武汉纺织大学

Web应用开发课程设计

**基于SSM的在线翻译**

**学 院： 数学与计算机学院**

**班 级： 物联网11802**

**姓 名： 梁少伟**

**学 号： 1802220113**

**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](#_Toc59986598)

[1.1普通文本翻译 1](#_Toc59986599)

[1.2图片翻译 1](#_Toc59986600)

[1.3语音翻译 1](#_Toc59986601)

[2 系统设计 1](#_Toc59986602)

[2.1用户用例图 1](#_Toc59986603)

[2.2 ER图 1](#_Toc59986604)

[2.3 UML类图（Class Diagram） 2](#_Toc59986605)

[2.4 UML时序图（Sequence Diagram） 3](#_Toc59986606)

[2.4.1 语言的导入 3](#_Toc59986607)

[2.4.2 普通文本翻译 4](#_Toc59986608)

[2.4.3 图片翻译 4](#_Toc59986609)

[2.4.4 语音翻译 4](#_Toc59986610)

[2.5 UML活动图（Activity Diagram） 4](#_Toc59986611)

[2.5.1 获取本地数据库的国家 4](#_Toc59986612)

[2.5.2 普通文本翻译 5](#_Toc59986613)

[2.5.3 图片翻译 5](#_Toc59986614)

[2.5.4 语音翻译 6](#_Toc59986615)

[3 系统实现 7](#_Toc59986616)

[3.1 项目结构 7](#_Toc59986617)

[3.2 配置文件 7](#_Toc59986618)

[3.2.1 jdbc.properties文件 7](#_Toc59986619)

[3.2.2 log4j.properties文件 8](#_Toc59986620)

[3.2.3 mybatis.xml文件 9](#_Toc59986621)

[3.2.4 spring-mvc.xml文件 9](#_Toc59986622)

[3.2.5 spring-mybatis.xml文件 10](#_Toc59986623)

[3.2.6 pom.xml文件 12](#_Toc59986624)

[3.2.7 web.xml文件 16](#_Toc59986625)

[3.3 VO类CountryToCode.java 18](#_Toc59986626)

[3.4 DAO接口类ICountryToCodeDAO.java 18](#_Toc59986627)

[3.5 接口映射文件CountryToCodeMapping.xml 19](#_Toc59986628)

[3.6 Service接口类 ICountryService.java 19](#_Toc59986629)

[3.7 Service.impl实现类 ICountryService.java 20](#_Toc59986630)

[3.8 com.baidu.translate.demo 百度官方包 21](#_Toc59986631)

[3.9 youdao 有道官方包 21](#_Toc59986632)

[3.10 lsw.html 主页面 21](#_Toc59986633)

[3.11 main.css 主页样式文件 23](#_Toc59986634)

[3.12 js文件 28](#_Toc59986635)

[4 系统测试 32](#_Toc59986636)

[4.1普通文本翻译 32](#_Toc59986637)

[4.2图片翻译 33](#_Toc59986638)

[4.3语音翻译 34](#_Toc59986639)

[5 系统总结 34](#_Toc59986640)

# 1 需求分析

设计一个在线翻译网页，实现30种语言的相互翻译转化，并且向前端上传本地的图片和音频，翻译其图片和音频等功能，具体要求如下：

## 1.1普通文本翻译

当前端运行时，显示“请选择原语言”和“请选择目的语言”，然后选择相应的语言后，输入你要翻译的文本，然后在右边的另一方文本框中就会显示出相应的翻译后的文本。要翻译的文本有字数限制，在此我设置了最多输入30个字。在输入框的右侧设置了一键清空按钮。

## 1.2图片翻译

点击选择图片后，会弹出一个文件资源管理器，选择一张本地要翻译的图片上传到前端，然后后端会进行相应的处理，然后把翻译后的文本显示到前端，显示在右边的文本框中。

## 1.3语音翻译

点击选择音频按钮后，会弹出一个文件资源管理器，选择一个要翻译的语音（在此只支持wav格式的音频），点击后会把音频发送到后端，后端把翻译后的文本再上传到前端中，显示在右边的文本框中。

# 2 系统设计

## 2.1用户用例图

用户用例如下图2-1所示：

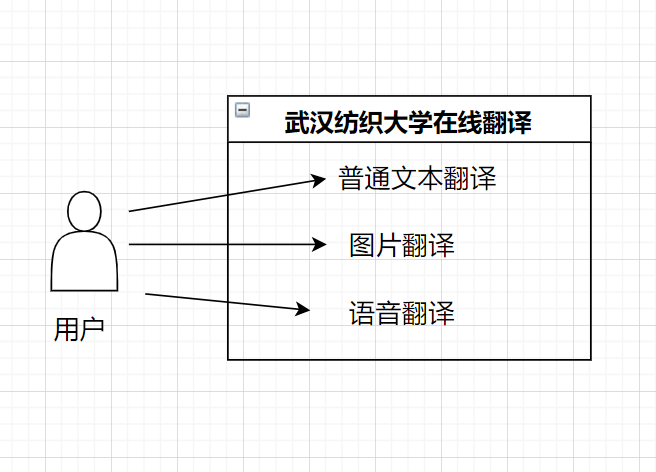
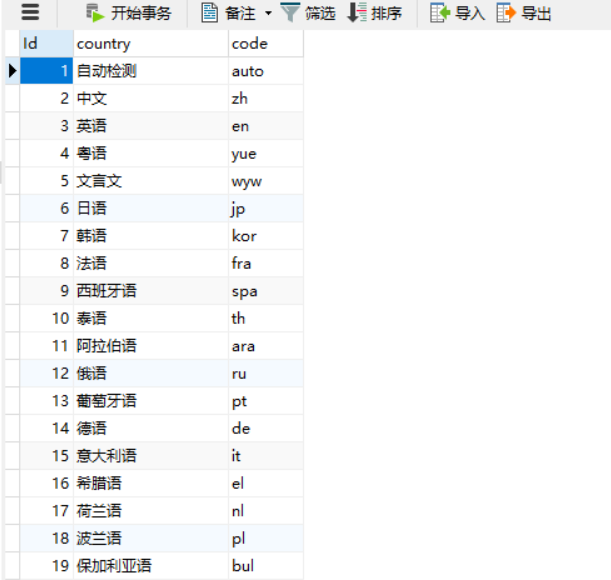
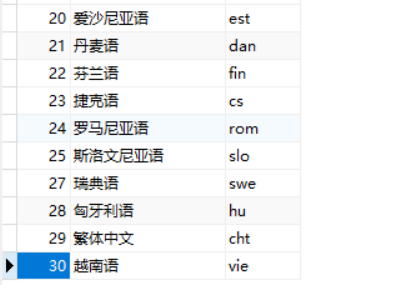


图2-1 用户用例图

## 2.2 ER图

数据库设计了一张表，语言对应的Code表，对应的ER图如图3-1所示。





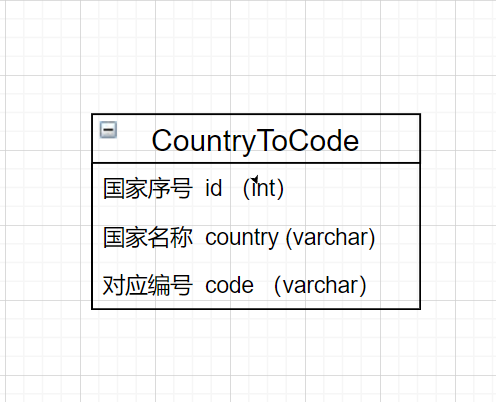
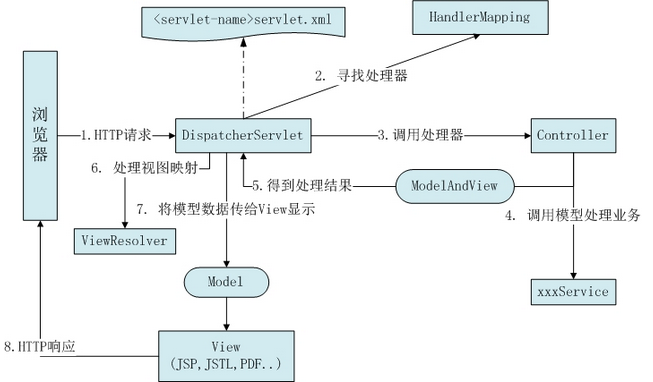


图3-1 数据库ER图

## 2.3 UML类图（Class Diagram）

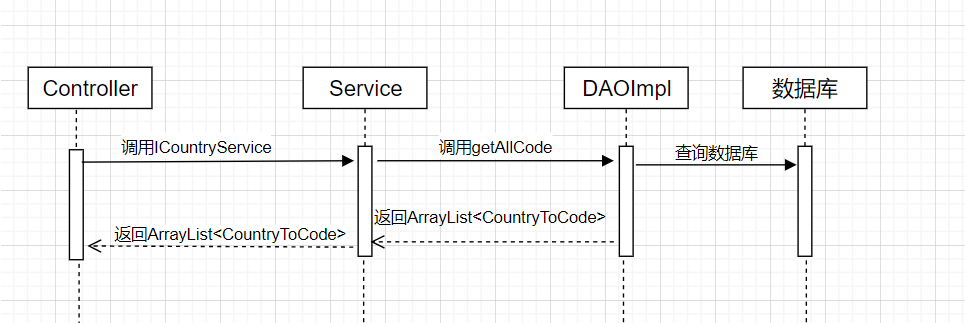
对于用户功能模块，共设计如下6个类。

* Jdbc.properties:数据库配置文件。
* VO实体类User：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录
* DAO接口ICountryToCodeDAO：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。
* DAO实现类UserDAOImpl：DAO接口的真实实现类，主要完成具体数据库操作，但不负责数据库的打开和关闭。
* 业务逻辑接口service：主要定义了业务逻辑的一些接口。
* 业务逻辑实现类CountryServiceImpl：service的真正实现类，主要完成业务逻辑上面的一些操作。
* com.baidu.translate.demo:百度官方demo，可以直接调用。
* youdao：有道官方demo，可以直接调用。
* log4j2：日志打印的配置文件
* spring-mvc：springmvc框架的相关配置，主要配置扫描类包和配置视图解析器
* spring-mybatis：mybatis框架的相关配置。

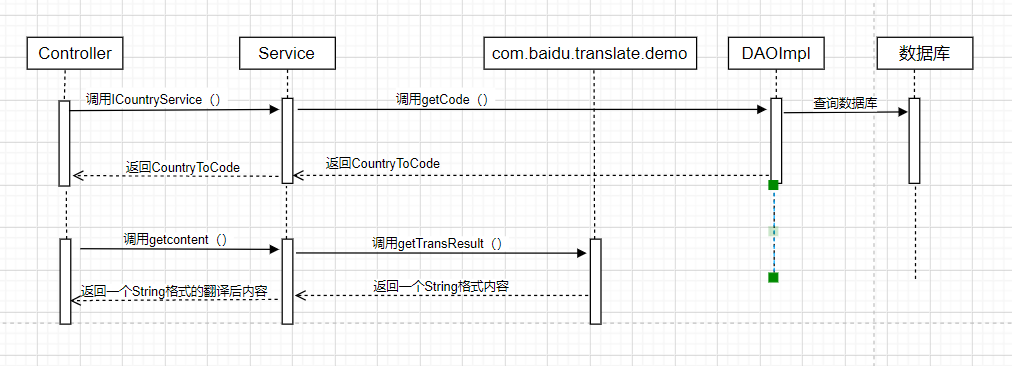


## 2.4 UML时序图（Sequence Diagram）

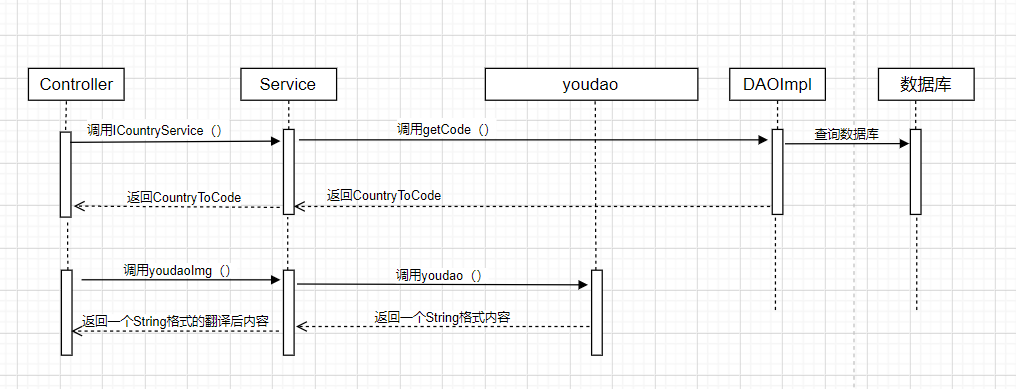
### 2.4.1 语言的导入



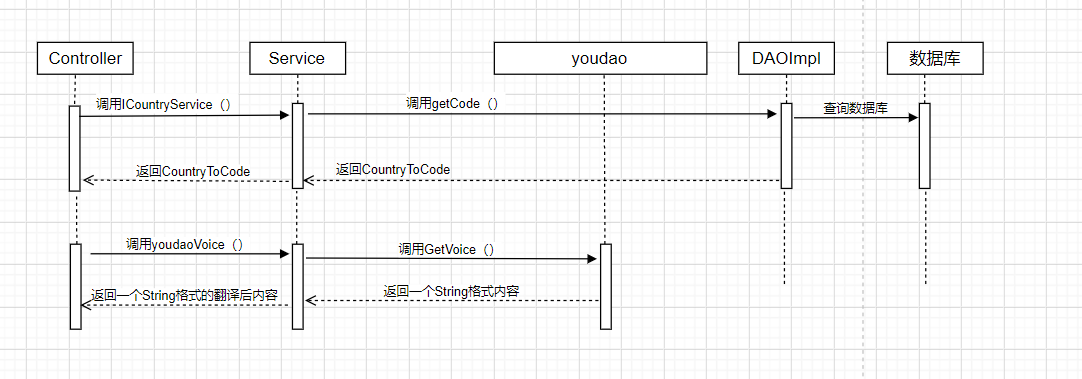
### 2.4.2 普通文本翻译



### 2.4.3 图片翻译



### 2.4.4 语音翻译



## 2.5 UML活动图（Activity Diagram）

### 2.5.1 获取本地数据库的国家

登录翻译页面后，发起ajax请求，获取到数据库中的国家队列，然后循环加入到option中，具体流程如图2-6所示。

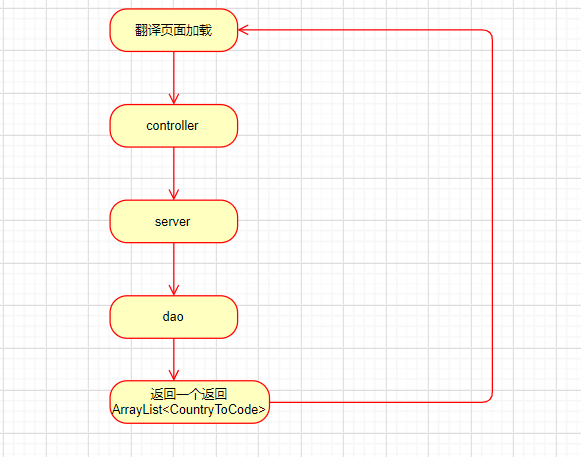


图2-2 获取本地数据库中的国家

### 2.5.2 普通文本翻译

前端把选中的源语言，目的语言，翻译文本传回到后端，然后后端把两个语言转化为相应的编码，然后把这三个参数传输到百度给的demo中，得到一个json的数据，然后把这个数据传输到前端，前端做出相应的解析并且显示出来。

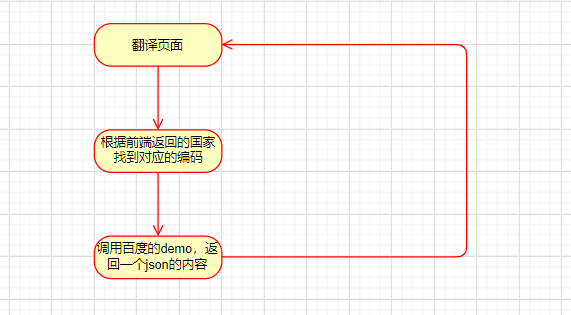
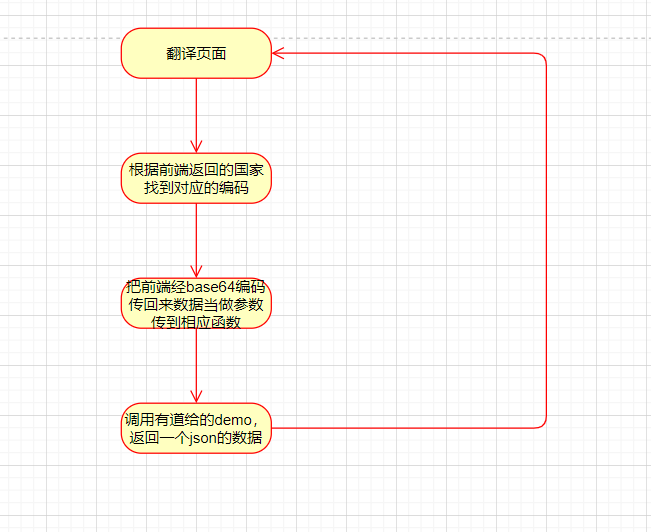


图2-5 普通文本翻译图

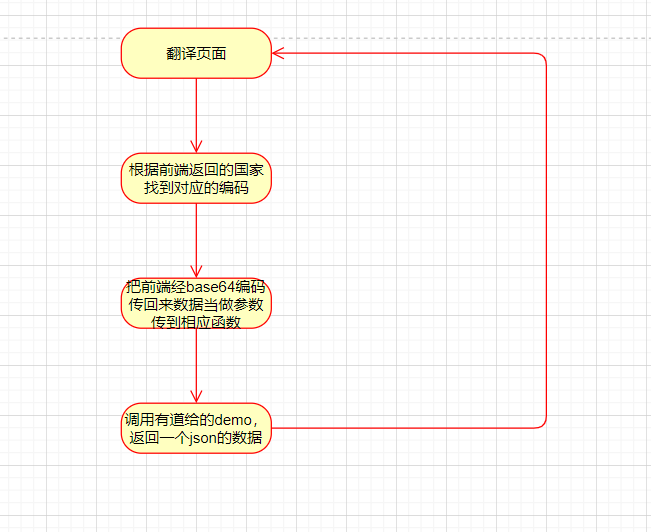
### 2.5.3 图片翻译

前端把选中的图片进行base64编码后传到后端，然后把参数传送到有道给的demo中，返回一个json格式的内容，然后传到前端做出相应的处理。



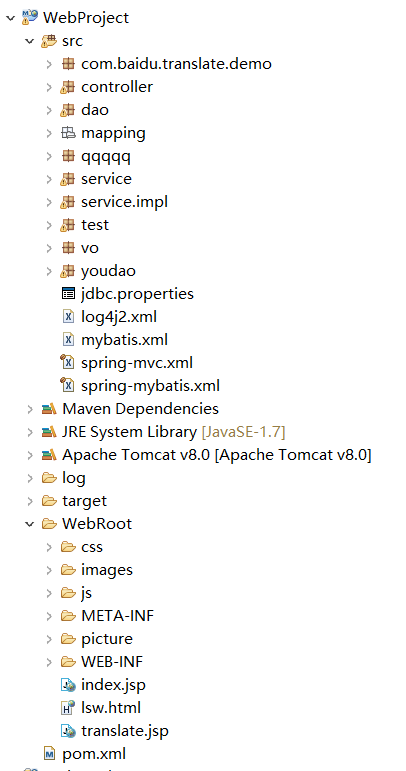
### 2.5.4 语音翻译

前端把选中的图片进行base64编码后传到后端，然后把参数传送到有道给的demo中，返回一个json格式的内容，然后传到前端做出相应的处理。



# 3 系统实现

## 3.1 项目结构



## 3.2 配置文件

### 3.2.1 jdbc.properties文件

该配置文件主要是以文件形式保存数据库的驱动类名称，连接数据库的URL地址，访问数据库的用户名及对应的密码，程序运行时会读取该文件相关信息，避免硬编码，当相关信息发生变化时，只需修改配置文件而不用修改源代码，增加程序的可扩展型。

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/translate?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=UTC

username=root

password=123456

#\u5B9A\u4E49\u521D\u59CB\u8FDE\u63A5\u6570

initialSize=10

#\u5B9A\u4E49\u6700\u5927\u8FDE\u63A5\u6570

maxActive=20

#\u5B9A\u4E49\u6700\u5C0F\u7A7A\u95F2

minIdle=1

#\u5B9A\u4E49\u6700\u957F\u7B49\u5F85\u65F6\u95F4

maxWait=60000

#\u9A8C\u8BC1\u8FDE\u63A5\u662F\u5426\u53EF\u7528\uFF0C\u4F7F\u7528\u7684SQL\u8BED\u53E5

validationQuery =SELECT 1

### 3.2.2 log4j.properties文件

描述该文件作用，并对里面的配置进行解释

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

<!--日志级别以及优先级排序: OFF > FATAL > ERROR > WARN > INFO > DEBUG > TRACE > ALL -->

<!-- status配置Log4j2启动和加载配置文件时的日志输出登记 -->

<Configuration status="info">

<!--appenders:定义输出内容,输出格式,输出方式,日志保存策略等,常用其下三种标签[console,File,RollingFile] -->

<!--Appender可以理解为日志的输出目的地 -->

<Appenders>

<!--console :控制台输出的配置 -->

<Console name="Console" target="SYSTEM\_OUT">

<!--

%d 输出日志时间点的日期或时间

%t 输出产生该日志事件的线程名

%p 输出优先级，即DEBUG,INFO,WARN,ERROR,FATAL

%F 输出日志消息产生时所在的文件名称

%c 输出所属的类目,通常就是所在类的全名

%L 输出代码中的行号

-->

<PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} %-5p [%c,%L]- %msg%n" />

</Console>

<RollingFile name="RollingFile" filename="log/ZycTest.log"

filepattern="${logPath}/%d{YYYYMMddHHmmss}-fargo.log">

<PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} %-5p [%c,%L] - %msg%n" />

<Policies>

<SizeBasedTriggeringPolicy size="100 MB" />

</Policies>

<DefaultRolloverStrategy max="20" />

</RollingFile>

</Appenders>

<!--定义logger，只有定义了logger并引入的appender，appender才会生效 -->

<Loggers>

<!-- Logger节点用来单独指定日志的形式，可以为通过name属性设置指定包下的class指定不同的日志级别等 ,

可以设置Logger的additivity="false"只在自定义的Appender中进行输出 -->

<logger name="dao" level="debug" additivity="false">

<appender-ref ref="Console" />

</logger>

<!-- Root节点用来指定项目的根日志，如果没有单独指定Logger，那么就会默认使用该Root日志输出 -->

<Root level="info">

<AppenderRef ref="Console" />

<AppenderRef ref="RollingFile" />

</Root>

</Loggers>

</Configuration>

### 3.2.3 mybatis.xml文件

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

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<settings>

<setting name="logImpl" value="LOG4J2" />

</settings>

</configuration>

### 3.2.4 spring-mvc.xml文件

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd">

<mvc:annotation-driven />

<!-- 扫描类包，将标注Spring注解的类自动转化Bean，同时完成Bean的注入 -->

<context:component-scan base-package="controller" />

<context:component-scan base-package="service" />

<!--配置视图解析器 -->

<!-- ViewResolver 视图解析器 用于将返回的ModelAndView对象进行分离

InternalResourceViewResolver：用于支持Servlet、JSP视图解析；

viewClass：JstlView表示JSP模板页面需要使用JSTL标签库，classpath中必须包含jstl的相关jar包； prefix和suffix：查找视图页面的前缀和后缀（前缀[逻辑视图名]后缀），

比如传进来的逻辑视图名为hello，则该该jsp视图页面应该存放在“/hello.jsp”； -->

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/"></property> <!-- 视图放在webroot/下 -->

<property name="suffix" value=".jsp"></property>

</bean>

</beans>

### 3.2.5 spring-mybatis.xml文件

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<!-- 扫描类包，将标注Spring注解的类自动转化Bean，同时完成Bean的注入 -->

<context:component-scan base-package="service" />

<!-- 引入jdbc配置文件 -->

<bean id="propertyConfigurer"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="classpath:jdbc.properties" />

</bean>

<!-- 数据库连接池 -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"

destroy-method="close">

<property name="driverClassName" value="${driver}" />

<property name="url" value="${url}" />

<property name="username" value="${username}" />

<property name="password" value="${password}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="${initialSize}"></property>

<!-- 连接池最大数量 -->

<property name="maxActive" value="${maxActive}"></property>

<!-- 连接池最小空闲 -->

<property name="minIdle" value="${minIdle}"></property>

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="${maxWait}"></property>

<!-- 验证连接是否可用，使用的SQL语句 -->

<property name="validationQuery" value="SELECT 1"></property>

</bean>

<!-- spring和MyBatis整合，不需要mybatis的配置映射文件 -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="configLocation" value="classpath:mybatis.xml"></property>

<!-- 自动扫描mapping.xml文件 -->

<property name="mapperLocations" value="classpath\*:mapping/\*.xml"></property>

</bean>

<!-- DAO接口所在包名，Spring会自动查找其下的类 -->

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="dao" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>

</bean>

<!-- (事务管理)transaction manager-->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

</beans>

### 3.2.6 pom.xml文件

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>WebProject</groupId>

<artifactId>WebProject</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>WebProject</name>

<description/>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<springversion>5.0.0.RELEASE</springversion> <!-- spring版本号 -->

<mybatisversion>3.3.0</mybatisversion> <!-- mybatis版本号 -->

<log4jversion>2.11.2</log4jversion> <!-- log4j版本号 -->

</properties>

<!-- 项目所需依赖包 -->

<dependencies>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.0.1</version>

<!-- 用于编译和测试的环境，将不会被打包到lib目录下。 -->

<scope>provided</scope>

</dependency>

<!--springframework -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${springversion}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${springversion}</version>

</dependency>

<!-- log4j2，用于日志管理 -->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>${log4jversion}</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>${log4jversion}</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-web</artifactId>

<version>${log4jversion}</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.1</version>

</dependency>

<!-- junit单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<!-- 表示开发的时候引入，发布的时候不会加载此包 -->

<scope>test</scope>

</dependency>

<!-- 用于spingMVC中的java对象和json自动转换 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.1</version>

</dependency>

<!-- mybatis核心包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatisversion}</version>

</dependency>

<!-- mybatis/spring包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<!--mysql驱动包 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.49</version>

</dependency>

<!-- 连接池，用来加载DruidDataSource -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.0.9</version>

</dependency>

<!-- JSTL标签库 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<resources>

<resource>

<directory>src</directory>

<excludes>

<exclude>\*\*/\*.java</exclude>

</excludes>

</resource>

</resources>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.6</version>

<configuration>

<warSourceDirectory>${basedir}/WebRoot</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

</project>

### 3.2.7 web.xml文件

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>WebProject</display-name>

<welcome-file-list>

<welcome-file>translate.jsp</welcome-file>

</welcome-file-list>

<!-- Spring和mybatis的配置文件 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mybatis.xml</param-value>

</context-param>

<!-- log4j2的配置文件 -->

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:log4j2.xml</param-value>

</context-param>

<!-- 编码过滤器 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<async-supported>true</async-supported>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- Spring监听器 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 防止Spring内存溢出监听器 -->

<listener>

<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>

</listener>

<servlet>

<servlet-name>SpringMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mvc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

<async-supported>true</async-supported>

</servlet>

<servlet-mapping>

<servlet-name>SpringMVC</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

</web-app>

## 3.3 VO类CountryToCode.java

public class CountryToCode {

private int id;

private String country;

private String code;

public CountryToCode() {

super();

}

public CountryToCode(int id, String country, String code) {

super();

this.id = id;

this.country = country;

this.code = code;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

}

## 3.4 DAO接口类ICountryToCodeDAO.java

package dao;

import java.util.ArrayList;

import org.apache.ibatis.annotations.Param;

import vo.CountryToCode;

public interface ICountryToCodeDAO {

///查找数据库中所有数据，为下拉框做出准备

public ArrayList<CountryToCode> getAll();

///按照中文名查找，找出对应的Code

public CountryToCode getByCountry(String Country);

}

## 3.5 接口映射文件CountryToCodeMapping.xml

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

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="dao.ICountryToCodeDAO">

<select id="getAll" resultType="vo.CountryToCode">

select \* from t\_country

</select>

<select id="getByCountry" resultType="vo.CountryToCode" >

select \* from t\_country where country=#{country}

</select>

</mapper>

## 3.6 Service接口类 ICountryService.java

package service;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Map;

import org.springframework.stereotype.Service;

import com.sun.tools.javac.util.Context.Key;

import vo.CountryToCode;

public interface ICountryService {

//得到所有的Code

public ArrayList<CountryToCode> getAllCode() ;

//得到某个Code

public CountryToCode getCode(String country);

//取到翻译内容

public String getcontent(String in,String out,String outputcontent);

//图片翻译

public String youdaoImg(String urlAddress) ;

//语音翻译

public String youdaoVoice(String urlAddress);

}

## 3.7 Service.impl实现类 ICountryService.java

package service.impl;

import java.io.IOException;

import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.baidu.translate.demo.TransApi;

import dao.ICountryToCodeDAO;

import service.ICountryService;

import vo.CountryToCode;

@Service

public class CountryServiceImpl implements ICountryService{

@Autowired

private ICountryToCodeDAO dao;

@Override

public ArrayList<CountryToCode> getAllCode() {

// TODO Auto-generated method stub

return dao.getAll();

}

@Override

public CountryToCode getCode(String country) {

return dao.getByCountry(country);

}

@Override

public String getcontent(String in, String out, String outputcontent) {

// TODO Auto-generated method stub

TransApi api = new TransApi("20201214000647158", "nteLk8UfgilvaZ0lLE43");

String query=api.getTransResult(outputcontent, in, out);

return query;

}

@Override

public String youdaoImg(String urlAddress) {

System.out.println("asdjfkalsdfjadkls");

String contentImg=youdao.yuodaoapi.youdao(urlAddress);

return contentImg;

}

@Override

public String youdaoVoice(String urlAddress) {

String contentString=youdao.youdaoapiVoice.GetVoice(urlAddress);

return contentString;

}

}

## 3.8 com.baidu.translate.demo 百度官方包

百度给的官方demo，这里就不进行展示了

## 3.9 youdao 有道官方包

有道给的官方demo，这里就不进行展示了

## 3.10 lsw.html 主页面

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="/WebProject/js/jquery-3.5.0.min.js"></script>

<script src="/WebProject/js/main.js" type="text/javascript"></script>

<link rel="stylesheet" type="text/css" href="/WebProject/css/main.css" />

</head>

<body>

<div id="logo">

<img src="/WebProject/images/xuexiao.png">

</div>

<div id="divide-wrap">

</div>

<div id="head">

<div id="in">

<select name="incode" id="incode" style="width:120px; height: 30px; border-radius: 6px;">

<option value="">请选择原语种</option>

</select>

</div>

<div id="jiantou">

<img src="/WebProject/images/箭头.png" >

</div>

<div id="out">

<select name="outcode" id="outcode" style="width:130px; height: 30px; border-radius: 6px;">

<option value ="">请选择目的语种</option>

</select>

</div>

<div id="bt">

<input type="button" value="点击翻译" id="clickbutton">

</div>

<div id="localImag">

<a href="javascript:;" class="file">选择图片

<input type="file" name="img" id="docfile" onchange="setImagePreview();">

</a>

</div>

<div id="voice">

<a href="javascript:;" class="file">选择音频

<input type="file" name="img" id="docfileVoice" onchange="setImagePreview2();">

</a>

</div>

</div>

<div id="content">

<div id="from">

<textarea name="text" id="textarea" maxlength="30" cols="47" rows="10" oninput="font\_size()" ></textarea>

<div class="son">

<input type="button" id="delete" >

</div>

<div class="son2">

<p><span class="title2">0/30</span></p>

</div>

</div>

<div id="to">

<textarea name="text" id="textareaprint" style="resize: none;" cols="47" rows="10" readonly oninput="font\_size()"></textarea>

</div>

</div>

</body>

</html>

## 3.11 main.css 主页样式文件

@CHARSET "UTF-8";

body{

/\* background: linear-gradient(to left, rgba( 255 228 225), rgba(230 230 250)); \*/

}

#logo{

width: auto;

height: 80px;

margin-left: 40px;

}

#divide-wrap{

width: 100%;

height: 2px;

background: linear-gradient(to left,rgba(100 149 237),rgba(72 61 139));

}

#head{

height: 30px;

width: 80%;

/\* border:1px solid #ddd ; \*/

margin-left: 40px;

margin-top: 40px;

margin-bottom: 10px;

display: flex;

}

#in{

height: 30px;

width: 10%;

margin-right: 20px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

display: flex;

}

#jiantou{

height: 30px;

width: 2%;

margin-right: 20px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

display: flex;

}

#out{

height: 30px;

width: 10%;

margin-right: 10px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

display: flex;

}

#bt{

height: 30px;

width: 150px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

display: flex;

}

#clickbutton{

width:100px;

height: 30px;

background-color:#4395ff;

color: floralwhite;

text-align: center;

/\* text-decoration:blink; \*/

border-radius: 6px;

}

#localImag{

height: 30px;

width: 10%;

margin-right: 10px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

text-align: center;

position: relative;

display: flex;

}

#voice{

height: 30px;

width: 10%;

margin-right: 10px;

margin-bottom: 3px;

/\* border:1px #000000 solid; \*/

float: left;

text-align: center;

position: relative;

display: flex;

}

/\*

#docfile{

width:100px;

height: 30px;

background-color:floralwhite;

color: #4395ff;

text-align: center;

border-radius: 6px;

}\*/

#content{

height: 300px;

width: 75%;

margin-left: 40px;

margin-top: 10px;

margin-bottom: 10px;

}

#from{

height: 280px;

width: 45%;

padding: 10px;

border:1px solid #000000 ;

background-color:white;

float: left;

display: flex;

position: relative;

}

#from .son{

position: absolute;

width: 35px;

height: 20px;

right: 20px;

top: 100;

bottom: 200;

}

#delete{

width:35px;

height: 20px;

outline: none;

border: solid 0px;

padding: 0;

background:url("../images/delete2.svg");

background-size:100% 100%;

background-repeat:no-repeat;

}

#from .son2{

position: absolute;

width: 35px;

height: 35px;

color: black;

right: 20px;

bottom: 20px;

}

#to{

height: 280px;

width: 45%;

padding: 10px;

border:1px solid #000000 ;

float: left;

display: flex;

background-color: lightgray;

}

#textarea{

background-color:white;

font-size: 20px;

resize: none;

outline: none;

border: solid 0px;

}

#textareaprint{

background-color:lightgray;

font-size: 20px;

resize: none;

outline: none;

border: solid 0px;

readonly:readonly

}

/\*

.file {

position: relative;

display: inline-block;

background: white;

border: 5px solid #99D3F5;

border-radius: 6px;

padding: 4px 12px;

overflow: hidden;

color: #1E88C7;

text-decoration: none;

text-indent: 0;

font-size: 150px;

line-height: 20px;

border-radius: 6px;

}

.file input {

height: 30px;

position: absolute;

font-size: 150px;

right: 100px;

top: 100px;

opacity: 1;

}

.file:hover {

background: #AADFFD;

border-color: #78C3F3;

color: #004974;

text-decoration: none;

}\*/

.file {

position: relative;

display: inline-block;

background: #D0EEFF;

border: 1px solid #99D3F5;

border-radius: 4px;

padding: 4px 12px;

overflow: hidden;

color: #1E88C7;

text-decoration: none;

text-indent: 0;

line-height: 20px;

}

.file input {

position: absolute;

font-size: 100px;

right: 0;

top: 0;

opacity: 0;

}

.file:hover {

background: #AADFFD;

border-color: #78C3F3;

color: #004974;

text-decoration: none;

}

## 3.12 js文件

**function font\_size() {**

**var num = $("#textarea").val().length;**

**if(num<=30){**

**$(".title2").html('<span></span>/30');**

**$(".title2 span").text(num);**

**}**

**}**

**function setImagePreview() {**

**var docObj = document.getElementById("docfile");**

**// var imgObjPreview = document.getElementById("preview");**

**console.log(docObj.files[0]);**

**if (docObj.files && docObj.files[0]) {**

**var reader = new FileReader();**

**reader.readAsDataURL(docObj.files[0]);//发起异步请求**

**reader.onload = function(){**

**//读取完成后，数据保存在对象的result属性中**

**console.log(this.result)**

**// imgObjPreview.src=this.result;**

**var data={img:this.result};**

**$.ajax({**

**type:"POST",**

**url:"ajaxImg.do",**

**data:data,**

**dataType:"json",**

**success:function(response){**

**$("#textarea").val("");**

**$("#textareaprint").val("");**

**var content=response.resRegions;**

**var index;**

**for(index=0;index<content.length;index++){**

**if(content[index].tranContent != "\* ?"){**

**var pr=$("#textareaprint").val();**

**$("#textareaprint").val(pr+content[index].tranContent+"\r\n");**

**}**

**}**

**}**

**})**

**}**

**}**

**}**

**function setImagePreview2() {**

**var docObj2 = document.getElementById("docfileVoice");**

**// var imgObjPreview = document.getElementById("preview");**

**console.log(docObj2.files[0]);**

**if (docObj2.files && docObj2.files[0]) {**

**var reader = new FileReader();**

**reader.readAsDataURL(docObj2.files[0]);//发起异步请求**

**reader.onload = function(){**

**//读取完成后，数据保存在对象的result属性中**

**// console.log(this.result)**

**// imgObjPreview.src=this.result;**

**var data2={voice:this.result};**

**// console.log(this.result);**

**console.log(data2);**

**$.ajax({**

**type:"post",**

**url:"ajaxVoice.do",**

**data:data2,**

**dataType:"json",**

**success:function(response){**

**$("#textarea").val("");**

**$("#textareaprint").val("");**

**console.log(response);**

**var content=response.translation;**

**console.log(content);**

**$("#textareaprint").val(content+"\r\n");**

**}**

**})**

**}**

**}**

**}**

**$(document).ready(function(){**

**$("#delete").click(function(){**

**var q=$("#textarea").val();**

**console.log(q);**

**$("#textarea").val("");**

**$(".title2 span").text(0);**

**});**

**// console.log("lsw");**

**$.ajax({**

**type:"post",**

**url:"ajaxGetCountry.do",**

**data:{},**

**dataType:"json",**

**success:function(response){**

**var index;**

**var index2;**

**for(index=0;index<response.length;index++){**

**var option=$("<option>").val(response[index].id).text(response[index].country);**

**$("#incode").append(option);**

**// $("#outcode").append(option);**

**}**

**for(index2=1;index2<response.length;index2++){**

**var option=$("<option>").val(response[index2].id).text(response[index2].country);**

**$("#outcode").append(option);**

**}**

**}**

**});**

**// var data={incode:$("#incode option:selected").val(),outcode:$("#outcode option:selected").val(),text:$("#textarea").val()};**

**// $("#incode").change(function(){**

**// var inincode=$("#incode").find("option:selected").text();**

**// console.log(inincode);**

**// });**

**$("#clickbutton").click(function(){**

**var incode=$("#incode").find("option:selected").text();**

**// console.log(incode);**

**var outcode=$("#outcode").find("option:selected").text();**

**// console.log(outcode);**

**var text=$("#textarea").val();**

**// console.log(text);**

**var data={incode:incode,outcode:outcode,text:text};**

**console.log(data);**

**$.ajax({**

**type:"post",**

**url:"ajaxGetTranslate.do",**

**data:data,**

**dataType:"json",**

**success:function(response){**

**var translatecontext=response.trans\_result**

**console.log(response);**

**console.log(translatecontext);**

**console.log(translatecontext.dst);**

**console.log(translatecontext[0]);**

**console.log(translatecontext[0].dst);**

**$("#textareaprint").val(translatecontext[0].dst);**

**}**

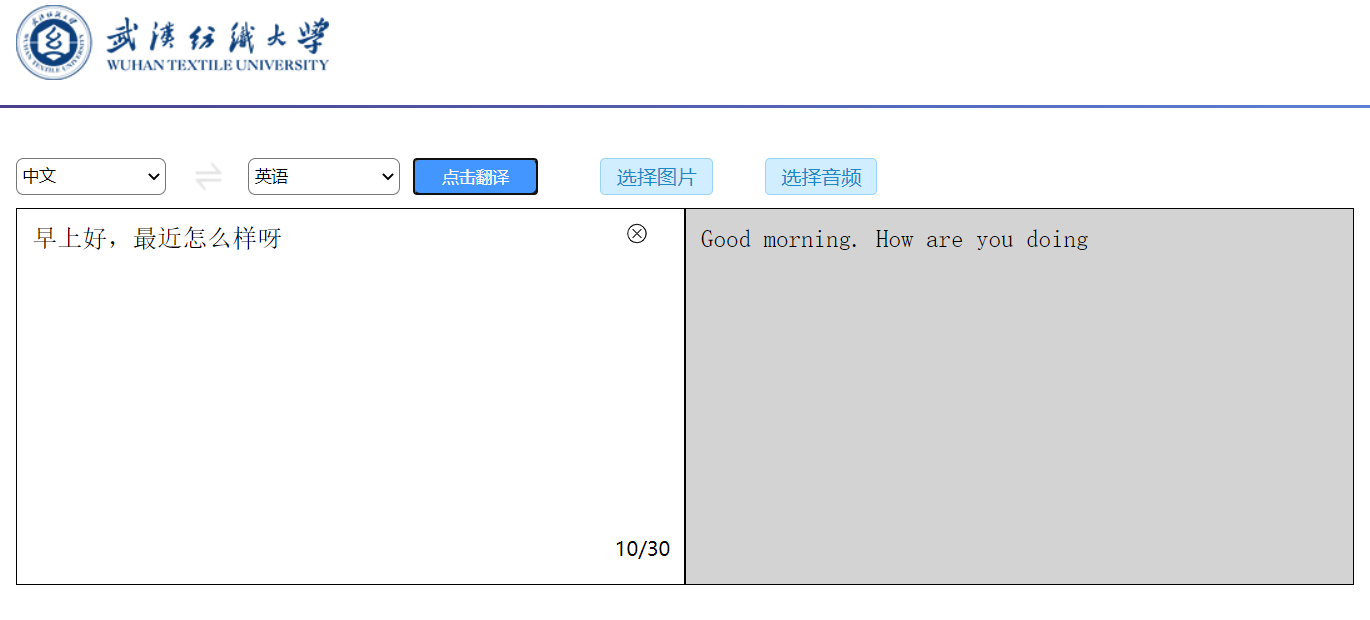
**})**

**});**

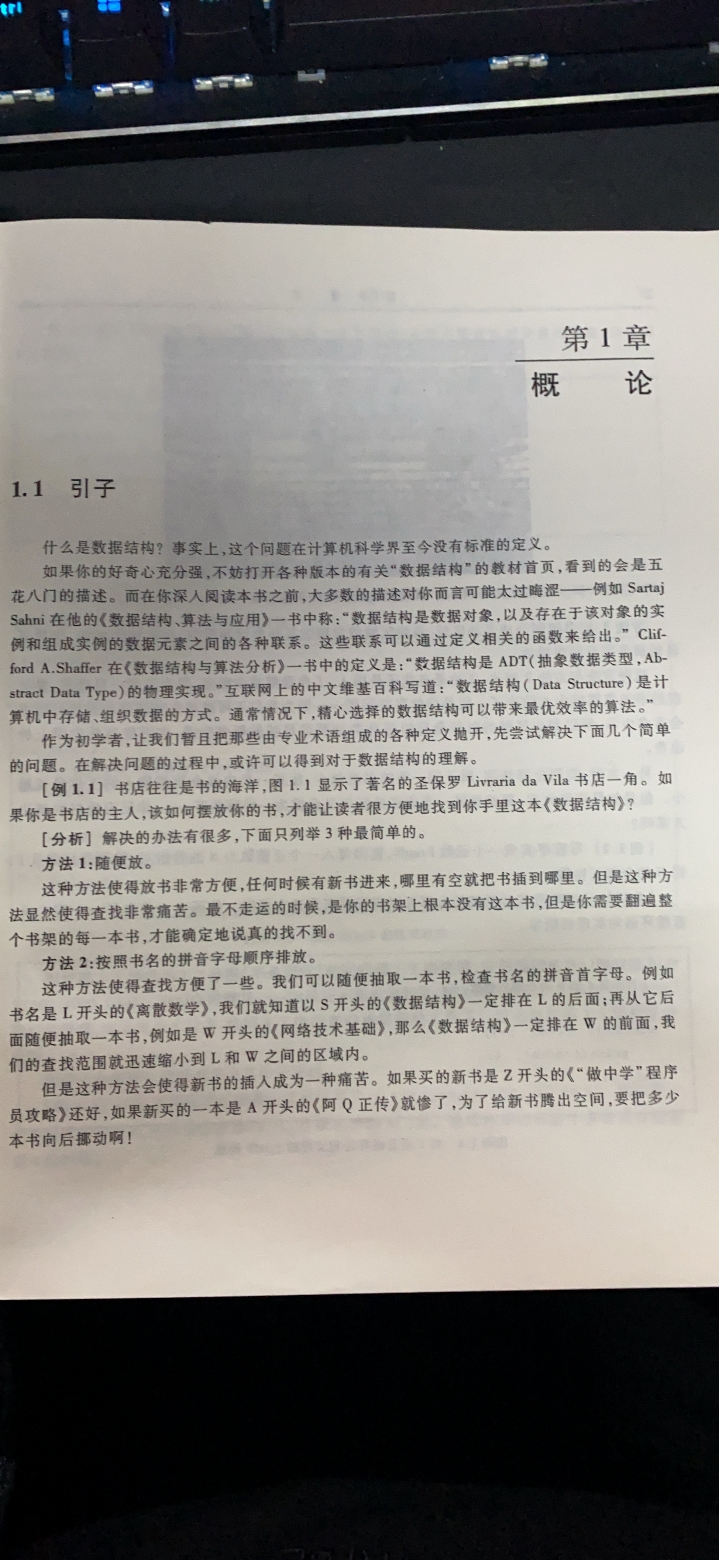
**})**

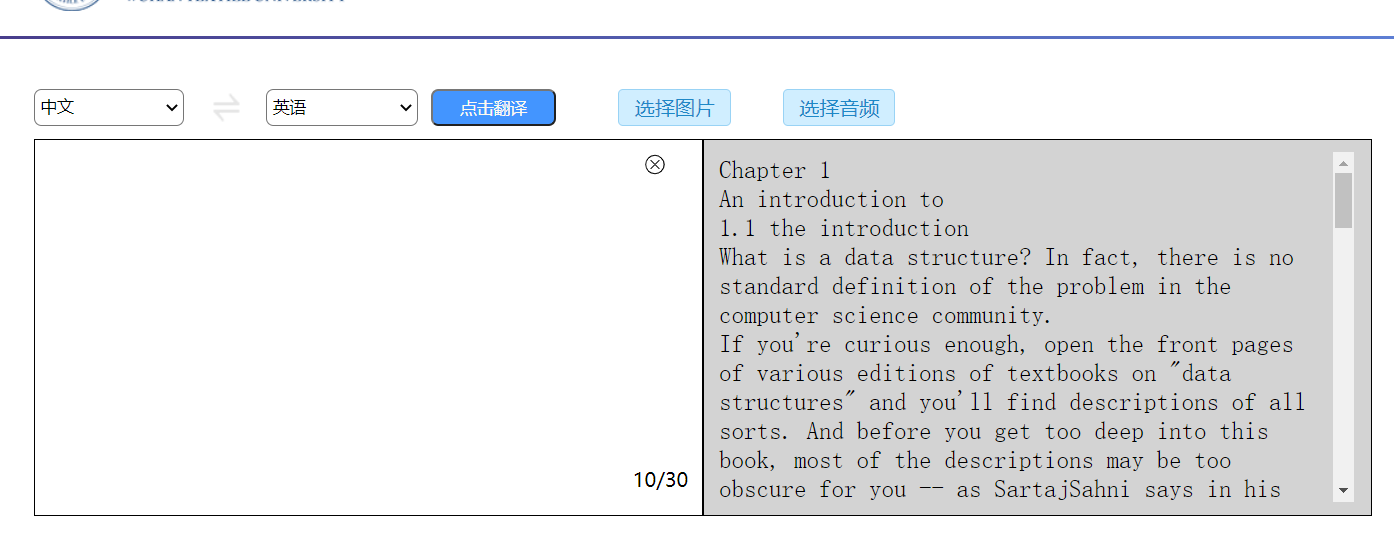
# 4 系统测试

## 4.1普通文本翻译

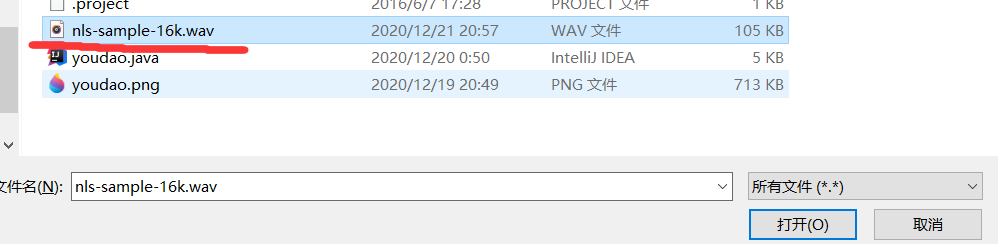


## 4.2图片翻译





## 4.3语音翻译





# 5 系统总结

本学期主要学习了javaWeb的后端部分，做这次课设的时候采用了ssm框架，大部分时间都花费在了后端的ssm框架的搭建上，一开始搭建起来的怎么都不能运行，最后经过网上查找和询问同学，一步步调试发现，我在建立这个项目的时候，系统就已经为我配好了tomcat的虚拟目录，不需要自己在去配置这个虚拟目录了，否则会显示spring容器创建失败。在导入官方的demo时，自己为了方便直接去apache下载了相应的包，然后直接把下载的包导入，导致包的之间的依赖发生冲突，最后老老实实的去使用maven去管理。

在听了其他人的答辩，自己可以使用cookies使自己的翻译变的更快。搭建前端时没有使用任何框架，自己的页面不是很美观。自己会在寒假期间学习一下vue和bootstrap。

我认为自己这次课设应该能获得90分，因为前端没有使用任何框架，后端使用了老师讲的框架，老师的要求全部实现，并且在此基础上，实现了语音翻译和图片翻译。