四川省省级通用数据交换(含批量数据)服务管理平台

**过程记录**

**四川久远银海软件股份有限公司**

**2020年11月**

目录

[1. 项目管理 2](#_Toc5368)

[1.1 代码管理 2](#_Toc28138)

[1.2 文档管理 2](#_Toc15121)

[2. 开发问题 2](#_Toc1239)

[2.1 关于MirthConnect启动报错 2](#_Toc10311)

[2.2 下载的webStart.jnlp文件如何运行？ 2](#_Toc15971)

[2.3 MirthConnect官网资料或社区资料？ 3](#_Toc12983)

[2.4 MirthConnect汉化？ 4](#_Toc18930)

[2.5 MirthConnect的Docker版本？ 5](#_Toc7948)

[2.6 Docker安装问题？ 5](#_Toc10511)

[2.7 Mc3.9.0的docker镜像制作？ 5](#_Toc8710)

[2.8 Prometheus监控对接？ 6](#_Toc6479)

[2.9 数据交换通道监控？ 6](#_Toc22357)

[2.10 MirthConnect的基础使用？ 6](#_Toc4690)

[2.11 通道的代码调试？ 6](#_Toc8224)

[2.12 如何封装通用函数？ 6](#_Toc8345)

[2.13 数据交换通道是否需要页面交互？ 7](#_Toc12017)

[2.14 数据交换通道如何返回页面？ 7](#_Toc27731)

[2.15 页面之间如何跳转？ 7](#_Toc29663)

[2.16 多个destination如何控制？ 8](#_Toc29520)

[2.17 保证页面参数可复用性？ 9](#_Toc29487)

[2.18 提高数据交换性能？ 9](#_Toc30640)

[2.19 不同数据库互通？ 10](#_Toc8636)

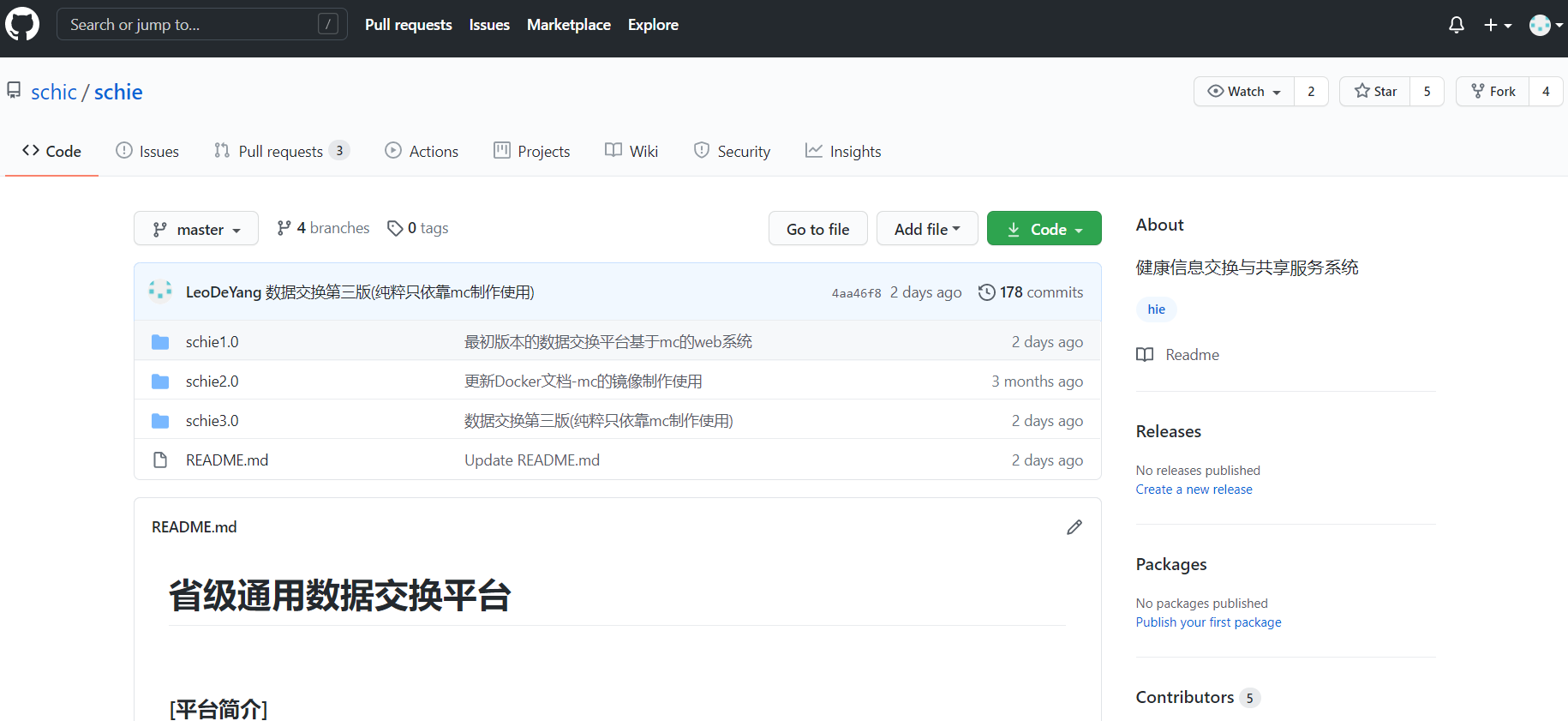
[2.20 数据交换增量更新方案？ 10](#_Toc28955)

# 

# 项目管理

## 代码管理

本项目应甲方要求可见可查共享协作的开发方式，将代码都集中上传至要求的github项目小组。如图：



如图所示，本项目可在github网站中直接搜索到，方便大家一同探索该技术方向。项目经历的过程schie1.0是第一版本，该版本是一套web管理系统，目的是通用数据批量交换，在MirthConnect的基础上结合Java的SpringBoot框架实现数据全量复制和定时增量更新。工程中schie2.0是第二版本，该版本是在schie1.0基础上更加的丰富了数据监控功能的一套web管理系统。最后schie3.0是第三版本，是应甲方简洁明了只需要围绕MirthConnect开发的要求下开发出的数据交换通道。该版本没有任何外部应用支持，整个业务逻辑、代码逻辑以及页面都建立在MirthConnect应用之上，以此来达到数据全量和增量交换。

## 文档管理

本项目工程相关文档同代码一起上传至github工程中。在整个开发过程中，形成了需求及功能设计、概要设计说明书、四川省省级通用数据交换(含批量数据)服务管理平台功能测试报告、四川省省级通用数据交换(含批量数据)服务管理平台性能测试报告、docker指南、MirthConnect指南、Prometheus监控指南、四川省省级通用数据交换(含批量数据)服务管理平台部署方案、四川省省级通用数据交换(含批量数据)服务管理平台开发文档、四川省省级通用数据交换(含批量数据)服务管理平台用户操作手册、数据交换(含批量数据)通道用户操作手册、数据交换(含批量数据)通道开发文档、数据交换(含批量数据)通道过程记录。

# 开发问题

## 关于MirthConnect启动报错

MirthConnect是由Java语言编写的软件，因此服务端运行需服务器或电脑必须安装好jdk环境。安装好的情况下，mc服务端仍然报错，那么只能反复重配整个Java环境并再次尝试。

## 下载的webStart.jnlp文件如何运行？

下载好mc客户端下载文件webStart.jnlp之后，查询资料经验证，右键打开方式，寻找到Java(TM) Web Start Launcher打开即可，该程序在jdk环境中。

## MirthConnect官网资料或社区资料？

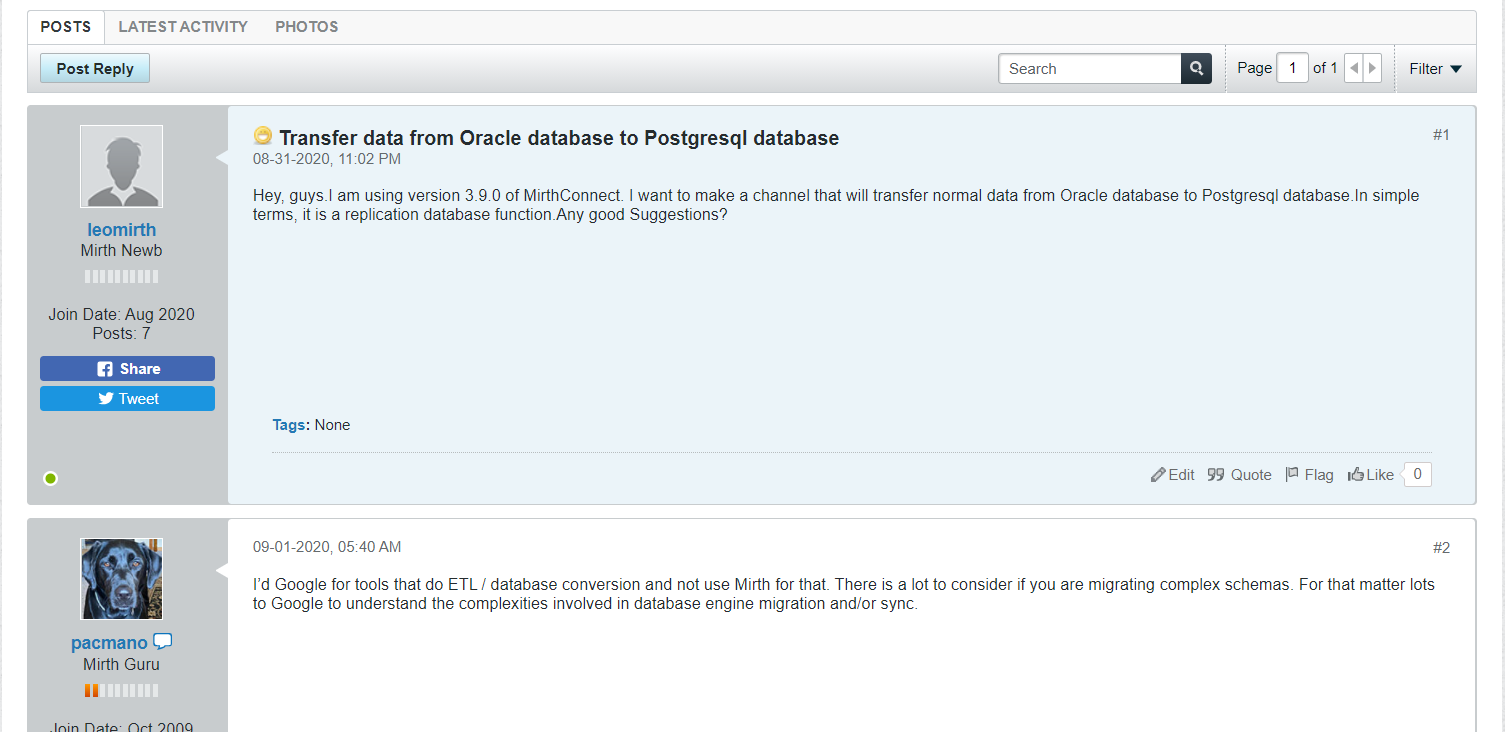
MirthConnect官网:

<http://www.mirthcorp.com/community/wiki/display/mirth/Getting+Started+Guide>

MirthConnect社区：

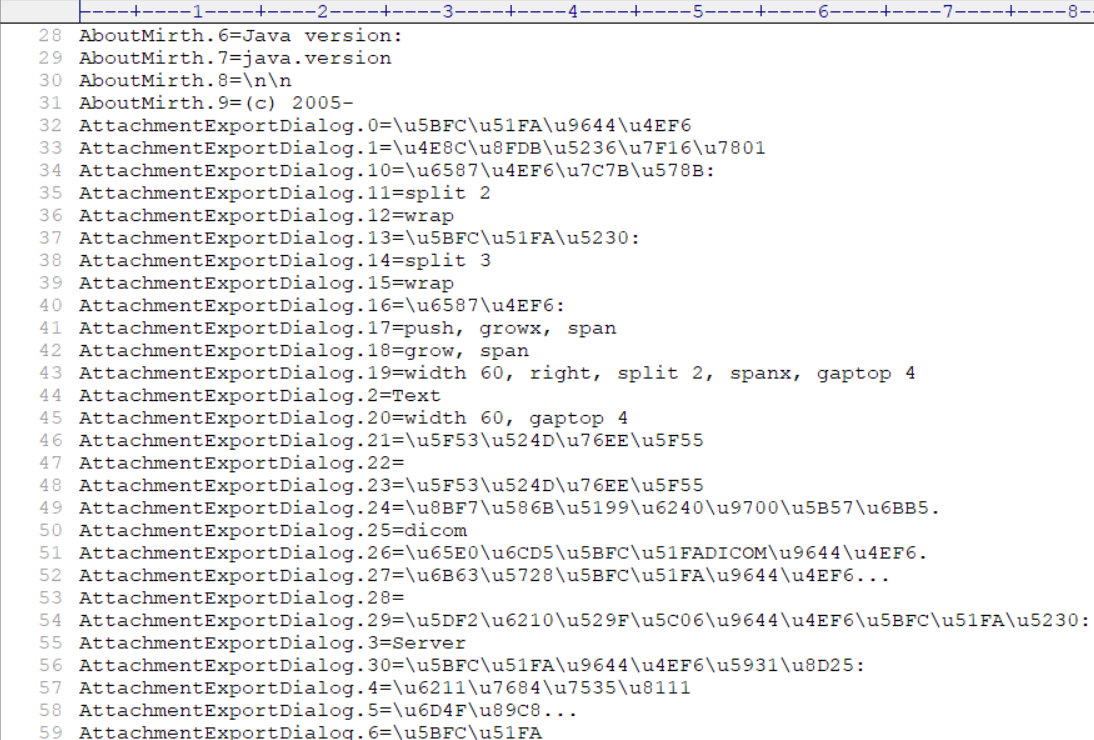
https://forums.mirthproject.io/register

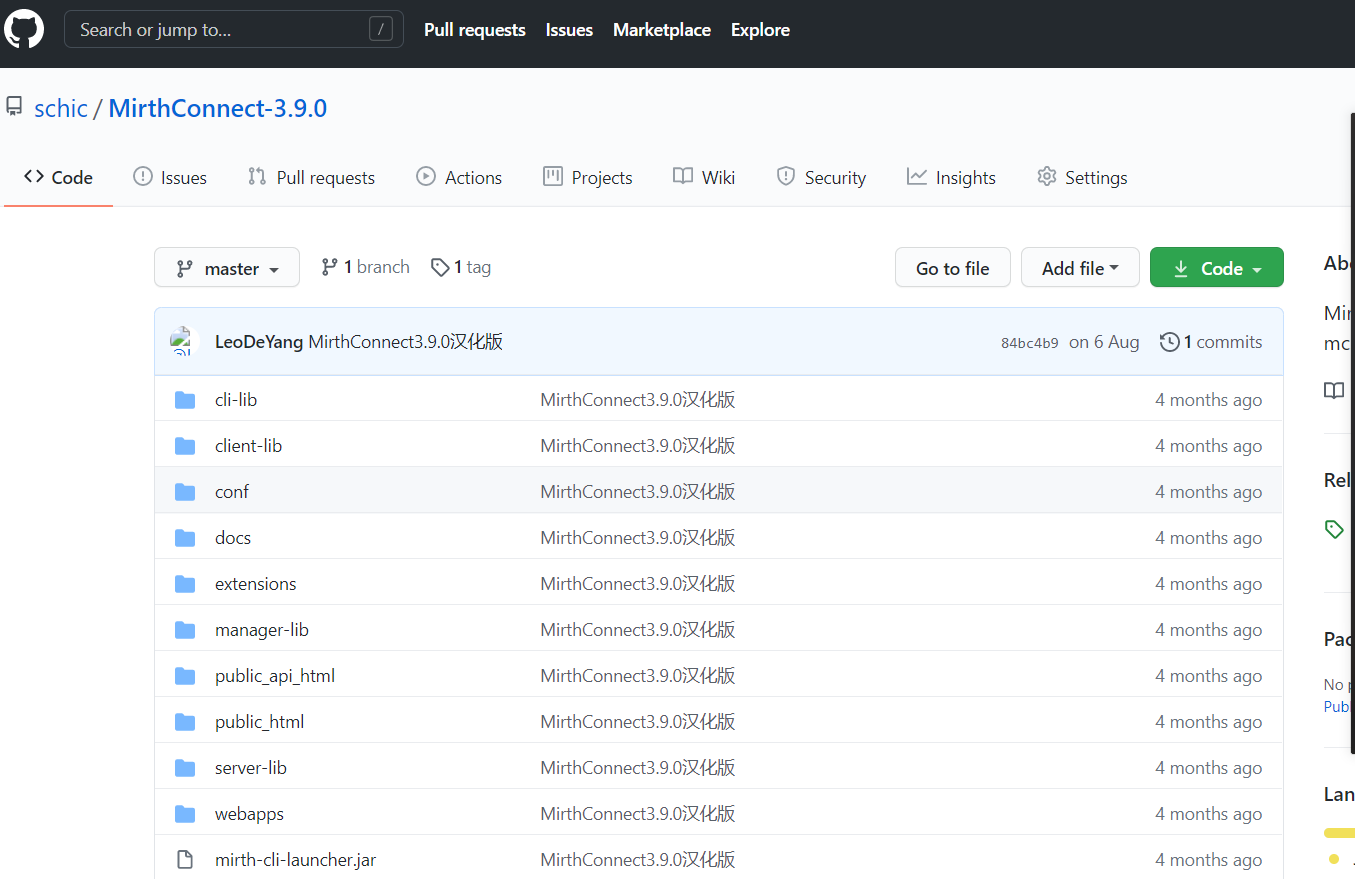
其社区可注册账号发帖交流，注册并不复杂，两个网站都属于外网，访问较慢，可以查询相关资料或提问。Mc的源码在svn库或者直接在github搜索下载。



## MirthConnect汉化？

MirthConnect汉化方式最初并没有任何资料也没有摸索出好的方式，最后经过林晓东博士指导。将mc3.9.0版本成功汉化并发布到github。





## MirthConnect的Docker版本？

Docker作为时下最流行的应用容器引擎之一，应要求查阅相关资料之后，制作出mc3.9汉化版本的mc\_docker,方便部署使用。同时结合学习经验制作经验编写了docker指南文档，并同步到github上。



## Docker安装问题？

Docker在Windows10家庭版的安装下是非常头疼的事情，经过反反复复安装卸载。最终还是安装到了自己的阿里云服务器上进行mc docker容器的制作。

## Mc3.9.0的docker镜像制作？

在经过对docker的学习摸索后，对docker已经熟练使用。但并没有类似相关mc的docker镜像比对，也没有任何解读，需要自己去摸索思考。最终反反复复编辑测试，还是制作出了mc3.9.0中文版的docker镜像。

## Prometheus监控对接？

Prometheus是(由go语言开发的)一套开源的报警时间序列数据库组合的监控系统，适合监控docker容器。随着它自身的发展，越来越多的产品公司制作出了对应的监控组件。比如，mysql、Oracle、Postgresql、华为云阿里云等等都有针对Prometheus开发出监控组件。针对MirthConnect，暂时并没有组件监控，监控组件的制作以及监控指标成为了最大的难题。最终在反复学习查询资料之后，并没有得到有效解决。只是编写了Prometheus监控指南文档。

## 数据交换通道监控？

目前mc自身的监控是客户端Dashboard标签面板提供，针对通道的来往信息，source和destination的转换信息有详细记录。Web页面上也有一个mc的监控，该监控只是针对通道状态和信息数量的统计监控。前者是针对通道详细数据信息的监控，后者只是对通道状态和信息量一个统计。那么，针对数据交换通道具体的数据监控该如何解决呢？经过反复思考尝试，只能做到在全量和增量数据之后做到数据交换的统计信息反馈。具体交换的数据、批量情况都无法做到实时监控。

## MirthConnect的基础使用？

在结合官网资料，社区资料以及mc指南PDF文档的帮助下，摸索前进。最终对mc的客户端基础使用有了初步了解，如何在新建通道、自定义通道、裁剪日志、不同Source和destination结合、通道交互等有了更深的认识。

## 通道的代码调试？

通道的代码编辑器属于js和Java混编，最终整个通道会形成一个脚本在mc中运行。运行中没有办法断点调试，因此最后只能是反复通过日志输出语句调试。

## 如何封装通用函数？

客户端左侧菜单有编辑代码标签，但是在编辑器中可用语言并不明白。尝试和摸索后，发现是JS和Java混编。渐渐上手之后，编辑好通用的函数后右侧勾选使用的地方。仍然最后还是报错了，经过反复检查发现，原来是最后使用函数的通道，依赖中需要再次勾选一次。

## 数据交换通道是否需要页面交互？

只依靠mc做出数据交换通道考虑的第一个问题即是需要页面或不需要页面？若不需要页面，那么使用者需熟知mc操作基础知识以及基本数据库知识，在mc的客户端来配置操作数据交换达到自己的交换目的。若需要页面，那么使用者只需要基本数据库知识，在页面上完成和使用者的沟通反馈。经过讨论后决定，还是需要用页面和用户沟通反馈。大大降低了使用者的难度。

## 数据交换通道如何返回页面？

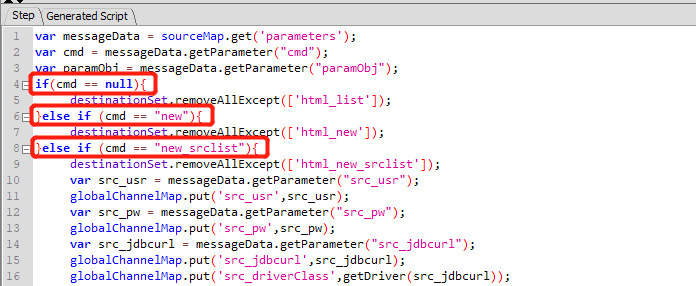
在mc这个应用上，是一个source对应着多个destination，在最终应返回一个响应。首先通道source类型选择到HTTP Listener最佳，因为毕竟最终需要在浏览器通过一个访问地址来访问到该通道做数据交换相关操作。既然是HTTP请求，有请求就应该有响应。在反复的查询资料和摸索后，经过林博士指导，用到的是destination的代码编辑器中写var resp = “” $r('ret',resp);然后在source的response标签里选择ret，这样的自定义响应方式来编辑响应。具体核心暂时未知。因此，返回到请求的页面就必须格式化为一个字符串，给到resp变量，然后放入响应。整个页面的格式化手动拼接修改，经过试验和反复调试，最终成功得到了想要的页面。在之后的反复调试中，仍需要反复返回来修改页面，重新拼接页面。

## 页面之间如何跳转？

在mc应用上，数据交换通道收到一个请求A，通道的destination A处理之后返回一个页面A。用户在返回的A页面上继续操作发出请求B，又该如何跳转？那么就需要提前在A页面上确定好url和参数确定好请求B的去向。因此，下一个页面的跳转修改是在A页面上，也就需要反复的修改格式化页面再写到destination A的编辑器中去。页面并非一个文件，而是编辑器中的一段拼接的字符串，这给调试代码带来了很大的难度。

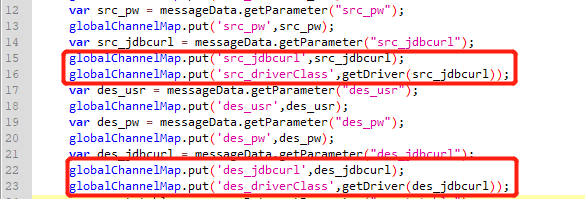
## 多个destination如何控制？

在数据交换通道中，为了该通道的复用性，让它通用起来。那么尽可能的将多个操作都糅合到一个通道中来。但是通道只有一个唯一入口source标签，对应着多个destination，如何区分本次请求去哪一个destination呢？反复查阅资料后，在FHIR的示例中找到了解决办法。首先需要在source标签的Edit Transformer中解析请求参数，通过参数命令的控制，然后用到由mc的jdk提供的destinationSet.removeAllExcept()方法，来确定本次请求对应处理的destination。最终形成一个请求一个source对应到一个destination处理返回。



## 保证页面参数可复用性？

在一般的web系统中，页面传递的参数传递到后台之后，要么保存起来，要么直接被使用掉且再次返回信息反馈给用户，达到良好的用户体验。在mc中，要做到这种效果，需要解决的问题就是存储参数和使用参数后的反馈。通过讨论后，设计一张配置表来存储数据交换任务。并依托mc自带map将一次请求的参数能够用到整个通道。如下图：





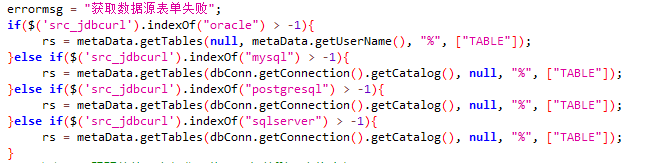
## 提高数据交换性能？

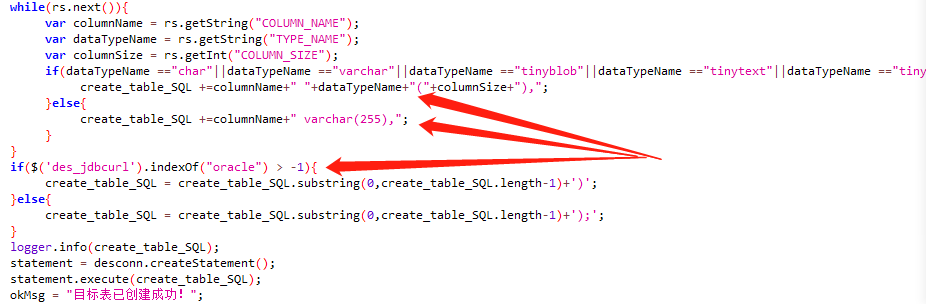
在数据交换通道中，数据之间交换，是用到的jdbc技术。数据是由Insert语句一条一条的插入到目标数据库中。为尽可能的提高性能，因此用到了事务批处理插入数据。每完成一张表数据交换提交一次，该表的数据量每1000条数据提交一次。如下图：



## 不同数据库互通？

在市面上的ETL工具中，基本都能满足不同数据库之间的数据传递和表结构复制。数据交换通道如何解决这一点？每个数据库SQL大体类似，但细微的差别导致并不能一句SQL通用所有类型数据库。因此，在交换通道中，只能适当的去适应不同数据库列出不同SQL语句。而且不同数据库之间的数据库字段略微有差别，为了达到保证目标表的结构和数据成功复制，也做了相应的处理，如下图：





## 数据交换增量更新方案？

数据交换的核心除了全量数据的传输，还有增量数据的同步。普遍处理方式有如下几种：

1. 基于全量数据对比得出增量数据。该方式需要将每一条数据逐一对比，在数据源数据库目标数据库及mc之间，这种频繁的I/O流操作对于大数据表，无疑就是直接降低整个性能。
2. 基于日志文件读取增量数据。数据源的数据库，对于增量的数据大部分都有日志记录，删除了哪些增加了哪些数据修改了哪些数据。该方案会对数据源源库的用户造成较大性能压力且需要大力的配合才可实现。
3. 基于时间字段切割区分增量数据。这是相比之下，成本较低速度较快的方式。只是因为没有日志支撑，无法获取到被删除的数据。