结项报告

项目信息

项目名称

为 DolphinScheduler 添加服务初始化的工作流 demo。

方案描述

用户在启动 DolphinScheduler 服务后，可以使用 demo-tool 程序预置工作流 demo，包括但不限于简单的 shell 任务、逻辑组件任务（switch、dependent、subprocess、condition）、参数传递、清除日志的功能，引导用户更方便地使用 DolphinScheduler。

启动教程

服务准备部分

本次开发使用Java进行程序的编写，DolphinScheduler服务可在Linux系统中搭建环境运行，快速学习DolphinScheduler具体功能可在官方文档中查看。

使用说明

提交PR前方案设计及操作

```
#在本地运行 demo功能 的时候，可以配置jvm参数，用于开启服务
-Ddemo=true
```

目标操作流程和界面：

1. 搭建环境完成后，选择在`编辑器`IDEA中修改`程序设置环境变量`，添加服务启动的JVM参数即可运行程序开启服务。

   ```
   VM options: -Ddemo=true
   ```

2. 默认初始化的工作流demo服务是建立在管理员用户中，打开服务启动的地址，根据提示输入管理员用户登录账号密码admin/dolphinscheduler123，点击`登录`。

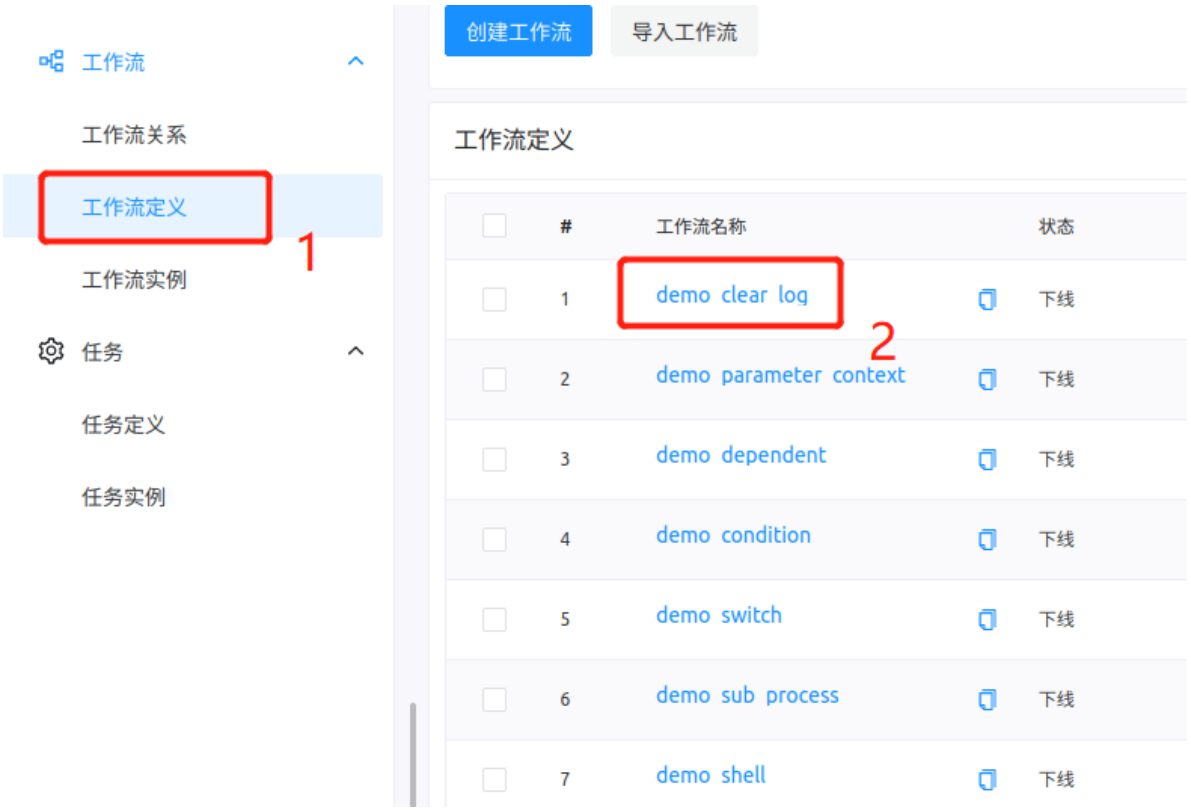
3. 认证成功后即进入软件主界面，找到安全中心租户管理进行创建租户，租户名称与操作系统用户名保持一致。

4. 创建完租户后就可以找到项目管理，点击项目名为demo的项目，进行demo工作流的操作。

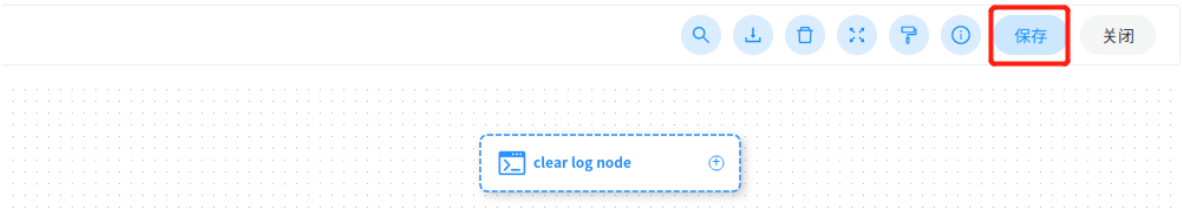
5. 然后找到工作流定义，在想要运行的demo工作流中选择其中一个点击，此时该项目中的demo工作流在服务启动时就创建好了。

其中demo服务所支持选择简单的 shell 任务、逻辑组件任务（switch、dependent、subprocess、condition）、参数传递、清除日志功能。对于demo工作流内容的代码编写，这里我和导师进行多次讨论，决定对demo case添加趣味性的形式，并设计具体通俗的实际场景应用样例如下：

demo工作流名称	**实际场景应用样例**
demo_shell	生产、加工、销售一系列工艺流程
demo_switch	根据条件确定要执行的任务
demo_dependent	检查日常工作的完成情况
demo_sub_process	启动外部生产线
demo_condition	掷硬币
demo_parameter_context	上游或下游任务节点参数传输
demo_clear_log	清除30天前的 DS 日志文件



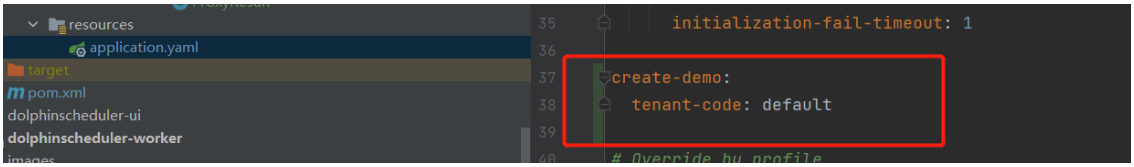
6. 随后可以点击`保存`按钮，修改租户信息改成之前新建的租户，在进一步`确定`后就支持对当前操作系统的管理员用户对该工作流的一系列操作。



提交PR后方案设计及操作

1. 根据社区成员的反馈，对于之前demo功能的启动存在一定的问题。因为demo功能开发逻辑不变，所以启动服务时的具体操作与提交Pr之前类似；在这基础上需要改变的是启动方式和设置租户信息变化，在和导师进一步讨论之后，决定通过创建tools启动类完成启动服务。
2. 选择在模块 dolphinscheduler-tools 下新建demo功能的启动类CreateProcessDemo，并通过创建请求代理工具类ProxyProcessDefinitionController，使用Http工具执行创建demo工作流定义的方法。
3. 在原先方案设计中需要手动创建当前操作系统的租户，提交Pr之后看到激活参数设置，我决定通过配置文件来创建租户，用户只需要直接修改配置文件代替手动创建租户的繁琐创建。在模块 dolphinscheduler-tools 配置文件中修改create-demo：tenant-code中的信息，替换成用户自己操作系统的名称。

```
#默认租户是default，用户可以根据自己操作系统用户名创建租户，从而代替手动创建
create-demo:
  tenant-code: default
```



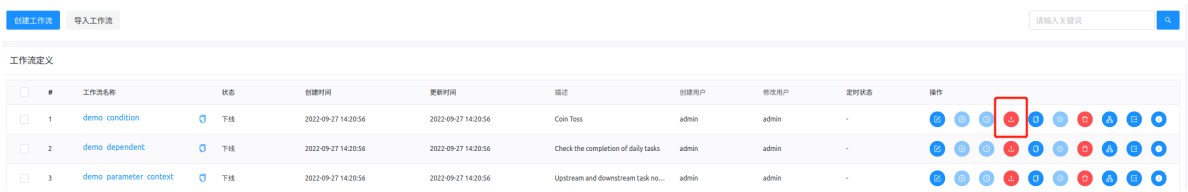
4. 在开发后续过程中，发现dev分支上的代码发生了变动，需要验证该tools环境，只有输入demo激活参数，才能把指定的环境注册到容器中，并让整个配置类里面的所有配置开始生效，同时也需要配置对应数据库信息，然后启动服务就可以实现预置工作流 demo 功能。

```
Active profiles:demo,mysql
```

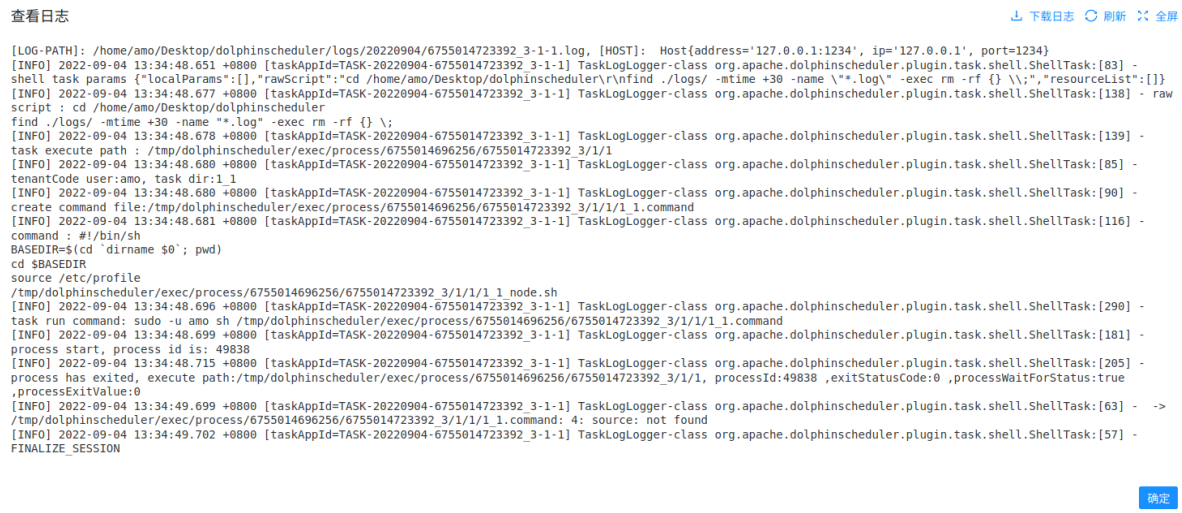
5. 同样用户也可以通过执行脚本，运行命令`sh ./tools/bin/demo-start.sh`开启该服务。

测试

使用不同服务器部署环境，选取各大操作系统镜像，都可以正常进行开启初始化服务，正常引导系统。测试中用户点击`上线`后都可以将其工作流转化为实例运行。



点击`查看日志`按钮，将开启服务后结果和写入好的demo功能对应起来可以提示运行成功，不对应则会正常提示运行失败。



其它交互过程中的测试都一切正常，无异常之处。

时间规划

- 第1周 (7.1 - 5, 2022)

1. 进一步了解DolphinScheduler，查阅相关资料。

- 第 2-6周 (7.6 - 8.9, 2022) 共计5周

1. 仿照DolphinScheduler官方文档，初步设计好工作流定义。
2. 进一步详细学习创建工作流全过程的原理。
3. 如果在此期间时间有剩余，可以帮助社区解决一些其它Issue，贡献社区。

- 第7周 (8.10 - 15, 2022)

1. 对用户使用DolphinScheduler系统常用功能进行调研。
2. 做好实现demo程序的准备工作。
3. 读懂系统的开发大体逻辑，进行开发文档的记录总结。

- 第8-13周 (8.17 - 9.24, 2022) 共计5周半

1. 依照初步设计好的工作流定义进行开发，实现初始化工作流的逻辑操作。
2. 进行图形界面的逻辑编写，实现完成服务启动脚本添加 init 模式功能。
3. 进行项目开发的文档记录，思考可以改进或者补充的地方。

- 第13-14周 (9.25 – 9.30, 2022)

1. 对完成内容进行总结，准备结项报告。
2. 为可能的拖延，预留的赶进度时间。

- 结项(9.30, 2022)

1. 为服务启动脚本添加init模式。可以完成预置工作流demo的程序。

- 未来 (10.1, 2022 -)

1. 在成功完成此暑期项目后，我将继续为DolphinScheduler做出贡献并尽我所能继续维护。

项目总结

项目产出

可在DolphinScheduler的standalone模式和普通模式开发环境下运行预置工作流demo程序，具备shell、逻辑组件任务（switch、 dependent、subprocess、condition）、参数传递、清除日志的功能，另外支持启动脚本添加init模式功能。已完成，项目成果提交仓库

方案进度

- 开发阶段第1周-第2周（2022/07/01-2022/07/15）0%

1. 学习创建工作流全过程的原理。

- 开发阶段第3周-第4周（2022/07/16-2022/07/31）20%

1. 实践工作流创建原理，并运行工作流实例（在Linux服务器上准备环境并进行测试）。
2. 初步进行demo工作流内容的方案设计。
3. 完成了一部分创建工作流方法的功能。

- 开发阶段第5周-第6周（2022/08/01-2022/08/15）40%

1. 进行预置工作流demo功能的实现。
2. 对各个工作流demo内容代码逻辑的编写。

- 空闲时间 (8.9 - 8.30, 2022)

1. 帮助社区解决一些其它Issue，类似修改官方文档中对于工作流的操作示例来提交PR。
2. 学习DolphinScheduler项目，提出一些Issue贡献社区。

- 开发阶段第7周-第8周（2022/08/16-2022/08/31）60%

1. 进一步完善和补充demo工作流的内容。
2. 完成代码逻辑的编写，完成预置工作流demo的tool程序。
3. 服务启动脚本添加init模式。

- 开发阶段第9周-第10周（2022/09/01-2022/09/15）80%

1. 完成程序绝大部分内容，提交PR。
2. 对程序进行测试。

- 开发阶段第11周-第12周（2022/09/16-2022/09/30）100%

1. 根据社区反馈对程序进行进一步修改，完成后进行测试提交PR。
2. 编写使用教程，完善相关文档。
3. 准备结项。

遇到的问题及解决方案

1. 学习阶段工作流实例创建不了，运行的工作流示例报错。=> 查看日志报错，非法参数异常，操作系统用户不一致，通过Linux服务器搭建环境；
2. 服务启动通过前端按钮操作过于繁琐和不必要 => 尝试通过直接调用JVM参数进行预置工作流demo来实现；
3. 写入工作流内容不确定，对于demo中的具体内容进行一个探究。=> demo case添加趣味性的形式；补充demo case，dependent任务、任务参数传递、shell清除30天前的ds日志文件；
4. 调用服务接口验证不通过，实例数据不清晰 => 验证则直接使用postman进行实现，通过创建token接口功能、Standalone切换元数据库实现调用测试；
5. 启动加载类报错，Spring管理的Service bean没有扫描到，导致无法进行实例化。=> 扫描具体的包再进行过滤，添加启动类的registry配置文件信息；

6. demo代码与生产代码结合起来，增加了可维护性的复杂性 => 把该代码移至模块dolphinscheduler-tools，不依赖生产代码，可在启动DS服务之后执行该tool程序；
7. 创建租户信息操作方式过于繁琐，在开启服务后需要手动添加租户 => 在tool程序启动配置文件中添加租户信息，创建方法读取配置从而代替手动创建租户；
8. 模块的不合理设计，tools服务不应该依赖于api-server => 直接去除api模块的依赖，设计请求代理工具类，并通过Http工具类直接调用创建工作流接口。

项目完成质量

项目总体来说完全实现了原有计划，完成情况较好，后续功能需要维护会及时处理，并会在已有的demo功能上添加新的内容。

与导师沟通及反馈情况

每双周定时开会讨论任务进度，开会结束撰写项目任务进展报告，导师都能及时给予积极反馈，并且给予下个阶段的项目开发目标，在这里感谢导师的悉心指导，导师的帮助对本次开发起到很大的作用。