Guidewire training materials contain Guidewire proprietary information that is subject to confidentiality and non-disclosure agreements. You agree to use the information in this manual solely for the purpose of training to implement Guidewire software solutions. You also agree not to disclose the information in this manual to third parties or copy this manual without prior written consent from Guidewire. Guidewire training may be given only by Guidewire employees or certified Guidewire partners under the appropriate agreement with Guidewire.

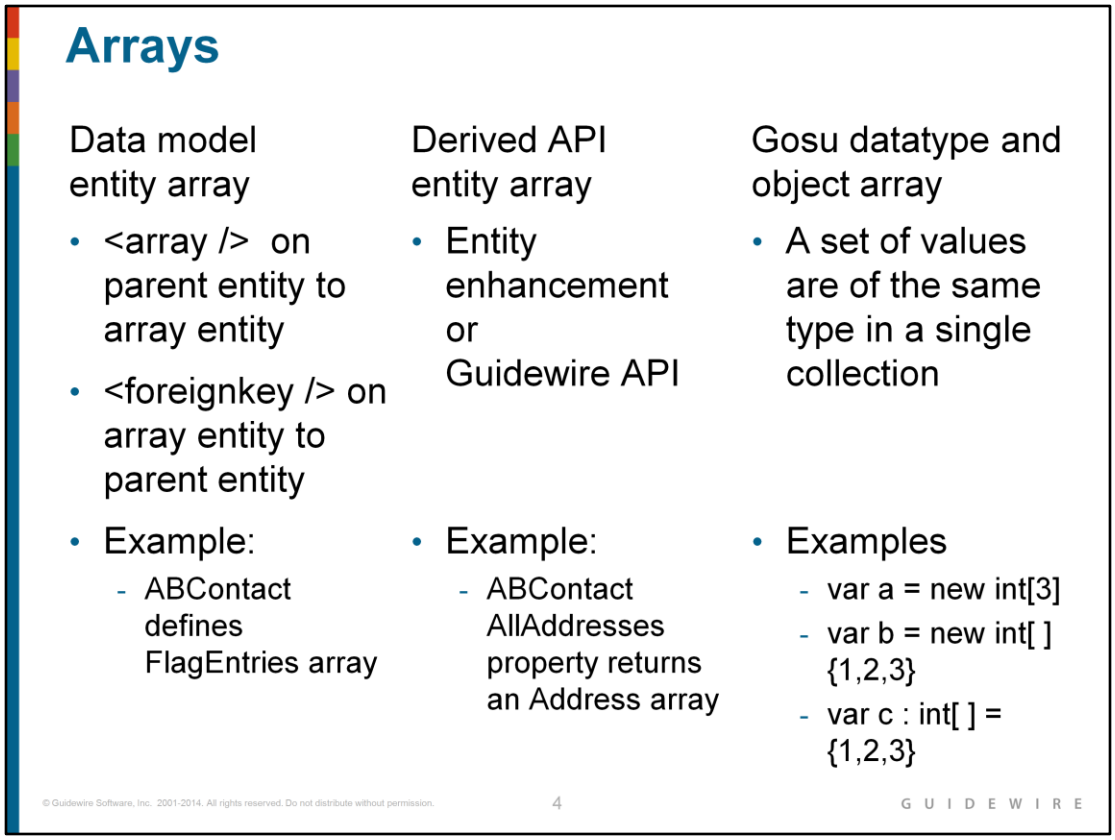
Lesson objectives

- By the end of this lesson, you should be able to:
  - Describe what is an array
  - Determine the number of elements in an array
  - Loop through each object in an array
  - Add object to (and remove object from) an array
  - Determine if any objects with a given criterion exists in an array
  - Retrieve objects in an array that match a given criterion

This lesson uses the notes section for additional explanation and information.
To view the notes in PowerPoint, select View → Normal or View → Notes Page.
When printing notes, select Note Pages and Print hidden slides.

Lesson outline

- Arrays
- Array method expressions



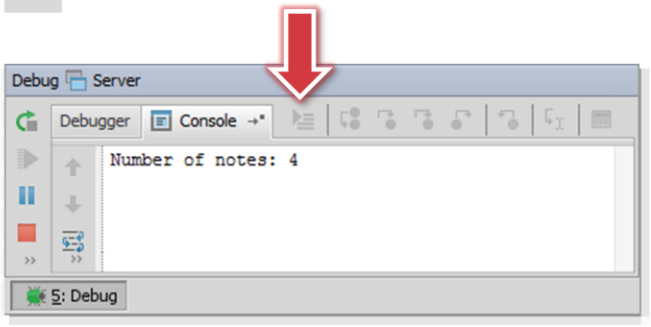
Configuration of Guidewire applications often focus on working with data model entity arrays. An array defines a set of additional entities of the same type to associate with the main entity. A related data model array is the associative array. An associative array provides a mapping between a set of keys and the values that the keys represent. A common example of this type of mapping is a telephone book, in which a name maps to a telephone number. Another common example is a dictionary, which maps terms to their definitions.

An array is a collection of data values, with each element of the array associated with a number or index.

In typical Gosu code, simply use angle brackets after the type name, such as String[] to represent an array of String objects. Use a zero-based index number to access an array member. If you create an array, you must explicitly define the size of the array or implicitly define the size by simultaneously defining the array elements. To access the elements of an array, you use an index expression.

Array length

```
1  uses ta.QueryUtil
2  var out : String = "Notes"
3  var anABContact = QueryUtil.findContact("5")
4  var notes = anABContact.ContactNotes
5  print(out + notes.length)
```

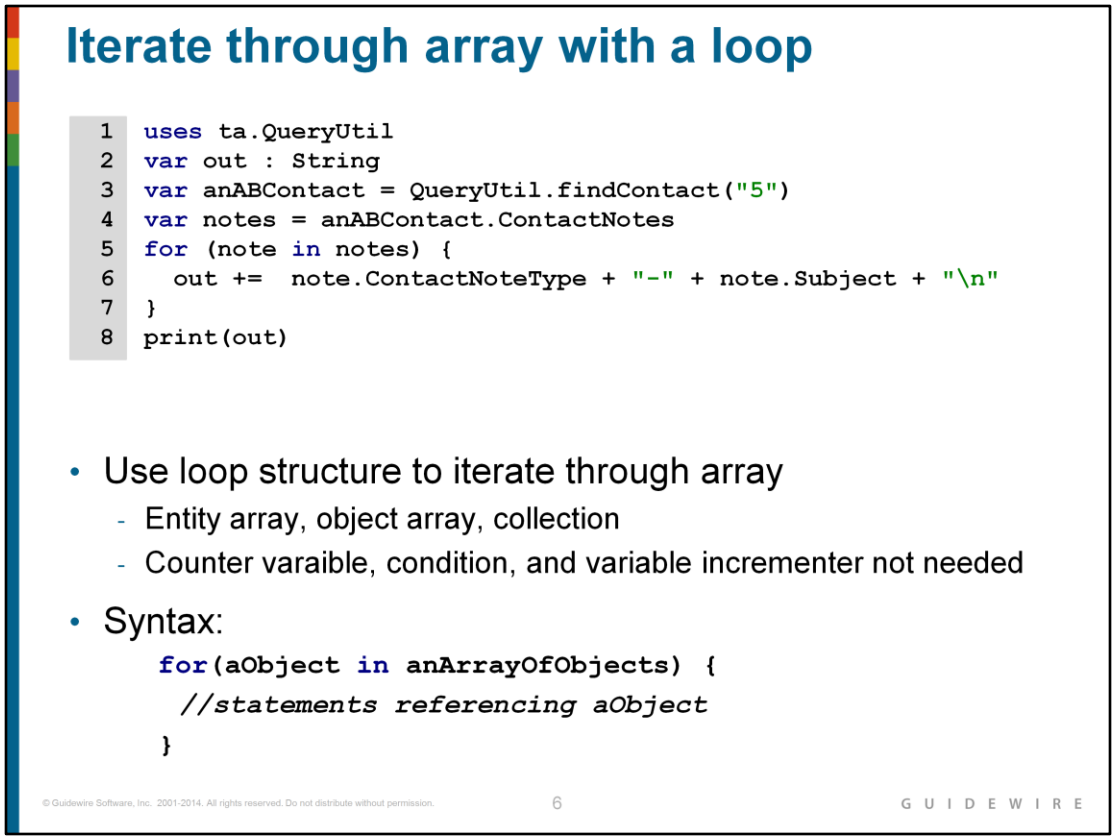


- Determine the length (size) of an array
- Syntax:

  arrayName.length

The array.Count property is an ehancement property for determining the number of element (objects) in the array.

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.



Many programming langauges create a loop using a counter variable, a condition, and a counter variable incrementer, such as:

```
for (x=0, x<anArray.length, x++) {
   // statements referencing anArray[x]
}
```

The syntax for Gosu loops does not require a counter, condition, and incrementer. You only need specify a name for the current object (the currentObject placeholder). The name of the object automatically references the current object in the array. The for loop inherently advances to the next object in the array when the end of the loop is reached.

Gosu also supports these While and Do…While loops, but in practice they are often not needed. For more information, refer to the *Gosu Reference Guide*

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Add an index for a loop

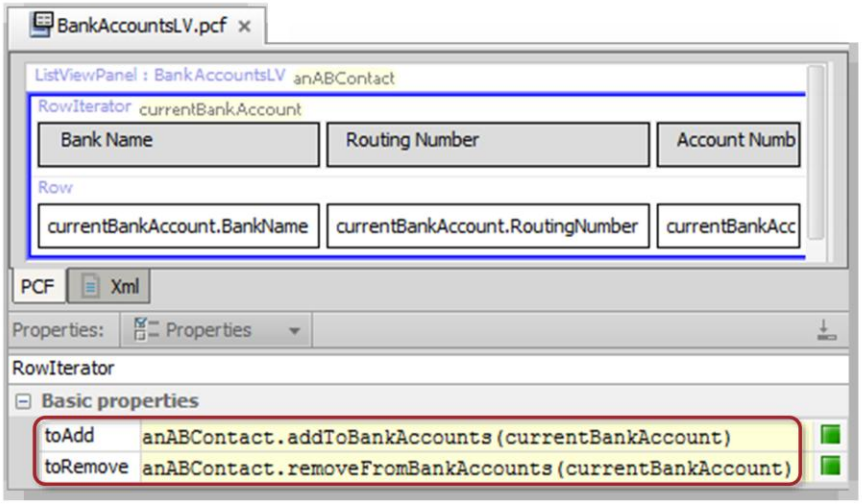
```
1  uses ta.QueryUtil
2  var out : String
3  var anABContact = QueryUtil.findContact("5")
4  var notes = anABContact.ContactNotes
5  for (note in notes index i) {
6    out += i + ")" note.ContactNoteType + "-" + note.Subject + "\n"
7  }
8  print(out)
```

- Add an index using the index keyword
- Syntax:

```
for(aObject in anArrayOfObjects index indexVariable) {
  //statements referencing aObject
}
```


You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Row iterator array methods



- Define expressions in row iterator properties to add and to remove elements from data model entity array
  - addTo...() and removeFrom...() methods

The addTo...() and removeFrom...() methods work only for arrays that are declared in the data model entities.

The slide example is from the BankAccountsLV row iterator. The list view panel is The parent container has a toolbar with Edit buttons and references the BankAccountLV.

Clicking the Add button creates a new bank account object which can be referred to as currentBankAccount. The toAdd property specifies the Gosu to add the object to the BankAccounts array on anABContact.

The addTo and removeFrom methods are available whe Gosu runs in an application context with a current editable object. If you call these methods in Gosu Scratchpad, Scratchpad needs to run in a debug server process and the object needs to be within a bundle transaction scope.

Lesson outline

- Arrays
- Array method expressions

Array methods that require logic

- Array methods include
  - Getting information about the array
  - Getting members of the array that match a given condition
- Some methods require an argument
- Argument must be an expression of a condition, such as:
  - Returns true if any row in the array matches the condition
  - Returns the first row that matches the condition
  - Returns all rows that match the condition
- Argument must be a Gosu block expression

Blocks argument expression

```
                                  ① ②   ③
var notes = someNotes.where( \ note ->
                          ④ {   note.NoteType ==
                                typekey.NoteType.TC_GENERAL
                            )
```

- A **block** is an expression of logic passed to a method as an argument
- Some array methods require conditions using a block
- The block consists of four parts:
  1. \ , which identifies that the following is a block
  2. An element name representing each element of the array
  3. **->**, which identifies the start of the condition
  4. A condition, which relates to the element

Blocks are used in many situations where a method requires an expression as an input parameter, such as in arrays, queries, and Transaction bundles for database transactions.

Arrays (discussed in Arrays lesson)

Queries (discussed in Queries lesson)
or() - used by queries objects to join multiple conditions together with OR logic
orderBy() - used by result set objects to specify logic for how to order the results in the set (in ascending order)
orderByDescending() - used by result set objects to specify logic for how to order the results in the set (in descending order)

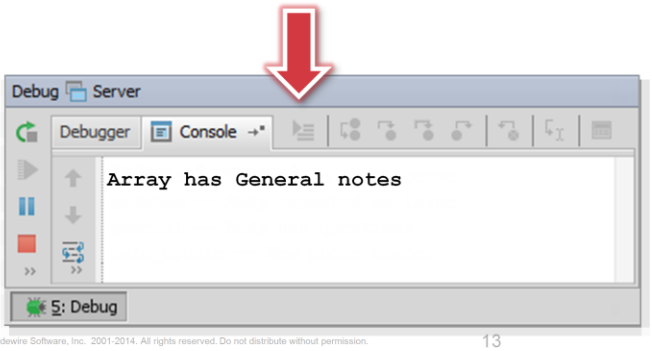
Database transactions and bundles (discussed in Application courses)
runwithNewBundle() - used to create a bundle of run-time objects and then commit data in those objects to the database

Common array methods

- **`hasMatch(condition)`**
  - Determine if any element in an array matches a given condition
  - Syntax: **`array.hasMatch(\ name -> conditionToMatch)`**
- **`countWhere(condition)`**
  - Returns count of elements that match a given condition in an array
  - Syntax: **`array.countWhere(\ name -> conditionToMatch)`**
- **`firstWhere(condition)`**
  - Retrieve the first element in an array that matches a given condition
  - Syntax: **`array.firstWhere(\ name -> conditionToMatch)`**
- **`where(condition)`**
  - Retrieves a target array that consists of all the members of a source array that match a given condition
  - Syntax: **`array.where(\ name -> conditionToMatch)`**

Example: hasMatch()

```
1  uses ta.QueryUtil
2  var out : String
3  var anABContact = QueryUtil.findContact("5")
4  var notes = anABContact.ContactNotes
5  if (notes.hasMatch( \ note ->
                       note.ContactNoteType ==
                       typekey.ContactNoteType.TC_GENERAL
                     )){
6    out = "Array has General notes"
7  }
8  print(out)
```



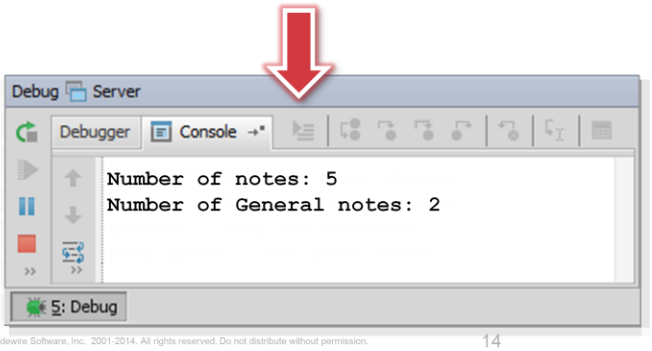
GUIDEWIRE

In the slide example, Line 5 illustrates an example of a hasMatch() method block for an entity data model array. The hasMatch() method returns a boolean if any of the objects in the source array that match a given condition. In Line 5, the condition statement tests to see if there is a ContactNote for a given contact that has a ContactNoteType with the TC_General typecode.

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Example: countWhere()

```
1  uses ta.QueryUtil
2  var out : String
3  var anABContact = QueryUtil.findContact("5")
4  var notes = anABContact.ContactNotes
5  var count = notes.countWhere( \ note ->
                                 note.ContactNoteType ==
                                 typekey.ContactNoteType.TC_GENERAL
                                 )
6  out = "Number of notes: " + notes.length
         + "\n  Number of General notes: " + count
7  print(out)
```

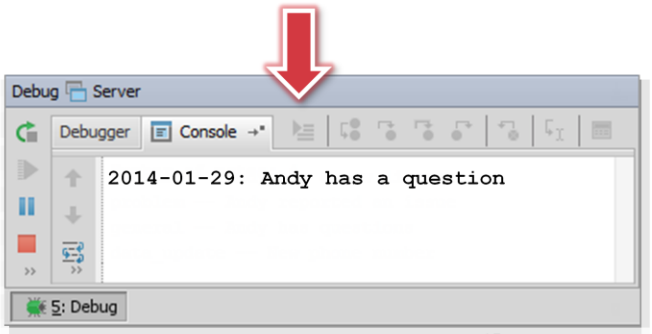


In the slide example, Line 5 illustrates an example of a countWhere() method block for an entity data model array. The countWhere() method returns an integer for the count of all the objects in a source array that match a given condition. In Line 5, the condition statement is to determine if a ContactNote for a given contact has a ContactNoteType with the the TC_General typecode.

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Example: firstWhere

```
1  uses ta.QueryUtil
2  var out : String
3  var anABContact = QueryUtil.findContact("5")
4  var aNote = anABContact.ContactNotes.firstWhere( \ note ->
                              note.ContactNoteType ==
                              typekey.ContactNoteType.TC_GENERAL
                              )
5  out = aNote.CreateDate + ": " + aNote.Subject + "\n"
6  print(out)
```

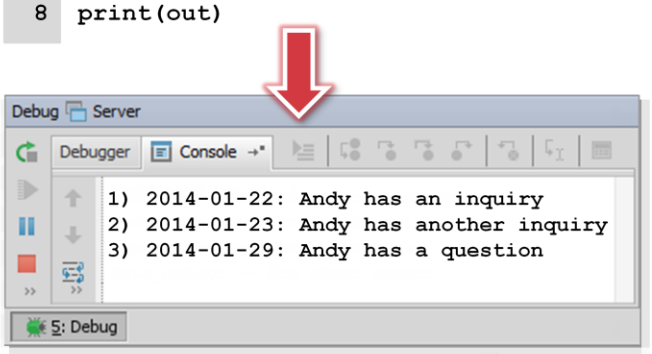


In the slide example, Line 4 illustrates an example of a firstWhere() method block for an entity data model array. The firstwhere() method returns an object from the source array that matches a given condition. In Line 4, the condition is to return a ContactNote object for a given contact where the ContactNoteType uses the TC_General typecode.

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Example: where()

```
1  uses ta.QueryUtil
2  var out : String
3  var anABContact = QueryUtil.findContact("5")
4  var notes = anABContact.ContactNotes.where( \ note ->
                              note.ContactNoteType ==
                              typekey.ContactNoteType.TC_GENERAL
                              )
5  for (aNote in notes index i) {
6    out += i + ") " + aNote.CreateDate
               + ": " + aNote.Subject + "\n"
7  }
8  print(out)
```



GUIDEWIRE

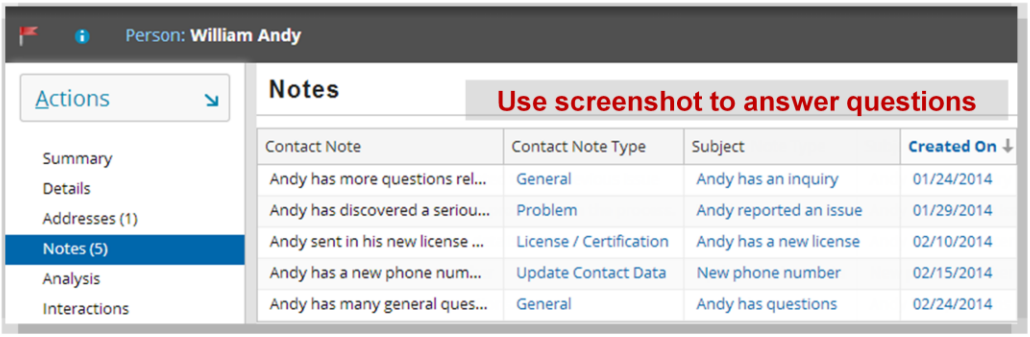
In the slide example, Line 4 illustrates an example of a where condition block for an entity data model array. The where() method returns a target array that consists of all the members of a source array that match a given condition. In Line 4, the condition is to return an array of ContactNotes for a given contact where the ContactNoteType uses the TC_General typecode.

You can execute the code in the slide example in the Debug Server process. To debug the server, select Main menu → Run → Debug 'Server' or Main menu → Run → Debug… → Server. In the Debug tools window, confirm that the application is running and is ready (***** AppName ready *****). Then, open Gosu Scratchpad using Main menu → Tools → Gosu Scratchpad or with the keystroke ALT+SHIFT+S. In Gosu Scratchpad toolbar, click Run in Debug Process.

Lesson objectives review

- You should now be able to:
  - Describe what is an array
  - Determine the number of elements in an array
  - Loop through each object in an array
  - Add object to (and remove object from) an array
  - Determine if any objects with a given criterion exists in an array
  - Retrieve objects in an array that match a given criterion

Review questions



1. What is the array method to return the number of objects in the ContactNotes array? What is the value?
2. What is the expression to return the count of notes that are regarding a problem? What is the count value?
3. What is an expression to return an array of ContactNotes where the note was created after January 31, 2014?

Answers
1) var count = array.length or array.Count.
The total number of elements in the ContactNotes array is 5.
2) var countNotes = notes.countWhere(\ note -> note.ContactNoteType == typekey.ContactNoteType.TC_PROBLEM)
The total number of elements in the ContactNotes array is 1.
3) var filteredNotes = notes.where(\ note -> note.CreateTime >= "02/01/2014" as java.util.Date)

Notices