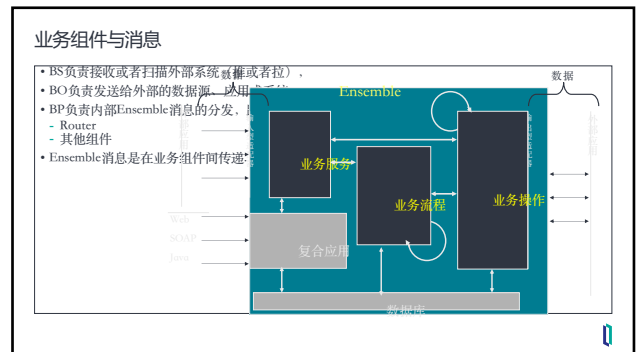
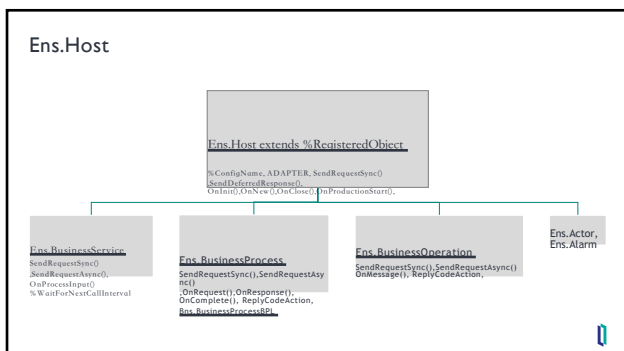




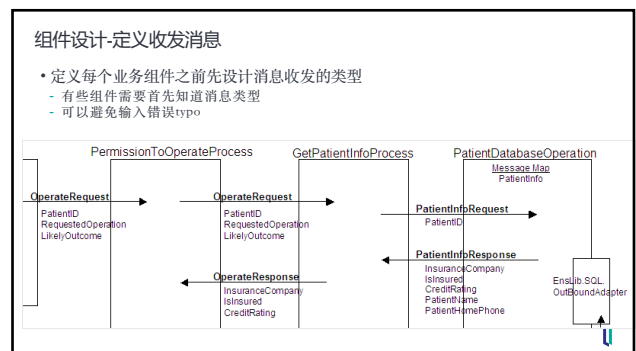
1



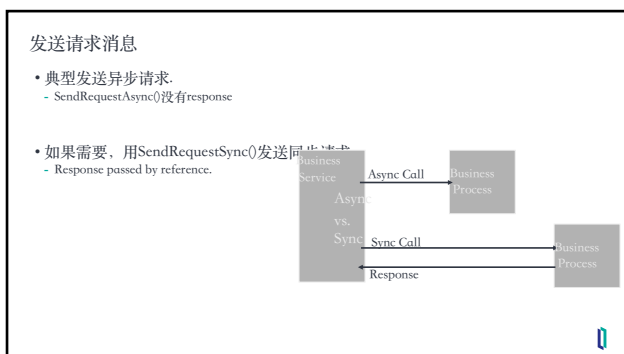
2



3



4



5



6

## 嵌入的BS

- EnsLib.DICOM.Service.File
- EnsLib.DotNetGateway.Service
- EnsLib.EDL.ASTM.Service.Standard
- EnsLib.EDL.EDIFACT.Service.Standard
- EnsLib.EDL.X12.Service.Standard
- EnsLib.FTP.PassthroughService
- EnsLib.File.PassthroughService
- EnsLib.XML.FileService
- EnsLib.HTTP.Service
- EnsLib.JavaGateway.Service
- EnsLib.MQSeries.PassthroughService
- EnsLib.Printing.PrintService
- EnsLib.PushNotifications.AppService
- EnsLib.RecordMap.Service.Base
- EnsLib.RecordMap.Service.ComplexBatchStandard
- EnsLib.SOAP.Service
- EnsLib.TCP.Framed.PassthroughService
- EnsLib.TCP.PassthroughService
- EnsLib.TCP.StatusService
- EnsLib.Testing.Service
- EnsLib.XSL.T.TransformedService

7

## 嵌入的BS

- EnsLib.HL7.Service.AckInStandard
  - EnsLib.HL7.Service.HTTPAckInService
  - EnsLib.HL7.Service.TCPAckInService
- EnsLib.HL7.Service.Standard
  - EnsLib.HL7.Service.FTPService
  - EnsLib.HL7.Service.FileService
  - EnsLib.HL7.Service.HTTPService
  - EnsLib.HL7.Service.SOAPService
  - EnsLib.HL7.Service.TCPService
- EnsLib.EDL.XML.Service.Standard
  - EnsLib.EDL.XML.Service.FTPService
  - EnsLib.EDL.XML.Service.FileService
  - EnsLib.EDL.XML.Service.TCPFramed
- EnsLib.XML.Object.Service.Standard
  - EnsLib.XML.Object.Service.FTPService
  - EnsLib.XML.Object.Service.FileService

8

## BS工作原理

- BS用于接收外部请求
  - BS invoked by 客户端应用或者"input object", 不是一个request
  - 等待特定外部事件(应用发出的通知, 接收到的TCP消息)
  - 读取、解析、验证数据
  - 返回信息, 握手应答信息等
  - 创建一个primary request, 分配message ID和the session ID, 发送至BP或者BO
- BS调用方式
  - Inbound Adapter
  - Direct Invocation(直接调用)

9

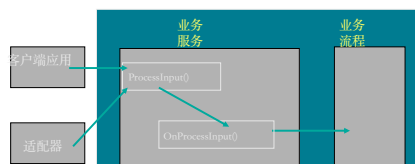
## BS的实现

- 使用HealthConnect内嵌的BusinessService类
- 自定义业务实现类, 继承Ens.BusinessService
  - 选择是否使用Adapter
  - 自动产生OnProcessInput()并实现
- BS向其他组件发消息
  - SendRequestAsync(), SendRequestSync()
- BS类的回调函数
  - OnProcessInput()
  - OnError()
  - OnInit()
  - OnProductionStart(), OnProductionStop()
  - %OnNew() .. and more

10

## OnProcessInput()

- Adapter或者外部客户端调用ProcessInput(). OnProcessInput()是内部方法 ProcessInput()的回调函数, 通常情况下这是创建BS唯一必须实现的方法。每个BS必须有OnProcessInput()方法
- Web Services使用特殊的BS, 无论是否使用adapter, 执行WebMethod而不是OnProcessInput()



11

## 实现OnProcessInput()

- Method OnProcessInput(pInput As %RegisteredObject, pOutput As %RegisteredObject) As %Status**
- 设置%相关参数, 比如%WaitForNextCallInterval控制Ensemble调用OnTask()方法的频率(只适用于有Adapter的情况)???
  - 验证传入的对象(可选)
  - 对于接收到的对象决定如何处理
  - 创建发送给下一个目录的Ensemble请求消息request, %New()
  - 给request各个字段赋值
  - 发送request至BP或者BO
  - 设置output argument (pOutput)
  - 返回合适的状态%Status
  - (request消息不需要%Save())

12

## OnProcessInput()范例

```

Class Demo.HL7v3.Service.FileIn Extends Ens.BusinessService {
Parameter ADAPTER = "EnsLib.File.InboundAdapter";
Method OnProcessInput(pInput As %FileCharacterStream, pOutput As Ens.Response) As %Status {
    Set tStatus = ##class(%XML.XPATH.Document).CreateFromStream(pInput, tDocument)
    Set tStatus = tDocument.EvaluateExpression("/*/", "name()", tResults) Set tStatus = pInput.Rewind()
    If (tResults.Count() > 0) Set tRoot = tResults.GetAt(1).Value Else Set tRoot = "<errorNoRootElement>"
    Set tRequest = ##class(Demo.HL7v3.Message).%New()
    Set tRequest.Name = tRoot Set tRequest.DocType = ""
    Set tRequest.Source = pInput.Attributes("Filename")
    Do tRequest.Content.CopyFrom(pInput)
    Set tStatus = ..SendRequestSync("Route and Transform Message", tRequest, tResponse)
    Quit $$$OK
}
}

```

13

## 练习

- 使用Studio创建Test.BSSample1，实现OnInit()，异步发送发送消息给。。。
- 实现OnProcessInput()，同步发送上节课的自定义消息给。。。
- 研究消息头，消息体的保存，SQL查询
- 删除消息，验证消息可以被正确删除

14

## 调用BS-Direct Invocation

通常用来直接调用BS的应用有Web应用，CSP页面，Java 应用等等，尤其是REST服务，必须使用直接调用BS来实现。

```

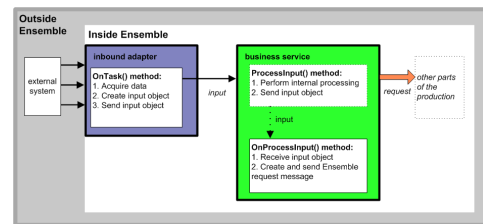
Set tSC = ##class(Ens.Director).CreateBusinessService("MyService", tService)
If ($IsObject(tService)) {
    Set input = ##class(MyObject).%New()
    Set input.Value = 22
    Set tSC = tService.ProcessInput(input, output)
}

```

15

## 调用BS - InboundAdaptor

Adapter负责外部系统的连接，接收外部请求，定时查看数据源(Call Interval)。如果有输入，输入被发给Business Service。



16

## BS设计要点

- BS应尽量简单，主要职能是收到请求，产生ensemble消息并发给其他组件。异步调用时无需等待其他组件的返回消息即可产生返回消息给外部系统。应该把复杂的逻辑处理放到BP
- BS不接收Ensemble消息，因此没有消息队列
- BS调用其他组件尽量使用异步

19

## 练习

- 创建BS使用TCP Adapter发送字符串
- 创建BS使用...???

20

## Business Operation

21

### BO工作原理

- BO可被用来访问Ensemble之外的功能。例如从外部数据源读取信息，从外部网站检索信息，启动第三方应用的操作，或向用户开发的应用传输数据等等。  
- 除了连接外部应用，BO可以调用其他BO、BP(很少用)
- BO通常使用适配器访问Ensemble外部，或者通过调用外部应用的API或其他方法。
- BO可以被BS或BP调用
- BO根据请求消息同步或异步，可以返回或不返回Response消息。

22

### BO的创建

- 使用内置的BusinessOperation类
- 用户自定义BO
  - 继承Ens.BusinessOperation
  - 定义method,以及method使用的请求消息和响应消息
  - 定义Message Map

可以使用Studio的向导创建自定义BO

23

```
Class SEDemo.Common.DummyBO2 Extends Ens.BusinessOperation {
    Parameter ADAPTER = "EnsLib.FTP.OutboundAdapter";
    Property Adapter As EnsLib.FTP.OutboundAdapter;
    Parameter INVOCATION = "Queue";

    Method processStringRequest(pRequest As Ens.StringRequest, Output pResponse As Ens.StringResponse) As %Status {
        Quit $$$ERROR($$$NotImplemented)
    }
}

XData MessageMap {
    <MapItem>
        <MapItem MessageType="Ens.StringRequest">
            <Method>processStringRequest</Method>
        </MapItem>
    </MapItems>
}
```

24

### Message Mapping

- 收到的请求消息到执行消息的对应，必须是一一对应。
- BO收到message map中没有定义的消息，将产生错误“TheRequestMessage not handled”
 

```
<MapItems>
    <MapItem MessageType="Ens.StringRequest">
        <Method>processStringRequest</Method>
    </MapItem>
    <MapItem MessageType="Ens.StringContainer">
        <Method>processStringContainer</Method>
    </MapItem>
    <MapItem MessageType="Ens.StreamContainer">
        <Method>processStreamContainer</Method>
    </MapItem>
</MapItems>
```

25

### OnMessage()

- OnMessage()是直接代码里定义“收到的请求”和“处理请求的method”之间的一一对应。在某些复杂的编程中使用OnMessage()比使用Message Mapping更方便，起的作用是一样的。
- 参见EnsLib.EDI.XML.Operation.FileOperation 里的OnMessage()定义
- ???When a BS calls ProcessInput it is wrapped with a Try/Catch and when a BO calls OnMessage, that is enclosed in an \$ZT error trap. So any expected errors are handled fairly cleanly.

26

## 练习

- 创建一个BO, 收到上节课的请求消息后在terminal打印 “Yes, I received message”

27

## Outbound Adaptor

- EnsLib.Email.OutboundAdapter
- EnsLib.FTP.OutboundAdapter
- EnsLib.File.OutboundAdapter
- EnsLib.HTTP.OutboundAdapter
- EnsLib.JMS.OutboundAdapter
- EnsLib.JavaGateway.OutboundAdapter
- EnsLib.LDAP.Adapter.Outbound
- EnsLib.LDAP.OutboundAdapter
- EnsLib.MFT.Adapter.Outbound
- EnsLib.MQSeries.OutboundAdapter
- EnsLib.MQTT.Adapter.Outbound
- EnsLib.PEX.OutboundAdapter
- EnsLib.Pipe.OutboundAdapter
- EnsLib.SOAP.OutboundAdapter
- EnsLib.SQL.OutboundAdapter
- EnsLib.TCP.DuplexAdapter
- EnsLib.TCP.OutboundAdapter
- EnsLib.Telnet.OutboundAdapter
- EnsLib.UDP.OutboundAdapter

28

## 练习

- 使用EnsLib.Email.OutboundAdapter发送邮件到。。。
- 问题:

29

## Retry(重试)

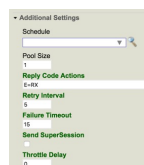
- 重试间隔 (RetryInterval )
- 故障超时 (FailureTimeout)
  - 重试时间长度, 该时间过后返回为错误
- 延迟处理时间 (ThrottleDelay )
  - 处理下一条消息之前强制闲置时间, 单位ms

30

## Reply Code Action

`<code>=<actions>, (<code>,<code>)=<actions>,...`

- Code
  - E - 返回错误状态
  - E#<statusCode> - 返回错误状态并且返回错误代码
  - E\* <text> - 返回错误状态并且返回错误信息
  - X - 没有任何返回信息
- Action
  - C - 按照消息完成OK处理
  - W - 按照警告处理。如果没有其他非警告信息产生则按照完成OK处理。
  - R - 重试消息。按照RetryInterval以及FailureTimeout设置
  - S - 挂起消息, 记录错误, 处理下一条消息。
  - D - 不启用该操作项目。记录错误。把消息放置到队列头的位置。
  - F - 失败, 记录错误。处理下一条消息。



31

## Reply Code Action

- Default value is E=F
- 从左到右边评估, E=RD, 先重试再disable.

32

### Reply Code Action

例子:

```
'E#6301=R,E#<Ens>ErrGeneral=R,E=F'
```

- This specification will result in a retry when either error code 6301 or error code <Ens>ErrGeneral is encountered.
- Any other errors will cause the Operation to fail processing of the current message and return the error status to the caller.



33

### Reply Code Action for HL7 V2 ACK

- :A - 匹配AA or CA values (Accept)
  - :E - 匹配AE or CE values (Error)
  - :R - 匹配AR or CR values (Reject)
  - :~ - 回复 MSA:1字段为空
  - :\* - MSA:1值不相符 (default=F)
  - :~ - 匹配回复不包含MSA段
  - :!P - 回复消息中MSA:2 与原始消息ControlId字段不相符
  - :!T - 回复消息中MSH:9与原消息 Type name不相符
- Default
    - :R=RF,:E=S,:~S,:A=C,:\*S,:!P=W,:!T=C'
    - 返回为AR或者CR则重试
    - 返回AE或者CE、不包含MSA段则挂起消息
    - 返回AA或者CA则标示完成
    - 回复消息MSH: 9不符则标示完成
    - 回复消息MSH: 2不符则标示警告



34

## Business Process

35

### Business Process基本概念

- BP实现业务逻辑，决定一个外部请求消息应该发到哪里，怎么发； 怎样实现业务协作(Orchestrate)，怎样转发或者生成返回消息给外部业务系统。
- BPL和DTL使业务人员而不是开发人员可以参与开发和产品的维护
- 消息路由(Message Router)是BP,使用路由规则(routing rule)
- BS应该尽量简单，把复杂的逻辑放在BP
  - 数据过滤、校验、转换、汇总
  - 消息校验、路由、
  - 业务协调、术语管理
  - 订阅发布、工作流、
  - 告警管理
  - ...



36

### Business Process基本概念

BP只发送请求消息给BO或者另一个BP

- 请求可以是同步或者异步的，
  - BP可以要等一个响应消息等很久，等待过程中，BP会被写到磁盘，收到响应才重新调出。
- BP发送的请求消息和收到的响应消息都在Ensemble内部，不连接外部系统。



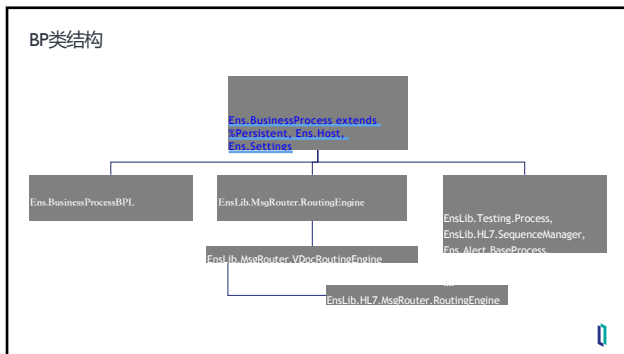
37

### BP相关的工具和技术

- BPL
- DTL
- 通用规则(General Rule)
- 消息规则(Message Rule)
- 订阅发布(PubSub)
- Workflow
- 查找表(Lookup Table)
- 客户定制函数(Custom Functions)
- ...



38



39

### 预置消息路由

- 不同的Message Rule使用不同的RouteEngine, 被称为Router, 用于常用消息的路由。
  - General Msg Router
  - Vdoc Message Router
  - Alert Message Router
  - HL7 Message Router
- 创建Message Rule, 生成Message Router包含另外的章节

40

### 预置消息路由

通常不需要修改程序, 直接加入Production, 并配置Routing Rule

- EnsLib.MessageRouter.RoutingEngine.
- EnsLib.HL7.MessageRouter.RoutingEngine.
- EnsLib.MessageRouter.VDocRoutingEngine.

按照不同的特性路由消息, 比如:

- Source component.
- Messages class.
- Values within message.

41

### 开发BP的两种方式

- BPL.
  - 继承Ens.BusinessProcessBPL.
  - 图形编辑工具
  - 编译后生成XDATA
- 代码实现(客户定制BP)
  - 继承Ens.BusinessProcess.
  - 必须实现OnRequest()
  - 可以override OnResponse(), OnComplete()

Class Demo.Loan.FindRateDecisionProcessCustom Extends Ens.BusinessProcess

43

### BPL

- BPL 编译后生成继承Ens.BusinessProcessBPL.
- 业务逻辑生成XDATA



```

// BPL Definition
XData BPL [ XMLNamespace = "http://www.interayama.com/bpl" ]
{
  <process language='objectscript' request='RBN.MSG.reqUploadPatient' response='RBN.MSG.resUploadPatient' >
    <context>
      <sequence name='557' yypid='1331' >
        <transform name='Msg transform' class='RBN.OT.Test' source='request' target='context' >
          <call name='Log Patients' target='DataLog' async='0' xpsid='550' yypid='1331' >
            <request type='RBN.MSG.reqLogPatient' >
              <assign property='callrequest' value='context.reqLogPatient' action='Pre' >
                <response type='RBN.MSG.resLogPatient' />
            </call>
          <call name='Check Patient' target='PMI' async='0' xpsid='550' yypid='1332' >
            <annotation>This step will check if patient exist in PMI</annotation>
          </call>
        </transform>
      </sequence>
    </context>
  </process>
}

```

44

### 客户定制BP(Custom Code BP)

- 在某些情况下, 客户可能需要处理复杂的BP逻辑, 选择直接从代码实现而不是使用BPL。举例说明, 当对内置的Router Engine需要修改时; 或者直接大量重复使用COS语句的工具时。

- 继承Ens.BusinessProcess.
- 必须实现OnRequest()
- 可以override OnResponse(), OnComplete()

Class Demo.Loan.FindRateDecisionProcessCustom Extends Ens.BusinessProcess

45

- 同步调用
  - 实现OnRequest()
- 异步调用
  - 实现OnRequest()
  - 实现OnResponse()
    - 如果等待几个不同消息，用completekey
  - 可能需要实现OnComplete()
    - 知道等待的消息可能不会回来时
    - 具体例子参见EnsDemo命名空间的

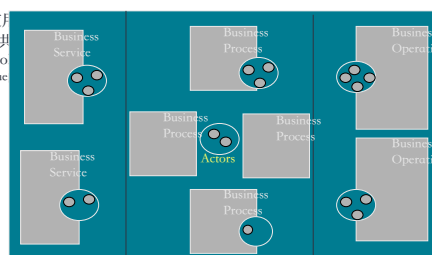
- Demo.FloodMonitor.CustomBusinessProcess
- Demo.Loan.BankUS
- Demo.Loan.FindRateDecision
- EnsLib.MsgRouter.RoutingEngine

```

graph TD
    Start([Start]) --> EnsDirector[Ens.Director method  
startProduction()]
    EnsDirector --> Foreach[Foreach component]
    Foreach --> JobEns[Job Ens. Job method  
start()]
    JobEns --> Instantiate[Instantiate component  
class-call Oninit()]
    Instantiate --> LoopOnTask[Loop calling OnTask()]
    LoopOnTask --> OnTaskMethod[OnTask()]
    
    OnTaskMethod -- Service --> CallAdapter[Call Adapter's OnTask()]
    OnTaskMethod -- Process --> CallQueueReq[Call Ens.Queue method  
Request()]
    OnTaskMethod -- Operation --> CallQueueReq
    
    CallAdapter --> OnTaskCallsService[OnTask() calls Service  
OnProcess(prop)]
    OnTaskCallsService --> UserCode1[User Code]
    
    CallQueueReq --> PassToBizProc[Pass to Business Process  
MessageHandler(handler)]
    CallQueueReq --> PassToOp[Pass to Operation  
MessageHandler(handler)]
    
    PassToBizProc --> OnReqOnErr[OnRequest(),  
OnResponse(), or OnError()]
    OnReqOnErr --> UserCode2[User Code]
    
    PassToOp --> PassToSpec[Pass to method specified  
by MessageHandler]
    PassToSpec --> UserCode3[User Code]
  
```

- Pool Size: 组件可以使用

- Pool Size: 组件可以使用
- Actor: 供给BP使用的共
- 组件要定义自己的Pool
- 只用于Invocation=Queue



Production的Actor Pool Size配置和每个业务组件的Pool size配置决定了对做某个工作有多少job可以提供。这写配置数字是Production设计中重要的一部分而且在Production部署上线后通常不用改动。设置更大的数字并不见得有帮助，大多数的pool size设置为0或者1，设置大于1的数字可能会有严重的后果。

The choice of Actor Pool Size for the production, and Pool Size for each business host, determines how many jobs are available to perform which types of work for the production. These numbers are an essential part of the production design and are unlikely to need adjustment once the production is deployed live. Larger numbers are not necessarily helpful; most pool sizes are best set to either 0 or 1, and there can be serious consequences when sizes are set to a number greater than 1.

[https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=ECONFIG\\_PoolSize](https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=ECONFIG_PoolSize)

- 对于BP:
  - 默认使用Actor Pool Size, 默认值为2, 随着BP数量的增多或者业务量的增多, 应该适当的提供Actor Pool Size的数量, 以测试中不会在BP上产生队列为标准。
    - Actor Pool Size的默认值100
  - 如果某个BP使用public queue, 速度令人满意, 尝试使用Private Queue, BP的运行会更快。使用Private Queue会减少此BP受到其他BP运行效率的影响, 尤其是其他BP运行性能不佳时。
  - 使用Private queue, 设置Pool Size初始值为1, 随需要增加。
  - Pool size=1, 可以确保消息队列的处理的FIFO。




Pool Size和Actor Pool Size设置

- 对于BO:
  - 任何INVOCATION = Queue的BO需要设置PoolSize, 默认值为1, 按需要增加增加后测试观察。
  - 非1的Pool Size设置不保证FIFO
- 对于BS
  - 任何不使用Adaptor的BS, PoolSize设置会被忽略。(设置为0)
  - 对于HTTP, SOAP, REST接口, 如果是使用CSP Server的方式, 设置PoolSize=0 (Direct Invocation)
  - 使用Inbound Adaptor的BS, 需要为每个adaptor的后台进程设置PoolSize, PoolSize=n, 意味着有n个adaptor的后台进程可以同时工作, 比如设置pool size=3, 3个job都定时去检查是否有input。默认设置为1。
  - 对于SQL Inbound Adaptor, 通常设置PoolSize=1, 要保证FIFO
  - 使用TCP adaptor时, 如果设置"job Per connection"(为每个作业创建连接), pool size数就是同时的最大连接数。(HIL7不选)

52

组件配置项

黑色: 来自产品设置  
绿色: 来自类设置  
蓝色: 来自default Setting page for the namespace



53

定制组件配置项

- 1. 添加要配置项的名字作为一个属性
- 2. 添加或者修改class parameter SETTINGS
- 3. 设置项的可选项, 放置的位置

```
Class EnsLib.EDLXML.Operation.Standard Extends Ens.BusinessOperationProperty Format As %String;  
Parameter SETTINGS =  
"Format:Basic,SearchTableClass=selector?context={Ens.ContextSearch/SearchTableClasses?host=EnsLib.EDLXML.Operation.Standard}";
```

- 如果要删除已有的设置项, 在类定义里修改  
Parameter SETTING="-foo" 可从配置项中删除foo这个选项

54

添加业务活动监控

- 生产环境的Production需要业务活动监控组件以自动采集业务活动数据
  - Ens.Activity.Operation.Local
- 活动数据将被保存在数据库中, 方便分析利用
- 内建BI分析模型, 直接针对活动数据进行分析(不需要DeepSee license)



55

添加业务活动监控

- 可以修改活动监控数据存储的位置, 避免对正常生产系统产生影响
- 应考虑定期删除活动监控数据
  - PurgeActivityData task
  - 该任务应该在存储业务活动数据的命名空间上运行!!!
  - 它与Ensemble管理数据purge是不同的

56

配置Ens.Alert

- 生产环境中一般需要配置向外部监控系统发送警告和错误。当Production中有名为Ens.Alert的组件时, 应该将所有的管理报警发送给它。Ens.Alert可以是:
  - BO: 例如EnsLib.Email.AlertOperation
  - BP: 例如一个消息路由
  - 或报警管理器 / Alert Manager
- 在所有其他组件中设置以下两项,
  - Alert On Error: 针对错误报警
  - Alert On Bad Message: 消息验证错误报警
- 在有基准数据的情况下, 可以设置业务组件的以下触发条件, 以发出管理报警
  - Alert Grace Retry Period: 发送重试宽限期, 通常针对BO, 在宽限期内重试成功, 则不发送警告
  - Inactivity Timeout: 非活动超时报警, 通常针对BS, 在指定时间内没有收到消息, 则报警
  - Queue Wait Alert: 队列头消息等待处理超时报警
  - Queue Count: 队列长度报警

57



58

### 业务操作的调用模式-Queue与InProc

```

class Demo.Loan.FindRateEmailOperation Extends Ens.BusinessOperation
{
    // 2 modes: Queue, InProc
    Parameter INVOCATION = "Queue";
    // Name of the adapter class
    Parameter ADAPTER = "EnsLib.Email.OutboundAdapter";
    XData MessageMap
    {
        <MapItem MessageType="Demo.Loan.Msg.SendReply">
            <Method>EmailSendReply</Method>
        </MapItem>
    }
}

```

59

### 消息的非持久化

业务组件的Parameter: INVOCATION  
"queue" and "inProc"

每个消息有个INVOCATION

- Queue: 一个消息在一个job里创建, 然后放到队列, 同时原job释放。(默认处理方式)
- InProc: 一个消息创建, 发送, 接收都是一个job.

#### • 消息有2种处理方式

- 排队(Queue): 消息由一个后台进程产生, 并放入队列, 之后由其它后台进程处理、发送
- 直接处理(InProc): 消息由一个后台进程产生、处理、发送
  - 此时消息不会放入队列

60

### Queue vs InProc

问题: InProc时, caller怎么知道要建一个job的instance? answer: 见Ens.host.SendRequestSyncMultiple(), inProc时不建session, 不建message loader, 不建Ens.Queue, 直接调用SendRequestSync, 所以说是发了一个persistent的消息过去, 但有没有message还真不确定。  
Business Operation 需要继承 Ens.BusinessOperation.  
The ADAPTER parameter 指定了outbound适配器, 它是Ens.OutboundAdapter的子类。  
Business Operation将调用适配器来向外部系统发送信息或获取信息。  
INVOCATION parameter:  
Queue - 此消息在一个job内创建, 并在该job被释放的时候被置入队列, 其他的job可以从该队列中获取并处理这个消息。如果INVOCATION参数被置为Queue, 该Business Operation会有一个或多个关联的背景进程, 并且拥有独立的消息队列。这是Business Operation的默认行为。  
InProc - 当请求消息被发送到Business Operation时, 该Business Operation的实例将在其调用者的进程中被创建, 并在同一进程中被执行。这意味着请求消息将存于同一进程中被创建、传递和对外发布。这样的Business Operation没有独立的消息队列。  
除非有特别的原因, Business Operation 不应被设为InProc。

61

### 业务操作的调用模式-Queue与InProc

```

class Demo.Loan.FindRateEmailOperation Extends Ens.BusinessOperation
{
    // 2 modes: Queue, InProc
    Parameter INVOCATION = "Queue";
    // Name of the adapter class
    Parameter ADAPTER = "EnsLib.Email.OutboundAdapter";
    XData MessageMap
    {
        <MapItem MessageType="Demo.Loan.Msg.SendReply">
            <Method>EmailSendReply</Method>
        </MapItem>
    }
}

```

62