NSRA Workshop exercises

Index

All the exercises are in ***PUG Workshop*** folder, inside ***sports*** folder.

This folder is divided in ***Basic***, ***Medium*** and ***Complex***, and each of this folders has the internal structure of the sports module:

dao (business classes)
dso (data source classes)
tablas (temp-table includes)
servicios (services)

All the exercises are already solved and working, so you can see what you should do, from the perspective of the user.

To start an exercise, copy the files listed in the next section, to the same folder in the sports module, replacing the existing file. That will break the application until you finish the implementation.

For service folder, most files names have the exercise it belogs as part of the name (xulclasif-ME.p for instance), you have to remove the exercise name from them.

Optional files are surrounded by squares ([]).

Remember that, as always, there are many ways of doing the same thing, and NSRA is not the exception.

Some of the Spanish names hasn't been changed to simplify deployment.

Follow source files comments to complete the exercises. You will probably need to restart WebSpeed agents each time you wan't to try your implementation.

Remember that, as complexity increases, you will have less provided structure and less clues in comments (more freedom to break!!).

Things sorrounded by <> are for you to replace with the right thing. Sometimes they are in strings ('<..>').

Excercise matrix

Complexity \ P. Space	*Items*	*Employee*	*Purchase Order*
Basic	**BI**	**BE**	**BP**
Medium	**MI**	**ME**	**MP**
Complex	**CI**	**CE**	**CP**

No OO knowledge? You'll be fine with Basic (BI,BE,BP)
Willing to suffer? Maybe Complex is hard enough (CI,CE,CP)
You think you are normal? Main diagonal is for you (BI,ME,CP).

Items (Articulo)

All the exercises are about the Items business entity, which is declared as Articulo.

BI (under Basic folder):
dao/Articulo.cls
[tablas/Articulo.i]

MI (under Medium folder):
dao/Articulo.cls
servicios/xulclasif-MI.p (must be named xulclasif.p)
[tablas/Articulo.cls]

CI (under Complex folder):
dao/Articulo.cls
servicios/xulclasif-CI.p (must be named xulclasif.p)
dso/Articulo.cls
[tablas/articulo.cls]

Employee (Empleado)

All the exercises are about the Employee business entity. Each employee has a family, a timesheet, vacations, and benefits.

Due to the fact that this is a Demo application, Benefits is implemented as a main class (instead of a "delegate" one). This is not a "good practice" just a way of showing that you can do strange things if they better fit your needs.

BE (under Basic folder):
dao/Empleado.cls
dao/Horario.cls
dao/Familia.cls
dao/Vacaciones.cls

[tablas/Empleado.i]
[tablas/Horario.i]
[tablas/Familia.i]
[tablas/Vacaciones.i]

ME (under Medium folder):

dao/Empleado.cls
dao/Horario.cls
dao/Familia.cls
dao/Vacaciones.cls
[tablas/Empleado.i]
[tablas/Horario.i]
[tablas/Familia.i]
[tablas/Vacaciones.i]
servicios/xulclasif-ME.p (must be named xulclasif.p)

CE (under Complex folder):

dao/Empleado.cls
dao/Horario.cls
dao/Familia.cls
dao/Vacaciones.cls
dso/Empleado.cls
dso/Horario.cls
dso/Familia.cls
dso/Vacaciones.cls
[tablas/Empleado.i]
[tablas/Horario.i]
[tablas/Familia.i]
[tablas/Vacaciones.i]
servicios/xulclasif-CE.p (must be named xulclasif.p)

Purchase Order (OrdenCompra)

We choose a very complex way of doing PurchaseOrder for the Demo, just to show off!. You can follow that implementation, or try your own.

The thing is that PurchaseOrder and Order are very different things in sports2000, but we choose an abstract view where both belong to the same inheritance chain, because they are both Documents. The chain is as follows (real names in brackets):

```
Document (Documento)
   |___ Document with Lines (DocumentoConDetalle)
            |_____ Purchase Order (OrdenCompra)
            |_____ Order (OrdenVenta)
```

Not happy enough with that, we use only one class for the Lines of both, the Order and the Purchase Order (LineaDocumento) which is "declared" at DocumentoConDetalle level. Even worse we choose to use a dynamic dataset,

instead of a static one (if you use an static one, you have to declare it in the last descendant class and have all the TT declared as PROTECTED).

Happy? Not yet!! We use only one data source class to handle the lines, for Purchase Order and for Order.

Is this the "good way" of doing things with NSRA? NO, really, NO!. This is just for you to see the amount of flexibility you have, that allows you to solve almost any problem (haven't found one you can't yet).

Clarifications:

1. Using dynamic datasets is the "good technique" when you don't know where the inheritance chain ends, or you want to leave it "open" for extension.

2. Using one DSO that selectively chooses what physical storage to use is a really bad idea, the right way of doing it is to let the definition of the lines temp-table to the subclasses, or do it dynamically (as with the dataset), so the "leave" subclass can choose the right name for the table (DSO class is choosen based on the TT name, if done through the ObjectBroker). You can also "manually" instantiate the right DSO, but that's not "recommended" due to the coupling you introduce doing so.

BP (under Basic folder):

dao/OrdenCompra.cls
dao/LineaDocumento.cls
tablas/OrdenCompra.i

MP (under Medium folder):

dao/OrdenCompra.cls
dao/LineaDocumento.cls
dao/DocumentoConDetalle.cls
tablas/OrdenCompra.i
tablas/LineaDocumento.i
servicios/xulcompra.p

CP (under Complex folder):

dao/OrdenCompra.cls
dao/LineaDocumento.cls
dao/DocumentoConDetalle.cls
dso/OrdenCompla.cls
dso/LineaDocumento.cls
tablas/OrdenCompra.i
tablas/LineaDocumento.i
servicios/xulcompra.p

Client side exercises

This exercises are all about writing the client with nsXUL. The reason behind the use of a XUL client is that it will later be transformed to other clients by using transformation templates, if XUL is not the desired interface.

You have to copy the indicated files into the xul/sports/screens folder, replacing the existing ones.

Look inside the files for /**/ comments.

Items (Basic folder)
xul/articulo.xul

Purchase Order (Medium folder)
xul/ordencompra.xul

Reporting exercises

This exercise will introduce you to the TemplateParsing engine, it is far from being a real life reporting job.
The general idea of reporting process is to make reports templates with OpenOffice xml documents (the template can be anything) to transform them (once processed) to the desired format.

In this exercise you will do a little work with TemplateParsing markup to change the behavior of the “show source code” report (the listing you get when you press the “Show Me!” button).

For this exercise you will break the department (departamento) window. Copy the file to the xul/sports/screens/ folder, replacing deparamento.xul and follow /**/ comments.

Aditonally you will need to modify the service. The service is in the file 'xulinfo.p' inside sports/servicios folder. Search for the instantiation of 'TmptParser' and change it to 'ABLTmptParser'. You can play with it to see what happends if you don't. Remember to restart WebSpeed after changing the service.

Purchase Order (Complex folder)
xul/departamento.xul.