Article
John Murray · Mar 14, 2016 3m read

Source control and the production class

When you create an Ensemble production your namespace acquires a new class definition. For example here is what the class that defines the Demo.Loan.BankUSProduction production in the ENSDEMO namespace looks like when opened in Studio:

```
Class Demo.Loan.BankUSProduction Extends Ens.Production [ ClassType = "", ProcedureBl
ock ]
{

XData ProductionDefinition
{
<Production Name="Demo.Loan.BankUSProduction" TestingEnabled="true">
  <ActorPoolSize>2</ActorPoolSize>
  <Item Name="Demo.Loan.BankUSTerminalService" ClassName="Demo.Loan.BankUSTerminalSer
vice" PoolSize="0" Enabled="true" Foreground="false" InactivityTimeout="0">
  </Item>
  <Item Name="Demo.Loan.WebOperations" ClassName="Demo.Loan.WebOperations" PoolSize="
1" Enabled="true" Foreground="false" InactivityTimeout="0">
  </Item>
  <Item Name="Demo.Loan.BankUS" ClassName="Demo.Loan.BankUS" PoolSize="1" Enabled="tr
ue" Foreground="false" InactivityTimeout="0">
  </Item>
</Production>
}

ClassMethod Test()
{
    Do ..Start()
    ;
    Set tSC=##class(Ens.Director).CreateBusinessService("Demo.Loan.BankUSTerminalServ
ice",.tBusinessService)
    If $$$ISERR(tSC) Do $system.OBJ.DisplayError(tSC) Quit
    For {
        Read !,"amount:name:taxid:nationality>",tInput,! Quit:tInput=""
        Set tSC=tBusinessService.ProcessInput(##class(Ens.StringContainer).%New(tInpu
t),.tOutput)
        If 'tSC Do $system.Status.DisplayError(tSC) Continue
        Write !,tOutput.StringValue
    }
    Set tBusinessService = $$$NULLOREF
    ;
    Do ..Stop()
    Quit
}

}
```

At the heart of that class is the XData block named ProductionDefinition. The XML within this block records the

configuration items that make up the production.

Each item has a name, specifies the name of the class that implements it, and lists the settings for the item.

Since your production definition is a class, it can be versioned in a source control system in the same way as you'd version the classes that implement the production's configuration items.

However, there are a few challenges. I hope this posting will start a useful discussion about them.

The first challenge is that a lot of our interaction with the production class happens through Portal rather than through Studio. And whereas Studio collaborates with source control (e.g. consults your source control class for permission to save an edited production class), Portal ignores source control when updating the production class.

The second challenge is that some configuration item settings need to be specific to the Ensemble instance where the production is being run. For instance, interfaces probably need to talk to different IP addresses when running in your test environment versus when running on your live environment. So if your source code management procedures mean that you export your production class from your test environment and import it into your live environment in order to deploy a new release of your production, you run the risk of making your live instance talk to test instances of your external systems.

How does your Ensemble development team deal with these issues?

#Ensemble