武汉纺织大学

Web应用开发课程设计

**基于API的在线翻译系统**

**学 院： 数学与计算机学院**

**班 级： 软件工程11801**

**姓 名： 马宇聪**

**学 号： 1804240713**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

目 录

[1 需求分析 1](#_Toc2617)

[1.1在线翻译 1](#_Toc5512)

[1.2主页面 1](#_Toc3125)

[1.3各项功能 1](#_Toc32687)

[1.3.1 选择语言 1](#_Toc22622)

[1.3.2 统计字数 1](#_Toc19531)

[1.3.3 翻译语言 1](#_Toc28229)

[1.3.4 显示结果 1](#_Toc26627)

[1.3.5 一键清空输入框 2](#_Toc4343)

[2 系统设计 2](#_Toc29697)

[2.1用户用例图 2](#_Toc4060)

[2.2 UML类图（Class Diagram） 2](#_Toc4437)

[2.4.1 在线翻译 3](#_Toc25092)

[2.5 UML活动图（Activity Diagram） 4](#_Toc13335)

[2.5.1 在线翻译 4](#_Toc26408)

[3 系统实现 6](#_Toc12549)

[3.1 项目结构 6](#_Toc19824)

[3.2 配置文件 6](#_Toc24728)

[3.2.1 web.xml文件 6](#_Toc10904)

[3.2.2 fanyi.xml文件 7](#_Toc19484)

[3.3 服务工具包 7](#_Toc2871)

[3.3.1 fanyiHelper.java 7](#_Toc15455)

[3.3.2 MD5Util.java 11](#_Toc4396)

[3.4 fanyiController.java 12](#_Toc32488)

[3.9 main.js 14](#_Toc17019)

[4 系统测试 15](#_Toc7641)

[5 系统总结 18](#_Toc29738)

# 1 需求分析

设计一个在线翻译网页，实现“选择语言”、“统计字数”、“翻译语言”、“显示结果”、“一键清空输入框”等功能，具体要求如下：

## 1.1在线翻译

当程序运行时，显示主界面，用户通过左侧输入框输入文字，选择要翻译的语言方向，点击翻译，在右侧文本框得到翻译结果。

## 1.2主页面

进入系统主界面，如下：

## 

## 1.3各项功能

### 1.3.1 选择语言

当选择“自动检测语言时”时，系统默认语言方向为“中文>>英文”，语言方向id“zh-CHS2en”；

当选择“中文>>英文”时，系统翻译从文本框输入的中文字符串为英文，语言方向id“zh-CHS2en”；

当选择“英文>>中文”时，系统翻译从文本框输入的英文字符串为中文，语言方向id“en2zh-CHS”；

其余语种还在开发中，敬请期待。

### 1.3.2 统计字数

当在输入框输入文本时，文本框的右下角会统计当前字符数，设置一个最大字符数，若输入字符数超过最大值给出提示；

### 1.3.3 翻译语言

当输入框有数据输入时，选择语种方向后点击翻译按钮，web前端取到三个数据，即要翻译的文本query，翻译方向from、to，通过ajax的post方法调用后端fanyiController进行处理翻译，后端得到翻译结果，通过response返回给前端，前端在右侧文本框中显示后端返回的翻译结果；

### 1.3.4 显示结果

在前端接收到从后端收到的返回结果并且返回的status为200正常时，在右侧块标签中通过span标签显示翻译结果

### 1.3.5 一键清空输入框

在输入框输入文本后，通过一键点击清空输入框中的内容

# **2 系统设计**

## 2.1用户用例图

用户用例如下图2-1所示：

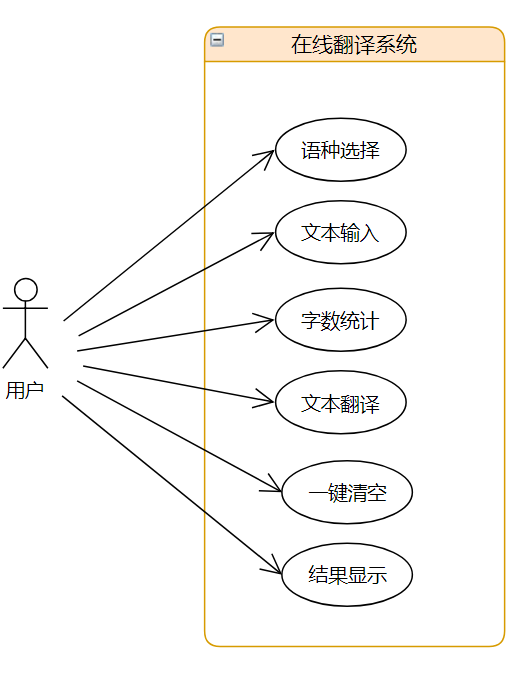


图2-1 用户用例图

## 2.2 UML类图（Class Diagram）

对于用在线翻译系统，共设计如下两个类。

* fanyiController：接收前端页面ajax传来的用户输入框中的信息，调用fanyiHeper请求API的数据，数据处理后以map的形式放在response中返回到前端ajax中
* fanyiHelper：在含参构造函数中，参数是fanyiController传进来的用户前端输入的待翻译的内容query(String类型)以及语种方向from、to(String)，在该方法中创建Apache包下面CloseableHttpClient类、HttpGet类和CloseableHttpResponse类，来向有道翻译API发送GET请求，并接受返回的数据，数据处理后返回的是翻译结果的json字符串(String类型)，通过对json字符串的解析得到所要的translation结果.

各类的结构及类之间的关系如图2-5所示：

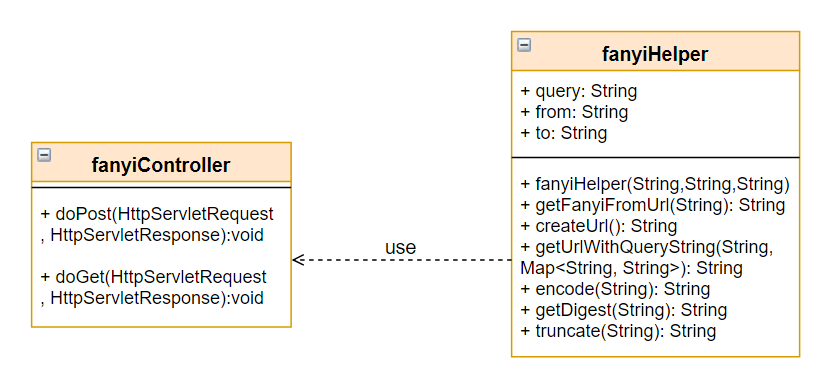
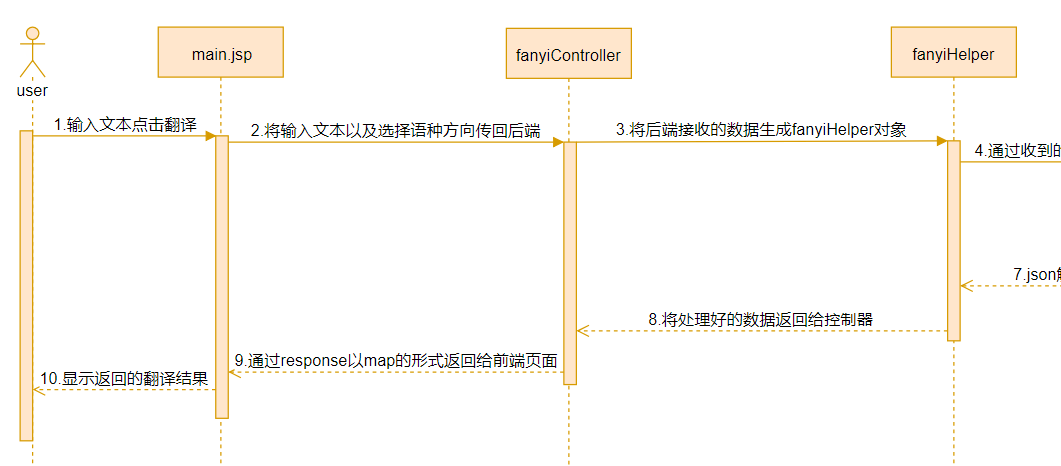
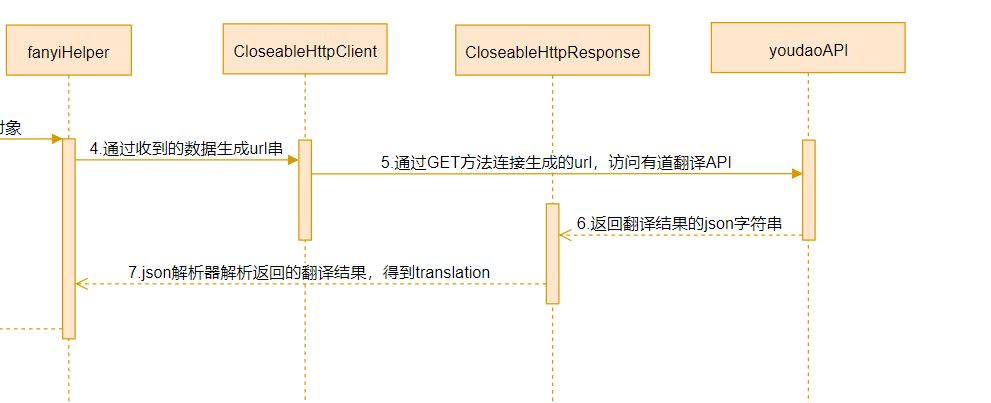
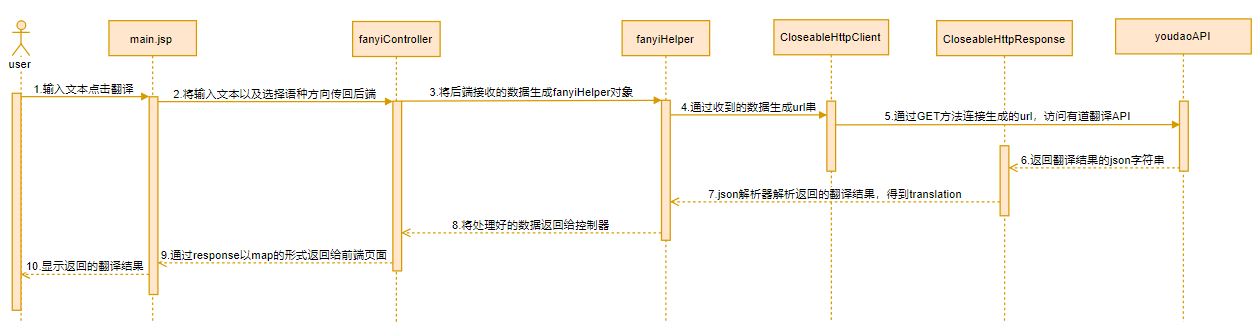


图2-5 用户功能类图

### 2.4.1 在线翻译







## 2.5 UML活动图（Activity Diagram）

### 2.5.1 在线翻译

用户在输入框输入文本，前端通过ajax调用post方法将收到的文本和语种方向来传给后端控制器，控制器调用服务类处理数据得到翻译结果，控制器再将翻译结果返回给前端显示，具体流程如图2-2所示。

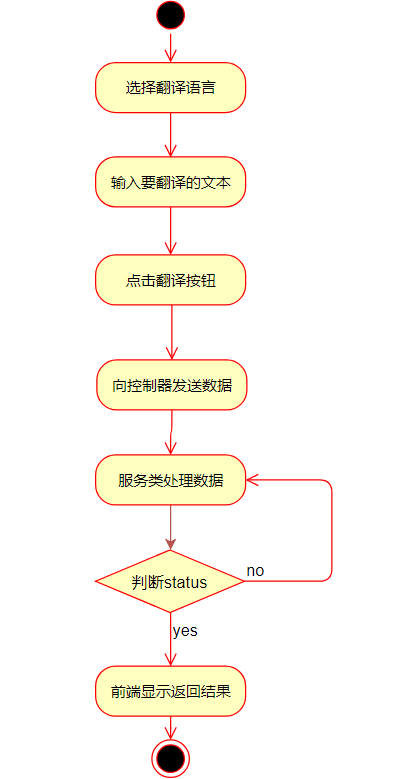
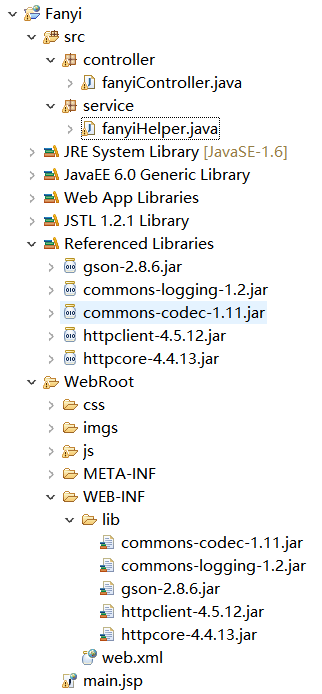


图2-2 在线翻译流程图

# **3 系统实现**

## 3.1 项目结构



## 3.2 配置文件

### 3.2.1 web.xml文件

配置进行http请求时访问的控制器的url路径

/<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<servlet>

<description>This is the description of my J2EE component</description>

<display-name>This is the display name of my J2EE component</display-name>

<servlet-name>fanyiController</servlet-name>

<servlet-class>controller.fanyiController</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>fanyiController</servlet-name>

<url-pattern>/servlet/fanyiController</url-pattern>

</servlet-mapping>

</web-app>

### 3.2.2 fanyi.xml文件

配置tomcat运行时访问的路径

<Context path="/fanyi" docBase="D:\MyEclipse2014\_Data\Fanyi\WebRoot" reloadable="true" />

## 3.3 服务工具包

### 3.3.1 fanyiHelper.java

主要用于生成一个fanyiHelper对象，通过在主函数中用对象来调用类中的方法生成url路径，以及访问路径得到翻译的json字符串结果，并对json字符串进行解析以得到所需要的翻译结果。

**package** service;

**import** java.io.IOException;

**import** java.io.UnsupportedEncodingException;

**import** java.net.URLEncoder;

**import** java.nio.charset.StandardCharsets;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.http.HttpEntity;

**import** org.apache.http.client.ClientProtocolException;

**import** org.apache.http.client.methods.CloseableHttpResponse;

**import** org.apache.http.client.methods.HttpGet;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.apache.http.impl.client.HttpClients;

**import** org.apache.http.util.EntityUtils;

**import** com.google.gson.JsonElement;

**import** com.google.gson.JsonObject;

**import** com.google.gson.JsonParser;

**public** **class** fanyiHelper {

**private** String query;

**private** String from;

**private** String to;

**public** String YOUDAO\_URL = "https://openapi.youdao.com/api";

**public** String APP\_KEY = "3f6680b71c3a16d8";

**public** String APP\_SECRET = "xKUAYaGE9T93J1QxD6u6KKwQrpT4YSNv";

**public** fanyiHelper(String query,String from ,String to) {

**this**.query = query;

**this**.from=from;

**this**.to=to;

}

**public** String getFanyiFromUrl(String url) **throws** Exception, IOException {

String fanyi = "";

// 访问API

CloseableHttpClient closeableHttpClient = HttpClients.*createDefault*();

HttpGet httpGet = **new** HttpGet(url);

CloseableHttpResponse closeableResponse = **null**;

**try** {

closeableResponse = closeableHttpClient.execute(httpGet);

**if** (closeableResponse.getStatusLine().getStatusCode() == 200)

{

HttpEntity entity = closeableResponse.getEntity();

String json = EntityUtils.*toString*(entity, "utf-8");

JsonParser jsonParser = **new** ~~JsonParser~~();

JsonElement jsonElement = jsonParser.~~parse~~(json);

JsonObject jsonObject = jsonElement.getAsJsonObject();

String msgJson = jsonObject.get("translation").getAsString();

fanyi = msgJson;

}

**return** fanyi;

} **finally** {

**if** (closeableResponse != **null**) {

closeableResponse.close();

}

closeableHttpClient.close();

}

}

**public** String createUrl(){

String url="";

Map<String, String> params = **new** HashMap<String,String>();

String salt = String.*valueOf*(System.*currentTimeMillis*());

String curtime = String.*valueOf*(System.*currentTimeMillis*() / 1000);

params.put("curtime", curtime);

String signStr = APP\_KEY + truncate(query) + salt + curtime + APP\_SECRET;

String sign = getDigest(signStr);

params.put("q", query);

params.put("from", from);

params.put("to", to);

params.put("signType", "v3");

params.put("sign", sign);

params.put("salt", salt);

params.put("appKey", APP\_KEY);

url = getUrlWithQueryString(YOUDAO\_URL,params);

**return** url;

}

**public** String getUrlWithQueryString(String url,

Map<String, String> params) {

**if** (params == **null**) {

**return** url;

}

StringBuilder builder = **new** StringBuilder(url);

**if** (url.contains("?")) {

builder.append("&");

} **else** {

builder.append("?");

}

**int** i = 0;

**for** (String key : params.keySet()) {

String value = params.get(key);

**if** (value == **null**) { // 过滤空的key

**continue**;

}

**if** (i != 0) {

builder.append('&');

}

builder.append(key);

builder.append('=');

builder.append(encode(value));

i++;

}

**return** builder.toString();

}

**public** String encode(String input) {

**if** (input == **null**) {

**return** "";

}

**try** {

**return** URLEncoder.*encode*(input, "utf-8");

} **catch** (UnsupportedEncodingException e) {

e.printStackTrace();

}

**return** input;

}

/\*\*

\* 生成加密字段

\*/

**public** String getDigest(String string) {

**if** (string == **null**) {

**return** **null**;

}

**char** hexDigits[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

**byte**[] btInput = string.getBytes(StandardCharsets.*UTF\_8*);

**try** {

MessageDigest mdInst = MessageDigest.*getInstance*("SHA-256");

mdInst.update(btInput);

**byte**[] md = mdInst.digest();

**int** j = md.length;

**char** str[] = **new** **char**[j \* 2];

**int** k = 0;

**for** (**byte** byte0 : md) {

str[k++] = hexDigits[byte0 >>> 4 & 0xf];

str[k++] = hexDigits[byte0 & 0xf];

}

**return** **new** String(str);

} **catch** (NoSuchAlgorithmException e) {

**return** **null**;

}

}

**public** String truncate(String q) {

**if** (q == **null**) {

**return** **null**;

}

**int** len = q.length();

String result;

**return** len <= 20 ? q : (q.substring(0, 10) + len + q.substring(len - 10, len));

}

}

### 3.3.2 MD5Util.java

该类主要实现对数据库中的用户密码进行MD5加密，描述该文件作用，并对里面的关键代码进行解释

**public** **class** MD5Util {

**public** **static** String md5(String str) {

**byte**[] digest = **null**;

**try** {

MessageDigest md5 = MessageDigest.*getInstance*("md5");

digest = md5.digest(str.getBytes("utf-8"));

} **catch** (NoSuchAlgorithmException e) {

e.printStackTrace();

} **catch** (UnsupportedEncodingException e) {

e.printStackTrace();

}

//16是表示转换为16进制数

String md5Str = **new** BigInteger(1, digest).toString(16);

**return** md5Str;

}

**public** **static** **void** main(String[] args) {

System.***out***.println(*md5*("admin"));

}

}

## 3.4 fanyiController.java

接收前端数据，调用服务类处理数据，返回处理结果

**package** controller;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.HashMap;

**import** java.util.Map;

**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** service.fanyiHelper;

**import** com.google.gson.Gson;

@WebServlet(urlPatterns = "/fanyi.do")

**public** **class** fanyiController **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charset=utf-8");

String query = request.getParameter("query");

String from = request.getParameter("from");

String to = request.getParameter("to");

fanyiHelper fyh = **new** fanyiHelper(query, "zh-CHS", "en");

String url = fyh.createUrl();

String translation;

**try** {

translation = fyh.getFanyiFromUrl(url);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

translation = e.getMessage();

}

// Map存放放回的信息

Map<String, Object> map = **new** HashMap<String, Object>();

map.put("translation", translation);

String jsonStr = **new** Gson().toJson(map);

// 字符流输出字符串

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

out.print(jsonStr);

out.flush();

out.close();

}

**public** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charset=utf-8");

String query = request.getParameter("query");

String from = request.getParameter("from");

String to = request.getParameter("to");

fanyiHelper fyh = **new** fanyiHelper(query, from, to);

String url = fyh.createUrl();

String translation;

**try** {

translation = fyh.getFanyiFromUrl(url);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

translation = e.getMessage();

}

// Map存放放回的信息

Map<String, Object> map = **new** HashMap<String, Object>();

map.put("translation", translation);

String jsonStr = **new** Gson().toJson(map);

// 字符流输出字符串

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

out.print(jsonStr);

out.flush();

out.close();

}

}

## 3.9 main.js

前端页面的一些函数，监听前端用户的动作

function putAnswer(answer){

$("#transTargetText").text(answer);

}

function clearAnswer(){

$("#transTargetText").text("");

}

function translate(){

clearAnswer();

var fromto = $("span.select-text").attr("id");

var str = fromto.split("2");

var from = str[0];

var to = str[1];

var inputOriginal = document.getElementById("inputOriginal").value;

$.ajax({

url: "fanyi.do",

type: "post",

data: {query:inputOriginal,from:from,to:to},

dataType: "json",

success: function(response) {

putAnswer(response.translation);

}

});

}

$(document).ready(function() {

$("#inputOriginal").focus(function(){

$(this).css("border","none");

$("#fanyi\_\_input\_\_bg").css("border","1px solid #7fa1c8");

$("div.input\_\_original\_\_bar").css("visibility","visible");

});

$('#inputOriginal').on('textarea focus keyup',function(){

var strs = $(this).val();

remain = strs.length;

var content\_msg = remain;

$("span.fonts\_\_over").html(content\_msg);

//.next() 方法返回被选元素的后一个同级元素。

var maxlength = $("span.fonts\_\_limited").html();

if(strs.length>maxlength){

$("span.fonts\_\_over").html(maxlength);

alert("已到达最大输入值");

}

});

$("#transMachine").click(function(){

translate();

});

$("#langSelect").click(function(){

if($("#languageSelect").css("display")=="none"){

$("#languageSelect").css("display","block");

}else{

$("#languageSelect").css("display","none");

}

});

$("#langSelect").on("click","ul li",function(){

$("span.select-text").text($(this).text());

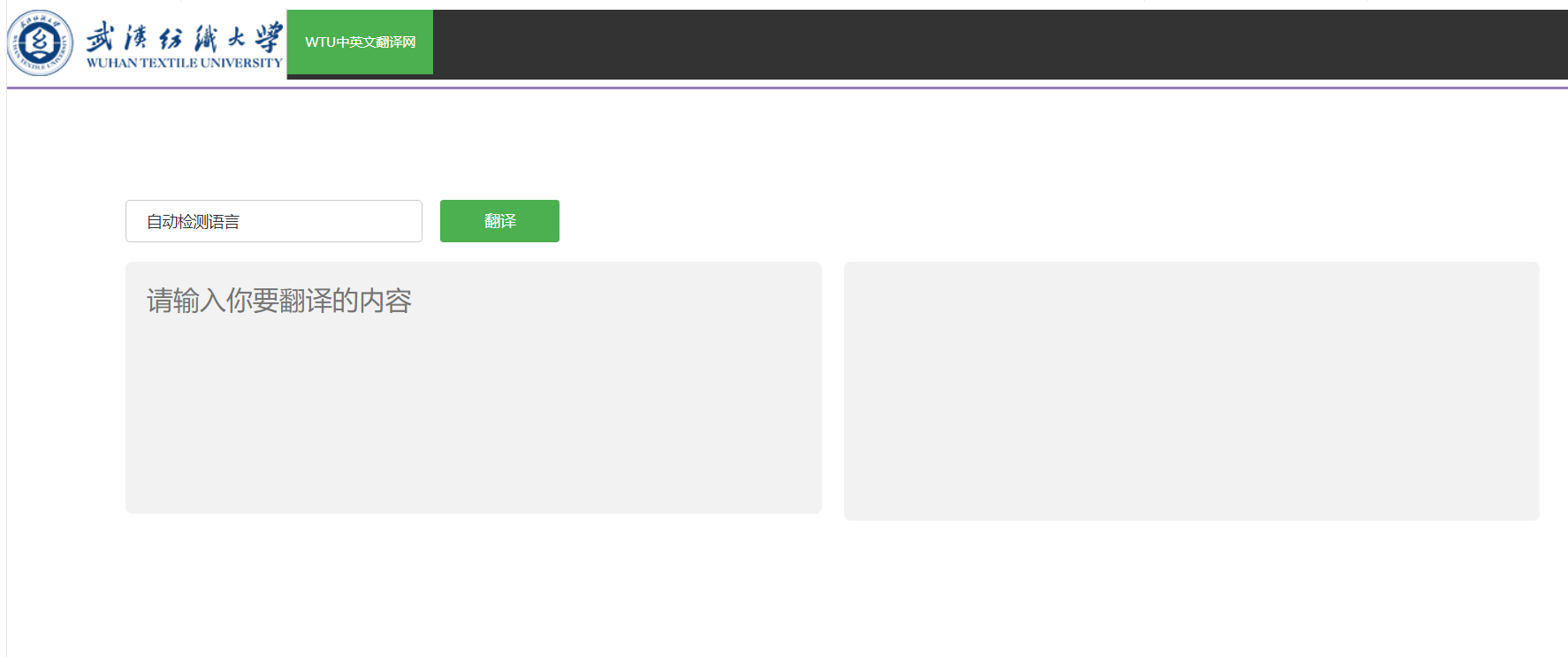
$("span.select-text").attr("id",$(this).attr("data-value"));

});

});

# **4 系统测试**

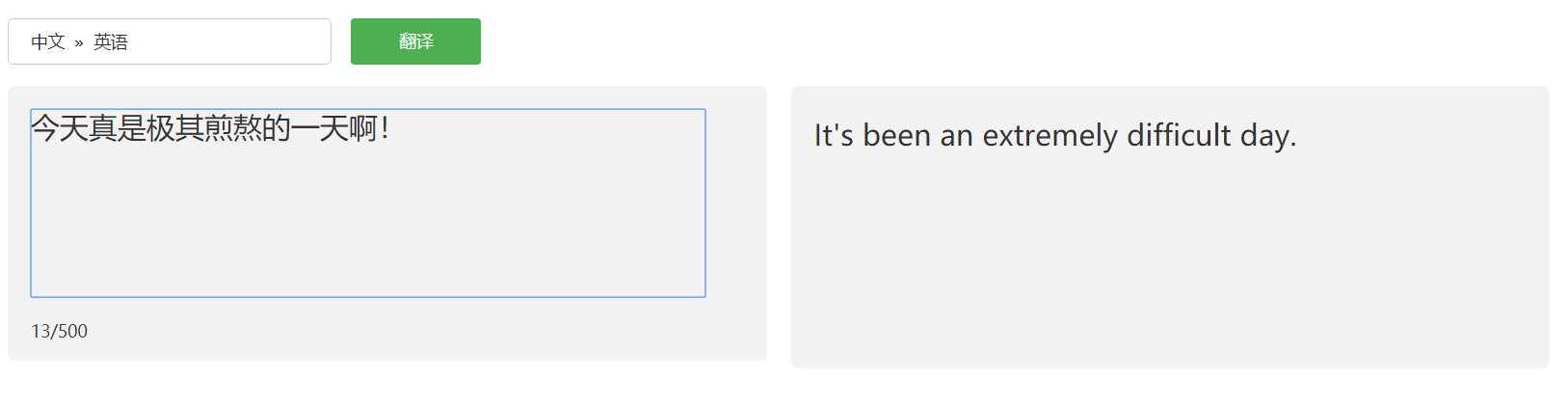
**4.1进入主界面**



**4.2选择语言**

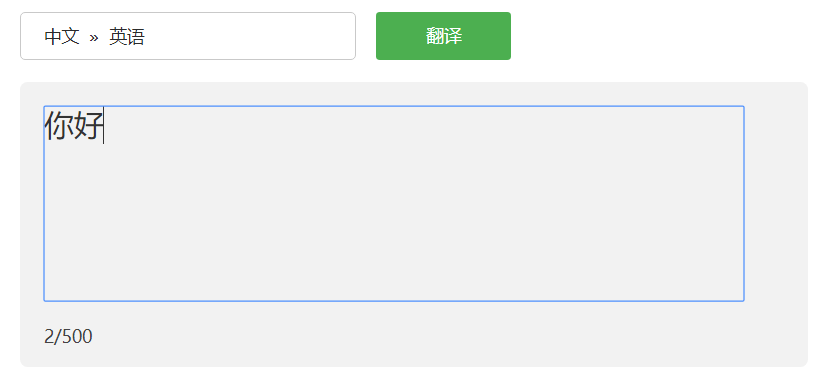


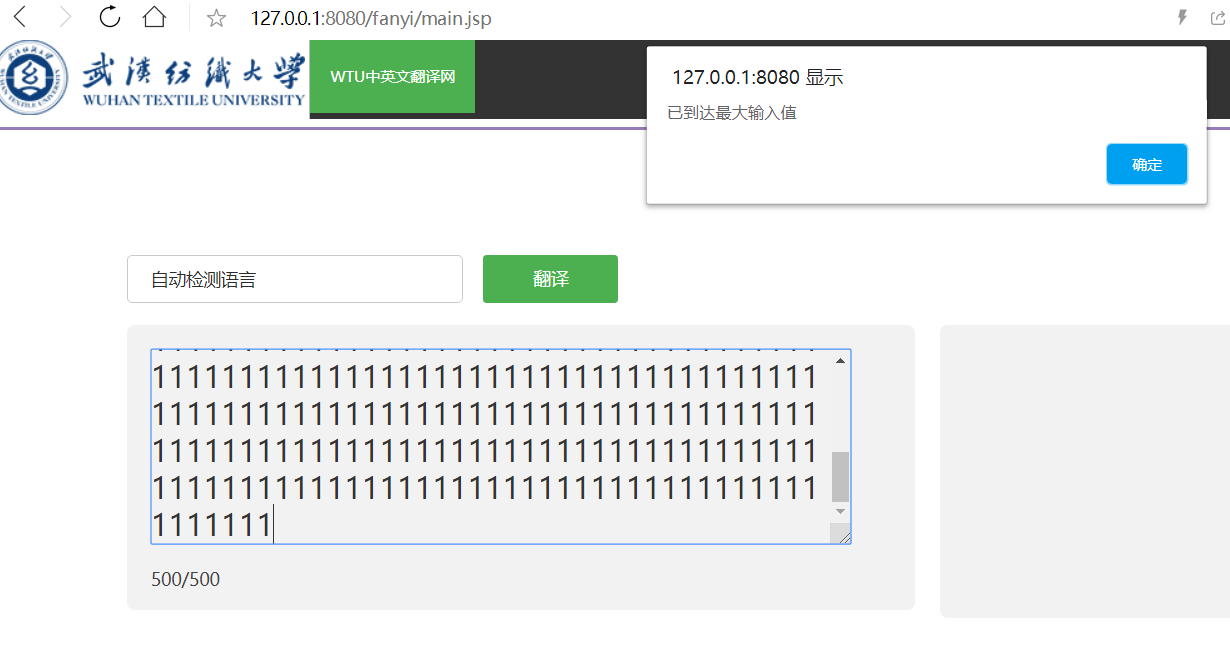
**4.3输入要翻译的文本**



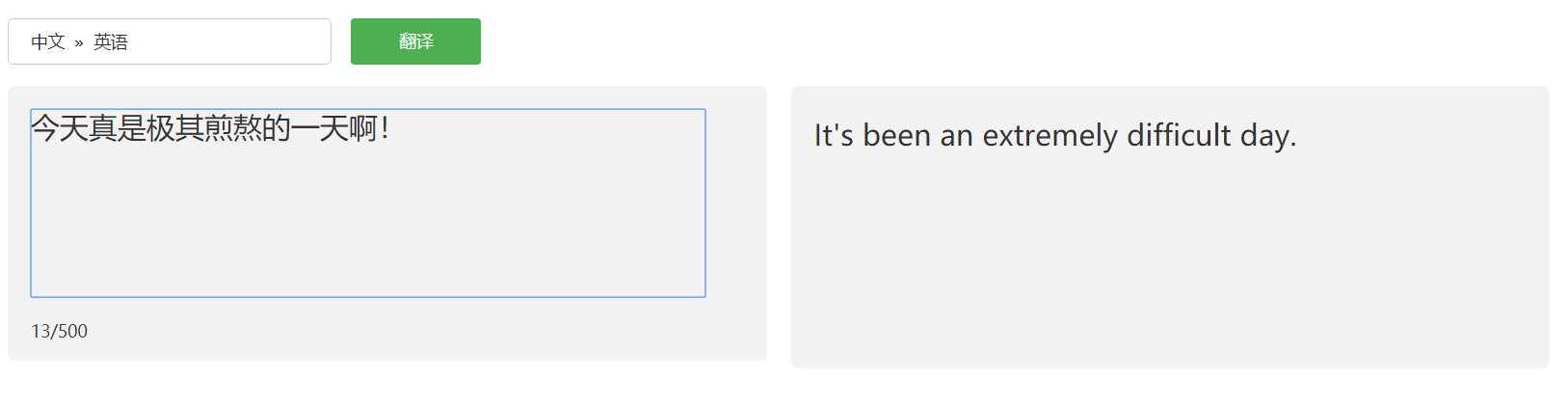


**4.4统计字数**



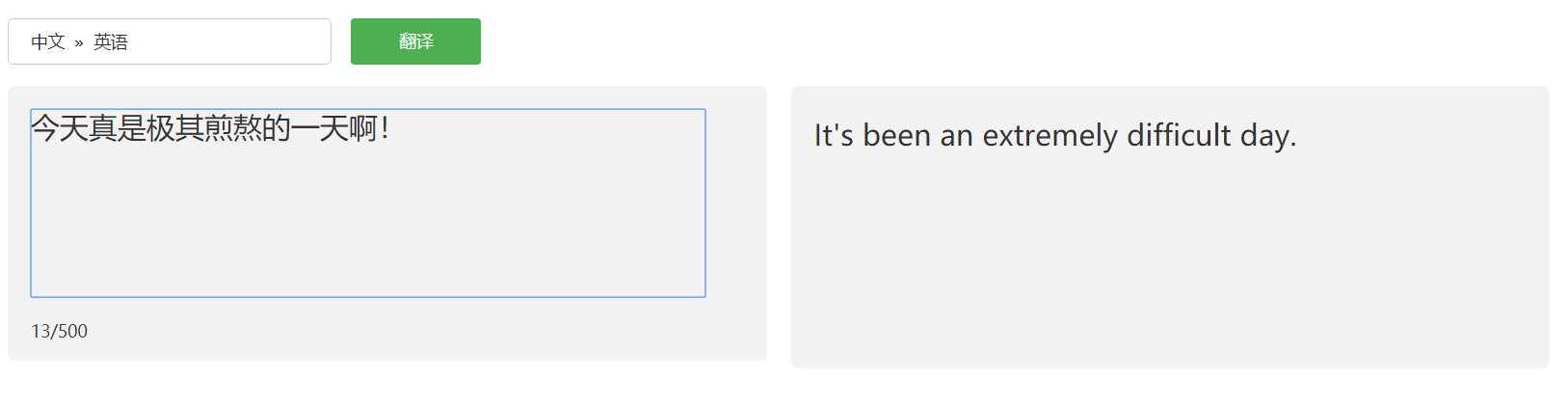


**4.5点击翻译按钮进行翻译**





**4.6显示翻译结果**





# **5 系统总结**

**主要问题**：主这个题目整体的目标要求和实现过程是非常清楚明白的，并没有用到什么框架之类的，主要就是前段页面和后端控制台之间数据的一些传送和接收，开始的时候花了一些时间在页面的界面上，通过检查有道翻译官网的页面源码得到了一些启发，主体也很简单，就是在**写js函数的时候会碰到一些不知道的函数**，经常查阅菜鸟教程等来找实现的方法，对整个JavaScript也有了更深的理解和使用，遇到最大的问题，就是**使用的MyEclipse中无法配置tomcat，这就导致了很多细节后端的问题不知道出在哪里**，只能通过前端返回的错误列表来慢慢找，前端可能就给一个500错误，在这里耽误了很多时间，最后发现是**使用CloseableHttpClient等方法时没有配置相关的liberary**，其实是个小问题但是卡了很久，系统不足的可能更多在于没有精化前端的细节吧，css包括jsp中的大体框架都很乱，想到哪写哪，没有一个大概的方向和架构，还是对前端开发不熟练，也希望能尽可能多的在考研之前多学习一些这方面知识。

**课程收获**：老师这学期教的一些MVC Spring以及SSM框架之类的都很符合实际工作需求，对未来工作有很多帮助和启发，从最开始写一个联网的网页到后面实现的功能越来越多、复杂，从开始的tomcat配置到后来SSM框架得配置，都让我发现还有很多不足不懂不熟练的地方，对自己的要求也相应的提高了，受益匪浅！

**课程建议**：Java web这门课其实学起来感觉老是您讲的多的是一些概念问题，虽然也有很多实际运用的例子，但是由于一次上课时间太长，不可能做到一直讲操作，会导致跟不上，但是大片大片讲概念也会让人很枯燥，注意力容易分散，就是这学期虽然是当面上课但是感觉上学期那种短时间精内容，课后还有详细文档和录屏回顾的方式学习效率更高，我个人这学期的感受是这样的。

**发展方向：**是想考研的，但是近年来随着考研难度的加大，对自己能考上的信心还不是很足，还是想试一试搏一搏，课上教的东西肯定还是首位，未来不管是那条路，这些知识是肯定能用得上的。

**自评成绩：90**