Gears properties manager for jira

本页内容

下载地址点击这里

插件作用

Gears properties manager主要目标是管理系统级别的、全局性的参数；

在一些开发能力较强的企业中，对于在各个插件中使用一些全局性的变量很不方便，比如在ScriptRunner中，经常通过自定义的一些脚本写入到工作流中，获得业务数据后向第三方系统进行发送。第三方的信息可能包括一些，URL,账户名和密码，或者其它的参数，当这些参数写的较多的时候，对于如要修改接口的URL地址，变更密码，那么修改这些工作流的工作量将是一个挑战。

因此，此插件的使用是定义全局性的参数，可以将之引用到所需要的任何地址，只需要记住其变更的键值，那么将会产生一处修改，全部修改的效果。

同时，因为对这些参数进行了全局性的可视化管理，在运营和运维中更加方便，只需要在系统中进行查询，而不用再问具体的开发人员是如何定义的。

使用场景

比如我们在ScriptRunner的定时任务、字段、PostFuction中写入了大量的脚本，这个脚本使用了一个自定义字段（开发负责 人）并使用它的值进行很多业务处理。

在这种情况下，我们写入了多个脚本并放在不同的地方，通常情况我们会在脚本里写这个自定义字段的获取值，

```
customFieldManager.getCustomFieldObjectByName ("开发负责人")
```

假如使用在这种场景下，开发负责人在业务上说需要修改一个名称变为：开发责任人，那么我们就需要将系统中所有用到这段代码的脚本都找到，来修改它的值。

或者我们通过以下方式这种方式来写,

```
customFieldManager.getCustomFieldObjec (123456L)
```

在这种情况下也有可能这个字段被不小心删除，又重新创建；那么它的ID值也会发生变化，依然需要需要将系统中所有用到这段代码的脚本都找到，来修改它的值。

或者有以下情况，这个ID值或者名称，在我们的生产环境 或者测试环境并不一致，这样也会导致我们在测试环境 调试好的脚本，在拿到生产环境的脚本上一一变更这个值才能运行起来。

针对以上的上场景，使用此插件我们可以在里面定义一个固定的变量，这个变量的值可以快速的进行可视化的管理，修改这个值后系统中所有用一这个变更的全部都会生效，这样就方便我们统一来管理这些变量，达到可视化、全局化来管理我们系统中到处使用的变量脚本。

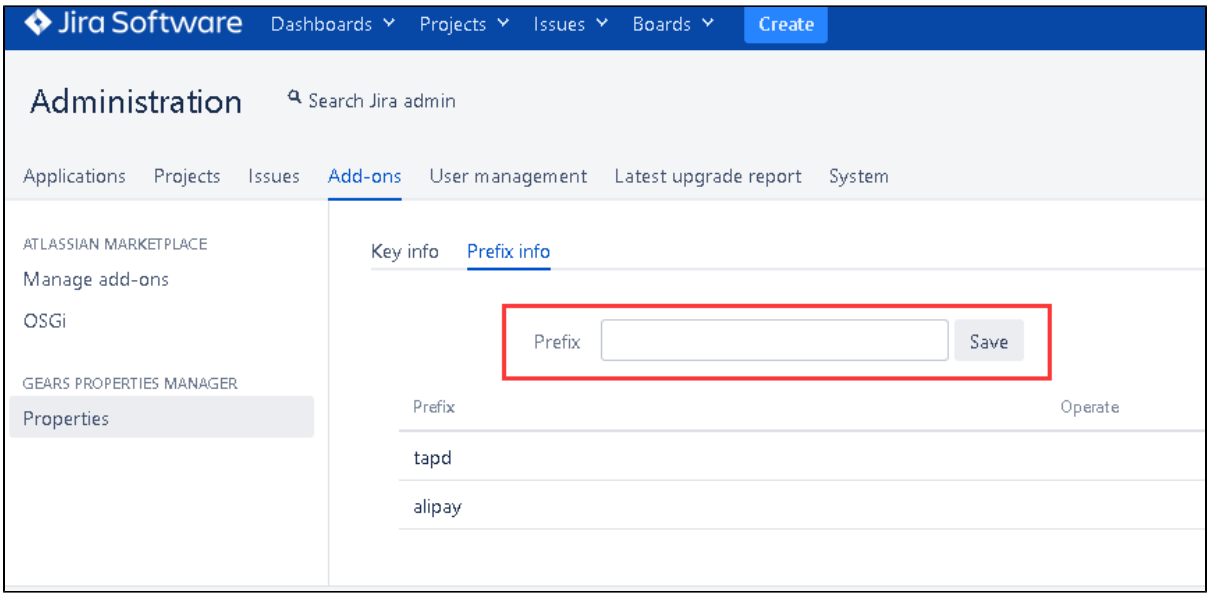
参数管理

Gears properties manager为了能够对全局性的参数进行科学管理，并方便管理人进行维护，比如管理员需要写入多个与第三方系统进行对接的参数，那么他可以将这些参数进行分组；

因此，对全局性分组，在此插件中将以前缀来进行区别；那么在对参数进行管理的时候，首先需要确定此参数应当归属哪一个前缀。

前缀信息

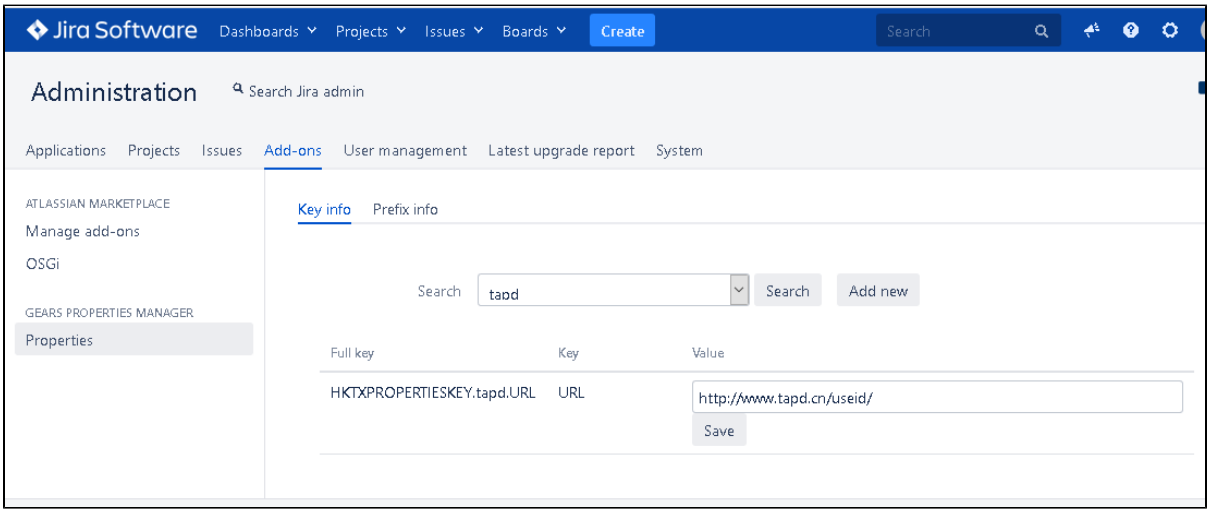
前缀信息是对全局性参数进行分组的一个定义，因此需要进行前缀的管理。



可以转到前缀信息中，对前缀进行增加管理。保存后将显示系统中已定义的前缀信息。

键值信息

键值信息是提供给业务使用的有效信息，允许在相应前缀中增加一个键值对，以便在系统中通过全键获取信息。



可以选择系统中定义的前缀查询此前缀中包括的所有键值对，当然对某一个键值对也可以进行修改操作。

如果需要在一个前缀中新增加一个键值对，那么可以新增一个，此时我们点击“新增”，即可弹出新增界面进行填写。

键值对使用

当对键值对进行定义新增加之后，那么可以在需要的地方获得它的信息，以便开展业务上的处理。

使用是，可以使用此类com.atlassian.jira.config.properties.ApplicationProperties，并使用它的getString()方法。

键值对使用
String value = applicationProperties.getString($fullkey) ;

```
 ## fullkeykey
++
```

REST API

获得一个前缀所有的参数信息

- 接口

/rest/gearsproperties/1.0/properties/prefix/{prefixkey}

- 方法

get

- 示例

/rest/gearsproperties/1.0/properties/prefix/ITDESK

```
{
"fullkey": "HKTXPROPERTIESKEY.ITDESK.StaffId",
"prefix": "ITDESK",
"key": "StaffId",
"value": "1123"
},
{
"fullkey": "HKTXPROPERTIESKEY.ITDESK.key1",
"prefix": "ITDESK",
"key": "key1",
"value": "value11"
},
{
"fullkey": "HKTXPROPERTIESKEY.ITDESK.key2",
"prefix": "ITDESK",
"key": "key2",
"value": "value22"
}
]
```


获得一个全Key参数信息

- 接口

/rest/gearsproperties/1.0/properties/fullkey/{fullkey}

- 方法

get

- 示例

/rest/gearsproperties/1.0/properties/fullkey/HKTXPROPERTIESKEY.ITDESK.StaffId

[


```
{
"fullkey": "HKTXPROPERTIESKEY.ITDESK.StaffId",
"value": "1123"
}
```


使用示例

我们还以“使用场景”中的示例来进行如何使用此插件。

定义变量值

首先我们在property插件中定义一个变量值（假定前缀是：SCRITRUNNER，KEY值为：DEV_OWNER），那么它的Fullkey即为：HKTXPROPERTIESKEY.SCRITRUNNER.DEV_OWNER ，其值为：123456

脚本中取

```
import com.atlassian.jira.config.properties.ApplicationProperties



//CustomField devOwnerCustomfield = customFieldManager.getCustomFieldObjec (123456L) 将改造为：

String devOwnerCustomfieldId = applicationProperties.getString("HKTXPROPERTIESKEY.SCRITRUNNER.DEV_OWNER") ;

CustomField devOwnerCustomfield =  customFieldManager.getCustomFieldObject(Long.parseLong(devOwnerCustomfieldId));

ApplicationUser devOwnerUser = issue.getCustomFieldValue(devOwnerCustomfield);
```

业务修改

当我们在业务上需要调整，需要将原来取“开发负责人”（ID为：123456），修改为去取“开发责任人”（ID为：654321）或者其它的值时，我们只需要在properties插件中，修改：SCRITRUNNER.DEV_OWNER 的值为 654321，即可完成业务的变更。

如果这类业务逻辑被 应用到N个脚本，我们只需要修改一处，即可完成N处的修改，减少了修改的工作量，也不至于漏掉某处而导致业务异常。