

# Expressions

**Expression Rule :** They perform a wide range of functions, including retrieving, formatting and transforming the data.

## Examples :

The screenshot displays the JUnit Expressions IDE interface. On the left, a code editor shows a rule definition for retrieving user information. The rule uses the `user()` function with `username` and `property` arguments. Below the code, the `user()` function is documented, stating it returns information for a user and lists supported properties like `displayName`, `email`, `firstName`, `lastName`, `middleName`, `status`, and `address1`.

The central pane shows the **Test Inputs** section with a table:

Rule Input Name	Expression	Value
ticketID (Number (Integer))	1	3

Below the table are buttons for [Save as Test Case](#) and **TEST RULE**.

The **Test Output** section shows the execution results:

- Time:** 4 ms ([View Performance](#))
- Type:** Text
- Value:** ☒ Formatted ☐ Raw ☐ Expression
- Output:** "Ticket #3 created by User" (Text)


On the right, the **RULE INPUTS** panel shows the `ticketID` input with a numeric value of 3.













The screenshot displays the JUnit Expressions IDE interface. The code editor on the left shows a rule definition using an `if()` statement. The condition is `sum(1,1)=2`, and the actions are `"It equals 2"` and `"It does not equal 2"`.

The central pane shows the **Test Inputs** section, which is currently empty, and a **TEST RULE** button.

The **Test Output** section shows the execution results:

- Time:** 1 ms ([View Performance](#))
- Type:** Text
- Value:** ☒ Formatted ☐ Raw ☐ Expression
- Output:** "It equals 2" (Text)

 AX\_L2Ex1



```
1 user(  
2   loggedInUser(),  
3   "firstName"  
4 )
```

Ad Hoc Test

Test Cases (0)


Test Inputs













Test Output

Time 4 ms [View Performance](#) Type Text

Value ☒ Formatted ☐ Raw ☐ Expression

"Admin" (Text)

 AX\_L2Ex1



```
1 user(loggedInUser(), "firstName")  
2 & " created on " &  
3 user(loggedInUser(), "created")
```

Ad Hoc Test

Test Cases (0)

Test Inputs

Test Output

Time 6 ms [View Performance](#) Type Text

Value ☒ Formatted ☐ Raw ☐ Expression

"Admin created on 10/6/2021 7:41 PM GMT+00:00" (Text)



Arrays (6:00)

The screenshot shows the AA\_Sandbox interface. On the left, a code editor displays a JSON object: `{make: "Honda", type: "sedan", year: 2005, mileage: 46597}`. The `.make` method is applied to this object. On the right, the 'Test Inputs' section is empty. The 'Test Output' section shows a 'Time' of 1 ms and a 'Value' of `"Honda"` (Text). The 'Type' is 'Any Type (Text)'. The 'Value' is displayed as 'Formatted'.

Arrays (6:00)

The screenshot shows the AA\_Sandbox interface. On the left, a code editor displays a JSON object: `{make: "Honda", type: "sedan", year: 2005, mileage: 46597}`. The `index` method is applied to the object, with the argument `"type"` specified. On the right, the 'Test Inputs' section is empty. The 'Test Output' section shows a 'Time' of 1 ms and a 'Value' of `"sedan"` (Text). The 'Type' is 'Any Type (Text)'. The 'Value' is displayed as 'Formatted'.

Arrays (6:00)

AA\_Sandbox

SAVE CHANGES

appian

1 a!map(  
2 make: "honda",  
3 type: "sedan",  
4 year: 2005,  
5 mileage: 46597  
6 ).make  
7

Ad Hoc Test

Test Cases (0)

Test Inputs

TEST RULE

Test Output

Time < 1 ms (View Performance) Type Text

Value ☒ Formatted ☐ Raw ☐ Expression

"honda" (Text)