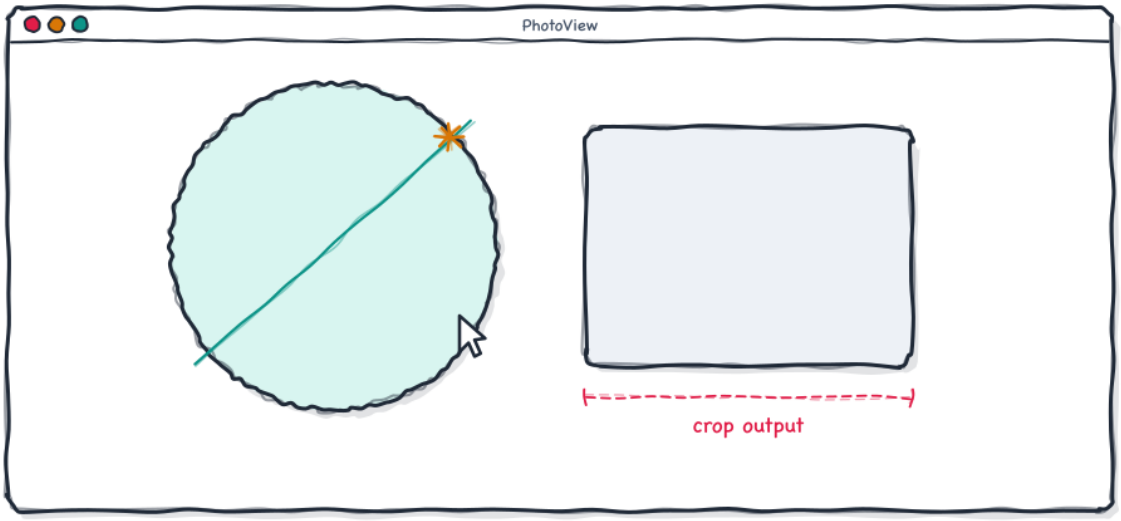
PhotoView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



Generated cartoon overview of PhotoView.

PhotoView displays and optionally edits a profile image. Users can load, drag, zoom, clip and crop an image while the control exposes the original and cropped versions.

Table of Contents

1. Introduction 2

1.1 Key features 2

1.2 Maven dependency 2

2. Getting started 2

3. Anatomy 3

4. Control API 4

4.1 Properties and callbacks 4

5. Behaviour 4

6. Layout and rendering 5

7. Styling 6

7.1 Style classes and pseudo classes 6

7.2 Styleable CSS properties 6

8. Localization 7

9. Accessibility 7

10. Recipes 7

10.1 Programmatic configuration 7

10.2 Practical checklist 7

11. Integration notes 7

12. See also 8

1. Introduction

PhotoView PhotoView displays and optionally edits a profile image. Users can load, drag, zoom, clip and crop an image while the control exposes the original and cropped versions.

1.1 Key features

- Editable by default with mouse drag, scroll wheel and pinch zoom.
- Default placeholder invites dropping or clicking to add an image.
- ClipShape.CIRCLE is the default; RECTANGLE is also supported.
- createCroppedImage defaults to true and updates croppedImage after a 200 ms delay.
- Supported drag/drop file extensions are .bmp, .png, .gif, .jpg and .jpeg.
- BACK_SPACE and DELETE remove the photo; SPACE and ENTER invoke the supplier.

1.2 Maven dependency

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

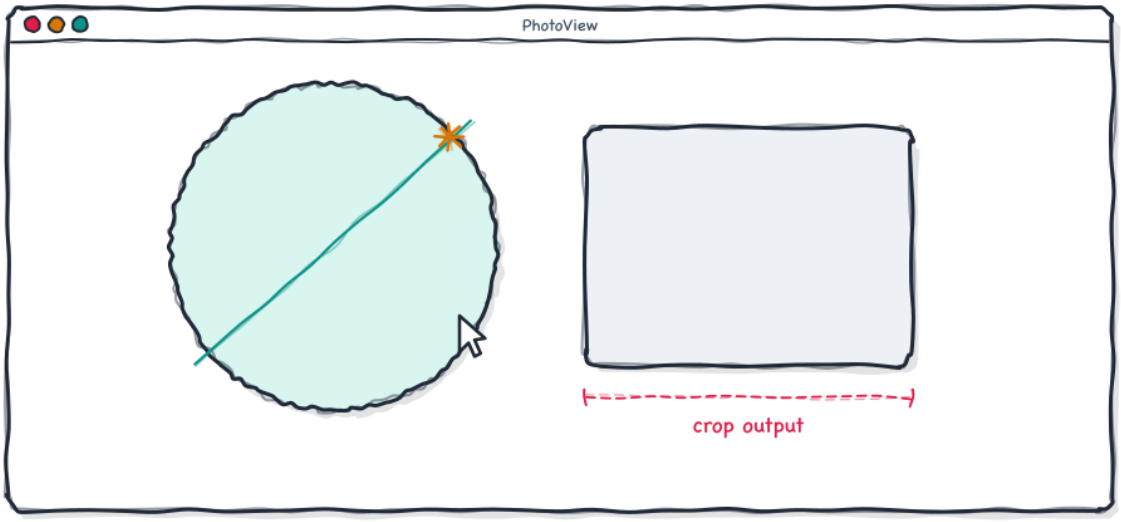
The control lives in module `com.dlsc.gemsfx`.

2. Getting started

The following snippet uses only public API verified in the control source.

```
PhotoView photoView = new PhotoView();
photoView.setClipShape(PhotoView.ClipShape.CIRCLE);
photoView.setMaxZoom(4);
photoView.setPhotoSupplier(() -> loadImageFromApplicationDialog());
photoView.croppedImageProperty().addListener((obs, old, img) -> savePreview(img));
```

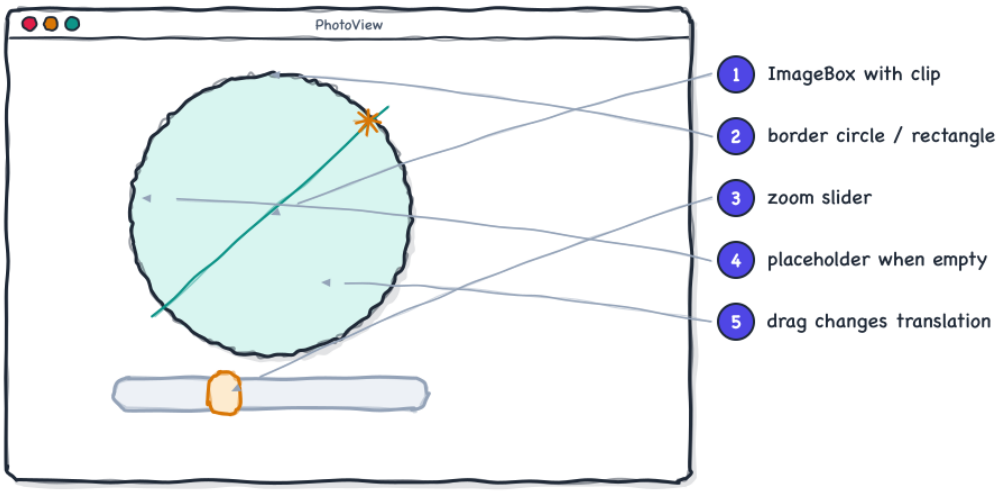
A compact setup for PhotoView.



A generated overview of PhotoView in use.

3. Anatomy

The anatomy diagram identifies the implementation pieces that matter when configuring, styling or debugging the control.



The parts of PhotoView.

Part	Verified detail
Root	Style class `.photo-view` is added by the constructor.
Stylesheet	User-agent stylesheet `photo-view.css` is returned by the control.

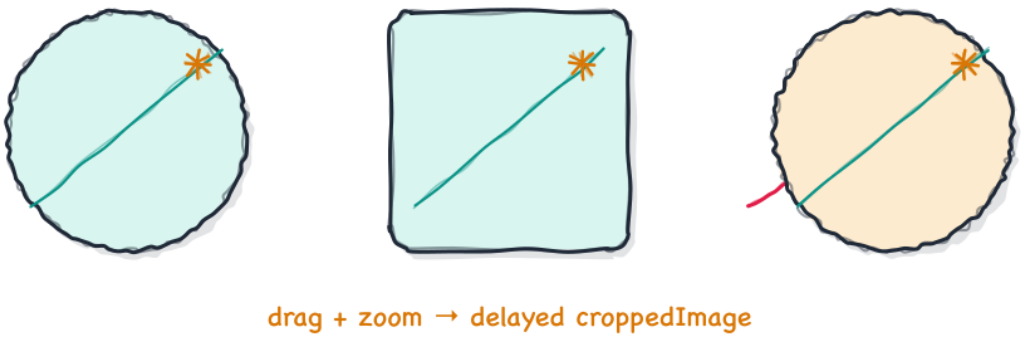
4. Control API

4.1 Properties and callbacks

Property	Type	Default	Description
photo	ObjectProperty<Image>	null	Original image.
croppedImage	ReadOnlyObjectProperty<Image>	null	Cropped image produced by the skin.
createCroppedImage	BooleanProperty	true	Enables delayed crop generation.
photoEffect	ObjectProperty<Effect>	null	Effect applied to the ImageView only.
placeholder	ObjectProperty<Node>	localized Label with upload icon	Shown when photo is null, editable is true and photoSupplier is non-null.
editable	BooleanProperty	true	Enables gestures and slider; styleable.
photoSupplier	ObjectProperty<Supplier<Image>>	FileChooser supplier	Provides images for click, SPACE and ENTER.
clipShape	ObjectProperty<ClipShape>	CIRCLE	CIRCLE or RECTANGLE; styleable.
photoZoom	DoubleProperty	1	User zoom; slider min is 1.
photoTranslateX / photoTranslateY	DoubleProperty	0	Percentage-based image translations.
maxZoom	DoubleProperty	5.0	Maximum user zoom; styleable.

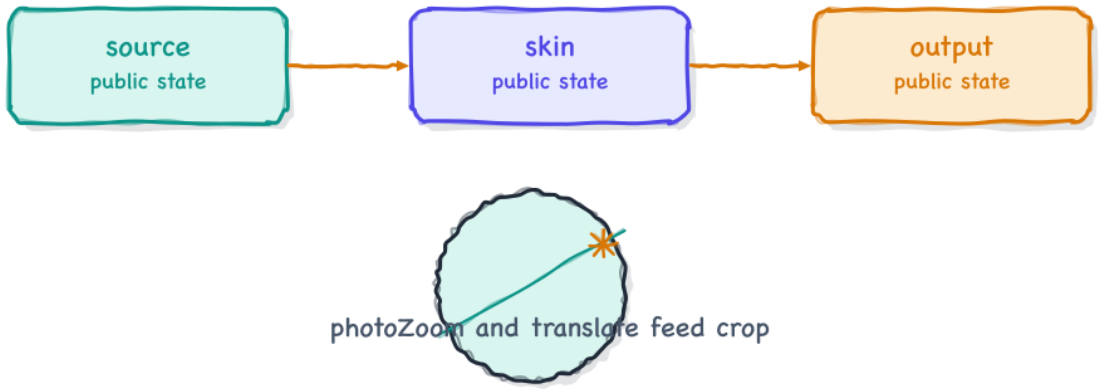
> **Note.** Defaults and property names in this table were checked against the Java source for this batch.

5. Behaviour



Important runtime states of PhotoView.

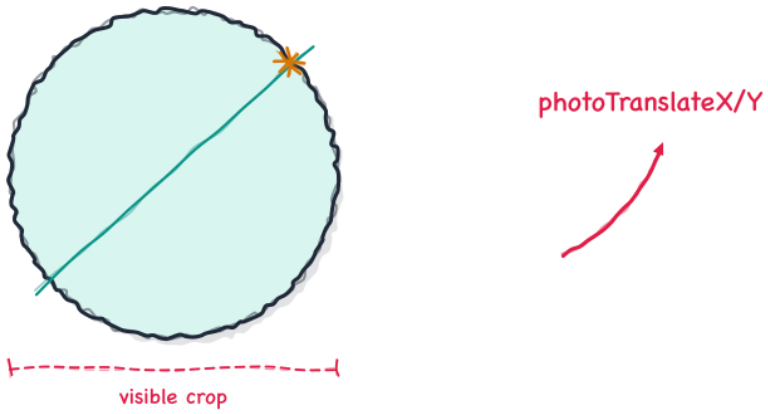
- Setting a new photo resets photoZoom to 1 and translations to 0.
- The skin fits the image so the clip area is filled, then applies user zoom and translation.
- Dragging changes percentage translations, so layout size changes do not lose the composition.
- Scroll wheel adds or subtracts 0.1 zoom; pinch zoom multiplies by the zoom factor.
- CropService waits 200 ms and writes the result to control properties under cropped.image.



How data and geometry flow through PhotoView.

6. Layout and rendering

Rendering centers the image, scales it to cover the clip area, then applies user zoom and percentage-based translations. Cropping reads the visible region after a short delay.

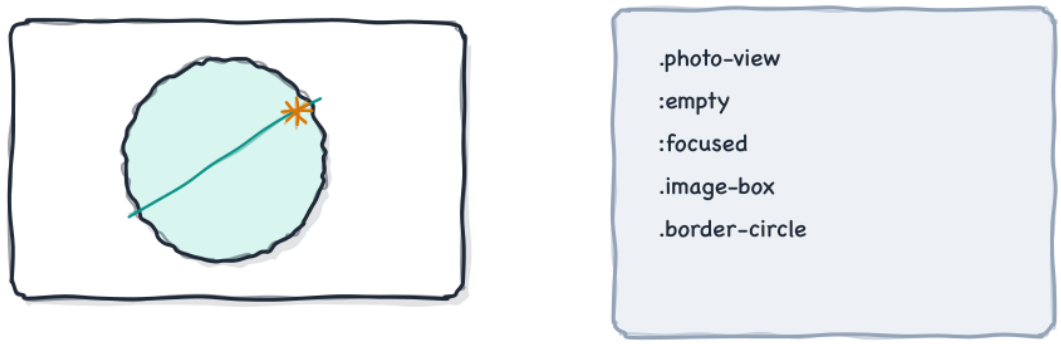


Rendering and sizing rules for PhotoView.

Concern	Rule
Clip	CIRCLE uses a centered Circle; RECTANGLE uses a centered square Rectangle.
Zoom	Slider range is 1 to maxZoom and binds bidirectionally to photoZoom.
Crop	Crop rectangle is computed from image size, fit size, zoom and translations.

7. Styling

The user-agent stylesheet is photo-view.css. The table lists selectors and pseudo classes that exist in source, skin or stylesheet.



Style hooks for PhotoView.

7.1 Style classes and pseudo classes

Selector / pseudo class	Purpose
`.photo-view`	Verified in source, skin or CSS.
`:empty`	Verified in source, skin or CSS.
`:focused`	Verified in source, skin or CSS.
`> .box`	Verified in source, skin or CSS.
`> .box > .image-box`	Verified in source, skin or CSS.
`> .box > .image-box > .placeholder`	Verified in source, skin or CSS.
`.upload-icon`	Verified in source, skin or CSS.
`.border-circle`	Verified in source, skin or CSS.
`.border-rectangle`	Verified in source, skin or CSS.
`.file-drag (CSS only; no source sets it)`	Verified in source, skin or CSS.

7.2 Styleable CSS properties

CSS property	Type	Default
`-fx-editable`	boolean	true
`-fx-clip-shape`	ClipShape	CIRCLE
`-fx-max-zoom`	size	5.0

```
.photo-view {
    /* start with the documented root selector */
}
```

CSS example using documented hooks.

8. Localization

Key	English text
`placeholder.drop-or-click`	DROP IMAGE FILE\nOR CLICK TO ADD
`file-chooser.title.load-image`	Load Image File
`file-chooser.filter.image-files`	Image Files
`accessible.role-description`	photo

9. Accessibility

PhotoView sets AccessibleRole.IMAGE_VIEW with localized role description "photo". It does not bind accessible text to the current image.

10. Recipes

10.1 Programmatic configuration

```
PhotoView photoView = new PhotoView();
photoView.setClipShape(PhotoView.ClipShape.CIRCLE);
photoView.setMaxZoom(4);
photoView.setPhotoSupplier(() -> loadImageFromApplicationDialog());
photoView.croppedImageProperty().addListener((obs, old, img) -> savePreview(img));
```

10.2 Practical checklist

1. Disable createCroppedImage if delayed crops are too expensive.
2. Use a custom photoSupplier when the default FileChooser is not appropriate.
3. Use the public properties listed in the API chapter.
4. Style only through documented selectors and styleable properties.
5. Do not depend on private skin node structure except for documented CSS selectors.

11. Integration notes

The CSS contains .photo-view.file-drag, but the verified source does not add that style class.

Topic	Recommendation
Threading	Keep image loading and expensive rendering off the UI path when the control exposes a background option.
Styling	Scope selectors under the documented root style class.

Topic	Recommendation
Accessibility	Preserve the source-defined accessible role and add app-specific text when the control does not bind it.
State	Prefer public properties over skin node lookup.

12. See also

- Demo application: `com.dlsc.gemsfx.demo.PhotoViewApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.PhotoViewApp`)
- Related GemsFX media controls: `AvatarView`, `PhotoView`, `SVGImageView`, `BeforeAfterView`, `MaskedView`, `ScreensView`.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/