PRIMETON TECHNOLOGIES，LTD.

上海普元信息技术股份有限公司

EOS 7.0开发规范

文档修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 版本号 | 修订日期 | 修订概述 | 修订人 | 备注 |
| 1 | 1.0 | 2013-6-24 | 初稿 | 曹宗伟 |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |

目录

1 引言 6

文档目的 6

文档范围 6

参考资料 6

2 资源命名规范 6

2.1 在线资源命名规范 8

2.1.1 模块 8

2.1.2 表单 8

2.1.3 视图 10

2.1.4 流程 11

2.2 离线资源命名规范 12

2.2.1 项目 12

2.2.2 构件包 13

2.2.3 逻辑构件 13

2.2.4 逻辑流 14

2.2.5 Java构件 14

2.2.6 组合构件 14

2.2.7 运算逻辑 15

2.2.8 页面流 15

2.2.9 工作流 16

2.2.10 数据集 16

2.2.11 数据实体 17

2.2.12 命名Sql集 17

2.2.13 命名Sql 17

2.2.14 页面资源 18

2.2.15 Spring Bean 18

2.2.16 Java 18

2.2.17 Javascript 20

2.2.18 包 20

2.2.19 WEB目录 21

2.2.20 国际化资源文件 21

2.3 配置资源 22

2.3.1 角色命名规范 22

2.3.2 功能菜单命名规范 23

2.3.3 业务字典命名规范 23

3 需求规范 24

3.1 需求分析规范 24

3.2 组件需求规范 26

4 设计规范 28

4.1 项目设计规范 28

4.2 构件包设计规范 29

4.3 数据建模规范 30

4.4 服务设计规范 31

4.5 组件设计规范 31

5 开发规范 32

5.1 在线开发规范 32

5.1.1 数据实体开发规范 32

5.1.2 表单开发规范 32

5.1.3 视图开发规范 33

5.1.4 流程开发规范 33

5.2 离线开发规范 34

5.2.1 逻辑流 34

5.2.2 Spring Bean 35

5.2.3 运算逻辑 37

5.2.4 页面流 39

5.2.5 工作流 40

5.2.6 命名Sql 40

5.2.7 页面资源 41

5.2.8 Javascript 42

5.2.9 Java 42

5.3 组件开发规范 46

5.3.1 组件开发规范 46

5.3.2 组件依赖规范 47

5.4 安全规范 48

5.4.1 防止SQL注入式攻击 48

5.4.2 防止跨站脚本(XSS)攻击 48

5.4.3 防止重定向漏洞 49

5.4.4 其他安全规范 49

5.5 性能规范 50

5.5.1 使用HttpSession 50

5.5.2 使用缓存(Cache) 50

5.6 其他开发规范 51

5.6.1 日志规范 51

5.6.2 异常规范 52

6 附录 53

6.1 术语表 53

# 引言

## 文档目的

本规范作为使用EOS7设计和开发业务系统的基本准则，用来指导和约束设计人员和开发人员如何更好地使用EOS7，是设计、开发工作的基础和依据，并作为项目质量评估的重要标准。通过规范开发过程，增强基于EOS7开发的设计模式、程序结构和开发框架的一致性以及代码的可维护性。

## 文档范围

在本规范中，“必须”表示该条目必须满足，违反这样的要求是原则性错误；“不允许”或“不可以”表示该条目绝对禁止；“应当”、“应该”或“建议”表示在某些特定条件下存在忽视该条目的理由，但该条目是否遵守需通过双方的共同约定；“不应当”或“不建议”表示所描述行为在某些特定条件下存在可接受或有效的理由，但该条目是否遵守需通过项目管理人员约定；“可以”表示该条目可选，但该条目是否遵守需通过项目管理人员约定。

## 参考资料

# 资源命名规范

为了使项目的文档、代码配置样式统一，使文档和程序有良好的可读性，使用EOS开发的项目必须遵循本章的命名规范。

无论数据库设计、程序文件命名、文档文件命名、变量命名，对于同一概念，如果需要使用缩写，统一使用如下定义。

|  |  |  |
| --- | --- | --- |
| **词语/缩写** | **英文描述** | **中文描述** |
| info | Information | 信息 |
| doc | Document | 文档 |
| src | Source Code | 源码 |
| demo |  | 项目的静态页面原型 |
| fea | Feature | 功能点，指经过设计后分解的功能 |
| req | Requirement | 需求点，指需求调研后形成的用户需求 |
| comp | Component | 业务组件 |
| query |  | 查询，在定义程序文件名称时，“查询”功能使用的英文为query，例如account\_query.jsp |
| add |  | 增加，在定义程序文件名称时，“增加”功能使用的英文名称为”add”，例如account\_add.jsp |
| update |  | 编辑，在定义程序文件名称时，“修改”功能使用的英文名称为“update”，例如account\_update.jsp |
| input |  | 输入，在定义程序文件名称时，“输入”功能使用英文名称为”input”，例如account\_input.jsp |
| delete | Delete | 删除，在定义程序文件名称时，“删除”功能使用的英文名称为“delete”，例如account\_delete.jsp |
| list |  | 结果列表使用的英文名称，例如account\_list.jsp |
| import | import | 导入，在定义程序文件名称时，“导入”功能使用的英文名称为“import”，例如dict\_import.jsp |
| export | export | 导出，在定义程序文件名称时，“导出”功能使用的英文名称为“emport”，例如dict\_export.jsp |
| rtn | Return | 返回，系统中表示返回的英文缩写 |
| success | Success | 系统中表示成功的英文单词均使用success |
| fault | fault | 系统中表示失败的英文单词均使用Fault |
| error | Error | 系统中表示错误的英文缩写 |

## 在线资源命名规范

EOS的在线开发环境为偏向业务的技术人员提供配置化的业务开发平台，包括配置数据实体、业务化表单、业务化流程、业务化视图等在线资源。

根据在线业务化配置的特点，在线资源应当包含部门和业务信息。本章节中的命名统一以XXX公司（简写xxx）研发部（简写rd）的请假业务（leave）为示例。

### 模块

* **概念**

模块是EOS7在线开发中基础的业务分组单元，其中包含了一组业务相关的数据实体、表单、视图和流程等资源。

* **命名规范**
  + 1. 模块ID使用英文字母、数字和下划线，以英文开头。
    2. 模块ID采用公司名简写 + “\_” + 部门简写+ “\_” +业务名称简写的方式，如xxx\_rd\_leave。
    3. 模块名称使用中文和”-”组成。
    4. 模块名称采用公司名中文简写 + “-” + 部门名+ “-” +业务名称中文简写的方式，如XXX公司-研发部-请假业务。

### 表单

* **概念**

表单是EOS7在线开发中的重要元素，采用可视化的方式配置信息录入界面，并和查询列表以及流程有机地结合，完成业务功能。

* **命名规范**

1. 表单ID使用英文字母、数字和下划线，以英文开头。
2. 表单ID采用公司名简写 + “\_” + 部门简写+ “\_” +业务名称简写 + “单据英文名称”的方式，如xxx\_rd\_leave\_leaveForm
3. 表单名称使用中文和”-”组成。
4. 表单名称采用公司名中文简写 + “-” + 部门名+ “-” +业务名称中文简写+ “单据名称”的方式，如XXX公司-研发部-请假业务-请假单。

#### 表单控件

* **概念**

表单控件是EOS7表单的基础元素，表单的可视化编辑主要是拖拽使用控件以及设置控件属性。

* **命名规范**

1. 表单控件ID使用英文字母、数字和下划线，以英文开头。
2. 表单控件ID在整个表单中必须唯一。
3. 在表单运行期，表单控件ID需要生成javascript变量，因此表单控件ID不能使用javascript的关键字，以免和javascript的全局变量冲突，表单控件ID允许使用break,delete,function,return,typeof,case,do,if,switch,var,catch,else,in,this,void,continue,false,instanceof,throw,while,debugger,finally,new,true,with,default,for,null,try等关键字；表单控件ID不建议使用frames,closed,defaultStatus,document,history,innerheight,innerwidth,length,location,name,Navigator,opener,outerheight,outerwidth,pageXOffset,pageYOffset,parent,Screen,self,status,top,window,screenLeft,screenTop,screenX,screenY,alert,blur,clearInterval,clearTimeout,close,confirm,createPopup,focus,moveBy,moveTo,open,print,prompt,resizeBy,resizeTo,scrollBy,scrollTo,setInterval,setTimeout等关键字。
4. 表单控件标签名使用中文和”：”组成，如果是必填项目，则需要以”(\*)”结尾。

#### 表单脚本

* **概念**

表单脚本是在表单的控件事件中编写的javascript代码，在浏览器中运行，与表单控件结合实现复杂的业务场景。

* **命名规范**

表单脚本必须符合javascript代码规范，请参考javascript代码规范。

### 视图

* **概念**

视图是EOS7在线开发中的重要元素，采用可视化的方式配置查询条件以及查询结果的展示界面。

* **命名规范**

1. 视图ID使用英文字母、数字和下划线，以英文开头。
2. 视图ID采用公司名简写 + “\_” + 部门简写+ “\_” +业务名称简写 + “视图英文名称”的方式，如xxx\_rd\_leave\_listLeaveForm。
3. 视图名称使用中文和”-”组成。
4. 视图名称采用公司名中文简写 + “-” + 部门名+ “-” +业务名称中文简写+ “视图名称”的方式，如XXX公司-研发部-请假业务-请假单列表。

#### 视图查询控件

* **概念**

视图控件是EOS7视图的基础元素，视图控件包括视图中的查询条件控件以及视图中的操作按钮。

* **命名规范**

1. 视图控件ID使用英文字母、数字和下划线，以英文开头。
2. 视图控件ID在整个视图中必须唯一。视图控件不允许使用视图中已经使用的ID，如result-submit-form,condition-form,list,condition-form-hidden,isPagination,page.isCount,page.length,page.count,page.isFirst,page.isLast,page.size,page.currentPage,result-submit-form,result-form。

#### 视图脚本

* **概念**

视图脚本是在视图的操作按钮上编写的javascript代码，在浏览器中运行，与视图结合实现复杂的业务场景。

* **命名规范**

视图脚本必须符合javascript代码规范，请参考javascript代码规范。

### 流程

* **概念**

流程是是EOS7在线开发中的重要元素，采用图形化连线的方式配置工作流程。

* **命名规范**

1. 流程ID使用英文字母、数字和下划线，以英文开头。
2. 流程ID采用公司名简写 + “\_” + 部门简写+ “\_” +业务名称简写 + “流程名称”的方式，如xxx\_rd\_leave\_leaveFlow。
3. 流程名称使用中文和”-”组成。
4. 流程名称采用公司名中文简写 + “-” + 部门名+ “-” +业务名称中文简写+ “流程名称”的方式，如XXX公司-研发部-请假业务-请假流程。

#### 流程图元

* **概念**

流程图元是EOS7流程的基本单元，采用图形化连线的方式配置工作流程。

* **命名规范**

1. 流程图元ID使用英文字母、数字和下划线，以英文开头。
2. 流程图元ID采用骆驼命名法，ID必须为业务化名称，不允许出现manualActivity1之类的名称。
3. 流程图元名称使用中文和”-”组成。
4. 流程图元名称必须是业务化的名称，符合主谓短语，譬如人力资源部领导审批。

#### 流程脚本

* **概念**

流程脚本是在流程的自动活动或图元事件中编写的javascript代码，在JVM虚拟机中运行，与后台的逻辑或服务结合实现复杂的业务场景。

* **命名规范**

表单脚本必须符合javascript代码规范，请参考javascript代码规范。

## 离线资源命名规范

EOS的离线开发环境为技术人员提供编码开发业务的技术开发平台，以实现扩展组件或复杂业务场景。

根据离线开发环境的特点，离线资源应当以名称空间为基础。本章节中的命名统一以XXX公司（简写xxx）的请假业务（leave）为示例，所有带包名的资源都必须以com.xxx.leave开头。

为保证程序逻辑或代码的可追溯性，要求每个文件都保留正确合适的开发人员信息，在开发工具安装的时候，有一个步骤是设置作者和姓名，这些设置的信息，也可以通过开发工具 Studio的首选项查看和重新设置，如下图：

针对创建者和部门，按照如下要求进行设置：

作者：输入开发环境使用者正式姓名的中文，例如张山，则输入”张山”，不允许使用开发人员的英文名或者系统缺省的用户名，如administrator

部门：输入开发环境使用者所在公司（或部门）的名称，例如“XXX公司 ”，“信息中心研发部”等

在做好上面的设置后，每个新建的文件均会按照设置填入缺省的姓名和部门。有利于系统生成有效的代码分析报告。

### 项目

* **概念**

项目是IT业务系统源代码的基础单元，是业务系统或业务组件源代码的基本组织形式，一个项目可以包含多个构件包，以及WEB、Java库、system.sdvx等固定的目录或文件。

* **命名规范**

1. EOS7的项目名称全部使用英文小写。
2. EOS7的业务项目采用长命名空间的方式，如com.xxx.leave。
3. EOS7的业务组件项目采用长命名空间的方式，如审批业务组件的项目 命名为com.xxx.components.approve。

### 构件包

* **概念**

构件包是EOS7项目中基础的业务实现和业务部署单元，其中包含了构件，构件实现以及一些额外需要的资源（如配置信息、页面资源等）

* **命名规范**

1. 构件包名称全部使用英文小写。
2. 构件包名称采用长命名空间的方式，如com.xxx.leave.org。
3. 构件包名＝系统命名空间+”.”+业务描述的英文单词（缩写），例如org（缩写参见资源命名规范）。
4. 构件包命名在设计期由设计人员确定，不允许开发期确定。
5. 显示名称描述为中文信息。
6. 构件包属于一个抽象概念，但构件包创建后，会创建一个与构件包同名的包名。

### 逻辑构件

* **概念**

逻辑构件是对外提供服务功能的单元，定义了构件的实现方式，以及构件提供了哪些服务，引用和属性，采用EOS的逻辑流实现。

* **命名规范**
  + 1. 逻辑构件名称为构件对象的单词（或缩写），其后可选如Manager等动词，首字母大写，与java类命名一致。如UserManager、MenuManager。
    2. 逻辑构件直接建在以构件包为名称的包下，原则上不增加新的包的定义。
    3. 逻辑构件名称不使用前缀或后缀。
    4. 显示名称描述为中文信息。
    5. 逻辑构件是包含一个单一实体管理功能的操作集合，即增(add)、删(delete)、改(update)、查(query)等。

### 逻辑流

* **概念**

逻辑流属于逻辑构件的操作，它是采用EOS图形化组装方式表达的一种程序逻辑。

* **命名规范**

1. 逻辑流名称参照java方法命名方式，首字母小写，如对Employee的增、删、改、查的完整名称就是 addEmployee、deleteEmployee、updateEmployee、queryEmployee。
2. 如果名称太长，建议使用词汇的缩写。
3. 显示名称描述为中文信息。

### Java构件

* **概念**

采用Java语言实现的构件，符合SCA\_JavaAnnotationsAndAPIs\_V100的规范。

* **命名规范**
  + 1. Java构件名称中单词的首字母大写，与java类命名一致，如UserManager。
    2. Java构件的包命名空间为：“com.”+”企业名称”+应用代号+”.” +构件包名（不含前缀）+“.service”，例如com.xxx.leave.comm.service。

### 组合构件

* **概念**

组合构件包含在构件装配目录下，由一组具有业务上相互联系的构件（逻辑构件、Java构件、组合构件）组成，本身也可以定义对外的服务和引用，其实质是一种更大粒度的构件。

组合构件可以定义构件的服务，引用关系，构件的实现等。

* **命名规范**

1. 组合构件名称为构件的业务对象描述的单词，首字母小写，如sample。
2. 组合构件直接建在以构件包为名称的包下，原则上不增加新的包的定义。
3. 组合构件名称不使用前缀或后缀。
4. 组合构件中，组装的构件名称为业务对象+后缀”Com”，例如sampleCom。
5. 只有当项目与外部应用存在服务接口时，才创建组合构件。
6. 显示名称描述为中文信息。
7. 组合构件中定义服务接口时的命名规范为：动作＋对象，例如finishActivityInsttance。

### 运算逻辑

* **概念**

运算逻辑是支撑业务处理的一组原子操作，对应于Java类的一个方法，运算逻辑可以在逻辑流中被组装。

EOS平台提供了一系列运算逻辑，通过对该系列运算逻辑的组合，基本可以实现业务的处理。

* **命名规范**

1. 运算逻辑类的包路径规划：“com.” +”企业名称”+应用代号+”.” +构件包名（不含前缀）+“.bizlets”例如com.xxx.leave.comm.bizlets
2. 运算逻辑类的命名规范遵循Java类命名规范，首字母大写，类根据业务来划分，类名建议使用业务对象名称，由多个单词构成时，每个单词首字母大写。
3. 每个运算逻辑类包含一组具有某些共同特性的运算逻辑，禁止将运算逻辑杂乱无章分布在不同的运算逻辑类中，正确的实例如业务异常操作类com.xxx.leave.comm.bizlets.BusiException.java,在该类下包含一组与业务异常相关的运算逻辑。
4. 运算逻辑命名符合java方法命名规则，其，例如writeLog，如果涉及多个单词（缩写），则每个单词（缩写）的首字母大写。
5. 运算逻辑名称不使用前缀或后缀（与EOS5对运算逻辑的命名规范有差异）。
6. 某些运算逻辑支持参数可以按组增加，则建议运算逻辑名后加“Batch”，如setNodeValueBatch。

### 页面流

* **概念**

页面流用来描述单个HTTP请求的处理过程或者多个请求/页面之间的跳转关系。

* **使用原则**

一个逻辑构件对应一个页面流，即一个页面流中是包含一个实体的所有页面操作。

一个功能模块可能涉及多个实体间的交互操作，可设计多个页面流，采用调用(对应实体的)子页面流方式。

* **命名规范**

1. 页面流名称＝业务对象＋动作，其中业务对象单词（缩写）全部小写，动作（缩写）的首字母大写，如employeeManager，原则上应与逻辑构件名称一致。
2. 与某个构件相关的页面流，创建一个与该构件名相同的包(放置在构件包同名的包下)，页面流创建于该包下，例如机构管理构件包存在一个工作组管理的构件（workgroup），则在页面流下面创建workgroup的包，与工作组管理相关的页面流创建在workgroup下。
3. 尽量不要在页面流上直接实现业务逻辑，而是在逻辑流中实现，在页面流中调用。
4. 页面流的名称为英文命名，表达使用动宾短语，宾语部分首字母大写。如：edit，query，queryEmp。
5. 页面流显示名称描述为中文信息。

### 工作流

* **概念**

工作流是针对工作中具有固定程序的常规活动而提出的一个概念，通过将工作活动分解定义良好的任务、角色、规则和过程来进行执行和监控，达到提高生产组织水平和工作效率的目的。

* **命名规范**

参见流程设计开发规范部分。

### 数据集

* **概念**

数据集是一系列实体的集合，用来统一管理实体。

* **命名规范**

1. 数据集名称采用小写，如session、employee、orginzation等。
2. 数据集直接建在以构件包为名称的包下，原则上不增加新的包的定义，例如com.xxx.leave.auth.auth.datasetx。
3. 所有数据集中设计的实体，以实体导入时的命名为准。

### 数据实体

* **概念**

数据实体是EOS提供的实体建模功能，开发人员可以使用一种图形化的方式对应用的数据进行建模。数据实体包括普通实体、持久化实体、查询实体。

* **命名规范**

1. 数据实体命名采用Java类名规范，如Orginzation，Employee、QueryUser等。
2. 数据实体对应的表名前缀必须是系统英文名称或模块英文名的缩写（大写），如ORG\_Orginzation。
3. 数据实体字段的名称必须是易于理解，能表达字段功能的英文单词或缩写英文单词。数据实体字段的名称采用Java类成员变量命名规范，如userId。
4. 如果多个表中存在很多同类型的字段，比如FirstName，要在这个字段的名字加上表前缀，如：custFirstName。

### 命名Sql集

* **概念**

是一组命名Sql的集合，体现为一个以.namingsqlx为后缀的XML描述文件。

* **命名规范**

1. 命名sql集名称采用首字母小写，并直接建在以构件包为名称的包下，如com.xxx.leave.auth.query.namingsqlx。
2. 命名sql集名称不使用前缀或后缀，直接以业务对象单词（或其缩写）命名。

### 命名Sql

* **概念**

命名SQL就是给一个sql起一个唯一标识(id)，以及配置该sql执行所需要的参数和如何处理sql执行完后的结果。然后在使用的时候，传入id和参数即可。

* **命名规范**

1. 命名Sql名称＝操作＋对象，操作全部小写，对象首字母大写，如deleteDictType，queryUserRoles。
2. 操作分为：query、update、insert、delete。
3. 命名SQL中，字段名用大写，映射结果集的属性也大写。

### 页面资源

* **概念**

页面资源的类型包括：JSP、HTML、JS、图片等。

* **命名规范**

1. 在构件包下为每个构件建立一个文件夹，名称与构件名称一致（如果逻辑构件名带”Manager” 可以去掉），文件夹下创建与该构件相关的页面资源，例如构件busidict，在页面资源下创建busidict目录。
2. 如果构件涉及较多的不同页面资源，如js、jsp、图片资源比较多，可以在构件同名的文件夹下再创建script和image，并将相应资源放置在目录下。
3. 页面资源文件一律使用小写字母，命名一般采用主谓格式，各个单词之间使用下划线“\_”连接，如org\_tree.jsp、dictionary.js。

### Spring Bean

### Java

* **概念**

EOS7中所有JAVA文件（包括运算逻辑和JAVA构件将自动呈现在Java目录下），另外，允许通过该目录开发项目中其他的JAVA程序，包括工具类，测试类、Tag类、Filter类等。

* **命名规范**

1. 建议在开发环境中设置好格式，开发过程当中用快捷方式自动格式化。
2. Package 的命名**：**Package 的名字全部采用小写字母，命名规则如下：“com.” +”企业名称”+应用代号+”.” +构件包名（不含前缀）+“类别”，例如：com.xxx.leave.comm.user。

此外，对于包名我们做如下约定：

1. 工具函数类包名前缀为.util，如com.xxx.leave.comm.util。
2. Servlet类包名前缀为.servlet，如com.xxx.leave.comm.servlet。
3. test case类包名前缀为.test，如com.xxx.leave.comm.test。
4. Tag类包名前缀为.tag，如com.xxx.leave.comm.tag。
5. Filter类包名前缀为.filter，如com.xxx.leave.comm.filter。
6. Class 的命名

Class 的名字必须由大写字母开头而其他字母都小写的单词组成，例如：DataFile或InfoParser。

1. Class 变量的命名

变量的名字必须用一个小写字母开头。后面的单词用大写字母开头，例如：debug 或 inputFileSize。

1. Static Final 变量的命名

Static Final 变量的名字应该都大写，并且指出完整含义，例如：MAX\_UPLOAD\_FILE\_SIZE=1024。

1. 参数的命名

参数的名字必须和变量的命名规范一致。

1. 数组的命名

数组应该总是用下面的方式来命名：

byte[] buffer;

而不是：

byte buffer[];

1. 方法的参数命名

使用有意义的参数命名，如果可能的话，使用和要赋值的字段一样的名字：

SetCounter(int size)

{

this.size = size;

}

### Javascript

* **概念**

Javascript是一种由Netscape的LiveScript发展而来的原型化继承的动态类型脚本语言。

* **命名规范**

遵循相关配置规范。

1. 文件编码统一为utf-8书写过程过， 每行代码结束必须有分号";"。
2. 变量命名: 驼峰式命名。 原生JavaScript变量要求是纯英文字母， 首字母须小写， 如iEmployee;私有变量要求首字符为’\_’， 其他与原生JavaScript 规则相同， 如: \_iEmployee。
3. 类命名: 首字母大写， 驼峰式命名。 如 IEmployee。
4. 函数命名: 首字母小写驼峰式命名。 如getEmployee()。
5. 命名语义化， 尽可能利用英文单词或其缩写。
6. 尽量避免使用存在兼容性及消耗资源的方法或属性， 比如eval() & innerText。
7. JavaScript非注释类中文字符须转换成unicode编码使用， 以避免编码错误时乱码显示。
8. 代码结构明了， 加适量注释。 提高函数重用率。
9. 注重与html分离， 减小reflow， 注重性能。

### 包

* **概念**

EOS允许在很多资源类型下创建包，用以构造项目的命名空间。

EOS创建构件包时，会自动创建构件包同名的包，创建逻辑构件时，也会创建与逻辑构件同名的包。

* **命名规范**

1. 包命名要求为全小写。
2. 包命名格式为：项目名 + 包功能名，例如：com.xxx.leave.emplyee。
3. 在各个资源下创建包的规范参见各个资源的命名规范，例如逻辑构件、组合构件等。

### WEB目录

* **概念**

WEB目录是针对整个项目的WEB配置目录，允许放置对整个应用有效的资源文件，包括公共的js、图片、页面、以及WEB-INF的配置等。

* **命名规范**

1. WEB目录下文件的命名规范。

目录结构

common -放置公共JSP文件。

images - 存放项目中运用的图片资源。

scripts - 存放项目中公共的JavaScript资源。

styles - 存放项目中运用的样式文件。

1. 构件包（功能）级资源严禁放在WEB目录下。

### 国际化资源文件

* **概念**

页面的信息的国际化内容都存放在构件包的配置信息下的resources/i18n目录下的资源文件中

* **命名规范**

1. 资源文件的命名规范

资源文件的命名按照构件包和子构件包划分，每个构件（子）包对应一组资源文件，每组资源文件包括多个语种的对应信息，如机构维护的构件包为oganization则国际化资源文件为oganization\_en\_US.properties,oganization\_zh\_CN.properties…等。

1. Key的命名
2. 全局使用的消息名：

固定名称、标题、按钮显示名称：以l\_开头加上英文单词

例如： 选择 l\_choose

确定 l\_comfirm

取消 l\_cancle

字段名： 实体名.属性名

例如：AcOrganization.orgid

提示信息：以m\_+”开头加上页面操作信息”+”\_”+”消息名称”

例如：删除时的提示信息m\_delete\_confirm

图片名称：以i\_开始加上英文单词

例如：i\_logo

1. 局部使用的消息名(构件包、页面流、JSP内使用)

局部使用的消息名称可在全局使用的消息命名规范基础上，在消息名前增加”页面流名称”+”\_”

如 选择：EmployeeManager.\_ l\_choose

字段名： EmployeeManager.\_ AcOrganization.orgid

删除时的提示信息EmployeeManager.\_m\_delete\_confirm

## 配置资源

配置资源是使用EOS提供的应用基础框架配置角色、功能菜单和业务字典等与应用相关的基础内容。上述配置内容在应用开发过程中与业务功能联系比较紧密，在多个应用集中部署和配置时容易引起混乱。在配置角色、功能菜单和业务字典等内容时请遵循以下规范：

### 角色命名规范

* **概念**

角色可以称成为权限集，表示系统中权限一个子集，用于控制用户可以使用的功能集合，赋予用户一个角色表示给用户一定功能的使用权限。角色的ID和名称应当包含角色范围（公司级或部门级）、业务类别等信息。

* **命名规范**

1. 角色ID使用英文字母、数字和下划线，以英文开头。
2. 角色ID采用公司名或部门名简写 + “\_” +业务名称简写 + “\_” + 角色名的方式，如xxx\_leave\_employee或者rd\_leave\_manager。
3. 角色名称使用中文和”-”组成。
4. 角色名称采用公司名或部门名中文简写 + “\_” +业务名称中文简写 + “\_” + 角色名的方式，如XXX公司-请假业务-管理员。

### 功能菜单命名规范

* **概念**

业务字典维护了业务编码与名称的对应关系，在数据库是存储简单字符，而在界面中展示业务化含义的文字。业务字典的ID应当包含业务类别等信息。

* **命名规范**

1. 菜单ID使用英文字母、数字和下划线，以英文开头。
2. 菜单ID采用公司名或部门名简写 + “\_” +业务名称简写 + “\_” + 菜单名的方式，如xxx\_leave\_queryEmployee。
3. 菜单名称使由中文组成。
4. 菜单名称应当简洁扼要，便于理解，长度控制在4到6个汉字。

### 业务字典命名规范

* **概念**

菜单是系统中作为用户功能访问入口显示信息，访问或者点击菜单是通过与功能的关联来间接访问功能。菜单的ID应当包含菜单范围（公司级或部门级）、业务类别等信息，而菜单名称则要简洁扼要，便于理解。

* **命名规范**

1. 业务字典ID使用英文字母、数字和下划线，以英文开头。
2. 业务字典类型代码一律使用大写。
3. 业务字典ID采用业务名称简写 + “\_” + 字典名的方式，如，业务系统之间共享的字典应当以COMMON开头，如COMMON\_GENDER。
4. 业务字典名称使由中文和-组成。
5. 业务字典名称应当采用业务名称中文简写 + “-” + 字典名的方式，如请假-请假类别，业务系统之间共享的字典应当以公共字典-开头，如公共字典-性别。

# 需求规范

EOS实现在线的业务配置与平台化的组件扩展相结合，解决快速开发以及技术团队分工合作的问题，建立领域业务平台，提高应用的快速交付和复制推广的能力。

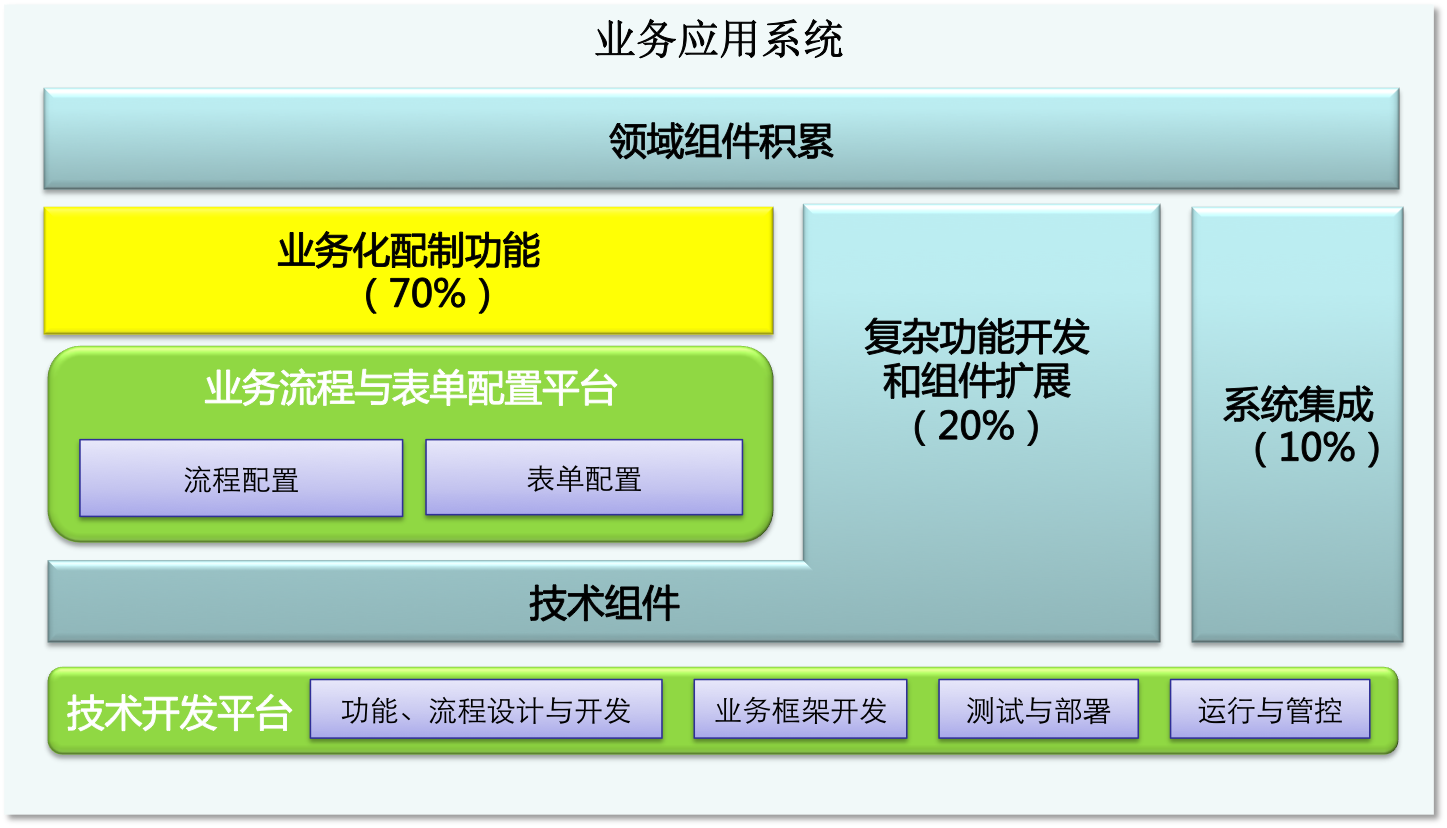


图 1业务应用系统分工

在一个IT业务系统中，增删改查类业务和流程类业务占到应用总功能的70~80%左右，这类业务可以通过EOS提供的业务化配置功能在线配置开发完成。复杂功能和系统集成功能占应用总功能的20~30%，这类功能需要高层次技术人员开发。通过EOS的组件扩展机制，复杂功能逐步积累成技术组件或业务组件。

## 需求分析规范

使用EOS开发业务系统时，需求人员应当从领域业务平台的维度审视业务需求，对需求进行分解，哪些需求是业务系统独有的需求，哪些需求是可以推广到其他客户的需求？哪些需求可以放进领域业务平台以提高开发效率？下面以某公司发文需求为例说明如何做领域业务平台的需求。

某公司发文的需求流程如下：

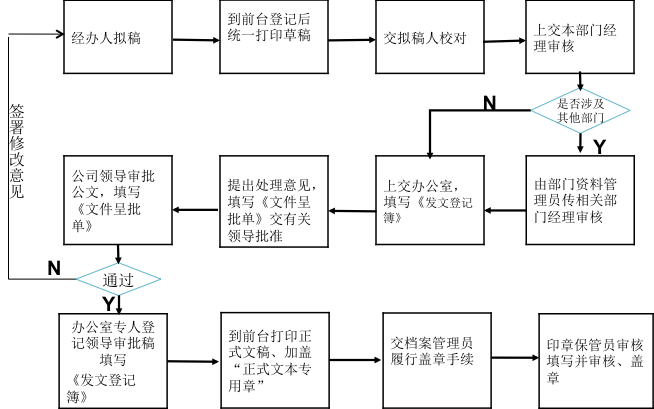


图 2 某公司发文流程

经过需求调研以及对发文的需求分解，得到的业务需求如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块 | 功能需求 | 已有组件 | 特有需求 | 推广需求 |
| 拟稿 | 填写稿纸头 | 表单组件 |  |  |
| 生成文稿 |  | 是 |  |
| 编辑文稿 |  |  | 文档类业务 |
| 保存草稿 |  |  | 办公类业务 |
| 从草稿打开 |  |  | 办公类业务 |
| 提交草稿 | 流程组件 |  |  |
| 校对 | 填写批注 |  |  | 文档类业务 |
| 提交文稿 | 流程组件 |  |  |
| 打回文稿 | 流程组件 |  |  |
| 审批 | 填写审批意见 | 流程组件 |  |  |
| 查看审批历史 | 流程组件 |  |  |
| 维护常用审批意见 | 流程组件 |  |  |
| 多级审批 |  |  | 管理类业务 |
| 电子签章 |  |  | 文档类业务 |
| 登记 | 登记发文单 |  | 是 |  |
| 电子签章 |  |  | 文档类业务 |
| 存档 | 流程组件 |  |  |
| 界面 | 稿纸头表单 | 表单组件 |  |  |
| 在线编辑文稿（word） |  |  | 文档类业务 |
| 选择送批的领导 |  |  | 管理类业务 |
| 填写审批意见 | 表单组件 |  |  |
| 电子签章 |  |  | 文档类业务 |
| 查看审批历史 | 表单组件 |  |  |
| 查看流程流转信息 | 表单组件 |  |  |
| 其他功能 | 传阅 | 流程组件 |  |  |
| 转发 | 流程组件 |  |  |
| 代办 | 流程组件 |  |  |

其中，在线编辑文稿（word）和电子签单在文档类业务中广泛使用，多级审批和选择送批的领导在管理类业务中广泛应用，保存草稿和从草稿中打开则在办公类业务中广泛使用。在做需求分析的时候，需要考虑把这些业务需求做到领域业务平台，在后续业务开发过程中，如果遇到类似的需求，则可以使用已有组件开发业务。

## 组件需求规范

与业务需求不同，业务组件的需求更抽象、更复杂，需要有良好的业务适应性，业务组件的需求分析过程应当遵循以下规范：

1. 业务组件的需求背景是领域业务平台，建立领域业务平台，提升业务配置开发效率，提高应用的快速交付和复制推广的能力。
2. 业务组件的使用者是开发人员和业务人员，开发人员使用业务组件开发出业务功能，业务人员使用业务功能。而业务的使用者则是业务人员，仅需要定义业务人员如何使用业务功能。以审批组件为例，开发人员开发审批的业务功能时配置在流程的哪些环节可以审批，哪些环节不能审批；而业务人员则在相应的环节填写审批意见，查看审批历史。
3. 业务组件的场景需要考虑开发人员使用业务组件开发不同的业务，满足不同业务的功能需求。以审批组件为例，在领导审批环节，业务需求多种多样，譬如本部门领导审批、二级单位领导审批、用户指定领导审批等多种类型的业务需求。业务组件应当满足不同的业务功能需求。
4. 通过对不同用户、不同业务、不同功能的分析和抽象，形成业务组件需求表格，以审批组件为例，形成的表格如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 组件功能 | 开发人员 | 业务人员 | 业务 |
| 设置流程环节是否需要审批 | 设置B、C环节需要审批，A环节不需要审批 | 在A环节无法填写审批意见，在B、C环节可以填写审批意见 | 发文 |
| 设置审批人 | 设置为指定的领导审批 | 提交给指定的领导审批 | 发文 |
| 设置为提交者所在部门经理审批 | 提交后流转到提交者所在部门经理 | 请假、报销 |
| 设置为协办单位审批 | 选择协办单位，提交后流转到协办单位审批 | 发文 |
| 用户指定领导审批 | 选择送审领导，提交后流转到指定的领导 | 发文、请假 |

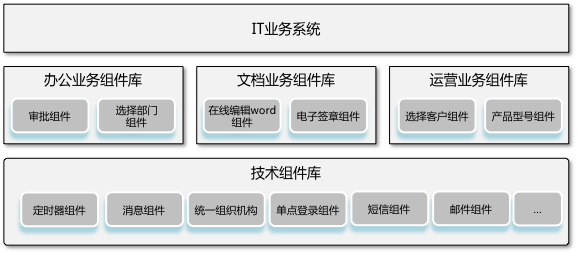
1. 业务组件的界面需求应当分为两部分：开发人员使用业务组件的界面和业务人员看到的业务界面。两者之间有很强的关联性，其界面描述和界面元素规则需要组合说明。

# 设计规范

EOS提供了强大的设计功能，包括项目管理、构件包设计、数据建模设计、服务设计、组件设计等功能。

## 项目设计规范

与开源框架的项目组织方式不同，EOS可以多个项目对应一个web应用，各自开发，分开部署。使用EOS开发业务系统时，设计人员应当从技术组件库与领域业务组件库的角度划分项目。



一般来说，IT系统的平台建设分为三层：技术平组件库、领域业务组件库和业务应用层。

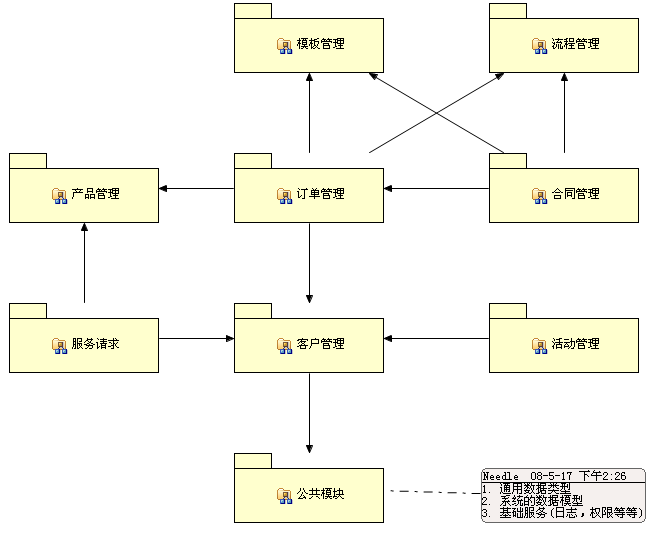
1. 技术组件库，该层提供IT业务系统所需要的基础技术能力，譬如短信组件、邮件组件等技术组件，大部分IT业务都需要用到基础技术组件。
2. 领域业务组件库，该层按照业务领域划分业务组件，譬如办公类业务的审批组件、选择部门组件等业务组件。
3. IT业务系统层，该层是基于技术组件库、领域业务组件库之上开发的业务功能，可以组合使用技术组件库和领域业务组件库中的组件。

使用EOS平台设计项目时应当遵循的规范如下：

1. 每个技术组件或业务组件必须是独立的EOS项目，代码分开存放。
2. IT业务系统的项目则根据业务系统的复杂度和模块划分建立项目，原则上建议采用模块化、松耦合的方式划分项目。
3. 项目之间通过EOS导出部署包的方式依赖。
4. 项目划分应当由架构师负责，架构师分析业务需求，综合考虑技术组件库、领域业务组件库和IT业务系统各层因素之后划分项目。

## 构件包设计规范

EOS提供可视化的构件包设计能力，为设计人员提供便利的工具在一个项目中划分构件包，在连线的方式配置构件包之间的依赖关系。



使用EOS设计构件包时应当遵循以下规范：

1. 模块进行切分时大小适中（功能相对独立和完整，适合1-2人独立完成）, 一个构件包相当于一个业务模块，避免构件包太大，影响业务的复用；避免构件包太细，导致项目构件包数量太多，不利于管理。
2. 清晰的模块结构、合理的依赖关系，特别是大型应用项目设计，需要避免多个模块之间建立复杂的网状依赖关系，需要规避循环依赖。在项目初期就需要设计合理的构件包依赖关系。构件包及其依赖关系应当在设计阶段由设计人员确定，开发人员不允许修改构件包的依赖关系。
3. 在构件包设计过程中，业务人员可以抽取出一些可以重用的构件包。比如将系统的所有数据实体放置到一个构件包中，将一些通用的服务放置到一个构件包中。
4. 一个构件包一般是具有相对独立的一组功能的子系统；会被多个模块调用的基础业务数据和功能最好作为独立的构件包存在，构件包之间应避免产生循环依赖关系。
5. 每个构件包对应一个与其名称一致的web路径，web路径地址必须以组件名或业务名为前缀，建立二级目录，如approve/userManager。

## 数据建模规范

数据模型是构建一个业务系统的基础，在进行具体的设计前，需要首先设计系统的数据模型。设计人员可以先通过其它数据建模工具创建数据库表后导入到EOS系统中生成数据实体。另外，设计人员也可以直接在EOS中进行数据实体设计，然后通过EOS创建数据库表的功能生成数据库表。

EOS中数据模型由三种类型的数据实体： 非持久化实体 、持久化实体 和查询实体 。在数据建模的视图中，EOS提供以下设计功能：

1. 设置实体间的关联关系。
2. 设置实体和数据库表的映射关系。
3. 将一个查询SQL语句映射成一个数据实体。
4. 根据数据库的Schema生成数据实体。

使用EOS设计数据模型应当遵循以下规范：

1. 禁止使用用户录入的内容做为主健，如社会保障号码或身份证号码，因为一旦输入错误，只能删除整个记录重新录入。
2. 数据实体设置尽可能设置为可以动态添加属性，以方便业务开发过程中使用扩展数据。
3. 数据实体的关联数据加载设置应当设置为懒加载，在不使用从表时提高数据访问性能。
4. 建议数据实体之间建立关联关系，以方便多表数据操作，而表之间不建立强约束。

## 服务设计规范

服务设计过程是对构件包设计中划分的每个构件包进行进一步细化，也是对应用框架中接口定义的进一步细化，包括对每个模块中的构件进行划分，对构件之间的引用关系、构件的服务接口进行设计等。EOS提供设计服务的能力，可以设置符合SOA规范的Web服务。使用EOS设计服务时需要遵循以下规范：

1. 应该保证事务的完整，一个接口里包含的所有业务都在一个事务内。在服务调用出错后事务全部回滚。调用方在批量调用接口的时候，须考虑事务的完整性和延续性，确保出错后能多次调用以完成整个事务，来保障整个业务事务的统一。
2. 必须进行安全配置及安全验证。业务系统必须在服务内部进行IP地址、密码校验等安全限制，非正常的系统访问予以拒绝，同时每次调用记录服务调用日志。
3. 服务的业务出现异常后，必须采用统一的方式定义异常编码，抛出异常信息，并采取相应的处理措施。

## 组件设计规范

组件是软件系统中具有相对独立功能、可以明确辨识、接口由契约指定、和语境有明显依赖关系、可独立部署的可组装软件实体。基于EOS的组件是包括页面流、逻辑构件、数据模型、服务、工作流、Web资源、配置信息和Java文件等资源的，物理上可独立、可隔离、可依赖、可部署的构件包级别组件，可完成完整的业务功能，并且实现业务功能的各层资源都可复用。

设计组件应当遵循以下规范：

1. 功能完整：功能完整并不是指构件不需要依赖于其他组件就能运行，而是指一组相似的、关联性较强的功能应该组织在同一个组件中进行管理。此外，组件还应该是其自身运行时依赖资源（例如：配置文件、图片等）的完整集合。
2. 自描述性**：**自描述性是指组件中描述自身提供的功能性和非功能性的信息，这些信息都是组件的一部分。例如：组件的功能描述、接口描述、数据描述、版本描述以及依赖关系等。通过这些描述，组件消费者可以不需要参考其他信息就能正确的使用该组件。
3. 抽象性：组件应当有适度的技术和业务抽象，在业务开发的过程中，组件需要被多个业务系统使用，所以组件需要具有广泛的适应性，这种适应性建立在设计人员对技术和业务的抽象基础之上。
4. 独立性：组件必须在物理上可独立、可隔离、可依赖、可部署。每个组件的源代码应当是隔离的，但是组件内部的源代码应当时高内聚的，以构件包的方式组合在一起。
5. 可组装：组件可以在定义良好的体系结构下方便地组装到软件系统中，也可以与其它组件组装成为粒度更大的组件。

# 开发规范

使用EOS开发业务，需要根据业务的情况决定开发方式，在线开发适用于企业内部业务系统之中增删改查类业务以及工作流程类业务，在线开发不适用于网站等对界面要求较高的业务。项目管理人员需要根据上述原则安排搭建业务开发的环境以及分配开发任务。

## 在线开发规范

EOS的在线开发环境为偏向业务的技术人员提供配置化的业务开发平台，适用于界面整齐的管理类信息系统，在线开发对开发人员的技术水平要求不高。

### 数据实体开发规范

使用EOS在线配置数据实体需要遵循以下规范：

1. 项目管理人员根据项目的开发团队状况以及项目阶段设置数据实体与数据库的同步策略（不同步、增量同步或全部同步）。
2. 在线的数据实体可以引用在EOS离线开发环境（EOS Studio）中定义的数据模型，下拉框和视图等只查询的在线资源可以使用引用的数据实体。
3. 在线配置如果有复杂的查询如跨多表查询应当使用查询实体。
4. 与流程相关的数据实体必须在表中增加字段processInstId，以方便与流程结合。

### 表单开发规范

使用EOS在线配置表单需要遵循以下规范：

1. 界面中使用面板或分组控件分为区块，使用两列或三列表格布局，同一行表格中放高度相近的控件，多行文本和富文本等较高的控件需要跨行排列。
2. 使用日期控件时必须设定日期的格式，日期格式必须符合企业IT业务系统的统一规范。
3. 输入控件需要增加校验，必填控件与可选填控件有明确的区分，譬如在标签上增加”\*”或者必填控件的背景有明显的区分。
4. 在不同流程环节（如录入环节、审批环节）时表单的控件是否显示以及是否可录入应当通过状态配置。界面变化较大的表单需要使用多个表单绑定到同一个数据实体上。
5. 表单与后台交互必须使用后台逻辑服务，后台逻辑服务通过逻辑流实现，与前端交互的采用HTTP + JSON的ajax方式交互。
6. 表单的扩展控件开发应当根据项目的时间和资源情况确定，在项目较紧急的情况下应当优先使用HTML控件满足业务需求，在项目结束后再抽取为业务组件。

### 视图开发规范

使用EOS在线配置视图需要遵循以下规范：

1. 视图的查询条件尽量采用模糊匹配，以增加用户体验。
2. 视图的查询结果渲染需要采用正确的方法，业务字典控件录入的内容需要使用相同的业务字典渲染，带其他样式或图片的渲染需要使用表达式。
3. 视图的查询结果默认应当根据日期倒序排列，如业务中有特殊要求时需要根据特殊要求排列。
4. 视图的操作按钮应当在运行时根据不同的角色设置不同的权限。
5. 视图与后台交互必须使用后台逻辑服务，后台逻辑服务通过逻辑流实现，与前端交互的采用HTTP + JSON的ajax方式交互。

### 流程开发规范

使用EOS在线配置流程需要遵循以下规范：

1. 建议使用人工开始作为流程的开始图元，人工开始图元支持先填表单后启动流程，并且取消时不会产生任务。
2. 在业务变化不大的情况下建议一个流程使用一个表单，人工图元环节应当与表单的状态绑定。如果业务变化较大则可以在某些人工图元环节绑定其他表单。
3. 应当使用流程的自动活动脚本和人工活动图元的事件脚本调用后台逻辑和后台服务，流程的脚本运行在JVM中，采用Javascript语法，内置流程上下文以及常用API。
4. 表单与流程交互数据时应当使用业务变量，表单保存数据时会自动更新流程中定义的业务变量。
5. 人工活动环节的参与者不确定时必须勾选允许前驱活动指派选项。
6. 在某些业务流程中环节较多，如果所有的活动放在同一个层次，会使得流程图拓扑结构复杂而不容易识别。应当把业务功能相近或者相关的活动组组成子流程，可以使整个业务流程表达清晰简单。
7. 当活动图元个数较多、流程图中活动布局拥挤时，建议少用路由分支；当分支条件很复杂时，适当使用路由分支可以简化分支条件时，建议使用路由分支。

## 离线开发规范

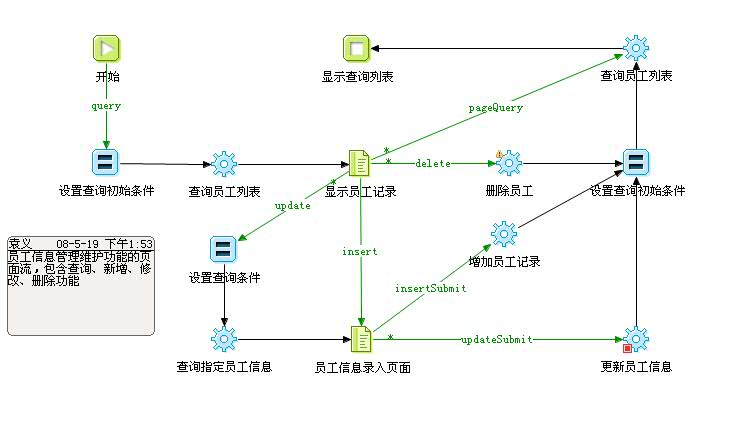
EOS在线配置平台提供的功能有限，不能满足复杂界面以及复杂的扩展需求，EOS离线开发工具则提供了强大的一体化开发工具配合在线配置平台完成业务功能。离线开发主要用于以下场景：

1. 配合开发在线配置平台调用的后台逻辑或服务。
2. 扩展开发在线配置平台所需要的控件。
3. 实现复杂的业务功能。

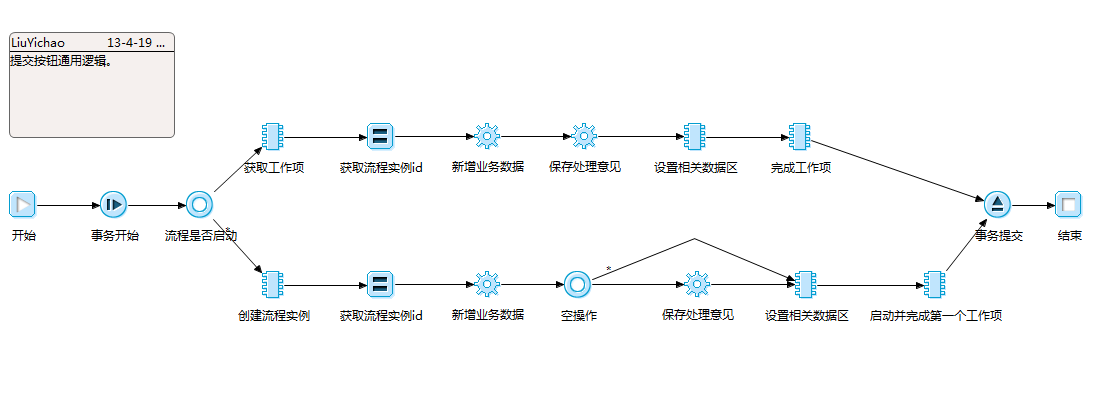
### 逻辑流

1. 要求逻辑流尽可能做到简洁功能独立，简洁是指所实现的逻辑流条理清晰、业务算法简单易于理解，要求在熟悉EOS体系架构和已有构件（可能是EOS提供的，也可能是自身的积累）的基础上，选择最佳的算法实现；功能独立是指在一个逻辑流中的业务处理是独立的，复杂的逻辑流考虑细分为多个小的逻辑流，以提高逻辑流的复用度。例如，某个功能要求查询获得多个数据表的结果，在编写逻辑流时不建议将所有查询处理放到一个逻辑流中实现，而是在不同的逻辑流中实现，再通过页面流调用多个逻辑流，这样，编写的逻辑流可能还能在别的功能中得到复用，充分利用到面向构件积累的优势。

以下是某项目中页面流的截图，该逻辑响应用户界面上查询员工信息的详细资料（包括增加员工信息，删除员工信息，查询员工列表）的动作，对应信息的查询完全可以在一个逻辑流中实现，也可以分为三个逻辑流实现（如图所示），然后通过页面流分别调用，这样分别实现的三个逻辑流除了满足此功能外，还可以在员工管理、组织机构管理等功能中得到复用。



1. 编写的逻辑流要求以**业务的语义**描述每个处理环节（避免使用程序语义描述或者不描述），即更改逻辑流中调用的运算逻辑的显示名称，同时逻辑图形编排清晰，尽可能不出现交叉线，另外可以在逻辑流中增加注释图元，注明逻辑流的功能等，这样有利于别人的阅读和理解，也方便日后系统的维护，以下是某项目中的提交逻辑流截图：



### Spring Bean

1. 要求Spring Bean的开发都是面向接口的编程，所有的Spring Bean的实现都有两部分，一个是功能接口类，一个是功能接口实现类。下面是创建Spring Bean向导页面截图：



1. Spring Bean的事务控制，使用声明式事务控制，spring.xml配置文件中的默认事务控制策略配置部分内容如下（从配置可以看出，方法名称定义需要注意前缀）：

|  |
| --- |
| <!-- default transaction policy -->  <bean id="DefaultNamingTransactionAttribute" class="org.springframework.transaction.interceptor.NameMatchTransactionAttributeSource">  <property name="properties">  <props>  <prop key="add\*">PROPAGATION\_REQUIRED</prop>  <prop key="insert\*">PROPAGATION\_REQUIRED</prop>  <prop key="create\*">PROPAGATION\_REQUIRED</prop>  <prop key="update\*">PROPAGATION\_REQUIRED</prop>  <prop key="delete\*">PROPAGATION\_REQUIRED</prop>  <prop key="save\*">PROPAGATION\_REQUIRED</prop>  <prop key="remove\*">PROPAGATION\_REQUIRED</prop>  <prop key="query\*">PROPAGATION\_REQUIRED,readOnly</prop>  <prop key="\*">PROPAGATION\_REQUIRED,readOnly</prop>  </props>  </property>  </bean> |

1. Spring Bean除了配置文件之外，都是Java代码，所以需要遵循Java开发规范。

### 运算逻辑

1. 首要原则：运算逻辑的开发是受限的

**项目组的成员不能随意增加运算逻辑**，业务处理尽可能通过在逻辑流中复用已有的运算逻辑来组装，如果自己想增加一个运算逻辑需要和系统设计负责人及时沟通讨论后再做决定（这样做有利于业务与代码的分离，便于逻辑流未来的修改和维护，同时基于java代码实现的运算逻辑，可能会给系统带来诸多隐形的BUG，给系统的稳定运行造成隐患，也不利于系统的后期维护）。

1. 第二原则：尽管不提倡开发过多的运算逻辑，那么什么场景下需要开发运算逻辑呢?
   1. 一些没有包含在已有构件库中的与业务无关的原子服务，对于该类运算逻辑，要求由研发支持的接口人提供实现，并提交给研发
   2. 业务处理中需要调用外部系统或其他第三方产品的接口，需要将这些接口通过运算逻辑进行封装，例如与第三方系统（如CAS）存在协议的接口处理、调用ESB的接口、调用LDAP认证的接口等等
   3. 既有系统已经有一些基础处理的JAR包，其中提供的某些操作在EOS的运算逻辑中没有，希望通过运算逻辑封装这些操作形成在EOS逻辑流中可以拖拽的运算逻辑（使用EOS提供放入运算逻辑导入功能）
   4. 系统中存在某些业务处理需要对大量业务数据进行汇总统计，或者针对一个很大的查询结果（超过1000行以上）进行迭代处理，而迭代处理中涉及大量的操作。
2. 编写运算逻辑时，要求填写完整相应描述信息，包括运算逻辑的描述及参数的描述
3. 运算逻辑原则上建议不抛出异常，而是通过捕捉异常后设置返回值返回，异常信息可以通过日志接口记录到日志文件中。如果抛出异常，要求能在构件对应的文档中明示抛出哪些异常，建议不要抛出基类异常。
4. 在运算逻辑中如果存在多处错误返回，应根据不同错误返回不同返回码（如-1、-2、-3……），这样有助于进行错误定位。
5. 运算逻辑中有关调试、出错提示等，一律使用EOS平台发布的日志记录接口，详细用法参见EOS的日志接口文档。
6. 一般情况下，运算逻辑正常返回值为1；返回负值表示捕捉到异常或因数据不合要求导致处理中断，需要进行分支处理；返回0或其他正数表示处理完成，但可能存在某些警告信息，例如用expandEntity查询指定条件的某条记录时，可能符合查询条件的记录有多行，则返回第一行记录，而返回值为记录行数。
7. 运算逻辑参数项设置原则：
8. 原则一：对于可以设置多组参数的运算逻辑，最后一个接口参数，在类型后面增加…，表示在调用运算逻辑时，可以动态增加参数的个数，例如String…，Date…
9. 原则二：不建议运算逻辑设置太多参数项（对于支持多组参数的除外），如果参数项超过5个，建议充分利用SDO对象，通过Entity的节点方式处理，例如发送邮件的运算逻辑sendMail，将整个邮件信息（如发件人、收件人、抄送人、主题、内容、附件等等）放入一个SDO对象中以一个参数传入，而不是每项作为一个参数传入，在运算逻辑中通过取SDO对象的属性获得各项参数。
10. 运算逻辑是通过JAVA代码实现的，JAVA代码的编写请参考有关JAVA代码编写规范的资料。
11. 关于运算逻辑中事务处理的建议：

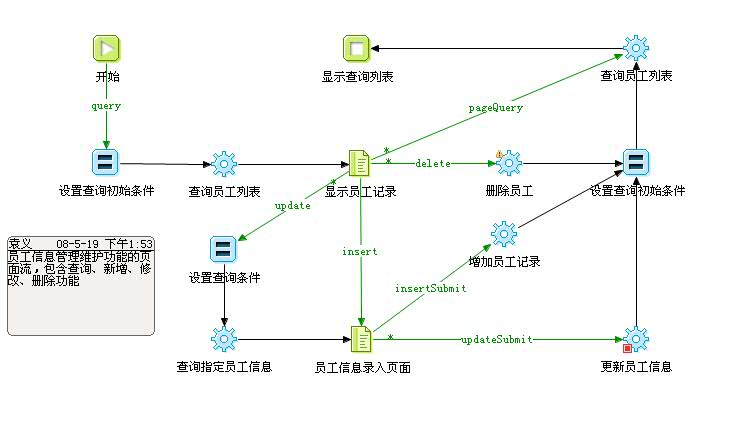
建议在运算逻辑中尽量不使用事务处理，而是将事务的控制放到调用运算逻辑的逻辑流中（通过事务图元进行控制），这样避免因为运算逻辑和逻辑流中事务使用混乱导致出现事务一致性的问题。

注意事务中的数据库操作时间不要过长，因为事务会锁表，时间过长，别的连接对同一表操作会长时间等待，而且可能以为因为锁冲突导致死锁。本注意事项可能与上面的建议发生冲突，例如在运算逻辑中因为操作数据库比较多事务时间必然较长的时候，还是需要在运算逻辑处理事务。

1. 不要在运算逻辑方法中使用流，在datacontext尽量避免用流，防止未知异常导致流无法正常关闭！
2. 注释规范：功能简要说明，输入，输出，关键点说明。

### 页面流

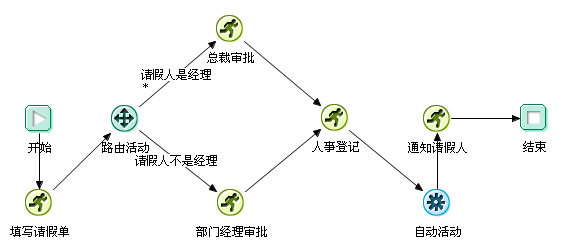
1. 编写的页面流要求以业务的语义表达每个处理环节（避免出现程序语义或者不描述），同时逻辑图形编排清晰，尽可能不出现交叉线，另外要求在页面流中增加注释图元，注明页面流的功能等，这样有利于别人的阅读和理解，也方便的日后系统的维护，以下是某项目中的页面流截图：



1. 页面流的图形应该简单明了，如果一个页面流上面出现超过20个以上的图元(不包含开始和结束图元)，则需要考虑该逻辑是否应该优化，如判断是否处理繁琐冗余，是否可以变为多个页面流，是否调用的逻辑流粒度太细等。
2. 考虑逻辑流调用的返回值，针对非正常的返回要有统一的出错处理（如定位到统一的错误页面等）
3. EOS中支持在页面流中调用运算逻辑，为了项目中层次清晰，原则上不建议在页面流中调用运算逻辑，尤其禁止在页面流中调用对数据产生更新的运算逻辑。而是将该类操作封装在逻辑流中，页面流调用逻辑流的方式
4. 页面流对于Session数据的处理原则：
   1. SessionContext数据将在会话期间占用内存，只适合保存会话期间的全局数据，尽可能避免在会话数据中存放过多数据量
   2. 对于MUO对象，由于会自动复制到requestContext和bizContext，更加需要控制存放的数据大小。
5. 注释规范：功能简要说明，关键点说明。

### 工作流

1. 简单的工作流不建议离线开发，业务复杂的工作流需要离线开发。
2. 编写的工作流要求以业务的语义表达每个环节（避免出现程序语义或者不描述），同时逻辑图形编排清晰，注意连线的横平竖直，尽可能不出现交叉线，另外要求在工作流中增加注释图元，注明工作流的功能等，这样有利于别人的阅读和理解，也方便的日后系统的维护。



1. 在工作流应用中设计业务表的时候，在业务表中保存流程实例id的字段PROCESSINSTID一定要设计为number，不然通过根据业务实体查询工作项的查询语句效率会非常慢。
2. 注释规范：功能简要说明，关键点说明。

### 命名Sql

1. sql写法要求符合5.6节的SQL语言开发规范。

### 页面资源

1. 建议使用NUI方式进行页面开发，NUI和Richweb框架不允许混合使用。
2. 使用NUI开的的JSP界面需要在HTML前增加DOCTYPE声明，如：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> |

1. JavaScript的开发，要放在<body>标签中，建议放在结尾，不允许放在一般标签中（比如div等）。
2. JSP页面尽量少用Java，尤其是复杂的算法。
3. JSP页面中不允许进行数据库的操作。
4. JSP中引用资源必须使用绝对路径，以应用名开关，应用名使用字符串定义，如：

|  |
| --- |
| <%  String contextPath=request.getContextPath();  %>  <link id="css\_skin" rel="stylesheet" type="text/css" href="<%=contextPath%>/ap/sce/preview/resources/css/style.css"/>  <script type="text/javascript" src="<%=contextPath%>/common/nui/nui.js"></script> |

1. JSP的文件编码、pageEncoding，以及<head>中的contentType必须指定为UTF-8编码，如：

|  |
| --- |
| <%@page pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>用户管理</title>  </head>  <body>  </body>  </html> |

### Javascript

1. 保证代码压缩后不出错，为了提高性能，项目上线时应当对项目包含的所有Javascript文件进行压缩处理，编写Javascript应当保证压缩后不出错。
2. 保证代码能通过特定Eclipse的自动格式化功能，帮助团队实现统一换行、缩进、空格等代码编排。
3. 不使用生偏语法，不建议使用三元表达式以提高代码可读性，如：

不建议使用：

|  |
| --- |
| function getText(type){  return type == 2 ? "正常" : (type == 3 ? "异常" : (type == 4 ? "告警" : null));  } |

建议使用：

|  |
| --- |
| function getText(type){  var typeData={"2":"正常","3":"异常","4":"告警"};  if (typeData[type]){  return typeData[type];  }else{  return null;  }  } |

1. 应当减少使用全局变量和全局函数。
2. 尽量避免使用存在兼容性及消耗资源的方法或属性, 比如eval() & innerText。
3. 当代码调整或重构后，之前编写的不再使用的代码应该及时删除，以降低维护成本。

### Java

1. 文件头声明

源文件的头部需要一个history段，对于每次对源文件的重大改动，都需要在history段中注明。该段定义在package和import之间，例如：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 描 述：Exception 处理的基类

\* 前置条件:

\* 后置条件:无

\* 实施资源:

\* 调用者 :

\* 被调用者:

\* 重载说明:

\* Copyright: Copyright (c) 2013

\* Company: XXX.

\* @author testuser testuser@xxx.com

\* @version 1.0 2013/3/12

\* @see

\* HISTORY

\* 2013/06/01 Biz 创建文件

\* 2013/06/19 kevin 增加提示功能

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. import顺序

import包按一下顺序：

1. jdk标准包
2. java扩展包（例如servlet，javamail，jce等）
3. 使用的外部库的包（例如xml parser）
4. 使用的项目的公共包
5. 使用的模块的其他包

每一类import后面加一个换行。例如：

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.mail.\*;

import org.apache.xml.\*;

import com.bankcomm.wms.\*;

import com.bankcomm.wms.util.\*;

1. 代码块书写格式

可以选择以下任意一种代码块的书写方式：

if (true){

//body

}

if (true)

{

//body

}

建议使用第二种书写方式。如果是修改他人的代码，必须使用代码原来的书写方式。

对于代码块过长，超过1屏以上，}后面要说明属于那个代码块，例如：

if (i > 100)

{

//too many lines more than one screen

}// if (i > 100)

1. 缩进规范

缩进使用4个连续空格，不要在源文件中保存tab字符， 请注意调整所用的IDE工具，打开将tab转换为空格功能。

1. 页宽规范

页宽应该设置为80字符. 源代码一般不会超过这个宽度, 并导致无法完整显示, 但这一设置也可以灵活调整. 在任何情况下, 超长的语句应该在一个逗号或者一个操作符后折行. 一条语句折行后, 应该比原来的语句再缩进4个空格。

1. 操作符规范

操作符左右各用一个空格分隔。

例如：

int a = b;

if (a > 0);

1. 类和方法定义规范

类定义或方法定义过长需要换行书写，例如：

public class CounterSet

extends Observable

implements Cloneable

private PortletSet getPortlets( Portlets portlets,

RunData rundata,

boolean application,

boolean applicationsOnly )

{

}

1. 注释规范

public 和 protected的成员变量和方法必须写javadoc注释。超过1句以上的注释使用中文书写。对于代码多于10行的private方法也要写javadoc注释。

对于代码中的逻辑分支或循环条件需要书写注释，例如：

if (some condition)

{

//符合某个条件，应该这样处理

}

else

{

//否则应该那样处理

}

1. 属性规范

类中的属性不能定义为public变量直接存取，而是定义成protect变量并编写get/set方法，例如：

protect String myName;

public String getMyName()

{

return myName;

}

public void setMyName(String myName)

{

this.myName=myName;

}

## 组件开发规范

按照组件的适用范围，组件分为两种：技术组件和业务域组件。技术组件是专门解决某个技术问题或者提供基础的技术能力，在IT系统建设中可用性范围比较广泛；业务域组件则是只适合在某个业务领域使用的组件，其特点是组件的颗粒度较大，业务化程度较高。

### 组件开发规范

组件的开发必须遵循以下规范：

1. 组件应当由一个或多个EOS构件包组成，提供包括数据模型、Java类、运算逻辑、逻辑流、服务、UI、页面流以及工作流等可供其他组件或业务功能调用的综合性技术能力。
2. 组件应当及时更新版本号，组件发布时应当更新构件包的版本号，以方便使用者了解组件的相关信息。
3. 组件更新时应当保证组件的兼容性，组件提供的API以及服务接口和业务规则应当兼容以前版本。
4. 在组件中实现启动时处理业务的功能时，应当通过构件包的ContributionListener实现。构件包的ContributionListener配置在handler-contribution.xml中，配置示例如下：

|  |
| --- |
| <handlers>  <handler handle-class="com.primeton.runtime.resource.event.ContributionDemoListener"/>  </handlers> |

1. 在组件中配置参数时应当使用构件包的配置文件能力，即：使用构件包的contribution.eosinf配置文件，该配置文件采用三段式的方式，配置文件示例如下：

|  |
| --- |
| <module name="Module1">  <group name="Group1">  <configValue key="key1">value1</configValue>  <configValue key="key2">value2</configValue>  …  </group>  …  </module>  … |

1. 组件的web资源包括JSP、HTML、Javascript、CSS、图片等必须放在构件包的WEB目录下，不允许放在项目的WEB目录下。
2. 组件开发标签、Web Service服务等相关的资源时，所需要的TLD或配置文件可以放到WEB目录的WEB-INF目录下面。
3. 组件不应当修改web.xml，如需要增加拦截器时应当通过EOS的API增加对某一后缀资源的拦截或处理。

### 组件依赖规范

组件的依赖关系必须遵循以下规范：

1. 在组件中使用第三方jar包时，应当放入WEB目录的WEB-INF/lib目录下，使用第三方jar包时必须保证与EOS和其他组件使用的jar包不冲突。
2. 组件之间的依赖关系必须明确，只允许同级依赖或向下依赖，不允许反向依赖，如：技术组件不允许依赖业务组件。
3. 组件之间不允许产生循环依赖，如需相互调用资源时应当使用扩展机制。
4. 技术组件转为业务组件时，必须先解决其他技术组件依赖该技术组件的依赖关系，确认没有其他技术组件依赖该技术组件时才允许迁移。
5. 业务组件下沉为技术组件时，必须先解决该业务组件依赖其他业务组件的依赖关系，确认该业务组件没有依赖其他业务组件时才允许下沉为技术组件。

## 安全规范

IT业务系统开发应当符合一般的安全性要求，防止常见的网络攻击如：SQL注入攻击和跨站脚本攻击，同时，应当避免产生类似重定向漏洞和session安全等漏洞。

### 防止SQL注入式攻击

SQL注入就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令获得数据的非法攻击行为。

防范SQL攻击需要遵循如下规范：

1. 对用户的输入进行校验。可以通过正则表达式检查或限制输入长度，必须对输入的单引号和 双"-"进行转换。
2. 一般使用参数化的SQL或者使用存储过程进行数据查询存取，而不是动态拼装SQL。遇到某些特殊情况时（如元数据不明确等），可以动态拼装SQL，但是必须考虑SQL的安全性。
3. 不能使用[管理员](http://baike.baidu.com/view/315045.htm)权限进行数据库连接。为每个应用使用单独的权限和有限的数据库连接。
4. 不要在代码中直接填写机密信息。加密或者hash掉密码和敏感的信息。
5. 必要的情况下可以使用专业的代码漏洞扫描工具来检查是否存在SQL注入的漏洞。

### 防止跨站脚本(XSS)攻击

跨站脚本攻击(XSS),指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到攻击者的特殊目的。

防范跨站脚本攻击需要遵循如下规范：

1. 进行输入验证。某个数据被接受为可被显示或[存储](http://baike.baidu.com/view/87682.htm)之前，使用标准输入验证机制，验证所有输入数据的长度、类型、语法以及业务规则等。
2. 强壮的输出编码。数据输出前，确保用户提交的数据已被正确进行编码，建议对所有字符进行编码而不仅局限于某个子集。
3. 明确指定输出的编码方式(如UTF-8)。不要允许攻击者为用户选择编码方式。
4. 注意黑名单验证方式的局限性。仅仅查找或替换一些字符(如"<" ">"或类似"script"的关键字)，很容易被XSS变种攻击绕过验证机制。
5. 警惕规范化错误。验证输入之前，必须进行解码及规范化以符合应用程序当前的内部表示方法；同时应用程序对同一输入不做两次解码。

### 防止重定向漏洞

如果应用程序提取用户可控制的输入，并使用这个数据执行一个重定向，指示用户的浏览器访问一个不同于用户要求的URL，诱使受害者访问一个欺骗性Web站点并输入敏感信息，就会造成重定向漏洞。

防范重定向漏洞需要遵循如下规范：

1. 从应用程序中删除重定向页面，用直接指向相关目标URL的链接替代指向重定向页面的链接。
2. 建立一个包含所有有效重定向URL的列表。不以参数的形式向重定向页面传送目标 URL，相反，传送这个列表的一个索引。重定向页面应在它的列表中查询这个索引，并返回一个指向相关URL的重定向。
3. 应用程序应在所有重定向中使用相对 URL，重定向页面应严格确认它收到的URL是一个相对URL。

### 其他安全规范

1. 不在URL中传输session标识符。
2. 不在输出流或者日志功能输出系统数据或调试信息，产生信息泄漏。
3. 不使用 System.out 或 System.err，而要用专门的日志记录工具(Log4j)来记录日志。
4. 系统上线前，要删除只用于Debug的代码和不会使用到的代码。
5. 尽量不允许用户上传文件，如果一定要上传文件，应将上传的文件存储在一个无法通过 Web 访问的目录中，并且去除文件执行权限。
6. 不使用硬编码（Hard-coded）存放身份验证信息，如用户口令密码。
7. 对于异常和例外情况进行正确的捕捉和检查，比如不使用空的Catch块。
8. 对下载的数据进行完整性检查。
9. 防止XML注入，包括XML数据注入、可扩展样式表语言转换（Extensible Stylesheet Language Transformation，XSLT）注入和XPath/XQuery 注入，对系统使用的XML输入报文、XSLT和XPath等进行严格校验。

## 性能规范

一般情况下，使用EOS开发IT业务系统时，应当谨慎使用HttpSession以及Cache，以避免出现性能问题，使用HttpSession以及Cache时应当遵循相应的规范。

### 使用HttpSession

需要使用HttpSession应遵循如下规范：

1. 在JSP页面中不要创建默认的HttpSession，可使用以下页面指令加以避免：< %@ [page](http://whatis.ctocio.com.cn/searchwhatis/395/6025895.shtml) session="false"%>。
2. HttpSession中只能存放较小的数据，每个用户的数据在2-4K左右，比如用户名等基本信息。不要将大的对象存储在HttpSession中，避免序列化带来的计算开销，影响系统性能和吞吐量。
3. 用完后释放HttpSession。当不再使用HttpSession时，调用HttpSession.invalidate()方法使session失效。
4. 设置合适的超时值。servlet引擎有一个默认的超时值。如果不删除session，而是等待session超时，超时值越大，对系统性能的影响也越大。超时值设置分为三种情况：非交互Web 页面（网站）设置为2分钟；Portal设置为15分钟；其他设置为60分钟。
5. 当某个请求导致系统后端处理时间超过HttpSession超时值时，应采用异步请求机制，不需要操作员等待处理完成。

### 使用缓存(Cache)

在以下情况下使用缓存：

1. 大量的频繁的数据库IO操作。例如从数据库循环获取信息时，如果从缓存中读取则能提高系统性能。
2. 一些常用的页面JSP以及相关的Javascript、CSS的加载，可以通过缓存减少频繁的下载、页面渲染等操作。

应用程序使用的缓存技术包括：

1. 对象缓存
2. 可以使用由O/R Mapping框架（如Hibernate）提供的缓存机制，透明性访问，细颗粒度缓存数据库查询结果，无需业务代码显式编程。
3. 对象缓存分类包括：对映射数据库表记录的entity对象进行缓存，对1对n关系的集合进行缓存，对n对1关系的关联对象进行缓存。
4. 查询缓存
5. 对数据库查询结果集进行缓存，类似数据库的Query Cache。
6. 适用于一些耗时，但是时效性要求比较低的场景。查询缓存和对象缓存适用的场景不一样，两者互为补充。
7. 当查询结果集涉及的表记录被修改以后，需要清理缓存。
8. 页面缓存
9. 对频繁使用的页面或者请求链接应进行缓存处理，以提高页面渲染速度。
10. 页面缓存应该包括动态页面静态化，Servlet缓存和页面局部缓存。
11. 动态页面静态化 利用模板技术将访问过一次的动态页面生成静态html，同时修改页面链接，下一次请求直接访问静态链接页面。
12. Servlet缓存 针对URL访问返回的页面结果进行缓存，适用于粗粒度的页面缓存，例如新闻发布等。
13. 页面局部缓存 针对动态页面的局部片断内容进行缓存，适用于一些个性化但不经常更新的页面。
14. 对于过期或者数据变更需要支持自动清理缓存，应避免手动清除客户端浏览器缓存。

## 其他开发规范

使用EOS开发应用时，还应当遵循日志和异常规范。

### 日志规范

使用EOS开发应用时，应当遵循以下日志规范：

1. 应当事先规划好日志，譬如业务日志、安全日志等日志信息，根据业务系统需要决定记录到文件系统或数据库中。
2. 应用相关的日志禁止使用EOS平台日志的appender，特别是system.log和engine.log。system.log是EOS的系统日志，用于分析系统运行状况和性能；engine.log是EOS的引擎日志，用于业务的全路径跟踪、业务调错和业务性能分析。
3. 应用如下情况需要记录业务日志：
   1. 所有对业务数据的更新，如增加、修改、删除
   2. 重要的操作，如登录、退出、敏感信息的查询
4. 针对业务功能写业务日志，而不是针对每个数据库操作写业务日志，例如，增加员工只写一次业务日志，而增加员工在处理中可能做了几个表的数据变动。
5. 写业务日志的逻辑调用放在逻辑流中，并且放置在业务处理的事务之外，以防止回滚时擦除日志。

### 异常规范

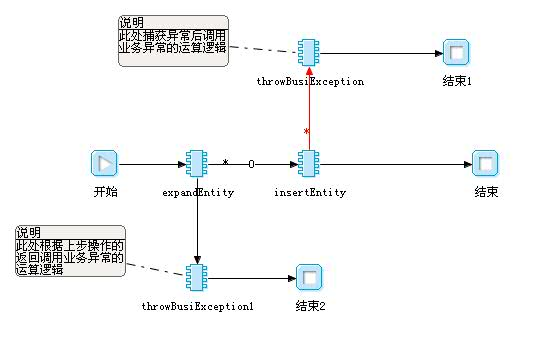
使用EOS开发应用时，应当遵循以下异常规范：

1. 在编码开始之前分配模块的错误码，错误码的构成规则：构件包名+”\_”+四位数字：例如auth\_0001，以保证在出异常时可以快速定位到错误原因。
2. 在构件包中建立异常资源文件，针对错误码，包括显示在页面的提示和真正的错误提示，例如：

auth\_0001=无效的用户名或者密码![用户{0}密码错误！]

其中{0}表示变量，如果异常提示中有多个变量，依次为{0}，{1}…[]内的信息为真正的错误信息。如果没有，则和前面信息一致。显示给用户的信息为[]前的信息。

1. 异常资源文件为构件包资源配置目录下resources/exception/exception.properties
2. 必须在处理异常的逻辑之后增加结束图元，例如：



1. 处理了异常的情况下，不再需要调用写业务日志的逻辑。

# 附录

## 术语表

|  |  |
| --- | --- |
| 名词 | 说明 |
| IT业务系统 | IT业务系统是由计算机硬件、网络和通讯设备、计算机软件、信息资源、信息用户和规章制度组成的以处理信息流为目的的人机一体化系统。 |
| 业务日志 | 业务日志主要包括操作日志和安全日志，各IT业务系统的业务日志必须挂接到业务协同平台。 |
| 菜单 | 菜单是系统中作为用户功能访问入口显示信息，用户通过点击菜单访问IT业务系统的功能。 |
| 业务组件库 | 业务组件库是一套针对组件的生产、改进、管理、沉淀和发展的完整软件管理机制。 |
| 构件 | 构件对外提供服务功能的单元，定义了构件的实现方式，以及构件提供了哪些服务，引用和属性。 |
| 构件包 | 构件包是EOS系统发布和复用的基本单位，它由逻辑流、页面流、服务构件、Java代码、页面资源等组成。一个构件包通常能够完成一个相对独立、完整的业务功能。 |
| 逻辑流 | 逻辑流用于完成后端业务功能。它是一种基于图形化编程的构件实现模式。一个逻辑构件是由多个逻辑流构成的，每个逻辑流采用图形化的方式将小粒度的运算构件组装成一段业务逻辑。它由开始节点、结束节点、运算逻辑节点、事务处理节点和连线等组成的图形。 |
| 页面流 | 页面流负责管理UI交互。用于描述用户界面元素流转关系的图形化流程，管理页面和页面间的跳转关系，以及页面和页面间共享的数据。它由开始节点、结束节点、逻辑流节点、服务节点（可选节点）、返回页面节点和连线等组成的。 |
| 工作流 | 工作流用于完成包含人工和自动的业务活动流程的构件，通过BPS for EOS开发环境来开发的流程管理的图形化描述。 |
| 运算逻辑 | 运算逻辑用来封装一些最基础的运算方法。一个运算逻辑对应于一个Java类，不同的是，运算逻辑是加了特定的JDK5.0 Annotation（注释）的类。运算逻辑方法又称BL（Bizlet）方法，对应一个Java类的静态或实例方法，用来完成特定的计算处理任务。运算逻辑可以为静态Java方法也可以为非静态的Java方法。 |
| BPS | BPS是针对工作中具有固定程序的常规活动而提出的一个概念，通过将工作活动分解来定义良好的任务、角色、规则和过程，进而进行执行和监控，达到提高生产组织水平和工作效率的目的。 |
| 数据实体 | 数据实体是EOS提供的实体建模功能，开发人员可以使用一种图形化的方式对应用的数据进行建模。数据实体包括普通实体、持久化实体、查询实体。 |
| 命名SQL | 命名SQL给一个SQL起一个唯一标识，并且指定如何处理SQL所需要的参数和执行完后的结果。命名SQL可以在参数和返回值中使用实体，用户可以用命名SQL执行对应SQL的全部操作。 |