Oracle Fusion Cloud Applications

Groovy Scripting Examples

Oracle Fusion Cloud Applications
Groovy Scripting Examples

F77851-03

Author: Kristin Penaskovic

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Use help icons to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 About This Guide

Introduction

Groovy is a standard, dynamic scripting language for the Java platform supported by Application Composer. Use the Groovy scripting language to enhance your applications.

You write Groovy scripts using Application Composer's expression builder, which appears in many places as you modify existing objects or create new custom ones.

This document provides a collection of simple examples of using Groovy in all of the different supported contexts in your application.

Related Guides

To fully understand all the scripting features available to you in Application Composer, you should also review the following guides.

Guide	Description
Groovy Scripting Reference	Explains the basics of how to use the Groovy scripting language to enhance your application functionality.
Configuring Applications Using Application Composer	Describes how and where you can use Groovy scripting in Application Composer.

Related Topics

- Oracle Help Center
- Groovy Scripting Reference
- Configuring Applications Using Application Composer

2 Before You Begin

Best Practices for Using Groovy Scripts

Before you begin, note the following best practices and recommendations for optimal performance of your Groovy scripts:

- Always use a sandbox to test your changes before publishing to mainline.
- When publishing your sandbox, check for any warnings and don't ignore them. The warnings may cause errors once you publish the sandbox.
- Whenever you use `setAttribute()` to set attribute values, add an `if` statement to check the terminal condition. Otherwise, you may receive a "Post threshold limit reached. Some entities yet to be posted." error.

  For example:

  ```
  if(AttributeA != valueA)
   setAttribute('AttributeA', valueA)
  ```

- Use `setAttributeValues()` to set composite key or related attribute values. For example:

  ```
  setAttributeValues(['ReasonWonLostCode','ReasonWonLostCodeSetId'], [xxx,0])
  setAttributeValues(['DecisionLevelCode','DecisionLevelCodeSetId'], [xxx,0])
  ```

- Create rows using NameValuePairs. For example, if you want to create an Opportunity partner row from the Opportunity object, you can assign an ID as shown below:

  ```
  def nvp = new oracle.jbo.NameValuePairs()
  nvp.setAttribute("OptyId", OpId);
  nvp.setAttribute("PartOrgPartyId", partOrgId);
  nvp.setAttribute("RevnId", primaryRevnId);

  def OptyPartner = getAttribute('RevenuePartnerPrimary')
  def vOP = OptyPartner.createAndInitRow(nvp)
  ```

- Use the **ID** field instead of the **Name** field in Groovy logic.

  For example, if SalesStage is the **Name** field, you should use it for display purpose only. It returns the descriptive name in the user session language. Use SalesStageId, instead.

  Another example is, suppose you're using a dynamic choice list Building_c, use isAttributeChanged('Building_Id_c') instead of its name isAttributeChanged('Building_c').
- Use Before* triggers instead of After* triggers for better performance, unless you need to overwrite the standard logic.

3 Account

Add Classification Code

This example shows how to add a classification code to access the standard object Industry. The associated (related) collection name of Industry is CodeAssignment.

To access the Industry object:

1. Create the custom choice list, Industry_xxx_c, in Account.
2. Create a trigger in Account as shown below:

```
def ClassCategory_t = '1972 SIC'
def ClassCode_t = nvl(Industry_xxx_c, '0')
def CreatedByModule_t = 'ZCM'
def OwnerTableName_t ='HZ_PARTIES'
def OwnerTableId_t = PartyId
def oldIndustryCollection = CodeAssignment
def vo = newView('CodeAssignment')
if(ClassCode_t)
{
def newIndustry = vo.createRow()
newIndustry.setAttribute('ClassCategory', ClassCategory_t)
newIndustry.setAttribute('ClassCode', ClassCode_t)
newIndustry.setAttribute('CreatedByModule', CreatedByModule_t)
newIndustry.setAttribute('OwnerTableName', OwnerTableName_t)
newIndustry.setAttribute('OwnerTableId', OwnerTableId_t)
vo.insertRow(newIndustry)
}
```

Make Account Field Values Default

You can hide the required fields in Account from the Create Competitor page by creating an After Create trigger on the Account object. The below example is for COMPETITOR party usage.

Create a trigger in Account as shown below:

```
def party = getAttribute('OrganizationParty')
def partyUsage = party.getAttribute('PartyUsageCode')

if (partyUsage == 'COMPETITOR') {
 // setAttribute() for the required Account fields
}
```

To change the displayed fields in the upper part of the Create Competitor page, do the following:

1. Go to the Account object and click the **Pages** node.
2. Click the **Desktop Pages** tab.
3. In the Details Page section, click **Edit Organization Details Region**.
4. Toggle fields between the **Available Fields** and **Selected Fields** panels to show or hide them.

Create Original System References

The following example illustrates how you can use Groovy to create Original System References for an object.

1. Before you create the trigger, go to Setup and Maintenance and verify that the following source systems are defined and have the required settings:
    - Manage Trading Community Source Systems (enabled with **Enable for Trading Community Members**)
    - Manage Source System Entities
2. Create a trigger in Account, as shown below:

```
def newRef = OriginalSystemReference.createRow();
newRef.setAttribute('OrigSystem', 'XXXXX'); //provide the desired Source System Code
newRef.setAttribute('OrigSystemReference', 'XXXXX'); //provide a literal or variable with the Source
 System Reference
newRef.setAttribute('CreatedByModule', 'FUSE'); //populate the attribute CreatedByModule with a value
 from Setup and Maintenance > Manage Standard Lookups > HZ_CREATED_BY_MODULES
OriginalSystemReference.insertRow(newRef);
```

4 Activity

Calculate Weekdays between Two Dates

This example shows how you can use Groovy to calculate the number of weekdays between two dates.

```
// Global Function getWorkingDaysBetweenTwoDates
// Returns: Long
// Parameters: ActivityStartDate (Date), ActivityEndDate (Date)

Calendar startCal = Calendar.getInstance();
startCal.setTime(ActivityStartDate);
Calendar endCal = Calendar.getInstance();
endCal.setTime(ActivityEndDate);
int workDays = 0;

//Return 0 if start and end are the same
if(startCal.getTimeInMillis() == endCal.getTimeInMillis()){
 return 0;
}
if(startCal.getTimeInMillis() > endCal.getTimeInMillis()){
 startCal.setTime(ActivityEndDate);
 endCal.setTime(ActivityStartDate);
}
//excluding end date
while(startCal.getTimeInMillis() < endCal.getTimeInMillis()){
 startCal.add(Calendar.DAY_OF_MONTH, 1);
 if (startCal.get(Calendar.DAY_OF_WEEK) != Calendar.SATURDAY && startCal.get(Calendar.DAY_OF_WEEK) !=
 Calendar.SUNDAY) {
 ++workDays;
 }
}
return workDays;
```

Calculate Difference between Dates

This example shows how you can use Groovy to trigger an email alert exactly after 10 days of Employee Probation End Date, by calculating the difference between the dates.

1. Create a field in Activity with the name ProbationEndDate.
2. Use the following code to trigger an email alert exactly after 10 days of Employee Probation End Date by calculating the difference:

```
//Difference between dates
//Business Scenario: To trigger an email alert exactly after 10 days of Employee Probation End Date.
if(TRUNC(SYSDATE) - Trunc(${ProbationEndDate}) == 10)
{
 // Business logic
}

//To reference the application server's current date and time in any groovy expression, use:
adf.currentDate
```

```
adf.currentDateTime

//To reference the database's current date and time in any groovy expression, use:
adf.currentDBDate
adf.currentDBDateTime

//Returns: the current date, with no time
println(today())

//Returns the current date and time
println(now())

//1. Difference between two dates
def today = new Date()
def yesterday = today - 1

assert 1 == today.minus(yesterday)
assert 1 == today - yesterday

//2. Date.parse() to convert String to Date.
def date = new Date().parse('yyyy/MM/dd', '2019/12/09')

//3. We can use [] or getAt() to get date fields.
assert 2019 == date[Calendar.YEAR]
assert 11 == date[Calendar.MONTH]
assert 9 == date.getAt(Calendar.DATE)

//Returns a number representing the month
def currentMonth=month(new Date())
def nextMonth = currentMonth + 1

// We can use the + and - operators to add or subtract days.
def date = new Date().parse('yyyy/MM/dd', '2019/07/22')
def dateNext = date.clone() +1 // prints --> Tue Jul 23 00:00:00 UTC 2019
def datePrevious = date.clone() -1 // prints --> Sun Jul 21 00:00:00 UTC 2019
def nextDay = date + 1 // Or date.plus(1)
def previousDay = date - 1 // Or date.minus(1)

// ++ operator to move one day ahead.
dateNext++ // Or dateNext.next()
assert dateNext == nextDay

// -- operator to move one day back.
datePrevious-- // Or datePrevious.previous()
assert datePrevious == previousDay

def otherDate = new Date().parse('yyyy/MM/dd', '2019/07/25')
// Dates can be used in ranges.
println((otherDate..<date).size()) // prints -->3

// Date.format() uses java.text.SimpleDateFormat.
assert '9 December, 2019' == date.format("d MMMM, yyyy")
assert '12/9/19' == date.getDateString()
```

5 Opportunity

Show Related Lead number in Opportunity

The following example illustrates how you can get an Opportunity page to display the associated lead number:

1. Create a new sandbox in Application Composer.

   See "Create and Activate Sandboxes" in the Configuring and Extending Applications guide.
2. In Application Composer, expand Opportunity and do the following:
   a. Go to the Fields page and create a custom text field named **Lead Number**.

      See "Text Fields" in the Configuring Applications Using Application Composer guide.
   b. On the **Triggers** tab of the Scripts page, add a new **BeforeUpdate** trigger and enter the following in the **Trigger Definition** script section:

```
if (LeadNumber_c == null || LeadNumber_c == ''){
 def leads = OpportunityLead

 if (leads.hasNext()){
 def a = leads.first()
 LeadNumber_c = a.LeadNumber
 }
}
```

      See "Server Scripts" in the Configuring Applications Using Application Composer guide for more information.

Related Topics

- Create and Activate Sandboxes
- Text Fields
- Server Scripts

Filter Using Multiple Criteria on the Same Attribute

This example script on OpportunityVO shows how you can filter using multiple criteria on the same attribute. For instance, you can search with multiple conditions like "where value > 20 and value < 30" or "Opportunity Name starting with A".

You can use the below sample script to retrieve opportunity records with names that start with an A or AMMM.

You can't use `ensureCriteriaItem()` to set multiple conditions on the same field. If you use `ensureCriteriaItem()` to set multiple conditions, only the last condition applies. So, instead of using two view criteria, you can create two `ViewCriteriaRow` elements on the same `ViewCriteria`.

```
println("Start ensureCriteriaItem test")
def vo = newView('OpportunityVO');
def vc = vo.appendViewCriteria("Name like 'A%' and Name = 'AMMM'")
vo.executeQuery();
while(vo.hasNext())
{
println("inside while")
def row = vo.next()
println(row.Name)
}
println("End ensureCriteriaItem test")
```

Access Parent-Child Objects

In this sample script, you're creating a child record of the Interaction object from the parent Opportunity object. Place this script in the 'Before Insert in Database' trigger event of the Opportunity object.

While creating parent-child records with composite associations, you must ensure that the parent record IDs are present in the child records at the time of creation since the child has no existence outside the context of the parent. You must add the parent record to the correct rowset. Only then, the child record will be able to identify the parent record to which it belongs.

```
def opptyUpdatedName = Name
def opptyOwnerId = OwnerResourcePartyId
def opptyAccountId = TargetPartyId
def voInteraction = newView('InteractionVO')
def createInt = voInteraction.createRow()
voInteraction.insertRow(createInt) // <-- This is shifted to a place before accessing the children.
def currentDateTime = now()
def interactionName = 'OptyNameModifed' + currentDateTime
createInt.setAttribute('InteractionDescription', interactionName)
createInt.setAttribute('InteractionStartDate', currentDateTime)
createInt.setAttribute('CustomerId',opptyAccountId)
def voIntAssociation = createInt.InteractionAssociation
def createIntAssc = voIntAssociation.createRow()
createIntAssc.setAttribute('AssociatedObjectUid',OptyId )
createIntAssc.setAttribute('AssociatedObjectCode','OPPORTUNITY')
```

Get Count of Records Matching a Search Condition

The sample script shows how to retrieve the count of Opportunity records with the name 'Pinnacle Systems'. You can place the script in the Before Insert section in the Database trigger event of the Opportunity object.

```
def v = newView('OpportunityVO') // VO can be changed according to the Object used for counting the records
def vc = newViewCriteria(v)
def vcr = vc.createRow()
def vciName = vcr.ensureCriteriaItem("Name") // Field name of the View Object used for the Criteria
vciName.setOperator('=')
vciName.setValue("Pinnacle Systems")
vc.insertRow(vcr)
v.appendViewCriteria(vc)
long count = 0;
v.executeQuery()
count = v.getEstimatedRowCount()
```

```
println('Opportunities found: ' + count)
```

Note: The getEstimatedRowCount() function returns an estimate. You can't use it in places where accurate values are expected all the time.

Create Opportunity

This example shows how to create an opportunity from the Product object.

1. In Application Composer, expand the Product standard object and click the **Actions and Links** node.
2. In the Script region of the Create Action or Link page, click the New icon to build your script and enter the following:

```
def vo = newView('OpportunityVO');

//create the opportunity
def newRecord = vo.createRow();
newRecord.setAttribute('Name',<ItemNumber attribute for Opportunity object> + " - " + now());

//create revenue lines
def revenueLine = newRecord.ChildRevenue;
def createRevenue = revenueLine.createRow();
createRevenue.setAttribute('ProductType','Item');
createRevenue.setAttribute('InventoryItemId',<InventoryItemId attribute for Opportunity object>);

createRevenue.setAttribute('InventoryOrgId',<InvOrgId attribute for Opportunity object>);

createRevenue.setAttribute('Quantity',10);
createRevenue.setAttribute('UnitPrice',10);

revenueLine.insertRow(createRevenue);
vo.insertRow(newRecord);
showmessage()
```

The `showmessage()` method is optional. It's a custom function that generates a warning message telling your users that a new opportunity is created and that they need to click **Save** or **Save and Close**. So, it can be removed from the code.

To display the message in a dialog box, do the following:

1. Create a Text field in the Product object. You don't have to display it in the layout.
2. Create an object function as shown below:
   - Name: `showmessage`
   - Type: Void
   - Code:

     ```
     def ran = new Random()
     setAttribute('Showmessage_c',ran.nextInt())
     ```
3. Create a Field Validation.

   Enter an error message: "A new opportunity was created, please click Save or Save and Close.

   Code:

```
adf.error.warn(null)
```

6 Common Groovy Use Cases

Format Date Values Using a Formatter

This example shows how you can use Groovy to format a date value.

```
def dateVal = currentDate
def locale = adf.context.getLocale();

//To format a date value dateVal to display the day of the week, month name, the day, and the year, you can
 do:
Date dv1 = dateVal as Date
def fmt1 = new Formatter(locale)
def ret1 = (dv1 != null) ? fmt1.format('%tA, %tB %te, %tY',dv1,dv1,dv1,dv1) : null
```

Format Datetime Values Using a Formatter

This example shows how you can use Groovy to format a datetime value.

```
def datetimeVal = currentDate
def locale = adf.context.getLocale();

//To format a datetime value datetimeVal to display only the hours and minutes in 24-hour format, you can
 do:
Date dv = datetimeVal as Date
def fmt = new Formatter(locale)
def ret = (dv != null) ? fmt.format('%tH:%tM', dv, dv) : null
```

Format Numbers Using a Formatter

This example shows how you can use Groovy to format numbers.

```
//To format a number numberVal as a floating point value with two (2) decimal places and thousands separator
 you can do:
def locale = adf.context.getLocale();
def numberVal = 12345;


Double dv = numberVal as Double
def fmt = new Formatter(locale)
def ret = (dv != null) ? fmt.format('%,.2f', dv) : null

//To format a number numberVal as a floating point value with three (3) decimal places and no thousands
 separator you can do:
Double dv1 = numberVal as Double
def fmt1 = new Formatter(locale)
def ret1 = (dv1 != null) ? fmt1.format('%.3f', dv1) : null
```

```
//To format a number value with no decimal places to have a zero-padded width of 8, you can do:
Long lv = numberVal as Long
def fmt2 = new Formatter(locale)
def ret2 = (lv != null) ? fmt2.format('%08d', lv) : null
```

Use Groovy Maps and Lists with Web Services

When passing and receiving structured data from a web service, a Groovy map represents an object and its properties.

For example, an Employee object with properties `Empno`, `Ename`, `Sal`, and `Hiredate` is represented by a Map object with four key/value pairs, where the names of the properties are the keys. The following example shows how you can create and use a Groovy map.

```
//You can create an empty Map using the syntax:
def newEmp =[:]

//Then, you can add properties to the map using the explicit put() method like this
newEmp.put("Empno",1234)
newEmp.put("Ename","Sean")
newEmp.put("Sal",9876)
newEmp.put("Hiredate",date(2013,8,11))

//Alternatively, you can assign and/or update map key/value pairs using a simpler direct assignment notation
 like this:
newEmp.Empno = 1234
newEmp.Ename = "Sean"
newEmp.Sal = 9876
newEmp.Hiredate = date(2013,8,11)

//You can also create a new map and assign some or all of its properties at once using the constructor
 syntax:
def newEmp1 = [Empno : 1234,
 Ename : "Sean",
 Sal : 9876,
 Hiredate : date(2013,8,11)]

//To create a collection of objects you use the Groovy List object.
//You can create one object at a time and then create an empty list, and call the list's add() method to add
 both objects to the list:
def dependent1 = [Name : "Dave",
 BirthYear : 1996]
def dependent2 = [Name : "Jenna",
 BirthYear : 1999]
def listOfDependents = []
listOfDependents.add(dependent1)
listOfDependents.add(dependent2)

//To save a few steps, the last three lines above can be done in a single line by constructing a new list
//with the two desired elements in one line like this:
def listOfDependents1 = [dependent1, dependent2]

//Note that you can also construct a new employee with nested dependents all in one statement by further
 nesting the constructor syntax:
def newEmp2 = [Empno : 1234,
 Ename : "Sean",
 Sal : 9876,
 Hiredate : date(2013,8,11),
 Dependents : [
 [Name : "Dave",
 BirthYear : 1996],
```

```
[Name : "Jenna",
BirthYear : 1999]]
]
```