N-Gitea

基于Gitea的通知扩展

概要设计说明书

[v0.2]

北京航空航天大学计算机学院

二〇二〇年四月三十日

文档修改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 | 审核人 |
| 0.1 | 2020.4.30 |  | 初稿完成 | 孙维华 | 麦梓健 |
| 0.2 | 2020.5.2 | 全章节 | 字体格式、排版校对、内容润色 | 麦梓健 | 孙维华 |

目录

[1 引言 4](#_Toc39144046)

[1.1 编写目的 4](#_Toc39144047)

[1.2 背景 4](#_Toc39144048)

[1.3 定义 4](#_Toc39144049)

[1.4 参考资料 4](#_Toc39144050)

[2 总体设计 4](#_Toc39144051)

[2.1 需求规定 4](#_Toc39144052)

[2.2 运行环境 5](#_Toc39144053)

[2.3 总体结构 5](#_Toc39144054)

[2.4 处理流程 5](#_Toc39144055)

[2.5 功能需求和程序的关系 7](#_Toc39144056)

[2.6 人工处理过程 8](#_Toc39144057)

[3 接口设计 8](#_Toc39144058)

[3.1 用户接口 8](#_Toc39144059)

[3.2 外部接口 8](#_Toc39144060)

[3.3 内部接口 8](#_Toc39144061)

[4 数据结构设计 8](#_Toc39144062)

[4.1 逻辑结构设计 8](#_Toc39144063)

[4.2 物理结构设计 9](#_Toc39144064)

[4.3 数据结构与程序的关系 9](#_Toc39144065)

[5 系统出错处理设计 9](#_Toc39144066)

[5.1 出错信息 9](#_Toc39144067)

[5.2 补救措施 9](#_Toc39144068)

[5.3 系统维护设计 9](#_Toc39144069)

# 引言

## 编写目的

本部分在已有需求分析的基础上对Gitea的功能扩展做进一步的分析设计，细化待开发的功能概貌。预期读者为软件开发者和测试者。

## 背景

本项目基于Gitea的功能框架，对Gitea的通知模块进行了扩充。

## 定义

|  |  |
| --- | --- |
| 名词 | 解释 |
| gRPC | gRPC是一个开源的远程过程调用框架，用于服务器间的高效对接。通过可插拔接口有效连接不同语言编写的服务器，进行负载平衡、跟踪、运行状况检查和身份验证。默认情况下，gRPC通过一种轻便高效的结构化存储格式，对数据序列化。 |
| 部署者 | 部署者在受支持的操作系统上安装并配置Gitea的运行环境。对部署的Gitea能够提供的服务类型以及提供服务的方式进行选择和配置。 |
| 开发者 | 开发者根据用户要求完成相应软件系统或者系统中的软件部分的过程。 |
| 测试者 | 测试者在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估。 |
| 使用者 | 使用者是Gitea主要服务的用户群体。他们能够使用Gitea提供的大部分功能，并和其他用户进行交互。 |
| OpenID | OpenID 是一个以用户为中心的数字身份识别框架，它具有开放、分散性。本文中特指微信服务中用户的身份标识。 |
| URL | URL是因特网的万维网服务程序上用于指定信息位置的表示方法。 |
| HTTP1.1协议 | HTTP1.1( Hypertext Transfer Protocol Version 1.1 )是超文本传输协议版本1.1。它是用来在Internet上传送超文本的传送协议。它是运行在TCP/IP协议簇之上的HTTP应用协议，它可以使浏览器更加高效，使网络传输减少。 |
| Web框架 | Web框架（Web framework）是一种开发框架，用来支持动态网站、网络应用程序及网络服务的开发。其类型有基于请求的和基于组件的两种框架。 |

## 参考资料

1. Python 3.8.2 文档：<https://docs.python.org/zh-cn/3/>
2. Go语言官方文档：<https://golang.org/doc/>
3. Macaron文档：<https://go-macaron.com/>
4. Gitea文档：<https://docs.gitea.io/zh-cn/>
5. GB8567-88

# 总体设计

本节概述了功能扩展的总体设计以及功能模块划分和描述。

## 需求规定

依据需求规格说明书中的需求分析内容，功能扩展需要完成如下的功能需求：

1. 通知方式开发接口：作为一个统一的通知信息分发渠道，该接口向开发者提供灵活的通知方式扩展方法。开发者能够通过该接口，进行新通知方式的开发扩展，并且可以采取模块包的形式进行独立发布。另一方面，部署者和管理者能够动态地加载通知方式功能模块。
2. 微信通知模块：该模块作为应用上述通知方式管理接口的一个样例模块，即采取微信作为通知方式，提供了使用管理接口提供通知方式的一个范本。
3. 通知事件定义接口：向Gitea的开发者提供统一的方式向系统注册新的通知事件类型。
4. 通知事件管理：使用者可以按仓库对通知事件进行激活或取消，对于不同的通知事件可以选择不同的通知方式。

除了上述功能需求以外，功能扩展还需要兼容已有的通知功能，保持Gitea的跨平台特性。向Gitea添加的代码必须保证可测试性，与已有的代码保持风格一致。

## 运行环境

功能扩展依托于Gitea，因此必须满足Gitea的硬件和软件运行环境要求。

TODO：添加运行环境依赖表

## 总体结构

Go语言对于动态加载模块的支持并不完善，且缺乏跨平台特性的支持。为了更好地进行开发工作，功能扩展使用Python完成模块的加载、运行。Python代码向Gitea提供gRPC远程调用接口，Gitea通过调用Python代码完成通知的设置和分发功能。本文档的下述部分将Python部分简称为通知服务模块。

## 处理流程

功能扩展涉及到的流程有两种，本节具体描述了功能扩展运行时的处理流程。

1. 通知方式功能模块的加载和设置流程。

通知方式功能模块的加载由部署者和管理者进行。部署者可以通过Gitea的安装界面完成模块的加载，管理者可以在管理面板Web界面完成模块加载。本地模块文件被选择并上传后，Gitea会将加载模块请求送至通知服务模块。通知服务模块加载模块文件内容后，校验模块文件的正确性和规范性。通知方式功能模块加载成功后，Gitea会向通知服务模块请求对应模块的全局设置页面代码，渲染后传给前端。部署者完成设置内容填写后，Gitea将收到的设置内容传给通知服务模块，通知服务模块将调用通知方式功能模块的对应方法完成全局设置。图1.1描述了功能模块加载的整个过程，其中部署者和管理者被统称为用户实体。



图1.1通知方式功能模块加载数据流图

除了全局设置以外，Gitea的每一个使用者都需要完成对应模块的个人设置才能正常使用模块功能。一般流程如下：Gitea向通知服务模块请求对应模块的个人设置界面代码，渲染后传递给前端。使用者填写完成后Gitea将数据回传给通知服务模块。图1.2具体展现了上述流程。通知服务模块调用模块个人设置代码完成个人通知设置。但上述流程并不适用于某些特殊的通知方式。比如对于微信通知设置，Gitea需要使用者的OpenId完成通知的分发操作，但是使用者一般是不知道其OpenId，因此Gitea需要访问微信服务号，以完成个人OpenId的获取操作。为了满足这类模块需求，通知方式功能模块在全局设置时可向Gitea注册特定的URL连接，将Gitea作为代理，转发该URL请求中的数据至通知服务模块。这类特殊的功能模块能够通过注册URL的方式实现和外部的通信，完成使用者个人的通知设置。图1.3以微信为例描述了通过注册URL进行个人设置的过程。图中微信服务系统通过访问预先添加的URL传递关注用户的数据。



图1.2个人设置一般流程数据流图



图1.3微信通知个人设置数据流图

1. 通知事件分发流程

通知事件是由使用者动态产生的。使用者可以在通知事件设置界面中针对仓库激活对应的通知事件，并选择Gitea已经注册的通知方式完成通知。当有通知事件时，Gitea会根据事件类型和设置选择需要通知的使用者和对应的通知方式，然后将通知事件以及相应操作发送至通知服务模块。通知服务模块读取个人设置信息后相继调用对应功能模块的格式化和发送代码完成通知的发送。图1.4以工单创建事件为例描述了从使用者A触发通知到使用者B接收到通知推送的整个过程。



图1.4通知事件数据流图

## 功能需求和程序的关系

通知服务模块作为管理功能模块的部件，定义了功能模块的文件结构，并从代码实现上限制了功能模块需要实现的功能函数，因此基于该模块提供的开发接口，开发者可以用统一的方式去实现各种通知方式扩展。

从Gitea原有的通知框架出发，功能扩展基于gRPC协议重新定义了各个通知事件的类型以及通信内容规范。为了添加新的通知事件，开发者需要完成以下工作：

1. 添加新的通知事件类型标识ID，调整内容规范；
2. 在触发通知的代码处添加通知事件生成代码。

开发者通过上述统一的开发步骤即可向功能扩展添加新的通知事件。

Gitea通知事件管理模块提供了更为细致的通知事件管理。配合该模块，功能扩展向使用者提供了统一的通知事件管理页面，完成了对不同仓库的通知事件的管理。

## 人工处理过程

根据功能扩展的架构设计，通知服务模块从Gitea本身的架构中分离，使得部署者针对不同的通知方式需要完成额外的配置工作。首先，部署者需要修改Gitea的配置文件，分别为Gitea和通知服务模块分配监听地址和通信端口。在Gitea的启动操作也包含了启动通知服务模块这一步骤。

其次，对于一些特殊的通知方式功能模块，需要部署者或管理者手动在提供通知服务的外部实体中添加一些配置。以微信模块为例，启用该模块要求申请一个微信服务号，并在服务号设置中添加对应URL信息，以完成额外的信息数据交互。

# 接口设计

## 用户接口

功能扩展向用户提供了通知方式功能模块的加载和设置界面，后者包含全局设置界面以及个人设置界面。这些界面集成到了Gitea原有的Web框架中。对于通知事件，功能扩展提供了统一的事件设置界面，可以按仓库激活事件、选择通知方式。

## 外部接口

功能扩展通过HTTP1.1协议向使用者提供服务。

对于开发者，功能扩展提供了一系列预定义的开发接口以供开发者开发新的通知方式功能模块。

## 内部接口

内部主要有通知服务模块以及Gitea本身的数据交互，这部分交互通过gRPC协议实现远程方法调用。

# 数据结构设计

## 逻辑结构设计

1. 通知方式功能模块文件结构

表4.1通知方式模块文件结构

|  |  |
| --- | --- |
| 文件名称 | 描述 |
| config.json | 对功能模块的名称、版本、作者、全局属性、个人属性的描述文件 |
| assets/ | 功能模块的资源文件目录 |
| entry.py | 模块的功能函数实现文件，其中必须以指定的方式实现所有的功能函数 |
| others | 其他Python源文件 |

表1.1列举的所有文件通过zip压缩格式压缩，即可被通知服务模块使用。

1. 数据库结构

通知服务模块需要存储各个功能模块在config.json中提出的个人设置字段。数据库为通知服务模块提供了单独的一张通知属性表，其中仅默认包含了用户ID字段，其他字段在加载功能模块后动态添加。通知事件设置表记录了用户通知事件个性化设置的数据。

## 物理结构设计

表4.2通知属性表(NotificationAttribute)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 字段含义 | 类型 | 长度 | 默认值 | 允许空 | 主键 |
| 1 | ID | 用户ID | INT | 8 |  |  | 是 |

表4.3通知事件设置表(EventSetting)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 字段含义 | 类型 | 长度 | 默认值 | 允许空 | 主键 |
| 1 | ID | 编号 | INT | 8 |  |  | 是 |
| 2 | UID | 用户ID | INT | 8 |  |  |  |
| 3 | RepositoryID | 仓库编号 | INT | 8 |  |  |  |
| 4 | EventID | 事件编号 | INT | 8 |  |  |  |
| 5 | MethodID | 通知方式编号 | INT | 8 |  |  |  |

# 系统出错处理设计

自定义通知功能由于涉及到自定义模块的gRPC调用，以及在实现特定类型通知时向第三方额外的网络请求（例如微信通知需要请求微信API），使得系统容易在处理这些操作时产生异常，及时捕捉到这些异常并通过方便的方式供系统维护人员进行查看对于系统的正常运行十分重要。本节介绍自定义通知功能的异常处理设计。

## 异常记录

程序可能会在很多地方出现异常，异常的处理准则为：底层模块内产生的异常应该向上传递给调用方，上层模块统一处理异常并通过日志对异常进行记录。

系统支持7种日志等级，分别为：

* TRACE : 最低等级的日志
* DEBUG : 记录调试等级的日志
* INFO : 记录一般信息的日志等级
* WARN : 记录警告信息的日志等级
* CRITICAL : 记录严重信息的日志等级
* FATAL : 记录致命错误的日志等级

除了对日志进行分级，系统在日志记录时同样加入了对输出日志的代码位置、记录时间、函数调用栈的记录，方便进行错误追踪。

## 补救措施

对于系统产生的异常，除了使用日志进行记录，还需要尝试对异常进行补救。系统通过错误重试和错误回退机制提供了错误补救。

1. 重试机制

程序的某些异常可能是由于网络的抖动等一些外界随机因素造成，这个时候程序可以尝试重新执行操作，可以解决一些异常的情况。

1. 回退机制

某些情况下，程序由于缺少必要的数据而产生异常(例如发送微信通知时发现用户微信id不正确)，这时候可以尝试对当前操作进行回退。例如在微信通知无法正常发送时，可以回退到通过用户站内信或者邮件进行通知。

## 系统维护设计

系统通过日志过滤、自定义日志输出两方面为系统维护提供便利：

程序在代码中出现异常时根据问题的严重性选择不同日志等级进行日志记录，系统管理员可以设置关注的日志等级，低于关注等级的日志不会被实际记录。

同时日志系统被设计成允许多种日志输出形式。默认情况下日志以具有良好可读性的形式被记录到文件中。同时系统还支持设置邮件通知，当高于关注等级的日志输出时便将日志内容以邮件的形式发给维护人员；系统也支持输出结构化日志(例如json格式)，以便将日志导入其他系统(例如ELK日志分析系统)进行分析，最后日志系统允许开发者注册自定义的日志输出类型，方便开发者对日志进行更加自定义的控制。