Idea Live Templates 代码模板

程序员的工作不是写程序，而是写程序解决问题。

我们每天都在写代码，有些代码有结构性的相似，但不是所有的代码都可以被抽成方法。在这种情况下，我们应该考虑使用 template 的方式加快我们的开发速度。这篇文章会先介绍 IntelliJ 下已经有的一些 code/live template，然后介绍如何自定义 template。

1.IntelliJ 下已有的 template

在 Java 中，我们时常会遇到遍历一个 iterable 的情况。如下所示：

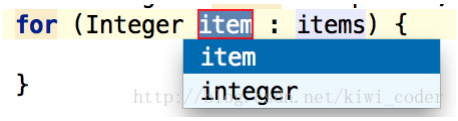
```
for (Integer item : items) {
    //...
}
```

你可以选择一个字母，一个字母的敲，但是在 IntelliJ 下面，你只需要输入 iter 再按 TAB 键，就可以自动生成这个代码，和原来的代码比起来，你需要敲键盘的次数少了几倍。

```
for (Integer item : items) {

}
```


这个时候，红色的框框部分就是当前你的编辑区域，因为我之前有一个 items 的变量，这里 IntelliJ 会建议 Iterable 是 items。如果你觉得不对也可以修改，确认后敲 Enter。



敲了回车以后，就进入下一个编辑的部分，而不需要你手动的去移动光标。确实很 Intelligent 啊。

IntelliJ 中提供了很多现有的 Code/Live Template，你可以在 Settings 中找到这些已有的 template：

比如 sout 可以输出 System.out.println(...), ifn 可以输入 if (xxx == null) 等等。学习这些 template，并且灵活运用，可以缩短你敲键盘的速度。

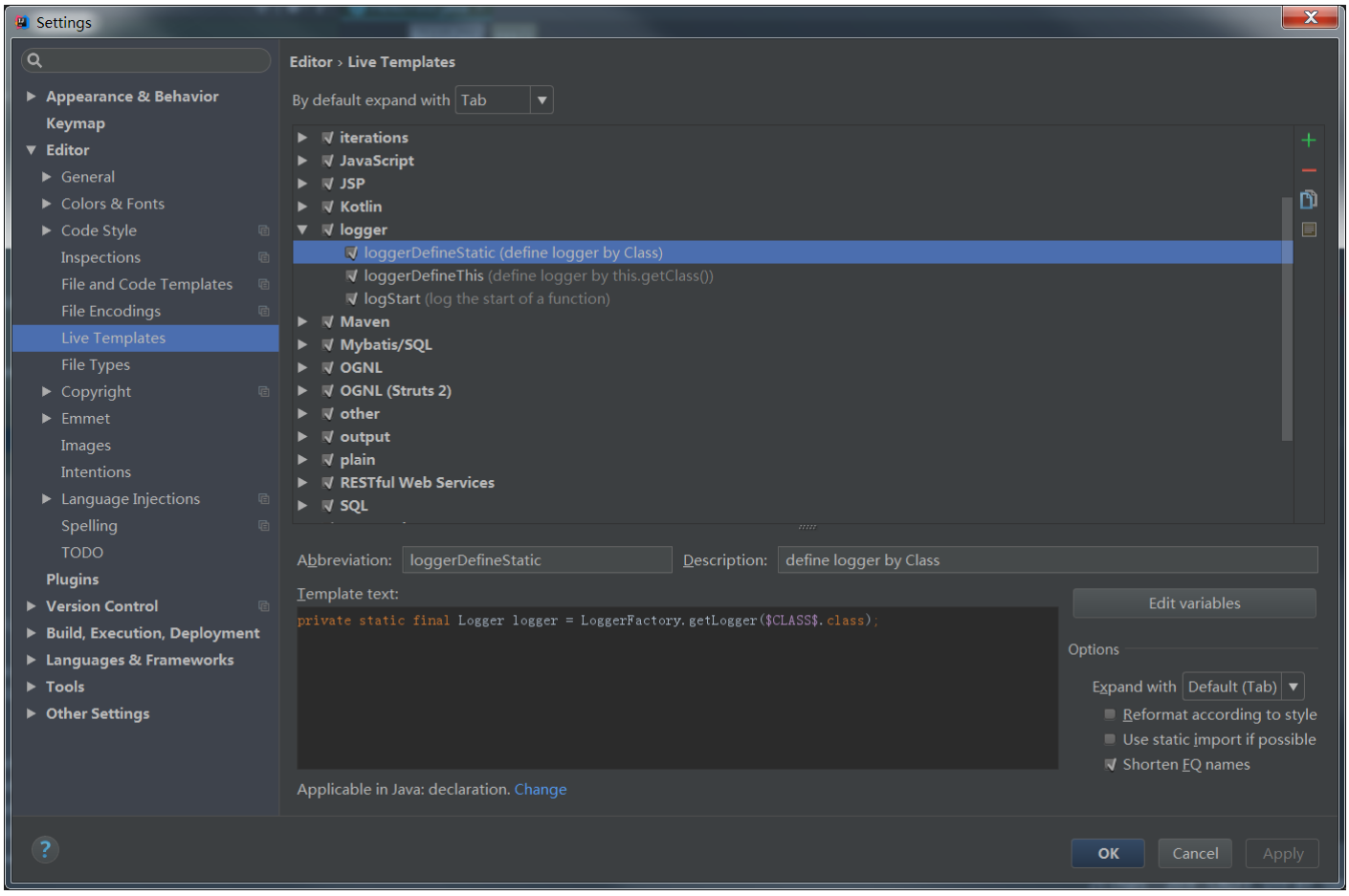
2.如何自定义 Live Template

IntelliJ 提供了很多现成的 template。但你也可以根据自己的需要创建新的 template。

如我们经常要写 logger 的定义：private static final Logger logger = LoggerFactory.getLogger(MyClass.class);，如果每次都手敲或复制是不是有点繁琐，这里就可以借用 sout 这种 Live Templates 实现快速代码编写。用好了这个功能，以后妈妈再也不用担心我们经常加班了。

Settings-->Editor-->Live Templates...

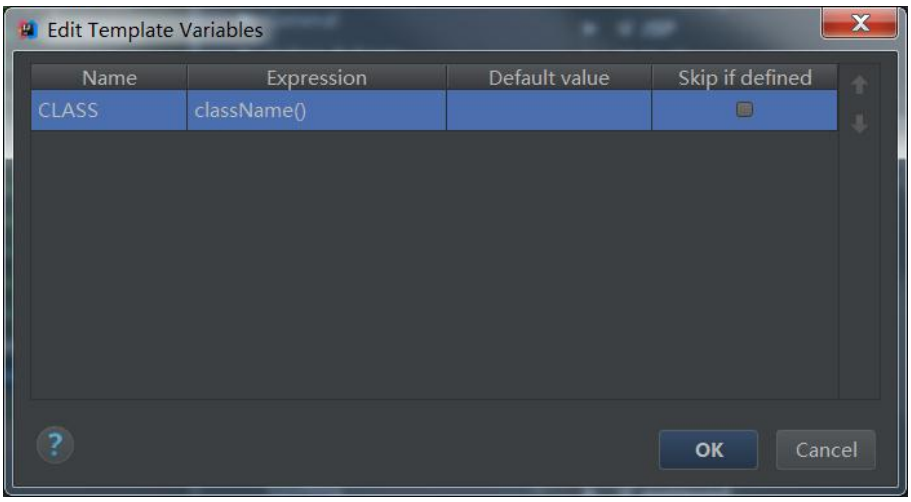
2.1 编写模板



- Abbreviation：模板的缩略名称，如 sout；
- Template text：模板的代码片段，可以用自定义变量。

2.2 自定义变量

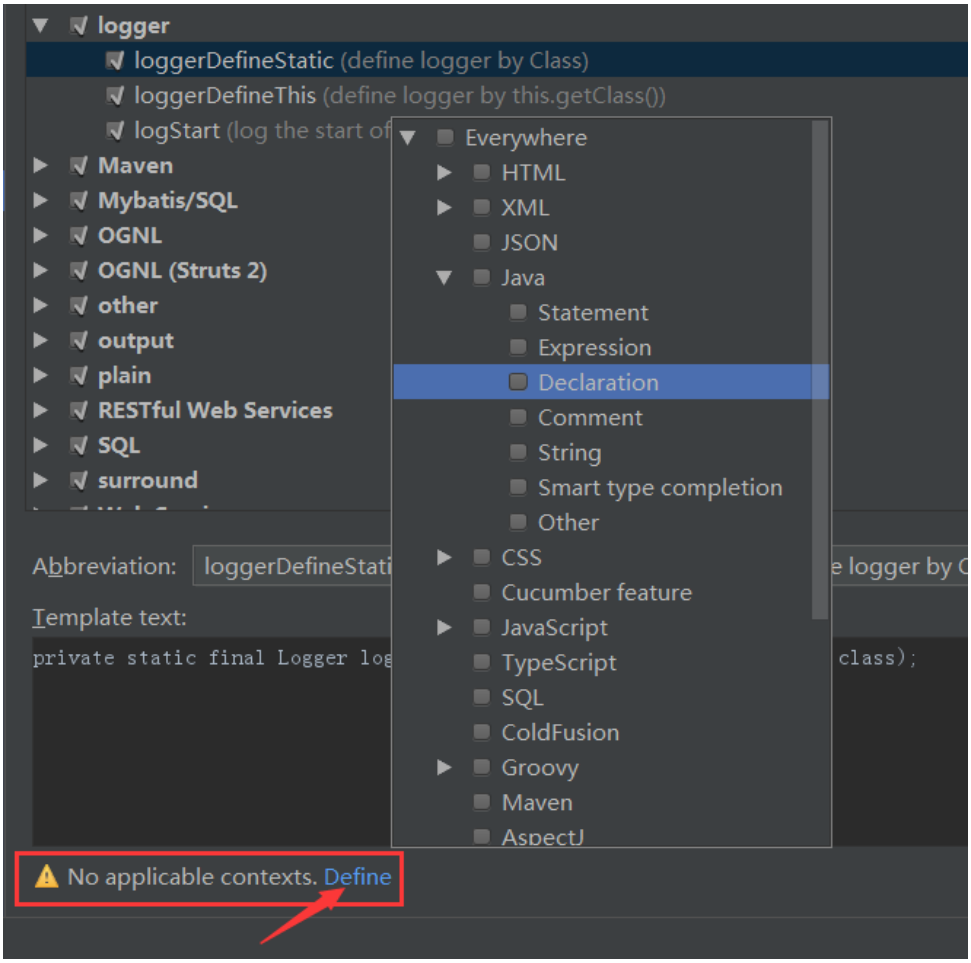
这里我们自定义了 CLASS 变量（使用时需要前后都加上$）,但 Idea 并不识别，这就需要对自定义的变量进行编辑，Edit Variables。



- Name:我们的自定义变量名称；
- Expression:变量替换表达式，这里我们使用了 Idea 模板支持的内置函数 className()表示类名，更多的内置函数参考：Creating_and_Editing_Template_Variables
- Default value：表达式计算失败时的默认值。

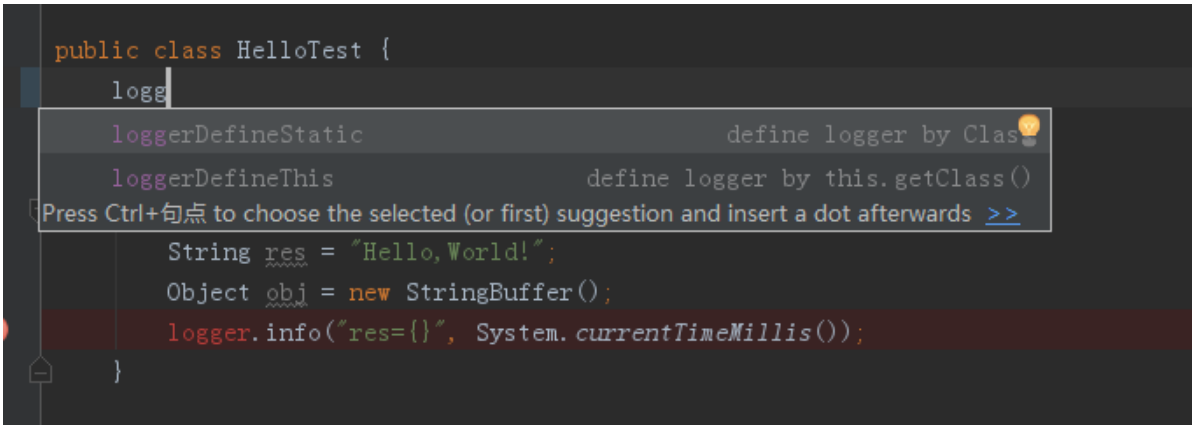
2.3 设置应用范围

模板代码设置完后，我们还要设置其应用范围，即 loggerDefineStatic 在哪里会生效。



这里，我们选择 loggerDefineStatic 在 Java 的声明里生效。

OK，到这里，loggerDefineStatic 的设置就算完成了，接下来就可以像 sout 一样使用了，是不是很帅！



3.常用模板

3.1 loggerDefineStatic

```
1 private static final Logger logger = LoggerFactory.getLogger($CLASS_NAME$.class);
```

1)CLASS_NAME

```
1 className()
```

3.2 loggerDefineThis

```
1 private final Logger logger = LoggerFactory.getLogger(this.getClass());
```

3.3 logStart

```
1 logger.info("op=start_$METHOD_NAME$, $PARAMS_FORMAT$", $PARAMS$);
```

1)METHOD_NAME

```
1 methodName()
```

2)PARAMS_FORMAT

```
1 groovyScript("_1.collect{it+'={}'}.join(', ')", methodParameters())
```

3)PARAMS

```
1 groovyScript("_1.collect{it}.join(', ')", methodParameters())
```