Introduction

Nova.Avalonia.UI is a control library built for Avalonia. It focuses on controls that are themeable, accessible, and ready to drop into desktop, web, and mobile experiences.

Available controls

- **Avatar** and **AvatarGroup**: Identity visuals with initials, images, status badges, and grouping support.
- **Badge**: Notifications, status indicators, and counters with customizable placement and overflow handling.
- **BarcodeGenerator**: Generate QR codes, 1D barcodes, and 2D matrix codes with customizable styling.
- **RatingControl**: Interactive star ratings with customizable shapes and precision levels.
- **Shimmer**: Skeleton loading effect for async data scenarios.

How to use these docs

- Start with Getting Started to install the package and register the styles.
- Browse the individual control pages under **Controls** for API details and usage patterns.
- Refer to the API reference for full class members when you need to extend or customize behaviors.

Getting Started

Follow these steps to install Nova.Avalonia.UI, register its styles, and place your first control in a view.

Prerequisites

- Avalonia 11 or later
- .NET 9 (the library currently targets `net9.0`)

Install the NuGet package

From your application project, install the library:

```
dotnet add package Nova.Avalonia.UI
```

Register the control styles

Add the Nova styles to your `Application.Styles` so the controls pick up their templates. Keep your base theme (for example, `FluentTheme`) before the style include.

```
<Application xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             x:Class="MyApp.App">
    <Application.Styles>
        <FluentTheme />
        <StyleInclude Source="avares://Nova.Avalonia.UI/Themes/Controls.axaml" />
    </Application.Styles>
</Application>
```

Use the controls in XAML

Declare the namespace for the controls and drop them into your layout. This example shows a shimmer placeholder wrapping content and a simple avatar.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:nova="clr-
namespace:Nova.Avalonia.UI.Controls;assembly=Nova.Avalonia.UI">

    <StackPanel Spacing="16">
        <nova:Shimmer IsLoading="True" LoadingText="Loading profile">
            <StackPanel Spacing="8">
                <TextBlock FontSize="18" Text="Profile" />
```

```xml
                <Border Height="120" CornerRadius="12" Background="#1F1F1F" />
            </StackPanel>
        </nova:Shimmer>

        <nova:Avatar DisplayName="Avery Patel" Status="Online" />
    </StackPanel>
</UserControl>
```

Next, explore the individual control pages to see customization options and platform-specific notes.

Avatar

The `Avatar` control presents a person's identity using an image, initials, icon, or custom content. It includes automatic background generation, size presets, and optional presence status indicators.

Create an avatar

Declare an `Avatar` and set `DisplayName`. With the default `DisplayMode` of `Auto`, the control will render initials when no image or icon is provided.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:nova="clr-namespace:Nova.Avalonia.UI.Controls;assembly=Nova.Avalonia.

    <nova:Avatar DisplayName="Alex Martin" />
</UserControl>
```

Use images, icons, or custom content

Choose a display mode explicitly when you need to control the visual output:

- `DisplayMode="Image"` uses the `ImageSource` bitmap.
- `DisplayMode="Icon"` shows the provided `Icon` content.
- `DisplayMode="Content"` renders any custom `Content`.

```
<StackPanel Spacing="12">
    <nova:Avatar DisplayName="Jamie Fox" ImageSource="avares://Assets/jamie.png" Display
    <nova:Avatar DisplayName="Operations" DisplayMode="Icon">
        <nova:Avatar.Icon>
            <PathIcon Data="M18,13 L6,13 6,11 18,11z" />
        </nova:Avatar.Icon>
    </nova:Avatar>
    <nova:Avatar DisplayName="Admin" DisplayMode="Content">
        <nova:Avatar.Content>
            <Ellipse Fill="#F59E0B" Width="18" Height="18" />
        </nova:Avatar.Content>
    </nova:Avatar>
</StackPanel>
```

NOTE

If `DisplayMode` is left as `Auto`, the control picks an image when available, otherwise initials, then icon, then content.

Size, shape, and color

`Avatar` supports preset sizes via `Size` (`ExtraSmall` through `ExtraLarge`) and a `Custom` option controlled by `CustomSize`. Use `Shape` to switch between `Circle`, `Square`, and `Rectangle` corners.

The control can auto-generate a background color from the display name when `AutoGenerateBackground` is `True`, or you can set `BackgroundColor` and `ForegroundColor` directly.

```
<UniformGrid Columns="3" Rows="1" Margin="0,12,0,0">
    <nova:Avatar DisplayName="Kim Lee" Size="Small" />
    <nova:Avatar DisplayName="Drew Parker" Size="Large" Shape="Square" BackgroundColor="
    <nova:Avatar DisplayName="Avery Patel" Size="Custom" CustomSize="80" Shape="Rectangl
</UniformGrid>
```

Show presence status

Attach a status indicator with the `Status` property. You can override the default color per status with `StatusColor`.

```
<StackPanel Orientation="Horizontal" Spacing="10">
    <nova:Avatar DisplayName="Taylor Reed" Status="Online" />
    <nova:Avatar DisplayName="Morgan" Status="Away" />
    <nova:Avatar DisplayName="Jordan" Status="Busy" StatusColor="#C026D3" />
</StackPanel>
```

Tooltips automatically display the `DisplayName` when `ShowTooltip` is `True`, which helps identify users when only initials or icons are visible.

Arrange multiple avatars with AvatarGroup

Use `AvatarGroup` to stack or wrap multiple `Avatar` controls with configurable overlap and overflow handling. Combine `Spacing` and `MaxVisibleAvatars` to control layout, and place any remaining avatars in an overflow badge.

```
<StackPanel Spacing="12">
    <nova:AvatarGroup MaxVisibleAvatars="3" Spacing="-8">
        <nova:Avatar DisplayName="Taylor Reed" Status="Online" />
        <nova:Avatar DisplayName="Morgan Lee" Status="Away" />
        <nova:Avatar DisplayName="Jamie Fox" Status="Busy" />
        <nova:Avatar DisplayName="Avery Patel" />
    </nova:AvatarGroup>

    <nova:AvatarGroup Orientation="Vertical" Spacing="4">
        <nova:Avatar DisplayName="Ops" DisplayMode="Icon">
            <nova:Avatar.Icon>
                <PathIcon Data="M18,13 L6,13 6,11 18,11z" />
            </nova:Avatar.Icon>
        </nova:Avatar>
        <nova:Avatar DisplayName="Engineering" Status="Online" />
    </nova:AvatarGroup>
</StackPanel>
```

`AvatarGroup` also exposes `BorderBrush` and `BorderThickness` to add a ring around the stack when you need a stronger visual boundary against busy backgrounds.

Badge

The `Badge` control displays notifications, status indicators, or short information (like counts) attached to another element or as a standalone indicator. It supports different placements, themes (colors), shapes, and overflow handling for large numbers.

Create a badge

Wrap any content (like a `Button` or `Icon`) with the `Badge` control. Set the `BadgeContent` to define what is displayed inside the badge.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:controls="using:Nova.Avalonia.UI.Controls">

    <controls:Badge BadgeContent="5">
        <Button Content="Notifications" />
    </controls:Badge>
</UserControl>
```

Placements

Control where the badge appears relative to its content using the `BadgePlacement` property. Supported values are:

- `TopLeft`, `Top`, `TopRight` (Default)
- `Left`, `Right`
- `BottomLeft`, `Bottom`, `BottomRight`

```
<StackPanel Spacing="20" Orientation="Horizontal">
    <!-- Default is TopRight -->
    <controls:Badge BadgeContent="1">
        <Border Width="40" Height="40" Background="LightGray" CornerRadius="4"/>
    </controls:Badge>

    <controls:Badge BadgeContent="2" BadgePlacement="BottomRight">
        <Border Width="40" Height="40" Background="LightGray" CornerRadius="4"/>
    </controls:Badge>

    <controls:Badge BadgeContent="3" BadgePlacement="TopLeft">
        <Border Width="40" Height="40" Background="LightGray" CornerRadius="4"/>
    </controls:Badge>
</StackPanel>
```

Themes and Colors

The control includes several built-in themes for common semantic colors. You can apply them using the `Theme` property with a `StaticResource`.

Solid Themes:

- `PrimaryBadge`, `SecondaryBadge`
- `SuccessBadge`, `WarningBadge`, `DangerBadge`, `InfoBadge`
- `LightBadge`, `DarkBadge`

Outline Themes:

- `PrimaryOutlineBadge`, `SuccessOutlineBadge`, etc.

```
<WrapPanel>
    <controls:Badge BadgeContent="Success" Theme="{StaticResource SuccessBadge}" Margin=
    <controls:Badge BadgeContent="Warning" Theme="{StaticResource WarningBadge}" Margin=
    <controls:Badge BadgeContent="Danger" Theme="{StaticResource DangerBadge}" Margin="5
    <controls:Badge BadgeContent="Outline" Theme="{StaticResource PrimaryOutlineBadge}"
</WrapPanel>
```

Shapes and Sizes

You can customize the shape and size using themes or properties.

- **Shapes**: `SquareBadge`, `PillBadge`
- **Sizes**: `SmallBadge`, `LargeBadge`

```
<StackPanel Spacing="10" Orientation="Horizontal">
    <controls:Badge BadgeContent="Pill" Theme="{StaticResource PillBadge}" />
    <controls:Badge BadgeContent="Square" Theme="{StaticResource SquareBadge}" />
    <controls:Badge BadgeContent="Small" Theme="{StaticResource SmallBadge}" />
</StackPanel>
```

Dot Badges

If you only need a simple indicator without text, use `Kind="Dot"`. You can combined this with size themes like `SmallDotBadge` or `LargeDotBadge`.

```
<controls:Badge Kind="Dot" Theme="{StaticResource SuccessBadge}">
    <Button Content="Status" />
</controls:Badge>
```

Overflow Handling (MaxCount)

When displaying numbers, you can limit the maximum value shown using `MaxCount`. If the `BadgeContent` exceeds this limit, it will display the limit followed by a `+` (e.g., "99+").

- The default `MaxCount` is 99. If the content is numeric and exceeds this value, it will be truncated with a `+` suffix logic.

```
<!-- Displays "99+" if content is > 99 -->
<controls:Badge BadgeContent="150" Theme="{StaticResource DangerBadge}">
    <Button Content="Inbox" />
</controls:Badge>

<!-- Custom limit Example (Conceptual) -->
<controls:Badge BadgeContent="10" MaxCount="9" Theme="{StaticResource WarningBadge}">
    <Button Content="Messages" />
</controls:Badge>
```

Standalone Usage

The `Badge` control can also be used without wrapping content, acting as a standalone tag or label.

```
<StackPanel Orientation="Horizontal" Spacing="5">
    <TextBlock Text="Status:" VerticalAlignment="Center"/>
    <controls:Badge BadgeContent="New" Theme="{StaticResource InfoBadge}" />
</StackPanel>
```

Properties reference

Property	**Type**	**Description**
`BadgeContent`	`object`	The content to display inside the badge (text, number, etc.).
`BadgePlacement`	`BadgePlacement`	The position of the badge relative to the content. Default is `TopRight`.
`Kind`	`BadgeKind`	The visual style of the badge content. Values: `Content` (default), `Dot`.
`MaxCount`	`int`	The maximum numeric value to display before showing a `+` suffix. Default is 99.
`IsBadgeVisible`	`bool`	Controls the visibility of the badge itself. Default is `True`.
`BadgeOffset`	`double`	Additional X/Y offset to fine-tune the badge position.

BarcodeGenerator

The `BarcodeGenerator` control generates and renders various barcode symbologies including QR codes, Data Matrix, Code 128, and more. It uses the ZXing library for encoding and supports customizable colors, error correction, and logo overlays.

Create a barcode

Declare a `BarcodeGenerator` and set the `Value` and `Symbology` properties.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:bc="clr-namespace:Nova.Avalonia.UI.BarcodeGenerator;assembly=Nova.Ava

    <bc:BarcodeGenerator Value="https://avaloniaui.net"
                         Symbology="QRCode"
                         Width="200" Height="200" />
</UserControl>
```

Supported symbologies

The control supports multiple barcode formats via the `Symbology` property:

Symbology	**Type**	**Use Case**
`QRCode`	2D	URLs, contact info, general data
`DataMatrix`	2D	Small items, electronics
`Aztec`	2D	Transport tickets, boarding passes
`PDF417`	2D	ID cards, shipping labels
`Code128`	1D	Shipping, logistics
`Code39`	1D	Automotive, defense
`Code93`	1D	Postal services
`EAN13`	1D	Retail products (Europe)
`EAN8`	1D	Small retail products
`UPCA`	1D	Retail products (North America)
`UPCE`	1D	Small packages

Symbology	Type	Use Case
Codabar	1D	Libraries, blood banks
ITF	1D	Logistics, shipping

```
<StackPanel Spacing="12">
    <bc:BarcodeGenerator Value="PRODUCT123" Symbology="Code128" Width="250" Height="80"
    <bc:BarcodeGenerator Value="5901234123457" Symbology="EAN13" Width="200" Height="70"
    <bc:BarcodeGenerator Value="DATA-MATRIX" Symbology="DataMatrix" Width="150" Height="
</StackPanel>
```

Custom colors

Customize the barcode appearance using `BarBrush` and `BackgroundBrush`.

```
<bc:BarcodeGenerator Value="Styled QR"
                     Symbology="QRCode"
                     BarBrush="#1565C0"
                     BackgroundBrush="#E3F2FD"
                     Width="200" Height="200" />
```

TIP

Use `DynamicResource` bindings for theme-aware colors that adapt to light and dark modes.

Display text

Show the encoded value below the barcode with `ShowText`. Customize the text appearance with related properties.

```
<bc:BarcodeGenerator Value="AVALONIA2025"
                     Symbology="Code128"
                     ShowText="True"
                     TextFontSize="14"
                     TextAlignment="Center"
                     TextMargin="0,8,0,0"
                     Width="300" Height="100" />
```

Error correction (QR and Aztec)

For QR codes and Aztec codes, set the `ErrorCorrectionLevel` to control damage recovery:

Level	Recovery	Description
`L`	~7%	Low - smallest code size
`M`	~15%	Medium (default)
`Q`	~25%	Quartile
`H`	~30%	High - best for logos

```
<bc:BarcodeGenerator Value="https://avaloniaui.net"
                     Symbology="QRCode"
                     ErrorCorrectionLevel="H"
                     Width="200" Height="200" />
```

QR code with logo

Overlay a logo in the center of QR or Aztec codes. Use `ErrorCorrectionLevel="H"` for best results.

```
<bc:BarcodeGenerator Value="https://avaloniaui.net"
                     Symbology="QRCode"
                     ErrorCorrectionLevel="H"
                     LogoSizePercent="0.25"
                     Width="200" Height="200">
    <bc:BarcodeGenerator.Logo>
        <Bitmap>avares://MyApp/Assets/logo.png</Bitmap>
    </bc:BarcodeGenerator.Logo>
</bc:BarcodeGenerator>
```

The `LogoSizePercent` property controls the logo size relative to the barcode (0.1 to 0.4).

Quiet zone

The `QuietZone` property sets the margin around the barcode in modules. The default is 2.

```
<bc:BarcodeGenerator Value="QR" Symbology="QRCode" QuietZone="4" />
```

Handle events

Subscribe to `BarcodeGenerated` and `BarcodeError` events for generation feedback.

```
barcode.BarcodeGenerated += (s, e) =>
{
    Console.WriteLine($"Generated {e.Matrix.Width}x{e.Matrix.Height} matrix");
};
```

```
barcode.BarcodeError += (s, e) =>
{
    Console.WriteLine($"Error: {e.Exception.Message}");
};
```

Properties

Property	Type	Default	Description
`Value`	`string`	`""`	Data to encode
`Symbology`	`BarcodeSymbology`	`QRCode`	Barcode format
`BarBrush`	`IBrush`	`Black`	Fill for barcode bars
`BackgroundBrush`	`IBrush`	`White`	Background fill
`QuietZone`	`int`	`2`	Margin in modules
`ShowText`	`bool`	`false`	Show encoded value as text
`TextFontSize`	`double`	`14`	Text font size
`TextForeground`	`IBrush`	`Black`	Text color
`TextMargin`	`Thickness`	`0,8,0,0`	Text margin
`TextAlignment`	`TextAlignment`	`Center`	Text alignment
`ErrorCorrectionLevel`	`QRErrorCorrectionLevel`	`M`	Error recovery level
`Logo`	`IImage`	`null`	Center logo image
`LogoSizePercent`	`double`	`0.25`	Logo size (0.1-0.4)
`ErrorMessage`	`string`	`null`	Last error message

Events

Event	Description
`BarcodeGenerated`	Raised when barcode is successfully generated
`BarcodeError`	Raised when generation fails

RatingControl

The `RatingControl` allows users to view and set ratings using interactive items such as stars, hearts, or custom shapes. It supports multiple precision levels, customizable appearance, and full keyboard and pointer interaction.

Create a rating control

Declare a `RatingControl` and set the `Value` property. By default, the control displays 5 star-shaped items.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:nova="clr-namespace:Nova.Avalonia.UI.Controls;assembly=Nova.Avalonia.

    <nova:RatingControl Value="3" />
</UserControl>
```

Precision modes

Control how values are selected using the `Precision` property:

- `Full` - Only whole numbers (1, 2, 3, etc.)
- `Half` - Half-step increments (1.5, 2.5, etc.)
- `Exact` - Continuous values based on pointer position

```
<StackPanel Spacing="12">
    <nova:RatingControl Value="3" Precision="Full" />
    <nova:RatingControl Value="3.5" Precision="Half" />
    <nova:RatingControl Value="4.2" Precision="Exact" />
</StackPanel>
```

NOTE

Keyboard navigation respects precision: arrow keys increment by 1.0 for `Full`, 0.5 for `Half`, and 0.1 for `Exact`.

Shapes

Choose from built-in shapes or provide a custom geometry:

- `Star` (default)
- `Heart`
- `Circle`

- `Diamond`
- `Custom` - Uses the `CustomGeometry` property

```
<StackPanel Spacing="12">
    <nova:RatingControl Shape="Star" Value="3" />
    <nova:RatingControl Shape="Heart" Value="4" RatedFill="#E91E63" />
    <nova:RatingControl Shape="Circle" Value="2" />
    <nova:RatingControl Shape="Diamond" Value="5" />
    <nova:RatingControl Shape="Custom"
                        CustomGeometry="M 12,2 L 2,22 L 22,22 Z"
                        Value="3" />
</StackPanel>
```

Colors and styling

Customize the appearance using fill and stroke properties for both rated and unrated states. A separate set of preview colors can highlight items during hover.

```
<nova:RatingControl Value="4"
                    ItemSize="40"
                    ItemSpacing="10"
                    RatedFill="#00BCD4"
                    UnratedFill="#B2EBF2"
                    PreviewFill="#4DD0E1"
                    StrokeThickness="1"
                    RatedStroke="#0097A7" />
```

Property	Description
`RatedFill`	Fill brush for rated (active) items
`UnratedFill`	Fill brush for unrated (inactive) items
`RatedStroke`	Stroke brush for rated items
`UnratedStroke`	Stroke brush for unrated items
`PreviewFill`	Fill brush shown when hovering
`PreviewStroke`	Stroke brush shown when hovering
`StrokeThickness`	Thickness of the item stroke

Size and layout

Control the size and spacing of rating items, and choose between horizontal or vertical orientation.

```
<StackPanel Spacing="12">
    <nova:RatingControl Value="3" ItemSize="24" ItemSpacing="4" />
    <nova:RatingControl Value="3" ItemSize="48" ItemSpacing="12" />
    <nova:RatingControl Value="3" Orientation="Vertical" />
</StackPanel>
```

Item count

Set the number of rating items with `ItemCount`. The default is 5, but any positive number is supported.

```
<StackPanel Spacing="12">
    <nova:RatingControl Value="2" ItemCount="3" />
    <nova:RatingControl Value="7" ItemCount="10" ItemSize="20" />
</StackPanel>
```

Read-only mode

Use `IsReadOnly` to display a rating without allowing user interaction. This is useful for showing existing ratings.

```
<nova:RatingControl Value="3.7" IsReadOnly="True" Precision="Exact" />
```

Handle value changes

Subscribe to the `ValueChanged` event to respond when the user changes the rating.

```
<nova:RatingControl x:Name="MyRating"
                    Value="0"
                    ValueChanged="OnRatingValueChanged" />
```

```
private void OnRatingValueChanged(object? sender, RoutedEventArgs e)
{
    if (sender is RatingControl rating)
    {
        Debug.WriteLine($"New rating: {rating.Value}");
    }
}
```

Keyboard support

The control is fully accessible via keyboard:

Key	Action
`Right` / `Up`	Increase value by step

Key	Action
`Left` / `Down`	Decrease value by step
`Home`	Set value to 0
`End`	Set value to maximum

The step size depends on the `Precision` setting (1.0, 0.5, or 0.1).

Properties

Property	Type	Default	Description
`Value`	`double`	`0`	Current rating value
`ItemCount`	`int`	`5`	Number of rating items
`Precision`	`RatingPrecision`	`Full`	Selection precision (Full, Half, Exact)
`IsReadOnly`	`bool`	`false`	Whether the control is read-only
`Shape`	`RatingShape`	`Star`	Shape of rating items
`CustomGeometry`	`Geometry`	`null`	Custom geometry when Shape is Custom
`ItemSize`	`double`	`32`	Size of each rating item
`ItemSpacing`	`double`	`6`	Spacing between items
`Orientation`	`Orientation`	`Horizontal`	Layout orientation
`RatedFill`	`IBrush`	`Gold`	Fill for rated items
`UnratedFill`	`IBrush`	`LightGray`	Fill for unrated items
`RatedStroke`	`IBrush`	`null`	Stroke for rated items
`UnratedStroke`	`IBrush`	`null`	Stroke for unrated items
`PreviewFill`	`IBrush`	`Orange`	Fill during hover preview
`PreviewStroke`	`IBrush`	`null`	Stroke during hover preview
`StrokeThickness`	`double`	`0`	Stroke thickness

Events

Event	Description
`ValueChanged`	Raised when the `Value` property changes

Shimmer

The `Shimmer` control shows a lightweight skeleton while your content is loading. It inspects the visual tree beneath it to draw shapes that match text, images, and buttons, then animates a gradient sweep over the placeholders.

Add a Shimmer placeholder

Wrap the content that loads asynchronously in a `Shimmer`. Toggle `IsLoading` to switch between the placeholder and the real content.

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:nova="clr-namespace:Nova.Avalonia.UI.Controls;assembly=Nova.Avalonia.

    <nova:Shimmer IsLoading="True" LoadingText="Loading profile">
        <StackPanel Spacing="8">
            <TextBlock FontSize="20" Text="Profile" />
            <Border Height="160" CornerRadius="12" Background="#202020" />
            <Button Content="Refresh" Width="120" />
        </StackPanel>
    </nova:Shimmer>
</UserControl>
```

When `IsLoading` is `True`, Shimmer disables hit testing on the child content and announces the loading state to screen readers.

Customize the effect

Use the following properties to align the effect with your theme:

- `HighlightBrush` sets the moving gradient. Bind it to a `DynamicResource` for theme switching.
- `ShimmerOpacity` adjusts the overlay opacity. The default is `0.5`.
- `ShimmerAngle` sets the gradient angle in degrees.
- `LoadingText` defines the automation name announced while loading.

```
<nova:Shimmer IsLoading="True"
              HighlightBrush="{DynamicResource AccentGradient}"
              ShimmerOpacity="0.35"
              ShimmerAngle="12"
              LoadingText="Loading dashboard cards" />
```

Show loaded content

Set `IsLoading` to `False` when your data is ready. The child content becomes visible and interactive, and the automation name is cleared.

```
  // ViewModel
  public bool IsBusy { get; set; }
```

```
<nova:Shimmer IsLoading="{Binding IsBusy}">
    <ItemsControl Items="{Binding Orders}" />
</nova:Shimmer>
```