eコミマップ

国際化対応開発者用ドキュメント

Ver. 1.1
2013/4/8

独立行政法人 防災科学技術研究所

変更履歴

Version	変更日付	変更内容
1.0	2012/12/28	新規作成
1.1	2013/04/08	インストール用とアップグレード用の言語リソースファイルを統一した。 また、アップグレード画面に言語リソースファイルのアップグレードボタンを追加した。

目次

1　概要 2
2　本ドキュメントで使用される言葉の定義 2
3　国際化対応版 e コミマップの国際化の仕組み 3
4　国際化に対応した e コミマップ各画面の言語設定方法 4
4.1　各画面が使用する言語設定 4
4.2　言語の決定ロジック 4
5　国際化に対応した Java ファイルの作成 4
6　国際化に対応した JSP ファイルの作成 6
7　国際化に対応した JS ファイルの作成 7
7.1　Javascript における言語メッセージの取得方法 7
7.2　Javascript における制限事項 7
8　国際化に対応した XSL ファイルの作成 8
8.1　XSL ファイル内言語メッセージの記述について 8
8.2　XSL ファイルの読み込むを伴う JSP ファイルの国際化対応 9
9　言語リソースファイルへの文字列の追加 9
9.1　言語リソースファイルの内容 9
9.2　言語により語順が異なる場合の対応 11
9.3　メッセージ文字列に関する共通ルール 11
9.4　英文メッセージキーが重複する場合の扱い 11
10　メッセージテーブルの DB 登録方法 13
10.1　インストール時の登録 13
10.2　アップグレード時の登録 13
11　インストーラーの国際化対応 15
付録Ａ　Javascript 用言語リソースのロードサイズの低減について 16
付録Ｂ　XSL ファイルの国際化対応によって追加するメソッド 17

1　概要

国際化対応版 e コミマップを利用すると、e コミマップインストーラー、e コミマップ管理画面、e コミマップ地図操作画面のメッセージを任意の言語で表示することができる。(2012 年 12 月現在、英語と日本語のメッセージを表示させることができる。)

本ドキュメントでは、国際化対応版 e コミマップの開発を行う場合に、開発者が注意すべき事項について記述する。

2　本ドキュメントで使用される言葉の定義

日本語の言葉	英訳	詳細
本ドキュメントで使われる言葉と定義	Terms and definitions	―
e コミマップ国際化対応開発者用ドキュメント	e-Community Map Internationalization Manual for Developers	開発者用ドキュメントのタイトル
システム言語	system language	e コミマップシステムにおいて共通の設定言語（管理画面においてデフォルト項目、ユーザー管理、サーバー設定、サイト管理、アップグレード画面に適用される）
サイト言語	site language	サイト毎に設定される言語（管理画面において、項目、マップ、グループ、ユーザー、バックアップ、設定、ウィジェット各画面に適用される）
言語リソースファイル	message resource file	開発者がインストール時、アップグレード時に提供する言語コード、メッセージ ID、メッセージで構成されたテキストファイル
メッセージ ID (単に ID)	message ID	メッセージを表示する部分のソースコードに埋め込まれた英文文字列。言語メッセージ格納テーブルのメッセージ ID 列の文字列と一致した場合に該当レコードのメッセージを e コミマップの画面に表示する
メッセージ	message	設定された言語で画面に表示される文字列
開発者	developer	e コミマップシステムの開発を行うもの
ユーザー	user	管理者権限を持ちインポート／エクスポート機能を利用してメッセージを修正できるもの

3　国際化対応版 e コミマップの国際化の仕組み

① 設定言語としてシステム言語とサイト言語がある。システム言語は、 _option テーブル、サイト言語は、 _community テーブルに格納される。
例：システム設定言語を「日本語」とした場合、その言語コード”jp”が取得される。

② 設定言語とメッセージキーの組み合わせで、_message テーブルを検索し、設定言語のメッセージデータを取得する。
例：出力するメッセージキーが”Update”となっていた場合、設定言語”jp”とメッセージキー”Update”から、設定言語のメッセージデータ「更新」が取得される。

③ 設定言語のメッセージデータを画面に出力する。
例：設定言語のメッセージデータである「更新」が画面に表示される。

4　国際化に対応したｅコミマップ各画面の言語設定方法

4.1　各画面が使用する言語設定

国際化に対応したｅコミマップでは、サイト言語とシステム言語の２種類の言語設定が可能である。（詳細は、画面設計書を参考のこと）

マップ画面、サイト管理画面、システム管理画面、インストール画面では、それぞれ、下表に示される言語が設定される。

画面／権限	一般ユーザ	サイト管理者	システム管理者
マップ画面	サイト言語	サイト言語	サイト言語
サイト管理画面	×	サイト言語	サイト言語
システム管理画面	×	×	システム言語
インストール画面	×	×	システム言語

4.2　言語の決定ロジック

画面で使用する言語は、URL のパラメータにより以下のルールにより決定される。（上のルールが優先される）

1. 以下の２画面は、システム言語を利用する。
   - ・サーバ設定画面
   - ・アップグレード画面
2. URL パラメータに lcid[1]がある場合はそのサイト言語を使用する
3. URL パラメータに admin がある場合、または URL パラメータが cid=0 となっている場合はシステム言語を利用する
4. URL パラメータに cid がある場合はそのサイト言語を使用する
5. session に "communityid" が設定されている場合はそのサイト言語を使用する
6. 1~5 のどれにも該当しない場合、システム言語を使用する

5　国際化に対応した Java ファイルの作成

（１）　java ファイルに共通クラス DBLang と LangUtils がインポートされていない場合、ファイルに追加する。（リスト５－１　①参照）

（２）　DBLang のインスタンスを取得する。（リスト５－１　②参照）

（３）　メソッド lang._() の引数に英文キーを指定する。これにより引数の英文キーに対応する

[1] lcid：言語決定のための community ID。lcid=0 はシステム言語とする。
webapps/map/admin/admin.js の adminGet, adminPost 関数により自動的にこのパラメータが追加される。現在、これらの関数以外で lcid を使用している個所はない。

言語メッセージが取得される。

リスト５－１

```
/* ¥src¥jp¥ecom_plat¥map¥pdf¥PdfMapWriter.java
 */
  （中略）
  import de.micromata.opengis.kml.v_2_2_0.Feature;

  import jp.ecom_plat.map.util.DBLang;
  import jp.ecom_plat.map.util.LangUtils;

  /**
   * 地図を PDF 出力するためのクラス.<br/>
   * PDFServlet から呼ばれる
   */
  public class PdfMapWriter
  {
   （中略）
    private void printMapPdf(Document doc, PdfWriter pdfWriter, Rectangle pageRect, int[]
  progressCounts, JSONObject prevJSON, int epsgCode) throws Exception
    {
      DBLang lang = LangUtils.getSiteDBLang();

      //用紙にあわせたフォントサイズと余白を計算（A4 横=842 ポイント）

   （中略）
  //インデックス出力 別ページ
          if (printIndex) {
            float indexTitleFontSize = Math.min(pageRect.getWidth(), pageRect.getHeight())/30;
            //表題
            String indexString = lang._("Map Index");
   （以下略）
```

①

②

③

6　国際化に対応した JSP ファイルの作成

（１）　JSP ファイルに共通インクルード lang_resource.jsp ファイルまたは、jsp_lang.jsp ファイルを追加する。（リスト６－１　①参照）

- JSP ファイルが Javascript を含む場合は、lang_resource.jsp ファイルを指定する。
- JSP ファイルが Javascript を含まない場合は、jsp_lang.jsp ファイルを指定する。

（２）　JSP 内で記述する Java ソースコードは、下記のようになる。（リスト６－１　②参照）

例）String test = lang._(“Enter name”);

また、HTML タグ内で記述する場合は、以下のようになる。（リスト６－１　③参照）

例）<%=lang._(“Enter name”)%>

リスト６－１

```
<% /* ¥webapps¥map¥map¥pdf.jsp
 */
%><%@page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"
%><%@page import="jp.ecom_plat.map.pdf.PdfMapWriter"
%><%@page import="jp.ecom_plat.map.db.FeatureResult.AttrResult"
%><%@include file="/include/lang_resource.jsp"          ①
%>
<%@include file="/include/include.jsp"
%><%

  //ログ
  logger.info(session.getId()+":"+userInfo.userId+"¥t"+request.getQueryString());
  //パラメータチェック
    long mapId;
    try {
      mapId = Long.parseLong(request.getParameter("mid"));
                                                          ②
    } catch (Exception e) {
      showErrorPage(pageContext, lang._("Parameter Error"), "mid="+request.getParameter("mid"));
      return;
    }
 (中略)
<div class="pdf_form_block">
  <table cellspacing="2" cellpadding="0">
                                                          ③
  <tr><td valign="top" nowrap="nowrap" width="10%"><%=lang._("Map Title")%> : <br/>
```

```
    <textarea id="maptitle" name="maptitle" style="width:100%;height:35px;"
    onchange="pdf.showPreview();"><%=StringUtils.escapeHTML(mapInfo.title)%></textarea>
  </td></tr>
  <tr><td valign="top" nowrap="nowrap"><%=lang._("Explanation and Comment")%> : <br/>
    <textarea id="description" name="description" style="width:100%;height:60px;"
    onchange="pdf.showPreview();"><%=StringUtils.escapeHTML(mapInfo.description)%></textarea>
  </td></tr>
  </table>
</div>
 （以下略）
```

③

7　国際化に対応した JS ファイルの作成

7.1　Javascript における言語メッセージの取得方法

Javascript 内では、下記のように呼び出す。ここではシングルクォテーションで文字列を囲んでいるが、ダブルクォテーションでもよい。（リスト７－１参照）

例）var test = lang._('Display')

7.2　Javascript における制限事項

Javascript で使用されている国際化対象文字列は、クライアントへ渡す言語メッセージのサイズを極力少なくするため、プログラムにより *.js ファイルから自動抽出される。（付録Ａ参照）

このため以下の制約がある。

（１）　jsp ファイル内の javascript で lang._(...) を使用しない

例：

正：　alert('<%=lang._("Input Name")%>');

誤：　alert(lang._('Input Name'));

（２）　lang._(...) のカッコの中で式や変数を使用しない（パラメータはこの限りでない）

例：

正：　dbUpdate?lang._('register'):lang._('get')

誤：　lang._(dbUpdate?'register':'get')

現在 *.js ファイルの探索は map, admin, wizard, ecom ディレクトリ（サブディレクトリを含む）を対象とする。

これ以外のディレクトリ下の *.js ファイルで lang._(…) を使用する場合は LangUtils.java の以下の指定を変更する必要がある。

static protected final String[] jsDirs = { "map", "admin", "wizard", "ecom" };

リスト７－１

```
/* ¥webapps¥map¥map¥PdfWriter.js
 */
/**
 * @class e コミマップ編集関連クラス. 編集開始時に動的に読み込まれ初期化される.
 */
 (中略)
/** 編集ダイアログを表示 (Ajax で属性情報読み込み後に生成) */
showEditDialog : function(buttons)
{
  if (!this.editLoadingDialog) {
    dojo.require("map.dijit.ModelessDialog");
    var div = document.createElement('div');
    div.innerHTML = lang._('Loading');        ①
    this.editLoadingDialog = new map.dijit.ModelessDialog ({id: 'editLoadingDialog' , title :
lang._('Register Edit Window'), duration:10, right:2, top:2 }, div);
  }
 (以下略)
```

8　国際化に対応した XSL ファイルの作成

8.1　XSL ファイル内言語メッセージの記述について

XSL ファイル内のメッセージ（リスト 8-1 参照）を国際化対応のため、lang.() 関数を利用してリスト 8-2 のように変更する。

リスト８－１

```
<xsl:when test="count(csw:SearchResults/csw:Record)=0">
何も検索されませんでした。
</xsl:when>
```

リスト８－２

```
<xsl:when test="count(csw:SearchResults/csw:Record)=0">
lang._("Nothing returned")
</xsl:when>
```

8.2　XSL ファイルの読み込むを伴う JSP ファイルの国際化対応

従来の XSL 出力処理の前に XSL ファイルの lang._(“~”)の部分を設定された言語のメッセージに置き換える。この処理が、LangUtils.java の readXSLTFile メソッドの処理となる。（従来の処理をリスト 8-3 に示す。今回の国際化対応による処理をリスト 8-4 に示す。）readXSLTFile メソッドの詳細については付録Ｂを参照のこと。

リスト８－３

```
String xsl = "csw/getRecordsResponseMap.xsl";
StreamSource  ss = new StreamSource(getServletContext().getRealPath(xsl));
Transformer transformer = factory.newTransformer(ss);
```

リスト８－４

```
String xsl = "csw/getRecordsResponseMap.xsl";
StreamSource  ss = new StreamSource(LangUtils.readXSLTFile(getServletContext().getRealPath(xsl)));
Transformer transformer = factory.newTransformer(ss);
```

9　言語リソースファイルへの文字列の追加

9.1　言語リソースファイルの内容

言語リソースファイル lang_resource.txt は、タブ区切りのテキストファイル（UTF-8 フォーマット）で左から言語コード、ID、メッセージ、コメントで構成される。

行末のコメントとして、”//”以降にコメントが記載できる。また、行間コメントとして、表９．１に示すように言語コードの後に、コメントが記載できるようになっている。

行が”//”で始まる場合、その以降行末まで、言語リソースファイルをインストール時、あるいはアップグレード時に読み込む場合、無視されます。

表９．１　言語リソースファイル

```
言語コード　　ID　　Message　　コメント
//
// 日本語
//
（中略）
ja　　Current DB version　　現在の DB バージョン
ja　　Current application key:　　現在のアプリケーションキー：
ja　　Current name　　現在の名前// ここはコメントです
ja　　DB Error　　DB エラー
ja //ここもコメントです
ja　　DB Info　　データベース情報
ja　　DB Name　　データベース名
（以下　略）
```

行頭に"//"がある場合その行は無視されます

行間にコメントが記入できる
言語コードの後の"//"以降は、コメントとして、DB に登録されます

表９．２　言語メッセージ格納テーブル

①

No.	言語コード	ID	Message	コメント
（中略）				
128	ja	Current DB version	現在の DB バージョン	
129	ja	Current application key:	現在のアプリケーションキー：	
130	ja	Current name	現在の名前	ここはコメントです
131	ja	DB Error	DB エラー	
132	ja			ここもコメントです
133	ja	DB Info	データベース情報	
134	ja	DB Name	データベース名	
（以下　略）				

②

9.2　言語により語順が異なる場合の対応

和文の「地図を編集」を英語に翻訳する場合、”Edit a map”となるが、この場合、目的語の地図, “a map”をパラメータ({0}, {1}, ...)に設定すると、目的語の部分に対応する翻訳を行えば、動詞の部分「を編集」,”Edit”は、そのまま利用できる。

ただ、この場合、英語と日本語では、目的語の位置が動詞の後か、動詞の前かで異なる。その場合は、言語リソースファイルに、

ja　　Edit {0}　　{0}を編集

として（タブ区切り)、プログラムソース内に、

<%=lang._(“Edit {0}”, mapTypeName)%>

と記述すればよい。(JSP で HTML タグ内に記述する場合)

9.3　メッセージ文字列に関する共通ルール

- 文字列はタブを含んではならない
- 文字列中に改行を含む場合は ¥n を使用する
- 文字列中に ¥（バックスラッシュ）を含む場合は ¥¥ を使用する

9.4　英文メッセージキーが重複する場合の扱い

英文が同一で日本語訳が異なる場合は、これらを区別するため英文の末尾に接尾語(<!--(¥d+)-->) をつける。(¥d+)には、数字を入力すること。

例：

Map List　　マップ一覧表示
Map List<!--2-->　　マップ一覧
Map List<!--3-->　　マップ一覧ページ
Map List<!--4-->　　地図一覧

この接尾語は、画面に表示されることはない。

9.5　言語を追加する場合の対応

インストール時に言語コードと言語名を追加する場合、webapps/map/WEB-INF/classes/ResourceInfo.properties ファイル（リスト９－１）の LANG_CODES, LANG_NAMES に追加する。

（例）言語コード vn, 言語名　Ti¥u1ebfng Vi¥u1ec7t としてベトナム語を追加する。

LANG_CODES=en,ja,vn
LANG_NAMES=English,¥u65e5¥u672c¥u8a9e,Ti¥u1ebfng Vi¥u1ec7t

リスト９－１

```
NAMESPACE=map

MAPDB=jdbc/bosai
GEOSERVER_DATA_STORE=map
GEOSERVER_DEFAULT_SLD=default

FONT=/WEB-INF/fonts/ipagp.ttf
FONT_G=/WEB-INF/fonts/ipagp.ttf
FONT_M=/WEB-INF/fonts/ipamp.ttf
#FONT=/WEB-INF/fonts/msgothic.ttc,1

SMTP_HOST_NAME=localhost
SMTP_FROM=info@ecom-plat.jp

NOTICE_DAYS=10

#PREVIEW_NOAUTH=1

SYSTEM_CLOCK_USES_UTC=0

### Default supported languages ###
        #Language list used only in install forms
LANG_CODES_INSTALL=en,ja
LANG_NAMES_INSTALL=English,¥u65e5¥u672c¥u8a9e
        #Language list inserted to DB for system use
LANG_CODES=en,ja
LANG_NAMES=English,¥u65e5¥u672c¥u8a9e
```

言語コードと言語名

10 メッセージテーブルの DB 登録方法

10.1 インストール時の登録

webapps/map/install フォルダの lang_resource.txt が言語リソースファイルである。（図 10.1）インストール時は、自動で、この言語リソースファイルが読み取られ、言語メッセージ格納テーブルに格納される。

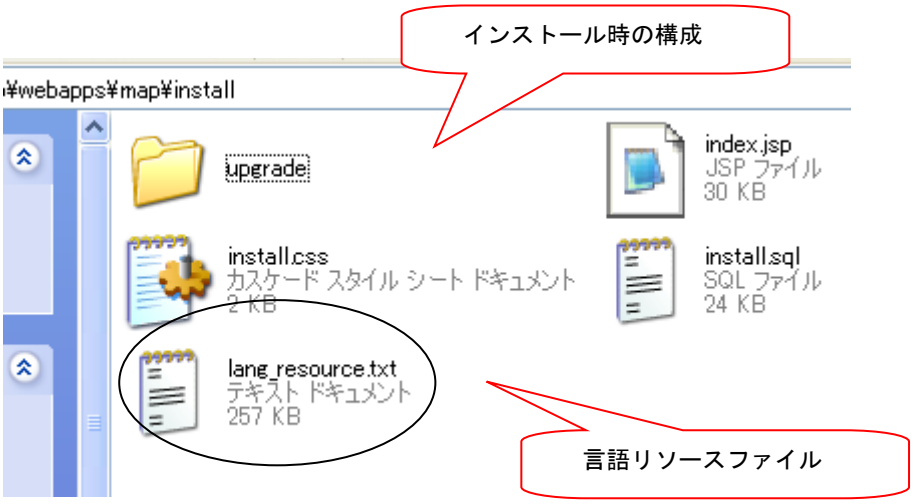

図 10.1

10.2 アップグレード時の登録

アップグレードにおいて言語メッセージの追加、変更がある場合、開発者は、インストール時と同じフォルダに lang_resource.txt を配置する。追加、変更が無い場合は、提供する必要はない。

アップグレード管理画面（図 10.3）の「アップグレード実行」ボタンを押下すると、lang_resource.txt を読み込んで言語メッセージ格納テーブル(_message)を変更する。lang_resource.txt が無い場合は、データベーステーブルの更新は行わない。

図 10.2 に示すアップグレード画面は、更新する言語リソースファイルが無い場合、図 10.3 に示すアップグレード画面は、更新する言語リソースファイルがある場合である。

また、アップグレード時のデータベーステーブルの変更においては、ユーザーが言語リソースインポート機能を利用してメッセージの変更を行ったレコードは、アップグレード対象外となる。


System Administrator (システム管理者) でログイン中
ログアウト
e-Community Map 管理画面
サイト表示
共通項目 登録情報 地図 グループ ユーザ バックアップ 設定
ウィジェット
デフォルト項目 ユーザ管理 サイト管理 サーバ設定 アップグレード
アップグレード
アップグレードは必要ありません。
現在のDBバージョン:2.1.0
更新の必要はありません
言語リソースアップグレード
前回の更新 : 2013/04/08 11:20:11
言語リソースの更新はありません。

図 10.2


System Administrator (システム管理者) でログイン中
ログアウト
e-Community Map 管理画面
サイト表示
共通項目 登録情報 地図 グループ ユーザ バックアップ 設定 ウィジェット
デフォルト項目 ユーザ管理 サイト管理 サーバ設定 アップグレード
アップグレード
アップグレードは必要ありません。
アップグレード実行で言語
メッセージを更新します
現在のDBバージョン:2.1.0
言語リソースアップグレード
前回の更新 : 2013/04/08 11:20:11
言語リソースファイル日時 : 2013/04/08 12:17:50
アップグレード実行

図 10.3

11 インストーラーの国際化対応

インストール画面に文字列を追加する場合は以下のファイルに文字列の定義を追加する。

（１） src/ecommap_lang.properties 英文文字列

（２） src/ecommap_lang_en.properties 英文文字列

（３） src/ecommap_lang_ja.properties 和文文字列

注意：

・1 と 2 の内容は同一で以下の形式をとる。（リスト８－１　参照）

英文文字列=英文文字列

・3 は以下の形式をとる。（リスト１１－１　参照）

英文文字列=和文文字列

・文字列中の空白の前に ¥（バックスラッシュ）を追加する

・和文文字列は Unicode エスケープを使用する（Unicode エスケープの変換は JDK 付属の native2ascii を使用する）

リスト１１－１

英文の例：

```
Add¥ available¥ user¥ to¥ PostgreSQL=Add¥ available¥ user¥ to¥ PostgreSQL
```

和文の例：

```
Add¥ available¥ user¥ to¥ PostgreSQL=¥u30c7¥u30fc¥u30bf¥u30d9¥u30fc¥u30b9¥u5229¥u7528¥u53ef¥u80fd¥u306a¥u30e6¥u30fc¥u30b6¥u3092PostgreSQL¥u306b¥u8ffd¥u52a0
```

付録Ａ　Javascript 用言語リソースのロードサイズの低減について

課題：Javascript 用言語リソースのロードサイズの低減について

解決案：*.js ファイルのみから lang._() で使用されている文字列を拾い出す。
（*.jsp ファイルの中の javascript の中で lang._() が使用されていないことは確認済み）

具体的には、map, admin, wizard, ecom のディレクトリ（サブディレクトリを含む）下の *.js ファイルを走査して lang._() で指定されれている文字列を拾い出します。

一度取得した結果は static 変数に保存され、次回からはそれが使用されます。つまり、実際に探索が行われるのは、１回のサーバ起動に対して１回だけです。

タイミング：js ファイルの探索は、サーバが起動されて最初にブラウザからのリクエストを受け付けた時点で実行されます。

結果：メッセージ数は、2296 件から 319 件に減少。

注意：lang._() の使用に関していくつかの制限が発生します。
（以下の制約は、現状ない）

1. *.jsp ファイルの中の javascript 部分で lang._() を使用できない
2. lang._() のカッコの中に変数や式を書けない（パラメータは除く）

付録Ｂ　XSL ファイルの国際化対応によって追加するメソッド

LangUtils.java において追加するメソッド

```
/**
 * <div lang="ja">
 * *.xsl ファイルを読み込み、表示系を国際化する。
 *
 * @param String xslFile 絶対ディレクトリ含むファイル名
 * @return StringReader 文字列の文字ストリーム
 * </div>
 *
 * <div lang="en">
 * Read *.xsl files for multiple language processing.
 *
 * @param String xslFile File Name with absolute directory
 * @return StringReader A character stream whose source is a string
 * </div>
 */
static public StringReader readXSLTFile(String xslFile) throws Exception {
  logger.info("XSL FILE:"+xslFile);
  BufferedReader br = null;
  String xslString = null;
  try {
       br = new BufferedReader(new InputStreamReader(new FileInputStream(xslFile)));
       StringBuffer sb = new StringBuffer();
       int i;
       while ((i = br.read()) != -1) {
           sb.append((char) i);
       }
       xslString = sb.toString();
       xslString = replaceLanguages(xslString);
       StringReader reader = new StringReader(xslString);
       return reader;
     } finally {
        br.close();
     }
}
```

設定された言語のメッセージに置き換えるメソッドは以下のようになる。

```
private static String replaceLanguages(String inString) {
 DBLang lang = LangUtils.getSiteDBLang();
 BufferedReader br = null;
 HashSet<String> hs = new HashSet<String>();
 try {
  br = new BufferedReader(new StringReader(inString));
  StreamTokenizer st = new StreamTokenizer(br);
  st.slashStarComments(true);
  st.slashSlashComments(true);
  st.ordinaryChar('.');
  st.ordinaryChar('-');
  st.ordinaryChar('/');
  st.wordChars('_', '_');
```

```
    do {
     if (st.nextToken() == StreamTokenizer.TT_WORD
         && st.sval.equals("lang") && st.nextToken() == '.'
         && st.nextToken() == StreamTokenizer.TT_WORD
         && st.sval.equals("_") && st.nextToken() == '('
         && (st.nextToken() == '"' || st.ttype == '¥'')) {
         hs.add(st.sval);
         st.nextToken();
     }
    } while (st.ttype != StreamTokenizer.TT_EOF);
    br.close();
   } catch (java.io.IOException e) {}
   Iterator it = hs.iterator();
   while (it.hasNext()) {
   String lang_key = it.next().toString();
   inString = inString.replace("lang._(¥""+lang_key+"¥")", lang._(lang_key));
   }
   return inString;
  }
```