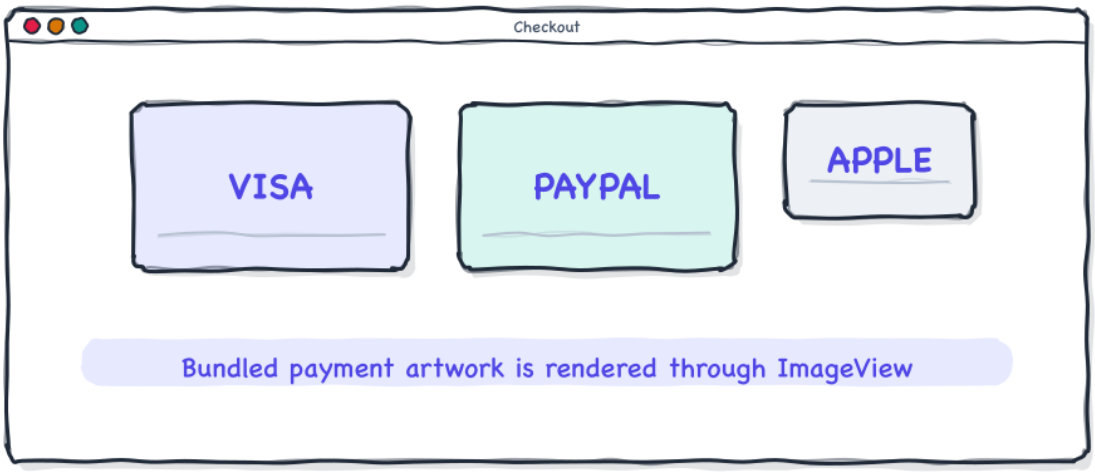
PaymentOptionView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



PaymentOptionView displays payment option artwork with dark or light bundled PNG resources.

PaymentOptionView is a small ImageView subclass that renders bundled payment option images. The option enum chooses a provider or card brand, the theme enum chooses dark or light artwork, and ImageView sizing preserves the resource aspect ratio.

Table of Contents

1. Introduction

PaymentOptionView extends `ImageView`. It does not use a skin or CSS file; changing the option or theme selects a bundled PNG resource from `paymentoptions/`.

1.1 Key features

- 55 option enum constants including card networks, wallets, bank transfer and providers.
- Two resource themes: DARK and LIGHT.
- Default option MASTERCARD and default theme DARK.
- Default fitWidth 100 with preserveRatio true.
- UNKNOWN clears the image.

1.2 Maven dependency

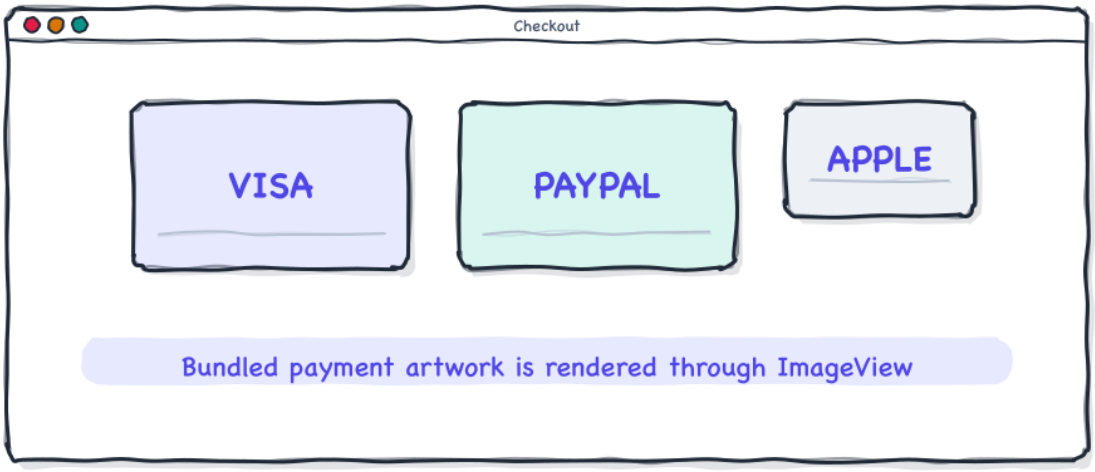
```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

GemsFX controls live in module `com.dlsc.gemsfx`.

2. Getting started

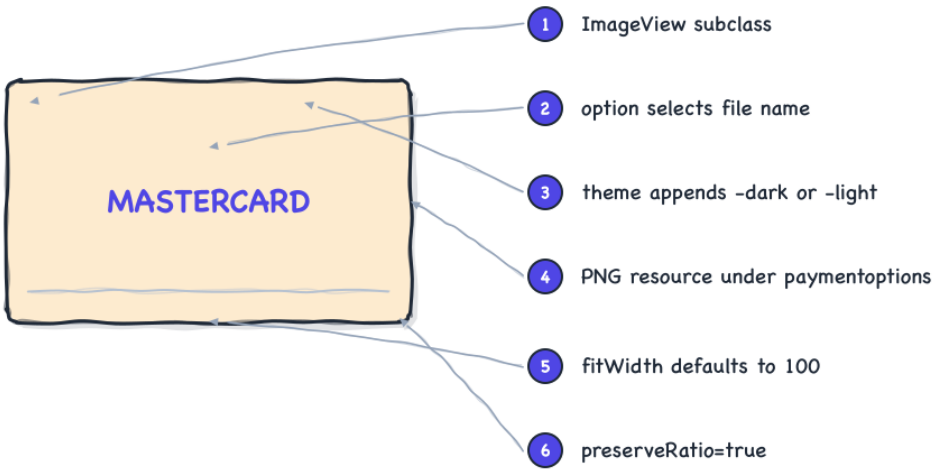
```
PaymentOptionView view = new PaymentOptionView();

ComboBox<PaymentOptionView.Option> optionsBox = new ComboBox<>();
optionsBox.getItems().setAll(PaymentOptionView.Option.values());
optionsBox.valueProperty().bindBidirectional(view.optionProperty());

ComboBox<PaymentOptionView.Theme> themeBox = new ComboBox<>();
themeBox.getItems().setAll(PaymentOptionView.Theme.values());
themeBox.valueProperty().bindBidirectional(view.themeProperty());

view.setFitWidth(160);
view.setPreserveRatio(true);
```

The PaymentOptionApp demo binds option and theme combo boxes directly to the view.



Payment option artwork in a checkout-style layout.

3. Anatomy



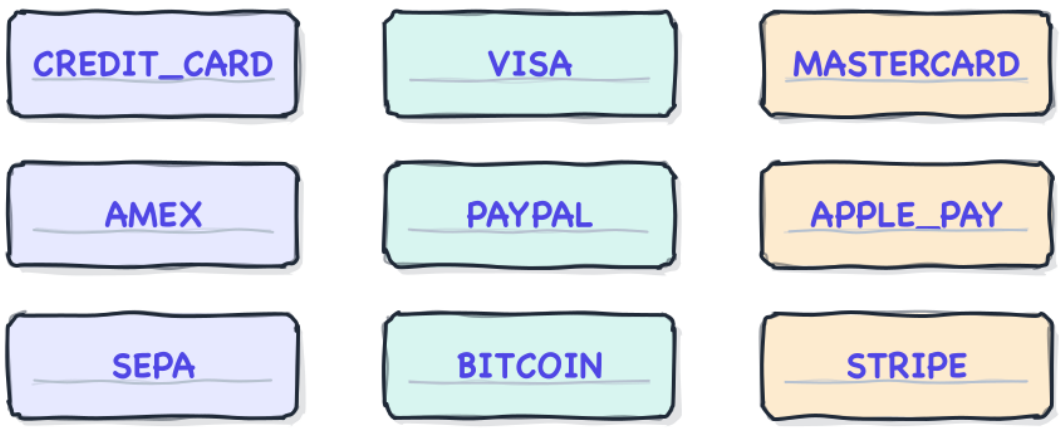
How option, theme and ImageView sizing work together.

Part	Description
ImageView	The control itself; no skin is created.
option	Selects the base resource file name.
theme	Appends -dark.png or -light.png.
paymentoptions resources	Bundled PNG images, 500 by 300 in the source comment.
fitWidth	Set to 100 by the constructor.
preserveRatio	Set to true by the constructor.

4. Control API

Property	Type	Default	Description
option	ObjectProperty<Option>	MASTERCARD	Payment option artwork to display. UNKNOWN clears the image.
theme	ObjectProperty<Theme>	DARK	DARK uses colored backgrounds; LIGHT uses white-background artwork.
fitWidth	DoubleProperty from ImageView	100	Constructor-set ImageView width.
preserveRatio	BooleanProperty from ImageView	true	Constructor-set ImageView ratio preservation.

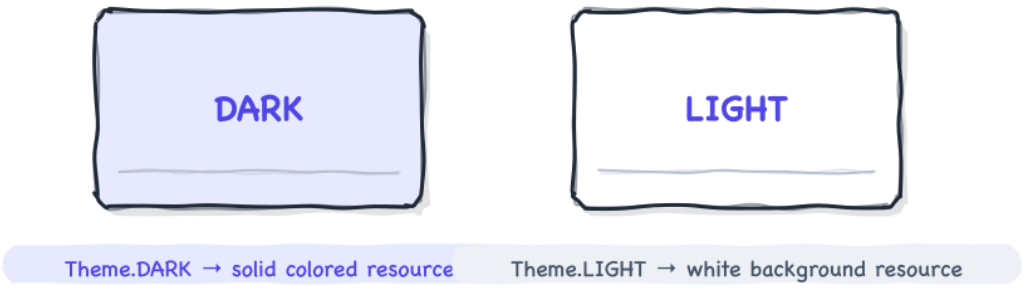
5. Option enum



Representative payment options from the enum.

`UNKNOWN, CREDIT_CARD, CHECKOUT2, ALI_PAY, AMAZON, AMERICAN_EXPRESS, APPLE_PAY, BANCONTACT, BITCOIN, BITPAY, CIRRUS, CLICKANDBUY, COINKITE, DINERSCLUB, DIRECTDEBIT, DISCOVER, DWOLLA, EBAY, EWAY, GIROPAY, GOOGLEWALLET, INGENICO, JCB, KLARNA, LASER, MAESTRO, MASTERCARD, MONERO, NETELLER, OGONE, OKPAY, PAYBOX, PAYMILL, PAYONE, PAYONEER, PAYPAL, PAYSAFECARD, PAYU, PAYZA, RIPPLE, SAGE, SEPA, SHOPIFY, SKRILL, SOLO, SQUARE, STRIPE, SWITCH, UKASH, UNIONPAY, VERIFONE, VERISIGN, VISA, WEBMONEY, WESTERNUNION, WORLDPAY`

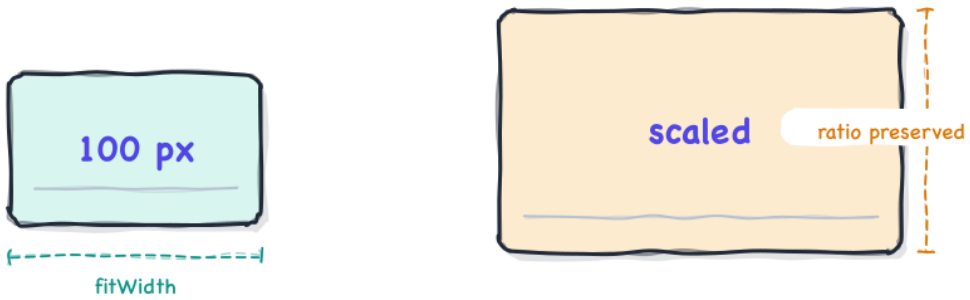
6. Theme and resources



The two themes map to -dark.png and -light.png resources.

- Resource file names are hard-coded in updateView().
- Most options map directly to a provider name such as Visa, Paypal or MasterCard.
- UNKNOWN leaves the file name blank and sets image to null.
- Invalid enum handling in the switch throws an exception for unexpected values.

7. Sizing and layout



PaymentOptionView uses ImageView sizing; preserveRatio keeps card proportions.

Because the class extends ImageView, use standard ImageView sizing APIs such as fitWidth, fitHeight, preserveRatio and smooth. The constructor initializes only fitWidth and preserveRatio.

The demos keep the artwork at a consistent width and let layout containers do the rest: `PaymentOptionApp` centers one view above option/theme selectors, while `PaymentOptionTilesApp` places one view for every enum constant in a padded `TilePane` with 10-pixel gaps.

8. Styling

There is no `payment-option-view.css`, no style class added by the class, and no styleable CSS properties. Style layout containers around the ImageView or use ImageView effects directly.

9. Localization and accessibility

PaymentOptionView does not use ResourceBundleManager and does not set an AccessibleRole or accessible text in the verified source. Applications that need screen-reader text should set accessibleText themselves, for example to the selected brand name.

10. Recipes

10.1 Option selector

```
ComboBox<PaymentOptionView.Option> optionsBox = new ComboBox<>();
optionsBox.getItems().setAll(PaymentOptionView.Option.values());
optionsBox.valueProperty().bindBidirectional(view.optionProperty());
```

10.2 Theme selector with contrasting background

```
themeBox.valueProperty().bindBidirectional(view.themeProperty());
themeBox.valueProperty().addListener(it -> {
    switch (themeBox.getValue()) {
        case DARK -> parent.setStyle("-fx-background-color: white;");
        case LIGHT -> parent.setStyle("-fx-background-color: rgb(57,73,92);");
    }
});
```

10.3 Tile all options

```
TilePane pane = new TilePane();
pane.setHgap(10);
pane.setVgap(10);
for (PaymentOptionView.Option option : PaymentOptionView.Option.values()) {
    PaymentOptionView tile = new PaymentOptionView();
    tile.setFitWidth(100);
    tile.setPreserveRatio(true);
    tile.themeProperty().bind(theme);
    tile.setOption(option);
    pane.getChildren().add(tile);
}
```

10.4 Clear unsupported payment

```
view.setOption(PaymentOptionView.Option.UNKNOWN);
```

10.5 Checklist

1. Use combo boxes or model bindings for option and theme selection.
2. Use UNKNOWN for unsupported or not-yet-detected payment types.
3. Keep preserveRatio true unless intentionally distorting artwork.
4. Use TilePane for a compact gallery of many payment options.
5. Set accessibleText in the application when the image conveys payment meaning.
6. Do not expect CSS hooks; it is an ImageView subclass.

11. See also

- Demo app: `PaymentOptionApp`. Run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.PaymentOptionApp`.
- Demo app: `PaymentOptionTilesApp`. Run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.PaymentOptionTilesApp`.
- Related JavaFX class: ImageView.
- API docs: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/