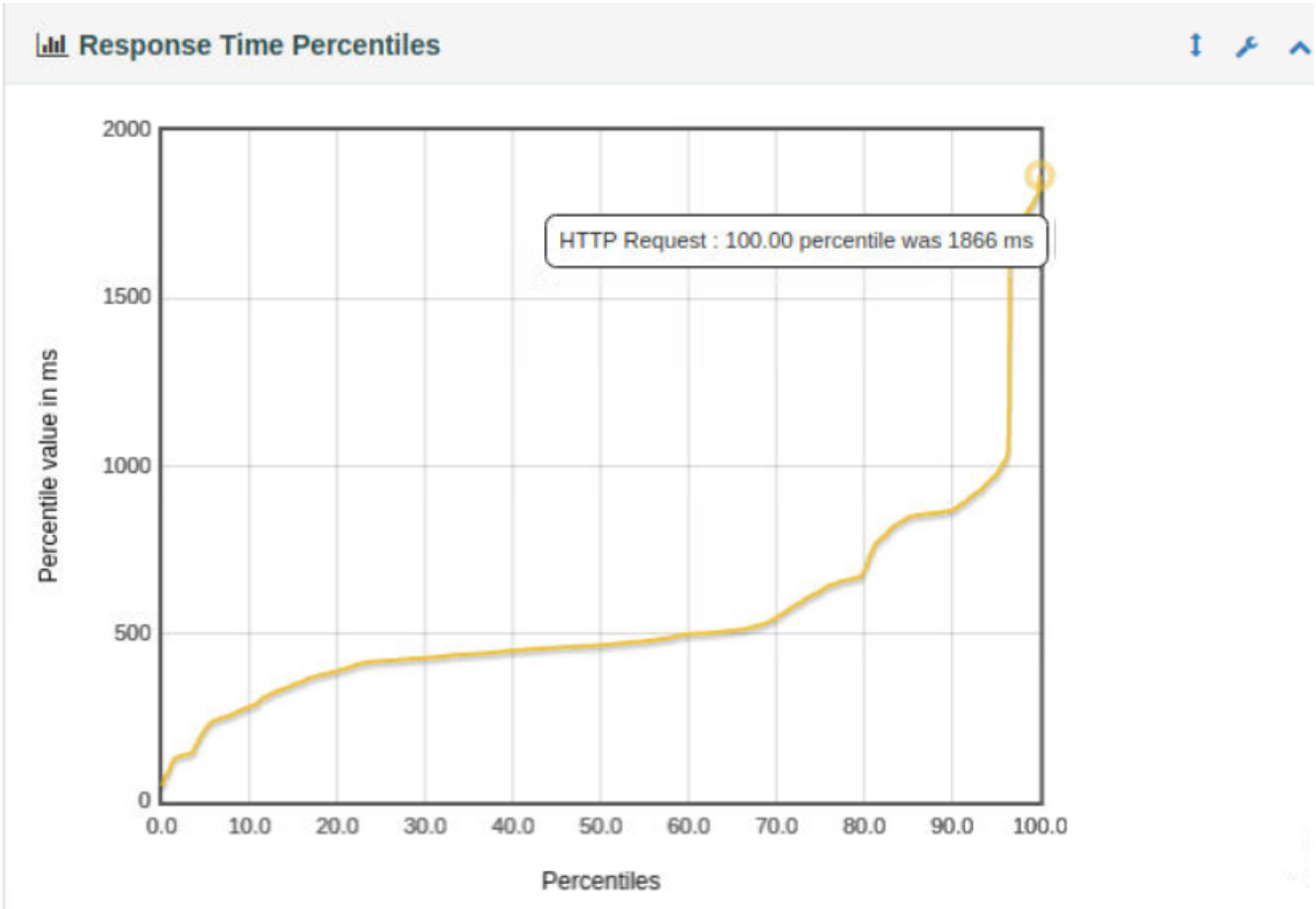
将三种测试后的数据作为 jtl 文件输出，并最终输出为 html 文件以便于观察

七、 结果分析

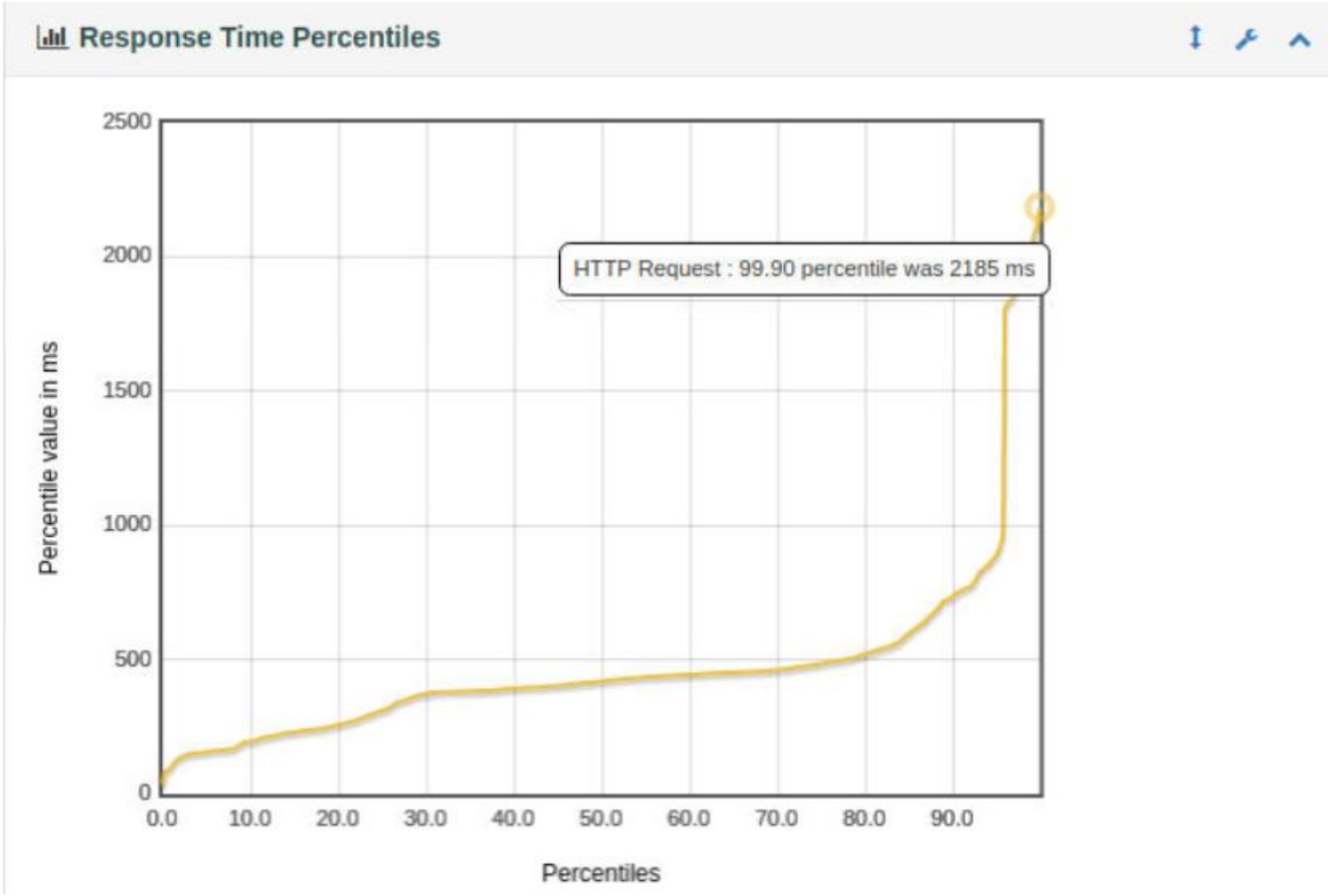
1. Response Time Percentiles 响应时间百分比


Response Time Percentiles
HTTP Request : 99.90 percentile was 3693 ms
Percentile value in ms
Percentiles
4000
3500
3000
2500
2000
1500
1000
500
0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0

图一：抛出异常


Response Time Percentiles
HTTP Request : 100.00 percentile was 1866 ms
Percentile value in ms
Percentiles
2000
1500
1000
500
0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0

图二：手动合法性检查

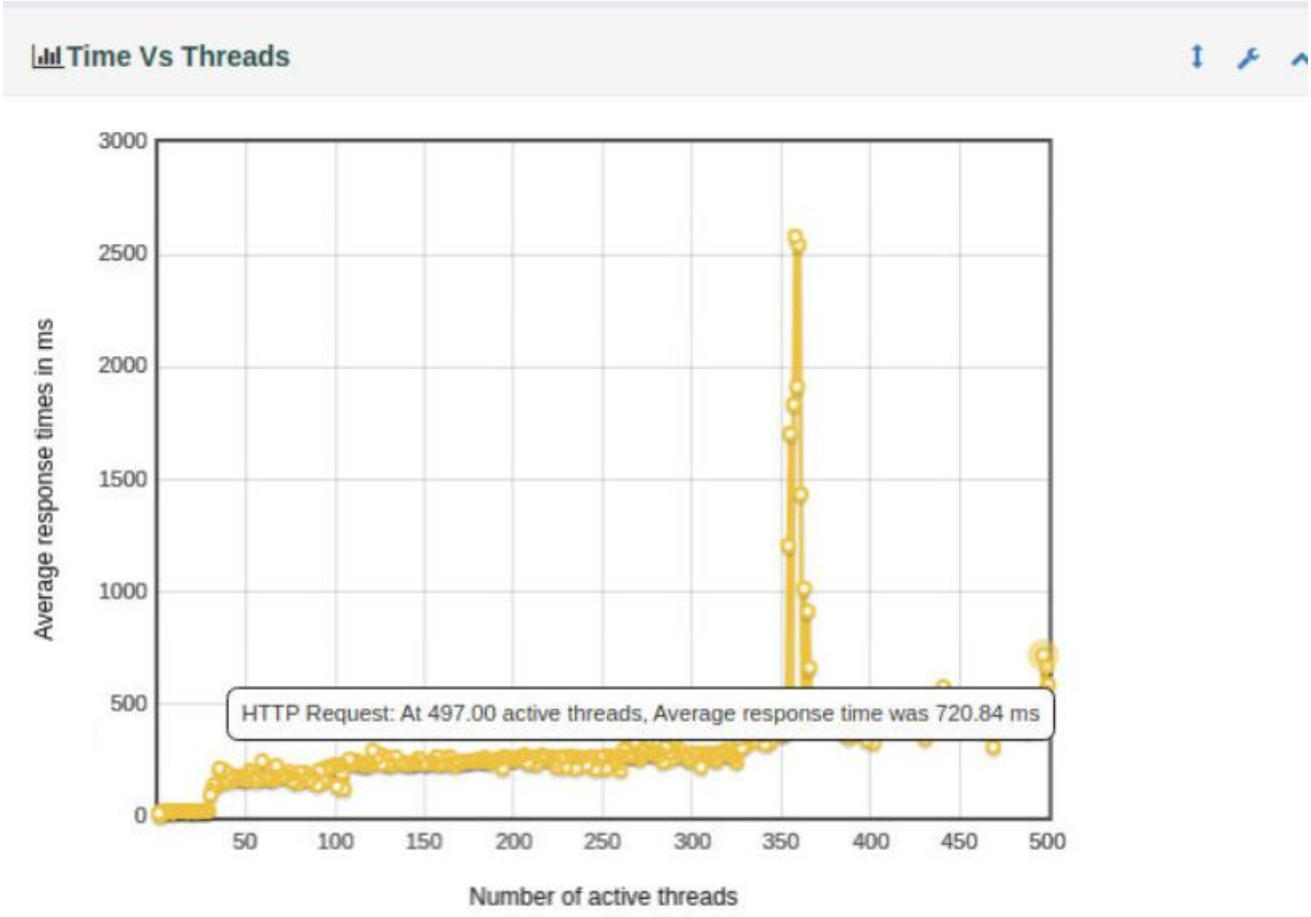


图三：使用 BindingResult

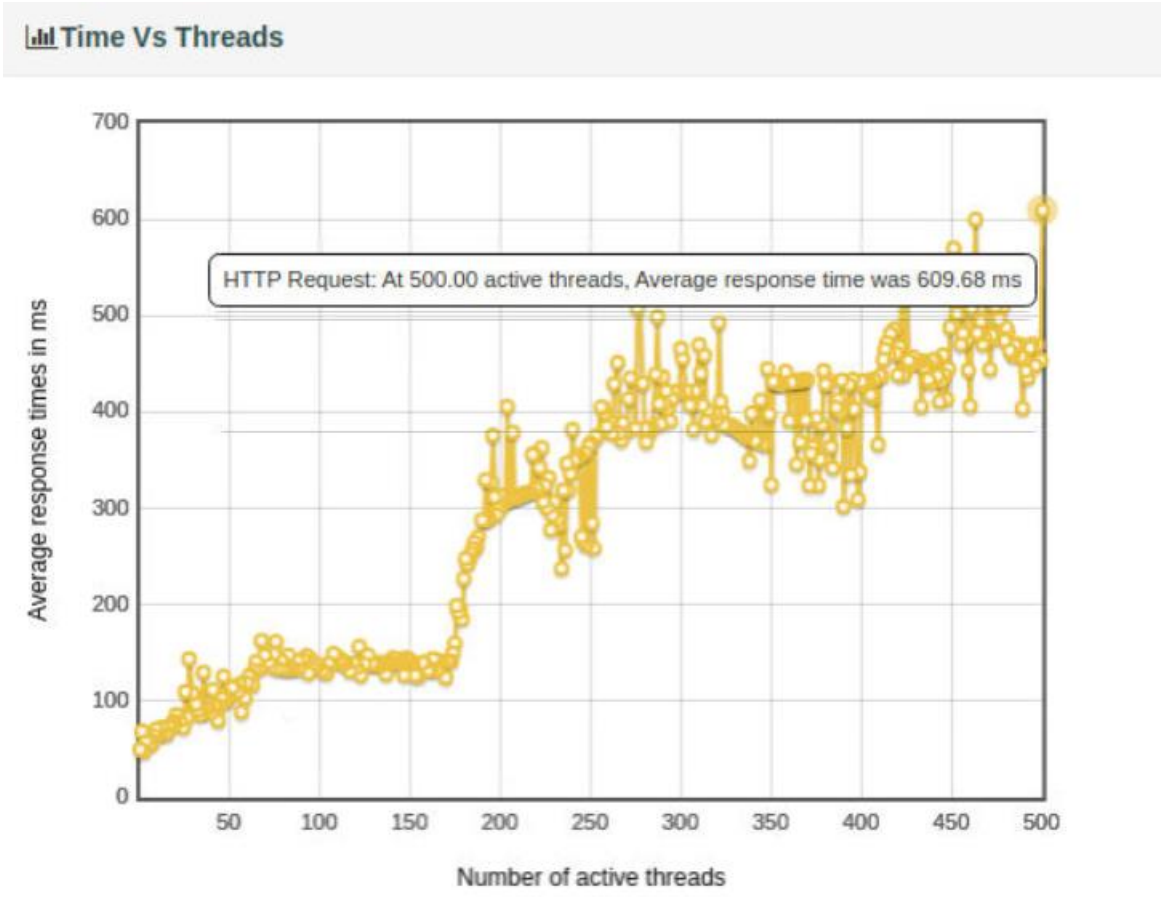
分析：

可以看出，手动合法性检查的最长响应时间最短，约为 1900ms；抛出异常最长响应时间最长，约为 3700ms。三种处理方式的大部分请求响应时间都在 500ms 以内

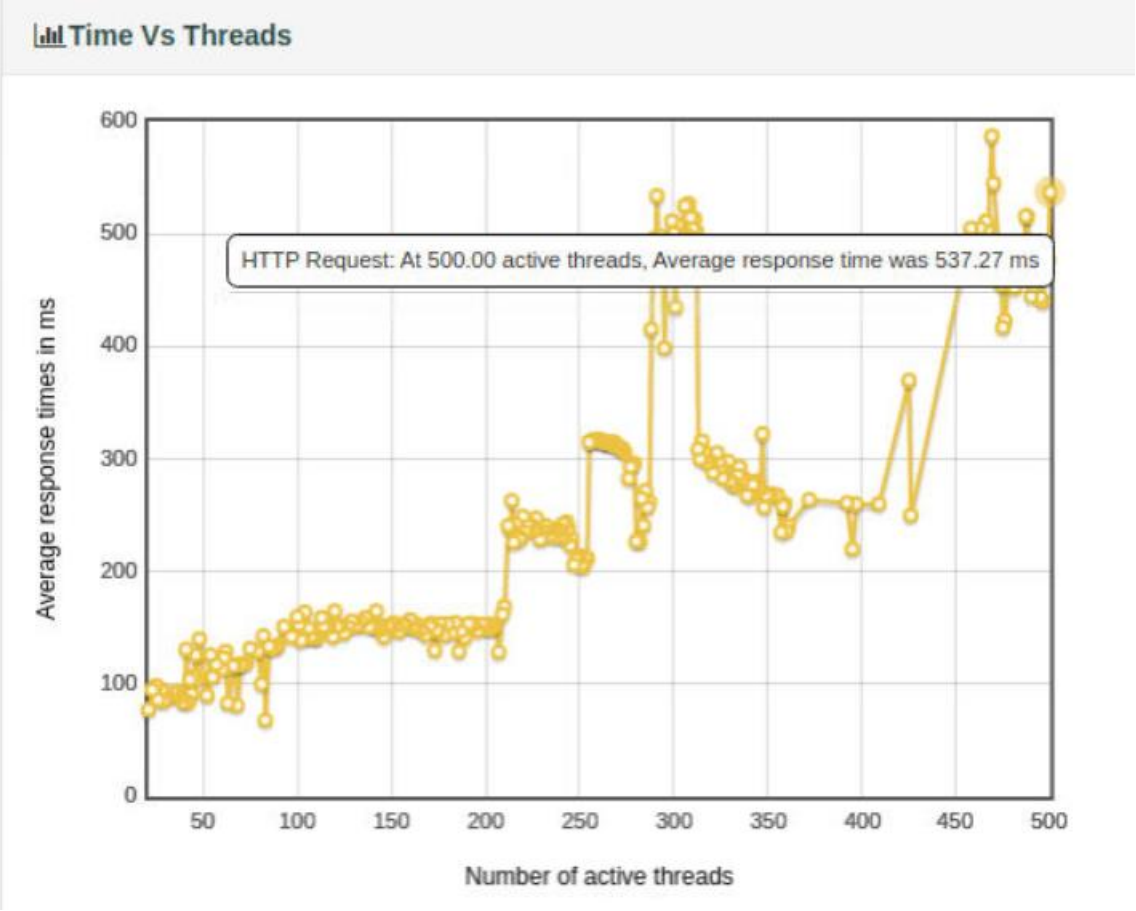
2. 平均响应时间


Time Vs Threads
Average response times in ms
Number of active threads
HTTP Request: At 497.00 active threads, Average response time was 720.84 ms
0
500
1000
1500
2000
2500
3000
50
100
150
200
250
300
350
400
450
500

图一：抛出异常


Time Vs Threads
Average response times in ms
Number of active threads
HTTP Request: At 500.00 active threads, Average response time was 609.68 ms
0
100
200
300
400
500
600
700
50
100
150
200
250
300
350
400
450
500

图二：手动检查



图三：使用 BindingResult

分析：

可以看出，使用抛出异常的方式，随着线程数增多，最终的响应时间最长，而且在约 370 个线程时，平均响应时间突然急剧增长

使用 BindingResult 的平均响应时间最短

3. 总体数据

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**5000**	**2081**	**41.62%**	**469.20**	**5**	**3746**	**366.00**	**582.70**	**2045.95**	**2252.99**	**887.47**	**232.59**	**241.95**
HTTP Request	5000	2081	41.62%	469.20	5	3746	366.00	582.70	2045.95	2252.99	887.47	232.59	241.95

图一：抛出异常

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	5000	2081	41.62%	552.01	48	1866	467.00	869.00	974.95	1773.00	846.31	221.80	238.99
HTTP Request	5000	2081	41.62%	552.01	48	1866	467.00	869.00	974.95	1773.00	846.31	221.80	238.99

图二：手动合法性检查

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	5000	2081	41.62%	476.08	35	2394	420.00	738.00	888.95	2047.97	949.85	248.94	271.94
HTTP Request	5000	2081	41.62%	476.08	35	2394	420.00	738.00	888.95	2047.97	949.85	248.94	271.94

图三：使用 BindingResult

分析：

我们主要关注是三个数据：平均响应时间、每秒吞吐量 Transactions/s、网络吞吐量

可以看出，虽然使用抛出异常进行处理的平均响应时间最短，但其并不稳定，最大响应时间达到 3746ms，响应时间中位数与平均值差别较大；且网络吞吐量并不高，因此我们可以认为抛出 Exception 是一种较为低效的处理方法

在其余两种方法中，响应时间变化都比较稳定。但手写合法性检查的事务处理速度与网络吞吐量低于使用 BindResult，而且手写合法性检查在实际编写时需要人为考虑各种异常情况，开发效率也较低

八、 附加文件

测试数据：位于 Exp_1/data 目录下（csv 文件存在编码错误，暂未解决）
原始 htl 测试文件：位于 Exp_1/testfile 目录下