Java 注释规范 (配合 EasyYapi 使用)

Java 注释规范 (配合 EasyYapi 使用)

> 使用范例

类注释

示例

```
/**
 * 分类名称
 * 分类备注/描述
 * @module 归属项目
 * @author Allen
 * @date 2020/6/5 下午2:25
 * @copyright 2020 barm Inc. All rights reserved
 */
@RestController
```

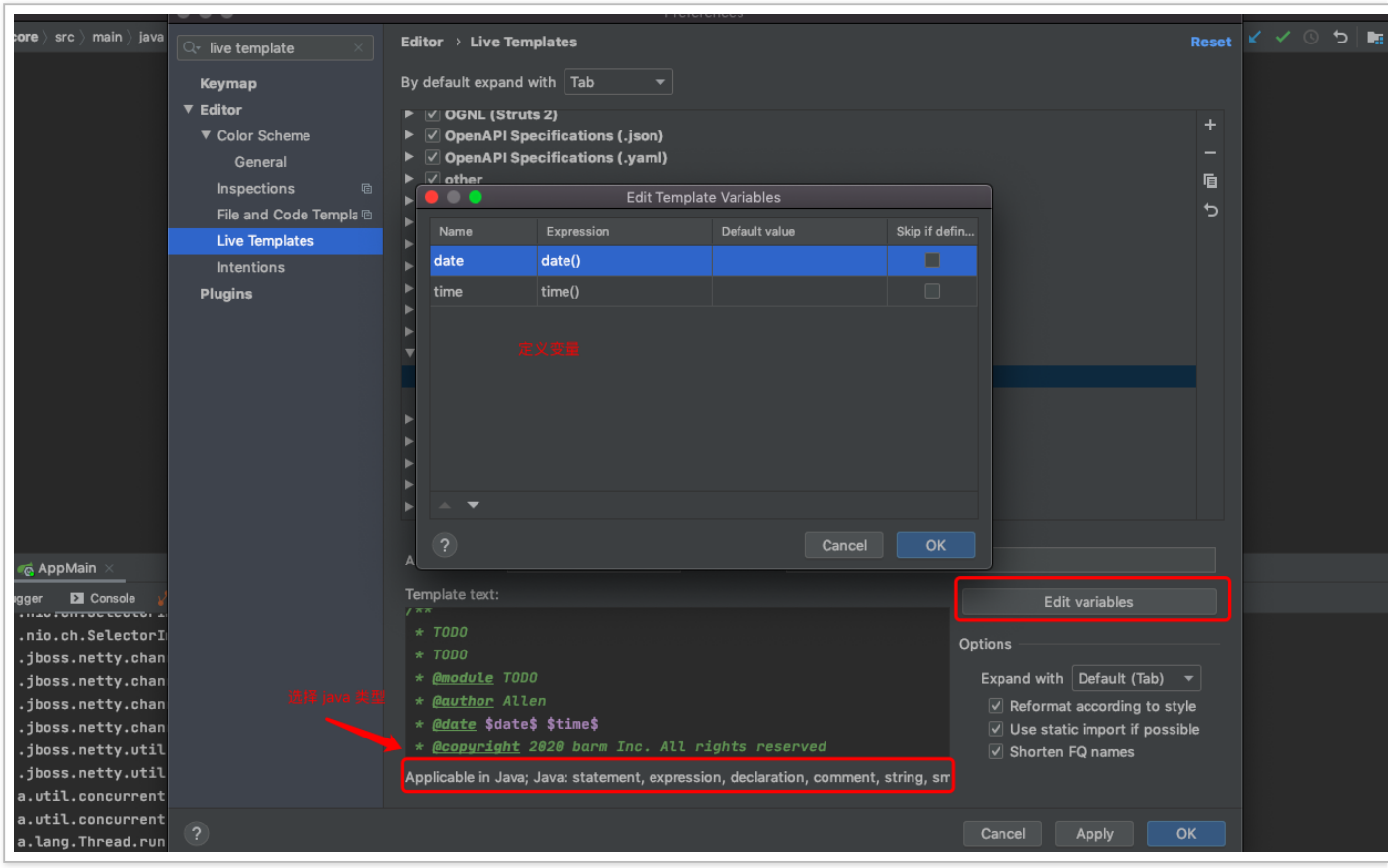
```
@RequestMapping("/barm")
public class DemoController {
```

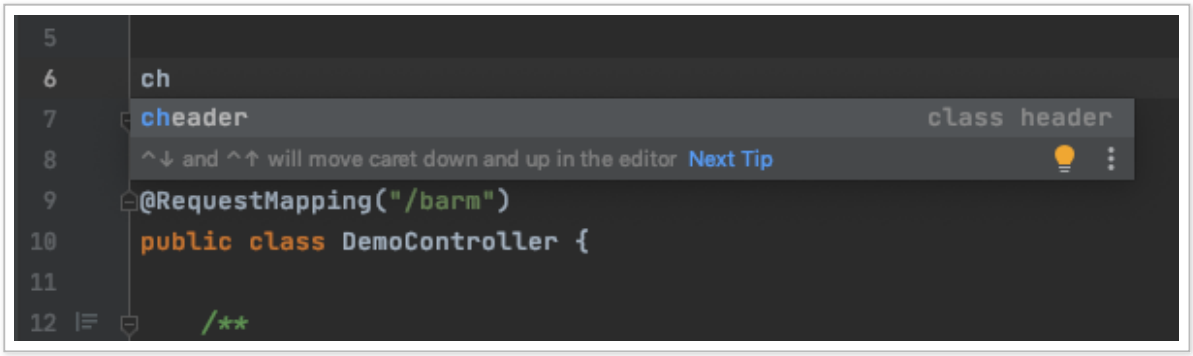
- 第一行默认是 接口 `分类名称`
- 第一行到第一个以 @开头的行之前的为 `接口描述`
- 请注意规范, 类请明确 `@author` , `@date` , `@copyright`
- `@module` 用于分类 api , 详情见下文

通过 live templates 配置类注释模板

```
/**
 * TODO
 * TODO
 * @module TODO
 * @author Allen
 * @date $date$ $time$
 * @copyright 2020 barm Inc. All rights reserved
 */
```

进入 `live templates > 新增group > 添加模板cheader`





IDEA 配置

- file and code template
- live template

方法注释

示例

```
/**
 * api名称
 * api描述
 * @param paramA 参数 A
 * @param paramB 参数 B
 * @return {@link ResultVO}
 */
@PostMapping("/pathOfApi1")
public ResultVO methodName1(String paramA, String paramB){
    log.info("paramA {} , paramB {} ", paramA, paramB);
    ResultVO resultVO = new ResultVO();
    resultVO.setCode(000);
    resultVO.setMessage("success");
    return resultVO;
}
```

- 第一行默认 `api名称`
- 第二行默认 `api描述`
- 入参名后 + 空格 后表示 参数的描述
- @RequestMapping 请明确请求的类型 `GET, POST …`

```
/**
 * yapi接口2
```

```
     * 默认使用`application/x-www-form-urlencoded`,
     * 对于`@RequestBody`将使用`application/json`
     * 用注释`@deprecated`来表示api废弃
     *
     * @deprecated 改用 {@link DemoController#methodName3(MockDtoOrVo)}
     */
    @Deprecated
    @PostMapping(value = "/pathOfApi2")
    public ResultVO methodName2(@RequestBody MockDtoOrVo jsonModel){
        ResultVO resultVO = new ResultVO();
        resultVO.setCode(000);
        resultVO.setMessage("success");
        return resultVO;
    }
```

- 默认使用 `application/x-www-form-urlencoded`
- 对于 `@RequestBody` 将使用 `application/json`
- 注释 `@deprecated` 来表示 api 废弃

不确定的返回类型

```
/**
  * @result {@link Result}
  * @result {@link Result<UserInfo>}
  */
 public Result mockString() {
     ...
 }
```

IDEA 配置

直接在 方法前输入 `/**` 按 `Enter` 键出

参数注释

```
/**
 * 字段注释
 */
private Long field1;
```

注释`@deprecated`来表示 api 废弃

```
/**
 * 用注释`@deprecated`来表示字段被废弃
 * @deprecated It's a secret
 */
private int field5;
```

使用 @NotBlank/@NotNull 表示字段必须

```
/**
 * 如果使用javax.validation的话
 * 可以使用@NotBlank/@NotNull表示字段必须
 */
@NotBlank
@NotNull
private String field6;
```

字段可能有不同类型的值

```
/**
  * @maybe {@link UserInfo}
  * @maybe {@link java.lang.String}
  */
 public Object target;
```

@see 的使用

```
/**
 * 使用@see来说明当前字段的取值是某个Constant
 * @see DemoConstant#desc
 */
private int field3;

/**
 * 当目标枚举字段与当前字段名不一致,额外指定
 * @see DemoEnum#getCode()
 */
private int field4;
```

@link 指定类型关联具体类

- 使用前使用 " " 与其他字句区分开

```
/**
  * @result {@link Result}
  */
 public Result mockString() {
     ...
 }
```

@module API 到 指定项目

> `module` 用于分类 api
>
> - 导出 `postman` 时，每个 `module` 将作为一个单独的文件夹
> - 导出 `yapi` 时，每个 `module` 需要配置相应的 `token`，即对应一个 `yapi` 中的项目
> - 默认情况下取当前模块名 (单模块项目取项目名)

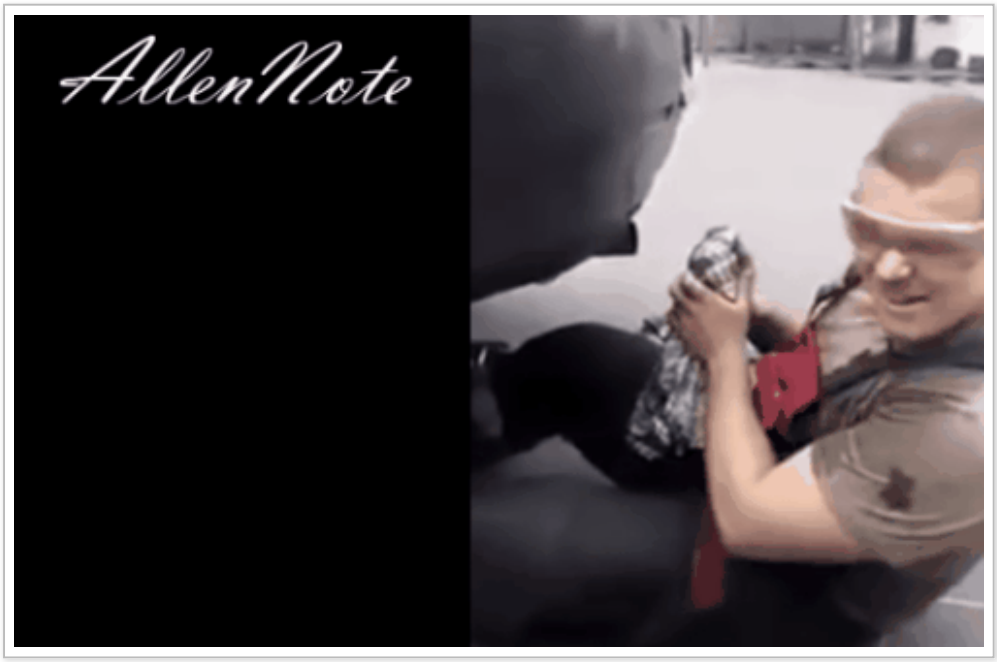
@ignore 忽略 API

`当在接口注释使用 @ignore时候` 导出被忽略

```
/**
     * @ignore
     * @param param
     * @return
     */
    @PostMapping(value = "/pathOfApi4")
    public ResultVO methodName4(@RequestParam String param){
        log.info("param {}", param);
        ResultVO resultVO = new ResultVO();
        resultVO.setCode(000);
        resultVO.setMessage("success");
        return resultVO;
    }
```

写在最后

还是老三样. 欢迎 点赞, 转发, 评论 ~



全文完