Internationalisation

Bruno MERMET
Septembre 2010

Externalisation des ressources

Introduction

- Définition

  Séparation les « ressources » (texte, images, sons, etc.) du code

- Intérêt

  Pouvoir « localiser » un logiciel sans devoir le recompiler

- Principe

  Les ressources, chacune identifiée par une clé (chaîne de caractères), sont regroupées (*ResourceBundle*) et leur valeur est

  - soit stockées dans un fichier texte (pour les ressources de type texte)
  - soit produites par une classe indépendante

Identification d'un ResourceBundle

- Identifiant de base
  - Un nom
- Localisation
  - Un suffixe pour la langue
  - (Un deuxième suffixe pour le pays)
    Différencier anglais britannique/américain par exemple
  - (Un troisième suffixe pour le système/navigateur)
- Exemples d'un même ensemble de ressources :
  - mesTextes, mesTextes_fr
  - mesTextes_fr_FR, mesTextes_fr_FR_UNIX
  - mesTextes_en_GB

Spécifier un lieu

- Principe
  - Créer une instance de la classe java.util.Locale
    - Locale(String langue)
    - Locale(String langue, String pays)
    - Locale(String langue, String pays, String variant)
- Chaînes références
  - Constantes dans Locale (Locale.FRENCH, Locale.FRANCE, etc.).
  - Normes ISO-639 et ISO-3166
- Spécifier la localisation par défaut

  public static void setDefault(Locale l)

Recherche de la version locale d'un ensemble de ressources

- Ordre
  - Du plus précis au moins précis
  - D'abord pour la localisation spécifiée, puis pour la localisation par défaut
- Exemple
  - Contexte
    - Localisation spécifiée : fr_FR_WIN
    - Localisation par défaut : en_GB
  - Ordre de recherche
    - mesTextes_fr_FR_WIN, mesTextes_fr_FR, mesTextes_fr
    - mesTextes_en_GB, mesTextes_en, mesTextes
- Conséquence

  Pour éviter une exception, toujours prévoir une version non suffixée

Utilisation des ressources

- Création de l'objet Locale :

  Locale ici = new Locale(*maLangue*, *monPays*);
- Obtention de l'ensemble de ressources

  ResourceBundle mesRessources =
  ResourceBundle.getBundle(*monEnsemble*, ici);
- Obtention d'une ressource

  String maRessource =
  mesRessources.getString(*identifiantDeMaRessource*);

  (ou getObject)
- Retrouver la description des ressources
  - Via un ClassLoader
  - Les « . » sont remplacés par des « / »
  - Les classes sont prioritaires sur les fichiers texte

Spécifier des ressources dans des fichiers textes

- Application

  Uniquement pour les chaînes de caractères !

- Fichiers
  - Un fichier par localisation d'ensemble de ressource
  - Nom : *nomEnsembleRessources.localisation*.properties
  - Exemples
    - mesTextes_fr_FR.properties
    - mesTextes.properties
    - mesTextes_gb.properties
- Structure d'un fichier
  - Une ligne par ressource :

    clef = valeur
  - Des commentaires sur des lignes commençant par un #

Spécifier des ressources dans des classes

Format des classes

- Nom : nomEnsembleRessources_localisation
- Hériter de ListResourceBundle
- Définir la méthode Object[][] getContents() renvoyant un tableau de tableaux [clé, valeur]

Conseils de structuration

- Préférer plusieurs ensembles de ressource à un seul gros
- Regrouper les ressources selon une structuration logique
- Hiérarchiser les ensembles de ressources pour mieux les structurer

Affichage « localisé » des nombres, des dates et des heures

Nombres

- Problématique
  - Nombres, sommes d'argent, pourcentage ne sont pas affichées de la même façon suivant les pays
- Solution
  - Utiliser la classe NumberFormat en la paramétrant par une instance de java.lang.Locale
- Exemple

```
Locale ici = new Locale(« fr », « FR »);
Double d = new Double(1234.56);
NumberFormat nf = NumberFormat.getNumberInstance(ici);
String sortie = nf.format(d);
```

- Prolongation
  - getCurrencyInstance, getPercentInstance()

Dates et heures

- Utiliser les méthodes de la classe DateFormat :
  - DateFormat getDateInstance(int format, Locale ici)
  - DateFormat getTimeInstance(int format, Locale ici)
  - DateFormat getDateTimeInstance(int format, Locale ici)
- Format :
  - DateFormat.DEFAULT
  - DateFormat.SHORT
  - DateFormat.MEDIUM
  - DateFormat.LONG
  - DateFormat.FULL

Localisation des messages composés

Problématique

- Exemple
  - Soit un logiciel
    - demandant à un utilisateur
      - Son animal préféré (exemple : âne)
      - Sa couleur préférée (exemple : noir)
    - Et affichant une phrase de synthèse type :
      - Vous aimeriez avoir un âne noir
  - En anglais, traduire mot-à-mot donnerait
    - You would like to have a donkey black
  - Or la version correcte serait :
    - You would like to have a black donkey
- Le problème

  La phrase à afficher est paramétrée par des éléments eux-mêmes internationalisés
- Remarque : limiter ce genre de situation

Solution (1)

- Principe
  - Externationnaliser des phrases paramétrées
  - Utiliser la classe MessageFormat pour les concrétiser
- Concrètement
  - Dans la phrase, faire figurer les paramètres ainsi :
    - {numero} (pour les chaînes simples, par exemple)
    - {numero,typeFormat}
    - {numero,typeFormat,styleFormat}

typeFormat	StyleFormat possibles
number	integer, currency, percent
date	short, medium, long, full
time	short, medium, long, full

Solution (2)

- Application sur l'exemple
  - Fichier mesTextes_fr.properties
    - Chat = chat
    - Noir = noir
    - Synthese = Vous aimeriez avoir un petit {0} {1}.
  - Fichier mesTextes_en.properties
    - Chat = cat
    - Noir = black
    - Synthese = You would like to have a {1} {0}.

Solution (3)

- Création de la phrase de synthèse

```
Locale ici = new Locale(« fr », « FR »);

ResourceBundle ressources =
   ResourceBundle.getBundle(« mesTextes », ici);

Object [] parametres = {ressources.getString(« Chat »),
   ressources.getString(« Noir »)};

MessageFormat formateur = new MessageFormat(« »);

formateur.setLocale(ici);// pas indispensable ici où on n'a ni
   date ni nombre

formateur.applyPattern(ressources.getString(« Synthese »));

String sortie = formateur.format(parametres);
```

Localisation : gestion du pluriel

Exemple

- Exemple repris de http://download.oracle.com/javase/tutorial/i18n/format/choiceFormat.html
  - On veut afficher le nombre de fichiers trouvés sur un disque ; 3 cas :
    - *Il* n'y a pas de fichier *sur* monDisque
    - *Il* y a 1 fichier *sur* monDisque
    - *Il* y a 2 fichiers *sur* monDisque (ou plus)
  - Après avoir identifié les paramètres, on va stocker le motif et les différentes possibilités dans un ensemble de ressource, et utiliser un objet ChoiceFormat pour gérer ce qui dépend de la quantité
- Fichier de ressource mesTextes_fr_FR
  - Phrase = Il {0} sur {1}
  - Aucun = n'y a pas de fichier
  - Un = y a 1 fichier
  - Plusieurs = y a {2} fichiers

ChoiceFormat

Classe permettant un affichage conditionnel

- Soit le programme suivant :

```
Double[] limites = {0,2,6};
String [] textes = {« t1 », « t2 », « t3 »};
ChoiceFormat f = new ChoiceFormat(limites,textes);
For (int i = -1 ; i < 8 ; i++) {
        System.out.print(f.format(i)+ « »);
}
```

- L'affichage produit sera :

  t1 t1 t1 t2 t2 t2 t2 t3 t3

- Explications
  - Si limites[i] <= i < limites[i+1], on affiche textes[i]
  - Si i < limites[0], on affiche textes[0]
  - Si i > limites[limites.length-1], on affiche textes[textes.length-1]

Retour sur MessageFormat

- Rappel : utilisation vue de MessageFormat :

```
Object [] parametres = {new Double(1), « Chat », « Noir »};
MessageFormat formateur = new MessageFormat(« »);
formateur.setLocale(ici);
formateur.applyPattern(«Vous aimeriez avoir {2,number,integer} {0}
    {1} »);
String sortie = formateur.format(parametres);
```

- Mais on peut aussi écrire :

```
Object [] parametres = {new Double(1), « Chat », « Noir »};
MessageFormat formateur = new MessageFormat(« »);
formateur.setLocale(ici);
formateur.applyPattern(«Vous aimeriez avoir {2,} {0} {1} »);
Format [] f= {NumbertFormat.getIntegerInstance(ici),null,null};
formateur.setFormats(f);
String sortie = formateur.format(parametres);
```

MessageFormat et ChoiceFormat

- Récursivité

  Dans le cas où l'un des « formats » d'un paramètre d'un MessageFormat est ChoiceFormat, la substitution des paramètres est réappliquée une foit le ChoiceFormat appliqué

- Application

```
String disque = « monDisque »;//par exemple
int quantité = 1; //par exemple
String phrase = « Il {0} sur {1} »;
double[] limites = {0, 1, 2};
String[] debuts = {« n'y a pas de fichier », « y a un fichier », « y a {2}
    fichiers »};
ChoiceFormat cf = new ChoiceFormat(limites, debuts);
MessageFormat mf = new MessageFormat(« »);
mf.applyPattern(phrase);
Format [] f = {cf, null};
mf.setFormats(f);
System.out.println(mf.format({quantite,disque,quantite});
```