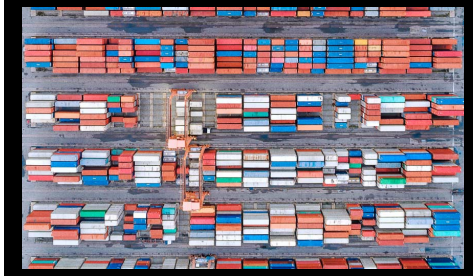
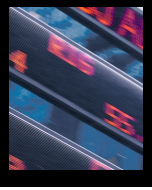
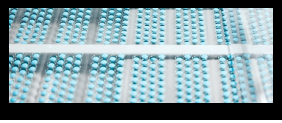
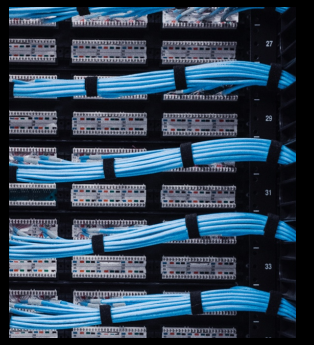
Using SQL Adapter in Production

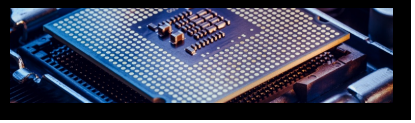
InterSystems®
Creative data technology











学习资料

- 在线文档

  https://docs.intersystems.com/healthconnectlatest/csp/docbook/Doc.View.cls?KEY=ESQL_inbound
  https://docs.intersystems.com/healthconnectlatest/csp/docbook/Doc.View.cls?KEY=ESQL_outbound
  https://docs.intersystems.com/healthconnectlatest/csp/docbook/Doc.View.cls?KEY=ESQL_adapter_methods_creating

- Learning Service 课程

提纲

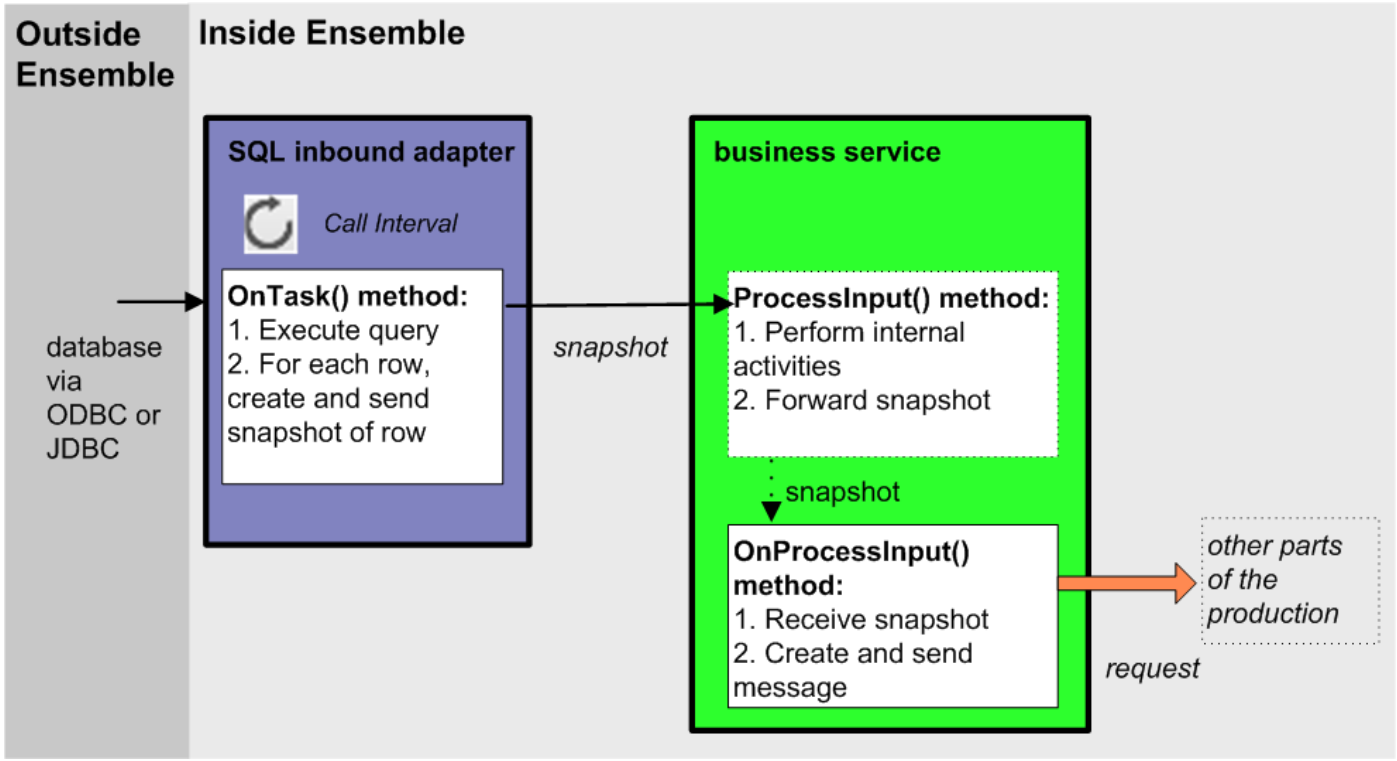
- SQL Inbound Adapter
  - 基本原理
  - 使用SQL adapte的BS
- SQL Outbound Adapter
- 调用存储过程
- SQL事务性调用

SQL Inbound Adapter

EnsLib.SQL.InboundAdapter

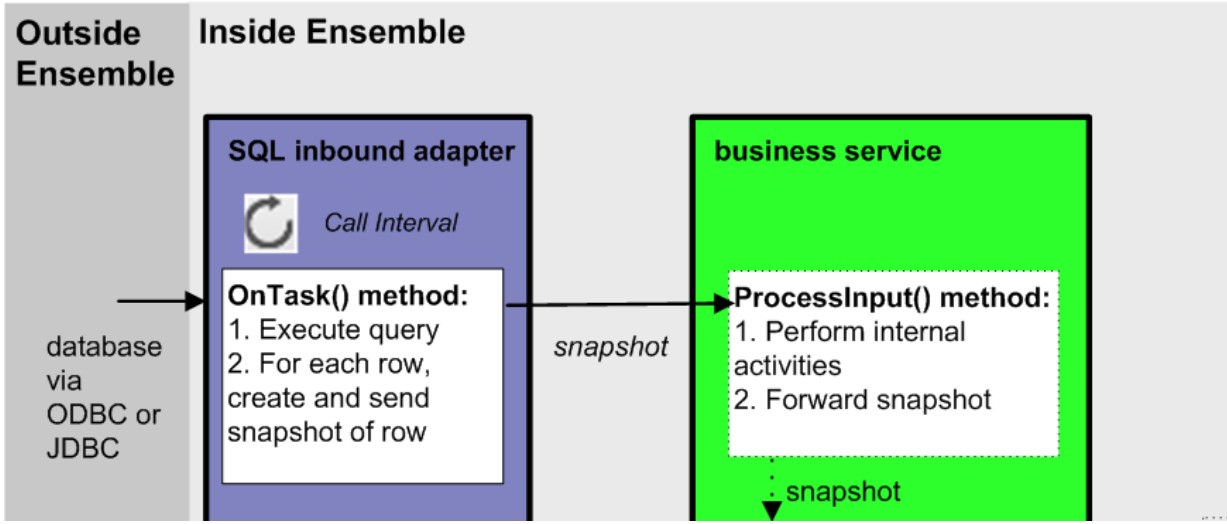
SQL Inbound adapter的工作原理

- 自动发现数据库的改动，以只抽取更新的数据。
- 用户必须自己生成BS类，通用的方法是使用Studio的向导创建，创建的类如下页



SQL Inbound adapter

- ODBC/JDBC连接数据库
- 周期执行OnTask()方法，执行数据库查询语句
  - 循环周期通过BS的CallInterval 设置, 默认5秒
- 查询返回结果集，iterate结果集中的每一行数据
  - 该数据已经被处理过并且未被修改，则忽略(KeyFieldName)
  - 该数据已被处理并出错，则忽略该记录直到下次重启
  - 否则，创建EnsLib.SQL.Snapshot实例，发给BS的ProcessInput()处理
- ProcessInput()包含一系列BS要求的的监控和错误日志
- ProcessInput()调用OnProcessInput(), 用户应实现处理EnsLib.SQL.Snapshot的逻辑，并产生ensemble message发给其他组件



- asdfa
- ODBC/JDBC连接数据库
- 周期执行OnTask()方法，执行数据库查询语句
    - 循环周期通过BS的CallInterval 设置, 默认5秒

—查询返回结果集，iterate结果集中的每一行数据

该数据已经被处理过并且未被修改，则忽略(KeyFieldName)

该数据已被处理并出错，则忽略该记录直到下次重启

否则，创建EnsLib.SQL.Snapshot实例，发给BS的ProcessInput()处理

创建一个BS使用Adapter

- 使用Studio向导产生的类, 开发者只需要

```
Class Practice.BS.sampleSQL Extends Ens.BusinessService
{

Parameter ADAPTER = "EnsLib.SQL.InboundAdapter";

Method OnProcessInput(pInput As EnsLib.SQL.Snapshot, Output pOutput As %RegisteredObject) As %Status
{

        Quit $$$ERROR($$$NotImplemented)
}

}
```

使用Snapshot对象

参加链接文档https://docs.intersystems.com/healthconnectlatest/csp/docbook/Doc.View.cls?KEY=ESQL_inbound#ESQL_snapshot_using

注意: Snapshot有属性和方法处理多个记录，但在Inbound Adapter的BS里，所有记录都是一条条处理的。 在BO中使用Snapshot可以处理多个记录。

```
Class EnsLib.SQL.Snapshot Extends (%Persistent, %IResultSet, %XML.Adaptor)
method Get(pName As %String, pRow=..%CurrentRow) returns %String
method GetColumnId(pName As %String) returns %Integer
method GetData(pColumn As %Integer, pRow=..%CurrentRow) returns %String 默认第一列为1
method GetColumnName(pColumn As %Integer = 0)
method GetColumnSize(pColumn As %Integer = 0)
method GetColumnType(pColumn As %Integer = 0)
```

Snapshot使用Get()

```
Class ESQL.NewService1 Extends Ens.BusinessService
{
Parameter ADAPTER = "EnsLib.SQL.InboundAdapter";

Method OnProcessInput(pInput As EnsLib.SQL.Snapshot, pOutput As %RegisteredObject) As
%Status
{
    set req=##class(ESQL.request).%New()
    set req.CustomerID=pInput.Get("CustomerID")
    set req.Name=pInput.Get("Name")
    set sc=..SendRequestSync("ESQL.operation",req,.pOutput)
}
}
```

初始化Adapter

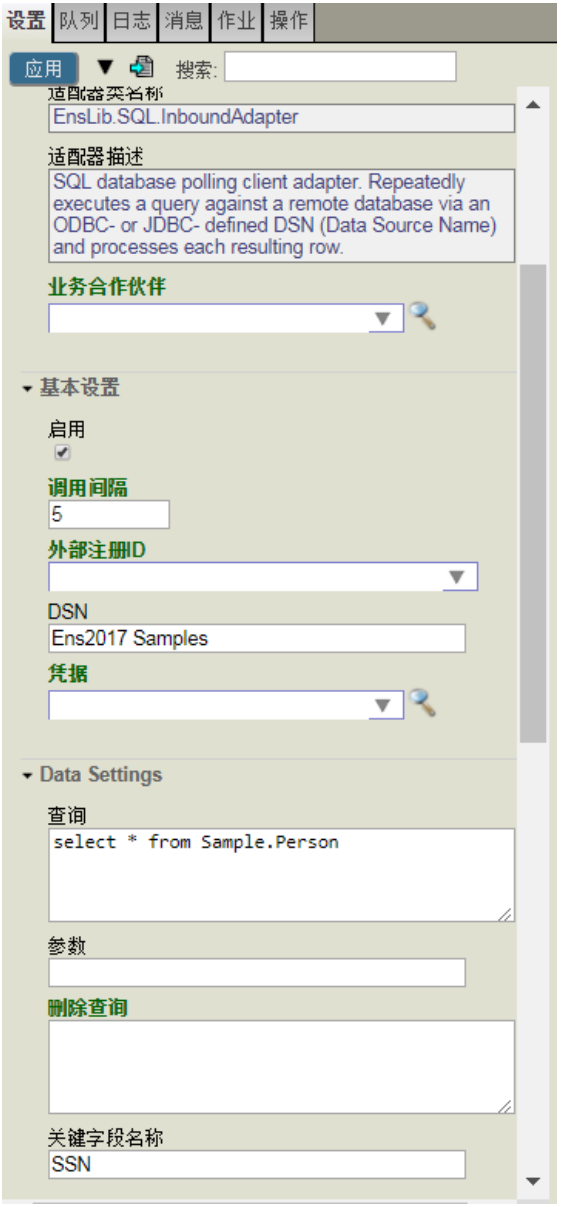
```
Method OnInit() As %Status
{ #; initialize persistent last key value
Do ..Adapter.InitializePersistentValue(..%ConfigName,,0) Quit $$$OK
}

Method OnInit() As %Status
{ #; must initialize so the query can do a numeric comparison
Do ..Adapter.InitializePersistentValue(..%ConfigName,"TopSales",0)
Quit $$$OK
}
```

添加组件到Production-设置数据连接

- 设置数据源
  - ODBC连接需要加入组件DSN设置，DSN连接的顺序
    - Caché SQL Gateway(可以是ODBC或者JDBC)　　accessplayground
    - JDBC URL(如果包含冒号)　　jdbc:Cache://localhost:9982/Samples
    - 操作系统ODBC DSN accessplayground
  - JDBC连接可以使用DSN设置连接一个SQL Gateway， 或者在组件中设置
    - JGService: Production中需要使用EnsLib.JavaGateway.Service
    - JDBCDriver
    - JDBCClasspath
- 轮询时间间隔
  - 控制多长时间查询有没有新数据记录(CallInterval)
- Credentials
  - 用于连接外部数据源的用户密码配置

添加组件到Production-SQL语句

- 查询
  - SELECT * FROM Sample.Person WHERE Age > ?, PostalCode = ?
  - 使用字段而不是*可避免通过JDBC连接的数据源更新的bug
- 参数
  - value,value,value,...
  - 以$开始：适配器property
    - Parameters中定义
  - 以&开始：适配器常量
    - OnInit()中设置键值对
    - &%LastKey, 适配器上次处理过的行的IDKey
- 删除查询Delete Query
  - 执行查询后，删除或者更新该行
- 关键字段名称
  - 记录被处理数据的该值，以供检查是否重复处理



避免重复处理数据

- 记录数据库的一列为已处理的标志位（适合递增的记录）

  ^Ens.AppData("Training.BS.FromSQL","adapter.sqlparam","%LastKey")=210

- 记录处理过的记录

  以SSN为keyID的Sample.Person记录

  LEARNING>zw ^Ens.AppData

  ^Ens.AppData("Practice.BS.SQLFromSamples","adapter.sqlparam","%LastKey")="123-33-4444"

  ^Ens.AppData("Practice.BS.SQLFromSamples","adapter.sqlrow","101-56-6435")=1

  ^Ens.AppData("Practice.BS.SQLFromSamples","adapter.sqlrow","108-94-1237")=1

  ……

- 修改数据源标志位字段，查询标志位为未处理的记录。 多用于源数据修改触发的中间表记录

- PoolSize = 1， （否则会重复处理记录）

避免重复处理数据－使用KeyFieldName

- 查询全表， 对数据源压力大，效率最低
- 适配器记录所有查询过的KeyFieldName.
- 可以通过拼接字段(Compsite)来发现源表不同字段
  - select stats, Year||Month as ID…
  - KeyFieldName使用ID

Setting	Value
Query	`SELECT * FROM Customer`
DeleteQuery	*none*
KeyFieldName	`CustomerID`
Parameters	*none*

避免重复处理数据－使用&%LastKey或者%LastKey

- 对于源表中有单调增长字段的情况， 可以记录该字段最后处理的值。 查询源表时只查询部分记录, 查询效率提高。

- %LastKey
- PersistentValue in OnInit()

Setting	Value
Query	`SELECT * FROM Customer WHERE ID>?`
DeleteQuery	*none*
KeyFieldName	`CustomerID`
Parameters	`&%LastKey`

- 如果用&%LastKey, 处理时为了保证完全不重复处理
  - select * from Customer where ID>? order by customerID

避免重复处理数据－使用DeleteQuery

- 配置KeyFieldName和DeleteQuery
- DeleteQuery可能很慢，因为delet或者
  - 使用batch delete/updat可以提高效率

Setting	Value
Query	`SELECT * FROM Customer WHERE Done=0`
DeleteQuery	`UPDATE Customer SET Done=1 WHERE CustomerID=?`
KeyFieldName	`CustomerID`
Parameters	*none*

清除Adapter的处理记录

仅限于测试环境！！！

- **EnsLib.SQL.InboundAdapter**

  //每次BS重启该数据自动清除， 在哪?

  - ClassMethod ClearRuntimeAppData(pConfigName As %String)

  - ClassMethod ClearStaticAppData(pConfigName As %String)

  - ClearAllAppData()

SQL Outbound Adapter

创建使用SQL OutBoundAdapter的BO

- 创建BO， 使用EnsLib.SQL.OutBoundAdapter
- 在组件设置界面, 设置数据源，用户，密码等，
- 在BO方法里，通过用代码调用adapter命令实现SQL语句的执行
  - 增删改查
  - 执行存储过程

```
Method InsertPerson(pRequest As Practice.PersonCreateUpdate, Output pResponse As Ens.StringResponse) As %Status
{
        Set sqlString = "Insert into Sample.Person"_" (Name,SSN)"_" values(?,?)"
        Set tSC=..Adapter.ExecuteUpdate(.rows,sqlString, pRequest.Name, pRequest.SSN)
        Quit:$$$ISERR(tSC) tSC
        set pResponse=##class(Ens.StringResponse).%New()
        set pResponse.StringValue=" record intput"
        Quit tSC
}
```

执行查询操作

method ExecuteQuery(ByRef pRS As EnsLib.SQL.GatewayResultSet, pQueryStatement As %String, pParms...) as %Status

//如果driver不支持SQLDescribeParam
method ExecuteQueryParmArray(ByRef pRS As EnsLib.SQL.GatewayResultSet, pQueryStatement As %String, ByRef pParms) as %Status

执行插入，修改，删除

method ExecuteUpdate(Output pNumRowsAffected As %Integer, pUpdateStatement As %String, pParms...) as %Status

//如果driver不支持SQLDescribeParam
method ExecuteUpdateParmArray(Output pNumRowsAffected As %Integer, pUpdateStatement As %String, ByRef pParms) as %Status

执行存储过程

```
Method ExecuteProcedure(ByRef pResultSnapshots As %ListOfObjects,
        Output pOutputParms As %ListOfDataTypes,
        pQueryStatement As %String,
        pIO As %String = "",
        pInputParms...) As %Status
```

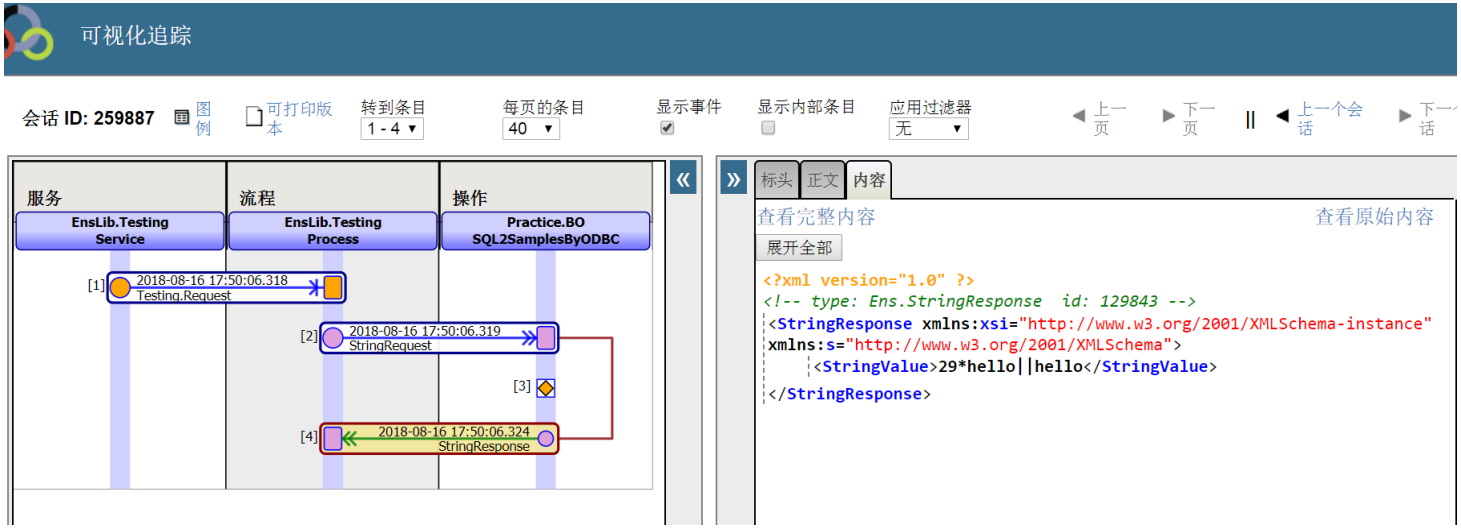
//如果需要定义多个参数而且ODBC driver不支持SQLDescribeParam

```
Method ExecuteProcedureParmArray(ByRef pResultSnapshots As %ListOfObjects, Output pOutputParms As %ListOfDataTypes,
        pQueryStatement As %String, pIO As %String = "", ByRef pIOParms) As %Status
```

举例说明:

```
 Set tQuery="{ ?=call Sample.Employee_StoredProcTest(?,?) }"
 Set tSC = ..Adapter.ExecuteProcedure(.tRTs,.tOutParms,tQuery,"oio",pReq.Parameters.GetAt(1))
 Set tRes.ParametersOut = tOutParms
```

```
Method TestProcedure(pRequest As Ens.StringRequest, pResponse As Ens.Response) As %Status
{
        Set tQuery="{ ?=call Sample.Employee_StoredProcTest(?,?) }"
         Set tSC = ..Adapter.ExecuteProcedure(.tRTs,.tOutParms,tQuery,"oio",,pRequest.StringValue,)
         set pResponse = ##class(Ens.StringResponse).%New()
        Set pResponse.StringValue = tOutParms.GetAt(1)_"*"_tOutParms.GetAt(2)
        Quit $$$OK
}
```



执行事务型操作

SetAutoCommit()
Commit ()
Rollback()

以上只适用于配置了DSN的情况，不要设置stayConnected=0(长连接)

```
Method TransactionExample(pRequest As common.examples.msgRequest2, Output pResponse As common.examples.msgResponse) As %Status

{ #Include %occStatus

try { //initialize variables and objects s

et tSC = $$$OK set pResponse = ##class(common.examples.msgResponse).%New()

#; start the transaction. Set autocommit to 0

set tSC = ..Adapter.SetAutoCommit(0) $$$ThrowOnError(tSC)
//Example UPDATE, INSERT, DELETE

set tQueryIns="insert into common_examples.mytable(name,age,datetime)" _" values ('SAMPLE"_$random(9999)_"',40,'"_$zdt($h,3)_"')"

set tSC = ..Adapter.ExecuteUpdate(.tAffectedRows,tQueryIns) $$$ThrowOnError(tSC)

// finalize transaction

set tSC=..Adapter.Commit() $$$ThrowOnError(tSC) }

catch err{ if (err.%ClassName(1)="common.err.exception") && ($$$ISERR(err.status))

{ set tSC = err.status } else { set tSC = $system.Status.Error(err.Code,err.Name,err.Location,err.InnerException) } set pResponse.status = tSC

do ..Adapter.Rollback() } Quit $$$OK }
```