Good Morning, Buenos Dias, Dobry den Mobile Internationalization in Action

Martin Brehovsky
NetBeans Mobility Pack
Sun Microsystems

Tomas Brandalik
Sun Java Wireless Toolkit
Sun Microsystems

TS-4589

Session Goal

Learn essentials of JSR 238 and how to develop localized MIDP applications in the NetBeans™ IDE and Sun Java™ Wireless Toolkit

Agenda

I18n and L10n Brief Introduction
Solving I18n Without JSR 238
JSR 238 Overview
JSR 238 API Details and Code Samples
JSR 238 and Sun Java Wireless Toolkit 2.5
I18n and L10n and NetBeans Mobility Pack
Q&A

Agenda

I18n and L10n Brief Introduction

Solving I18n Without JSR 238

JSR 238 Overview

JSR 238 API Details and Code Samples

JSR 238 and Sun Java Wireless Toolkit 2.5

I18n and L10n and NetBeans Mobility Pack

Q&A

Internationalization and Localization

Global World Requires Global Software

- I18n
  - Designing software products to support multiple languages and cultures
- L10n
  - After a product has been internationalized, this is the process of making it ready for a specific market
- Locale
  - String identification of user's language and country
  - Language (2 lower case letters)—country (2 upper case letters)—variant
  - en-US, ja-JP, zh-CN

Agenda

I18n and L10n Brief Introduction

Solving I18n Without JSR 238

JSR 238 Overview

JSR 238 API Details and Code Samples

JSR 238 and Sun Java Wireless Toolkit 2.5

I18n and L10n and NetBeans Mobility Pack

Q&A

Possible Solutions for I18n Without JSR 238

When the Tools Don't Help

- Use one class to hold localized resources
  - + Easy to develop, handles text and binary data
  - − Text mixed with source code, needs recompilation, memory consumption
- Store localized data into text file
  - + Easy to translate, no recompilation is necessary
  - − Needs custom parser, can handle only text data
  - − Parsing of larger text files can be slow

Agenda

I18n and L10n Brief Introduction

Solving I18n Without JSR 238

JSR 238 Overview

JSR 238 API Details and Code Samples

JSR 238 and Sun Java Wireless Toolkit 2.5

I18n and L10n and NetBeans Mobility Pack

Q&A

JSR 238 Overview

Mobile Internationalization API

- Java Community Process℠ services standardized Java ME optional package for development of localized applications
- Nokia—specification lead and reference implementation
- Mandatory part of MSA 1.0 (JSR 248)
  - Targeted platforms CLDC and CDC
- MSA RI → Sun Java Wireless Toolkit 2.5

JSR 238 Complements CLDC/MIDP

CLDC/MIDP

- **Character encoding**
  - Unicode
- **Locale identification**
  - Locale naming RFC3066, microedition.locale property
- **Dates, times, zones**
  - java.util.Date, Calendar, TimeZone
- **Stream IO**
  - java.io.InputStreamReader, OutputStreamWriter

JSR 238

- **Resource files**
  - Storage and access
- **Localized formatting**
  - Texts, numbers, date/times, currencies
- **Localized strings collation**

Resource Files

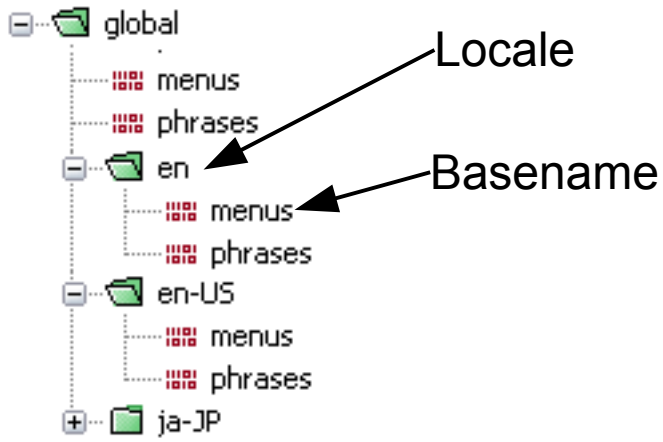
Tactics for Separation of Localizable Resources from the Code

- Application resources
  - Bundled into application jar
  - Supplied by application developer
  - Structure and format must adhere to JSR 238
- Device resources
  - Stored permanently on device
  - Supplied by manufacturer
  - Have public identifiers
  - Can't be change

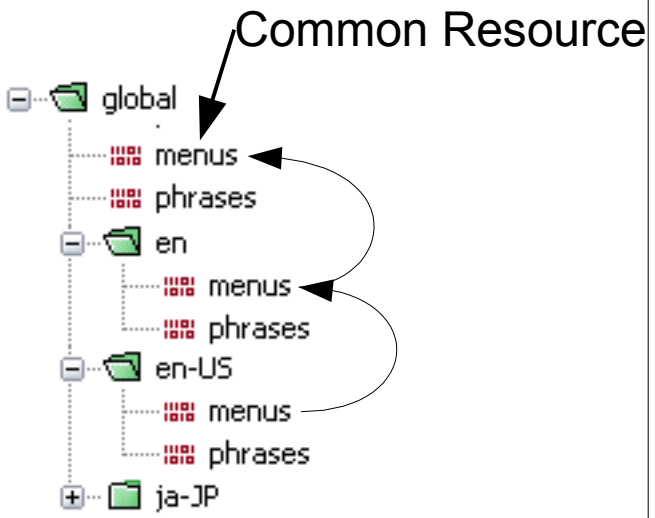
Resource Files

Identification and Inheritance

Resource File Is Identified by Locale and Base Name



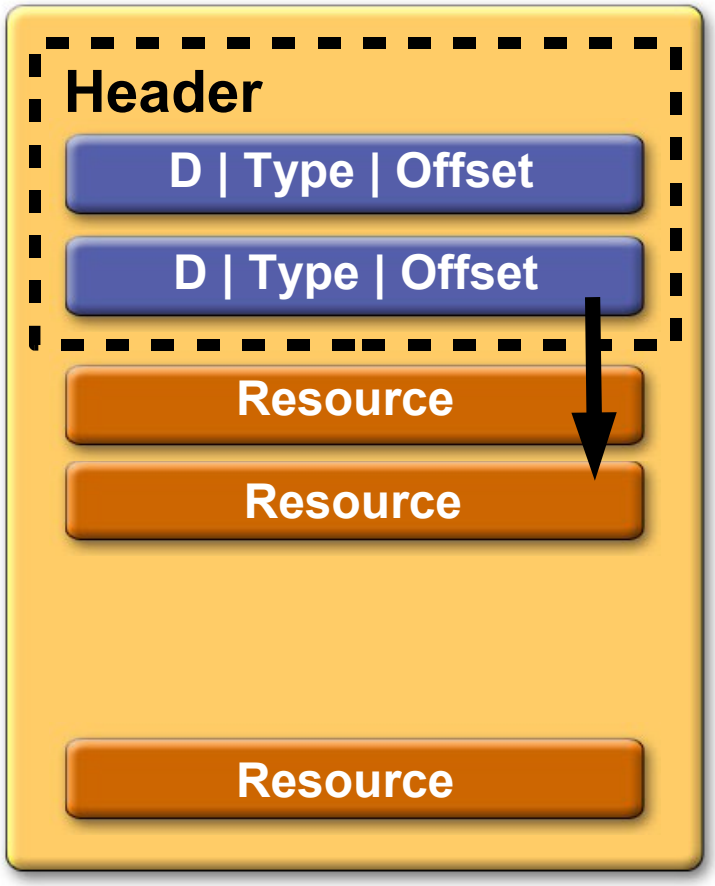
Resource Inheritance



Resource File Format

JSR 238 Defines Format for Application Resource Files

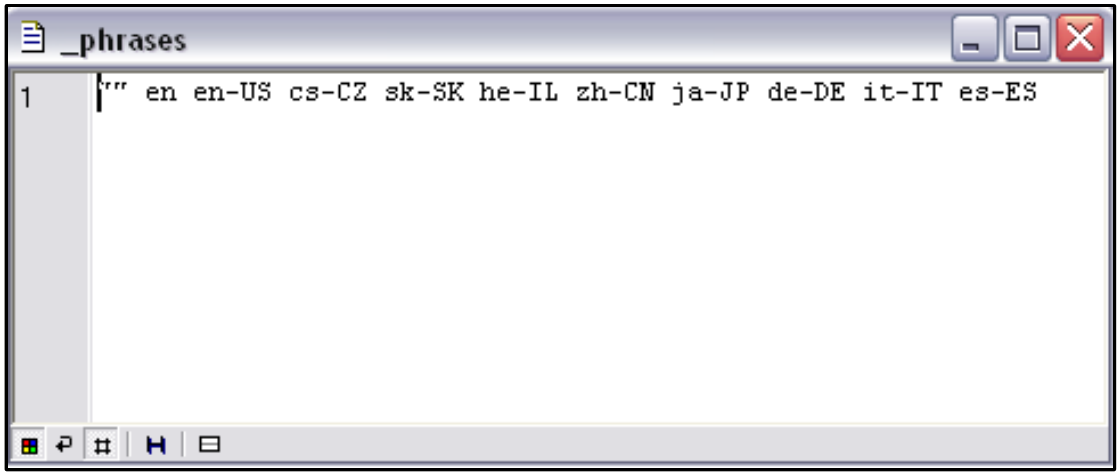
- Contains texts and binary data
- Indexed for faster access
- Efficient reading
- Takes less memory
  - id = integer
  - text encoding utf-8



Resource Metafiles

Meta-Information File for Each Base Name Speeds a Access

- Enumerates all locales of given resource file
- Metafile is checked before accessing resource in jar
- Simple text file



Agenda

I18n and L10n Brief Introduction

Solving I18n Without JSR 238

JSR 238 Overview

JSR 238 API Details and Code Samples

JSR 238 and Sun Java Wireless Toolkit 2.5

I18n and L10n and NetBeans Mobility Pack

Q&A

Accessing Resources (Part One)

Use Instance of ResourceManager to Retrieve Resources

```
import javax.microedition.global.*

try {
        // look for phrases under default locale
        rm = ResourceManager.getManager("phrases");

        // look for phrases under en-US locale
        rmUS = ResourceManager.getManager("phrases", "en-US");

        // get device resources
        rmDev = ResourceManager.getManager("");

    } catch (ResourceException re) {

    }
```

Accessing Resources (Part Two)

Retrieve Strings or Binary Data

```
public static final int STRING_MESSAGE_ID = 1234;
public static final int IMAGE_ID = 5678;

try {
            ResourceManager rm =
                    ResourceManager.getManager("common");

        // get string id is in range (0 .. 0x7fffffff)
        String s = rm.getString(STRING_MESSAGE_ID);

        // get image as binary data
        byte[] idata = rm.getData(IMAGE_ID);
        Image img = Image.createImage(idata, 0, idata.length);

        // get data as java.lang.Object
        Image img2 = (Image)rm.getResource(IMAGE_ID);

    } catch (ResourceException re) {
    }
```

Formatting

Locale Specific and Locale Neutral Formatting

- Neutral formatter
  - Renders numbers through toString() method
  - Date/time conforms to Internet date/time profile (ISO 8601)
- Locale specific formatter
  - Dates/times
  - Numbers (long, double)
  - Currency amounts
  - Percentages
  - Message formatting

Formatter API

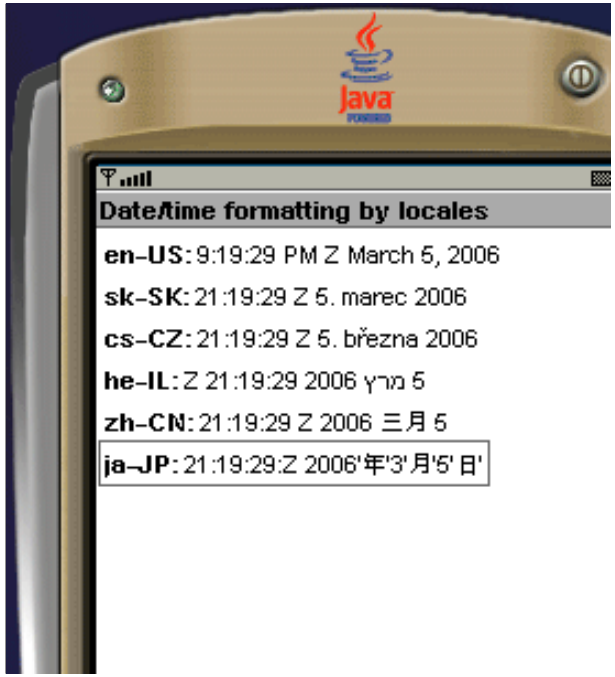
Formatter Instance Creation and Usage

```
// locale dependent formatter based on default locale
Formatter lFmt = new Formatter();

// locale dependent formatter
Formatter lFmt = new Formatter("en-US");

// locale neutral formatter
Formatter nFmt = new Formatter(null);

// important! Locales supported by formatter
String[] locales = lFmt.getSupportedLocales();
```

Date/Time Formatting

Use Styles to Determine Formatter Output

- Date/time styles
  - Short date, long date, short time, long time, short datetime, long datetime

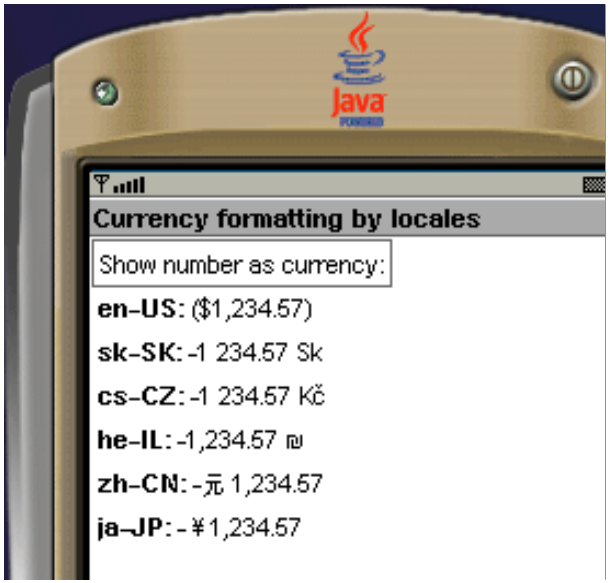



```
Calendar cal = Calendar.getInstance();
Formatter lFmt = new Formatter();

String output =
lFmt.formatDateTime(cal,DATETIME_LONG);
```

Formatting Numbers

Numbers, Currency Amounts, Percentages

```
// get formatter instance
Formatter lFmt = new Formatter();

// specify number of decimals in output
lFmt.formatNumber(doubleNumber, numOfDecimals);

// specify number of decimals in output
lFmt.formatPercentage(floatNumber, numOfDecimals);

// format currency
lFmt.formatCurrency(doubleNumber);
```



String Collation

Locale Specific Collation

- Uses Unicode Collation Algorithm 4.0
  - If locale is unspecified generic collation algorithm is used
  - Four levels of collation
    - Level 1—differences between alphabetical characters
    - Level 2—differences between accented characters
    - Level 3—character case differences
    - Identical—all differences

Source: http://www.unicode.org/reports/tr10

String Collation Usage

Create StringComparator to Compare Strings

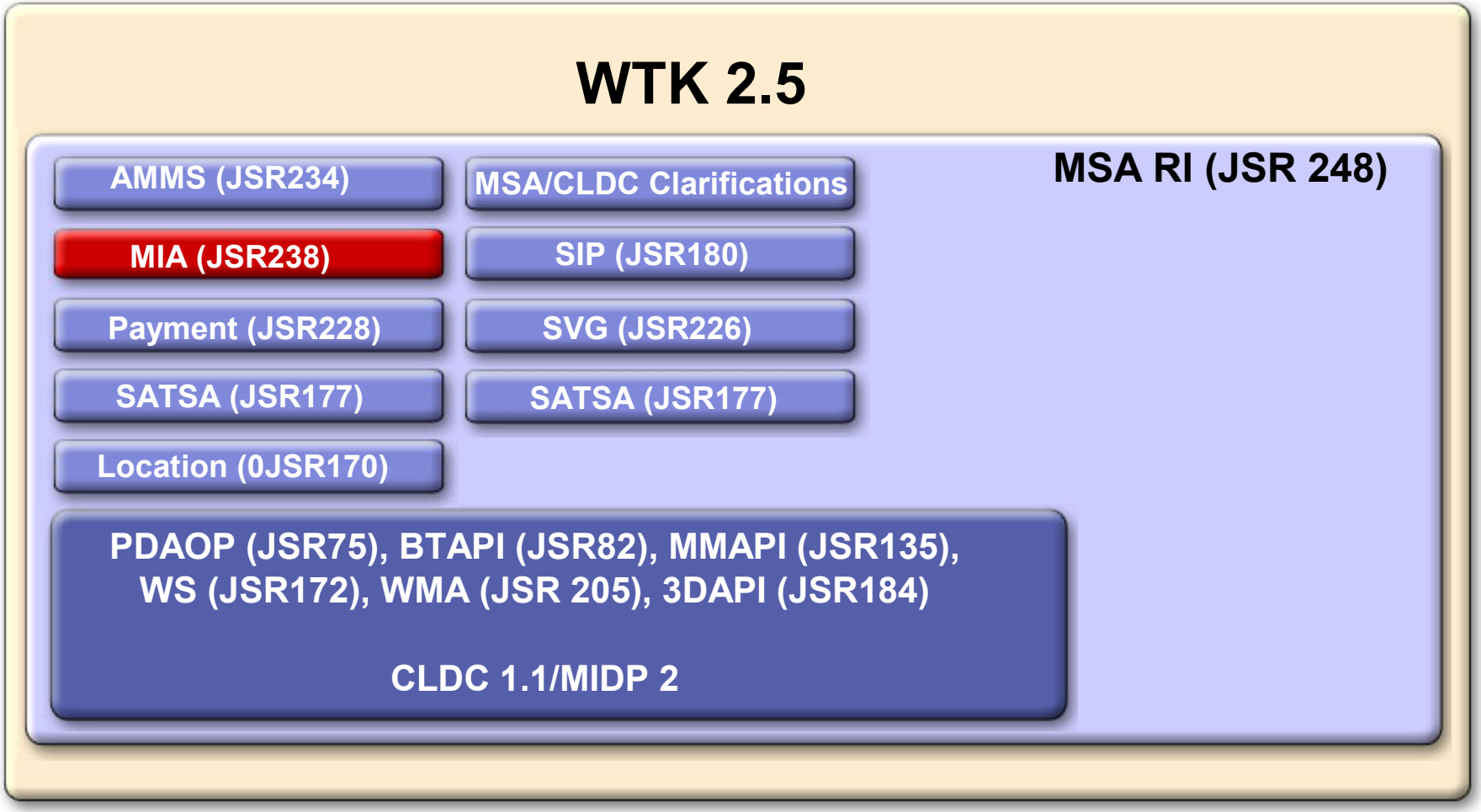
```
try {
  // get comparator for default locale
 defComparator =
      new StringComparator(null, StringComparator.IDENTICAL);

  // get comparator for japan locale
 skyComparator =
      new StringComparator("ja-JP",StringComparator.IDENTICAL);

  // compare 2 strings
  if (skyComparator.compare(str1, str2) > 0) {
      // swap(str1, str2);
  }

} catch (UnsupportedLocaleException ex) {
}
```

Agenda

I18n and L10n Brief Introduction
Solving I18n Without JSR 238
JSR 238 Overview
JSR 238 API Details and Code Samples
JSR 238 and Sun Java Wireless Toolkit 2.5
I18n and L10n and NetBeans Mobility Pack
Q&A

Sun Java Wireless Toolkit

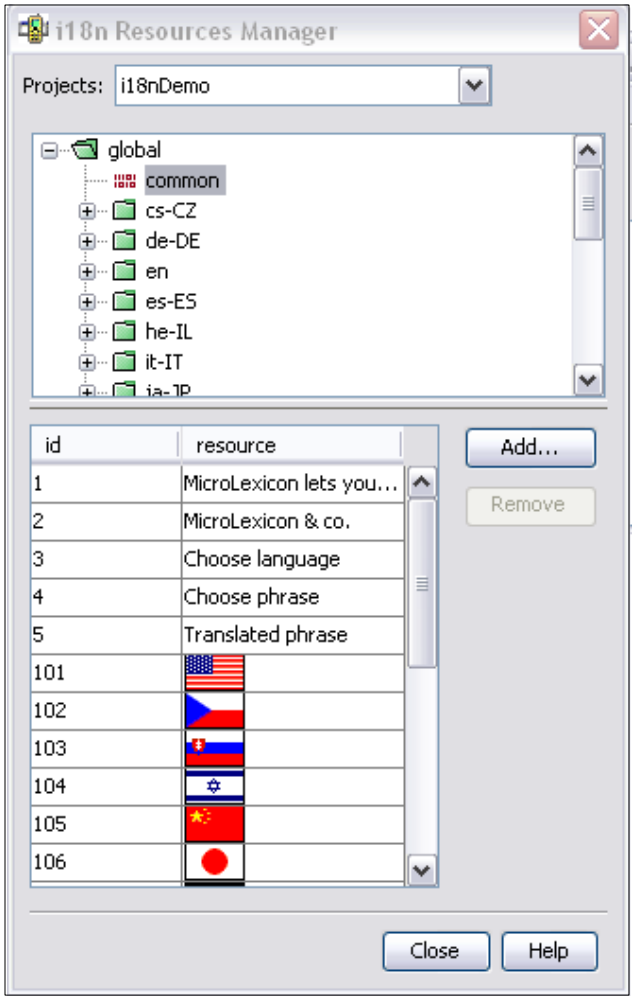
SR 238 Part of Next Major WTK Release



Resource Files Management in WTK 2.5

I18n Resource Manager

- Integrated into Sun Java Wireless Toolkit kToolbar
- Manage locales and resource files
- Add strings and binary data
- Drag and drop image files and property files



DEMO

Sun Java Wireless Toolkit

Agenda

I18n and L10n Brief Introduction
Solving I18n Without JSR 238
JSR 238 Overview
JSR 238 API Details and Code Samples
JSR 238 and Sun Java Wireless Toolkit 2.5
I18n and L10n and NetBeans Mobility Pack
Q&A

JSR-238 Support in NetBeans Mobility Pack

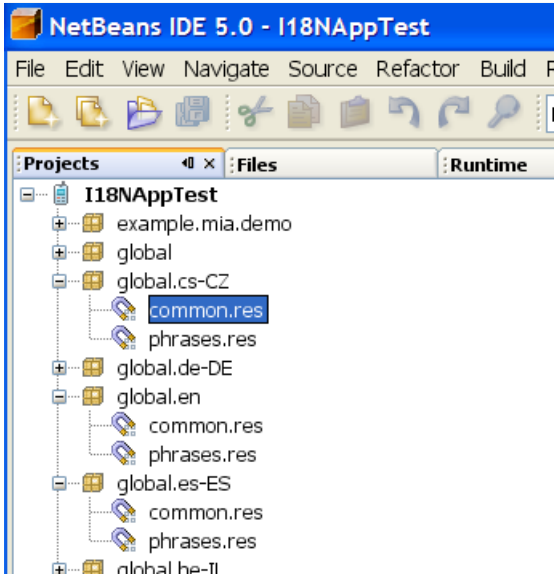
NetBeans Mobility Pack—Overview

- Complete and feature rich solution for Java ME/MIDP application development
  - Supports full development cycle for mobile applications: Edit, Build, Run, Debug, Deploy
  - Includes Visual Mobile Designer, tools for developing end-to-end applications, tools simplifying porting applications to different devices
  - Includes Sun Java Wireless Toolkit 2.2
- Uses NetBeans release schedule
- Free product
- For further info and downloads see http://www.netbeans.org/products/mobility/

JSR-238 Support in NetBeans Mobility Pack

Mobile Internationalization Support Module

- Experimental module for NetBeans Mobility Pack 5.0
- Provides specialized editor for JSR-238 resource files
- Automatically creates resource metafiles during the build process
- Use project configurations to create multiple localized distributions



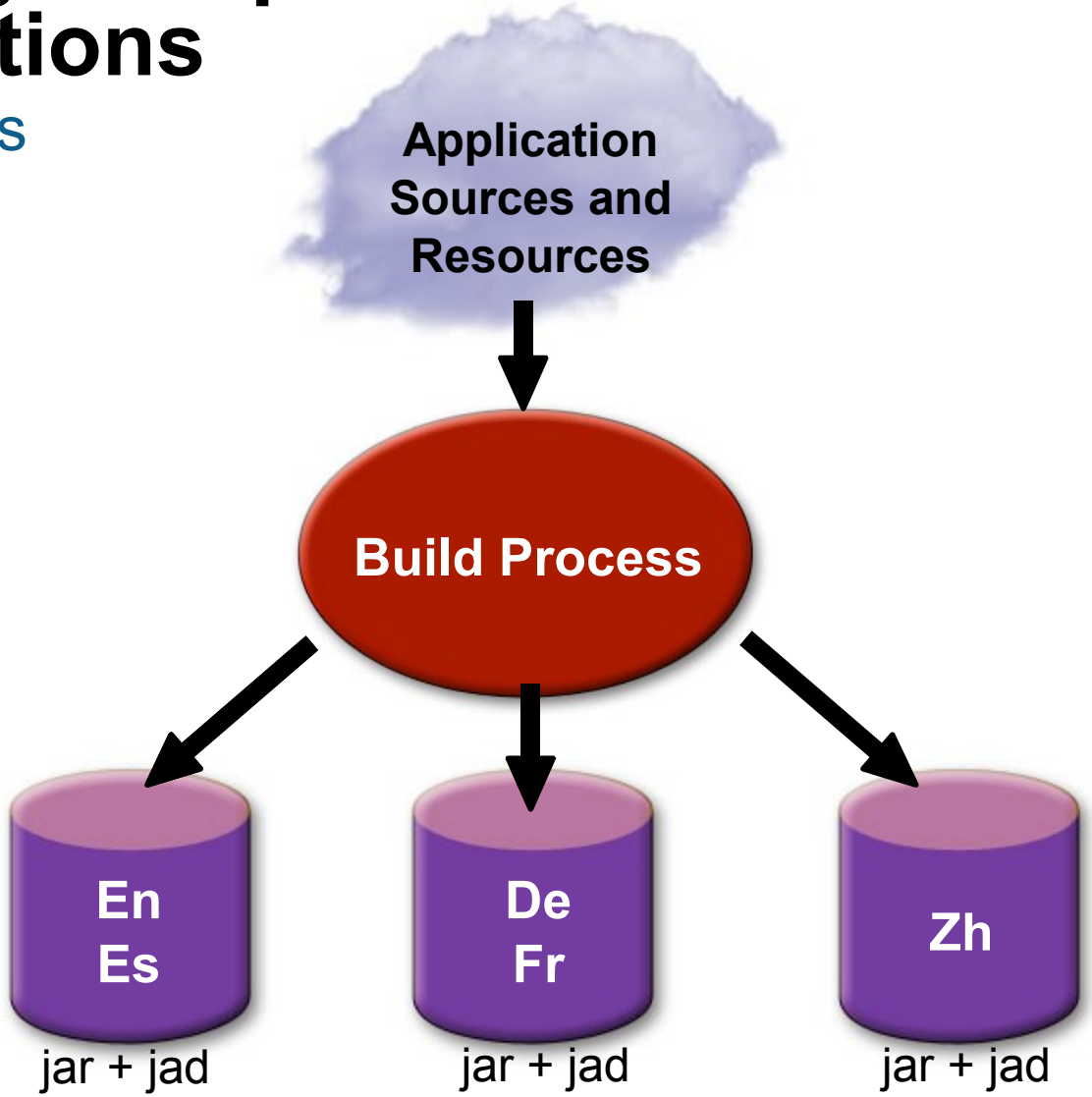
Creating Multiple Localized Distributions

Utilizing Mobility Pack's Project Configurations for I18n/L10n

- Using project configurations you can easily create multiple localized distributions of the application based on the same source code
- IDE automatically creates jar/jad files with appropriate localization
- MIDlet name can be easily localized
- Resource metafiles are automatically generated based on the resource files presented in the distribution

Creating Multiple Localized Distributions

How It Works



DEMO

NetBeans Mobility Pack

NetBeans I18N Future Plans

Consolidated I18N Support

- Provide more integrated I18n support in the IDE
- Common tools for Java SE/Java ME
- Seamless support in Visual Mobile Designer
- Wizards for externalization of Strings and other resources in the application
- Send us your feedback/wish list to nbusers@netbeans.org

Summary

- JSR 238 delivers a long awaited standard for I18n and L10n to CLDC/MIDP platform
- JSR 238 is part of MSA 1.0 (JSR 248 and JSR 249)
- WTK 2.5 contains JSR 238 support
- NetBeans Mobility Pack provides further support for developing I18n and L10n aware Java ME applications

For More Information

- JSR 238:
  http://www.jcp.org/en/jsr/detail?id=238
- Sun Java Wireless Toolkit 2.3 Beta:
  http://java.sun.com/products/sjwtoolkit/download-2_3.html
- NetBeans Mobility Pack:
  http://www.netbeans.org/products/mobility/
- Sun Java Wireless Toolkit BOFs:
  BOF-2461
- NetBeans Mobility Pack TS/BOFs:
  TS-1878, TS-3301, TS-5454, BOF-2704
- Look for us in tools area and mobility area of the Sun booth

Q&A

Tomas Brandalik
Martin Brehovsky

Good Morning, Buenos Dias, Dobry den Mobile Internationalization in Action

Martin Brehovsky
NetBeans Mobility Pack
Sun Microsystems

Tomas Brandalik
Sun Java Wireless Toolkit
Sun Microsystems

TS-4589