**国际化技术解决方案**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 制作人 | 备注 |
| 1.0.0 | 2020/2/6 | 朱耀明 |  |
| 1.0.1 | 2020/2/11 | 朱耀明 | 根据讨论完善了方案 |
| 1.0.2 | 2020/2/19 | 朱耀明 | 补充业务集成 |

目录

[1. 前言 3](#_Toc31894932)

[2. 国际化目标 3](#_Toc31894933)

[3. 国际化实现方案 3](#_Toc31894934)

[3.1. 前端方案 3](#_Toc31894935)

[3.2. 后端方案 3](#_Toc31894936)

[3.2.1. 方案一(本地资源文件) 3](#_Toc31894937)

[3.2.2. 方案二(数据库读取MessageResource)(建议方案) 4](#_Toc31894938)

[4. 业务集成 4](#_Toc31894939)

[使用说明 4](#_Toc31894940)

# 前言

国际化是指应用程序运行时，可根据客户端OS的国家/地区、语言的不同而显示不同的界面，用户也可以手动选择不同语言切换.国际化的英文单词是Internationalization，单词较长，通常简称I18N.

# 国际化实现方案

## 前端方案

现在前端框架几乎都支持国际化,使用框架国际化处理即可,与后台交互需要将语种在http协议header中植入.方便后台同步实现国际化值返回前端.

### React

React Intl 用于国际化 React 组件.前端自定义多语言,并提供用户语言切换, 参考<https://www.jianshu.com/p/574f6cea4f26>

### Vue

Vue国际化可以参考<https://www.jianshu.com/p/8e1b4632c0a8>

## 后端方案

### 方案一(本地资源文件)

定义相应的i18n资源文件 ，我们将资源文件放在一个资源包（就是平常的一个包）中 ，一个资源包中每个资源文件必须拥有共同的基名 。除了基名 ，每个文件的名称中还必须有标识其本地信息的附加部分 ，例如 ：一个资源包的基名是 “ message ” ，则与中文（中国）、英文（美国）环境相对应的资源文件名分别为 ：

message\_zh\_CN.properties (中国,中文)

message\_en\_US.propertie (美国,英文)

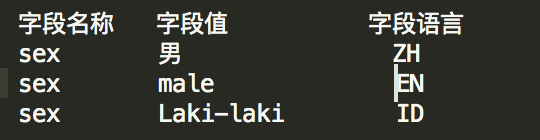
message\_id\_ID.properties (印尼,印尼文)

并且资源文件中采用的是 properties 格式文件 ，所以文件中的所有字符都必须是 ascll 码,不能保存为中文的.

核心思路根据ResourceBundle加载指定Locale对应的资源文件,再取得该资源文件中指定key对应的消息

### 方案二(数据库读取MessageResource)(建议方案)

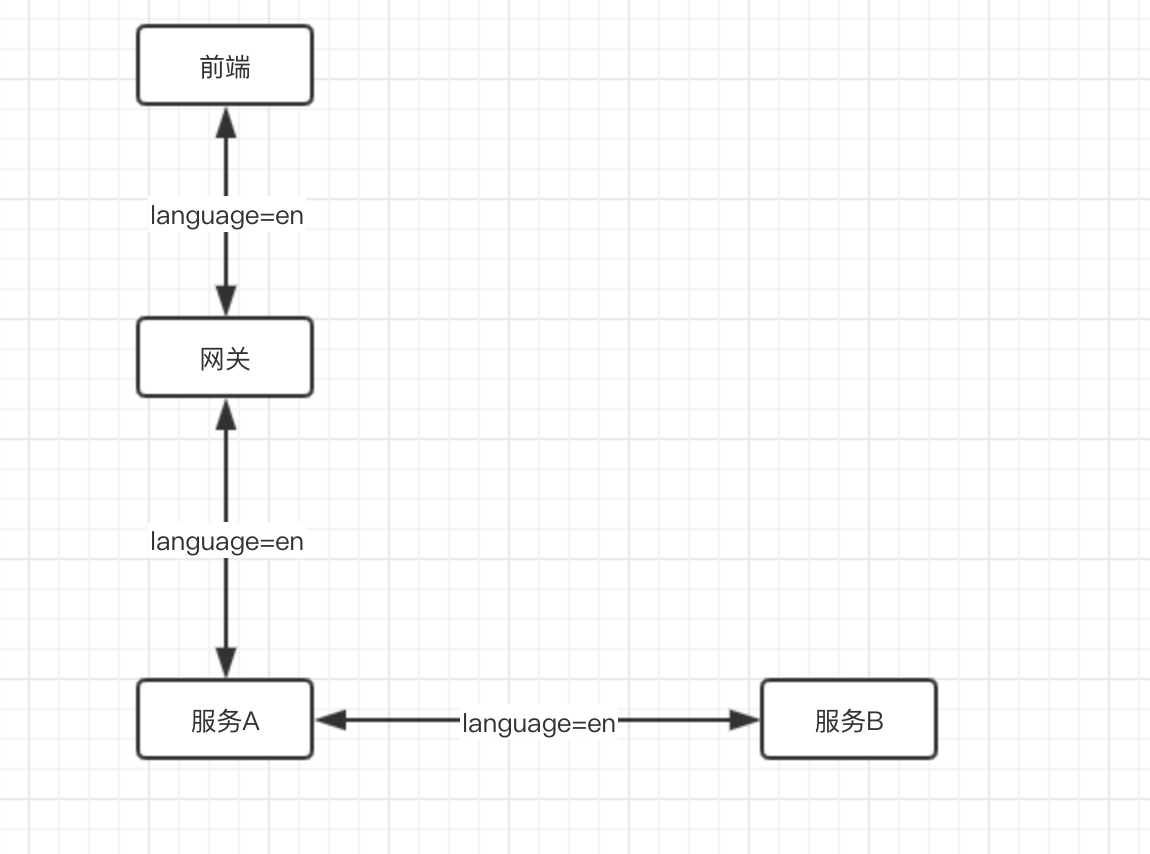
从数据库中读取MessageResource,将动态内容通过数据字段和语言进行字典匹配,根据前端传入的language进行拦截并做转化为适应的国际化资源.



这种方案①通过**自定义注解,对需要国际化的字段进行注解,返回国际化后值**. ② 语言需要传递到各微服务,链路很长的服务,如果需要底层做国际化,需要传递到底层微服务进行国际化处理.

从数据库读取messageSource可以将properties文件的管理界面化，数据化。

后端字段值国际化主要是针对**范围内**的字段进行编码转化.用户手写值无法通过国际化进行转化,通过翻译软件接口存在错误风险.而且一些翻译软件是收费的.



## 场景汇总

### 代码转值场景

code->value(默认语种)-> Value(语种1)->Value(语种2)-> Value(语种3)->Value(语种4)

### 输入框描述展示

value(默认语种)-> Value(语种1)->Value(语种2)-> Value(语种3)->Value(语种4)

### 固定值展示(温馨提示,错误提示,警告提示)

value(默认语种)-> Value(语种1)->Value(语种2)-> Value(语种3)->Value(语种4)

### 输入框(用户手写)

这种情况系统后台不可知语种类型,国际化比较困难

## 语种链路传递

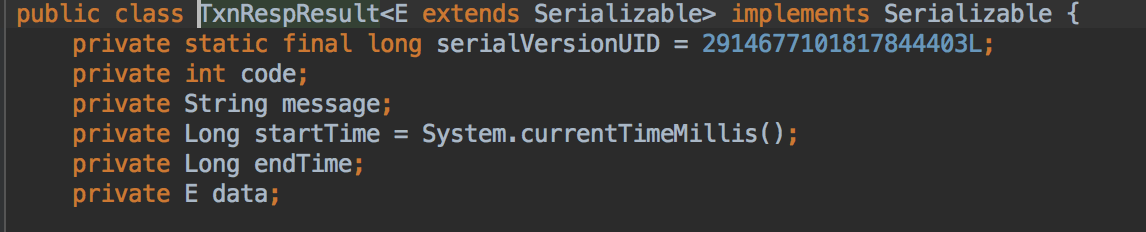
国际化应该模块自身维护, 这样可以减少各模块之间依赖,各模块国际化仅仅在出口controller层进行处理.所以比较解决语种能传递到低层服务.

服务之间通讯统一使用Feign进行通讯,语种要求前端在header中进行保存,通过ThreadLoacal线程保存语种,并针对Feign的Http通讯协议改造,植入language=zh.供下游服务使用.

## 返回格式标准化

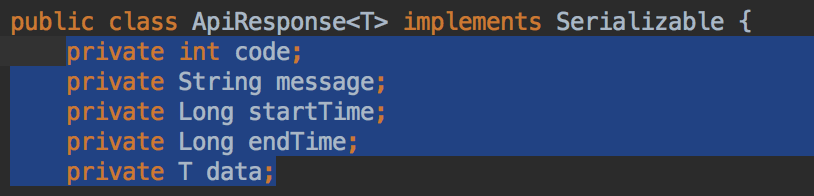
TXN:

TxnRespResult<T>



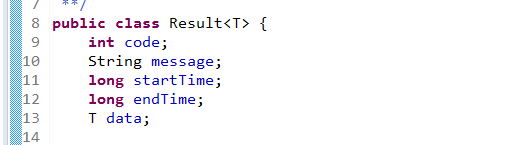
API:

ApiResponse<T>



Task:

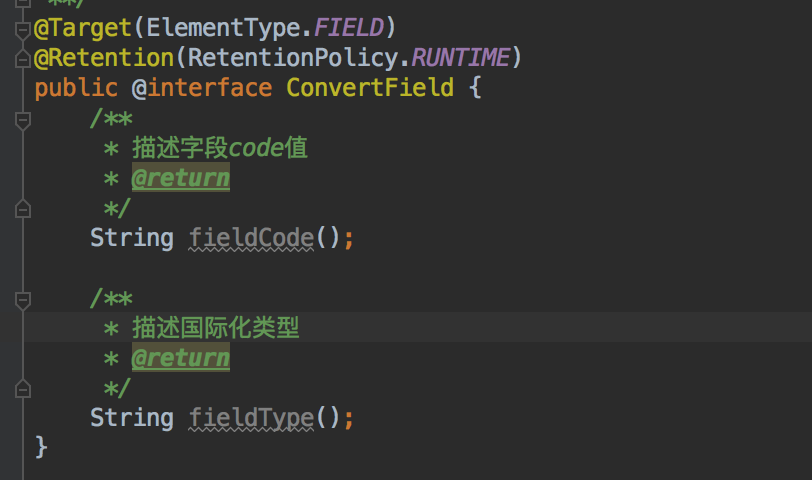
Result<T>



anyForm:

返回Object

需要针对data做深度循环判断返回值是否存在注解@ ConvertField,并结合语种获取对应的值返回.



## 表设计

### 转码值国际化表结构设计参考(租户级)

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **中文描述** | **类型** | **备注** |
| **id** |  | bigint |  |
| **tenant\_id** |  | char(4) |  |
| **table\_name** | 表名称 | varchar(64) |  |
| **type** | 类型 | varchar(64) |  |
| **code** | 类型Code值 | Varchar(64) |  |
| **language** | 语种 | Varchar(32) |  |
| **value** | 语种对应值 | Varchar(128) |  |
| **desc** | 语种对应描述 | Varchar(128) |  |
| **create\_time** | 创建时间 | timestamp |  |
| **update\_time** | 更新时间 | timestamp |  |

### 固定值国际化表结构设计(全局共用)

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **中文描述** | **类型** | **备注** |
| **id** | 主键 | bigint |  |
| **code** | 编码 | varchar(4) | 非必要 |
| **msg** | 消息 | varchar(64) |  |
| **language** | 语种 | varchar(32) |  |
| **value** | 语种对应转码值 | varchar(128) |  |
| **create\_time** | 创建时间 | timestamp |  |
| **update\_time** | 更新时间 | timestamp |  |

# 业务集成

第一: 集成核心思路解决语种链路传递.

第二: 注解根据返回类型获取多语言.

第三: 集成端涉及到国际化的字段表需要调整.

## 使用说明

-- ----------------------------

-- Table structure for `i18n\_core\_fix`

-- ----------------------------

DROP TABLE IF EXISTS `i18n\_core\_fix`;

CREATE TABLE `i18n\_core\_fix` (

`id` bigint(20) NOT NULL,

`tenant\_id` char(4) NOT NULL,

`t\_name` varchar(32) DEFAULT NULL,

`code\_id` varchar(32) DEFAULT NULL,

`msg` varchar(128) DEFAULT NULL,

`language\_type` varchar(32) DEFAULT NULL,

`i18n\_value` varchar(128) DEFAULT NULL,

`create\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

`update\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更新时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='国际化固定值转码';

-- ----------------------------

-- Records of `i18n\_core\_fix`

-- ----------------------------

BEGIN;

INSERT INTO `i18n\_core\_fix` VALUES ('1', '1', 'pr\_system\_parameter', null, '帆帆帆帆', 'zh', '帆帆帆帆', '2020-02-18 14:21:22', '2020-02-18 14:22:50'), ('2', '1', 'pr\_system\_parameter', null, '帆帆帆帆', 'en', 'abcd', '2020-02-18 14:23:01', '2020-02-18 14:23:23'), ('3', '1', 'pr\_system\_parameter', null, '帆帆帆帆', 'id', 'id\_id', '2020-02-18 14:23:29', '2020-02-18 14:23:49'), ('4', '1', 'pr\_system\_parameter', null, '123', 'en', '456', '2020-02-18 16:41:33', '2020-02-18 16:42:28'), ('5', '1', 'pr\_system\_parameter', null, '银行', 'en', 'bank', '2020-02-18 16:41:48', '2020-02-18 16:42:31'), ('6', '1', 'pr\_system\_parameter', null, '0.04', 'en', '0.04', '2020-02-18 16:42:01', '2020-02-18 16:42:36');

COMMIT;

-- ----------------------------

-- Table structure for `i18n\_core`

-- ----------------------------

DROP TABLE IF EXISTS `i18n\_core`;

CREATE TABLE `i18n\_core` (

`id` bigint(20) NOT NULL,

`tenant\_id` char(4) NOT NULL,

`t\_name` varchar(32) DEFAULT NULL COMMENT '表名称',

`type\_id` varchar(32) DEFAULT NULL,

`code\_id` varchar(32) DEFAULT NULL,

`language\_type` varchar(32) DEFAULT NULL COMMENT '语种类型',

`i18n\_value` varchar(128) DEFAULT NULL,

`i18n\_desc` varchar(128) DEFAULT NULL,

`create\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

`update\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '最后更新时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='国际化转码核心表\_1';

-- ----------------------------

-- Records of `i18n\_core`

-- ----------------------------

BEGIN;

INSERT INTO `i18n\_core` VALUES ('1', '1', 'pr\_code\_table', 'ENABLE\_STATUS', '1', 'zh', '正常', '状态正常', '2020-02-18 14:38:16', '2020-02-18 14:39:12'), ('2', '1', 'pr\_code\_table', 'ENABLE\_STATUS', '1', 'en', 'normal', 'normal', '2020-02-18 14:39:20', '2020-02-18 14:39:44'), ('3', '1', 'pr\_code\_table', 'ISVALID', '0', 'en', 'invalidation', null, '2020-02-19 14:07:22', '2020-02-19 14:08:47'), ('4', '1', 'pr\_code\_table', 'ISVALID', '1', 'en', 'effective', null, '2020-02-19 14:07:29', '2020-02-19 14:09:01');

COMMIT;

### TXN集成

### Jrx-Auth集成

* 集成i18n-starter

<dependency>

<groupId>com.jrx.anytxn</groupId>

<artifactId>anytxn-i18n-spring-boot-starter</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

* 添加Config 目的为了将前端或者上游访问的租户和语种传递下去.

@Configuration("\*\*\*Config")

public class WebConfiguration implements WebMvcConfigurer {

@Override

public void addInterceptors(InterceptorRegistry registry) {

// addPathPatterns 用于添加拦截规则 /service/\*\* 使用下面2个拦截器

// excludePathPatterns 用户排除拦截

//registry.addInterceptor(getServiceAuthRestInterceptor()).addPathPatterns("/service/\*\*");

registry.addInterceptor(getTenantAndLanguageInterceptor()).addPathPatterns("/param/systems/\*\*");// 对应的controller路径

}

@Bean

TenantAndLanguageInterceptor getTenantAndLanguageInterceptor() {

return new TenantAndLanguageInterceptor();

}

}

* 集成接口I18nService

public class \*\*\*\* implements I18nService {

// 类型1

@Override

public String getI18nKeyValueByTenantId(String tenantId, String code, String value, String tableName, String language) {

// 先缓存在数据库

return "";

}

@Override

public String getI18nTextValueByTenantId(String tenantId, String code, String value, String language) {

// 先缓存在数据库

return "";

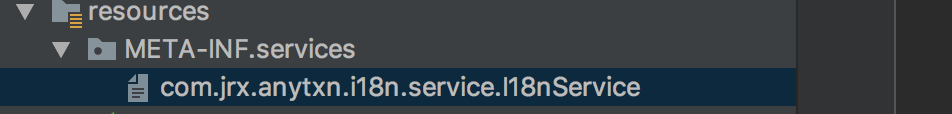
}

}

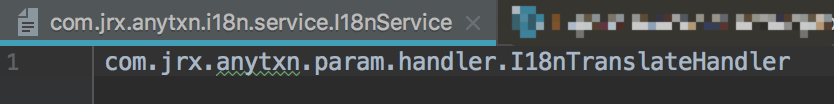
* 需要国际化的字段添加注解



* SPI反射配置添加



文件内容实现I18nService的类全路径 com.\*\*\*.\*\*\*



* jrx-auth-filter改造

filter 需要在请求中添加

// 租户ID

BaseContextHandler.set("tenantId",tenantId);

// 语种

BaseContextHandler.set("languageType",languageType);

* Jrx-auth与前端交互header头添加语种

/\*\*

\* 语种Header

\*/

String LANGUAGE\_HEADER = "x\_language\_header";

# 会议总结

1,需求不够详尽

2,传递服务链具体解决方案

3,后台PC端可以完善下.