Guía Avanzada de Localización en Java

1. Conceptos Clave de Localización

Localización (L10n) se refiere a la adaptación de una aplicación para diferentes idiomas, regiones y preferencias culturales de los usuarios.

Internacionalización (i18n) es el proceso de diseñar la aplicación para que soporte múltiples configuraciones regionales sin requerir modificaciones adicionales.

Principales Elementos para Localización en Java:

- **Locale**: Representa una región geográfica y cultural, incluyendo idioma y país.
- **ResourceBundle**: Gestiona y organiza recursos específicos para cada idioma/región.
- **Formatos de Datos (DateFormat, NumberFormat, etc.)**: Controlan el formato de datos como fechas y números.
- **Propiedades y Archivos de Recursos**: Archivos .properties que contienen recursos específicos para cada idioma.

2. Clase Locale

La clase `Locale` en Java representa una configuración regional específica.

```
Locale locale = new Locale("es", "ES");  // Idioma español para
España
Locale localeFr = Locale.FRENCH;         // Idioma francés
```

Creación de Objetos Locale

Constructor	Ejemplo
`new Locale(language)`	`new Locale("en")`
`new Locale(language, country)`	`new Locale("en", "US")`
`new Locale(language, country, variant)`	`new Locale("en", "US", "WIN")`

Métodos Útiles en Locale

- `getDisplayCountry()`: Devuelve el nombre del país.
- `getDisplayLanguage()`: Devuelve el nombre del idioma.
- `getISO3Country()`: Devuelve el código ISO de tres letras del país.

```
Locale locale = new Locale("fr", "FR");
System.out.println(locale.getDisplayCountry());  // Francia
```

3. ResourceBundle

`ResourceBundle` es una clase que gestiona el almacenamiento y acceso a los recursos específicos de cada idioma.

1. **Creación de Archivos de Recursos**:
    - Se crean archivos `.properties` para cada idioma, con nombres que incluyen el idioma y la región (por ejemplo, `MessagesBundle_en_US.properties`).
2. **Cargar un ResourceBundle**:
    - Java selecciona automáticamente el archivo que coincide con el `Locale` actual.

```
Locale locale = new Locale("es", "ES");
ResourceBundle bundle = ResourceBundle.getBundle("MessagesBundle",
locale);
String saludo = bundle.getString("greeting");
System.out.println(saludo);
```

Ejemplo de Archivos de Recursos:

`MessagesBundle_en_US.properties`:

properties
Copiar código
```
greeting=Hello
farewell=Goodbye
```

`MessagesBundle_es_ES.properties`:

properties
Copiar código
```
greeting=Hola
farewell=Adiós
```

4. Formato de Números y Monedas: NumberFormat

La clase `NumberFormat` permite dar formato a números, porcentajes y monedas de acuerdo con la configuración regional.

```
NumberFormat currencyFormatter =
NumberFormat.getCurrencyInstance(Locale.US);
System.out.println(currencyFormatter.format(1500)); // $1,500.00
```

Métodos Clave de NumberFormat

- `getCurrencyInstance(Locale)`: Para monedas.
- `getNumberInstance(Locale)`: Para números generales.
- `getPercentInstance(Locale)`: Para porcentajes.

```
NumberFormat percentFormatter = NumberFormat.getPercentInstance(new
Locale("fr", "FR"));
System.out.println(percentFormatter.format(0.75)); // 75 %
```

5. Formato de Fechas y Horas: DateFormat

`DateFormat` se usa para formatear y analizar fechas y horas.

```
DateFormat dateFormatter =
DateFormat.getDateInstance(DateFormat.LONG, Locale.UK);
System.out.println(dateFormatter.format(new Date())); // 1 January
2024
```

Niveles de Formato

- **FULL**: Muestra la fecha completa.
- **LONG**: Fecha detallada.
- **MEDIUM**: Fecha corta.
- **SHORT**: Fecha muy compacta.

```
DateFormat dateFormat =
DateFormat.getDateInstance(DateFormat.MEDIUM, Locale.ITALY);
System.out.println(dateFormat.format(new Date()));  // 1-gen-2024
```

6. Formateo de Mensajes con MessageFormat

`MessageFormat` facilita el formateo dinámico de mensajes, permitiendo la sustitución de valores en el texto.

```
Object[] args = {"Carlos", new Date()};
String message = MessageFormat.format("Hola, {0}. Hoy es {1}.",
args);
System.out.println(message); // Hola, Carlos. Hoy es 6 de noviembre
de 2024.
```

7. LocalDate y LocalTime (Java 8)

Con Java 8 y su API de tiempo, se mejora la internacionalización de fechas y horas.

Ejemplo de LocalDateTime con `DateTimeFormatter`

```
Locale locale = new Locale("es", "ES");
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE, d
'de' MMMM 'de' yyyy", locale);
LocalDate today = LocalDate.now();
System.out.println(today.format(formatter));  // miércoles, 6 de
noviembre de 2024
```

8. Prácticas Recomendadas de Localización

1. **Evita cadenas codificadas**: Usa `ResourceBundle` y `Locale` para almacenar texto e imágenes.
2. **Pruebas con diferentes Locales**: Ejecuta pruebas con distintos `Locale` para verificar la adaptabilidad de la aplicación.
3. **Gestión de configuraciones predeterminadas**: Configura valores `Locale` predeterminados para usuarios fuera de las regiones soportadas.

Este esquema cubre los conceptos fundamentales de localización en Java, siguiendo un formato y un estilo avanzado para la certificación y el dominio de los conceptos principales de `Locale`, `ResourceBundle`, `NumberFormat`, `DateFormat` y otros elementos de internacionalización.