**互联网开发培训教程**

**业务日志组件开发**

2019年



# 版权

©2019用友集团版权所有。

未经用友集团的书面许可，本文档描述任何整体或部分的内容不得被复制、复印、翻译或缩减以用于任何目的。本文档描述的内容在未经通知的情形下可能会发生改变，敬请留意。请注意：本文档描述的内容并不代表用友集团所做的承诺。

目录

[版权 2](#_Toc10596)

[第一章概述 4](#_Toc8423)

[1.1 业务场景 4](#_Toc20155)

[1.2 开发方案 4](#_Toc19417)

[第二章示例配置 5](#_Toc13733)

[2.1 开发环境配置 5](#_Toc13821)

[2.2 示例代码配置 5](#_Toc6192)

[2.3 日志模板 6](#_Toc12949)

[第三章代码开发示例 8](#_Toc8280)

[3.1 修改配置 8](#_Toc24680)

[3.2 添加业务日志注解 8](#_Toc924)

[3.3 配置日志模板映射文件 8](#_Toc12675)

# 概述

## 业务场景

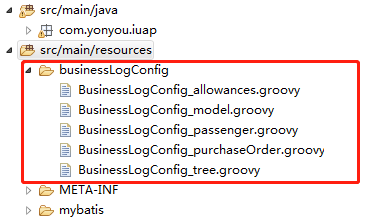
开发者经常遇到要保存操作日志信息的问题，IUAP提供便捷的保存日志方案，能让开发者仅需进行简单的配置即可实现此类需求。

## 开发方案

iuap-busilog组件采用Spring AOP拦截机制实现，能将记录日志与具体的业务方法解耦，日志模板采用groovy脚本配置方式，用户可自行定义输出参数，也可使用固定模板，输出到控制台或者写入到数据库中。

示例应用为busilog，主要代码工作为“配置”，代码结构如下：

* 代码结构：



# 示例配置

## 开发环境配置

1. 安装部署iuap-saas-busilog-service

首先完成iuap-saas-busilog-service应用工程的部署，此项工作已在IUAP一键式部署中实现，在此不再做详述。

1. 在工程中添加iuap-busilog组件的Maven依赖，具体如下：

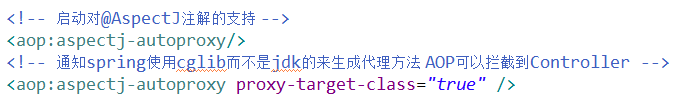
|  |
| --- |
| <dependency>  <groupId>com.yonyou.iuap.pap.busilog</groupId>  <artifactId>iuap-busilog-sdk</artifactId>  <version>${project.version}</version>  </dependency> |

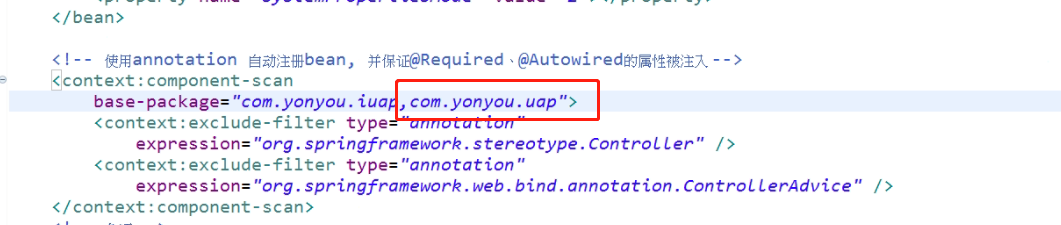
其中，${ project.version } 为平台在maven私服上发布的组件的version

## 示例代码配置

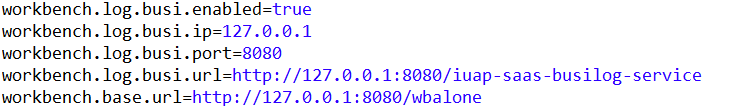
由于sdk使用LogRestWriter进行加签调用iuap-saas-busilog-service的服务，所以需要配置服务地址与加签相关信息

1. applicationContext.xml





1. application.properties



a) workbench.log.busi.enabled是业务日志输出的开关，当值为true时，会输出业务日志，当值为false时，不会输出业务日志；

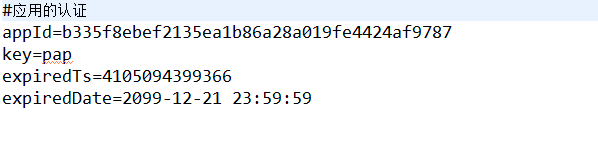
b) workbench.log.busi.url=http://ip:port/iuap-saas-busilog-service设置不为空时，优先使用

c) workbench.log.busi.ip在workbench.log.busi.url配置为空的情况下使用有效；用于配置业务日志服务的地址

d) workbench.log.busi.port在workbench.log.busi.url配置为空的情况下使用有效；用于配置业务日志服务的地址

e) 在workbench.log.busi.url与workbench.log.busi.ip及workbench.log.busi.port都不设置的情况下，默认值为127.0.0.1:8080；workbench.log.busi.enabled在没有设置的情况下，默认值为true

1. authfile.txt



## 日志模板

支持默认模板和自定义模板两种配置:

**默认模板：**

可以在不定义日志模板的情况下，可以使用默认模板，默认模板结构如下：

|  |
| --- |
| \_billName,执行: \_logOperation,IP地址:\_ip,USER用户:\_user,TIME操作时间:\_time执行结果: \_methodReturn |

**自定义模板：**

也可以通过groovy提供的模板文件定义的日志模板内容：

|  |
| --- |
| class BusinessLogConfig\_allowances {  def context;  def allowances\_save() {  "督办任务：执行保存方法:IP地址为${context.\_ip},USER用户为${context.\_user},TIME操作时间为${context.\_time},编码为${context.\_methodReturn.code},名称为${context.\_param0.name}"  }  } |

配置模板实际上是一个Groovy类，你可以在类中定义任何方法。如果方法为某个业务方法的别名（使用'@LogConfig'注解）那么，我们就认为它是一个业务日志方法。它的返回值（return或者放在方法最后一行的变量）将会被Set到'com.yonyou.uap.busilog.entity.LogDto'的实例中。日志方法返回值只返回一个String类型的日志文本；

在类中，还可以使用Groovy定义变量的方法：'def context' 定义一个变量。这个变量实际上是一个Map。 Map中存储的是业务方法的 '返回值'、'参数'。如果需要，你可以存储任何你需要的数据。你可以从这个context中取出你需要的内容，填充到你的日志中。至于如何取context中的内容，请看：   
在日志模板中取'context'的内容：

|  |  |
| --- | --- |
| key | value |
| \_methodReturn | 业务方法返回值 |
| \_param | 业务方法的参数, \_param0代表第一个参数 \_param1代表第二个参数，依此类推 |
| \_executeError | 业务方法执行失败的异常信息 |
| \_businessMethod | 业务方法 |
| \_user | 业务方法操作人 |
| \_time | 业务方法操作时间 |
| \_ip | ip地址 |

多文件配置：在类路径中加入'businessLogConfig'文件夹。在该文件夹中加入日志配置文件，文件名任意，只要符合Groovy类文件的命名规范即可。   
注：支持多文件配置，businessLogConfig'文件夹中的所有以'.groovy'结尾的文件都将被作为日志配置文件。

# 代码开发示例

## 修改配置

application.properties

|  |
| --- |
| workbench.log.busi.enabled=true  workbench.log.busi.url=http://127.0.0.1:8888/iuap-saas-busilog-service |

ApplicationContext.xml

|  |
| --- |
| <!-- 启动对@AspectJ注解的支持 -->  <aop:aspectj-autoproxy/>  <!-- 通知spring使用cglib而不是jdk的来生成代理方法 AOP可以拦截到Controller -->  <aop:aspectj-autoproxy proxy-target-class="true" />  <!-- 使用annotation 自动注册bean, 并保证@Required、@Autowired的属性被注入 -->  <context:component-scan  base-package="com.yonyou.uap,com.yonyou.iuap ">  <context:exclude-filter type="annotation"  expression="org.springframework.stereotype.Controller" />  <context:exclude-filter type="annotation"  expression="org.springframework.web.bind.annotation.ControllerAdvice" />  </context:component-scan> |

## 添加业务日志注解

为业务方法DictionaryService.save添加业务日志注解—

@BusiLogConfig("dictionary\_save")

|  |
| --- |
| @LogConfig(busiCode = "allowances\_saveMultiple", busiName = "员工津贴",operation = "员工津贴批量添加", templateId = "allowances\_saveMultiple")  @Transactional  **publicvoid**saveMultiple(List<Allowances>listData) {  **for** (**int**i = 0; i<listData.size(); i++) {  **this**.save(listData.get(i));  }  } |

## 配置日志模板映射文件

映射文件采用groovy文件。目前支持两种配置方式：

* **单文件配置**

在类路径下加入'BusinessLogConfig.groovy'

* **多文件配置**

1. 在类路径中加入'businessLogConfig'文件夹
2. 在该文件夹中加入日志映射配置文件

BusinessLogConfig.groovy

其中，文件名任意，但必须符合Groovy类文件命名规范

注：多文件配置方式与单文件配置方式不兼容。在此业务日志组件中，单文件配置方式优先。businessLogConfig文件夹中的所有以'.groovy'结尾的文件都将被作为日志配置文件。

* **groovy文件内容示例**

|  |
| --- |
| class BusinessLogConfig\_allowances {  def context;  defallowances\_saveMultiple() {  "员工津贴：执行批量添加方法:IP地址为${context.\_ip},USER用户为${context.\_user},TIME操作时间为${context.\_time}"  }  } |

1. 声明context

业务日志组件将把使用了 @BusiLogConfig的业务方法返回值、参数、当前用户等信息写入上下文context中，以便于记录业务日志使用。

1. 声明业务日志映射

“def allowances\_saveMultiple(){…}” 中方法名应该与annotation的声明保持一致，均为“templateId = "allowances\_saveMultiple"”。