后端编码规范

作者：蒋志文

一、 包

1、 包命名全部采用小写，不用下划线区分单词

2、 主包名采用[公司性质].[公司名称].[项目名称]的命名方式

例如：饭否公司的萤火团队做的外卖点餐系统包名 *com.fanfou.food*

3、 通用功能子包名采用[主包名].[通用名称]的命名方式

常见通用功能如下表：

包名	含义
[主包名].controller	Web 控制器所在包，负责处理用户请求
[主包名].service	系统提供的服务类所在的包
[主包名].dao	数据库操作相关所在的包
[主包名].domain	数据库操作相关所在的包
[主包名].annotation	项目注解所在的包
[主包名].exceptions	项目自定义的异常类所在的包
[主包名].fillters	Web 过滤器所在的包
[主包名].global	全局处理操作所在的包
[主包名].interceptor	Spring 拦截器所在的包
[主包名].utils	通用工具类所在的包

一般功能子包名采用[主包名].[模块名称].[子模块名称]的命名方式

例如：订单控制器相关模块包名 *edu.fanfou.controller.order*

4、 只需导入用到的类，不得用*导入包下所有类

5、 导入类时，系统类在上方，自定义类在下方

二、 代码

1、 代码主要采用大/小驼峰命名法，即除首字母外，每个单词首字母大写，整体首字母大小根据其它规范决定

2、 类名、接口名、枚举名等首字母大写，若由多个单词组成，则其后每个单词首字母大写

例如：

```
class AuthController{}
```

3、 接口一般使用 able、ible、er 等作为后缀

例如：

```
interface Observable{}
```

4、 继承自安卓组件的类，采用父类名作为后缀

例如：

```
class AuthenticationInterceptor extends Interceptor{}
```

5、 自定义异常必须以 Exception 结尾

6、 除 for 循环变量外，一律不得使用 i、j、k 等单字符作为变量名

7、 定义数组时方括号紧随在原始类型之后，数组名称一般使用复数形式

例如：

```
int[] arrays;
```

8、 常量、枚举等均采用大写形式，用下划线区分各单词

例如：

```
final static int DIALOG_ID_ALARM = 1;
enum Season{SPRING, SUMMER, AUTUMN, WINTER};
```

9、 全局变量添加所有者前缀：实例成员变量前缀 m（表示 member），类静态变量前缀 s（表示 static）

例如：

实例变量 `mRun`

类静态变量 `sInstance`

10、 除单例模式外一般不得使用静态变量

11、 常量一般使用 final static 修饰，根据需要使用可见性修饰符

例如：

```
public static final int VISIBLE = 0x00000000;
```

12、 一般方法名首字母小写，若由多个单词组成，则其后每个单词首字母大写

13、 仅在项目内使用的实体类不使用 JavaBean 进行封装，直接将成员变量访问修饰符修改为非 private

例如：

```
class User{public String name,pwd;}
```

14、 实体类中固定值的成员变量可设置成 final，并通过构造函数初始化

15、 实体类中不得随意修改的成员变量可添加下划线前缀以作区别

例如：

```
class User{public int _id;}
```

16、 一般不使用 System.out 输出，而是使用 Log 中的方法，并对 Log 进行封装，只在调试时输出重要信息，正式版不输出

17、 一般 try……catch 只捕获需要的异常

18、 catch 块不得为空，至少应当将异常信息输出

三、 注释

1、 开源项目必须添加文件注释，非开源项目建议添加

例如：

```
/*
 * @(#): Document.java
 * @project：IndentObjectNatation
 * @version: v1.1
 * @copyright: Copyright (C) 2013-2014 The Emerald Education
 * @description:
 *      This file is a part of Indent Object Notation project.
 *
 * @modify:
 * ---- No.1 Modified By Mr. Tang At 2014-05-06 11:32 Based On 1.0 ----
 *      Create this file.
 * ---- No.2 Modified By Mr. Zhang At 2014-05-06 11:32 Based On 1.0 ----
 *      Make the class Document extend from the class Node.
 */
```

2、 类定义一般需要写类注释，接口一般需要写接口注释，如果没有文件注释，则需要在类注释和接口注释中标出作者

例如：

```
/**
 * Root of the ION tree, provides the access to the document's data.
 * <p>
 * Subclass of {@link Node}, Since elements, comments, etc. cannot exist
 * outside the context of a Document, the Document also contains the
 * factory methods needed to create these objects.
 * </p>
 *
 * @author Mr. Jiang
 */
class Document {}
```

3、 成员变量、静态变量、常量等添加属性注释

例如：

```
/** This view is invisible. */
```

```
public static final int INVISIBLE = 0x00000004;
```

4、 关联性较大的多个成员变量等可以共用同一条注释

例如：

```
/** The width and height of View. */
private String username, password;
```

5、 public 和 protected 方法必须添加方法注释，default 和 private 方法建议添加方法注释

例如：

```
/**
 * Writes {@code count} characters starting at {@code offset} in {@code
buf}
 * to the target.
 *
 * @param buf
 *            the non-null character array to write.
 * @param offset
 *            the index of the first character in {@code buf} to write.
 * @param count
 *            the maximum number of characters to write.
 * @return {@code true} if success, otherwise {@code false}
 * @throws IndexOutOfBoundsException
 *             if {@code offset < 0} or {@code count < 0}, or if {@code
 *             offset + count} is greater than the size of {@code buf}.
 * @throws IOException
 *             if this writer is closed or another I/O error occurs.
 */
public abstract boolean write(char[] buf, int offset, int count) throws
IndexOutOfBoundsException, IOException;
```

6、 若覆盖基类的方法，则可以不写方法注释，但必须用@Override 标出

例如：

```
@Override
protected void onCreate(Bundle savedInstanceState) {}
```

7、 不建议继续使用的方法用@Deprecated 标出

8、 switch……case 的每个条件一般添加简短说明

例如：

```
switch (type) {
case 1:// Android apps
    break;
case 2:// Android games
    break;
case 3:// iOS apps
    break;
default:// Not a valid package
    break;
}
```

9、 如果 if 的条件大于 2 个，则必须写注释

例如：

```
if (exp1 // If the value of exp1 is true
        || exp2 // If the value of exp2 is true
        || exp3 // if the value of exp3 is true
) {}
```

10、 对于未完成的方法，使用 TODO 加以标记

例如：

```
void write(byte[] buf, File file) {
    // TODO: Write buf to file
}
```

11、 若功能已完成，但存在效率等潜在问题时，使用 XXX 加以标记

例如：

```
void parseXML(File file) {
    // XXX: Maybe SAX is better
    DocumentBuilder builder = DocumentBuilderFactory.newInstance()
            .newDocumentBuilder();
    Document doc = builder.parse(file);
}
```

12、 若代码存在严重问题或仅用于调试，使用 FIXME 加以标记（注：存在 FIXME 标记的代码不能作为正式版发布）

```
boolean login(String name, String pwd) {
    //FIXME: Remove this line before publishing
    System.out.println("name=" + name + ", password=" + pwd);
```

```
    if (users.containsKey(name) && users.get(name).equals(pwd))
        return true;
    return false;
}
```

13、 如果 for、while 等代码块过长，可以在结尾处标记循环变量

例如：

```
for (int position = 0; position < 10; position++) {
    ……
}// end for: position
while(mRun){
    ……
}// end while: mRun
```