



Build an Application: Step-by-Step #4

Exercise to Accompany
Expressions: Transform Your Data

The Appian Step-by-Step series consists of 12 exercises that accompany the courses in the Appian Developer learning path. Exercises build upon each other. Complete exercises in order and keep the app and all objects until you are done with the project.

- 1 Welcome to the Appian Developer Learning Path
- 2 Create an Application
- 3 Manage Users and Groups
- 4 Expressions**
- 5 Design Record Types
- 6 Sites
- 7 Query Your Data
- 8 Interfaces 101
- 9 Process Modeling 101: Part 1
- 10 Process Modeling 101: Part 2
- 11 Reports
- 12 Task Report

Exercise 4: Expressions	3
Create Constants	3
Create an Expression Rule	4
Troubleshooting Resources	7

Notice of Rights

This document was created by Appian Corporation, 7950 Jones Branch Dr, Tysons, Virginia 22102. Copyright 2025 by Appian Corporation. All rights reserved. Information in this document is subject to change. No part of this document may be reproduced or transmitted in any form by any means, without prior written permission of Appian Corporation. For more information on obtaining permission for reprints or excerpts, contact Appian Training at academyonline@appian.com.

This exercise was developed for **Appian 25.4**. If Appian Community Edition is on a later Appian version, functionality might be different. Go to academy.appian.com to download the latest exercise.

Exercise 4: Expressions

In this exercise, you will create two types of rule objects: six constants and one expression rule.

Create Constants

A constant is a literal value that can be called from any expression, across objects in your application. In this section, you will create constants that point to the groups and folders you just created.

First, you will create constants that point to groups.

1. From within the **W#SA Constants** folder, click **NEW > Constant**.
2. In the **Create Constant** dialog, configure the following properties:
 - **Name:** Enter `W#SA_ADMINISTRATORS_GROUP_POINTER`.
 - **Description:** Enter A pointer to the W#SA application administrators group.
 - **Type:** Select **Group**.
 - **Value:** Enter and select **W#SA Administrators**.
 - **Environment Specific:** Keep this configuration unselected.
 - **Save In:** Ensure **W#SA Constants** is selected.
3. Click **CREATE**.
4. Follow steps 1–3 to create constants for the remaining four groups:
 - **W#SA Users**
 - **W#SA Supervisors**
 - **W#SA Mechanics**
 - **W#SA Registrars**

Next, you will create a constant that points to the W#SA Documents folder.

1. Click **NEW > Constant**.
2. In the **Create Constant** dialog, configure the following properties:
 - **Name:** Enter `W#SA_DOCUMENTS_FOLDER_POINTER`.
 - **Description:** Enter A pointer to the W#SA application documents folder.
 - **Type:** Select **Folder**.

- **Value:** Enter and select **W#SA Documents**.
- **Environment Specific:** Keep this configuration unselected.
- **Save In:** Ensure **W#SA Rules & Constants** is selected.

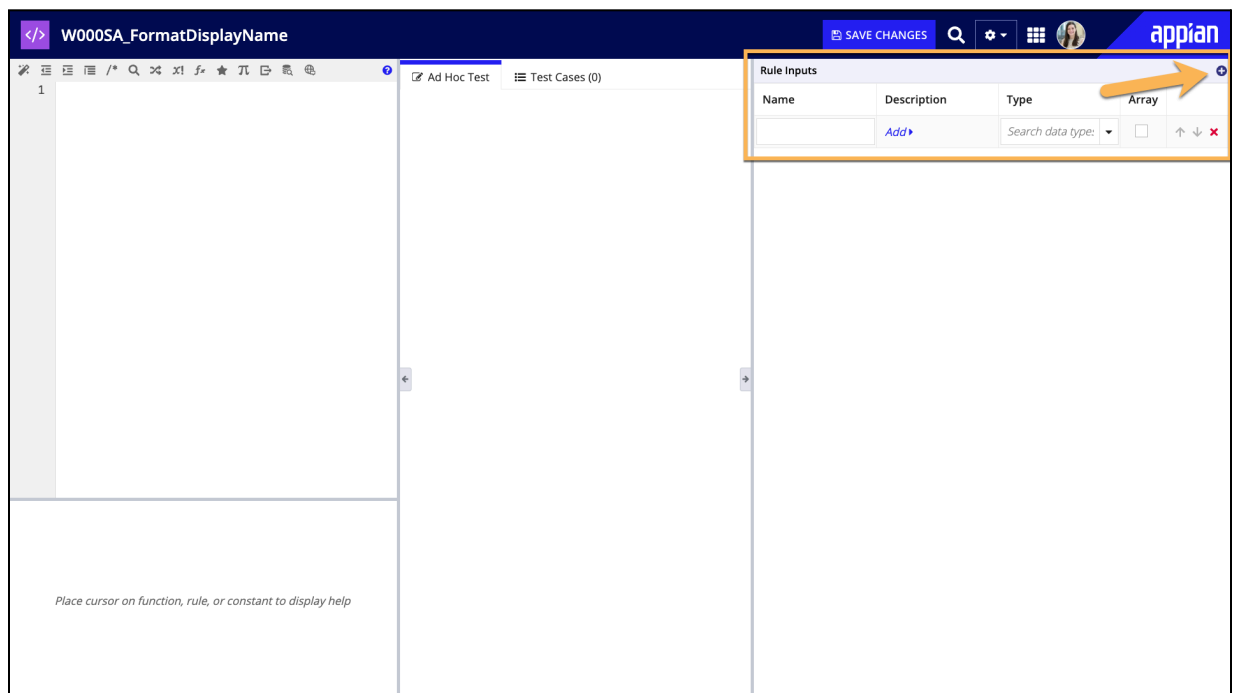
3. Click **CREATE**.

Create an Expression Rule

An expression rule is a reusable object containing a statement that evaluates to return a value. In this section, you will create an expression that formats a user's name. You will use this expression rule in a later exercise.

Follow the steps below to create this expression rule.

1. From within the **W#SA Expressions** folder, click **NEW > Expression Rule**.
2. In the **Create Expression Rule** dialog, configure the following properties:
 - **Name:** Enter `W#SA_FormatDisplayName`.
 - **Description:** Enter `Formats a user's name for the W#SA application.`
3. Click **CREATE**.
4. In the **Rule Inputs** pane, click the **New Rule Input** icon.




5. Configure the following properties:

- **Name:** Enter `user`.
- **Description:** Click **Add**.
 - **Description:** Enter `Username of logged in user`.
 - Click **ADD**.
- **Type:** Select **Text**.

6. Enter the following expression into the Expression Editor:

```
a!match(  
  value: ri!user,  
  whenTrue: isnull(fv!value),  
  then: "",  
  whenTrue: isusernetaken(fv!value),  
  then: proper(  
    user(fv!value, "firstName") & " " & user(fv!value,  
    "lastName")  
  ),  
  default: proper(joinarray(split(fv!value, "."), " "))  
)
```

In this expression, `a!match` evaluates the input (a username) against multiple conditions and returns a value (the formatted name) based on a match. Function variables, which use the domain prefix `fv!`, are special variables used within certain functions and interfaces. Here, you are using `fv!value` to reference the value parameter (`ri!user`).

7. In the toolbar, click **Format**  to automatically format your expression.

8. Click **SAVE**.

9. Test this expression:

- In **Test Inputs**, in the **Expression** column, enter the function `loggedInUser()`. This function returns the current user who is logged in.
- Click **TEST RULE** to see the formatted name.

Ad Hoc Test

Test Cases (0)

Test Inputs

Rule Input Name	Expression	Value
user (Text)	1 <code>loggedInUser()</code>	"julia.maier"

[Save as Test Case](#)

TEST RULE

Test Output

Time 11 ms [View Performance](#)

Type Text

Value

☒ Formatted
 ☐ Raw
 ☐ Expression

"Julia Maier" (Text)



Tip: About a!match

a!match evaluates a value against multiple conditions and returns a value based on a match. If no match is found, the default is returned.

- First, this expression checks whether the user's username is null. If the user's username is null, an empty string is returned.
- If the user's username is not null, then the expression checks whether the user is an active user. If the user is active, the first and last name associated with the username is returned.
- If the username is not null and the user is not active, the default is returned.
- The process for formatting users and non-users is different. The last line in the expression reformats the username input as First and Last Name for a non-user.

Visit the [Appian Community YouTube channel](#) and [Appian Documentation](#) to learn more about functions.

Troubleshooting Resources

Stuck on a step, or need help troubleshooting? Appian provides several support resources that you can use as you build:

1. **Acme Auto Solution Application** - The Acme Auto Solution Application (AS) is the solution to the exercises you are following in the Step-by-Steps. You can use the AS application as a reference tool. Review it to see how specific objects are configured, or test the application to see how the features work from a business user's perspective. This application is preloaded into your workspace. If you do not see it in the list of applications in your workspace, you can deploy it from the App Catalog. Refer to **Build an Application: Step-by-Step #1** for more information on how to use the App Catalog.
2. [Community Discussions for New Users](#) - Check out the **New to Appian** thread in Community. Join our community of experts to ask questions and find answers from past discussions.
3. [Appian Documentation](#) - Appian's product documentation will provide you with an overview of key Appian features, newest release information, additional tutorials, and helpful patterns and recipes to implement in your app.