SOLUTION REFERENCE:	Comp/AD-Callout/25May/Red1/1.0
APPLICATION:	Compiere ERP/CRM v2.5.0e on Oracle 9i
TOPIC:	*Yes! I Know What's Application Dictionary and Callout Coding*
VERSION:	2.0
DATE:	May, 2004, July(v2.0).
AUTHOR:	red1 red1Compiere Workshop, Malaysia
EMAIL CONTACT	redhuanoon@yahoo.com, red1@red1.org
CLIENT:	No Pain No Gain Sports Centre, MK Holdings Group

To Find What Others Want Is OK,
To Find What You Want Is Just Fantastic.

CONTENT

Is there one thing you can indulge in,
where you end up enriched, even the rest of us?

This brief exercise (if you can cut through the thick wood) seeks to instruct you on how to do some Application Dictionary stuff, and really stretch a Callout. Along the way you also learn how to bring the Compiere Source into Eclipse, and check out Resource Assignment and its cool Info Schedule.

- advertising space –
(a form of donation, dummy)

The Big Picture

Client's Requirements

The second part of the Client's need I wrote about earlier, (refer POSred.zip) requires the Application to manage their Facilities bookings and sales. The main facilities are 4 courts that can be converted within 5 minutes to play games such as Futsal, Basketball, Volleyball and Netball. They are open from 10am till midnight.

Customers will call up the Sports Centre and inquire about the available time slots for these courts. Some want to have the court for 2 hrs, some wants it for 1 ½ hrs. Most want to play during after-office hours. So, this leaves the office hours as non-peak hours. Peak hours are charged at RM100. Non-peak at RM55.

When the booking is made, the Schedule Info must display the name of the customer, the contact phone number, the type of game and other miscellaneous details. The Operations crew uses the game type info to prepare the court with the respective equipment.

When the customer arrives, the Sales Order screen must pull up the info easily for payment to be made. An invoice receipt is to be printed in triplicate. Even when a follow-up inquiry is made, the system must be robust and provide info on what booking is there, for what game and when, without bothering the customer to verbalise further. If there are change requests, or cancellations, the schedule must be updated. Other PCs in the network must access the same info real-time too. The users of the PCs should not be able to access other Point of Sales information or other financial processes.

Compiere – the Gap Muncher

The good news is that Compiere has an inbuilt Schedule Info with time slots. However there are bound to be some gaps that can't meet exact needs. The better news is that with Open Source, where the codes are available and well designed and documented, meeting the needs are desired rather than feared. Gaps meet its end when a developer meets the codes. It is highly gratfying, making the codes munch away the gaps wherever you find them.

Now lets savour the gaps. Firstly you have to change lots of standard Compiere labels to those that define the Client's business. Now isn't this exactly what the Application Dictionary (AD) is designed for? As we continue stating the other needs, we will enclose the Compiere equivalent in brackets to show you what we will be changing from - that we are on the same screen, so to speak. When you select the Facility (Resource Assignment product field (Sales Order Window, Orderline Tab)), it pops up another window expecting you to fill in the same name and phone number of the Customer (Business Partner) you are assigning that resource to. We shall program those pop fields to be auto-populated. That will be done through the Callout pattern in Compiere. To pin the game type to the order, we shall use a pull down menu selection (AD Reference List). For login by the Front Desk Executive (Sales Rep), she won't be able to access the back-end of the system (User / Role).

Another thing about the Schedule Info screen, the opening period doesn't cover midnight properly. It thinks that midnight is the same as 0 hr in the morning which is true, if you remove the date context. So what happens is that the slot array collapses when you have a 0hr to 12 midnight time-frame. Closing at 11.58pm will trick it into not collapsing. But then the 11pm slot thinks that it is not open and paints 'unavailable' over it!

The Application Dictionary

How do we remove coding work from the application? We first have to understand the behaviour of applications. Well, the fundamental work of an application is to process data, usually taking them from tables and putting them back into Tables. All that is done via an interface which we know as Windows. Let's use our imagination to separate what doesn't change from what will keep-changing. Fundamentally Windows display what and how we want to see things. The keep-changing elements are the look and feel, the arrangement of windows, tabs, and fields, and the reports.

The universal stuff such as *create* new, goto *next* record, *trash* this record, *search* for that nameless thingy is a constant and don't seem to change. The hooks for those tasks always remain on the menu and stays there so much so you forgot about them.

The next keep-changings are business rules or logic that concerns what-to-dos after or before fetching some data. Smart thinking has kept them contained in the middle tier away from the well-designed database level. On how that middle organises itself we shall leave it for another day.

Compiere attempts to absorb all these changes as much as it can through the use of metadata. Its data about data. Or to be exact more humane codes that replaces less humane codes. So instead of coding the interface changes entirely in a polynesian sounding Java codes, we just give instructions to another more english and visually humane interface called the Application Dictionary (AD). Much of the interface changes can be handled by the AD Menu, AD Window, AD Table & Column. For the logic and rules, there are AD Reference, AD Validation and meta SQL clauses.

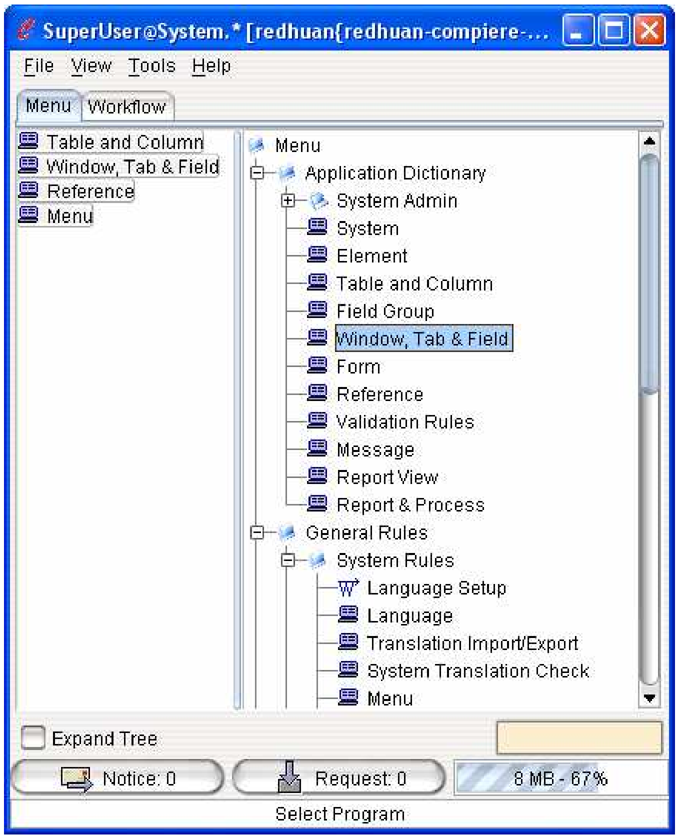


Figure 1 Application Dictionary, with 4 frequent selections placed on the left panel

But grand these desires maybe, there are certain plain Joe things we want to do further with a certain data field which may be beyond the AD.

For example, we may want to make it derive its value from the calculation of other fields. Or we want it to be proactive, jump around and check out another chick across town. Or we just want it to show whats going on without us asking. Sounds like what we could have done using a spreadsheet, another humane app. How do we deal with these in our application?

This is where the Callout comes in. It is designed in a container that gives us that spreadsheet capability but in Java terms. Doesn't this mean entering the oxymoron domain of a polynesian language?! Hahaha.

Developing with Eclipse

Eclipse is another Open Source project, which is getting monstrous and a developer’s winner to use in the battlefield. You will want to be able to use it in action to debug and change codes. So where Compiere leaves out, Eclipse will step in and let you have your day. Its no surprise that Compiere’s development platform has shifted from the proprietory JBuilder to the open all-plugs Eclipse.

Compiere allows further customisation to be done via Callouts. Ironically the concept of Callout in Compiere is to avoid programming! Strange, but in a way its true. When we put Compiere into the Eclipse (often reminds me of The Matrix movie), we won’t be facing messy streams of meaningless symbols. We will begin to notice patterns, elegance of design, well documented cues. If I may go extreme here with my Matrix analogy, the Callout is likened to the phone booth in the movie! You expect to look for it, to dial back into the system.

But still when we wish to bring on bigger challenges such as extending with Fixed Assets, or Payroll, you can reuse the Compiere framework and its inherent patterns, which like I said before, offers a more humane look at software. You will end up with top class software that is highly integrated, configurable and stable. The obvious rule is that when you have an industry strong app in your lap, you have an industry strong app!

Sure, there is a catch. It’s a steep learning curve, to go through endless streams of codes and patterns. It will help if you get to practice a lot. That means hands-on, burning holes in your keyboard. But my angle here is not to just practice but think too about the big picture. Look out not just the common patterns and streams, but also for the context, the history and the culture. Perchance we may understand The Matrix – oops typo!

Callout Classes

Callout classes are located in the package org.compiere.model. It is located in the Base folder of Compiere-all source. They have certain super classes that handle the context of the Window, Table and Column you are calling from. In this way they resemble the Spreadsheet concept we talked about. By taking away the worry of how fields are really handled, we can keep our focus on the business logic.

For example let's say that we have 3 fields in our Compiere window namely: 'Price','Qty', and 'Total'. To make 'Total' = 'Price' X 'Qty' will take the form:

```
Qty = (BigDecimal)mTab.getValue("Qty");
Price = ((BigDecimal)mTab.getValue("Price"));
BigDecimal Total = Qty.multiply(Price);

mTab.setValue("Total", Total);
```

The **getValue** pattern basically obtain the value from the Window field in scope. The **setValue** will then place a new value into the Window Field. The *Total* field changes as you put in a new value into either *Qty* or *Price*.

Since we are are on the go, lets go the next mile (notice I did not say last mile) and see how we are going to refer to somebody across town. Let's say you want the *Price* to come from the Product table which is not refered to by the window. Here's is an idea of what you must do prior to the above, in the form:

```
String sql = "SELECT p.M_Price "
                  + "FROM M_Product p "
                  + "WHERE p.M_Product_ID=?";
            try
            {
                  PreparedStatement pstmt = DB.prepareStatement(sql);
                  pstmt.setInt(1, M_Product_ID);
                  ResultSet rs = pstmt.executeQuery();
                  if (rs.next())
                  {
                        M_Price = rs.getInt (1);
                  }
            mTab.setValue ("Price", M_Price));
```

Please don't stare too long, it's just to give an idea.

A Callout is first addressed in the AD Table and Column, Field section. In a way the callout is at the fringe of the AD. It's the final straw where the AD won't know how to satisfy your need and surrender to a group of java classes that will handle it better. And yet the way the container is designed makes the necessary evil so angelic. There's no red pill to take. Just some red syrup.

Ok, this is about the only military pet talk you are gonna get. Cos, we are going to the battlefield now, just stick to the guys and ignore the rest as noise. Sign the insurance papers. Grab a chute.

Tasks Index

We shall itemise the stuff that we are going to attempt here. We are going to…

1. Create a pull down menu selection to select the Game Type.
    a. Create a reference list for GameType, that has Futsal, Volleyball, Basketball and Netball.
    b. Create a new field called GameType in the C_OrderLine Table.
    c. Configure Menu and Window to have the Facility screen.
2. Making the Resource Assignment Product Callout to
    a. Fetch the Customer name and telephone number
    b. Fetch the GameType value
    c. Arrange them nicely
    d. Place them in the Resource Assignment table
3. Go and plunder the forums.

Suddenly a commoner rush from the crowd and tugged at the Master's robes, signalling him to bend lower. After some whisper into the Master's ears, that manages to raise one of the Master's eyebrows, he return to compose himself on the carriage set on the shoulders of 6 warriors. With eyes squirming, he peered at the vast heavens.

Ok, so I forgotten something.

0. How to set up Eclipse
    a. Importing Compiere Project into Eclipse
    b. Setting Compiere Debug Run

Setting Up

Now we are getting down to business. In front of your PC now, which is connected to broadband. Don't have broadband? Migrate.

Eclipse Setup

Go to www.eclipse.org and download the latest binary zip version. It is version 3.0M9 at this time of writing. This version also has taken care of some bugs such as not getting the JDK classpath.

When you unzip it, you specify your target directory to be a root level, for example C:\, or D:\. This is because it will make its own subfolders starting with eclipse\. You don't want things to get too nested around here.

Then you can go to the eclipse\ folder and find the eclipse.exe. Just double push on the finger and off we go.

One more thing – you need to activate 2 perspectives: Java and Debug. Goto Window on the top menu bar, select Open Perspective and select those two.

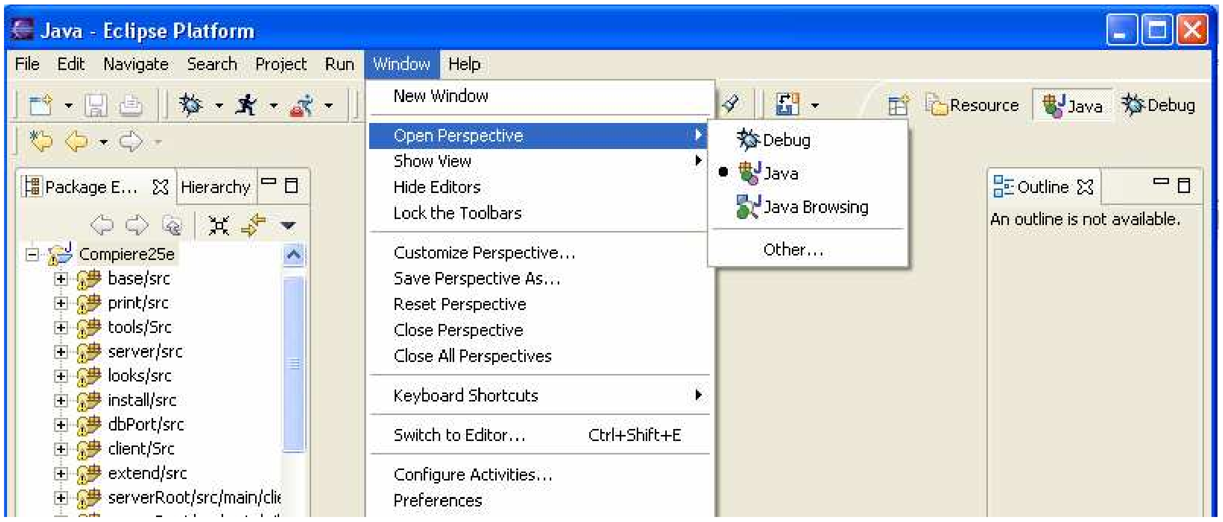
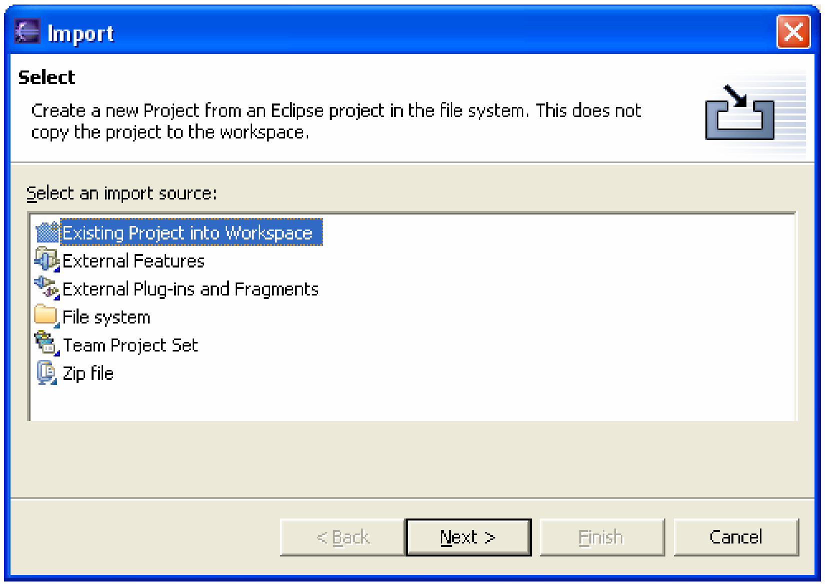


Figure 2 Getting the Debug and Java Perspectives out

You should spend some time reading up from the website on Eclipse, as well as check the FAQs and forums on common issues and tips. The more you know, the powerful you get, and the less work you do. And drop me a mail if at anytime you think you are getting smarter than me. Or else I will send Mr.Smith.

Importing Compiere Project

You can right click on the empty space in the left panel, and it should give you the choice of import. Otherwise call it the old way by selecting File on the top bar, and selecting Import.



Create a new Project from an Eclipse project in the file system. This does not copy the project to the workspace.

Select an import source:

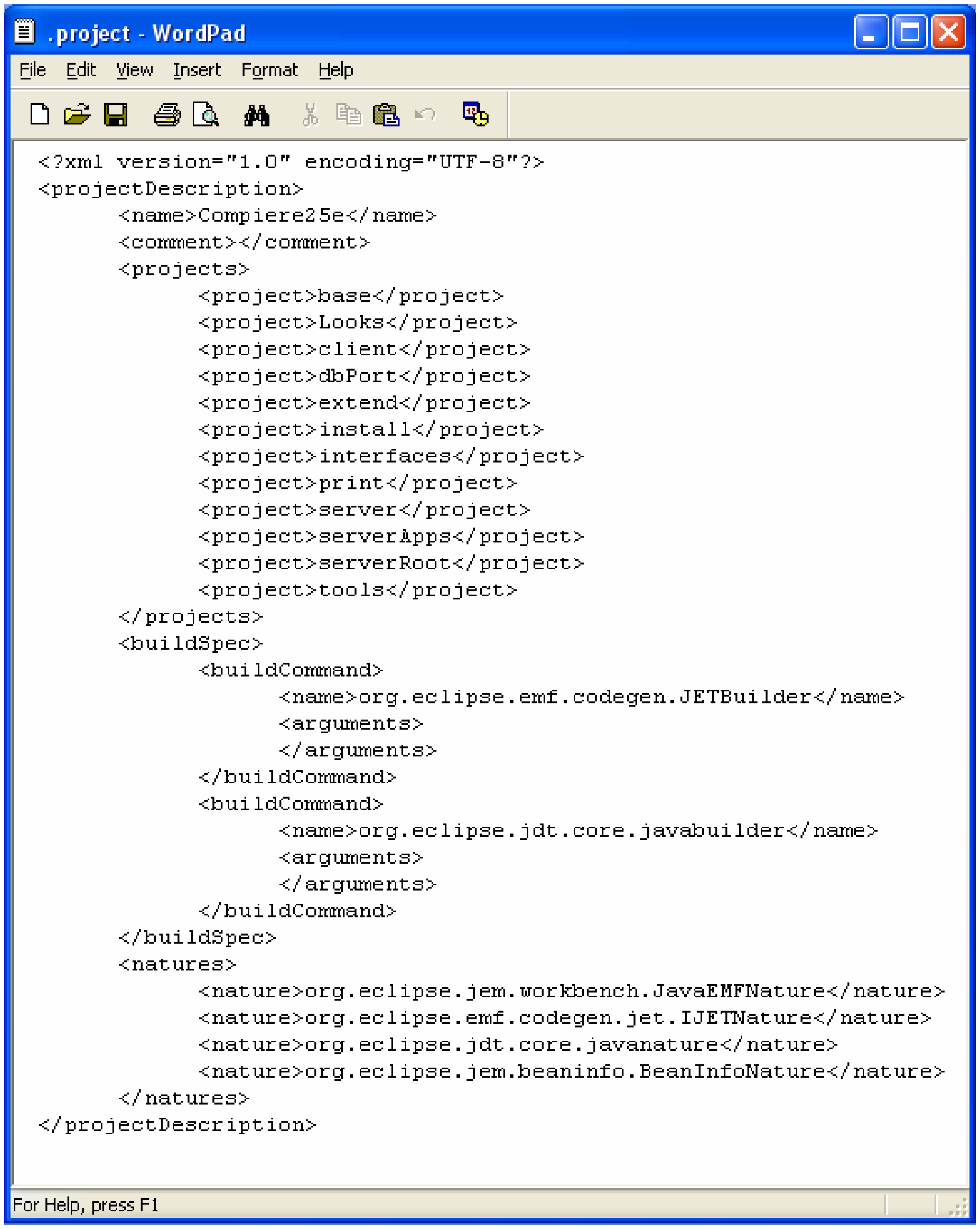
- Existing Project into Workspace
- External Features
- External Plug-ins and Fragments
- File system
- Team Project Set
- Zip file

Figure 3 Import Eclipse Project Dialog Box

Referring to this screen above, click on next, then browse for your Compiere source project file. You can download compiere from www.compiere.org (look for the download section, take the source zip which is 60Mb more, not the binary which is about 20Mb. Alternatively you can take the source from its CVS if you want the latest, but it may not be stable yet). When you explode Compiere source you will find a main sub directory: Compiere-all. That means that the source is with you. Finally, OBeOneCannotBe.

When you selected the Compiere-all folder, the Finish button will wake-up. This is because it found the .project file in the Compiere-all folder. If not, you can pick that with the dotClass file from http://compiere.red1.org/eclipsefiles.zip. Both these files are needed for Eclipse to grasp it tightly.

This is how the dotProject file looks like if you open it with WordPad.:



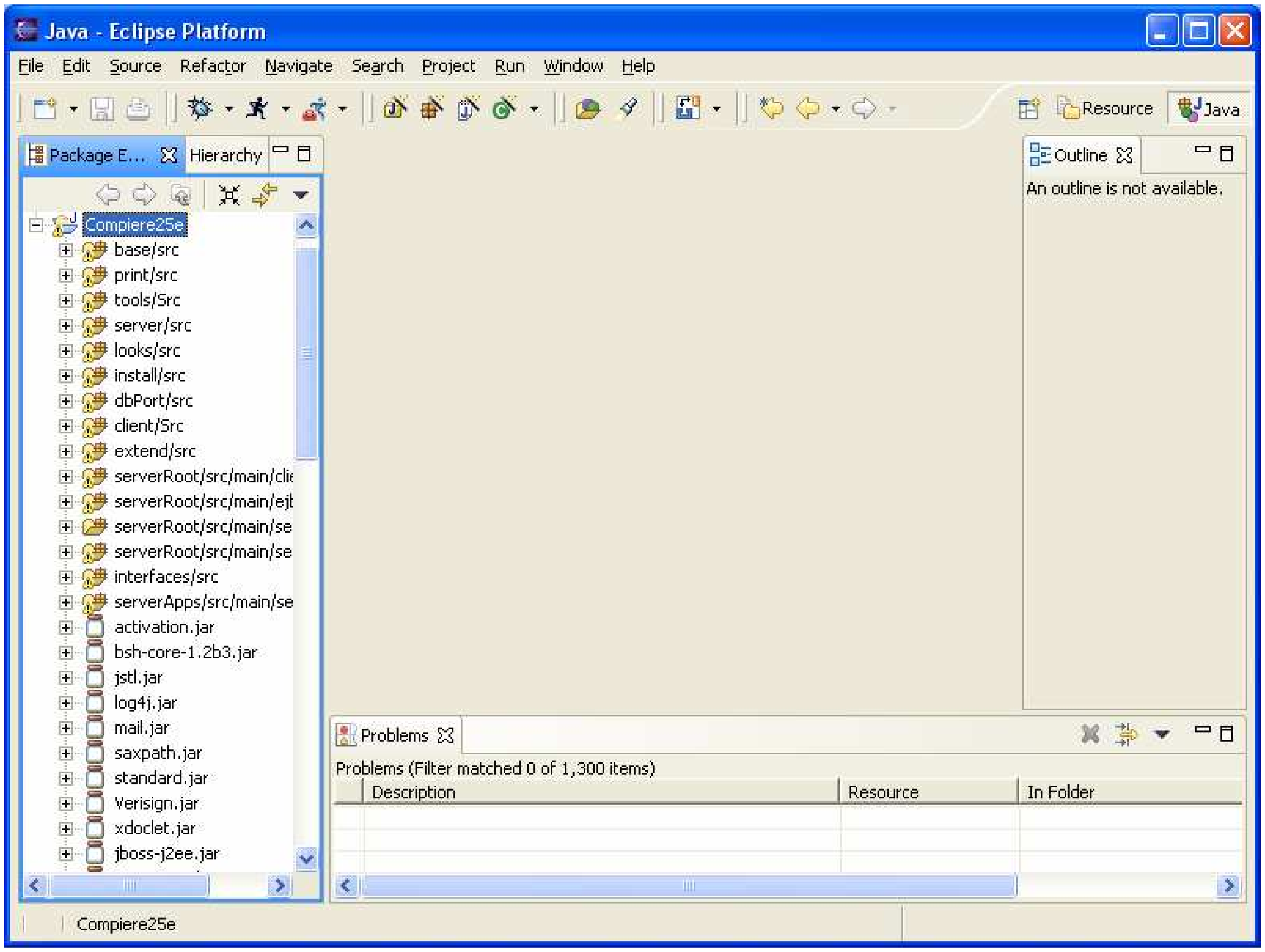
```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>Compiere25e</name>
	<comment></comment>
	<projects>
		<project>base</project>
		<project>Looks</project>
		<project>client</project>
		<project>dbPort</project>
		<project>extend</project>
		<project>install</project>
		<project>interfaces</project>
		<project>print</project>
		<project>server</project>
		<project>serverApps</project>
		<project>serverRoot</project>
		<project>tools</project>
	</projects>
	<buildSpec>
		<buildCommand>
			<name>org.eclipse.emf.codegen.JETBuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>org.eclipse.jdt.core.javabuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
	</buildSpec>
	<natures>
		<nature>org.eclipse.jem.workbench.JavaEMFNature</nature>
		<nature>org.eclipse.emf.codegen.jet.IJETNature</nature>
		<nature>org.eclipse.jdt.core.javanature</nature>
		<nature>org.eclipse.jem.beaninfo.BeanInfoNature</nature>
	</natures>
</projectDescription>
```

Figure 4 The .project file under Compiere-all folder

Note the text between the <projects> tags. That tells Eclipse what is in the package. Which will look like the next page when you hit the Finish button and give it a name, say: Compiere25e. (must I tell you everything? The Master examines the donation box, Guess so).

Running in Debug Mode

Your screen should look like the following. If not don’t panic. Study around a little. Things arent far off.



An outline is not available.

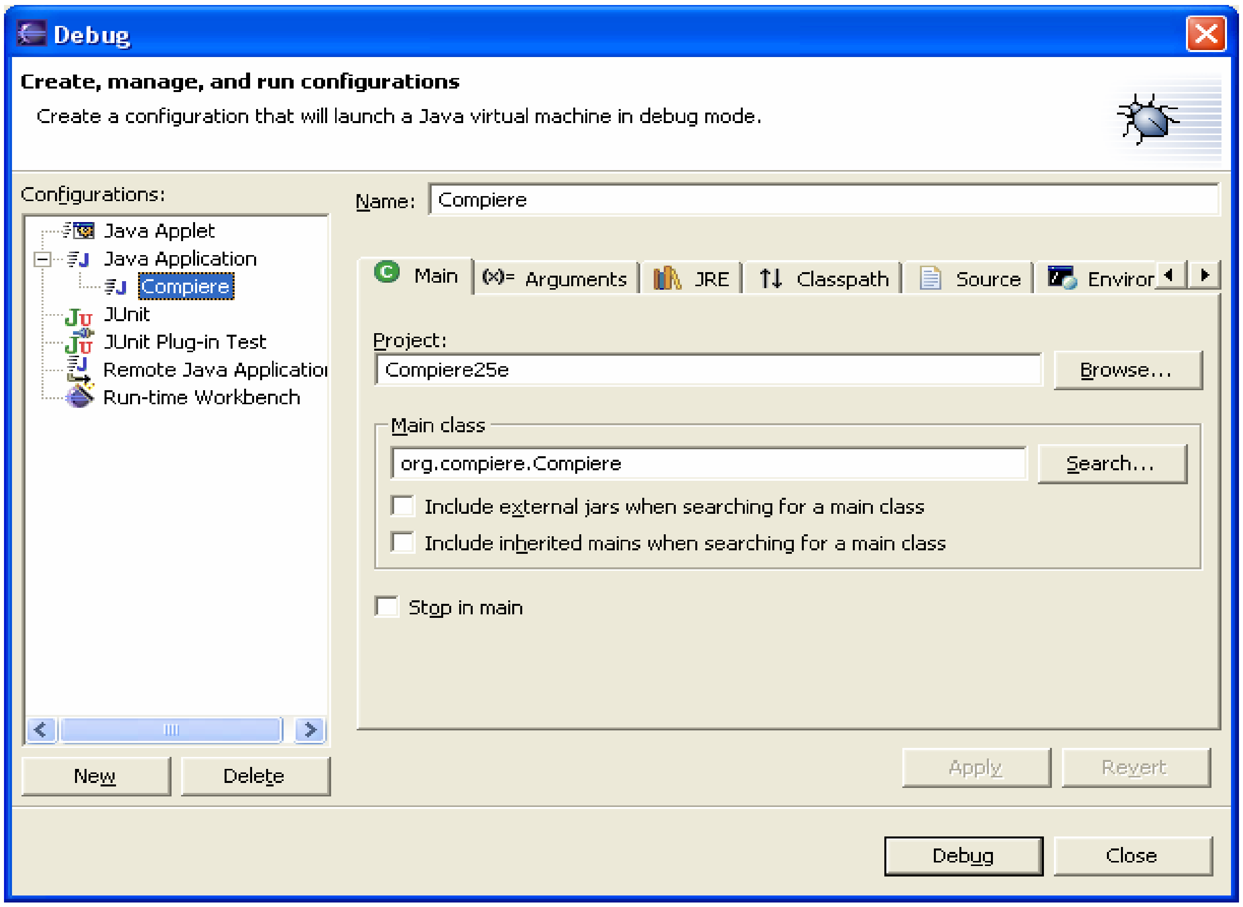
Problems (Filter matched 0 of 1,300 items)

Figure 5 Compiere Source package layout in Eclipse Java Perspective

If there is an error in your folders, and its in interfaces/src where there is a red mark on this file - CtrlMBean.java. Just zap it. Yeah! Delete it. Its alright, there is another copy somewhere for the deployment of JBoss service, which we cannot get to debug. We only debug this part of Compiere as it is on the JVM service. The other part – Accounts Posting and Web Interface runs on the JBoss application service, and you can only see their debug prompts off the console.

Heavy stuff, dude! It sounds like The Matrix II. Bring me to the Architect. I got this question: I wonder if you can tell me how I can setup both services to run on Eclipse the same time. ☺

Now where was I? Oh, I just finished loading the Compiere project into Eclipse, and I should be making Compiere to debug from it. To do that, look for that little bug icon up there (of course it looks like a bug, dummy). Click on it, or you can pull down and click on it if it has another default in the way. Create a New Java Application (select Java Application, then New), goto Browse for your project. Search for a main class, in the pop up you can pick Compiere. Otherwise you can physically type that in. Give it a name: 'Compiere'. In the final analysis this is what it would look like.


Debug

Create, manage, and run configurations

Create a configuration that will launch a Java virtual machine in debug mode.

Configurations:
- Java Applet
- Java Application
  - Compiere
- JUnit
- JUnit Plug-in Test
- Remote Java Applicatio
- Run-time Workbench

Name: Compiere

Project: Compiere25e

Main class: org.compiere.Compiere

Include external jars when searching for a main class

Include inherited mains when searching for a main class

Stop in main

Figure 6 Setting the main class to run/debug

See the debug button? Hit it! Are you in? You should be in the Debug perspective (click on the other bug, top, far right). If you see threads flying and the console prompts scribblings then you made it.

You can do the same setup for Run as you have done for Debug. For our AD Changes we can then use the Run mode, as we won't do code customisation yet. Let's get Compiere running and do our changes in the AD which are termed as configurations.

Configuring Compiere

All this while, we assume that you have setup Oracle and did the basic Compiere setup, so that when the debug mode runs it will access that database instance. Note that its not using the Compiere2 set of codes. Its running from the source itself. So whatever code changes will affect your Eclipse instance of Compiere. But whatever AD changes you do happen at the database level as they are metadata, not codes. So you may need to backup before proceeding. You can backup by going to Compiere2/utils/ and run DBExport.bat. Go ahead and do that, as it won’t bother if you are running or not. Its just a fast safe dump. Your backup will be in Compiere2/data/ as ExpDat.dmp.

Just to put it another way, you don’t worry about the Compiere2 binary directory, as you are using the Compiere-all directory when running from Eclipse. When you want to finally deploy, you will build the Jars and a fresh Compiere2 directory. Then you can send this to your production server. Clients that run their webstart will automatically refresh the CClient.jar that has been changed.

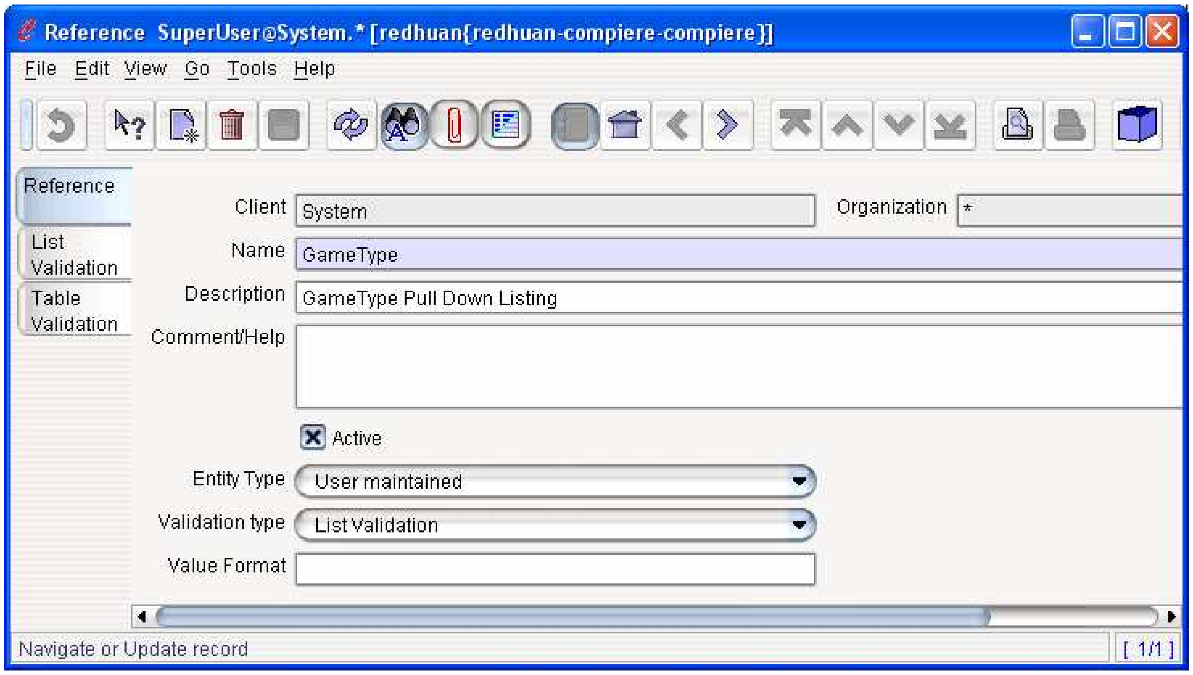
Alternatively when you build from some fresh changes, you only need to copy over CClient.jar which is more snappy. The other jars and JBoss were not changed, so why carry them around?

Let’s Run Compiere and login as System/System to do some AD changes. Don’t worry, we will resume the debug fun soon afterwards.

Configuring AD Reference

We will now do the pull down reference list of Game Type. We want it to appear in our Sales Orderline. In order to do that, we first define a reference list. Then we will reference that list in our Table and Column as a new field called GameType too. Then the Window must import this new field. Sorry if I keep repeating stuff all the time. Just don’t want you to fall onto the Dark Side.

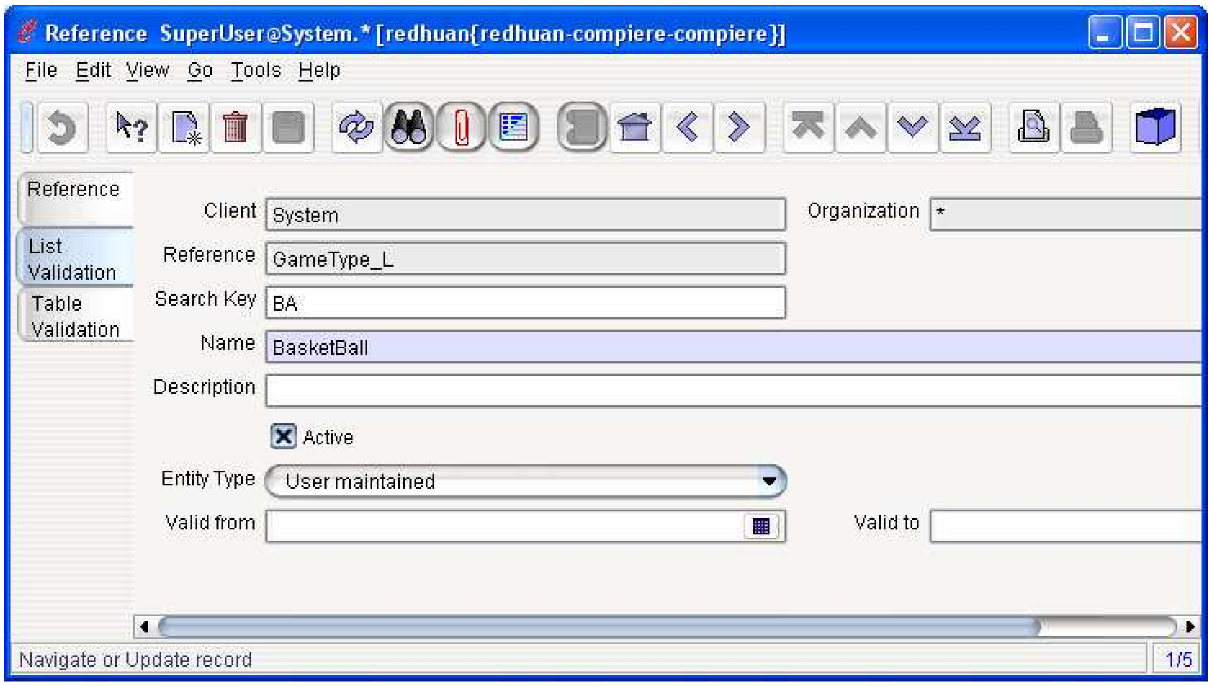
Click on Reference. Create a new list called GameType. Set the Validation type to List Validation.


Reference SuperUser@System.* [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Reference
List Validation
Table Validation
Navigate or Update record
[1/1]

Client: System
Organization: *
Name: GameType
Description: GameType Pull Down Listing
Comment/Help:
[✓ Active]
Entity Type: User maintained
Validation type: List Validation
Value Format:

Figure 7 Reference screen, main tab

Now that it is going to be a list, we have to specify the members of the list. Click on the List Validation tab.


Reference SuperUser@System.* [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Reference
List Validation
Table Validation
Navigate or Update record
1/5

Client: System
Organization: *
Reference: GameType_L
Search Key: BA
Name: BasketBall
Description:
[✓ Active]
Entity Type: User maintained
Valid from:
Valid to:

Figure 8 Reference screen, list validation tab

Now we create the different type of values that we want to appear in our menu list. The Search Key is the actual size and value that will be stored in the field of the Orderline afterwards. But the Name ‘Basketball’ or ‘Futsal’ will be the values you see when you pull down the list.

Configuring AD Table and Column

We will now create the GameType field in our Orderline table to store that value when the customer picks a choice during court booking. Go to the Menu and select Table and Column. Open the C_Orderline record. Select the Column tab, and create a new record. Note the length is 2 char.

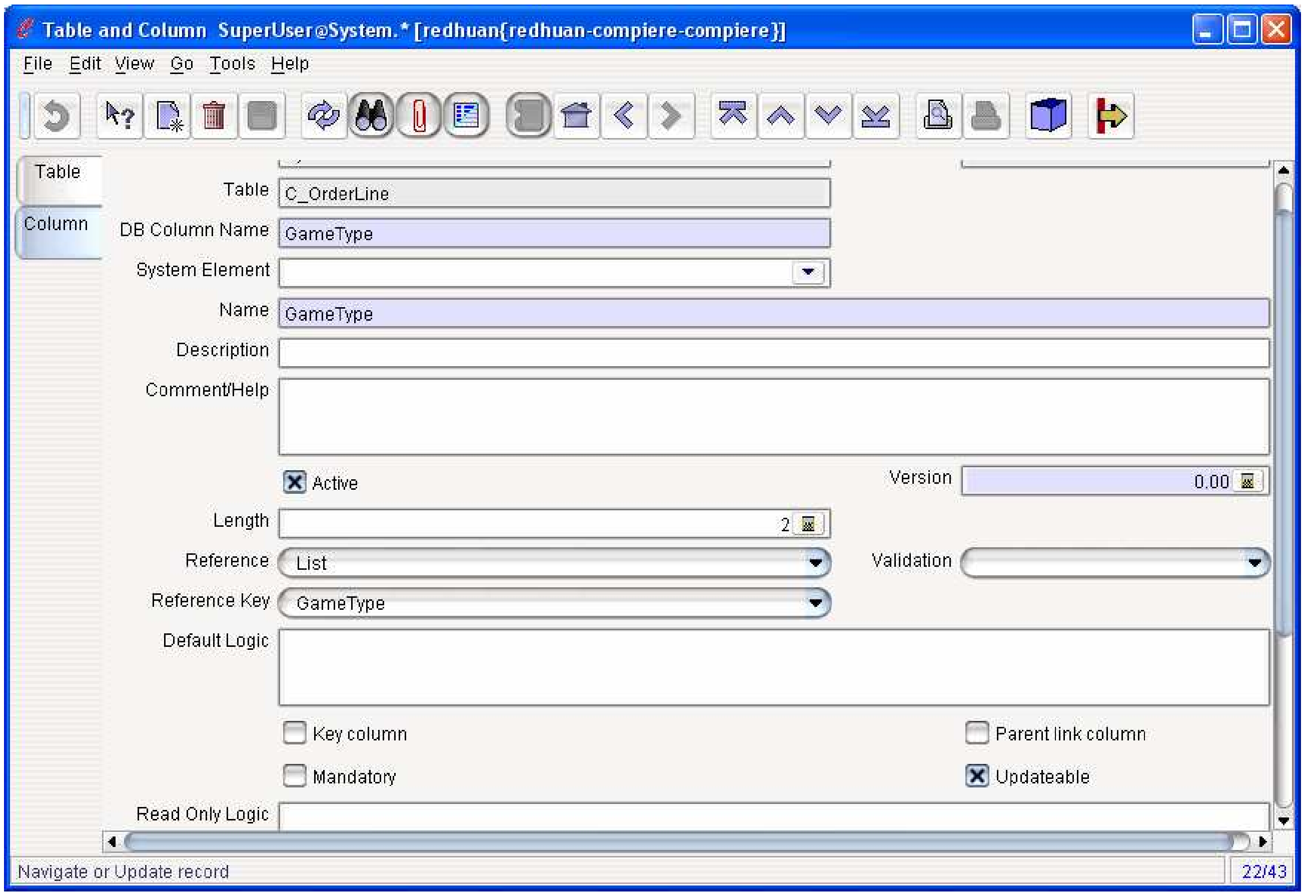
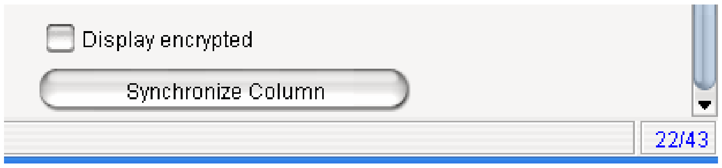


Table and Column SuperUser@System.* [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Table
Column
Navigate or Update record
22/43

Table: C_OrderLine
DB Column Name: GameType
System Element:
Name: GameType
Description:
Comment/Help:
☒ Active
Version: 0.00
Length: 2
Reference: List
Validation:
Reference Key: GameType
Default Logic:
☐ Key column
☐ Parent link column
☐ Mandatory
☒ Updateable
Read Only Logic:

Figure 9 C_Orderline table, create new record

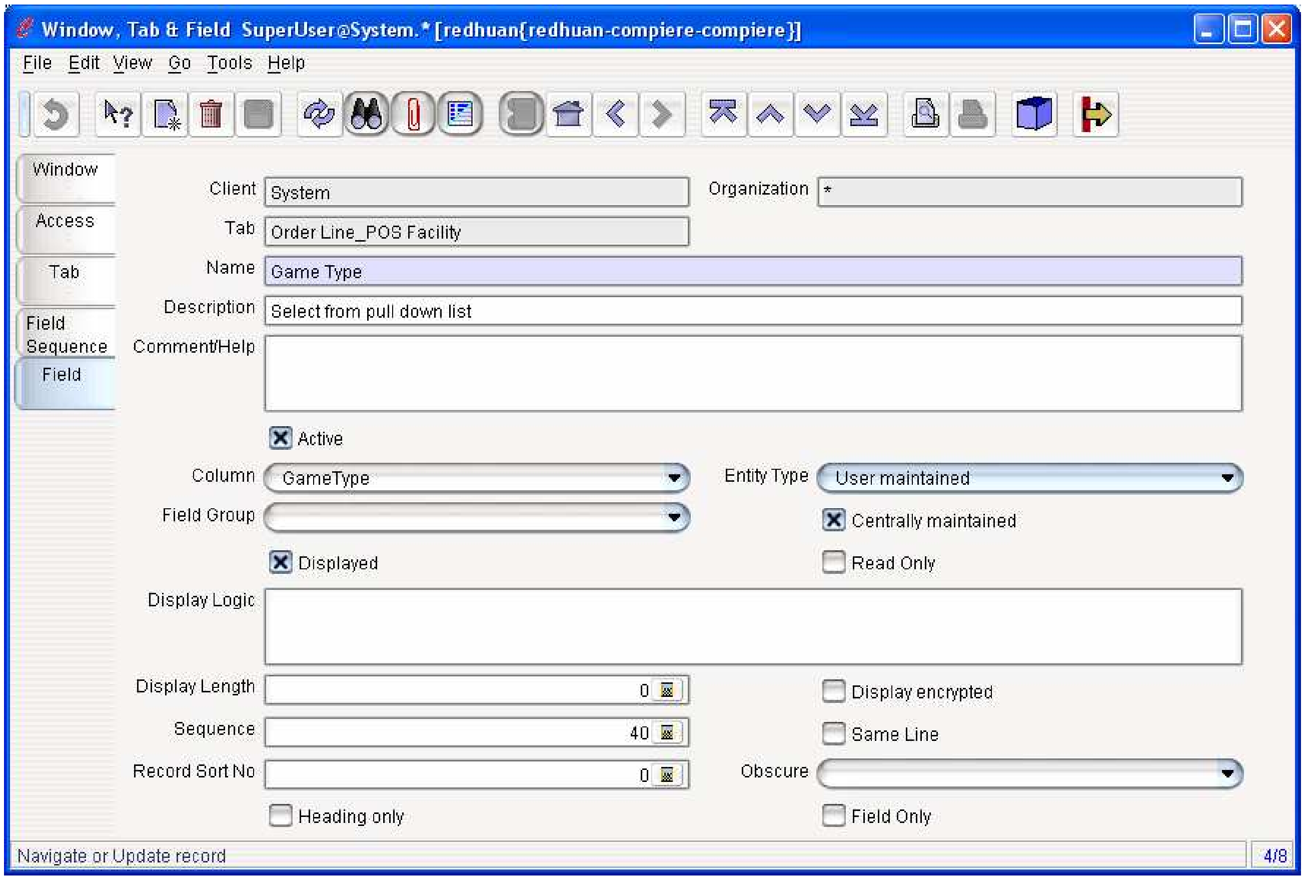
Give the *DB Column Name* and *Name* to be the same so as to avoid confusion, though it is not compulsory. You need not create a System Element, again not compulsory. We just want to proof the concept and learn quickly without getting mixed up. Next you pull down *Reference* to select List, and pull the Reference Key as GameType. In the end, remember to synchronise it to your Oracle database, by clicking on the button at the bottom (can't see on the above screen, have to scroll down) The process must complete succesfully.



Next we will introduce this new field into a Window. Its going to be POS Facility (Sales Order). Otherwise it won't appear when you run your GardenAdmin and launch the Sales Order screen. (Expert note: From our previous work when we created a POS window, we copied to POS Facility to reuse similar settings and avoid redoing it again in the AD Window).

Configuring AD Window

Open the Window, Tab & Field screen. Create a new record. Call it something, in my case, POS Facility. Copy Window Tabs from the Sales Order. Go to the 2nd tab, which is the orderline. Then go to the Field tab. Create a new record.


Window, Tab & Field SuperUser@System.* [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Window
Access
Tab
Field Sequence
Field
Navigate or Update record
4/8

Client	System	Organization	*
Tab	Order Line_POS Facility		
Name	Game Type		
Description	Select from pull down list		
Comment/Help			
[✓] Active			
Column	GameType	Entity Type	User maintained
Field Group		[✓] Centrally maintained	
[✓] Displayed		[ ] Read Only	
Display Logic			
Display Length	0	[ ] Display encrypted	
Sequence	40	[ ] Same Line	
Record Sort No	0	Obscure	
[ ] Heading only		[ ] Field Only	

Figure 10 New window, new Field – Game Type

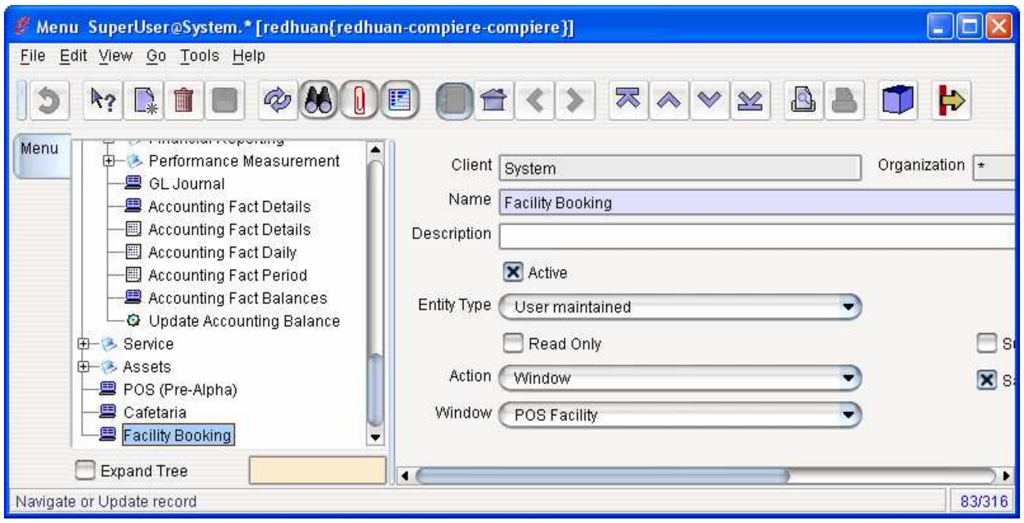
Give it a proper name as this will appear in the actual window. Under column is where you select from the field we created earlier in Table and Column. Make sure the *Read Only* box is NOT checked otherwise the field will appear inanimate and won't give a pull down menu list. You can check the *Same Line* box if you want that Game Type to be next to another field.

You can then go to *Field Sequence* to arrange the display order you want your field to appear. For my client, I even took out a lot of other unnecessary fields.

You have to Run another instance of Compiere, login as GardenAdmin (or whatever new client user ID you are working from), and view your changes. Oh! You still cannot view anything until you attach it to the AD Menu. So a bit more work now.

Configuring AD Menu

Under the General Rules, System Rules, you click on Menu to call up the Menu configuration screen.


Menu SuperUser@System.* [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Menu
Expand Tree
Navigate or Update record
83/316

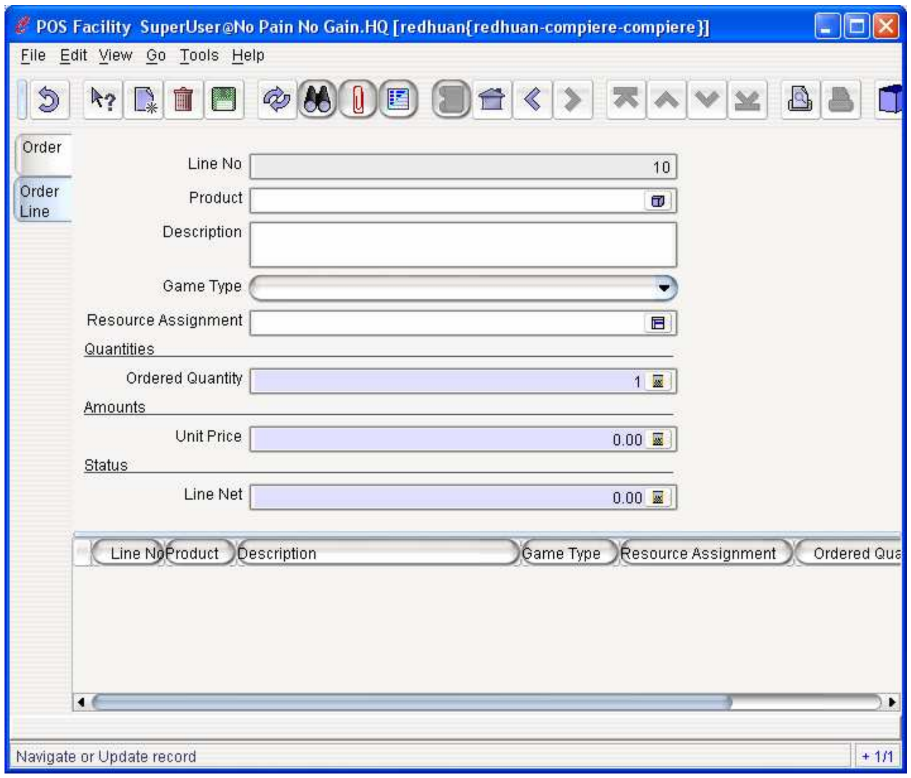
Performance Measurement
GL Journal
Accounting Fact Details
Accounting Fact Details
Accounting Fact Daily
Accounting Fact Period
Accounting Fact Balances
Update Accounting Balance
Service
Assets
POS (Pre-Alpha)
Cafetaria
Facility Booking

Client: System
Organization: *
Name: Facility Booking
Description:
Active: ☒
Entity Type: User maintained
Read Only: ☐
Action: Window
Window: POS Facility

Figure 11 Creating a new menu

Choose your action to be Window, so that you can then select the window you have in mind, which in my case is POS Facility. Now, that you have done this menu job, you can login as a User and view your new screen.

Now you can go to and fro between the System and Admin mode to tune your changes.


POS Facility SuperUser@No Pain No Gain.HQ [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Order
Order Line
Quantities
Amounts
Status
Line No
Product
Description
Game Type
Resource Assignment
Ordered Qua
Navigate or Update record
+ 1/1

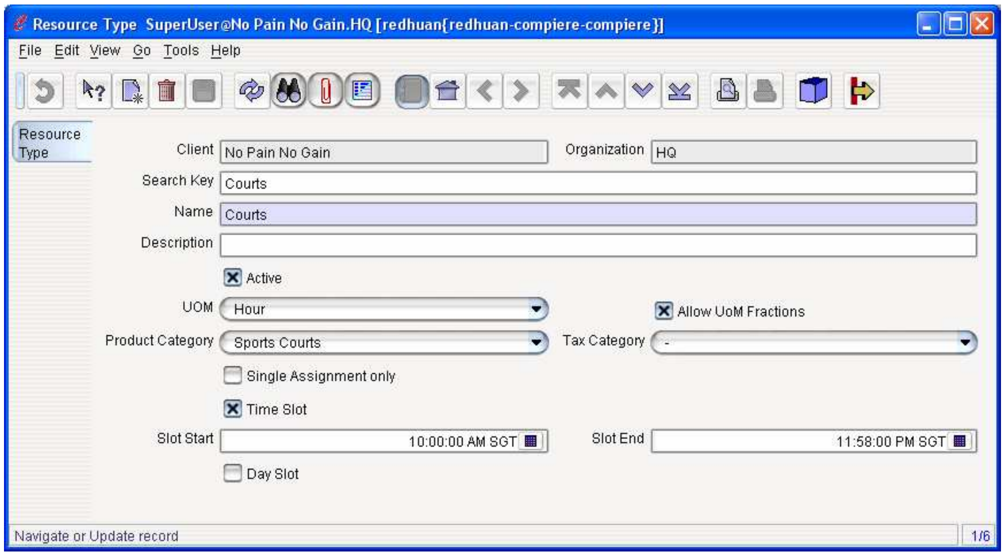
Line No: 10
Product:
Description:
Game Type:
Resource Assignment:
Ordered Quantity: 1
Unit Price: 0.00
Line Net: 0.00

From here, we shall be ready for the Resource Assignment field and amend the callout to copy the name and phone number of the customer into the Description field, and the Info Schedule as well. To understand it, you click on the box next to the field. A calendar-like schedule will pop-up and you can click anywhere on it to select the time and court that you want. Now you see the point of redundancy, where the user has to key in again the details at the schedule box. With our callout customisation, we won't need to. I see light at the end, Kimisabe!

Figure 12 A Window is born.

Configuring Resource Product

We seem to be jumping round a bit, so hope you don't get lost. We have to set up a court first to have it appear for selection at the *Info Schedule* screen. It involves creating a *Resource Type* and *Product*. So in your GardenAdmin menu, call up *Resource Type*. We shall create a new type and call it Courts, and then create a new product called Court A. We should also create a *Product Category* called Sports Courts for report grouping purposes. Use the screen below for guide.

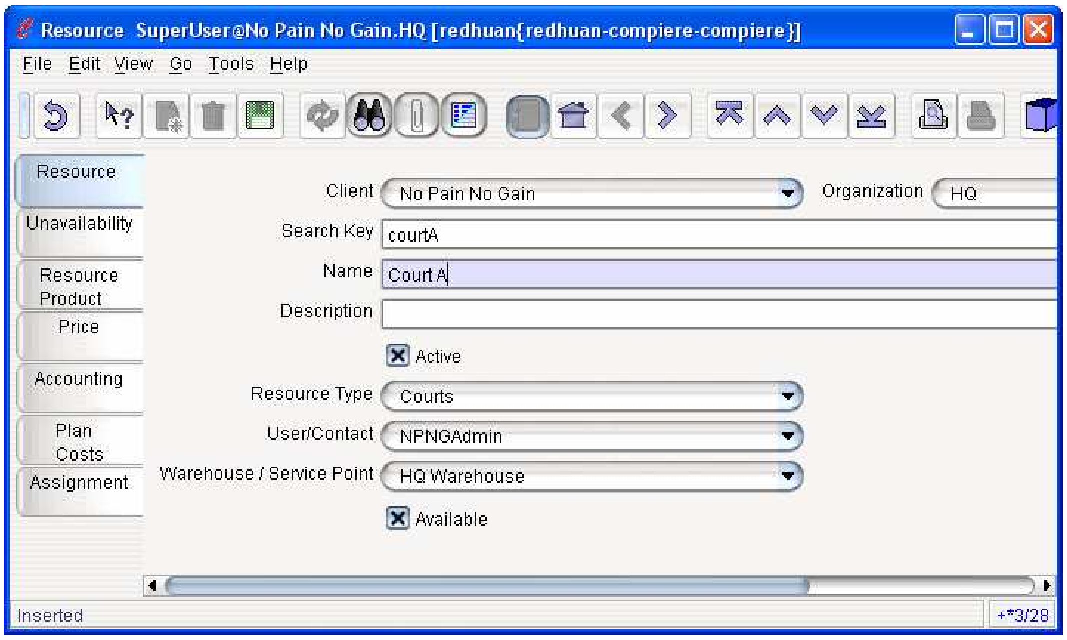

Resource Type SuperUser@No Pain No Gain.HQ [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Resource Type

Client: No Pain No Gain
Organization: HQ
Search Key: Courts
Name: Courts
Description:
[✓] Active
UOM: Hour
[✓] Allow UoM Fractions
Product Category: Sports Courts
Tax Category: -
[ ] Single Assignment only
[✓] Time Slot
Slot Start: 10:00:00 AM SGT
Slot End: 11:58:00 PM SGT
[ ] Day Slot
Navigate or Update record
1/6

Do not set the *Slot End* time to 11.58 as I did. It was to work-around a bug I described in a previous article. You can end your court time at 11pm, just for this exercise. After creating this, we call up the *Resource* screen.

Figure 13 Resource Type

Set your *Resource Type*. Inspect the *R. Product* tab. See the fields are properly set. Then go to the *Price* tab. Select a Price List and put in a Price. For our demonstration purpose, you can put $100 in all 3 price fields.

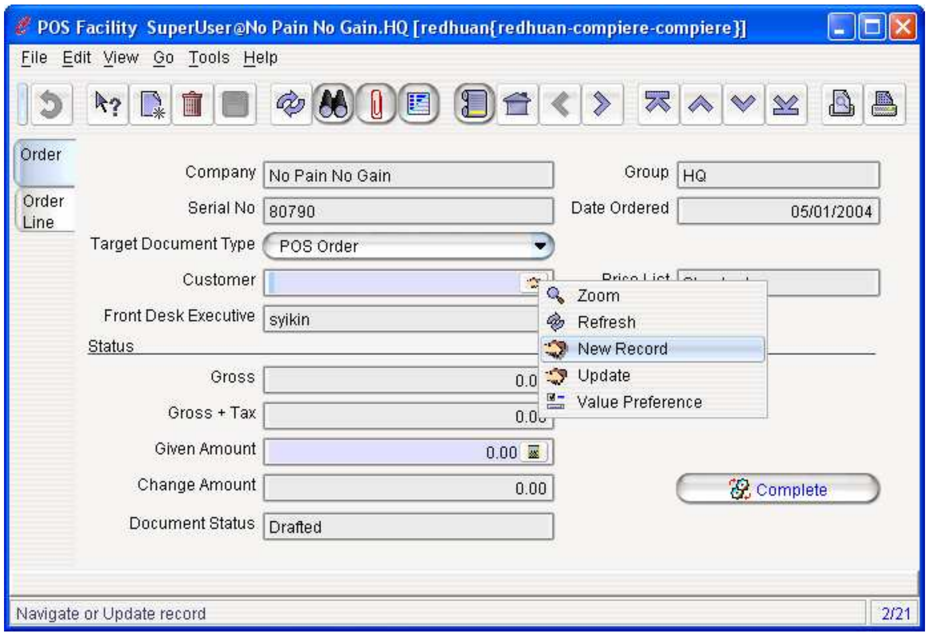

Resource SuperUser@No Pain No Gain.HQ [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Resource
Unavailability
Resource Product
Price
Accounting
Plan Costs
Assignment

Client: No Pain No Gain
Organization: HQ
Search Key: courtA
Name: Court A
Description:
[✓] Active
Resource Type: Courts
User/Contact: NPNGAdmin
Warehouse / Service Point: HQ Warehouse
[✓] Available
Inserted
+*3/28

Figure 14 Resource screen, R. Product tab, Price tab.

First Run

Now you can go and use the Facility screen, at the main tab, let's say that a new customer call up so you right-click at the Customer field (Business Partner). It will pop up a window for you. Select New Record. Key in the caller details, at least put in the key, name, location and phone.


POS Facility SuperUser@No Pain No Gain.HQ [redhuan{redhuan-compiere-compiere}]
File Edit View Go Tools Help
Order
Order Line
Zoom
Refresh
New Record
Update
Value Preference
Complete

Company: No Pain No Gain
Group: HQ
Serial No: 80790
Date Ordered: 05/01/2004
Target Document Type: POS Order
Customer:
Price List:
Front Desk Executive: syikin
Status
Gross: 0.0
Gross + Tax: 0.00
Given Amount: 0.00
Change Amount: 0.00
Document Status: Drafted

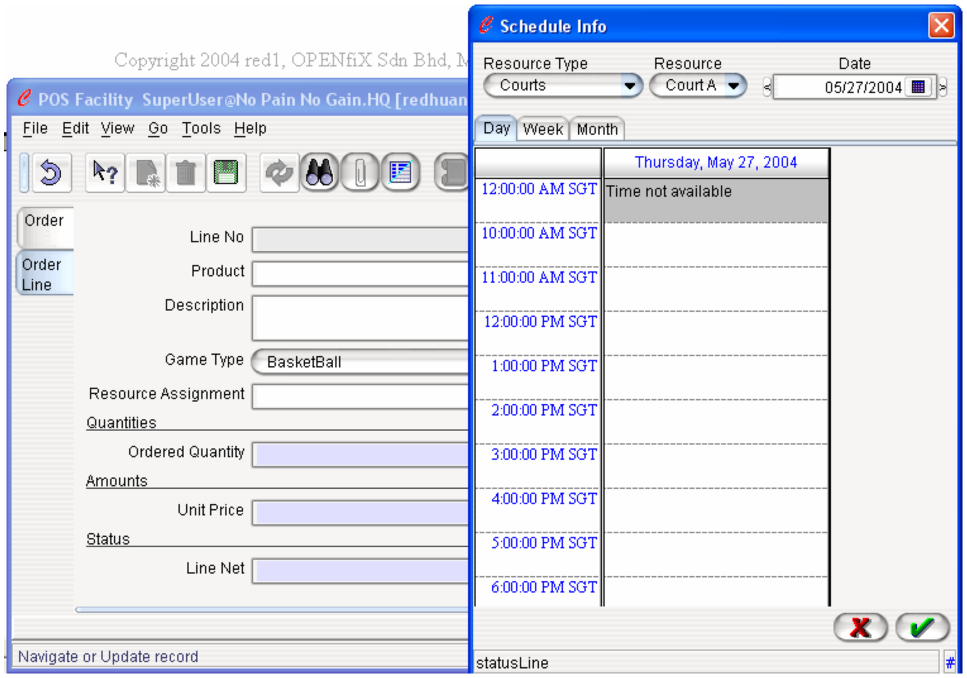
Navigate or Update record 2/21

Figure 15 Main order screen, caller calls and user request contact details

Notice that I have changed my Sales Rep to Front Desk Executive. That change is done in the AD Window. 'syikin' is the login user. That is defaulted by setting @#CreatedBy@ in the AD Column. It is also made to be read-only so that 'syikin' cannot pretend to be another user.

Go to the Order Line tab, ah yes! You can now pull down the Game Type and choose! Lovely isn't it? Then click on the Resource Assignment and out pops the Schedule Info, as shown below.

Figure 16 Game Type and Info Schedule


Copyright 2004 red1, OPENfiX Sdn Bhd, M
POS Facility SuperUser@No Pain No Gain.HQ [redhuan
File Edit View Go Tools Help
Order
Order Line
Schedule Info
Day Week Month

Line No
Product
Description
Game Type: BasketBall
Resource Assignment
Quantities
Ordered Quantity
Amounts
Unit Price
Status
Line Net

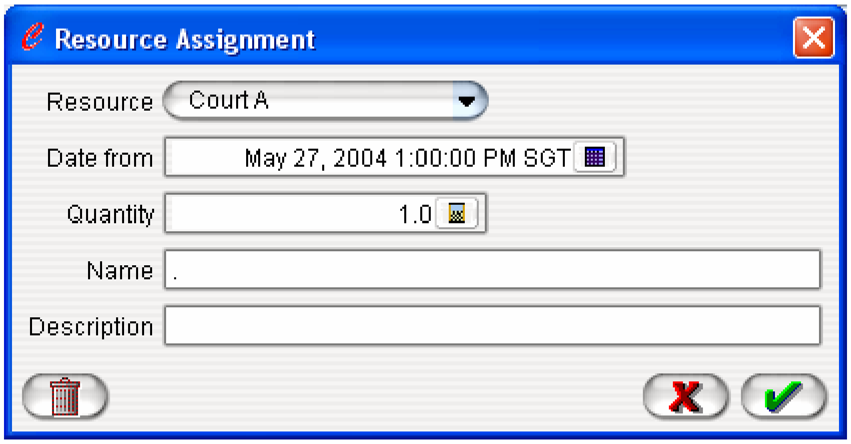
Navigate or Update record

Resource Type: Courts
Resource: Court A
Date: 05/27/2004

	Thursday, May 27, 2004
12:00:00 AM SGT	Time not available
10:00:00 AM SGT	
11:00:00 AM SGT	
12:00:00 PM SGT	
1:00:00 PM SGT	
2:00:00 PM SGT	
3:00:00 PM SGT	
4:00:00 PM SGT	
5:00:00 PM SGT	
6:00:00 PM SGT	

statusLine

Click on any time slot in the schedule and you see that another window pops up expecting you to rekey in some details. Notice that there is a dot on the Name field.



Resource	Court A
Date from	May 27, 2004 1:00:00 PM SGT
Quantity	1.0
Name	.
Description	

Figure 17 Resource Assignment Pop-up

Just key-in something after it, let's say the name of the person doing the booking. In the Description field key-in the contact telephone number. You can see this is annoying as you won't have the earlier screen showing you those details, and you already did that a while ago in the New Record screen. That means you always need to scribble those details somewhere while holding the phone and typing at the same time. (I just want you to feel the pain of necessity to invent here).

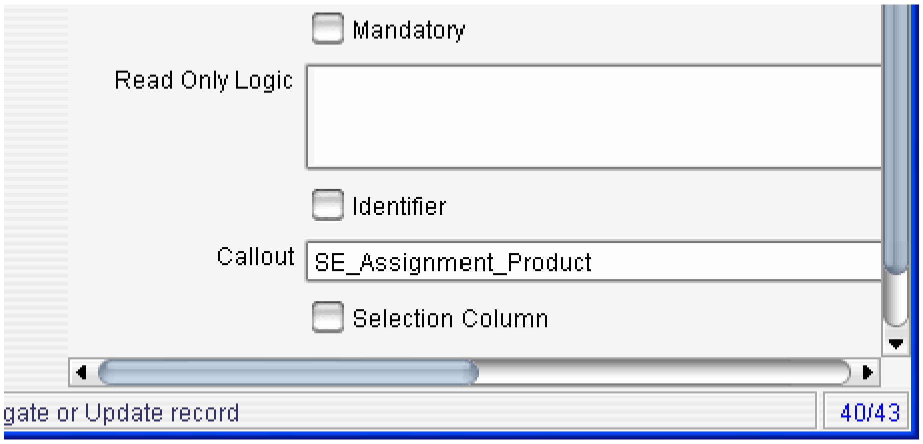
After accepting the input, we are returned to the Order Line screen which has its Product and the Description fields populated. Notice that the phone number is put in brackets. Who did all this?

Now here is the good reason who. Go to the Table and Column of System and find out the Resource Assignment column of the C_Orderline table. At the bottom, in the Callout section you will see that a callout is stated there, by the name of SE_Assignment_Product. (see figure below).

Ahah! So there is already a callout before this in the Resource Assignment field! Remembering my earlier point about noticing patterns, we will reuse that same callout to solve our problem. Its just plain lucky to have that pattern in place. But the time taken for me to find that out was long and dreary. I am giving you the beniefit of hindsight. If I leave you to figure out you will probably go through the same journey *chuckles*.

Now we have to make the pop-up window return without keying in any details but having the name and phone number appearing later in the schedule by itself.

Let's return to the Eclipse to have our unadulterated fun!



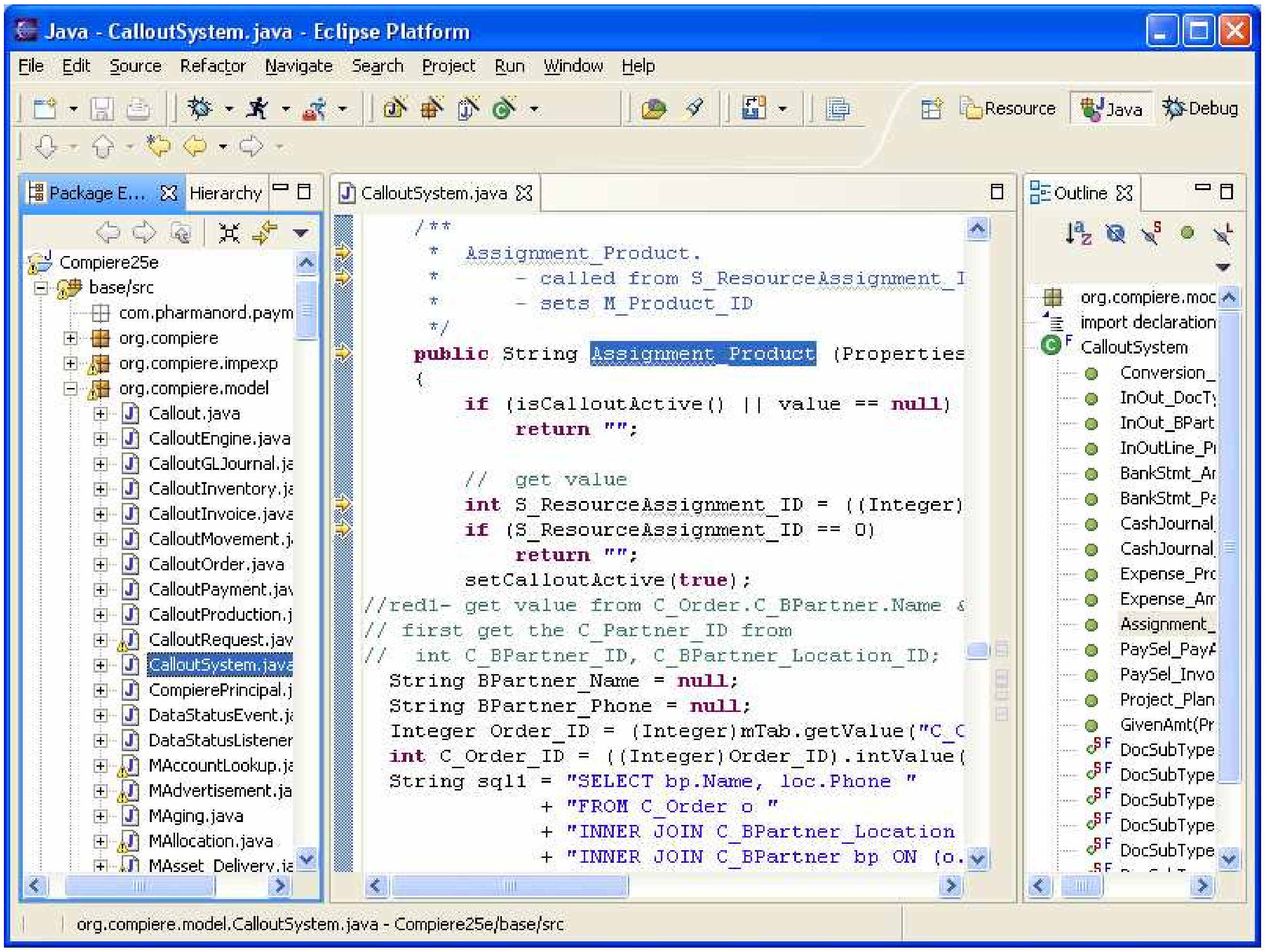
Callout: SE_Assignment_Product

Figure 18 Callout reference

Customising Callout

Eclipse in Action

In Eclipse, we will now locate the Callout. Previously I didn't have any clue as to where in the world the Callout was kept. Is it in the database, a procedure, or a code? I sniff it out by puncturing a callout – disfiguring it so that an error happens. From the debug log, I notice that it's a program. Finally I found it in the package org.compiere.model. So use your mouse and go to the Source project tree. Click on the base folder, and expand to the model package. Double click on CalloutSystem.java. You will notice the Assignment Product callout. Click on that too.


Java - CalloutSystem.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Resource
Java
Debug
Package E...
Hierarchy
CalloutSystem.java
Outline

```
    /**
     *  Assignment_Product.
     *      - called from S_ResourceAssignment_I
     *      - sets M_Product_ID
     */
    public String Assignment_Product (Properties
    {
        if (isCalloutActive() || value == null)
            return "";

        //  get value
        int S_ResourceAssignment_ID = ((Integer)
        if (S_ResourceAssignment_ID == 0)
            return "";
        setCalloutActive(true);
//red1- get value from C_Order.C_BPartner.Name &
// first get the C_Partner_ID from
//  int C_BPartner_ID, C_BPartner_Location_ID;
  String BPartner_Name = null;
  String BPartner_Phone = null;
  Integer Order_ID = (Integer)mTab.getValue("C_C
  int C_Order_ID = ((Integer)Order_ID).intValue(
  String sql1 = "SELECT bp.Name, loc.Phone "
              + "FROM C_Order o "
              + "INNER JOIN C_BPartner_Location
              + "INNER JOIN C_BPartner bp ON (o.
```

```
Compiere25e
  base/src
    com.pharmanord.paym
    org.compiere
    org.compiere.impexp
    org.compiere.model
      Callout.java
      CalloutEngine.java
      CalloutGLJournal.ja
      CalloutInventory.ja
      CalloutInvoice.java
      CalloutMovement.j
      CalloutOrder.java
      CalloutPayment.jav
      CalloutProduction.j
      CalloutRequest.jav
      CalloutSystem.java
      CompierePrincipal.j
      DataStatusEvent.ja
      DataStatusListener
      MAccountLookup.ja
      MAdvertisement.ja
      MAging.java
      MAllocation.java
      MAsset_Delivery.ja
```

```
org.compiere.moc
import declaration
CalloutSystem
  Conversion_
  InOut_DocTy
  InOut_BPart
  InOutLine_Pi
  BankStmt_Ar
  BankStmt_Pa
  CashJournal
  CashJournal
  Expense_Pro
  Expense_Am
  Assignment_
  PaySel_PayA
  PaySel_Invo
  Project_Plan
  GivenAmt(Pr
  DocSubType
  DocSubType
  DocSubType
  DocSubType
  DocSubType
```

org.compiere.model.CalloutSystem.java - Compiere25e/base/src

Figure 19 Getting to the Java and Method

Note that the SE_ is somehow used, must have been some parsing going on within Compiere, but that's not relevant here. What's important is we have tracked down that callout and are ready to do some high-rise construction.

Let’s examine the code in its original entirety here. See if we can get some explanations going.

```
public String Assignment_Product (Properties ctx, int WindowNo, MTab mTab, MField mField, Object value)
    {
        if (isCalloutActive() || value == null)
        return "";
    //  get value
    int S_ResourceAssignment_ID = ((Integer)value).intValue();
        if (S_ResourceAssignment_ID == 0)
            return "";
        setCalloutActive(true);

        int M_Product_ID = 0;
        String Name = null;
        String Description = null;
        BigDecimal Qty = null;
        String sql = "SELECT p.M_Product_ID, ra.Name, ra.Description, ra.Qty "
            + "FROM S_ResourceAssignment ra"
            + " INNER JOIN M_Product p ON (p.S_Resource_ID=ra.S_Resource_ID) "
            + "WHERE ra.S_ResourceAssignment_ID=?";
        try
        {
            PreparedStatement pstmt = DB.prepareStatement(sql);
            pstmt.setInt(1, S_ResourceAssignment_ID);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next())
            {
                M_Product_ID = rs.getInt (1);
                Name = rs.getString(2);
                Description = rs.getString(3);
                Qty = rs.getBigDecimal(4);
            }
            rs.close();
            pstmt.close();
        }
        catch (SQLException e)
        {
            log.error("Assignment_Product", e);
        }

        log.debug("S_ResourceAssignment_ID=" + S_ResourceAssignment_ID + " - M_Product_ID=" + M_Product_ID);
        if (M_Product_ID != 0)
        {
            mTab.setValue ("M_Product_ID", new Integer (M_Product_ID));
            if (Description != null)
                Name += " (" + Description + ")";
            if (!".".equals(Name))
                mTab.setValue("Description", Name);
            //
            String variable = "Qty";
            if (mTab.getTableName().startsWith("C_Order"))
                variable = "QtyOrdered";
            else if (mTab.getTableName().startsWith("C_Invoice"))
                variable = "QtyInvoiced";
            if (Qty != null)
                mTab.setValue(variable, Qty);
        }
        setCalloutActive(false);
        return "";
    }   //  Assignment_Product
```

What we have is not the whole Java class, but part of it, in fact it concerns only the Asignment_Product Method. It begins with certain arguments:

```
(Properties ctx, int WindowNo, MTab mTab, MField mField, Object value)
```

the WindowNo will inform the system which window is been refered to. This we can understand as when the callout happens, we were in a window screen. So when the callout finishes its job, the result update will appear in the same window.

MTab concerns the table that is in focus. To let you know how I know, you can hover your mouse arrow over any word and see the highlights. If you press the Ctrl key while you hover over them, and click, you may really dial in – to the class that handles the objects. You can explore further by opening the Parent class that it extends from such as the CalloutEngine.java, MTab.java among others. Won't want to rob you of the fun. Just going through the associate classes and reading the Javadocs can be a journey of self-discovery.

Let's turn our attention to the SQL statement on line 15 onwards. Its pulling out from the Resource Assignment table (line 16). Remembering that there was a pop-up window that allow us to key in the *Name* and *Description*. And not forgetting the *Name* field has a dot in it. So here it is retrieving from what that pop-up (line 26 to 29) and populate the OrderLine Description with it (line 42 to 54). It will also wrap the *R.A. Description* with brackets before presenting (line 44).

Notice also the pattern format – `mTab.getTableName().startsWith("C_Order")`

It seems to check which table the window is assoicated with. Hmm, very useful. Now go and hover over the getTableName, dial in if you can, and off you go again.

To shift into top gear, you can right click at the left edge of the editor panel and toggle a breakpoint. Then run in debug mode. When you access the Schedule Info pop-up and closes it, the program will stop at the breakpoint and you can step through slowly, while examining the program flow weaving in and out of other codes. At the same time the top right panel will display variables that are in focus, and show what their values are. This gives you a surgical path through the mesh of codes, slowly building your understanding of how things work as you go along. This at least is how I started about 5 months ago.

It can be like a cyber roller coaster ride, where you can toy for hours on end. I encourage it for that's the a really good way to jump into programming. For all the classes we skip (I mean real programming courses), what choice we got?

We can then get on to putting in our own codes.

Modifying CalloutSystem

First you have to plan out what exactly you want to do, detailed it further in database terms, and then write out what you plan.

- Access the Customer records to get the Name and Phone Number.
  - Find from the parent record, C_Order for the C_BPartner_ID
  - Lookup C_BPartner_Location for the Phone Number
- Updating the Orderline's Description.
  - Pull miscellaneous info from Resource Assignment
  - Append with Customer Name and Phone Number
  - Append with Game Type
  - Set Orderline Description field with this info
- Putting new info into Resource Assignment table.
  - Form the same info into Name and Description fields
  - Write back to Database

So as not to lost each other, I will dump the complete coding to achieve that here.Take a look at it and see if you can corelate to what we planned. Notice that we can still put in something in the pop-up and it will get appended to in the OrderLine and Resource Assignment records.

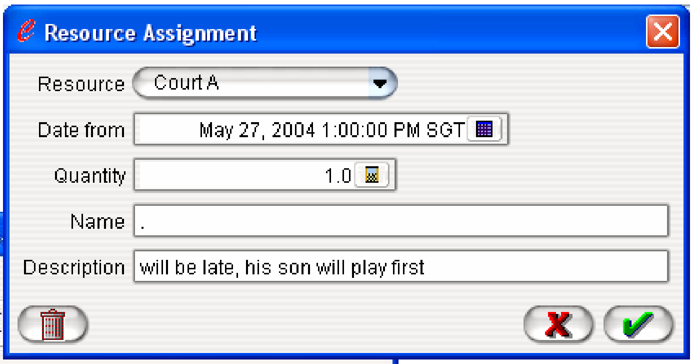
```
//red1- get value from C_Order.C_BPartner.Name & Location>Description, pass in here
// first get the C_Partner_ID from
//  int C_BPartner_ID, C_BPartner_Location_ID;
  String BPartner_Name = null;
  String BPartner_Phone = null;
  Integer Order_ID = (Integer)mTab.getValue("C_Order_ID");
  int C_Order_ID = ((Integer)Order_ID).intValue();
  String sql1 = "SELECT bp.Name, loc.Phone "
                      + "FROM C_Order o "
                      + "INNER JOIN C_BPartner_Location loc ON
(o.C_BPartner_Location_ID=loc.C_BPartner_Location_ID)"
                      + "INNER JOIN C_BPartner bp ON (o.C_BPartner_ID=bp.C_BPartner_ID)"
                      + "WHERE o.C_Order_ID=?";
               try
               {
                      PreparedStatement pstmt = DB.prepareStatement(sql1);
                      pstmt.setInt(1,C_Order_ID);
                      ResultSet rs = pstmt.executeQuery();
                      if (rs.next())
                      {
                           BPartner_Name = rs.getString(1);
                           BPartner_Phone = rs.getString(2);
                      }
                      rs.close();
                      pstmt.close();
               }
               catch (SQLException e)
               {
```

```
                        log.error("Assignment_Product - getting C_Order", e);
                }


//red1
                int M_Product_ID = 0;
                String Name = null;
                String Description = null;
                BigDecimal Qty = null;
                String sql = "SELECT p.M_Product_ID, ra.Name, ra.Description, ra.Qty "
                        + "FROM S_ResourceAssignment ra"
                        + " INNER JOIN M_Product p ON (p.S_Resource_ID=ra.S_Resource_ID) "
                        + "WHERE ra.S_ResourceAssignment_ID=?";
                try
                {
                        PreparedStatement pstmt = DB.prepareStatement(sql);
                        pstmt.setInt(1, S_ResourceAssignment_ID);
                        ResultSet rs = pstmt.executeQuery();
                        if (rs.next())
                        {
                                M_Product_ID = rs.getInt (1);
                                Name = rs.getString(2); //red1 - will append with BPartner
name/tel
                                Description = rs.getString(3); //red1 - will append to above
                                Qty = rs.getBigDecimal(4);
                        }
                        rs.close();
                        pstmt.close();
                }
                catch (SQLException e)
                {
                        log.error("Assignment_Product", e);
                }

                log.debug("S_ResourceAssignment_ID=" + S_ResourceAssignment_ID + " - 
M_Product_ID=" + M_Product_ID);
                if (M_Product_ID != 0)
                {
                        String GameType = (String)mTab.getValue("GameType");
                        mTab.setValue ("M_Product_ID", new Integer (M_Product_ID));
                        Name += BPartner_Name + ", tel:" + BPartner_Phone + " - " + GameType;
//red1 - put info now
                        if (Description != null)
                                Name += " (" + Description + ")";
//                      if (!".".equals(Name))   red1 - taken out as BPartner info populates
it
                                mTab.setValue("Description", Name);
                        //
                        String variable = "Qty";
                        if (mTab.getTableName().startsWith("C_Order"))
                                variable = "QtyOrdered";
                        else if (mTab.getTableName().startsWith("C_Invoice"))
                                variable = "QtyInvoiced";
                        if (Qty != null)
                                mTab.setValue(variable, Qty);

// red1 - UPDATE ResourceAssignment RECORD with Name, Description (telno, GameType)
                        Description=BPartner_Phone+" - "+GameType; //red1 - previously left
blank
                        String sql2 = "UPDATE S_ResourceAssignment ra SET"
                                    + " Name=" + DB.TO_STRING(BPartner_Name)+ ","
                                    + " Description=" + DB.TO_STRING(Description)
```

```
                                              + " WHERE S_ResourceAssignment_ID=" +
S_ResourceAssignment_ID;
             int i = DB.executeUpdate(sql2.toString());
             log.debug("S_ResourceAssignment_I - Updated if 1 => " + i);
             }
             setCalloutActive(false);
             return "";
      }      //     Assignment_Product
```

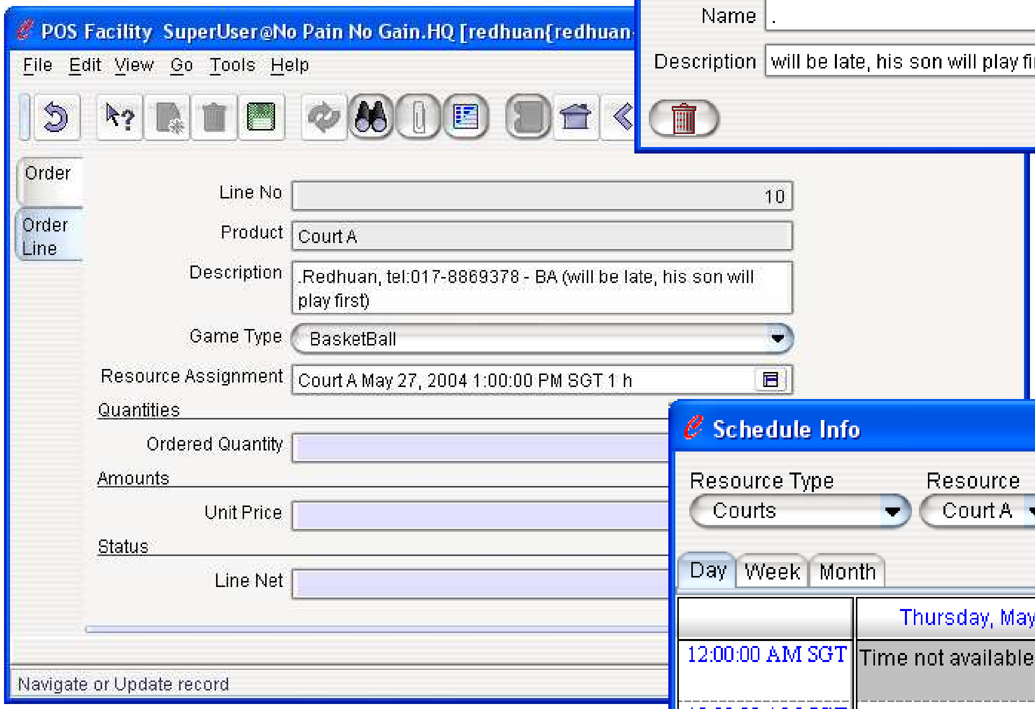
After this you can test the result and you should achieve the screens shown below. Just type in ‘will be late, his son will play first’ in the Description, and hit enter. That’s all!

Figure set 20 Callout power!



Resource Assignment

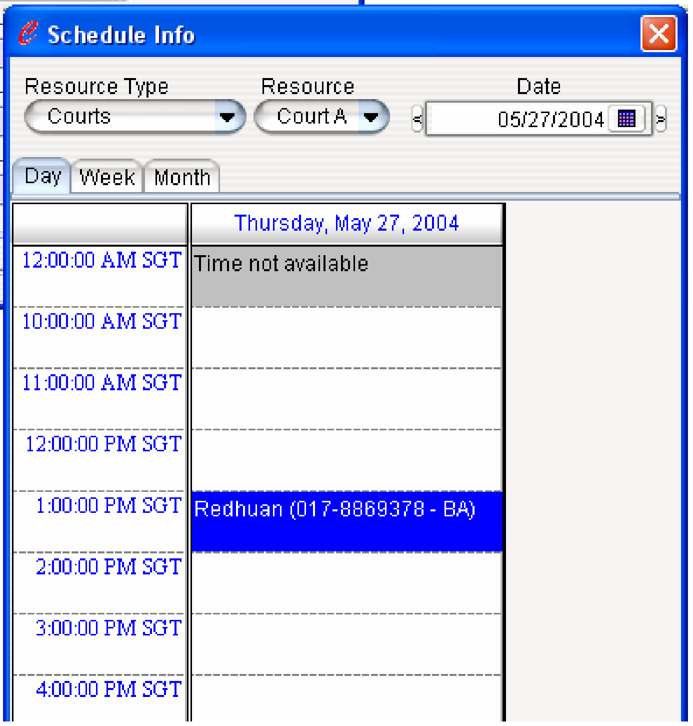
Resource	Court A
Date from	May 27, 2004 1:00:00 PM SGT
Quantity	1.0
Name	.
Description	will be late, his son will play first



POS Facility SuperUser@No Pain No Gain.HQ [redhuan{redhuan
File Edit View Go Tools Help
Order
Order Line
Quantities
Amounts
Status

Line No	10
Product	Court A
Description	.Redhuan, tel:017-8869378 - BA (will be late, his son will play first)
Game Type	BasketBall
Resource Assignment	Court A May 27, 2004 1:00:00 PM SGT 1 h
Ordered Quantity	
Unit Price	
Line Net	

Navigate or Update record



Schedule Info

Resource Type	Resource	Date
Courts	Court A	05/27/2004

	Thursday, May 27, 2004
12:00:00 AM SGT	Time not available
10:00:00 AM SGT	
11:00:00 AM SGT	
12:00:00 PM SGT	
1:00:00 PM SGT	Redhuan (017-8869378 - BA)
2:00:00 PM SGT	
3:00:00 PM SGT	
4:00:00 PM SGT	

Go to the top menu bar and pull down the View menu and select Schedule Info. Select your Court A, and you shall see the fruits of the amended callout.

I know, its ok, happened to me too, here, take some kleenex.

Closing

There are about 2 more tips that I am brushing through. First is on the 11.58pm array slot handling. It took me like 3 days non-stop to track down what was the cause and when I did, it took me another 2 days to remove it. Then I switch to the Weekly display mode, and the problem reemerges! Another period of 1 day to plug up. But that workaround is only short term, and its bad for a developer to lock-in such a code. OBeOne, don’t go to the Dark Side. So for the record I append the mischievous VSchedulePanel at the bottom. My inputs are coloured ‘red1’. You can put some breaks into it and debug it to see the problem. Perhaps you can give it the real solution.

The other is a simple yet powerful search function you can activate when we click on the Search (binocs) button from the top bar menu. By sepecifying the Business Partner field as Searchable in the AD, you can have that appearing with the % symbol for you to zoom in on the Customer who has made the booking. All that makes your app super-fast!

Do drop me a note whether it be to share a particular problem, or showing me what you have learn. Let’s share our power. The world is big enough. Peace be upon all.

Appendix – VSchedulePanel.java amended

```
package org.compiere.apps.search;

import javax.swing.*;
import java.awt.*;
import java.awt.font.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.sql.*;
import java.math.*;

import org.compiere.model.*;
import org.compiere.util.*;
import org.compiere.plaf.*;
import org.compiere.grid.ed.*;

/**
 *      Schedule Panel
 *
 *      @author         Jorg Janke
 *      @version        $Id: VSchedulePanel.java,v 1.9 2003/09/29 01:04:41 jjanke Exp $
 */
public class VSchedulePanel extends JComponent implements MouseListener
{
        /**
         *      Constructor
         */
        public VSchedulePanel ()
        {
                setHeight(250);
                addMouseListener(this);
        }       //      VSchedulePanel


        /**     Number of Days                  */
        private int                                     m_noDays = 1;
        /** Height                                      */
        private int                                     m_height = 250;
//red1 noted dimensions set here
        /**     Day Slot Width                  */
        private int                                     m_dayWidth = 170;

        /** TimePanel for layout info */
        private VScheduleTimePanel    m_timePanel = null;

        /** Assignment Slots            */
        private MAssignmentSlot[]     m_slots = null;
        /** Position of Slots           */
        private Rectangle[]                     m_where = null;
        /**     Start Date                              */
        private Timestamp                       m_startDate = null;
        /** If true creates new assignments     */
        private boolean                         m_createNew = false;
        /**     Resource ID                             */
        private int                                     m_S_Resource_ID = 0;

        private InfoSchedule          m_infoSchedule = null;

        /** Text Margin                                 */
        private static final int      MARGIN = 2;

        /**
         *      Set Type.
         *  Calculate number of days and set
         *      @param type schedule type - see VSchedule.TYPE_...
         */
        public void setType (int type)
```

```
{
        if (type == VSchedule.TYPE_MONTH)
                m_noDays = 31;
        else if (type == VSchedule.TYPE_WEEK)
                m_noDays = 7;
        else
                m_noDays = 1;
        setSize();
}       //      setType

/**
 *      Set InfoSchedule for callback
 *      @param is InfoSchedule
 */
public void setInfoSchedule (InfoSchedule is)
{
        m_infoSchedule = is;
}

/**
 *      Enable/disable to Create New Assignments
 *      @param createNew if true, allows to create new Assignments
 */
public void setCreateNew (boolean createNew)
{
        m_createNew = createNew;
}       //      setCreateNew

/**
 *      From height, Calculate & Set Size
 *  @param height height in pixels
 */
public void setHeight (int height)
{
        m_height = height;
        setSize();
}       //      setHeight

/**
 *      Set Size
 */
public void setSize ()
{
        //      width
        FontMetrics fm = null;
        Graphics g = getGraphics();
        if (g == null)
                g = Env.getGraphics(this);
        if (g != null)
                fm = g.getFontMetrics(g.getFont());          //      the "correct" way
        m_dayWidth = 0;
        for (int i = 0; i < m_noDays; i++)
        {
                if (fm != null)
                {
                        String string = getHeading(i);
                        int width = fm.stringWidth(string);
                        if (width > m_dayWidth)
                                m_dayWidth = width;
                }
        }
        m_dayWidth += 20;
        if (m_dayWidth < 180)  //      minimum width
                m_dayWidth = 180;

        int w = m_noDays * m_dayWidth;
        //
        Dimension size = new Dimension(w, m_height);
        setPreferredSize(size);
        setMinimumSize(size);
        setMaximumSize(size);
}       //      setHeight
```

```
	/**
	 *	Set time Panel for info about tile slots
	 *	@param timePanel time panel
	 */
	public void setTimePanel (VScheduleTimePanel timePanel)
	{
		m_timePanel = timePanel;
	}	//	setTimePanel

	/**************************************************************************/

	/**
	 *	Set Slots
	 *	@param mass Assignment Slots
	 *  @param S_Resource_ID resource
	 *  @param startDate start date
	 *  @param endDate end date
	 */
	public void setAssignmentSlots (MAssignmentSlot[] mass, int S_Resource_ID,
		Timestamp startDate, Timestamp endDate)
	{
		Log.trace(Log.l5_DData, "VSchedulePanel.setAssignmentSlots");
		m_S_Resource_ID = S_Resource_ID;
		m_noDays = TimeUtil.getDaysBetween (startDate, endDate);
		m_startDate = startDate;
		//
		m_slots = mass;
		m_where = new Rectangle[m_slots.length];
		//
		//	Calculate Assignments //red1 note last m_where rect is truncated as midnite is 
stated 11.58pm
		for (int i = 0; m_slots != null && i < m_slots.length; i++)
		{
			MAssignmentSlot mas = m_slots[i];
			int dayIndex = TimeUtil.getDaysBetween (m_startDate, mas.getStartTime());
			if (dayIndex < 0 || dayIndex >= m_noDays)
				System.out.println("Assignment " + i + " Invalid DateRange " + 
mas.getInfo());
			//
			int xWidth = m_dayWidth / mas.getXMax();
			int xStart = dayIndex * m_dayWidth;		//	start day slot
			xStart += mas.getXPos() * xWidth;		//	offset
			int xEnd = xStart + xWidth;

			int yStart = m_timePanel.getSlotYStart(mas.getYStart());
			int yEnd = m_timePanel.getSlotYEnd(mas.getYEnd());
		//	System.out.println("Assignment " + i + ", Xpos=" + mas.getXPos() + ", Xmax=" + 
mas.getXMax()
		//		+ ", Ystart=" + mas.getYStart() + ", Yend=" + mas.getYEnd() + " " + 
mas.getInfo());
			m_where[i] = new Rectangle(xStart+1, yStart+1, xWidth-1, yEnd-yStart-1);
		}	//	calculate assignments

		//
		setSize();
		repaint();
	}	//	setAssignmentSlots

	/**************************************************************************/

	/**
	 *	Paint it
	 *  @param g the <code>Graphics</code> object
	 */
	public void paint (Graphics g)
	{
	//	Log.trace(Log.l5_DData, "VSchedulePanel.paint", g.getClip());
		Graphics2D g2D = (Graphics2D)g;
		Dimension size = getPreferredSize();
		Rectangle clipBounds = g2D.getClipBounds();
		int w = size.width;
		int h = size.height;

		//	Paint Background
		g2D.setPaint(Color.white);
		g2D.fill3DRect(1, 1,  w-2, h-2, true);

		if (m_timePanel == null)	//	required
			return;
		int headerHeight = m_timePanel.getHeaderHeight();

		//	horizontal lines -
		g2D.setStroke(VScheduleTimePanel.getStroke(true));
		for (int i = 1; i < m_timePanel.getSlotCount(); i++)
		{
			g2D.setPaint(Color.gray);
			int yy = m_timePanel.getSlotYStart(i);
			g2D.drawLine(1, yy,  w-2, yy);	//	top horiz line
		}

		//	heading and right vertical lines |
		g2D.setStroke(VScheduleTimePanel.getStroke(false));
		for (int i = 0; i < m_noDays; i++)
		{
			Rectangle where = new Rectangle(i * m_dayWidth, 0, m_dayWidth, headerHeight);
			if (!where.intersects(clipBounds))
				continue;
			//	Header Background
			CompiereUtils.paint3Deffect(g2D, where, false, true);
			g2D.setPaint(Color.blue);
			TextLayout layout = new TextLayout (getHeading(i), g2D.getFont(),
g2D.getFontRenderContext());
			float hh = layout.getAscent() + layout.getDescent();
			layout.draw (g2D, where.x + (m_dayWidth - layout.getAdvance())/2,	//
	center
				((where.height - hh)/2) + layout.getAscent());		//
	center
			//	line
			g2D.setPaint(Color.black);
			int xEnd = (i+1) * m_dayWidth;
			g2D.drawLine(xEnd, 1,  xEnd, h-1);	//	right part
		}

		//	Paint Assignments //red1 note that upper/lower "Time not available"s are part
of this
//red1 'do not allow closing time thru "11:58:00 PM SGT" - to overwrite assignments

		for (int i = 0; m_slots != null && i < m_slots.length; i++)
//red1 put in "-1" to avoid last loop, previous solution, but cant work for week display. Remove “-1”
		{
			if (m_slots[i].toString().equals("11:58:00 PM SGT")) //red1 do not allow to
overwrite...
				continue;
			if (!m_where[i].intersects(clipBounds))
				continue;

			//	Background
			g2D.setColor(m_slots[i].getColor(true));
			g2D.fill(m_where[i]);
			//	Text
			String string = m_slots[i].getInfoNameDescription(); //red1
getInfoNameDescription
			AttributedString as = new AttributedString (string);
			as.addAttribute(TextAttribute.FONT, g2D.getFont());
			as.addAttribute(TextAttribute.FOREGROUND, m_slots[i].getColor(false));
			//	Italics for Description
			int startIt = string.indexOf('(');
			int endIt = string.lastIndexOf(')');
			if (startIt != -1 && endIt != -1)
				as.addAttribute(TextAttribute.POSTURE, TextAttribute.POSTURE_OBLIQUE,
startIt, endIt);
			//	Paint multiple lines if required
			AttributedCharacterIterator aci = as.getIterator();
```

```
                        LineBreakMeasurer measurer = new LineBreakMeasurer (aci,
g2D.getFontRenderContext());
                        float wrappingWidth = m_where[i].width - (2*MARGIN);
                        float curY = m_where[i].y + MARGIN;
                        TextLayout layout = null;
                        int yEnd = m_where[i].y + m_where[i].height;
                        while (measurer.getPosition() < aci.getEndIndex() && curY < yEnd)
                        {
                                layout = measurer.nextLayout(wrappingWidth);
                                curY += layout.getAscent();
                                if (curY < yEnd)
                                        layout.draw(g2D, m_where[i].x + MARGIN, curY);
                        }
                }       //      assignments

                //      Paint Borders
                g2D.setPaint(Color.black);
                g2D.setStroke(VScheduleTimePanel.getStroke(false));
                g2D.drawLine(1, 1,      1, h-1);        //      left
                g2D.drawLine(1, 1,              w-1, 1);        //      top
                //      heading line
                g2D.drawLine(1, headerHeight, w-1, headerHeight);
                //      Final lines
                g2D.setStroke(VScheduleTimePanel.getStroke(false));
                g2D.drawLine(w-1, 1,    w-1, h-1);      //      right
                g2D.drawLine(1, h-1,    w-1, h-1);      //      bottom line
        }       //      paint

        /**
         *      Return heading for index
         *      @param index index
         *      @return heading
         */
        private String getHeading (int index)
        {
                GregorianCalendar cal = new GregorianCalendar();
                if (m_startDate != null)
                        cal.setTime(m_startDate);
                cal.add(java.util.Calendar.DAY_OF_YEAR, index);
                //
                SimpleDateFormat format = (SimpleDateFormat)DateFormat.getDateInstance
                        (DateFormat.FULL, Language.getLanguage().getLocale());
                return format.format(cal.getTime());
        }       //      getHeading

        /**
         *      Mouse Clicked. Start AssignmentDialog
         *      @param e event
         */
        public void mouseClicked(MouseEvent e)
        {
                if (e.getClickCount() < 2)
                        return;

                Log.trace(Log.l3_Util, "VSchedulePanel.mouseClicked", e.getPoint());
                Rectangle hitRect = new Rectangle (e.getX()-1, e.getY()-1, 3, 3);

                //      Day
                int dayIndex = e.getX() / m_dayWidth;
                if (dayIndex >= m_noDays)
                        dayIndex = m_noDays-1;
        //      System.out.println("DayIndex=" + dayIndex + ": " + TimeUtil.addDays(m_startDate,
dayIndex));

                //      Time
                int timeIndex = m_timePanel.getTimeSlotIndex(e.getY());
        //      System.out.println("TimeIndex=" + timeIndex + ": " +
m_timePanel.getTimeSlot(timeIndex));

                //      check if there is an existing assignment//red1 overide 11.58pm
                for (int i = 0; i < m_slots.length; i++)
                {
                        if (m_where[i].intersects(hitRect))
                        if (!m_slots[i].toString().equals("11:58:00 PM SGT"))//red1 overiding...
                        {
                                MAssignmentSlot mas = m_slots[i];
                                System.out.println("Existing=" + mas.getInfo());
                                if (!mas.isAssignment())
        //
                                        return;
                                //
                                VAssignmentDialog vad = new VAssignmentDialog (Env.getFrame(this),
                                        m_slots[i].getMAssignment(), false, m_createNew);
                                m_infoSchedule.mAssignmentCallback(vad.getMAssignment());
                                return;
                        }
                }
                if (m_createNew)
                {
                        MAssignment ma = new MAssignment(Env.getCtx(), 0);
                        ma.setS_Resource_ID(m_S_Resource_ID);
                        ma.setAssignDateFrom(TimeUtil.getDayTime(TimeUtil.addDays(m_startDate,
dayIndex),
                                m_timePanel.getTimeSlot(timeIndex).getStartTime()));
                        ma.setQty(new BigDecimal(1));
                        VAssignmentDialog vad =  new VAssignmentDialog (Env.getFrame(this), ma, false,
m_createNew);
                        m_infoSchedule.mAssignmentCallback(vad.getMAssignment());
                        return;
                }
        }       //      mouseClicked


        public void mousePressed(MouseEvent e)
        {
        }
        public void mouseReleased(MouseEvent e)
        {
        }
        public void mouseEntered(MouseEvent e)
        {
        }
        public void mouseExited(MouseEvent e)
        {
        }

        /**
         *      Dispose
         */
        public void dispose()
        {
                m_infoSchedule = null;
                m_timePanel = null;
                m_where = null;
                m_slots = null;
                this.removeAll();
        }       //      dispose

}       //      VSchedulePanel
```