1、前端发送的请求方式要与服务器端要求的请求方式一致

当后端要求前端的请求方式为POST方式时，前端应该发送POST请求，若不是，服务器应该提示错误信息

搭建web项目进行测试：

先要解决的是如何获取请求方式：

我们知道在service方法中有两个形式参数，分别是ServletRequest类型定义的对象以及ServletResponse类型定义的对象，查看JavaEE的帮助文档，javax.servlet.ServletRequest接口有一个子接口javax.servlet.http.HttpServletRequest，该接口有个方法为getMethod()，而请求协议中的请求行由三部分组成，该方法正好获取的是请求行第一部分的请求方式，返回该请求方式的字符串。就是说，service中的servletrequest是ServletRequest接口的子接口HttpServletRequest的实现类对象，该实现类是web容器实现并提供的对象，为什么方法中用javax.servlet.ServletRequest定义，主要是由于面向接口编程，面向父类编程的思想。但是由于ServletRequest接口中没有getMethod()方法，所以使用servletrequest时需要强制转换成HttpServletRequest接口类型。

新建一个登录页面login.html，在web.xml中将其设置为欢迎页面，设置超链接请求方式为GET，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>login page</title>
</head>
<body>
    <form action="/idea-servlet-http-6/login" method="get">
        用户名<input type="text" name="username">
        密码<input type="password" name="password">
        <input type="submit" value="login">
    </form>
</body>
</html>
```

web.xml文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

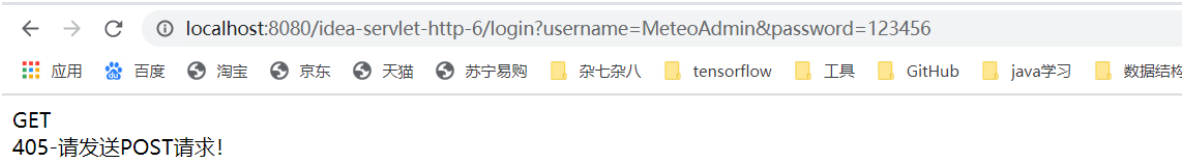
```
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <welcome-file-list>
        <welcome-file>welcome page</welcome-file>
        <welcome-file>login.html</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>login</servlet-name>
        <servlet-class>com.servlet.http.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>login</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
</web-app>
```

Java程序代码如下：

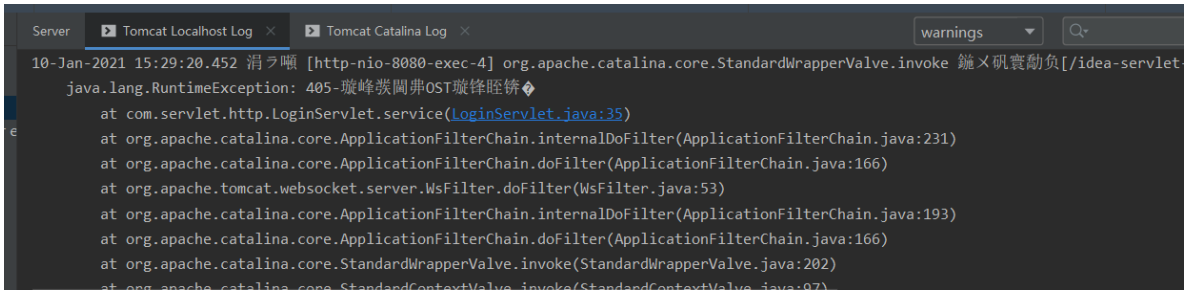
```
package com.servlet.http;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class LoginServlet extends GenericServlet {

    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

        //将两个形式参数强制转换成HttpServletRequest和HttpServletResponse类型
        HttpServletRequest request = (HttpServletRequest)servletRequest;
        HttpServletResponse response = (HttpServletResponse)servletResponse;

        //更改输出流位置以及输出信息数据格式和编码方式(注意：要先设置格式后再设置流）
        servletResponse.setContentType("text/html;charset=UTF-8");
        PrintWriter out = servletResponse.getWriter();

        //获取请求方式的字符串返回
        String method = request.getMethod();
        out.print(method);
        out.print("<br>");

        //要求前台需要用POST请求访问这个实现类对象的资源
        if ("GET".equals(method)){
            //前端提供错误提示
            out.print("405-请发送POST请求！");
            //后台要报错误
            throw new RuntimeException("405-请发送POST请求！");
        } else if ("POST".equals(method)) {
```

```
            //程序正常运转结束
            out.print("正在登录，请稍后.......");
        }

    }
}
```

部署项目后进入欢迎页面，点击登录运行结果如下图：



抛出的运行时异常截图如下：



输出的异常信息是在Tomcat Localhost Log这个日志模块中显示的，此时是乱码，之后会解决。

向上面这种方式来保持前端的请求方式与后端所要求的请求方式有点过于复杂，在实际开发中我们并不仅仅是一个Servlet对象。如果所有的实现类对象都向上面一样编写重复代码就会很麻烦，所以能否将某个片段代码块封装起来，以方便开发?

方案：

可以创建一个类HttpServlet继承GenericServlet类，我们在这个类的service方法指中封装获取请求方式以及条件判断语句块，然后我们以后的Servlet实现类就继承该类，无需再继承GernericServlet类。

需要解决的就是这httpServlet类是父类，以后的子类会有很多，子类中对请求方式的要求都是未知的，如何设计呢?

我们可以在HttpServlet类中编写两个方法：doGet()和doPost()。当获取到的请求方式是GET，执行doGet方法；获取到的方法是POST，就执行doPost方法。

在HttpServlet类中这两个方法中的内容都是向前端发送错误提示和向后台抛出异常。

而继承该类的子类就要重写这两个方法中的某一个方法，要求前端发送GET请求，就重写doGet方法；要求POST请求就重写doPost方法。重写的内容就是业务需求代码。

可以这样理解，后台一个Servlet对象资源需要POST请求方式，重写了doPost方法；

当前端发送了一个GET方式的请求这个Servlet对象的资源，被判断执行doGet方法，由于该Servlet对象只重写了doPost方法，继承的doGET方法的内容仍然是其父类中该方法的内容，执行结果就是向前端提示需要POST请求方式，后台抛出相应异常。

当前端发送了一个POST方式的请求这个Servlet对象的资源，被判断执行doPost方法，该Servlet对象重写了doPost方法，此时就会执行重写后的doPost方法，是正常执行结束。

具体实现代码如下：

```
ackage com.servlet.http;


import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class HttpServlet extends GenericServlet {


    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

        HttpServletRequest request = (HttpServletRequest)servletRequest;
        HttpServletResponse response = (HttpServletResponse)servletResponse;

        String method = request.getMethod();
```

```
        if ("GET".equals(method)){
            doGet(request,response);
        } else if ("POST".equals(method)) {
            doPost(request,response);
        }

    }

    public void doGet(HttpServletRequest request,HttpServletResponse response)
throws IOException {

        response.getWriter().print("405: 请发送POST请求！");
        throw new  RuntimeException("405: 请发送POST请求!");

    }


    public void doPost(HttpServletRequest request,HttpServletResponse response)
throws IOException {
        response.getWriter().print("405: 请发送GET请求！");
        throw new  RuntimeException("405: 请发送GET请求!");
    }

}
```

这样的封装类，在Servlet规范中SUN公司已经帮我们实现了，javax.servlet.http.HttpServlet。不过这个继承GenericServlet类中两个do方法中是执行ERROR,如下图

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String msg = lStrings.getString( key: "http.method_get_not_supported");
    this.sendMethodNotAllowed(req, resp, msg);
}

protected long getLastModified(HttpServletRequest req) { return -1L; }

protected void doHead(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    if (DispatcherType.INCLUDE.equals(req.getDispatcherType())) {
        this.doGet(req, resp);
    } else {
        NoBodyResponse response = new NoBodyResponse(resp);
        this.doGet(req, response);
        response.setContentLength();
    }

}

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String msg = lStrings.getString( key: "http.method_post_not_supported");
    this.sendMethodNotAllowed(req, resp, msg);
}
```

```
    private void sendMethodNotAllowed(HttpServletRequest req, HttpServletResponse resp, String msg) throws IOException {
        String protocol = req.getProtocol();
        if (protocol.length() != 0 && !protocol.endsWith("0.9") && !protocol.endsWith("1.0")) {
            resp.sendError( i: 405, msg);
        } else {
            resp.sendError( i: 400, msg);
        }

    }
```

虽然不需要我们实现，但是我们要学习这样的一种编程思想。还有就是以后写Servlet实现类只需要继承javax.servlet.http.HttpServlet类，然后根据需要前端发送什么类型的请求来选择重写doGet和doPost方法，重写的内容就是业务需求的实现。