

使用介绍：实现AI多轮对话功能

接上一篇《zPaaS低代码平台使用介绍：实现流程审批功能》，本篇主要介绍一下如何使用zPaaS低代码开发平台进行AI多轮对话功能的快速开发。

1. 开发的功能说明

本示例将基于ollama以及redis，通过后端业务逻辑编排实现AI多轮对话功能，并通过自定义功能模板实现简单的前端页面。

后端的大模型使用ollama运行qwen2.5:7b。

2. 前提条件

(1) 已经部署redis服务：

10.0.0.6:7001;10.0.0.6:7002;10.0.0.6:7003;10.0.0.6:7004;10.0.0.6:7005;10.0.0.6:7006

(2) 已经部署ollama服务且已经运行qwen2.5:7b模型：

http://10.0.0.6:11434

3. 准备工作

同《zPaaS低代码平台使用介绍：第一个功能开发》的准备工作，其中AI相关的功能及服务放到公共域Public。

(1) 在低代码平台中配置技术服务redis，如下图：

The screenshot displays the 'zPaaS低代码开发平台' (zPaaS Low-code Development Platform) interface. The top navigation bar includes the platform name, a '示例系统-勿修改' (Example System - Do Not Modify) warning, and a user profile icon. The left sidebar shows a tree view of the project structure with folders like 'Friends', 'Public', 'Resis', 'Studies', '数据库资源' (Database Resources), '资源资源' (Resource Resources), '技术服务' (Technical Services), and 'minio', 'ollama', 'redis'. The main content area is titled '技术服务 > 技术服务 > redis'. It contains a form for configuring the Redis service with the following fields: '标识' (Identifier) with value 'bfa86c0b-854f49e883a0e041584b8536', '名称' (Name) with value 'redis', '描述' (Description), '类型' (Type) with a dropdown menu, '用户名' (Username), '密码' (Password) with a masked input, '访问串' (Access String) with a long alphanumeric string, '服务资源地址' (Service Resource Address), '端口' (Port), '创建时间' (Creation Time) with value 'Jul 22, 2025, 1:47:50 AM', and '服务资源配置' (Service Resource Configuration) with a text area containing '(*ServerMode: "C")'. At the bottom right of the form are '保存' (Save) and '删除' (Delete) buttons.

(2) 在低代码平台中配置技术服务ollama，如下图：



(3) 在低代码平台中配置值传递对象AigcChatVo用于前后端传递数据



其中message属性为前端往后端传递的聊天消息；historyMessages属性为历史聊天消息。

4. 后端AI服务编排

(1) 使用工作台左上角的“+”按钮，创建服务对象AigcService



(2) 在该服务对象中创建多轮对话方法，如下所示：

zPaaS低代码开发平台 示例系统-勿修改

Public > 服务对象 > 服务对象 > AigcService-AigcService

标识: e68b73c68cd84370b3cd84196e871b65 编码: AigcService 名称: AigcService

对象类型: 服务对象 归属业务域: Public 描述: 保存 删除

对象方法 新建

编辑	删除	逻辑编辑	取消发布	multiAigcChat	名称	访问权限	业务流标识
				多轮对话			d3168c5dd05c4033bd527464aac44be

方法编辑

标识: 3caf8655bfde488c95a0784cd9e117fc 编码: multiAigcChat

名称: 多轮对话 状态: 有效

业务流标识: d3168c5dd05c4033bd527464aac44be

参数 新建

操作	编码	名称	是否入参	是否列表	参数类型
编辑 删除	message	消息	是	否	Java原生类型(java.lang.String)

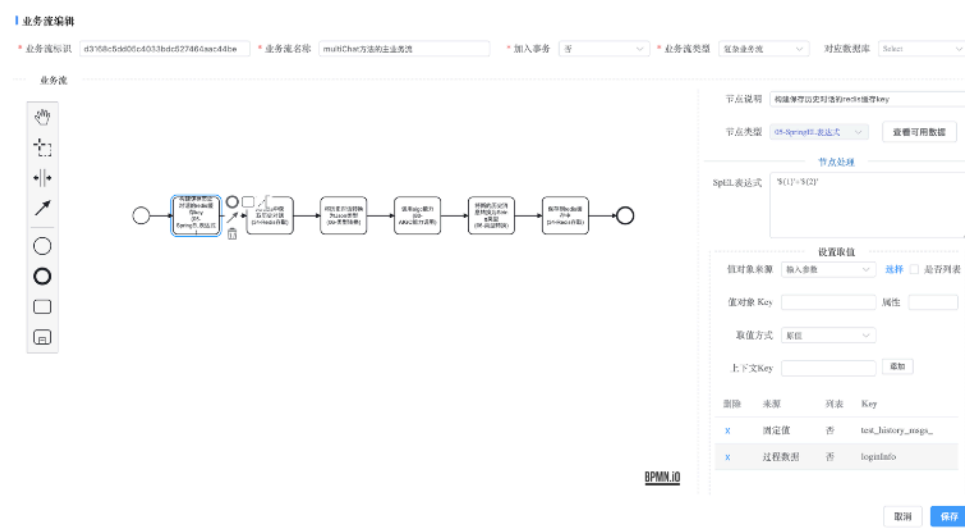
由入参校验规则设置

入参校验规则: 选择 清空

取消 保存

该方法只有一个message参数，没有返回值；返回信息通过流式的方式推送到前端。

(3) 编排该方法的业务逻辑，如下图所示：



(4) 第一步：构建保存历史对话的redis缓存key，使用SpringEL表达式节点。redis缓存key的组成格式为“test_history_msgs_”+“登录sessionId”；key生成后，使用“redisKey”作为关键字保存到过程数据中。

节点说明

构建保存历史对话的redis缓存key

节点类型

05-SpringEL-表达式

查看可用数据

节点处理

SpEL表达式

S(i)*S(i)

设置取值

值对象来源

输入参数

选择

是否列表

值对象Key

属性

取值方式

属性

上下文Key

添加

删除

来源

列表

Key

X

固定值

否

test_history_msgs_

X

过程数据

否

loginInfo

节点后处理

☒ 作为过程数据输出

存储关键字

redisKey

转换规则

设置 清空

☐ 作为领域对象输出

☐ 作为业务流结果输出

固定值	test_history_msgs_		原值
过程数据	loginInfo	_sessionId	原值

(5) 第二步：从redis中获取历史对话，使用redis存取节点。获取到的历史消息使用historyMsgs作为关键字存到过程数据中。

节点说明

从redis中获取历史对话

节点类型

34-Redis存取

查看可用数据

节点处理

redis命令

get

redis参数

redis

对象序列化

☐

设置参数列表

参数来源

输入参数

选择

是否列表

参数Key

属性

添加

删除

来源

列表

Key

属性

X

过程数据

否

redisKey

节点后处理

☒ 作为过程数据输出

存储关键字

historyMsgs

转换规则

设置 清空

☐ 作为领域对象输出

☐ 作为业务流结果输出

(6) 第三步：将历史对话转换为Json类型，使用类型转换节点。获取到的原始历史消息是字符串类型，需要转换为Json数组类型。转换好后的历史消息，覆盖过程数据中的historyMsgs。

节点说明

将历史消息转换为Json类型

节点类型

06-类型转换

查看可用数据

节点处理

源对象来源

过程数据

选择

源对象Key

historyMsgs

源对象属性

目标类型

JSON类型

执行结果类型

Select

☒ 是否列表

执行结果对象

选择 清空

节点后处理

☒ 作为过程数据输出

存储关键字

historyMsgs

转换规则

设置 清空

☐ 作为领域对象输出

☐ 作为业务流结果输出

- (7) 第四步：调用aigc能力，使用AIGC能力调用节点。
- 消息来源+消息Key+消息属性三元组确定最新的聊天消息
- 历史消息来源+历史消息Key+历史消息属性三元组确定历史聊天消息
- 用户名来源+用户名Key+用户名属性三元组确定发起聊天的用户名

助手名来源+助手名Key+助手名属性三元组确定显示的AIGC助手的名称

构成的聊天记录格式为：

`${用户名}: ${message}`

`${助手名}: AIGC服务返回的信息`

最新的历史消息覆盖过程数据中的historyMsgs。

节点说明: 调用AIGC能力

节点类型: 33-AIGC能力调用

节点处理:

- AIGC资源: vllama
- 模型名: gpt4o2.5-7b
- API类型: chat
- 模式调用: 是
- 模型参数:
- 消息来源: 传入参数
- 消息Key: message
- 消息属性:
- 历史消息来源: 过程数据
- 历史消息Key: historyMsgs
- 历史消息属性:
- 用户名来源: 过程数据
- 用户名Key: loginInfo
- 用户名属性: loginUser.realName

节点后处理:

- ☒ 作为过程数据输出
- 存储关键字: historyMsgs
- 转换规则:
- ☐ 作为领域对象输出
- ☐ 作为业务流结果输出

(8) 第五步：将新的历史消息转换为String类型，使用类型转换节点。转换好后的历史消息，覆盖过程数据中的historyMsgs。

节点说明: 将对象属性转换为String类型

节点类型: 06-类型转换

节点处理:

- 源对象来源: 过程数据
- 源对象Key: historyMsgs
- 源对象属性:
- 目标类型: 字符串类型
- 执行结果类型: Select
- 是否列表: ☐
- 执行结果对象:

节点后处理:

- ☒ 作为过程数据输出
- 存储关键字: historyMsgs
- 转换规则:
- ☐ 作为领域对象输出
- ☐ 作为业务流结果输出

(9) 第六步：将新的历史消息保存到redis缓存中，并设置有效期为120s，使用redis存取节点。

节点说明: 保存到redis缓存中

节点类型: 34-Redis存取

节点处理:

- redis命令: setex
- redis服务: redis
- 对象序列化: ☐
- 设置参数列表:
- 参数来源: 传入参数
- 参数Key:
- 属性:
- 删除:
- 来源:
- 列表:
- Key:
- 属性:

删除	来源	列表	Key	属性
X	过程数据	否	redisKey	
X	固定值	否	120	
X	过程数据	否	historyMsgs	

节点后处理:

- ☐ 作为过程数据输出
- ☐ 作为领域对象输出
- ☐ 作为业务流结果输出

(10) 发布该方法。目前流式返回的方法只支持“GET”的方式。

取消发布服务方法

发布标识: f3cd0cd25a7e4a70b8bb37c3909ee806 发布状态: 有效

服务编码: AigcService 方法编码: multiAigcChat

HttpMethod: GET 请求URL: /aigc/chat

创建时间: Jul 22, 2025, 2:32:29 AM 更新时间: Jul 22, 2025, 2:32:28 AM

取消 确定

5. 开发前端功能

(1) 开发《AI聊天》功能，同《zPaaS低代码平台使用介绍：第一个功能开发》。

zPaaS低代码开发平台 示例系统-勿修改

新建系统功能

名称: AI聊天 描述: 自定义功能

归属业务域: Public 功能模板: 自定义功能

创建方式: 根据领域对象 根据数据源表 根据数据对象

生成策略: 生成全部 生成配置 生成代码

功能模式: 动态配置 生成代码

绑定对象: 新建

操作: 绑定对象类型: 绑定对象: AigcChatVo 主对象: message

删除: 值传递对象

功能自定义页面: 新建从页面

页面名称: 自定义功能-主页面 是否主页面: 是

自定义功能-主页面: 是

取消 上一步 确定

zPaaS低代码开发平台 示例系统-勿修改

系统功能 > 系统功能 > AI聊天

标识: 42f00a0f00c34a2aaf75e5f5f550a087 名称: AI聊天 描述: 自定义功能

归属业务域: Public 功能模板: 自定义功能

绑定对象: 新建

操作: 对象绑定标识: 绑定对象类型: 绑定对象: AigcChatVo 主对象: message

删除: 可视觉设计

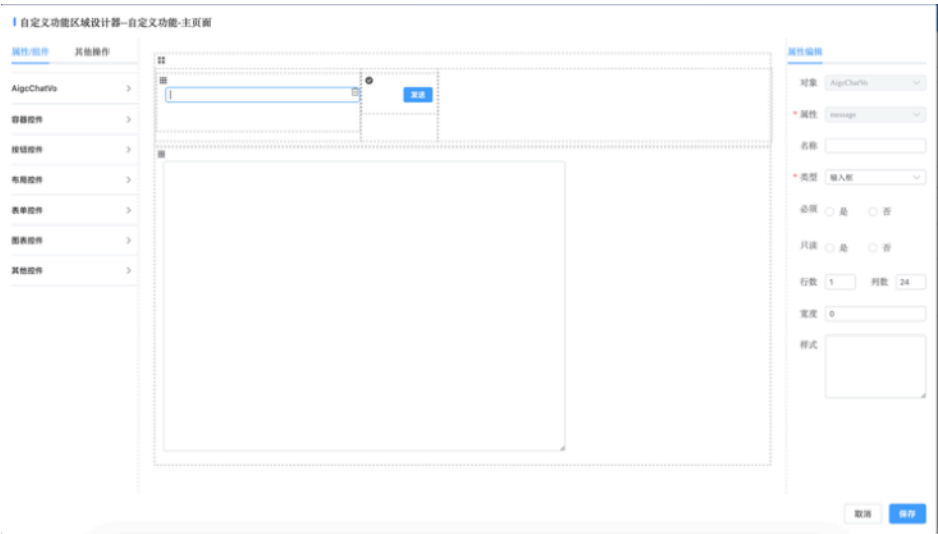
功能自定义页面: 新建从页面

页面标识: e8712d0a08f5457a489621d30e0eb1 页面名称: 自定义功能-主页面 是否主页面: 是

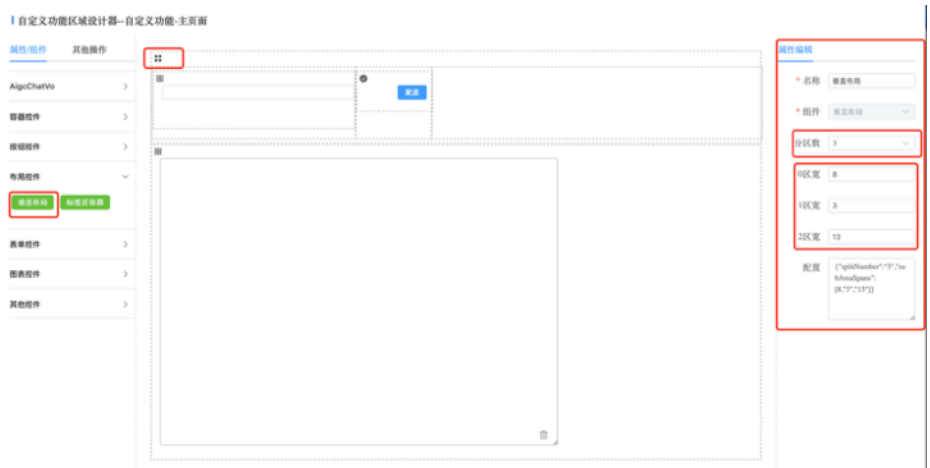
创建时间: Jul 22, 2025, 5:09:03 AM 更新时间: Jul 22, 2025, 5:09:03 AM

(2) 编排该功能的页面，该功能包含一个聊天消息输入框、一个发送按钮和一个历史消息展示框（文本域），如下所示：

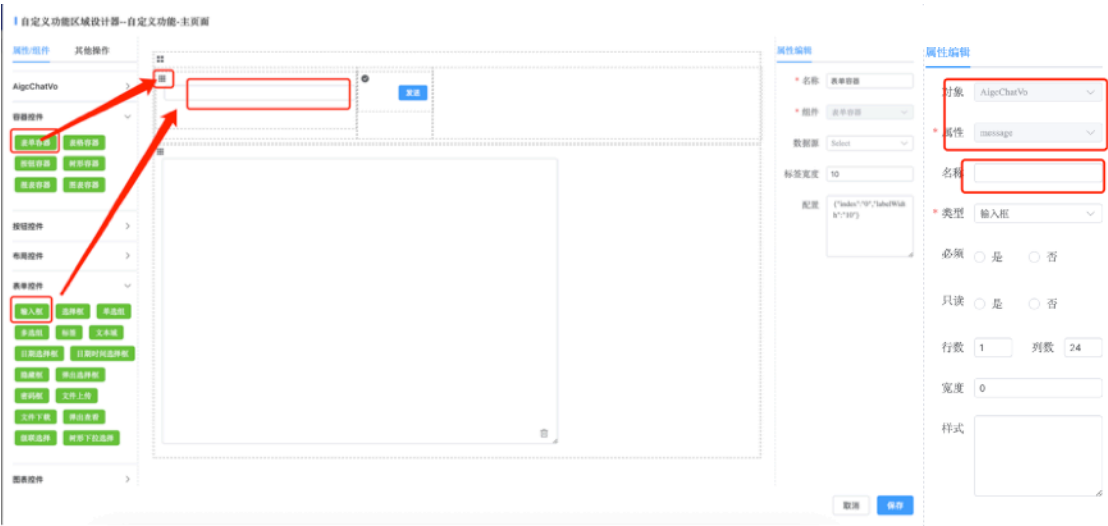
聊天消息输入框绑定AigcChatVo的message属性。历史消息展示框绑定AigcChatVo的historyMessages属性。



(3) 上半部先拖一个垂直布局组件，用于上半部页面的布局；设置三个分区，每个分布如下：

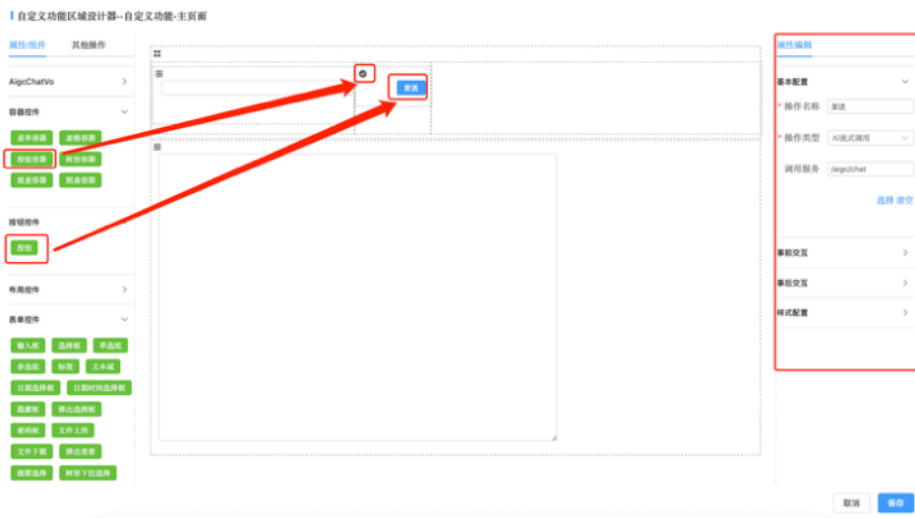


(4) 在第一个分区插入一个表单容器，并在表单容器中拖入一个输入框组件。



设置表彰容器的标签宽度为0；
绑定输入框到AigcChatVo的message属性，并清空名称属性。

(5) 在第二个分区拖入一个按钮容器，并在按钮容器中拖入一个按钮。



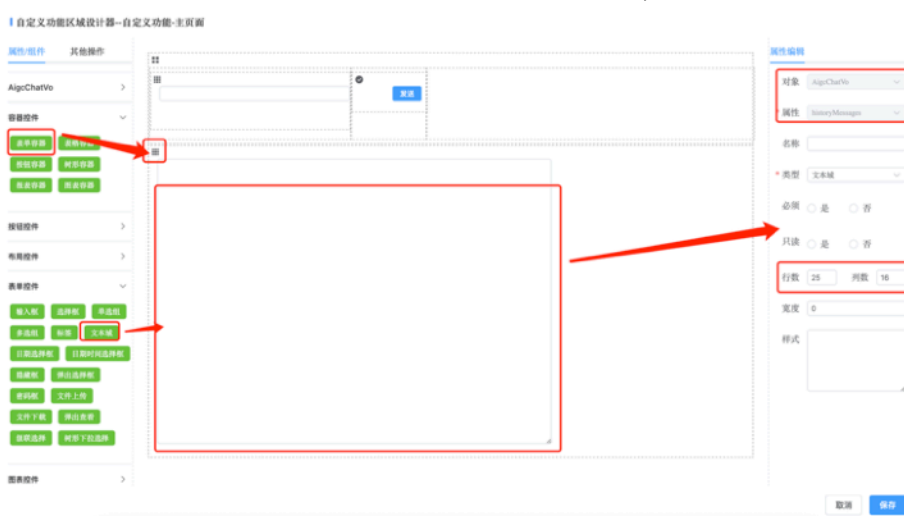
设置按钮的名称为“发送”，类型为“AI流式调用”，绑定后端服务“/aigc/chat”。

配置事前和事后交互：



其中回填映射中的“streamResponse”为约定的流式响应关键字。

(6) 在页面的下半部分拖入一个表单容器，并在表单容器中拖入一个文本域组件。



将该文本域组件绑定AigcChatVo的historyMessages属性。

(7) 刷新缓存后，可以预览该功能。最张效果如示例功能：



6. 其他

低代码平台的源码已经提交到GitHub：<https://github.com/zjyzju/zPaaS-lowcode>，后续将通过更多的文章，逐步对低代码平台的设计理念/思想、具体的设计、部署以及操作说明进行细化，敬请关注。

另外在一台云服务器上部署了一个演示环境，有兴趣的小伙伴可以通过演示环境进行试用：

访问地址：<https://129.153.118.144:11443>

试用账户：`demo/adl28m2@ma(a!*D2`

`demo1/adl28m2@ma(a!*D2`

注：（1）该演示环境部署在一台免费的云服务器上，且位于国外，在国内访问响应会比较慢

（2）系统管理子系统的菜单框架引用的@micro-zoe/micro-app@1.0.0-rc.24组件，该组件在不同浏览器中存在兼容性问题，苹果的safari浏览器访问一切正常，但是windows的Edge和谷歌的Chrome会存在页面刷不出来的情况，需要多等一会儿（有解决方案的可以邮件发给我，多谢！<https://github.com/jd-opensource/micro-app/issues/1559#issuecomment-2784997902>）。

如果有低代码平台以及AI相关的讨论，可以发送邮件到我的邮箱：zjyzju@163.com。