

15



Training Module

A new module for delivering and tracking user training.

This chapter shows how to create a ServiceCenter training module. This is a simple slide show type trainer. When the user completes a course, they can take a quiz. The quiz score is then recorded in a training record. This technique allows:

- Define courses with one or more lessons
- Presents material in each lesson with one or more slides
- Define and present multiple choice tests
- Keep a record for each operator on completions
- Remember which course is in progress, and returns to the current lesson

HOW IT WORKS

There are four files used by this module: A **courses** file holds a single record for each course defined. It stores information about the course, and who should take it. A **lessons** file holds the actual course material. Each record holds a slide. The material is stored in a text array, which is shown to the user. They can move from one slide to the next. A **quiz** file stores questions about the material. Each question has four possible answers. A correct answer is defined, and grades the results. A **training** file record is created for each login taking a course. It holds an array of each course taken, and the quiz score. It also keeps track of a course in progress, if the quiz is not completed.

COURSES

First, you must create the database file to hold the course records. The courses database has one record for each course offered. The unique field is a course ID holding a short course identifier. The longer name is held in the title field. A user type field identifies the type of user. This corresponds with the Resource Type field in the operator table. This determines which courses an operator should take.

- 1 Create the **ndpcourse** dbdict as shown.

Database Dictionary		
File Name	ndpcourse	
Field name	Type	Keys
course.id	char	unique
title	char	course.id
description	array/char	
user.type	num	
sysmoduser	char	
sysmodtime	d/t	

- 2 The format **ndpcourse** enters the information about a course. The **course.id** field holds a unique course number. This can be something like school course numbers. And they can group or sort the courses. The **title** field has the long name for the course. The **user.type** field corresponds to the User Type field, in the operator record. If using that field, you can use it to filter the list of courses for an operator. Create the format as shown.

Figure 19 A course record

- 3 Create a format control for **ndpcourse** to make the **Title** field mandatory, and to default the **User Type** to All.

Format control	
Name	ndpcourse
Calculations	
Comment	Default the User Type to 0
Add	true
Upd	true
Calculation	<i>user.type in \$file=nullsub(user.type in \$file, 0)</i>
Validations	
Comment	Make the Title mandatory
Add	true
Upd	true
Validation	not null (<i>title in \$file</i>)
Message	You must enter a Title
On Failed ...	title

 **Variables:**

Variable	How it is used
\$file	The file variable in format control.

 **Functions:**

Function	How it works
null()	Returns a true if the argument is null, false if it is not null.
nullsub()	If the first argument is null, then it uses the second argument.



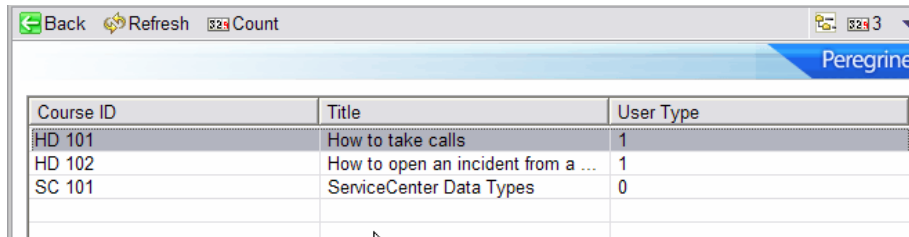
What is the null() function?

You can test if a variable is null, by using the null() function. The expression null(\$a) returns true if \$a is null. The expression \$a=NULL also returns true if \$a is null. Use cleanup(\$a) to set it to null. Using \$a=NULL also works, but \$a=NULL will NOT WORK.

Remember, a string can be blank if it is set with \$a="" but that is not null. If you print it, the output will be blank, but null(\$a) returns false.

4

Create the format **ndpcourse.qbe** to display a list of courses.



Course ID	Title	User Type
HD 101	How to take calls	1
HD 102	How to open an incident from a ...	1
SC 101	ServiceCenter Data Types	0

Figure 20 The qbe list that shows available courses

- 5 Create a globallist definition to create a list of courses, when a user starts the trainer. The global variable is created when called by the script.

Globallist	
List Name	ndp.training
Build List on Start Up?	false
List Variable	<i>\$G.courses.all</i>
Display Variable	<i>\$L.void</i>
List Field	course.id
Display Field	course.id
Filename	ndpcourse
Limiting SQL	true
User Defined List	false

Variables:

Variable	How it is used
<i>\$G.courses.all</i>	The global variable created when this globallist record is executed.
<i>\$L.void</i>	An unused variable creating the display list.

USER RECORDS

Another goal of the trainer is to maintain a history of what courses a user has taken. This is stored in the dbdict **ndptraining**. This database keeps track of what training operators have completed. It also records what lesson the user is working on. The course and status arrays maintain a list of courses the user has taken.

- 1 Create the dbdict **ndptraining** as shown.

Database Dictionary		
File Name	ndptraining	
Field name	Type	Keys
namne	char	unique
user.type	char	name
current.course	char	
current.lesson	num	
course	array/char	
status	array/char	
sysmoduser	char	
sysmodtime	d/t	

2 Create a format named **ndptraining** allows access to the database.

The screenshot shows a software window titled 'User Training' with a 'Peregrine' logo. The window contains a form with the following fields:

- Login Name:
- User Type:
- Current course:
- Current lesson:

Below the form is a table with two columns: 'Course' and 'Status'.

Course	Status
HD 101	0
HD 102	100

Figure 21 An individual record of training

PROCESS FLOW FOR LESSON AND QUIZZES



LESSONS

To hold the slides, you need the dbdict named **ndplesson**. This database holds the slides presented to the user. It can be broken down into lessons, with each lesson having more than one slide. Each lesson has a number, which orders the lessons. The **last.slide** field determines when the user can take the test to complete the course.

1 Create the dbdict **ndplession** as shown.

Database Dictionary		
File Name	ndplession	
Field name	Type	Keys
lesson.id	num	unique
course	char	lesson.id
lesson.num	num	Nulls & Duplicates
lesson	char	course
slide	num	lesson.num
last.slide	logical	slide
text	array/char	
sysmoduser	char	
sysmodtime	d/t	

2 This requires a counter for the **lesson.id** field. Create the counter record as shown.

Counter	
Table Name	ndplession
Column Name	course.id
Current Value	100



Why use a counter?

A counter insures you never get a duplicate key error, when adding a record. Since you may want to create your course out of order, you may create them all with the same slide number. Then go back and renumber them correctly.

3 You need two formats for **ndplession**. One creates the slide and the other presents it to the user. First create the format **ndplession** to create a slide.

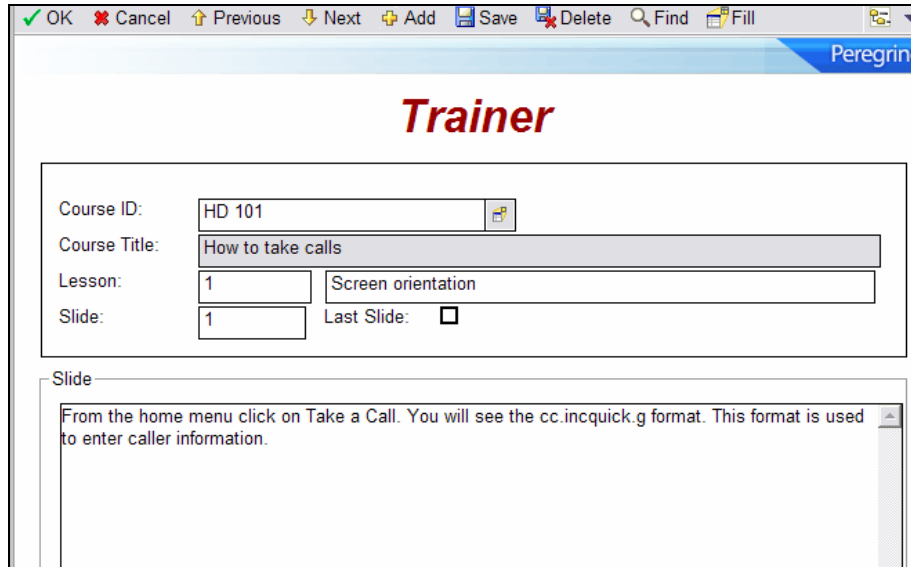


Figure 22 An example of a single lesson slide

Notice the Course Title is Read Only and there was no field for it in the dbdict. This field is a virtual join from the **ndpcourse** file. That way you never misspell it, even if it changes.

- 4** You must create a subformat named **ndpcourse.vj** with one field as shown.

Object Type	Text
Property	Value
Caption	
Input	title
X	0
Y	0
Height	2
Width	124

- 5** You need a format control for the **ndplession** format. It sets defaults and enforces required fields.

Format control	
Name	ndplession
Calculations	
Comment	Default last slide to false
Add	true
Upd	true
Calculation	<i>last.slide in \$file=nullsub(last.slide in \$file, false)</i>

Validations	
Comment	Make the lesson number mandatory
Add	true
Upd	true
Validation	not null (<i>lesson.num in \$file</i>)
Message	You must enter a Lesson Number
On Failed ...	<i>lesson.num</i>
Comment	Make the lesson title mandatory
Add	true
Upd	true
Validation	not null (<i>lesson in \$file</i>)
Message	You must enter a Lesson Title
On Failed ...	<i>lesson</i>
Comment	Make the slide number mandatory
Add	true
Upd	true
Validation	not null (<i>slide in \$file</i>)
Message	You must enter a Slide Number
On Failed ...	<i>slide</i>
Comment	Make the text mandatory
Add	true
Upd	true
Validation	not null (<i>text in \$file</i>)
Message	You must enter the Text
On Failed ...	<i>text</i>

 **Variables:**

Variable	How it is used
<i>\$file</i>	The file variable in format control.

 **Functions:**

Function	How it works
null()	Returns a true if the argument is null, false if it is not null.
nullsub()	If the first argument is null, then it uses the second argument.

6 Create the link record to go with the format **ndplession**.

Link	
Name	ndplesson
Comment	
Field on current format	course
File/format to get data from	ndpcourse
Field in file	course.id
Query	
QBE Format	
Expressions	
Target field	Source field
course	course.id

7 Next, you need to create a format for the user receiving the training. Create a format as shown. Notice the fields are Read Only and the course title is a virtual join.

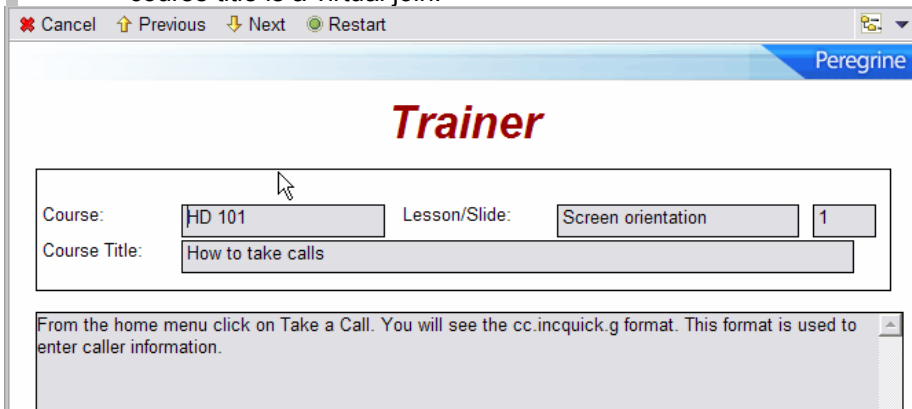


Figure 23 A slide as shown to a user

8 Create the format control record for **ndp.lesson** as shown. It updates the operator's training record indicating the course being taken.