软件测试 实验报告—JMeter

1、实验目的

学习压力测试工具 JMeter 的使用：编写一个简单的注册登录小程序，使用 JMeter 分别对注册和登录进行压力测试。要求：

（1） 注册和登录成功后要跳转到一个显示用户名的界面，并通过输入的用户名对跳转的页面进行验证；如果失败，则跳转到一个显示“操作失败”的页面，并对该返回页面的内容进行验证。

（2） 压力测试过程中要求使用参数配置功能，测试多组用户登录和注册操作，需要测试较多的并发用户数量，能够体现压力，例如执行时间至少 5 分钟。

（3） 通过增加 Tomcat 对 JVM 内存的分配来进行优化，给出优化前后的性能对比。

2、测试范围

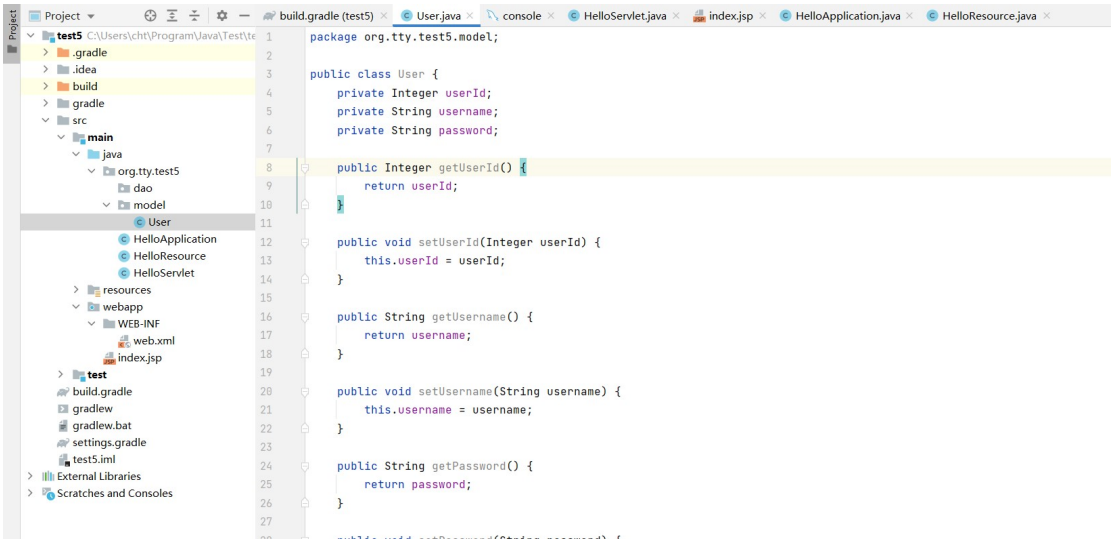
对 JMeter 中的注册和登录操作进行测试。

3、 测试过程

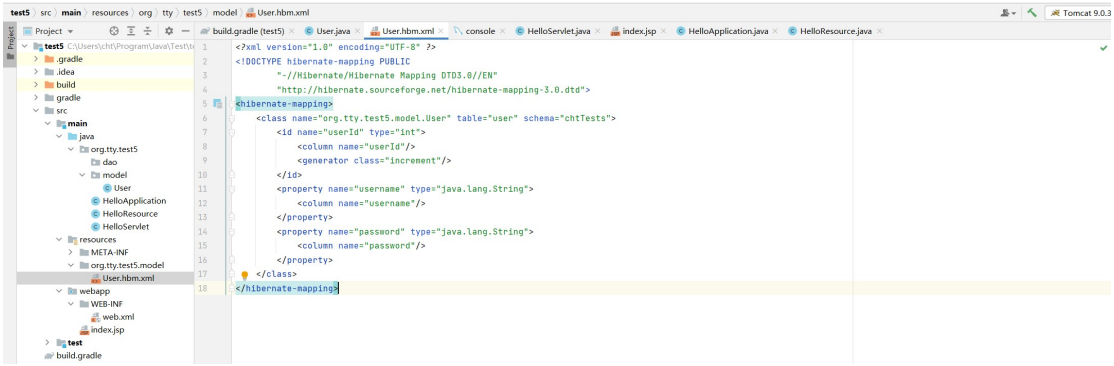
3.1：新建 test5 项目。进行登录和注册逻辑的编写。

创建实体类，model、dao 和 hibernate-mapping
首先创建数据库 chtTests，然后创建表 user

创建实体类 User

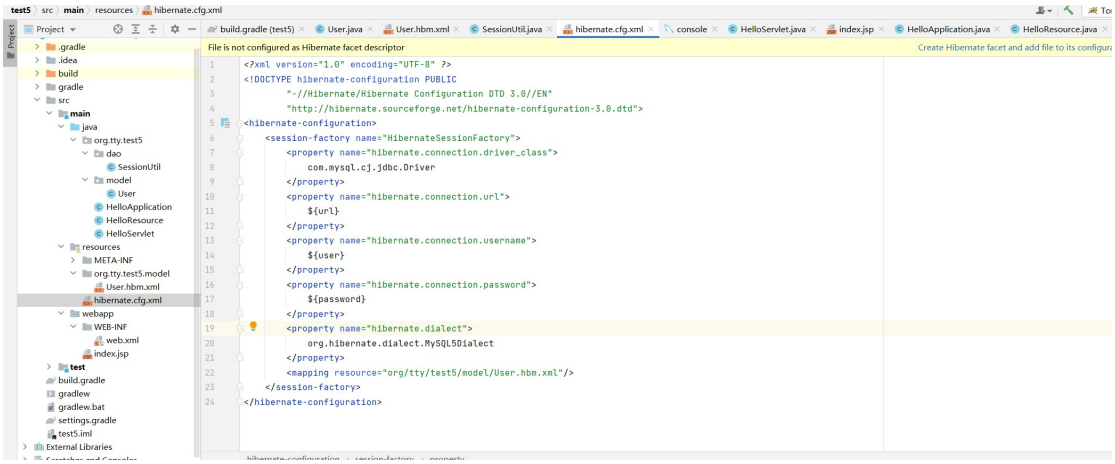


然后创建 User.hbm.xml



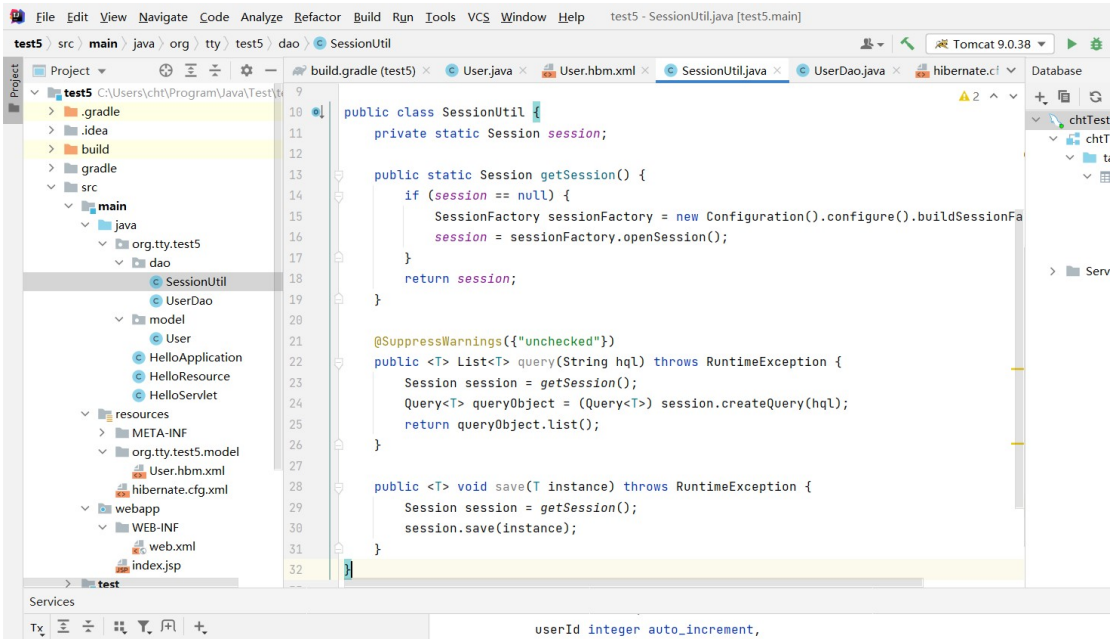
```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="org.tty.test5.model.User" table="user" schema="chtTests">
        <id name="userId" type="int">
            <column name="userId"/>
            <generator class="increment"/>
        </id>
        <property name="username" type="java.lang.String">
            <column name="username"/>
        </property>
        <property name="password" type="java.lang.String">
            <column name="password"/>
        </property>
    </class>
</hibernate-mapping>
```

然后创建 hibernate.cfg.xml 配置文件



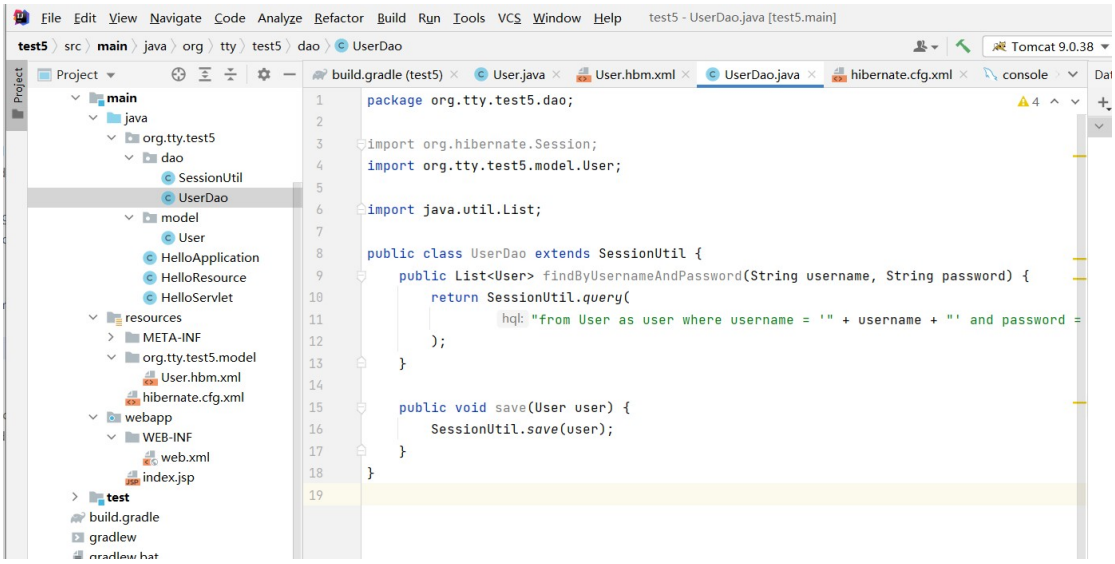
```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory name="HibernateSessionFactory">
        <property name="hibernate.connection.driver_class">
            com.mysql.cj.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            ${url}
        </property>
        <property name="hibernate.connection.username">
            ${user}
        </property>
        <property name="hibernate.connection.password">
            ${password}
        </property>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQL5Dialect
        </property>
        <mapping resource="org/tty/test5/model/User.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

然后添加 SessionUtil



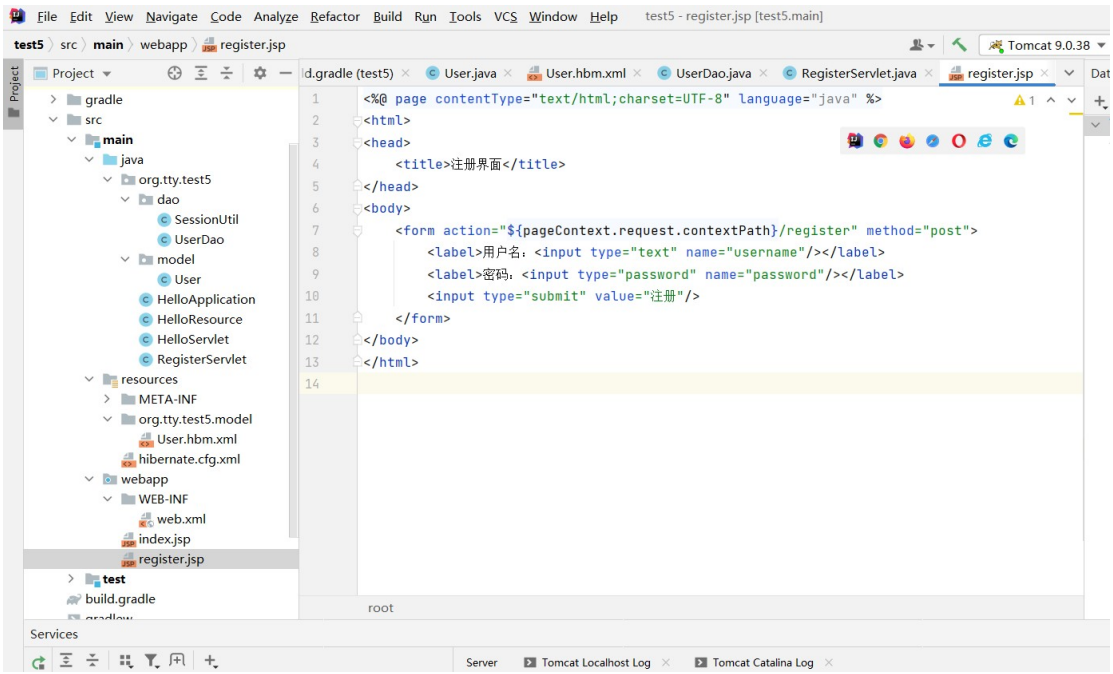
```java
public class SessionUtil {
    private static Session session;

    public static Session getSession() {
        if (session == null) {
            SessionFactory sessionFactory = new Configuration().configure().buildSessionFa
            session = sessionFactory.openSession();
        }
        return session;
    }

    @SuppressWarnings({"unchecked"})
    public <T> List<T> query(String hql) throws RuntimeException {
        Session session = getSession();
        Query<T> queryObject = (Query<T>) session.createQuery(hql);
        return queryObject.list();
    }

    public <T> void save(T instance) throws RuntimeException {
        Session session = getSession();
        session.save(instance);
    }
}
```

然后添加 UserDao

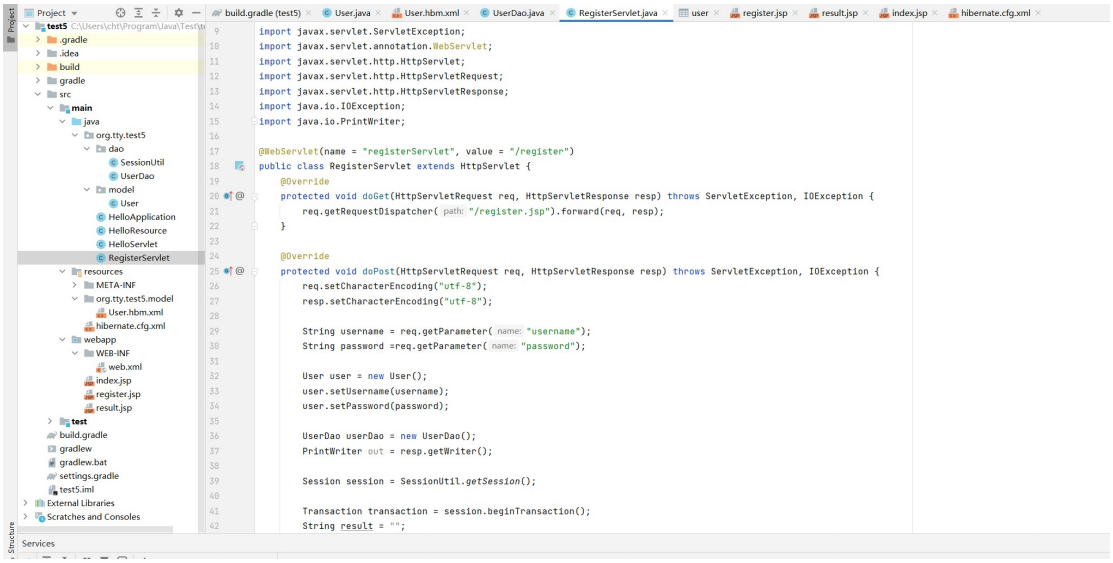


3.2：编写注册界面

register.jsp



RegisterServlet 的逻辑



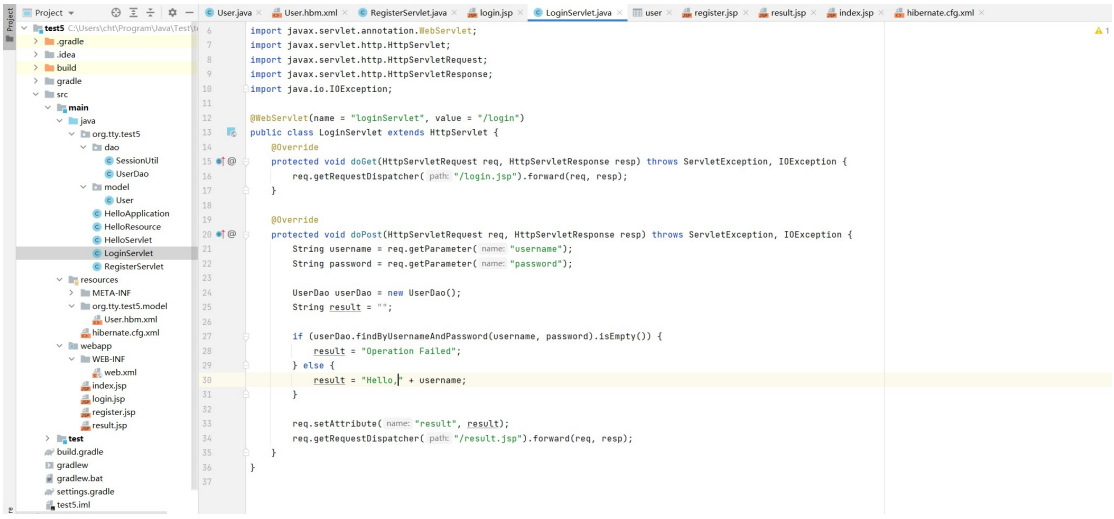
```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet(name = "registerServlet", value = "/register")
public class RegisterServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.getRequestDispatcher( path: "/register.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");

        String username = req.getParameter( name: "username");
        String password =req.getParameter( name: "password");

        User user = new User();
        user.setUsername(username);
        user.setPassword(password);

        UserDao userDao = new UserDao();
        PrintWriter out = resp.getWriter();

        Session session = SessionUtil.getSession();

        Transaction transaction = session.beginTransaction();
        String result = "";
```

3.3：编写登录界面

login.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>登录界面</title>
</head>
<body>
    <form action="${pageContext.request.contextPath}/login" method="post">
        <label>用户名，<input type="text" name="username"/></label>
        <label>密码，<input type="password" name="password"/></label>
        <input type="submit" value="登录"/>
    </form>
</body>
</html>
```

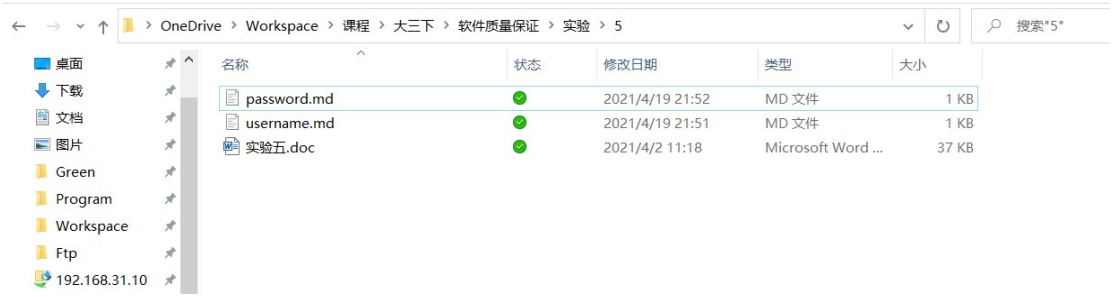
LoginServlet



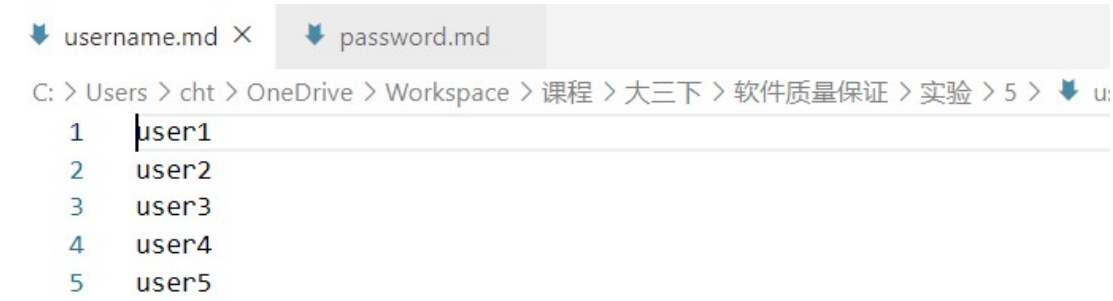
```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "loginServlet", value = "/login")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.getRequestDispatcher( path: "/login.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String username = req.getParameter( name: "username");
        String password = req.getParameter( name: "password");

        UserDao userDao = new UserDao();
        String result = "";

        if (userDao.findByUsernameAndPassword(username, password).isEmpty()) {
            result = "Operation Failed";
        } else {
            result = "Hello," + username;
        }

        req.setAttribute( name: "result", result);
        req.getRequestDispatcher( path: "/result.jsp").forward(req, resp);
    }
}
```

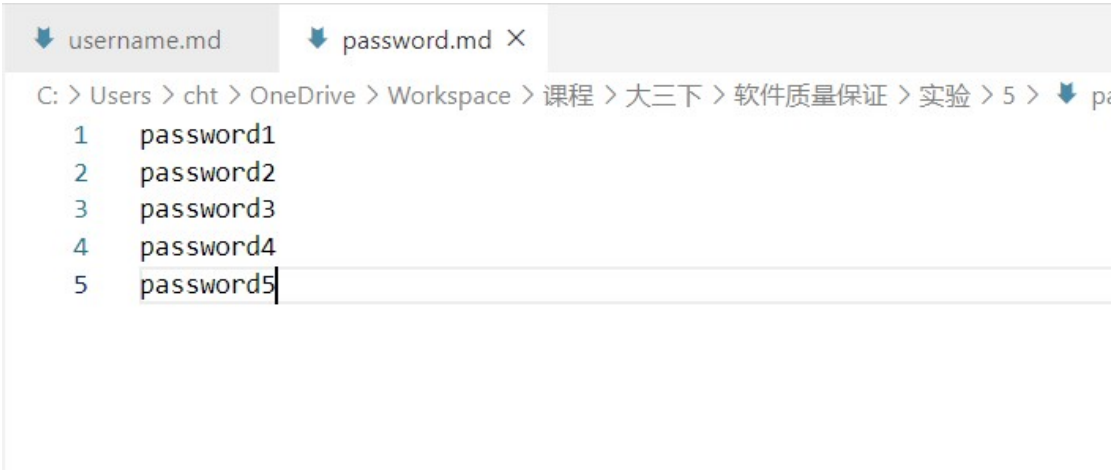
3.4：进行 JMeter 进行压力测试

1、添加用户变量配置文件



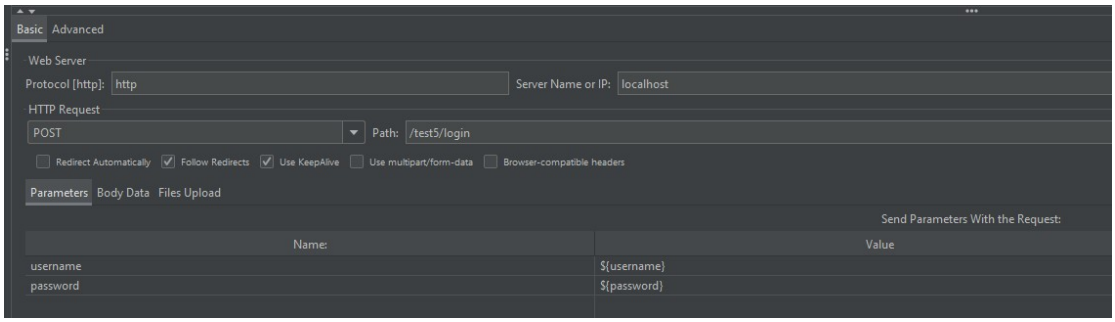
其中内容如下：





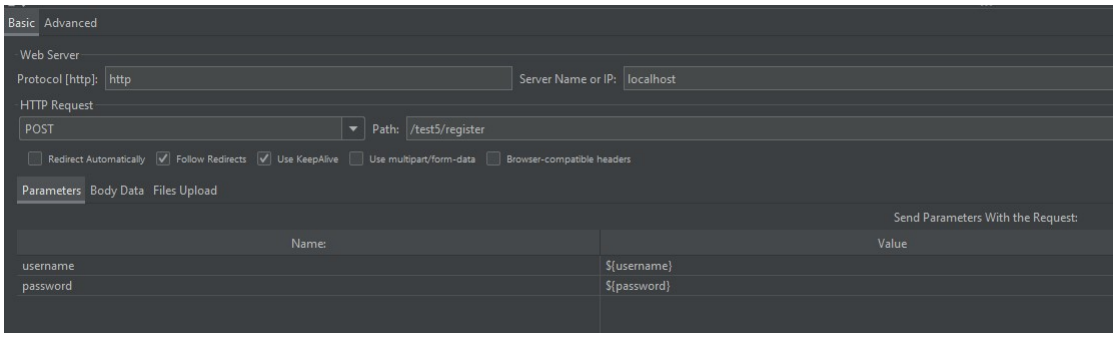
然后在线程组内添加 CSV 的数据文件配置，将刚才创建的两个变量配置上去。

2、添加对应的 Login 和 Register 的 HttpRequest 和 HttpResponse



其中登录的接口是/test5/login，使用 post 的方法进行传参，其中用户名和密码使用变量的方式从.md 文件中读取。

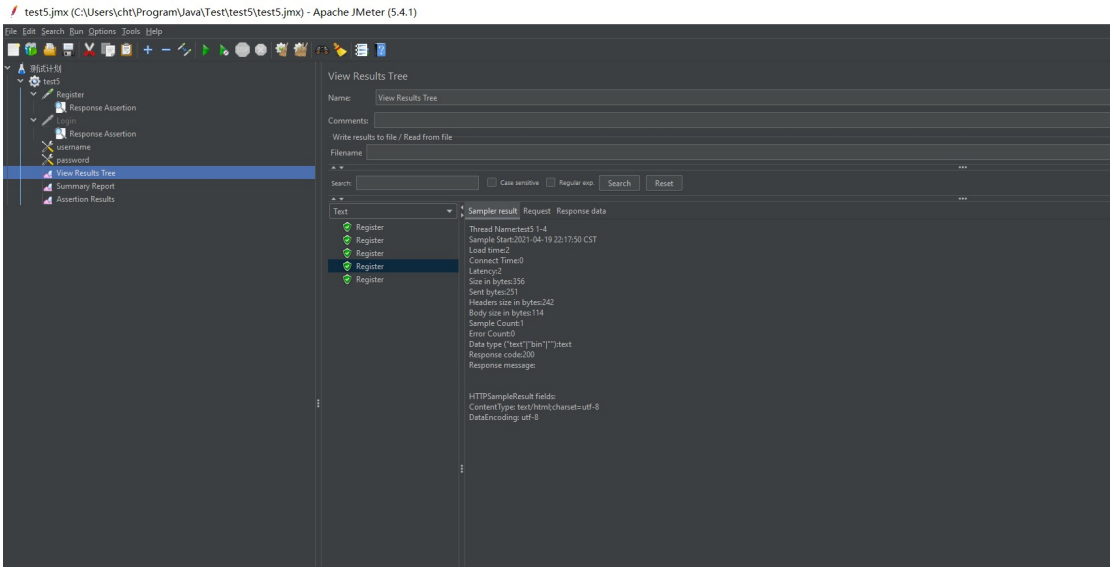
然后以同样的方式创建注册的 HttpRequest，配置方式和上面的类似。



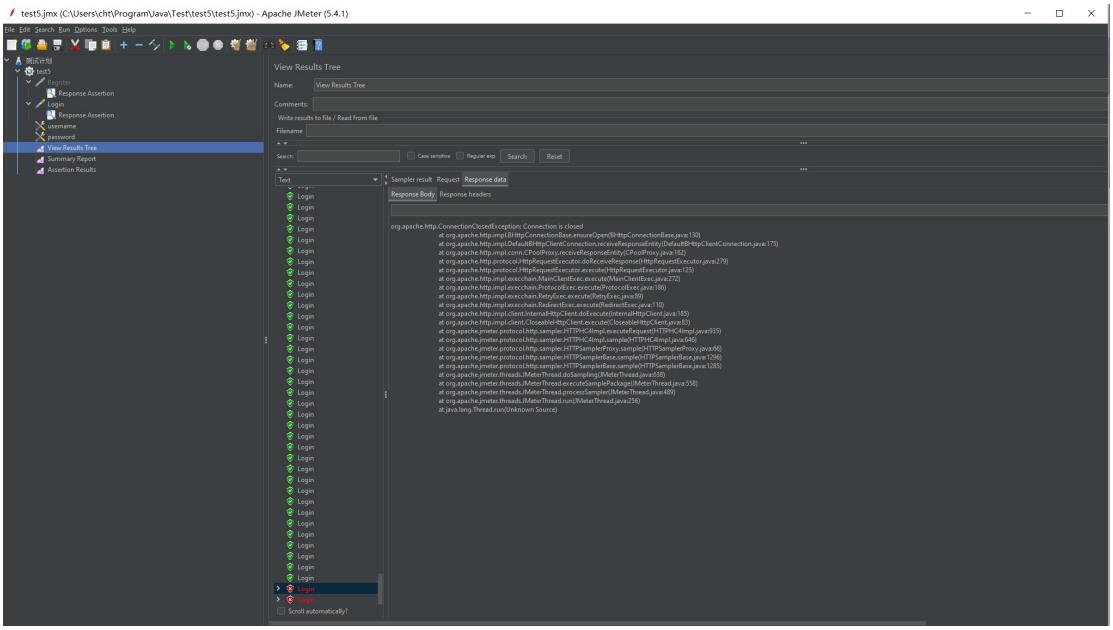
修改配置，线程数设置为 5，递增量设为 1，并进行测试。

3.5：测试结果分析

注册的压力测试结果

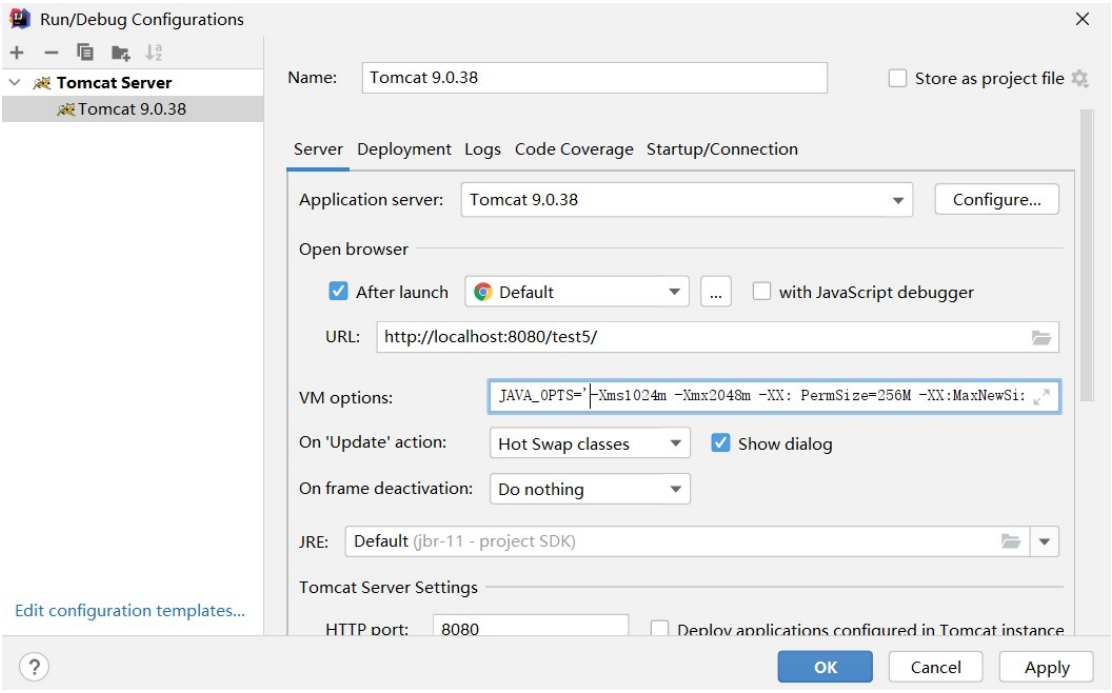


登录的压力测试结果，我们发现在第 1007 次登录后出现了错误。

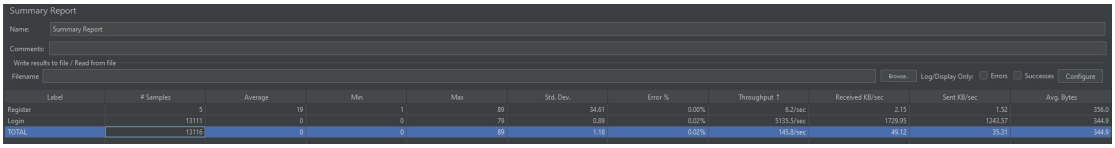


3.6：性能优化

我们尝试给 Tomcat 启动指定参数来扩大其启动的内存。



然后重新进行压力测试。



我们能够发现，相对优化前，流量提高了约 120%，相对来说这是一次比较好的改善。

4、总结与分析

在进行压力测试中，目前发现性能的瓶颈出现在数据库连接中。常常出现的问题有：

1、 数据库已经断开连接。

2、 数据库连接超时，因为同时过多的连接而拒绝连接。

相对于内存来说，数据库的压力会更大。当同时出现多个用户访问时，如果没有好的并发策略，非常容易出现长时间无响应，或者更严重的数据库崩溃情况。

一种常见的解决方式是使用连接池的方式进行管理，当然，连接池也会因为无法启用更多的连接导致崩溃。这个时候就需要进行流量限制，并使用数据库长连接来减少创建数据库连接的资源消耗。

同时，也可以使用流量屏蔽技术对大流量访问的 IP 进行屏蔽以减小大流量带来的冲击。