Java代码规范

主讲人：鄂乾宇

2017年07月

导读

- 越到后面越重要，一定要有耐心

命名规则

- 严禁汉语拼音
- **Package**名必须全部小写
- **包、类、变量**命名不要加下划线
- **常量**命名要大写,单词之间要用下划线分开
- **方法名**第一个字母小写，两个单词以上的变量名要驼峰命名
- 每个变量要用别人看得懂的词语来描述，尽量不要省略

注释相关

- 提交代码前如果有TODO、FIXME等标签,要删除掉,如果要标记没有做完的任务或者以后要改进的任务,用LXTODO
- 注释掉的代码提交前要删除,一般删除自己的注释的代码,但不要删除别人注释的代码.

变量相关

- 当一个变量在每个方法里面使用,并且每个方法里面包含的内容不一样,那么这个变量不要用字段变量,要用局部变量。
- 尽量不要在构造方法里初始化代码.
- 内部类的变量不要设为全局变量(public 字段变量)，尽量少用全局变量,多用局部变量
- 程序代码中不要出现魔法数字（user1/user2/user3）
- 同一个字符串出现两次要抽取到常量类

对象相关

- 在类中，重复了两次以上且公用的对象一般都要抽出成为类字段。

方法相关

- 如果性能要求比较高,就要用StringBuffer,用append进行字符串叠加
- 每个模块都会用到的代码要放到公共包
- 不要在系统里面写System.out.println，效率低，且不易于查log

Try catch

- 在catch(打印异常)里打印的log4j要用error方法
- 不要在控制台直接打出异常

- 错误代码：
- } catch (SQLException e) {
  // EEE 异常不能直接打印,要捕获
  e.printStackTrace();
  }
- 正确代码
- } catch (SQLException e) {
  logger.error(e);
  }

常量类、枚举类

- 常量类要用接口定义（interface），而且默认的修饰不用写
- 多个类中具有相同逻辑意义的相同的字符串或数字，应该放到常量类中。
- 需要用到状态编码的常量，应该统一放到枚举类中。如success/failed。

不推荐：

```
public interface Constant(){
    private static final String URL="/image/";
 }
```

推荐：

```
public interface Constant(){
    String URL="/image/";
 }
```

配置文件

- 跨系统的公共变量（如jenkins），建议不要定义在常量类中，统一从配置文件加载。
- 涉及到数据库的跨系统公共变量，变量统一从数据库加载。
- RPC框架下，需要序列化的对象所用到的公共变量，需要放在contract工程中的常量类或枚举类中。

数据库相关

- 数据库连接对象的生成不能跟Model层混搅，必须将其放在一个单独的类里，用**单例模式**来实现数据库的连接
- id一般不用int类型，用Long类型（不是long）
- 做主键的列没有任何业务逻辑，没有任何实际意义,而且任何时候都不能修改。
- sql语句一定要用占位符，不能用拼写组成的语句（否则会引起Sql注入安全问题）

```
public class UserDAOImpl(){
    private static Connection connection;
    public static synchronized Connection getConnection() {
        if (connection == null) {
            try{}catch (SQLException e){}
            catch (ClassNotFoundException e)
            { e.printStackTrace() }
        }
        return connection;
     }
}
```

正确

```
public class DBConnectionTools {
    private static Connection connection;
    private DBConnectionTools(){}
    public static synchronized Connection getConnection() {
        if (connection == null) {
            try {
                Class.forName(Constants.MYSQL_DRIVER);
                connection = DriverManager.getConnection(Constants.MYSQL_URL,
        Constants.MYSQL_USER_NAME,
                Constants.MYSQL_PASSWORD);
            }catch (SQLException e) {
                  e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
        return connection;
    }
}
```

错误

```
public void addUser()
{
    sql="insert into student(ID,name,sex,address)
  values(?,?,?,?)" //？则代表占位符
    //可用预处理来发送sql语句
}
```

Q&A

谢谢 THANKS