User Manual 4.10 Errors management and internationalization

De Wiki
Aller à : navigation, rechercher
User Manual 4.10 Errors management and internationalization

Sommaire

Introduction

Scope

A locale is a set of 2 parameters, one to define the user's language, the other one to define the country. A locale is used to identify a country, a language or a dialect.

The language parameter is composed by 2 lower cases whose list is defined by the ISO 639-2 Langage Code List [R1]. For instance, "fr" means French whereas "en" means English.

The country parameter is composed by 2 upper cases whose list is defined by the ISO 3166 French Country Names and Code Elements [R2]. France is designated by "FR", Belgium by "BE" and United Kingdom by "GB".

The association of a language parameter and a country parameter leads to a full definition of all language varieties. As a result, the fr_FR locale points out French language spoken in France whereas the fr_BE locale points out French language spoken in Belgium.

Here is a use case of the Locale object :

```
// French locale
Locale france = new Locale("fr", "FR");

// French locale language
Locale french = new Locale("fr");
```

```
// Belgium locale
Locale belgium = new Locale("fr", "BE");
```

Moreover, some locales are defined as constants of the class. This is the case for French language (Locale.FRENCH), French locale (Locale.FRENCH) or English language (Locale.ENGLISH).

The second Java component i18n is the ResourceBundle. It is in charge of retrieving a given locale. However, the ResourceBundle is an abstract class, therefore a concrete implementation of the ResoureBundle, namely the PropertyResourceBundle, is used. This implementation is based on a basic file name, “properties”, and, for a given locale, it goes for the translation if it is available. To do that, it checks if a properties file, whose name is <basic file name>_<language code>_<country code>, exists.

If necessary it goes for a translation that is more common, that is to say only based on the language and whose properties file would be named <basic file name>_<language code>.

If the need still arises, it goes for the basic file to retrieve a translation.

For example, in the following code :

```
ResourceBundle bundle = ResourceBundle.getBundle("message");
```

If the current locale (for example the default language of the operating system) is fr_FR, here is the order of the files research :

3.1. message_fr_FR.propertie,
3.2. message_fr.properties,
3.3. message.properties.

It is possible to get back a particular ResourceBundle by specifying explicitly the locale to be used :

```
ResourceBundle bundle = ResourceBundle.getBundle("message", Locale.ENGLISH);
```

Afterwards, in order to get an internationalized message, for example a message associated to the key “HelloWorld” in the properties file, the method getString(String key) must be used :

```
String message =  bundle.getString("HelloWorld");
```

Javadoc

Links

Useful Documents

Package Overview

Features Description

Exception and internationalization

As indicated in the SRS document (requirements PBD-LOG_850), we should create an error message only when it is the only option i.e. Java basic errors suit. In this case, it is mandatory to add a particular error message, the procedure is the following one :

- Find a unique identifier (one key, see the following chapter for the definition rule) for the message and a short and relevant text,
- Modify the PatriusMessages class and add it to the enumeration, in US English which is the default language, e;g. FOO_MESSAGE("foo message for testing purpose") or BAR_MESSAGE("another foo message for testing purpose {0} {1}") with 2 expected arguments,
- Add this message to the translation files PatriusMessages_<language>.properties, one per language, in the directory …/src/main/resources/META-INF/localization, with the following form “key=text”. The text is translated in the required language, one has to be carful with the arguments location if there are some of them (they can be inverted).

N.B. : For languages which use accents like French, each accented character has to be written with a specific coding. The list of the characters and their equivalent coding is available on the following web site : http://www.utf8-chartable.de/

If a class is necessary for a specific message type, it has to inherit from the one of the following java classes : Exception or RuntimeException.

Example :

```
public class MyFutileException extends Exception implements
ExceptionContextProvider {
   ...
   private ExceptionContext exceptionContext ; /** compulsory variable */
   ...
   MyFutileException(final Locale locale,final Object ... args) {

exceptionContext.addMessage(locale,PatriusMessages.PDB_FUTILE_EXCEPTION,args)
   }
   ...
}
```

In this case, a unique identifier as well as the original text of the message and its translations are also needed in the translation files.

Getting Started

Contents

Interfaces

Classes

Menu de navigation

Outils personnels

- 3.12.148.180
- Discussion avec cette adresse IP
- Créer un compte
- Se connecter

Espaces de noms

- Page
- Discussion

Variantes

Affichages

- Lire
- Voir le texte source
- Historique
- Exporter en PDF

Plus

Rechercher

Rechercher Lire

PATRIUS

- Welcome

Evolutions

- Main differences between V4.15 and V4.14
- Main differences between V4.14 and V4.13
- Main differences between V4.13 and V4.12
- Main differences between V4.12 and V4.11
- Main differences between V4.11 and V4.10
- Main differences between V4.10 and V4.9
- Main differences between V4.9 and V4.8
- Main differences between V4.8 and V4.7
- Main differences between V4.7 and V4.6.1
- Main differences between V4.6.1 and V4.5.1
- Main differences between V4.5.1 and V4.4
- Main differences between V4.4 and V4.3
- Main differences between V4.3 and V4.2
- Main differences between V4.2 and V4.1.1
- Main differences between V4.1.1 and V4.1
- Main differences between V4.1 and V4.0
- Main differences between V4.0 and V3.4.1

User Manual

- User Manual 4.15
- User Manual 4.14
- User Manual 4.13
- User Manual 4.12
- User Manual 4.11
- User Manual 4.10
- User Manual 4.9
- User Manual 4.8
- User Manual 4.7
- User Manual 4.6.1
- User Manual 4.5.1
- User Manual 4.4
- User Manual 4.3
- User Manual 4.2
- User Manual 4.1
- User Manual 4.0
- User Manual 3.4.1
- User Manual 3.3

Tutorials

- Tutorials 4.15
- Tutorials 4.14
- Tutorials 4.13.5
- Tutorials 4.12.1
- Tutorials 4.8.1
- Tutorials 4.5.1

- Tutorials 4.4
- Tutorials 4.1
- Tutorials 4.0

Links

- CNES freeware server

Navigation

- Accueil
- Modifications récentes
- Page au hasard
- Aide

Outils

- Pages liées
- Suivi des pages liées
- Pages spéciales
- Adresse de cette version
- Information sur la page
- Citer cette page

- Dernière modification de cette page le 23 mai 2023 à 07:46.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-