

# EB GUIDE tutorial

Creating a list

Copyright © 2017 Elektrobit Automotive GmbH

Legal notice

Confidential and proprietary information.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

ProOSEK®, tresos®, and street director® are registered trademarks of Elektrobit Automotive GmbH.

All brand names, trademarks and registered trademarks are property of their rightful owners and are used only for description.

# 1. Tutorial: Creating a list with dynamic content

**NOTE****Default window layout**

All instructions and screenshots of this user manual use the default window layout. If you want to follow the instructions, we recommend to reset the EB GUIDE Studio window to default layout by selecting **Layout > Reset to default layout**.

Instantiators allow creating lists dynamically during run-time. Based on a datapool item of a list type, an instantiator displays all list elements in a pre-defined layout. If the content of the datapool item is modified, so is the appearance of the instantiator.

The following instructions guide you through the process of creating a list with dynamic content. Each list element consists of a labeled rectangle.

Approximate duration: 15 minutes.

**Adding a datapool item**

The following instructions guide you through the process of adding a datapool item of type `String list`. The datapool item provides a value for every list element of the instantiator. If the content of the datapool item is modified, so is the appearance of the instantiator.

Prerequisite:

- The **Main** state machine contains an initial state and a view state.
- The initial state has a transition to the view state.

Step 1

To display content in your list, add a datapool item of type `String list`.

In the **Datapool** component, click **+**.

A menu expands.

Step 2

In the menu, click **String list**.

A new datapool item of type `String list` is added.

Step 3

Rename the datapool item to `MyStringList`.

Step 4

Click the  button.

An editor opens.

Step 4.1

Click **Add**.

A new entry is added to the table.

Step 4.2

Enter `One` in the `Value` text box.

Step 4.3

Add the values `Two`, `Three`, `Four`, and `Five` to the `MyStringList` datapool item.

Step 4.4

Click **Accept**.

You added a datapool item of type `String list`. The datapool item contains five entries.

The content of the list is displayed next to the `Value` property.



### Adding widgets

Prerequisite:

- You completed the previous instruction.

Step 1

To add widgets to your view, double-click the view state in the content area.

The view is displayed in the content area.

Step 2

In the **Navigation** component, expand the view state and the view.

Step 3

Drag an instantiator from the **Toolbox** into the view. Rename the instantiator to `MyInstantiator`.

Step 4

Drag a rectangle from the **Toolbox** into the instantiator. Rename the rectangle to `MyRectangle`.

Step 5

Drag a label from the **Toolbox** into the rectangle. Rename the label to `MyLabel`.

The widget hierarchy now looks as follows.

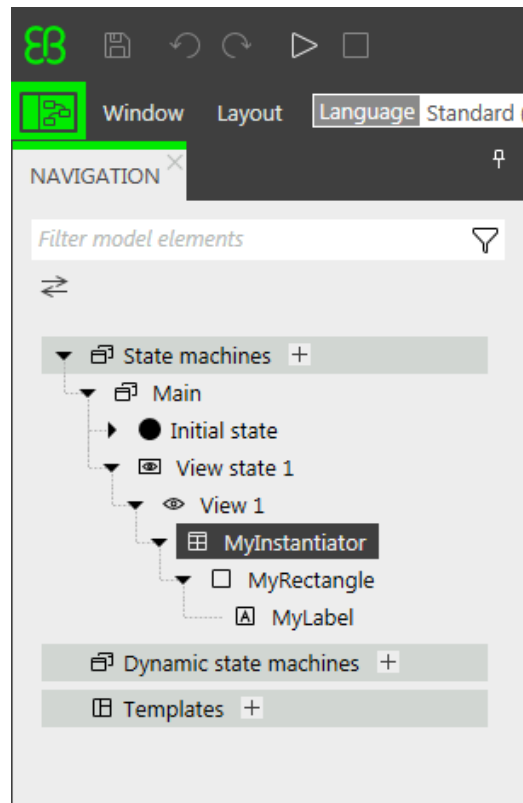


Figure 1. Widget hierarchy with an instantiator



## Configuring the instantiator

Prerequisite:

- You completed the previous instruction.

### Step 1

To change the properties of `MyInstantiator`, select the instantiator and go to the **Properties** component.

### Step 2

Enter 300 in the `width` text box, and in the `height` text box.

### Step 3

Enter 250 in the `x` text box.

### Step 4

Enter 150 in the `y` text box.

### Step 5

To calculate the length of the list dynamically, add a conditional script.

In the **User-defined properties** category, click **+**.

A menu expands.

[Step 5.1](#)

In the menu, click **Conditional script**.

[Step 5.2](#)

Rename the property to `calculateNumItems`.

[Step 5.3](#)

Next to the `calculateNumItems` property click **Edit**.

A script editor opens in the content area.

[Step 5.4](#)

Add the `MyStringList` datapool item to the **Trigger** list.

[Step 5.5](#)

Enter the following **On trigger** script:

```
function(v:arg0::bool)
{
  v:this.numItems = length dp:MyStringList;
  false
}
```

You added a script which automatically changes the number of list entries depending on the content of `MyStringList`.

[Step 6](#)

To arrange all labels within the instantiator, add a layout to it.

In the **Widget feature properties** category, click **Add/Remove**.

The **Widget features** dialog is displayed.

[Step 6.1](#)

Under **Available widget features**, expand the **Layout** category, and select the **Box layout** widget feature to arrange the labels side by side.

The related widget feature properties are added to the instantiator and displayed in the **Properties** component.

[Step 6.2](#)

Click **Accept**.

[Step 6.3](#)

Enter 5 in the `gap` text box to set a spacing of 5 px between each list element.

[Step 6.4](#)

Select **vertical (=1)** from the **layoutDirection** drop-down list box to arrange the labels among each other.

You configured the instantiator which defines the visual appearance of the list and adapts the number of list items dynamically.



## Configuring list element texts

Prerequisite:

- You completed the previous instruction.

### Step 1

To change the appearance of the label, select the label and go to the **Properties** component.

### Step 2

Enter 0 in the `x` and `y` text box.

### Step 3

Add a link from the label's `width` property to the rectangle's `width` property.

#### Step 3.1

Next to the `width` property, click the  button.

A menu expands.

#### Step 3.2

In the menu, click **Add link to widget property**.


A dialog opens.

#### Step 3.3

In the dialog, go to the rectangle, and select its `width` property.

#### Step 3.4


Click **Accept**.

The dialog closes. The  button is displayed next to the `width` property.

### Step 4

Add a link from the label's `height` property to the rectangle's `height` property.

### Step 5

Next to the `horizontalAlign` property, click .

You changed the appearance of the label. The label is now centered in the rectangle.



## Configuring list elements

Prerequisite:

- You completed the previous instruction.

### Step 1

To change the appearance of the rectangle, select the rectangle and go to the **Properties** component.

### Step 2

To make sure that the list elements use the available width, add a link from the rectangle's `width` property to the instantiator's `width` property.

### Step 3

Enter 50 in the `height` text box.

### Step 4

To define a unique position for each line of your list, add the **Line index** widget feature.

#### Step 4.1

In the **Widget feature properties** category, click **Add/Remove**.

The **Widget features** dialog is displayed.

#### Step 4.2

Under **Available widget features**, expand the **List management** category, and select the **Line index** widget feature.

The `lineIndex` property is added to the rectangle's properties.

### Step 5

To fill the labels of the list with the content of `MyStringList`, add a conditional script.

#### Step 5.1

Next to the **User-defined properties** category, click **+**.

A menu expands.

#### Step 5.2

In the menu, click **Conditional script**.

#### Step 5.3

Rename the property to `setText`.

#### Step 5.4

Next to the `setText` property, click **Edit**.

A script editor opens in the content area.

#### Step 5.5

Add the `lineIndex` property of the rectangle and the `MyStringList` datapool item to the **Trigger** list.

#### Step 5.6

Enter the following **On Trigger** script:

```
function (v:arg0::bool)
{
  v:this->MyLabel.text=dp:MyStringList[v:this.lineIndex];
  false
}
```

You changed the appearance of the rectangle. With the `setText` property, the labels of `MyStringList` are filled automatically with the content of `MyStringList`.



## Testing the EB GUIDE model

Prerequisite:

- You completed the previous instruction.

### Step 1

To start the simulation, click  in the command area.

Result:

Since `MyStringList` contains five datapool items, five rectangles that are labeled from one to five are displayed in vertical arrangement.

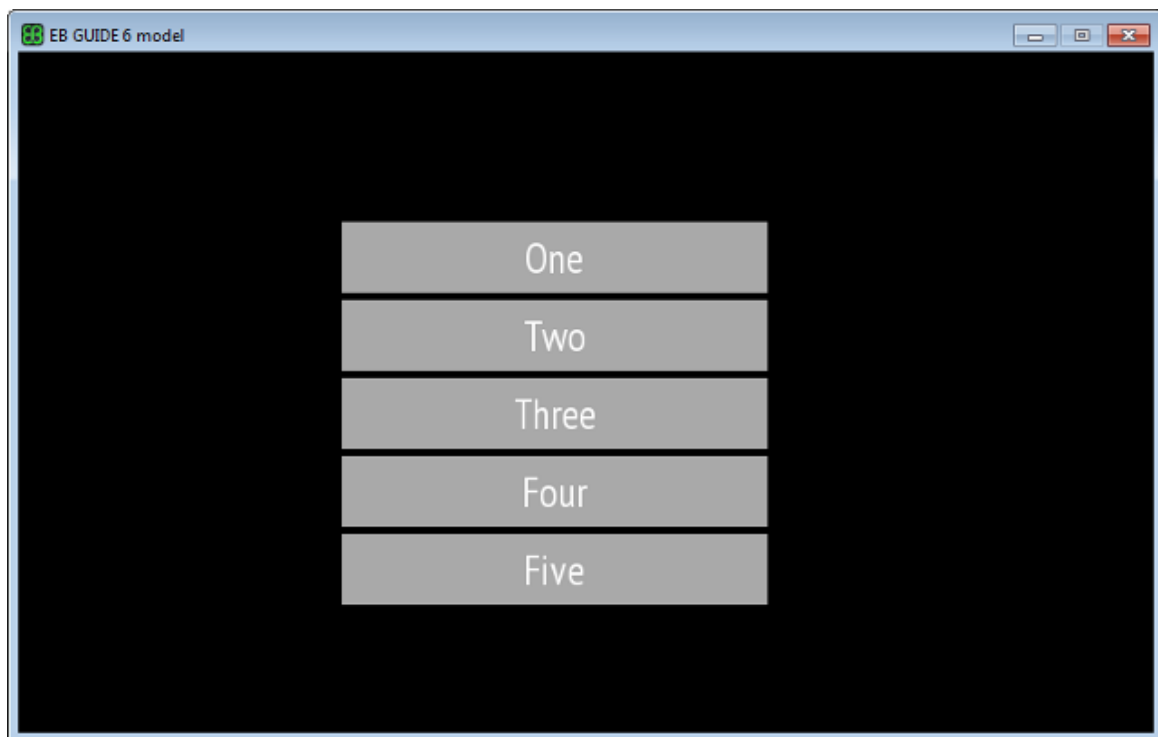


Figure 2. List created with an instantiator