Welcome to the UI Theme Park—Customizing the Java ME User Experience With JSR 258

Jere Käpyaho and Jarmo Korhonen

Development Platforms Standardization
Nokia Corporation
http://www.nokia.com

TS-5114

Introducing Java™ Specification Request (JSR) 258, the Mobile User Interface Customization API

New look for your UI

Learn about the capabilities and uses of the new Mobile User Interface Customization API in Java Platform, Micro Edition (Java ME platform)

Agenda

Overview and Status

Goals of JSR 258

Use Cases for Java ME platform Applications

Features and Structure of Customization API

XML-Based Portable Data File Format

Demo

Summary

Agenda

Overview and Status

Goals of JSR 258

Use Cases for Java ME platform Applications

Features and Structure of Customization API

XML-Based Portable Data File Format

Demo

Summary

Why UI Customization for Mobiles?

For the sake of differentiation

- Users personalize
  - My wallpaper, my colors, my style
- Manufacturers and carriers emphasize brand and services
  - Brand identity
  - Corporate look and feel
  - Unified service applications
- Developers make their applications skinnable
  - Alternate look and feel

Management API and Common Format

Cross-platform development of themes and skins

- Before:
  - No way to manage themes and skins in Java platform
  - Proprietary skinning mechanisms
  - Several proprietary theme data formats
- After:
  - Programmatic theme and skin control
  - Skinnable Java ME platform applications
  - Common format for describing theme and skin data

Status and Availability

Final release available

- JSR led by Nokia in the Java Community Process℠ (JCP) services
  - Expert group members: Motorola, Sony Ericsson, BenQ, Sprint, Ericsson AB, Paul Petronelli, Esmertec, Cingular, Sun, Telecom Italia
  - Ran from late 2004 to early 2007
- Final release available for licensing
  - Technology compatibility kit and reference implementation developed by Nokia
  - File format freely usable without licensing the API

Agenda

Overview and Status

Goals of JSR 258

Use Cases for Java ME platform Applications

Features and Structure of Customization API

XML-Based Portable Data File Format

Demo

Summary

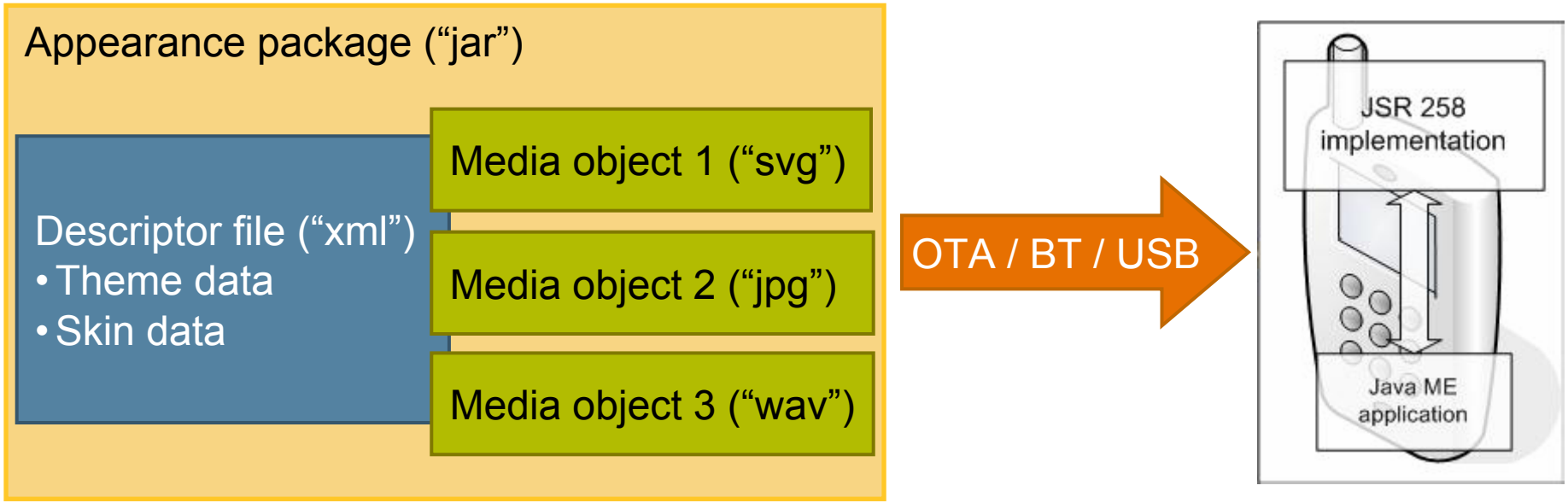
Goals of JSR 258

Unified customization API for Java ME platform applications

- Enable manipulation of UI themes and skins
- UI toolkit agnostic
  - Can be used with any Java ME platform UI toolkit
- Extensible framework for customizable properties
- Portability of customization data
- Handling DRM and security issues

High Level Model

Customization data and API implementation



Agenda

Overview and Status

Goals of JSR 258

Use Cases for Java ME Platform Applications

Features and Structure of Customization API

XML-Based Portable Data File Format

Demo

Summary

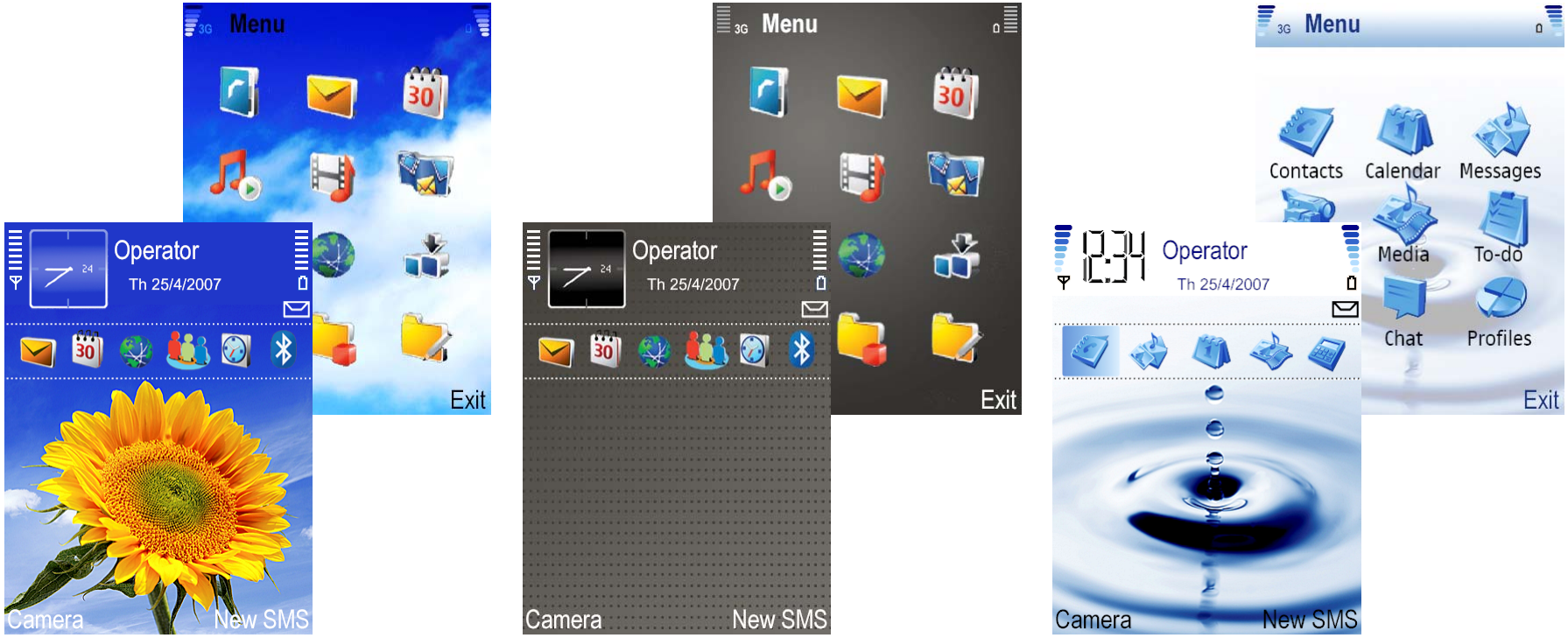
Use Cases for Java ME Platform Applications

Themes for system-wide effects

- Themes are for global appearance changes
- Preinstalled or downloadable appearance packages
- Personalization by users' personal interest
  - Movie/artist/sport/hobby, etc.
- Customization as a commercial promotional tool
  - Manufacturers (all users/specific customers)
  - Operators (subscribers/promotions of new services)
  - Corporations (employees/customers)

Use Cases for Java ME Platform Applications

Themes for system-wide effects



Three different Nokia S60 themes; same elements with different appearance

Use Cases for Java ME Platform Applications

Skins for application-specific effects

- Skins provide alternative UIs for applications
- Gives users a choice of different UI appearances
- Developers can provide alternative UIs without changing the code
- Shipped with application or downloaded later

Use Cases for Java ME Platform Applications

Skins for application-specific effects

Variant skins for the same application UI

Use Cases for Java ME Platform Applications

Querying customization data for custom components

- Create custom components that blend with the system appearance
- **Step 1:** Query system's current appearance data
  - Media objects
  - Color values
  - Primitive values
- **Step 2:** Use the data when painting custom components in an application

Use Cases for Java ME Platform Applications

Querying customization data for custom components

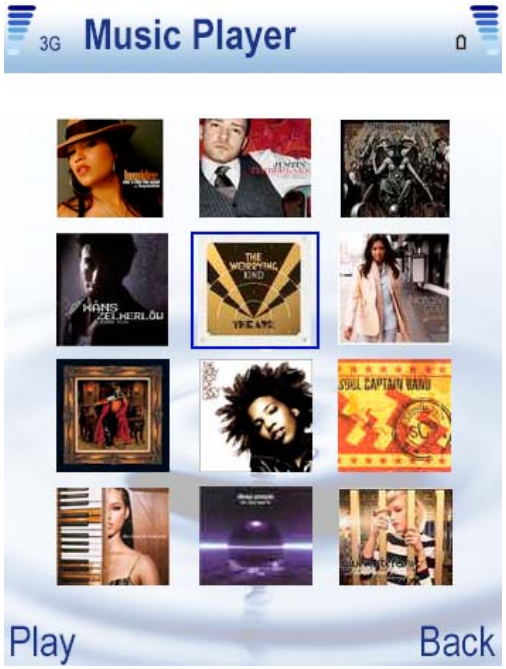
- Look of custom component is derived from system theme data

System theme appearance

Query, for example:

- Highlight color
- List border color
- Background image

Custom component in an application



Agenda

Overview and Status

Goals of JSR 258

Use Cases for Java ME platform Applications

Features and Structure of Customization API

XML-Based Portable Data File Format

Demo

Summary

Features of the Java ME Platform API

What can the Mobile UI Customization API do for you?

- Query and modify customization properties
  - What properties are there?
  - What content is allowed for each property?
- List and activate themes and skins
  - Show list to user and allow selection
  - Activate to see changes in system or application
- Listen to changes in themes and skins
  - Activated, modified, removed…

Features of the Java ME Platform API (Cont.)

Building blocks: themes, skins and elements

- Customizable elements
  - Identified by kind and role
  - Content is media or value
- Both themes and skins have elements inside
- Themes have system-wide elements
- Skins have application-specific elements
- Themes and skins have metadata
  - Name, author and creation data

Structure of the API

Classes and interfaces for mobile UI customization

- ThemeManager and SkinManager
  - Query installed themes and skins
  - Get reference to any theme or skin
  - Register as a change listener
- Theme and skin
  - Query metadata
  - Get information about customizable elements
- Element sub hierarchy
  - Media, colors, simple values
- Listener and exception classes

Getting and Using Themes

Using ThemeManager and theme names

- Themes are identified by unique name
- System default theme always present
  - Theme name is “_default_”
- For lists, show titles from metadata
  - Provides localizable titles for use in the UI
- ThemeManager methods work by name
  - activateTheme(*name*)
  - removeTheme(*name*)
  - createTheme(*name*) (optional feature)

Getting Themes and Activating Them

```
ThemeManager tm = ThemeManager.getInstance();
String[] themeNames = tm.getThemes();
// Insert theme names into a list component:
for (int i = 0; i < themeNames.length(); ++i) {
    String title = getThemeTitle(); // from metadata
    themeList.append(title);
}

// Get user selection
int sel = themeList.getSelectedIndex();
String name = themeNames[sel];
try {
    tm.activateTheme(name);
}
catch (ThemeException te) {
}
```

Registering as Theme Listener

```
public class MyThemeSwitcher implements ThemeListener {
    ThemeManager tm = ThemeManager.getInstance();
    boolean updateElements;

    public MyThemeSwitcher() {
        tm.addThemeListener(this);
    }

    public void themeActivated(Theme new, Theme old) {
        updateElements = true;
        // Repaint any UI components that depend on
        // theme contents.
    }

    // ... and other listener interface methods here
}
```

Get Element From Active Theme

```
ThemeManager tm = ThemeManager.getInstance();
Theme activeTheme = tm.getActiveTheme();
Element background;

// If a theme is activated, elements used for
// painting need to be fetched again
private void getPaintElements() {
    if (updateElements) {
        background = activeTheme.getElement(
            "activeScreen", "background");
        updateElements = false;
    }
}

public void paint(Graphics g) {
    getPaintElements();
    // render element(s)
}
```

Getting Skins and Activating Them

Using SkinManager and name/version/vendor

- One SkinManager instance per application
- Identified by name/version/vendor triplet
  - Agreed between developer of skinnable application and skin creators
- Applications start in skinless state
  - Can be restored later
- Can activate any skin known by SkinManager

Get Skins of Application

```
SkinManager sm = SkinManager.getInstance(
    "SkinSample:1.0:com.nokia");
String[] skinNames = sm.getSkins();

// user has selected skin, retrieve it
Skin skin = sm.getSkin(skinNames[selectedIndex]);

Element elem = skin.getApplicationElement(
    "playerButton", "play");

// create button with contents from element...
```

Base, Toolkit and Application Elements

Three sets of customizable elements

- Customizable elements come in three sets
- Base elements
  - Common to all implementations
- Toolkit elements
  - Apply to specific UI toolkits
- Application elements
  - Semantics depend on application
  - Developers communicate to skin makers

Element Lookup for Base and Toolkit

Layered approach to customization

- Need feature and role to look for element
- If supported, eventually finds at least the default customization (or throws exception)
- Lookup order: referring theme → skin → active theme → system default
- Lookup does not apply to application elements

Agenda

Overview and Status
Goals of JSR 258
Use Cases for Java ME platform Applications
Features and Structure of Customization API
XML-Based Portable Data File Format
Demo
Summary

XML-Based Portable File Format

Themes and skins packaged in a Java™ Archive (JAR) file

- Appearance descriptor is an application of XML 1.0
- Can be used without licensing the API
- Schema defined using RELAX NG
- Everything packaged in a JAR file
- Descriptor can refer to media elements
  - Relative URIs inside JAR file
  - Any media with a MIME type
- Implementation must parse and store appearance data

Appearance Descriptor Snippet

```
<appearance version="1.0">
    <theme name="ColorfulDay" version="1.0"
         vendor="com.nokia">
        <themeinfo> ... </themeinfo>
        <context name="generic">
            <base>
                <element feature="list"
                    role="itemHighlight"
                    kind="graphic" type="image/svg+xml"
                    src="2773874.svg" />
            </base>
            <toolkit name="midp20lcdui">
                <element feature="list"
                    role="background" ... />
            </toolkit>
        </context>
    </theme>
</appearance>
```

Agenda

Overview and Status
Goals of JSR 258
Use Cases for Java ME platform Applications
Features and Structure of Customization API
XML-Based Portable Data File Format
Demo
Summary

DEMO

Using the JSR 258 API

Agenda

Overview and Status
Goals of JSR 258
Use Cases for Java ME platform Applications
Features and Structure of Customization API
XML-Based Portable Data File Format
Demo
Summary

Summary

- JSR 258 provides theme and skin control for Java ME platform applications
- Gives access to data from system themes
- Enables skinnable Java ME platform applications
- Uses a portable XML-based descriptor format for theme and skin data
- Available for licensing from Nokia

For More Information

Visit:

- JCP services site of JSR 258
  http://www.jcp.org/en/jsr/detail?id=258
- Themes at Forum Nokia
  http://forum.nokia.com/main/resources/technologies/nokia_themes.html

Q&A

Welcome To the UI Theme Park – Customizing the Java ME User Experience With JSR 258

Jere Käpyaho and Jarmo Korhonen

Development Platforms Standardization
Nokia Corporation
http://www.nokia.com

TS-5114