

## Ejercicio 3 – Build and Editable Grid – Orlando Sánchez Acuña

### Part 1 - Create the Getting Started Objects

# Create Data Type

Create from scratch  
 Duplicate existing data type  
 Create from database table or view  
 Import XSD

**Namespace \***

Formatted as a URI, for example 'urn:com:appian:types:COB' for a client onboarding application

**Name \***

**Description**

Report

ADV\_Items     **appian**

[Learn more about data design](#)

**Fields (6)**

Name	Type	Length	Array	Key			
itemID	Number (Integer)		<input checked="" type="checkbox"/>		↑	↓	×
description	Text	255	<input type="checkbox"/>		↑	↓	×
category	Text	255	<input type="checkbox"/>		↑	↓	×
qty	Number (Integer)		<input type="checkbox"/>		↑	↓	×
unitPrice	Number (Decimal)		<input type="checkbox"/>		↑	↓	×
amount	Text	255	<input type="checkbox"/>		↑	↓	×

Data Entities

ADV\_Address ADV\_Address

Items ADV\_Items

[Add Entity](#)

Schema Management

Automatically update database schema

Applan will make necessary updates to the database schema when fields are added to a referenced data type or when this data store is imported.

[Download DDL Script](#)

Entity mappings verified

## Create Constant

Name \*

ADV\_PURCHASE\_REQUEST\_CATEGORIES

Description

Constant to hold the categories for the Purchase Request grid

Type \*

Text

Array (multiple values)

Values (4) ?

Office Supplies  
Hardware  
Software  
Miscellaneous



CANCEL

CREATE

Part 2 - Create the Layout and Add Variables

# Create Interface

Create from scratch  Duplicate existing interface

Name \*

ADV\_EditableGrid

Description

Interface for editable grid

Save In \*

ADV Interfaces

Create New Rule Folder

Browse Folder

CANCEL

CRF

## Add a Local Variable

The screenshot shows the Appian interface definition editor for 'ADV\_EditableGrid' in 'EXPRESSION MODE'. The left pane shows the rule definition with a local variable 'newGridRow' added to the 'a1FormLayout' component. The right pane shows a preview of the 'Purchase Request' form with 'CANCEL' and 'SUBMIT' buttons. The 'RULE INPUTS' and 'LOCAL VARIABLES' panels on the right show the configuration for the 'cancel' input and the 'newGridRow' local variable.

Name	Value
cancel	null

Name	Value
newGridRow	[itemID=, descripti...

## Add a Rule Input

# New Rule Input

Name \*

items

Type \*

ADV\_Items

Array (multiple values)

CANCEL

CREATE

## Part 3 - Configure the Editable Grid

### Add the Editable Grid Component

Array of column headers created with `a!gridLayoutHeaderCell()`.

```
6 | label: "Category"  
7 | ),  
8 | a!gridLayoutHeaderCell(  
9 |   label: "Qty"  
10 | ),  
11 | a!gridLayoutHeaderCell(  
12 |   label: "Unit Price"  
13 | ),  
14 | a!gridLayoutHeaderCell(  
15 |   label: "Amount"  
16 | ),  
17 | a!gridLayoutHeaderCell(  
18 |   label: ""  
19 | )  
20 | }
```

**a!gridLayoutHeaderCell** label, helpTooltip, align, showWhen  
Defines a column header for use in an editable grid (a!gridLayout).  
**label** (Text): Text to display as the column header.

### Adjust the Widths of Columns

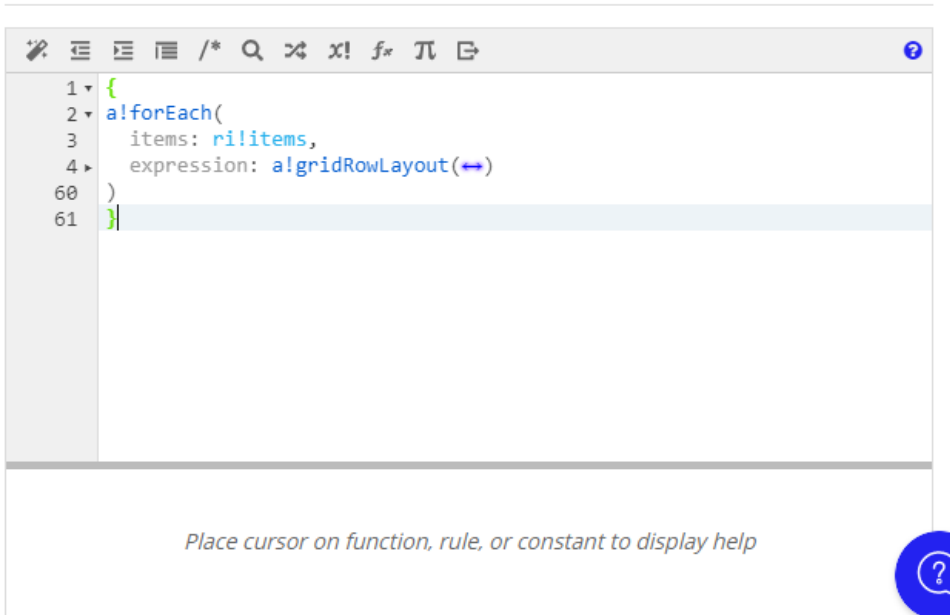
The screenshot shows the Appian ADV Editable Grid interface. The main window displays a 'Purchase Request' form with an 'Items' table. The table has columns for 'Description', 'Category', 'Qty', 'Unit Price', and 'Amount'. Below the table, there are 'CANCEL' and 'SUBMIT' buttons. The right-hand sidebar shows the 'COMPONENT CONFIGURATION' for the 'Editable Grid' component. Under 'Column Configurations', there are several 'Column Configuration' items. The 'ADD ITEM' button is highlighted with a blue circle.

### Configure the Components to Use in a Row

The screenshot shows the Appian ADV Editable Grid interface. The main window displays a 'Purchase Request' form with an 'Items' table. The table has columns for 'Description', 'Category', 'Qty', 'Unit Price', and 'Amount'. Below the table, there are 'CANCEL' and 'SUBMIT' buttons. The right-hand sidebar shows the 'COMPONENT CONFIGURATION' for the 'Row' component. Under 'Contents', there are several component options: 'Text (Text)', 'Dropdown (Dropdown)', 'Integer (Integer)', 'Decimal (Decimal)', 'Text (Text)', and 'Rich Text'. The 'ADD COMPONENT' button is highlighted with a blue circle.

Use the `forEach` function to Display a Row for Each Item

# Item 1



The screenshot shows a code editor window with a toolbar at the top containing icons for undo, redo, search, and other editing functions. The code is as follows:

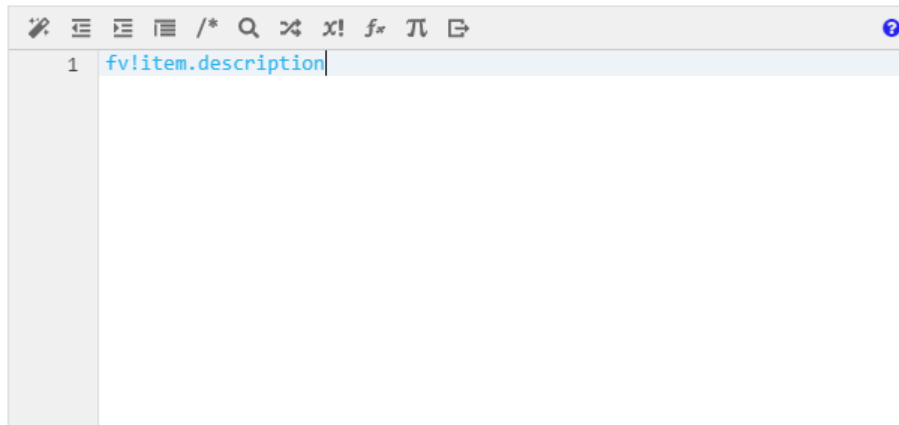
```
1 {  
2  a!forEach(  
3    items: ri!items,  
4    expression: a!gridRowLayout(↔)  
60 )  
61 }
```

A light blue horizontal bar highlights the code between lines 60 and 61. Below the code editor, a tooltip message reads: "Place cursor on function, rule, or constant to display help". A blue circular help icon with a white question mark is positioned at the bottom right of the editor area.

Configure the Display and Save Into Fields for Each Row Component

## Display Value (Text)

Text to display in the text field.



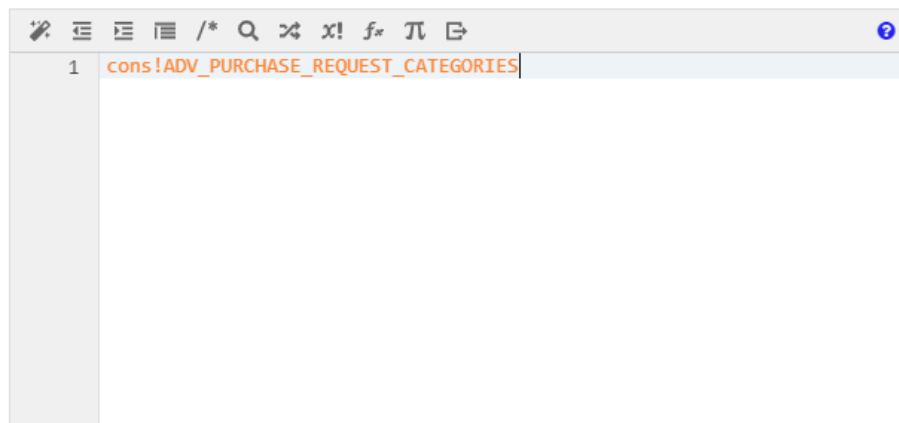
```
1 fv!item.description
```

*Place cursor on function, rule, or constant to display help*



## Choice Labels (List of Text String)

Array of options for the user to select.



```
1 cons!ADV_PURCHASE_REQUEST_CATEGORIES
```

**cons!ADV\_PURCHASE\_REQUEST\_CATEGORIES**

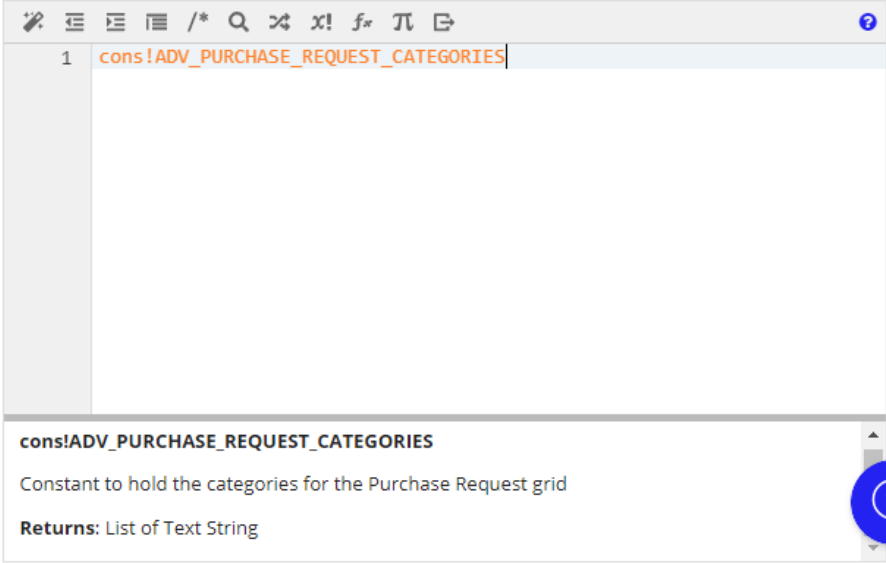
Constant to hold the categories for the Purchase Request grid

**Returns:** List of Text String



## Choice Values (List of Variant)

Array of values associated with the available choices.



The screenshot shows a code editor with a toolbar at the top containing icons for undo, redo, search, and other functions. The main text area contains a single line of code: `1 cons!ADV_PURCHASE_REQUEST_CATEGORIES`. Below the editor, a tooltip is displayed with the following information:

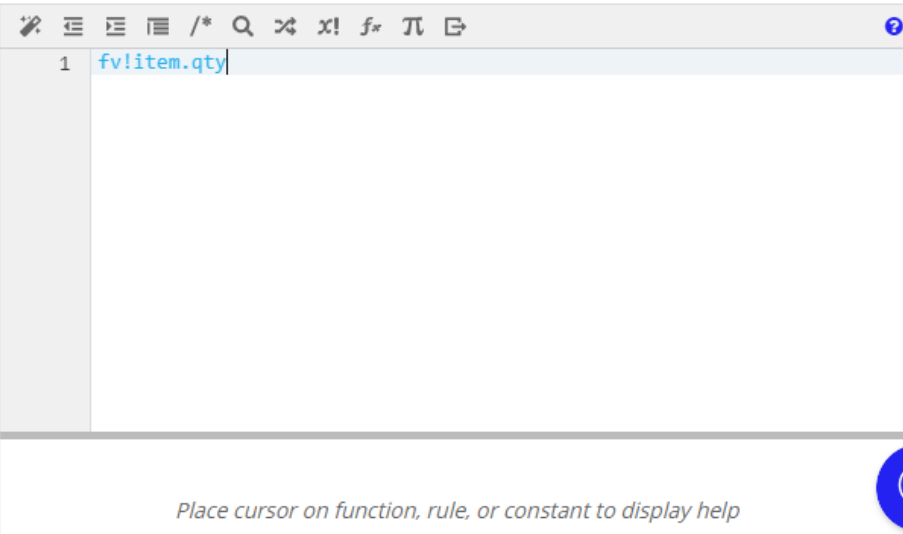
- cons!ADV\_PURCHASE\_REQUEST\_CATEGORIES**
- Constant to hold the categories for the Purchase Request grid
- Returns:** List of Text String

A blue circular help icon with a question mark is visible on the right side of the tooltip.

[Clear expression and reset value](#)

## Display Value (Number (Integer))

Number to display in the field.



The screenshot shows a code editor with a toolbar at the top containing icons for undo, redo, search, and other functions. The main text area contains a single line of code: `1 fv!item.qty`. Below the editor, a tooltip is displayed with the following information:

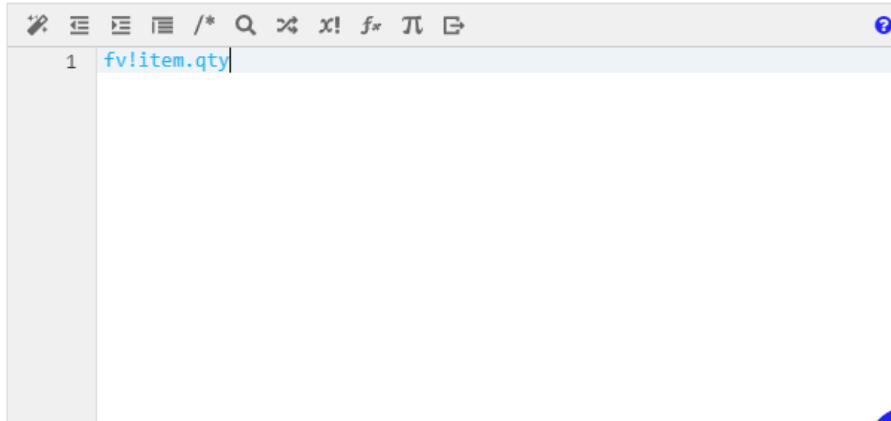
- fv!item.qty**
- Function to return the quantity of the selected item
- Returns:** Integer

A blue circular help icon with a question mark is visible on the right side of the tooltip.

*Place cursor on function, rule, or constant to display help*

## Save Input To (List of Save)

One or more variables that are updated with the integer when the user changes it. Use `alsave()` to save a modified or alternative value to a variable.



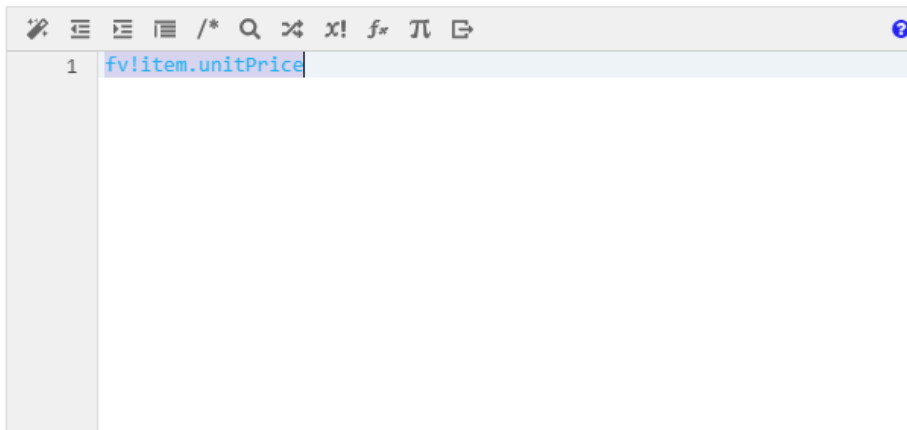
A screenshot of a code editor interface. The top toolbar contains icons for undo, redo, search, and other functions. The main text area shows a single line of code: `1 fv!item.qty`. The text is highlighted in light blue.

*Place cursor on function, rule, or constant to display help*



## Display Value (Number (Decimal))

Number to display in the field.



A screenshot of a code editor interface. The top toolbar contains icons for undo, redo, search, and other functions. The main text area shows a single line of code: `1 fv!item.unitPrice`. The text is highlighted in light blue.

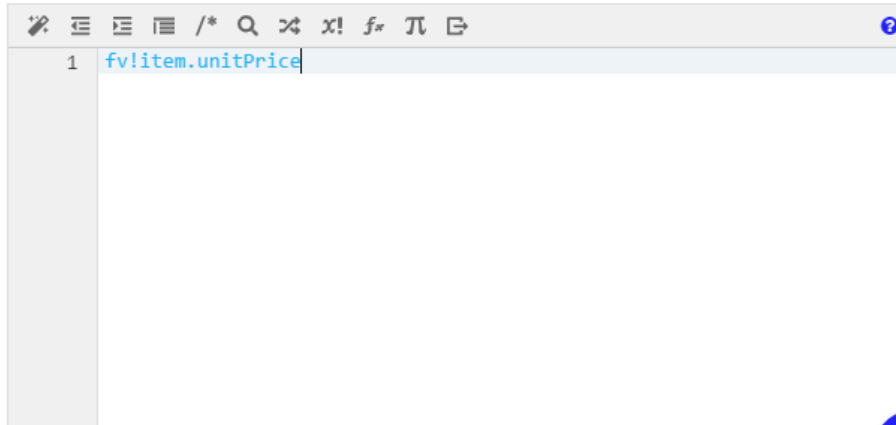
*Place cursor on function, rule, or constant to display help*





## Save Input To (List of Save)

One or more variables that are updated with the number when the user changes it. Use `alsave()` to save a modified or alternative value to a variable.



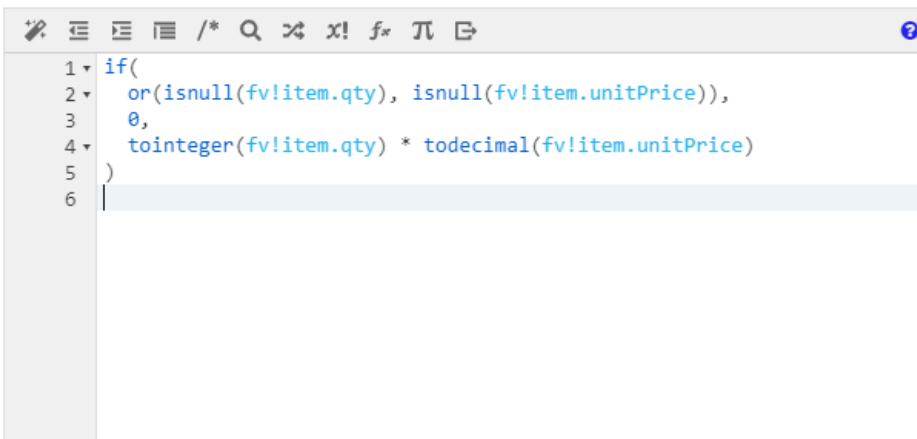
```
1 fv!item.unitPrice
```

*Place cursor on function, rule, or constant to display help*



## Display Value (Text)

Text to display in the text field.



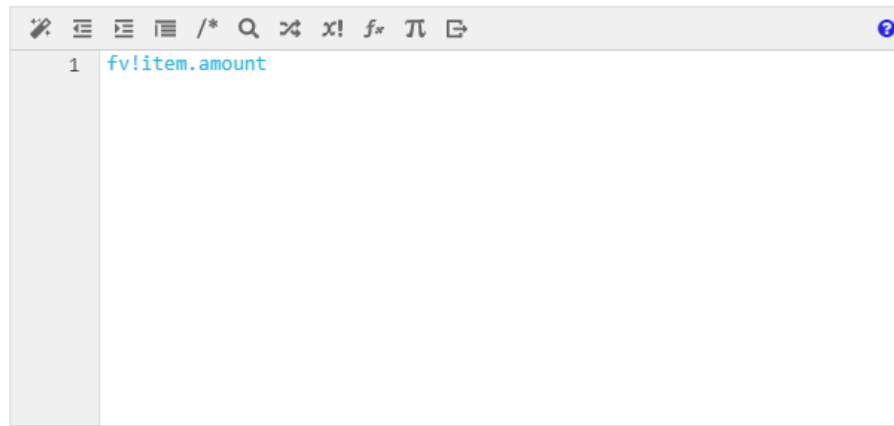
```
1 if(  
2   or(isnull(fv!item.qty), isnull(fv!item.unitPrice)),  
3   0,  
4   tointeger(fv!item.qty) * todecimal(fv!item.unitPrice)  
5 )  
6 |
```

*Place cursor on function, rule, or constant to display help*



## Save Input To (List of Save)

One or more variables that are updated with the text when the user changes it. Use `alsave()` to save a modified or alternative value to a variable.



```
1 fv!item.amount
```

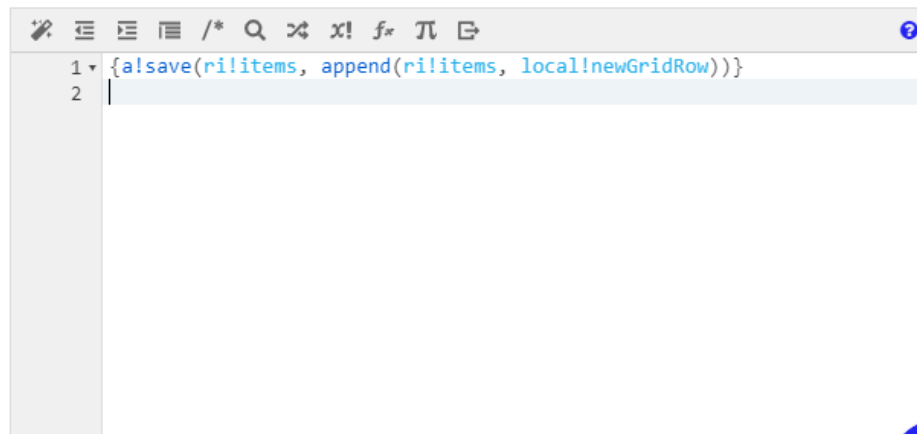
*Place cursor on function, rule, or constant to display help*



### Part 4 - Build the Add New Item Link

## Save Value To (List of Save)

One or more variables that are updated with the link's value when the user clicks it. Use `alsave()` to save a modified or alternative value to a variable.



```
1 {alsave(ri!items, append(ri!items, local!newGridRow))}  
2
```

*Place cursor on function, rule, or constant to display help*



### Part 5 - Build the In-Line Delete Button

# Save Value To (List of Save)

One or more variables that are updated with the link's value when the user clicks it. Use `!save()` to save a modified or alternative value to a variable.

The screenshot displays the Appian IDE interface. At the top, a rule editor shows the following expression: `1 {!save(rilitems, remove(rilitems, fv!index))}`. Below this, the main workspace shows a form titled "Purchase Request" in "ADV EditableGrid" design mode. The form contains a table with columns: Description, Category, Qty, Unit Price, and Amount. The Amount column has a value of 0 and a red 'x' icon. Below the table are "CANCEL" and "SUBMIT" buttons. On the right side, the "RULE INPUTS" and "COMPONENT CONFIGURATION" panels are visible. The "COMPONENT CONFIGURATION" panel shows the "Dynamic Link" property set to "Dynamic Link" and the "Value" property set to "Select a variable". The "Save Value To" property is set to "List of Save". The "Visibility" property is set to "Always show".