Localización en Java

La localización en Java se realiza mediante la clase `Locale` y el uso de `ResourceBundle`. Esta funcionalidad permite adaptar la aplicación a diferentes idiomas y regiones sin cambios en el código fuente. Aquí se resumen los conceptos clave, métodos y clases involucradas en la localización.

1. Clase Locale

La clase `Locale` identifica una región geográfica, política o cultural específica, que puede incluir:

- **Idioma**: Código de idioma, como `en` (inglés) o `fr` (francés).
- **Región**: Código de país o región, como `US` (Estados Unidos) o `CA` (Canadá).
- **Otros componentes**: Variante, script y extensiones.

Creación de Objetos Locale

Existen varias formas de crear un objeto `Locale`:

Constructores de Locale:

  - `Locale(String language)`: Crea un `Locale` solo con el idioma.
  - `Locale(String language, String country)`: Crea un `Locale` con idioma y región.
  - `Locale(String language, String country, String variant)`: Incluye una variante.

```
Locale deLocale = new Locale("de");
Locale ruLocale = new Locale("ru", "RU");
```

Método `forLanguageTag`: Permite crear un `Locale` a partir de una etiqueta de idioma en formato IETF BCP 47.

```
Locale enLocale = Locale.forLanguageTag("en-US");
```

Clase `Locale.Builder`: Construye objetos `Locale` usando métodos configuradores (`setLanguage`, `setRegion`, etc.) según la sintaxis IETF BCP 47.

```
Locale frLocale = new
Locale.Builder().setLanguage("fr").setRegion("CA").build();
```

Locales Predefinidos y Por Defecto

Java incluye constantes `Locale` predefinidas para algunos idiomas y regiones. Ejemplos:

```
Locale zhLocale = Locale.CHINESE; // Idioma chino
Locale itLocale = Locale.ITALY;   // Región Italia
Locale defaultLocale = Locale.getDefault(); // Locale por defecto
Locale.setDefault(Locale.FRANCE); // Cambia el Locale por defecto a
Francia
```

2. Clases de ResourceBundle

La clase `ResourceBundle` se utiliza para gestionar los recursos específicos de un `Locale`, como cadenas de texto. `ResourceBundle` permite la carga dinámica de recursos según el `Locale`.

Tipos de ResourceBundle

`PropertyResourceBundle`: Carga cadenas de texto desde archivos `.properties`.

```
ResourceBundle exam = ResourceBundle.getBundle("Exam");
```

Creación a partir de archivos:

```
PropertyResourceBundle bundle = new PropertyResourceBundle(new
FileReader("Exam.properties"));
```

`ListResourceBundle`: Recurso gestionado en un archivo `.java` específico para el `Locale`.

```
class Country_en_US extends ListResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] { { "capital", "Washington D.C." } };
    }
}
```

Métodos Clave de ResourceBundle

- `getBundle(String baseName)`: Carga el recurso de acuerdo al `Locale` actual.
- `getString(String key)`: Obtiene el valor asociado a una clave.

`getKeys` y `keySet()`: Devuelven todas las claves en el recurso.

```
ResourceBundle bundle = ResourceBundle.getBundle("Exam");
Set<String> keys = bundle.keySet();
keys.forEach(k -> System.out.println(k + ": " +
bundle.getString(k)));
```

3. Archivos de Propiedades

Los archivos `.properties` son archivos de texto que contienen pares de clave-valor, usados para almacenar cadenas de texto localizadas. Los nombres de archivo de los archivos de propiedades suelen seguir este formato:

```
<baseName>_<language>_<region>.properties
```

```
greeting = ¡Hola!
farewell = Adiós
```

4. Jerarquía de Búsqueda de ResourceBundle

Cuando un `Locale` específico no está disponible, `ResourceBundle` sigue una jerarquía de búsqueda para encontrar el recurso más cercano posible:

1. `MyBundle_fr_CA`
2. `MyBundle_fr`
3. `MyBundle_en_US`
4. `MyBundle_en`
5. `MyBundle`

Ejemplo de Localización con ListResourceBundle

Este ejemplo crea una implementación personalizada de `ListResourceBundle` para el `Locale en_US`:

```
class Country_en_US extends ListResourceBundle {
    protected Object[][] getContents() {
        return new Object[][] { { "area", 0 }, { "capital",
"Washington D.C." } };
    }
}
```

```
Locale us = new Locale("en", "US");
ResourceBundle bundle = ResourceBundle.getBundle("Country", us);
int area = (int) bundle.getObject("area");
String capital = bundle.getString("capital");
System.out.println("Area: " + area + ", Capital: " + capital);
```

Resumen

La localización en Java permite personalizar la aplicación para distintos `Locale` utilizando la clase `Locale` y `ResourceBundle`. Esta combinación facilita el uso de cadenas de texto y otros recursos en diferentes idiomas y regiones, optimizando el desarrollo de aplicaciones internacionalizadas.