日志规范

日志框架

本项目采用 Logback + Slf4j + lomback插件实现简单的日志调用。

```
@Slf4j
public class DemoServiceController implements DemoApi {

    public DemoBean get(@RequestParam(required = true, value = "id")
Integer id) {
        DemoBean demoBean = new DemoBean();
        demoBean.setId(200);
        demoBean.setMsg("成功");
        log.info("{}", "info");
        log.warn("{}", "warn");
        log.debug("{}", "debug");
        return demoBean;
    }
}
```

日志分级

```
fatal - 严重的，造成服务中断的错误；
error - 其他错误运行期错误；
warn - 警告信息，如程序调用了一个即将作废的接口，接口的不当使用，运行状态不是期望的但仍可继续处理等；
info - 有意义的事件信息，如程序启动，关闭事件，收到请求事件等；
debug - 调试信息，可记录详细的业务处理到哪一步了，以及当前的变量状态；
trace - 更详细的跟踪信息；
```

1.debug，trace只能在开发和测试环境下输出，不能在生产环境下输出。

输出规范

> 1.严格按照SLF4J输出格式,使用占位符

```
void foo() {
    log.info("{},{},{}",1,2,3);      // 正确
```

```
    log.info(“123”)                 // 错误
}
```

2.不允许出现System print(包括System.out.println和System.error.println)语句。

3.不允许出现printStackTrace。

```
void foo() {
     try {
             // do something…
     }catch ( Exception e ) {
            e.printStackTrace(); // 错误
            _LOG.error("Bad things : ", e ); //正确
         }
}
```

4.输出Exceptions的全部Throwable信息，因为logger.error(msg)和logger.error(msg,e.getMessage())这样的日志输出方法会丢失掉最重要的StackTrace信息。

```
void foo(){
     try {
            //  do something…
      }catch ( Exception e ){
          log.error(e.getMessage()); // 错误
          log.error("Bad things : ",e.getMessage()); // 错误
          log.error("Bad things : ",e); // 正确
      }
}
```

5.不允许记录日志后又抛出异常，因为这样会多次记录日志，只允许记录一次日志。

```
void foo() throws LogException{
     try{
         // do something…
     }catch ( Exception e ){
          log.error("Bad things : ", e);
          throw new LogException("Bad things : ",e);
       }
```

```
8 }
```