并发向流程引擎发出命令请求

如何解决并发问题?

采用ThreadLocal

- 全局命令上下文Context中持有全局的commandConextThreadLocal对象

```
--- Context.java
protected static ThreadLocal<Stack<CommandContext>>
commandContextThreadLocal = new ThreadLocal();
```

- 命令上下文拦截器中CommandContextInteceptor,通过全局上下文Context中commandContextThreadLocal对象会获取当前线程命令上下文，如果不存在，则创建新的命令上下文

```
public <T> T execute(CommandConfig config, Command<T> command) {
       CommandContext context = Context.getCommandContext();
       boolean contextReused = false;
       //这里有个重用的问题，可以不考虑
       //如果上下文不存在，则创建，否则重用
       if (config.isContextReusePossible() && context != null && context.getExceptio
           log.debug("Valid context found. Reusing it for the current command '{}'",
           contextReused = true;
           context.setReused(true);
       } else {
           context = this.commandContextFactory.createCommandContext(command);
       }

       try {
           //设置当前上下文到当前线程的ThreadLocalMap中
           Context.setCommandContext(context);
           Context.setProcessEngineConfiguration(this.processEngineConfiguration);
           ...
           //传递给后继拦截器执行
           Object var5 = this.next.execute(config, command);
           return var5;
       } catch (Throwable var31) {
           context.exception(var31);
       } finally {
           try {
               if (!contextReused) {
                   //上下问关闭的时候，会刷新会话缓存到数据库，会执行一些命令上下问关闭监听器cl
                   context.close();
               }
```

```
            } finally {
                //责任链回溯时，移除当前命令上下文
                Context.removeCommandContext();
                Context.removeProcessEngineConfiguration();
                Context.removeBpmnOverrideContext();
                Context.removeActiviti5CompatibilityHandler();
            }

        }

        return null;
    }
```

- 在命令调用者CommandInvoker中，会获取当前线程的命令上下文，然后在CommandContext中持有流程虚拟机对象ActivitiEngineAgenda,执行当前命令拦截器中产生的Runnable任务

```
--- CommandContext.java
...
//上下文关闭拦截器
protected List<CommandContextCloseListener> closeListeners;
protected Map<String, Object> attributes;
//上下文是否可重用
protected boolean reused;
//流程虚拟机对象
protected ActivitiEngineAgenda agenda;
//和流程运转相关的执行流<id-->ExectutionEntity>
protected Map<String, ExecutionEntity> involvedExecutions = new HashMap(1);
//返回结果栈
protected LinkedList<Object> resultStack = new LinkedList();
```