日志规范

1 日志工具

1.1 日志工具的选择

项目中仅使用抽象日志框架来处理日志，不得直接使用Log4J或LogBack，建议使用Slf4J。具体的日志框架实现，建议使用LogBack。

在Maven项目的依赖中，如果间接依赖了不同的日志框架，需要exclude掉，或者它们仅能出现在test的scope中。

以使用Slf4j和LogBack为例，如果有第三方库依赖了Apache Commons Logging或者Log4J框架，则可进行如下配置：

1、添加Slf4J对其他框架的支持

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.5</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>log4j-over-slf4j</artifactId>
    <version>1.7.5</version>
</dependency>
```

2、排除掉第三方库中的依赖，以Apache Commons Logging为例，在`<dependency>`中添加如下内容

```
<exclusions>
    <exclusion>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
    </exclusion>
</exclusions>
```

（后续的文档中，默认使用*Slf4J*进行说明。）

1.2 Logger的定义

在使用`Logger`时，需要用如下方式在类中定义Logger对象：

```
private static final Logger logger = LoggerFactory.getLogger(xxx.class);
```

如果有确定的日志名，则可以直接指定日志名：

```
private static final Logger logger = LoggerFactory.getLogger("日志名");
```

1.3 Logger的使用

输出日志时，不得使用字符串拼接的方式，即：

```
logger.info("foo" + bar);
```

需要使用以下方式，避免不必要的`toString`和字符串拼接操作：

```
logger.info("foo{}", bar);
```

Slf4J的方法支持可变参数，可以添加任意数量的{}，且无需在输出日志前进行isXXXEnabled判断。

如果使用Apache Commons Logging或者Log4J，需要在输出之前进行判断，例如：

```
if (logger.isInfoEnabled()) {
    logger.info("foo");
}
```

建议对`ERROR`和`WARN`级别以外的日志都增加此判断。

为了保证异常信息能够输出完整的异常堆栈，必须使用支持Throwable参数的方法输出异常，同时选择`ERROR`级别输出：

```
logger.error("处理XXX业务时发生异常", e);
```

使用`+`拼接描述和异常对象，这样会丢失异常堆栈，因此严禁使用以下方法：

```
logger.error("处理XXX业务时发生异常" + e);
```

2 日志文件命名

日志分为以下几类：

1. 业务日志
2. 错误日志，统一将`ERROR`级别的日志汇总到一个日志中
3. 摘要日志，以`-digest`作为主文件名后缀
4. 统计日志，以`-stat`作为主文件名后缀

日志文件可以有两种扩展名：

1. xxx.log
2. xxx.Nd.log，`N`表示天数，用来标识日志的保存天数

未标识保存天数的日志文件，运维可以按需要任意清除。

日志必须按照一定规则进行滚动，以避免单个日志文件过大，影响性能。

日志可以按天或者按小时滚动，滚动后的日志文件后缀名结尾方式如下：

1. 按天滚动，xxx.log.20131213
2. 按小时滚动，xxx.log.20131213_12

综上，一个按天滚动、需要保存7天的摘要日志完整文件名是这样的：

```
xxx-digest.7d.log.20131213
```

3 日志格式

日志需要遵循一定的格式，方便人与机器阅读和分析。日志一般由以下几部分组成：

1. 时间，一般精确到毫秒
2. 日志级别，即`ERROR`、`WARN`、`INFO`、`DEBUG`、`TRACE`
3. 唯一ID，一般是贯穿所有上下游系统的唯一ID，如果没有的话也可以是本系统内部的标识
4. 线程，输出日志的线程名称
5. 日志名称，一般是类名（可以做些限制，避免过长），也可以是特定的名称
6. 具体内容

3.1 业务日志

业务日志的格式建议如下：

```
时间 级别 [唯一ID][线程] 日志名 - 内容
```

例如：

```
2013-12-09 18:17:30,110 WARN [1049754451][http-nio-8081-exec-6] c.b.c.b.s.i.XXXServiceImpl - 请求XXX业务处理，传入的参数为空。
```

3.2 错误日志

格式同3.1。

3.3 摘要日志

大多数摘要日志都是按类型输出到不同文件中的，因此文件就代表了含义，不需要日志名。格式建议如下：

```
时间 级别 [唯一ID][线程] - 内容
```

不同的类型，内容有所不同，一般摘要可以有以下几类：

1. 数据层访问摘要，文件名一般为`xxx-dal-digest.log`
2. 调用下游服务摘要，文件名一般为`xxx-sal-digest.log`
3. 对上游提供服务摘要，文件名一般为`xxx-service-digest.log`
4. Web页面访问摘要，文件名一般为`xxx-page-digest.log`

一般的摘要日志内容中要包含以下内容：

1. 记录的对象
2. 结果，成功或失败
3. 耗时
4. 参数，用`,`分隔，可选
5. 返回值，用`,`分隔，可选

因此，完整的摘要日志格式会是这样的：

```
时间 级别 [唯一ID][线程] - [(对象,结果,耗时)(参数)(返回值)]
```

忽略的内容用`-`填充。

如果摘要是由系统框架级统一输出的，选择同样的级别输出，则可以忽略“级别”一项。

3.3.1 数据层访问摘要

每次数据库操作建议都记录摘要，以便后续分析系统处理请求过程中的瓶颈，借业务判断数据库状态等等。

```
2013-12-09 15:40:55,092 INFO  [1049754451][http-nio-8081-exec-6] - [(UserDao.findByNameAndType, Y, 10ms)(foo,bar)(-)]
```

3.3.2 调用下游服务摘要

格式同3.3.1。

3.3.3 对上游提供服务摘要

对上游提供服务的摘要中，最好能够记录下是哪个上游系统发起的请求，因此需要增加一项。

以REST风格的服务为例，其实质相当于处理了一次HTTP请求，可以采用如下格式：

```
时间 级别 [唯一ID][线程][上游系统] - [(HTTP方法,请求URL,对应处理方法,结果,总耗时,业务处理耗时,结果渲染耗时)(参数)(HTTP结果响应码,返回值)]
```

例如：

```
2013-12-09 15:40:55,578 INFO  [1049754451][http-nio-8081-exec-6][systemA] - [(POST,/user,UserController.find,Y,3ms,2ms,1ms)(foo,b
```

RPC风格的服务，则可以使用如下格式：

```
时间 级别 [唯一ID][线程][上游系统] - [(处理方法,结果,耗时)(参数)(返回值)]
```

3.3.4 Web页面访问摘要

格式同3.3.3的REST服务摘要日志。

3.4 统计日志

统计日志时对某些资源一段时间内使用情况的汇总，因此与具体的请求或线程无关，格式较为简单：

```
时间 - [(统计项1)(统计项2)]
```

例如对系统LOAD和内存的统计：

```
2013-12-13 16:08:05,532 - [(2.15)(367,64400,0,0,2486)]
```

4 日志内容

本节主要描述业务日志的内容，一般根据业务内容进行定义，但需要遵循以下要求：

1. 日志内容要完整，至少要能通过日志了解业务的情况。
2. 日志文件需要合理的拆分，不能全部放置于一个文件中。
3. 日志文件内容尽量不要重复输出，避免浪费存储空间。
4. 日志要有合理的日志级别，生产环境一般只打开`INFO`及以上的日志级别。
    1. 正常的业务日志记录，使用`INFO`级别。
    2. 可接受的或者可预见的业务错误，使用`WARN`级别。
    3. 系统异常或者严重的业务错误，使用`ERROR`级别。
    4. 仅供调试使用的日志，使用`DEBUG`级别。

出于安全角度的考虑，以下内容严禁完整出现在日志中，可以选择截取部分显示：

1. 身份证号码
2. 手机号码
3. 银行卡号码

关于信用卡，以下内容严禁出现在日志中：

1. 信用卡有效期
2. 信用卡CVV

5. 修订记录

2013-12-16: Version 0.2

> Logger使用中增加Exception相关内容，日志内容中增加安全内容

2013-12-13: Version 0.1

> 撰写初稿