**CIF系统概要设计**

**文档修改历史**

| 版本号 | 修改日期 | 编写 | 评审 | 批准 | 修改内容 |  |
| --- | --- | --- | --- | --- | --- | --- |
| V0.1 | 2016-10-24 | 李太平 |  |  | 新建 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**资邦金服网络科技集团有限公司**

**2016年10月**

目录

[1 引言 4](#_Toc466015141)

[1.1 文档编制目的 4](#_Toc466015142)

[1.2 系统设计的目标 4](#_Toc466015143)

[1.2.1 长远规划 4](#_Toc466015144)

[1.2.2 安全性高、统一认证 4](#_Toc466015145)

[1.2.3 流程简洁、高效 5](#_Toc466015146)

[1.2.4 易扩展、易维护 5](#_Toc466015147)

[1.3 阅读对象 5](#_Toc466015148)

[1.4 术语 6](#_Toc466015149)

[1.5 参考文档 6](#_Toc466015150)

[2 CIF优化总体设计 6](#_Toc466015151)

[2.1 系统物理架构 6](#_Toc466015152)

[2.2 系统逻辑架构 7](#_Toc466015153)

[2.3 软硬件配置方案 8](#_Toc466015154)

[2.4 总体流程 11](#_Toc466015155)

[2.4.1 注册服务 12](#_Toc466015156)

[2.4.2 登录服务 13](#_Toc466015157)

[2.4.3 查询服务 19](#_Toc466015158)

[2.4.4 会员等级服务 20](#_Toc466015159)

[2.4.5 认证服务 20](#_Toc466015160)

[2.4.6 银行卡服务 20](#_Toc466015161)

[2.4.7 手机服务 20](#_Toc466015162)

[2.5 运行环境 20](#_Toc466015163)

[2.5.1 网络拓扑结构 20](#_Toc466015164)

[2.5.2 部署方案详细说明 20](#_Toc466015165)

[3 CIF系统模块设计 22](#_Toc466015166)

[3.1 系统模块结构 22](#_Toc466015167)

[3.2 模块设计 23](#_Toc466015168)

[3.2.1 Web层 23](#_Toc466015169)

[3.2.2 系统调度层 23](#_Toc466015170)

[3.2.3 业务层 24](#_Toc466015171)

[3.2.4 对接服务接口层 24](#_Toc466015172)

[3.2.5 服务层 24](#_Toc466015173)

[3.2.6 系统支持层 24](#_Toc466015174)

[3.2.7 系统集成层 25](#_Toc466015175)

[3.2.8 数据访问层 25](#_Toc466015176)

[3.2.9 通用层 25](#_Toc466015177)

[4 CIF系统技术架构设计 25](#_Toc466015178)

[4.1 系统的应用架构 25](#_Toc466015179)

[4.1.1 应用架构说明 25](#_Toc466015180)

[4.2 数据架构 27](#_Toc466015181)

[4.2.1 表结构 27](#_Toc466015182)

[4.3 代码结构 30](#_Toc466015183)

[4.3.1 代码结构 30](#_Toc466015184)

[5 异常处理设计 31](#_Toc466015185)

[5.1 异常处理 31](#_Toc466015186)

[5.2 异常监控 31](#_Toc466015187)

[6 系统安全设计 31](#_Toc466015188)

[6.1 常见的攻击手段 31](#_Toc466015189)

[6.1.1 XSS 31](#_Toc466015190)

[6.1.2 CSRF 32](#_Toc466015191)

[6.1.3 SQL注入攻击 34](#_Toc466015192)

# 引言

## 文档编制目的

确定cif系统中所用的技术框架、核心业务主流程、根据需求划分系统模块、根据业务模型设计概论模型，为开放人员做详细设计提供依据。

## 系统设计的目标

### 长远规划

cif系统为业务提供统一的会员相关的服务。cif系统在满足公司运营和维护的需要的同时，还要为将来系统的扩展和升级提供相应的支持，以利于在现有系统提供的功能基础之上，通过配置和持续开发，提升系统的处理性能和业务服务功能。

### 安全性高、统一认证

cif系统作为核心业务系统之一，承担着前端各种会员相关的业务功能，系统需要保证以下几个方面的安全需求：

* 系统需要在处理、显示和验证过程中，保证交易数据的完整性以及订单和交易状态的正确性、一致性、可追溯性。
* 系统之间的数据交换必须满足数据安全性要求。
* 确保每个关键的操作行为都在操作日志中有详细的记录，系统提供了完备的事后回放功能。
* 系统运行情况需要进行系统日志中进行分级记录，日志存放和目录划分合理，系统日志至少应该包括运行日志、警告、错误及调试等相关日志，日志的格式应该规范、统一。

### 流程简洁、高效

CIF系统再接收到大量的请求数据后，要能够在限定时间内处理完成，并将处理后的状态返回到请求方，以保证业务处理的正确性和时效性。此外，系统需要能够支持不同的渠道、不同业务线和不同产品线的开展，此外，CIF系统的建设中，需要对产品、交易、客户和账务等多个方面进行深入分析，定义原有系统和交易的接入方式，为客户提供简洁、高效的服务。

### 易扩展、易维护

CIF系统需要严格遵守应用开发架构规范，在SOA和J2EE技术规范的基础上，融合最新的Browser/Server技术，构建具有可扩展性、可维护的核心平台。

CIF系统应识别出与之交互的其他业务系统，并支持不同的消息协议和传输协议在不同平台、不同系统之间的快速交换和处理。

CIF系统应支持主流的HTTP协议、硬件平台和操作系统。功能实用、无故障率高。从支撑平台配置、软件体系结构等方面，能保证数据分析提供系统在大用户量访问时系统的响应速度不受影响；系统能够通过负载均衡等技术，提升系统的性能。

## 阅读对象

本规范定义了CIF系统优化的设计原则、功能划分、架构设计，其主要用户对象包括但不局限于以下人员。

* 项目管理人员
* 技术管理人员
* 技术开发人员
* 测试人员
* 运维人员

## 术语

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 说明 |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

## 参考文档

|  |  |  |
| --- | --- | --- |
| 文档名称 |  |  |
|  |  |  |
|  |  |  |

# CIF优化总体设计

## 系统物理架构

CIF系统目前采用四台应用集群的方式部署，以后交易量增大，可以向集群中加

新的应用服务器。前端应用通过负载均衡访问CIF系统, CIF系统只允许业务系统通过内网访问。



系统物理结构说明：

1. 应用服务器一共4台（2台前台，2台后台管理）；
2. 数据库服务器一共4台（2主+2备）

3、缓存服务器3台（redis集群）

4、负载均衡服务器2台（1主+1备）

5、zookeeper服务器3台

## 系统逻辑架构

CIF系统整体技术框架采用J2EE（Java 2 platform, Enterprise Edition，J2EE，Java企业应用的开发平台），在Spring应用开发框架的基础上构建企业级应用系统。根据J2EE企业应用开发的一般方法，结合系统自身的特殊需求， CIF系统采用基于B/S结构的三层架构设计，依托spring作为J2EE的实现容器。

应用层：采用restfull风格的web服务模式。

业务逻辑层：采用基础业务逻辑+特定业务逻辑的实现方式，实现了基础业务逻辑的重用和特定业务逻辑的独立化。

数据访问层：采用mybatis作为其数据访问框架，有效的实现的sql的配置化以及数据操作的简易性。

## 软硬件配置方案

#### 硬件配置

CIF系统是采用J2EE规范和Java语言开发的企业级Web应用，考虑到性能、扩展以及业务连续性方面的问题，CIF系统采用集群方式部署，在生产环境中，四台PC服务器组成小集群，向前端各种业务系统提供高效、可靠的CIF服务。

* 生产设备

表 1 生产设备需求表

|  |  |  |
| --- | --- | --- |
| 设备名称 | 类型 | 配置需求 |
| 数据库服务器 | PC/Server | 4台 |
| 应用服务器 | PC/Server | 4台 |
| 缓存服务器 | PC/Server | 2台 |
| 存储 | 存储设备 | 8 \* 146G |

* 开发测试设备

为了能够让测试时的数据具有代表意义，建议测试时构建集群环境以真实反映系统上线后的运行状况，并为以后的性能提升提供可依据的数据参照。

表 2 开发测试设备需求表

|  |  |  |
| --- | --- | --- |
| **设备名称** | **类型** | **配置需求** |
| 数据库服务器 | PC/Server | 现有测试环境中的数据库服务器 |
| 应用服务器 | PC/Server | 2~3台PC机构建集群环境 |
| 缓存服务器 | PC/Server | 2台 |
| 存储其它 | 存储设备 |  |

#### 软件配置

CIF系统的开发环境可以在Windows操作平台的基础上进行，其生产运行环境建议RedHat Enterprise Server6.1及其以上版本。

表 3 软件需求表

|  |  |
| --- | --- |
| **操作系统** | **版本** |
| LINUX | RedHat Enterprise Server6.1 |

* 数据库

表 4 数据库选型

|  |  |
| --- | --- |
| **数据库** | **版本** |
| Mysql | MySQL5.5以上版本 |
| Redis | 3.0 |

* 应用服务器

表 5应用服务器的需求

|  |  |
| --- | --- |
| **中间件** | **版本** |
| Tomcat | Tomcat 7.0 |
| Zookeeper | 3.4及以上版本 |

#### 运行时环境

CIF系统构建与J2EE之上，因此，运行时需要Java虚拟机以及应用服务器数据库系统的支持，运行环境定义如下：

表 6 运行时的环境

|  |  |
| --- | --- |
| **名称** | **要求** |
| 操作系统 | Red Hat Enterprise Server6.1以上 |
| 运行时环境 | JDK 1.7 64bit |
| 数据库管理系统 | MySQL5.5或以上版本 |
| 应用服务器 | Tomcat 7.0 |
| 应用开发框架 | Spring 3.2以上版本 |

#### 运维方案

系统的运行维护配备专门的人员进行系统的日常维护工作：负责Web应用服务器、数据库服务器、网络通讯软件的用户与密码管理；负责Web应用服务器、数据库服务器、网络通讯软件的启动与停止；定期对数据库的数据永久保留，每日的数据备份设备循环使用；定期对系统日志的巡视、检查、清理与备份；定期对系统资源的运行状况的监控；对错误或异常进行分析与排除。

* **备份说明**
* 采用当前数据访问，历史数据访问与磁盘备份模式
* 数据库：本地备份 2天；磁带备份循环1个月
* 日志数据：本地保留10天，磁带备份循环1个月
* 业务报文联机保存时间为24月，脱机保存时间10年
* 进行日常数据清理
* 于交易当日21：00～次日5：00于批量数据加工前和结束后进行数据备份。根据以上数据量分析每次数据全备时间大约为20分钟
* 进行数据备份成功检查
* **监控说明**
* 日常运行监控内容：
  + 应用服务器Tomcat的运行状况
  + Redis运行状态
  + 应用系统通讯状态
* 监控异常情况时的特殊处理
* **业务异常**

记录错误步骤及原因。其他普通业务异常以文本的形式记录log中。

* **应用异常**

在报文处理过程中因为通信终端或通信超时，业务人员可以通过报文回执判断该操作是否已经成功执行。对于未成功执行的操作，业务人员可以选择交易重发或人工处理以保证交易的正确性。

## 总体流程



### 注册服务

#### 流程图

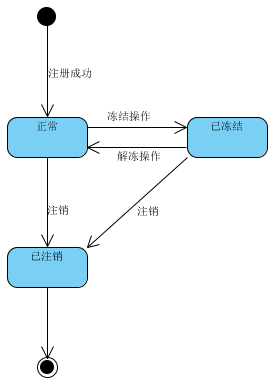




#### 流程说明

1. 业务系统创建个人会员注册请求对象调用cif会员注册接口；
2. 校验请求参数校验，包括非空、参数格式校验；
3. 验证该会员是否存在，若已经存在返回错误信息；
4. 获取分布式锁，防止并发注册；
5. 创建会员模型，初始化相关属性，登录密码MD5加密处理；
6. 落地会员模型，调用用户等级接口落地会员等级信息，释放分布式锁；
7. 创建和返回响应结果。

#### 状态图



### 更新会员信息服务

#### 流程图



#### 流程说明

1. 接收到更新会员信息后先做参数校验；
2. 参数验证成功后再查询该会员信息是否存在，若存在再校验用户的状态，若正常则更新会员信息，返回处理结果。若校验不成功直接返回错误信息。

### 登录服务

#### 登录流程图



#### 登录流程说明

1. 登录服务包括手机号+登录密码、手机号+手势密码、后台直接登录、邮箱+登录密码、用户名+登录密码登录；
2. 接受登录请求后先校验参数；
3. 参数验证成功后先到redis中查询是否存在该会员的登录信息，若存在更新用户的登录信息，返回登录结果给业务系统；
4. Redis中不存在用户登录信息，查询该会员是否存在，若不存在返回会员不存在，若存在判断用户的状态，状态为冻结或者是注销直接返回状态信息，若为正常状态再验证用户的登录密码，登录密码验证失败返回登录密码错误，登录密码验证通过后将用户的登录信息保存到redis中，返回登录结果给业务系统。
5. 若为后台直接登录则只需要验证该会员是否存在，若存在将用户登录信息保存/更新到redis中。

### 登录服务

#### 登出流程图



#### 登出流程说明

接收到业务系统登出请求后现在参数校验，校验通过或从redis中删除该会员的登录信息，若返回删除条数等于0返回该用户已经登出，若返回删除条数为1则返回登出成功。

#### 找回登录密码流程图



#### 找回登录密码流程说明

1. 接收到找回登录密码请求后先验证请求参数，包括非空和登录密码的强度；
2. 请求参数验证后验证该会员是否存在，若不存在返回会员不存在；
3. 会员存在，判断会员的状态，若状态为正常更新用户的登录密码，若为已冻结或者是已注销返回该会员账号已冻结或者是已注销。

#### 修改登录密码流程图



#### 修改登录密码流程说明

1. 接收到修改登录密码请求后先验证请求参数，包括非空和登录密码的强度；

2、请求参数验证后验证该会员是否存在，若不存在返回会员不存在；

1. 会员存在，判断会员的状态，若为已冻结或者是已注销返回该会员账号已冻结或者是已注销；
2. 若会员状态为正常再验证原登录密码，若原密码正确更新登录密码，若不正确返回原密码不正确。

### 查询服务

#### 会员信息查询流程图



#### 会员信息查询流程说明

1. 接收到会员信息查询请求后先验证请求参数，包括非空和登录密码的强度；
2. 请求参数验证后验证该会员是否存在，若不存在返回会员不存在，若存在封装响应对象，返回给业务系统。

### 会员等级更新服务



#### 会员等级更新流程图

1. 接收到会员积分更新请求后先验证请求参数；
2. 根据用户Id查询该用户积分是否存在，若不存在就创建并落地会员积分信息，返回处理结果，若存在就更新会员积分信息。

### 会员等级查询服务



#### 会员等级查询流程图

1. 接收到会员积分更新请求后先验证请求参数；

2、根据用户Id查询该用户积分是否存在，若不存在返回错误信息，若存在返回查询结果。

### 认证服务

### 银行卡服务

### 手机服务

## 运行环境

### 网络拓扑结构

### 部署方案详细说明

为同时满足CIF的高并发与高可靠性，需构建在大量并发用户下需要几乎不存在停机时间的Tomcat集群拓扑。该拓扑还允许未来进行横向扩展，提高性能和可靠性。

数据访问层面，可以根据用户的Id分表，当应用规模需要扩充时，可以按需扩展系统，以保证系统的性能。其主要优点如下：

* 多节点负载均衡;
* 提供高可用：故障容错和无缝切换功能，将硬件和软件错误造成的影响最小化;
* 通过并行执行技术提高事务响应时间（通常用于数据分析系统）;
* 通过横向扩展提高每秒交易数和连接数（通常对于联机事务系统）;
* 节约硬件成本，可以用多个廉价PC服务器代替昂贵的小型机或大型机，同时节约相应维护成本;
* 可扩展性好，可以方便添加删除节点，扩展硬件资源。

服务层面，使用双机热备模式（Keepalived）的Nginx作为软均衡器加之Tomcat，进行集群部署，并可横向扩展应用服务器，以提高吞吐量、并发能力以及减少系统响应时间。服务中间件以其提供的主/从模式进行部署，以避免单点故障。

# CIF系统模块设计

## 系统模块结构



## 模块设计

### Web层

#### 模块描述

Web层主要包括前端页面（jsp、css、js）和后端java代码（Control、VO、辅助类）。

前端页面。

Control代码尽量按照业务来区分，每个Control里面的方法不要过多，每个支付网关异步通知写一个独立的Control。Control里面主要接受前端请求、验证请求参数、调用业务层处理业务逻辑，返回处理结果给前端。

Web层需要注意以下几个问题：

* 需要对用户身份做校验；
* 页面里面不要过多编写js，js可以放在单独的js文件里面；
* 客户端通过js校验后服务端也要做校验；
* 由于客户端可以通过浏览器修改参数的值，关键的业务参数值需要在后端重新取，不能已客户端的数据为准。

### 系统调度层

#### 模块描述

系统调度主要包括数据处理、批量业务处理。

### 业务层

#### 模块描述

业务层主要处理业务逻辑，对外服务接口的实现，包括会员注册、登录、找回/修改登录密码、查询服务等。

### 对接服务接口层

#### 模块描述

接口层是对外提供服务，其他系统通过dubbo 框架rpc调用该接口，包括会员注册、登录、找回/修改登录密码、查询服务等。

### 服务层

#### 模块描述

服务层主要供业务层调用，通过调用数据访问层操作数据库、复杂的数据库操作需要做事务控制、管理事务、分库分表策略等。

### 系统支持层

#### 模块描述

系统支持层主要是提供一下系统功能的扩展，目前添加了数据访问层代码生成的支持，后续会陆续加入，包括分布式锁等。

### 系统集成层

#### 模块描述

系统集成层主要是集成第三方系统的服务接口。

### 数据访问层

#### 功能描述

数据访问层使用mybatis框架封装对数据库的操作，包括entity、dao、sqlmap。

### 通用层

#### 功能描述

通用层主要包括工具类、枚举、常量、上下文对象、异常、日志处理等。

# CIF系统技术架构设计

## 系统的应用架构

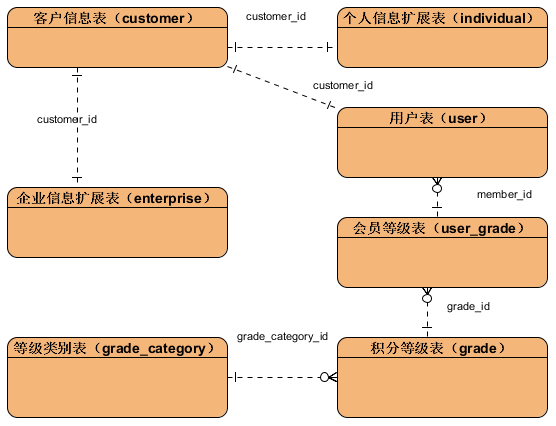
CIF系统在JavaEE规范基础之上，采用Java构建的基于Web的企业级应用系统，其应用架构如下图所示：

### 应用架构说明

* Web层采用SpringMVC框架，jsp作为视图；
* 事务层主要由Spring-tx构成。衔接Web层提供的数据，组织业务逻辑及数据访问接口，由Spring-tx通过AOP来提供事务管理，以确保数据的一致性，推荐使用注解来控制事务，事务的传播特性要根据业务的需要来判断是使用PROPAGATION\_REQUIRES\_NEW还是PROPAGATION\_REQUIRES\_NEW，spring默认是使用PROPAGATION\_REQUIRES；
* 缓存服务器使用redis；
* 分布式锁使用redis或者是zookeeper；
* 报文处理主要由XStream和BouncyCastle构成。XSteam负责对报文的拆分与组装。
* 分布式调用服务使用dubbo框架；
* 日志处理使用LogBack；
* 定时任务主要由Quartz构成。

## 数据架构

### 表结构



#### 客户表（customer）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** |
| 主键 | id | bigint | 20 |
| 客户Id | customer\_id | varchar | 22 |
| 客户号 | customer\_no | varchar | 22 |
| 客户名称 | customer\_name | varchar | 20 |
| 证件类型 | certificate\_type | tinyint | 1 |
| 证件号码 | certificate\_no | varchar | 32 |
| 创建时间 | create\_time | datetime |  |

#### 个人信息扩展表（individual）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** |
| 主键 | ID | bigint | 20 |
| 客户Id | customer\_id | varchar | 22 |
| 年龄 | age | int | 3 |
| 性别 | gender | varchar | 20 |
| 工作 | work | varchar | 32 |
| 工作年限 | work\_years | int | 2 |
| 常用地址 | common\_address | varchar | 256 |
| 婚姻状态 | marriage\_status | tinyint | 1 |
| 年收入 | annual\_salary | decimal | 12 |
| 创建时间 | create\_time | datetime |  |

#### 企业信息扩展表（enterprise）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** |
| 主键 | id | bigint | 20 |
| 客户Id | customer\_id | varchar | 22 |
| 工商注册号 | business\_registration\_no | varchar | 32 |
| 法人姓名 | legal\_person\_name | varchar | 128 |
| 法人证件类型 | legal\_person\_certificate\_type | bigint | 32 |
| 法人证件号码 | legal\_person\_certificate\_no | varchar | 32 |
| 企业名称 | enterprise\_name | varchar | 256 |
| 企业注册地址 | enterprise\_register\_addr | varchar | 256 |
| 创建时间 | create\_time | datetime |  |
| 修改时间 | modify\_time | datetime |  |

#### 用户表（user）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** |
| 主键ID | id | bigint | 20 |
| 用户ID | member\_id | varchar | 22 |
| appId | app\_id | tinyint | 1 |
| 客户ID | customer\_id | varchar | 20 |
| 手机号 | mobile | varchar | 18 |
| 邮箱 | email | varchar | 64 |
| 用户名 | user\_name | varchar | 32 |
| 登录密码 | password | varchar | 32 |
| 手势密码 | gesture\_pwd | varchar | 32 |
| 昵称 | nickname | varchar | 32 |
| 用户状态 | status | tinyint | 1 |
| 头像地址 | head\_img\_url | varchar | 128 |
| 实名时间 | real\_name\_time | datetime |  |
| 注册来源 | register\_source | tinyint | 1 |
| 用户等级(非认证) | grade | varchar |  |
| 记录创建时间 | create\_time | datetime |  |
| 记录最后修改时间 | modify\_time | datetime |  |
|  |  |  |  |

#### 会员等级表（user\_grade）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** | |
| 主键 | id | bigint | 20 | |
| 客户Id | member\_id | varchar | 22 | |
| 等级Id | grade\_id | int | 11 | |
| 等级名称 | grade\_name | varchar | 32 | |
| 创建时间 | create\_time | datetime |  | |
| 修改时间 | modify\_time | datetime |  | |

#### 等级表（grade）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** | |
| 主键 | id | int | 11 | |
| 等级类别 | grade\_category\_id | int | 11 | |
| 等级名称 | grade\_name | varchar | 32 | |
| 创建时间 | create\_time | datetime |  | |
| 修改时间 | modify\_time | datetime |  | |

#### 等级类别表（grade\_category）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **字段** | **字段定义** | |
| **类型** | **长度** | |
| 主键 | id | int | 11 | |
| 等级类别名称 | category\_name | varchar | 32 | |
| 创建时间 | create\_time | datetime |  | |
| 修改时间 | modify\_time | datetime |  | |
|  |  |  |  | |

## 代码结构

### 代码结构

基于Maven的构建与管理规范，主要的代码结构构成如下：

src

-main

–java java源代码文件

-web web模块

-daemon补单模块

-biz业务模块包

-service事务管理

-integration系统集成模块

-façade对外接口层

-dal数据访问

-common通用层

–resources 资源库，会自动复制到classes目录里

–webapp web应用的目录。WEB-INF、css、js等

-test

–java 单元测试java源代码文件

–resources 测试需要用的资源库

-site Site文档站点

target 编译/打包目标目录

# 异常处理设计

## 异常处理

* 基于错误码的异常记录
* 按照异常的严重程度划分不同的等级
* 将异常记录输出到统一的异常文件

## 异常监控

* 监控异常文件，异常文件中有新增的异常时通知相关运维人员。

# 系统安全设计

## 常见的攻击手段

### XSS

XSS攻击的全称是跨站脚本攻击(Cross Site Scripting)，为不跟层叠样式表(Cascading Style Sheets,CSS)的缩写混淆，故将跨站脚本攻击缩写为XSS,是WEB应用程序中最常见到的攻击手段之一。跨站脚本攻击指的是攻击者在网页中嵌入恶意脚本程序，当用户打开该网页时，脚本程序便开始在客户端的浏览器上执行，以盗取客户端cookie、盗取用户名密码、下载执行病毒木马程序甚至是获取客户端admin权限等等。

#### 防御

需要对用户输入的数据进行HTML转义处理，将其中的“尖括号”、“单引号”、“引号” 之类的特殊字符进行转义编码。

使用开源的开发框架jstl标签进行HTML转义，将变量输出。

### CSRF

CSRF攻击的全称是跨站请求伪造(cross site request forgery)，是一种对网站的恶意利用，尽管听起来跟XSS跨站脚本攻击有点相似，但事实上CSRF与XSS差别很大，XSS利用的是站点内的信任用户，而CSRF则是通过伪装来自受信任用户的请求来利用受信任的网站。你可以这么理解CSRF攻击：攻击者盗用了你的身份，以你的名义向第三方网站发送恶意请求。CRSF能做的事情包括利用你的身份发邮件、发短信、进行交易转账等等，甚至盗取你的账号。

#### 防御

* cookie设置为HttpOnly

CSRF攻击很大程度上是利用了浏览器的cookie，为了防止站内的XSS漏洞盗取cookie，需要在cookie中设置"HttpOnly"属性，这样通过程序(如JavascriptS脚本、Applet等)就无法读取到cookie信息，避免了攻击者伪造cookie的情况出现。

* 增加token

CSRF攻击之所以能够成功，是因为攻击者可以伪造用户的请求，该请求中所有的用户验证信息都存在于cookie中，因此攻击者可以在不知道用户验证信息的情况下直接利用用户的cookie来通过安全验证。由此可知，抵御CSRF攻击的关键在于：在请求中放入攻击者所不能伪造的信息，并且该信息不存在于cookie之中。鉴于此，系统开发人员可以在HTTP请求中以参数的形式加入一个随机产生的token，并在服务端进行token校验，如果请求中没有token或者token内容不正确，则认为是CSRF攻击而拒绝该请求。

* 通过Referer识别

根据HTTP协议，在HTTP头中有一个字段叫Referer，它记录了该HTTP请求的来源地址。在通常情况下，访问一个安全受限页面的请求都来自于同一个网站。比如某银行的转账是通过用户访问http://www.xxx.com/transfer.do页面完成，用户必须先登录www.xxx.com，然后通过点击页面上的提交按钮来触发转账事件。当用户提交请求时，该转账请求的Referer值就会是提交按钮所在页面的URL（本例为www.xxx.com/transfer.do）。如果攻击者要对银行网站实施CSRF攻击，他只能在其他的网站构造请求，当用户通过其他网站发送请求到银行时，该请求的Referer的值是其他网站的地址，而不是银行转账页面的地址。

因此，要防御CSRF攻击，银行网站只需要对于每一个转账请求验证其Referer值，如果是以www.xxx.com域名开头的地址，则说明该请求是来自银行网站自己的请求，是合法的。如果Referer是其他网站的话，就有可能是CSRF攻击，则拒绝该请求。

### SQL注入攻击

所谓SQL注入，就是通过把SQL命令伪装成正常的HTTP请求参数，传递到服务端，欺骗服务器最终执行恶意的SQL命令，达到入侵目的。攻击者可以利用SQL注入漏洞，查询非授权信息，修改数据库服务器的数据，改变表结构，甚至是获取服务器root权限。总而言之，SQL注入漏洞的危害极大，攻击者采用的SQL指令，决定攻击的威力。当前涉及到大批量数据泄露的攻击事件，大部分都是通过利用SQL注入来实施的。

#### 防御

* 使用预编译语句

预编译语句PreparedStatement是java.sql中的一个接口，继承自Statement接口。通过Statement对象执行SQL语句时，需要将SQL语句发送给DBMS，由DBMS先进行编译后再执行。而预编译语句和Statement不同，在创建PreparedStatement对象时就指定了SQL语句，该语句立即发送给DBMS进行编译，当该编译语句需要被执行时，DBMS直接运行编译后的SQL语句，而不需要像其他SQL语句那样首先将其编译。

前面介绍过，引发SQL注入的根本原因是恶意用户将SQL指令伪装成参数传递到后端数据库执行，作为一种更为安全的动态字符串的构建方法，预编译语句使用参数占位符来替代需要动态传入的参数，这样攻击者无法改变SQL语句的结构，SQL语句的语义不会发生改变，即便用户传入类似于前面' or '1'='1这样的字符串，数据库也会将其作为普通的字符串来处理。

* 使用ORM框架

防止SQL注入的关键在于对一些关键字符进行转义，而常见的一些ORM框架，如mybatis、hibernate等，都支持对相应的关键字或者特殊符号进行转义，可以通过简单的配置，很好的预防SQL注入漏洞，降低了普通的开发人员进行安全编程的门槛。通过#符号配置的变量，mybatis能够对输入变量的一些关键字进行转义，防止SQL注入攻击。

* 处理好相应的异常

后台的系统异常，很可能包含了一些如服务器版本、数据库版本、编程语言等等的信息，甚至是数据库连接的地址及用户名密码，攻击者可以按图索骥，找到对应版本的服务器漏洞或者数据库漏洞进行攻击，因此，必须要处理好后台的系统异常，重定向到相应的错误处理页面，而不是任由其直接输出到页面上。