

EPICSAAS平台开发规范参考

EPIC云平台

2014年6月

版本

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日期 | 作者 | 评审人 | 修订内容 | 版本 |
| 2014-06-3 | 郭华刚 |  | 创建目录结构及整理基本内容 | 0.1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 开发规范概述 - 1 -](#_Toc390345898)

[2 Spring框架 - 2 -](#_Toc390345899)

[2.1 Spring配置说明 - 2 -](#_Toc390345900)

[2.1.1 全局Spring配置 - 2 -](#_Toc390345901)

[2.1.2 数据源相关Spring配置 - 2 -](#_Toc390345902)

[2.1.3 服务相关Spring配置 - 2 -](#_Toc390345903)

[2.1.4 WEB相关Spring配置 - 2 -](#_Toc390345904)

[2.1.5 SSB相关Spring配置 - 2 -](#_Toc390345905)

[2.2 Spring注解使用要求 - 3 -](#_Toc390345906)

[2.2.1 将对象纳入Spring容器管理 - 3 -](#_Toc390345907)

[2.2.1.1 @Service注解使用要求 - 3 -](#_Toc390345908)

[2.2.1.2 @Component注解使用要求 - 3 -](#_Toc390345909)

[2.2.1.3 @Repository注解使用要求 - 3 -](#_Toc390345910)

[2.2.2 注入依赖对象 - 3 -](#_Toc390345911)

[2.2.2.1 @Resource注解使用要求 - 3 -](#_Toc390345912)

[2.2.3 Spring Bean生命周期管理注解 - 4 -](#_Toc390345913)

[3 持久层 - 5 -](#_Toc390345914)

[3.1 持久层类结构说明 - 5 -](#_Toc390345915)

[3.2 持久层包结构说明 - 6 -](#_Toc390345916)

[3.3 持久层开发步骤说明 - 6 -](#_Toc390345917)

[3.4 MyBatis使用要求 - 8 -](#_Toc390345918)

[3.5 MyBatis开发参考 - 8 -](#_Toc390345919)

[4 服务层 - 9 -](#_Toc390345920)

[4.1 服务层基本说明 - 9 -](#_Toc390345921)

[4.2 服务层开发说明 - 9 -](#_Toc390345922)

[4.3 服务事务管理说明 - 9 -](#_Toc390345923)

[4.4 服务本地导入与调用说明 - 9 -](#_Toc390345924)

[4.5 服务远程导出与调用说明 - 10 -](#_Toc390345925)

[5 WEB层 - 11 -](#_Toc390345926)

[5.1 控制器相关 - 11 -](#_Toc390345927)

[5.1.1 URL路径映射规则 - 11 -](#_Toc390345928)

[5.2 模型相关 - 12 -](#_Toc390345929)

[5.3 视图相关 - 13 -](#_Toc390345930)

[5.3.1 模版引擎Freemarker使用说明 - 13 -](#_Toc390345931)

[5.3.2 静态资源文件 - 13 -](#_Toc390345932)

[5.3.3 文件上传下载处理 - 13 -](#_Toc390345933)

[5.3.4 HTTP 会话使用要求 - 13 -](#_Toc390345934)

[5.3.5 标准表单开发说明 - 14 -](#_Toc390345935)

[6 测试框架 - 15 -](#_Toc390345936)

[6.1 Spring集成测试框架说明 - 15 -](#_Toc390345937)

[6.1.1 创建测试类 - 15 -](#_Toc390345938)

[6.1.2 使用依赖注入 - 15 -](#_Toc390345939)

[6.1.3 使用注解控制测试方法的事务 - 15 -](#_Toc390345940)

[6.1.4 测试WEB Controller - 16 -](#_Toc390345941)

[6.2 其他测试功能说明 - 17 -](#_Toc390345942)

[6.2.1 测试私有方法 - 17 -](#_Toc390345943)

[6.2.2 使用EasyMock进行测试 - 18 -](#_Toc390345944)

[7 日志处理框架 - 19 -](#_Toc390345945)

[7.1 调试日志记录要求 - 19 -](#_Toc390345946)

[7.2 审计日志记录要求 - 19 -](#_Toc390345947)

[8 异常处理框架 - 21 -](#_Toc390345948)

[8.1 DAO层异常处理 - 21 -](#_Toc390345949)

[8.2 服务层异常处理 - 21 -](#_Toc390345950)

[8.3 WEB层异常处理 - 21 -](#_Toc390345951)

[8.4 全局异常处理要求 - 21 -](#_Toc390345952)

[9 其他特性 - 23 -](#_Toc390345953)

[9.1 第三方库引入要求 - 23 -](#_Toc390345954)

[9.2 其他编码要求 - 23 -](#_Toc390345955)

[10 开发参考资源 - 24 -](#_Toc390345956)

# 开发规范概述

开发中请遵循“约定大于配置”的原则，请尽量遵守规范。

# Spring框架

## Spring配置说明

### 全局Spring配置

全局的Spring配置内容，主要包括Spring容器全局能够访问的实例定义等，存放在src/main/webapp/WEB-INF/spring/\*.xml配置文件里。

### 数据源相关Spring配置

数据源相关的配置内容，主要包括数据源定义，事务管理器，DAO层工厂类定义等，存放在src/main/webapp/WEB-INF/spring/dataAccessContext.xml 配置文件里。DAO层的具体Mapper对象实例会被MyBatis工厂类自动纳入Spring容器中。

### 服务相关Spring配置

服务层的Spring配置内容，主要包括Spring容器全局能够访问的服务实例定义等，存放在src/main/webapp/WEB-INF/spring/serviceContext.xml配置文件里。

### WEB相关Spring配置

WEB相关的配置，主要包括Web的控制器、视图分发器的定义等，内容存放在src/main/webapp/WEB-INF/spring/servletContext.xml 配置文件里。

Web层的具体控制器对象必须基于**@Controller**注解方式自动被纳入Spring容器中。

### SSB相关Spring配置

SSB(系统服务总线)相关的配置，主要包括远程服务导出，服务导出相关的协议端口等，内容存放在src/main/webapp/WEB-INF/ssb/\*.xml 配置文件里。

## Spring注解使用要求

### 将对象纳入Spring容器管理

目前构架中为了方便直观的管理Service，请使用基于XML配置文件的方式将相关对象纳入Spring容器中，不使用**@Service**, 不使用**@Component**，不使用**@Repository。**

请在src/main/webapp/WEB-INF/spring/serviceContext.xml文件中进行配置。

#### @Service注解使用要求

目前构架中不使用**@Service**注解， 使用基于XML配置文件的方式将相关对象纳入Spring容器中。

#### @Component注解使用要求

目前构架中不使用**@Component**注解， 使用基于XML配置文件的方式将相关对象纳入Spring容器中。

#### @Repository注解使用要求

 目前构架中不使用**@Repository**注解， 由MyBatis提供的实现将DAO自动注入Spring容器中。

### 注入依赖对象

使用**@Resource**, 不允许使用**@Autowired**

#### @Resource注解使用要求

在需要对象依赖的地方注入，基于实例变量声明，***变量名即服务名***

|  |
| --- |
| **@Resource****private** IExampleService exampleService |

### Spring Bean生命周期管理注解

**@Scope**, 对应<bean>标签的”scope”属性

**@Lazy**, 对应<bean>标签的”lazy-init”属性

**@PostConstruct**, 对应<bean>标签的” init-method”属性, 可绑定在多个方法上

**@PreDestroy,** 对应<bean>标签的” destroy-method”属性, 可绑定在多个方法上

# 持久层

## 持久层类结构说明

XxxxCustomizedMapper

+method(XxxxAO ao)

+m2(XxxxCriteria ec)

XxxxGeneratedMapper

+insert(XxxxAO ao)

+select(XxxxCriteria ec)

Xxxx

XxxxAO

XxxxCretiea

XxxxGeneratedMapper.xml

XxxxCustomizedMapper.xml

系统框架提供

自动生成

应用开发

首次生成

上图中Xxxx表示实体名。

通常，应用开发时，根据业务设计出相关数据库表结构，然后通过

mybatis-generator插件生成上图中”**自动生成**”和”**首次生成**”部分，这样可以得到最基本的DAO功能，当基本功能不满足时，应用开发手工创建”**应用开发**“部分， 同时根据需要编辑“**首次生成**”部分，这样既保障默认情况有大部分功能可用，同时保留了应用开发的灵活性。

注意：应用开发各层传递时都必须使用“XxxxAO”应用对象。

## 持久层包结构说明

|  |
| --- |
| **Cfg: src/main/java**|--com.epicsaas.app.xxx.dao.gen |--xxxGeneratedMapper<EntityAO, EntityCriteria> |--xxxCustomizedMapper<EntityAO, EntityCriteria>.......|-- com.epicsaas.app.xxx.entity.\*\* |--Xxxx |--XxxxCriteria |-- com.epicsaas.app.xxx.appobject.\*\* |--XxxxAO......|-- com.epicsaas.app.xxx.dao.\*\* |--XxxxGeneratedMapper<XxxxAO, XxxxCriteria> |--XxxxCustomizedMapper<XxxxAO, XxxxCriteria>**Cfg: src/main/resources**|-- com.epicsaas.app.xxx..\*\* |--XxxxGeneratedMapper.xml |--XxxxCustomizedMapper.xml |

其中，xxx表示实体名，\*\*表示根据子应用划分的子包。

## 持久层开发步骤说明

1. 基于相关业务数据库表结构定义，修改应用项目下的

src/main/resources/generatorConfig-{dbtype}.xml配置文件和pom.xml项目管理文件, 以配置DAO层自动代码生成工具。

1. 在应用项目里运行执行**mvn gen:gen**命令， 执行代码生成工具， 以生成“3.1章持久层类结构说明”中的“**首次生成部分**”以及 “**自动生成部分**”的相关代码。
2. 分析应用业务规则，修改应用对象和自定义Mapper文件。根据需要可选的创建对应的XxxxCustomizedMapper.java和XxxxCustomizedMapper.xml文件，并添加相应内容，同时根据实际需要在XxxxAO.java应用对象类中添加相应内容。
3. 分析相关业务规则，根据需要修改应用项目下的

src/main/resources/validation/validation-\*.xml文件，添加基于JSR303的数据验证规则到应用对象上，方便各层对该应用对象进行数据绑定操作时的数据验证。

1. 如果后续有更多业务变更需要改动MAPPER接口，那么只需要反复重复第4，第5两步即可，即可完成DAO层的设计开发目标。

如果后续有更多业务变更需要改动数据库表结构，那么只需要反复重复第1，第2，第3三步即可，即可完成DAO层的设计开发目标。

1. 生成XxxxGeneratorMapper.xml默认为两份，mysql目录中的是mysql备份。编写自定义XxxxCustomizedMapper.xml需要存一个mysql的备份,需要支持其他数据库按照这种格式自己建立目录。用来支持打包时生产多数据库版本的包。打包命令：mysql版 mvn clean install –Pmysql

 Oracle版 mvn clean install -Poracle



## MyBatis使用要求

1. 优先使用嵌套结果方式进行关联查询，特别是数据量大的表。

## MyBatis开发参考

请参考《Mybatis3用户指南.pdf》

# 服务层

## 服务层基本说明

服务层的服务可根据应用开发需要进行分包，每一个服务**必须**包含**一个服务接口类**和**一个或多个的服务实现类**。

服务接口类都须以“**IXxxxService**”方式进行命名， 而服务实现类都须以“**XxxxService**”方式命名服务。

## 服务层开发说明

|  |
| --- |
| **SDK中提供基础接口**|-- com.epicsaas.common.service.base|--IBaseAOService|-- AbstractBaseAOService |

对于普通服务接口定义则须继承“IBaseAOService”接口。服务接口中的所有服务方法的返回值都必须为“com.epicpaas.sdk.core.api.ServiceResult <T>”类型。

对于普通服务实现须继承“AbstractBaseAOService”抽象服务基类 ，服务实现中的非服务接口方法必须定义为private，返回值不作限定。

## 服务事务管理说明

所有事务管理通过@Transactional注解方式在服务**实现类**，或者服务**实现方法上**控制相应业务的事务管理。

## 服务本地导入与调用说明

服务本地导入Spring容器中是通过在src/main/webapp/WEB-INF/serviceContext.xml

配置文件中添加<bean/>服务配置项的方式完成， 而其调用方式则是通过使用@Resource注解将服务实例注入到调用对象中以便完成相应服务的调用。

## 服务远程导出与调用说明

服务远程导出的步骤如下：

1. 将服务通过基于src/main/webapp/WEB-INF/spring/serviceContext.xml配置文件的方式导入到本地Spring容器中。
2. 在src/main/webapp/WEB-INF/ssb/ssbProvider.xml配置文件里，添加如下配置项：

<ssb:service interface="com.xxx.xxx.XxxxService" ref="xxxxService" />

将该本地服务导出成一个远程服务。

而对于远程服务的调用步骤如下：

1. 在调用端的src/main/webapp/WEB-INF/ssbConsumer.xml配置文件里，添加如下配置项：

<ssb:reference id="xxxxService" interface="com.xxx.xxx.XxxxService" />

以便在调用端Spring容器里生成一个远程服务代理实例。

1. 通过@Resource注解将远程服务代理实例注入到调用对象中，以便完成相应的远程服务调用。

# WEB层

## 控制器相关

1. 所有Controller类放在”xxx.xxx.web.controller”包或其子包中。
2. Controller类必须使用 ”**@RequestMapping**” 注解配置URL映射。
3. Controller类必须使用“**@Resource**”注入服务层对象到本地，以便进行业务逻辑调用， 不允许使用new关键字实例化服务层对象。
4. 不允许在Controller 类中使用实例变量来保存共享状态。 如果需要，共享状态可放入SESSSION 或者Application Context。
5. 应用session请使用分布式Session服务。打开ssbConsumer.xml里的配置



 b) 应用Controller注入Session服务

 

1. Controller类中的方法返回值只能是如下两种：
	1. **String** 表示该方法返回某个MVC视图名。
	2. **@ResponseBody Object** 表示该方法返回的是一个JSON对象。
2. 当Controller类中的方法返回某个视图名时，必须使用枚举类

“com.epicsaas.app.{appName}.common.MVCViewName”的条目来指定视图名，禁止直接在代码使用String字符串方式。

### URL路径映射规则

URL路径映射基本规则如下：

业务操作名

应用名

可变参数

**http://ip:port/{app\_name}/{subapp\_name}/{operation}/{param}**

子应用名

具体细节为：

1. 一个Controller类代表一个子业务应用, 其默认类名为:

 “**子应用名+Controller**”， 子应用名可以是英文或者拼音全拼。

1. “子应用名”部分的URL映射定义在Controller类上, 映射值为“/{subapp\_name}”
2. “业务操作名”部分的URL映射定义在Controller类里的业务方法上。
3. “可变参数”部分的URL是可选部分， 也在业务方法上。

下面列举部分通用业务操作名定义

|  |  |  |  |
| --- | --- | --- | --- |
| 业务操作 | @RequestMapping映射值 | HTTP METHOD | 全局URL |
| 添加 | /add | GET/POST | [http://ip:port/{app}/{subapp\_name}/add](http://ip:port/%7Bapp%7D/%7Bsubapp_name%7D/add) |
| 列表 | /list | GET | http://ip:port/{app}/{subapp \_name}/list |
| 查看 | /view/${id} | GET | http://ip:port/{app}/{subapp \_name}/view/{id} |
| 更新 | /update/${id} | GET/POST | http://ip:port/{app}/{subapp \_name}/update/{id} |
| 删除 | /delete/${id} | GET | http://ip:port/{app}/{subapp \_name}/delete/{id} |
| 打开表单 | /showForm | GET | http://ip:port/{app}/{subapp \_name}/showForm |
| 提交表单 | /submitForm | POST | http://ip:port/{app}/{subapp \_name}/submitForm |
| AJAX方式提交表单 | /submitAJAXForm | POST | http://ip:port/{app}/{subapp \_name}/submitAJAXForm |
| 文件操作 | /file/ |  |  |
| 流程引擎 | /eflow/ |  |  |
| 其他业务操作 | /XXXXX |  |  |

## 模型相关

1. DTO 封装处理

使用**@ResponseBody**注解自动将DTO对象或者列表对象转换成JSON格式传入页面。

1. 表单数据绑定

不做强行限定，对于普通简单的业务逻辑，则将表单数据绑定到持久层AO上，对于业务逻辑复杂或者无对应持久层的表单数据，则可以定义对应的FORM BEAN, 放在”xxx.xxx.web.form”包中。

1. 表单数据验证

**客户端**：可使用JQuery Validator或者类似的Javascript组件进行验证。

**服务端**：通过**@Valid**注解配合[JSR303 Bean Validation](http://www.ibm.com/developerworks/cn/java/j-lo-jsr303/index.html)规范进行对应绑定的BEAN属性验证。

验证方式基于XML配置文件方式，而不是基于注解的方式,其验证文件在：src/main/resources/validation/validation-\*.xml里

**注意：**@Valid 注解标注的实体类，后需立即跟对应的BindingResult验证结果对象。 同时，如果需要更复杂的验证规则，请根据业务规则自定义实现满足JSR303规范的注解接口即可。

## 视图相关

### 模版引擎Freemarker使用说明

1. 所有模版文件放在src/main/webapp/WEB-INF/mvcViews/及其子目录中，默认模版后缀名为”**.ftl**”
2. 详细的Freemarker参考文档见使用手册《FreeMarker\_Manual\_zh\_CN.pdf》

### 静态资源文件

所有静态资源放在src/main/webapp/static及其子目录中。存放于此应用才能对该应用下静态资源加速。

### 文件上传下载处理

 默认使用基于Spring MVC提供的文件上传下载的封装组件，如果需要可自行进行扩展。

### HTTP 会话使用要求

禁止使用 HttpSession 接口写HTTP会话数据。

### 标准表单开发说明

当开发一个标准表单时，只需新建一个控制器并继承

“com.epicsaas.common.web.controller.base.AbstractFormController<EntityAO, EntityCriteria>”类, 实现其抽象方法即可。

 当应用开发需要处理自己的业务逻辑时，可根据需要重载下面两个方法即可：





# 测试框架

 对dao，service 的测试可以使用 spring 集成测试框架进行测试，对controller 的测试有一些特殊之处，在后面有专门的内容进行说明。

## Spring集成测试框架说明

### 创建测试类

源码（com.epicsaas.test.sample.SpringTestSample）

创建使用spring 集成测试框架的测试类很简单

创建一个pojo

使用spring 提供的注解对测试类进行标注（

@ContextConfiguration，@RunWith(SpringJUnit4ClassRunner.class)）

例如：



### 使用依赖注入

源码（com.epicsaas.test.sample.SpringTestSample）

使用spring 提供的：@Resource在测试类中注入需要测试的对象，例如dao，service等.



### 使用注解控制测试方法的事务

源码（com.epicsaas.test.sample.SpringTestSample）

1. 使用 @Transactional 控制整个测试方法使用一个事务



1. 使用 @Rollback：在测试方法结束时，事务回滚



1. 为使用 @Transactional 的测试准备测试数据



### 测试WEB Controller

源码（com.epicsaas.test.sample.SpringTestSample ， com.epicsaas.test.sample. TestController）

对于dao，service 的测试，使用上述方法，在测试类中注入待测试对象即可。

对于web controller 的测试，有一些特殊的情况：

1. 使用spring 提供的mock servlet 对象

如果controller 的方法参数使用了http servlet 对象，

例如HttpServletRequest，HttpServletResponse 等，

可使用 org.springframework.mock.web 下的相应对象来替代，例如：



一个controller 的方法使用http servlet 对象



在测试方法中使用http servlet 对象。

1. 使用 ModelAndViewAssert 对ModelAndView 进行校验

ModelAndView 是一个controller 返回的对象，ModelAndViewAssert 对象提供了一些辅助方法对ModelAndView 进行校验，如果controller 返回的结果是一个ModelAndView，那么可以使用ModelAndViewAssert 对其进行校验，例如：



controller 方法返回 ModelAndView



对ModelAndView 进行验证。

## 其他测试功能说明

### 测试私有方法

源码（com. epicsaas.test.sample. SpringTestSample , com. epicsaas.test.sample. TestBean）

如果需要直接调用某个对象的私有方法，可使用 com. epicsaas.framework.util. Reflections



某个类有一个私有方法



调用私有方法

### 使用EasyMock进行测试

源码（com. epicsaas.test.sample. SpringTestSample ,

com. epicsaas.test.sample. TestController, com.epicsaas.test.sample.IService1）

有时，测试环境中，某些对象不能创建或者创建的代价过高，可以使用mock技术对对这些service 进行模拟，例如：



有一个service，不能在测试环境中创建



但某部分测试代码依赖于这个service



使用easymock 创建mock 对象，并进行测试。

# 日志处理框架

## 调试日志记录要求

导入调试日志记录API, 然后使用LoggerFactory工厂类获取日志记录器即可使用。 示例如下：

![C:\Users\Administrator.PC-20120501YHRA\AppData\Roaming\Tencent\Users\249473086\QQ\WinTemp\RichOle\B{{J13[}IYYMJNS[1W`]9VX.jpg]() 

 例如

 LOG.debug("当前的调试信息 [%s]", "替换左边%s输出的内容" );

 LOG.debug("当前的调试信息 [%d]", 1 );

## 审计日志记录要求

审计日志即应用业务事件记录日志，在需要记录业务事件的业务方法上，通常这些方法就是服务层中服务类里的各种业务方法， 在**服务接口类**里的方法上使用**@BusinessEvent**注解记录审计日志。示例如下：



# 异常处理框架

## DAO层异常处理

由于DAO层的具体实现被隐含在Mybatis映射文件中，所以不需要进行任何异常处理，如果发生异常则由Mybatis抛出该异常到服务层，由服务层决定处理还是继续向上抛出。

## 服务层异常处理

1. 服务层检查异常必须使用

”**com.epicsaas.framework.exception.ServiceCheckedException**”类包装。

1. 服务层非检查（运行时）异常必须使用

”**com. epicsaas.framework.exception.ServiceUncheckedException**”类包装。

## WEB层异常处理

1. 对于需要将错误信息展示给终端客户的业务异常，WEB层应该捕获并包装成客户可读的消息。
2. 对于内部一些RuntimeException，则可以不用处理，由Spring全局异常处理机制进行处理。

## 全局异常处理要求

1. 不允许任何空Catch块，防止吃掉异常。
2. 不允许使用System.out.println(), e.printStackTrace()等方式将日志输出到控制台，请使用调试日志记录器。
3. 重新抛出的异常必须保留原来的异常，即throw new NewException("message", e); 而不能写成throw new NewException("message")。
4. 在所有异常被捕获且没有重新抛出的地方必须写日志（见7.1）。
5. Spring全局异常处理机制见src/main/webapp/WEB-INF/servletContext.xml中的<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">定义。

# 其他特性

## 第三方库引入要求

应用开发如需引入第三方库，请注意与SDK中提供jar版本冲突等问题。

## 其他编码要求

1. 重写方法必须使用**@Override**注解，可避免父类方法改变时导致重写函数失效，进而导致一些BUG出现。
2. 不需要关心的warning信息用@SuppressWarnings("unused"),

@SuppressWarnings("unchecked"), @SuppressWarnings("serial") 注解关闭WARNING信息。

1. 禁止将“javax.servlet.\*”，“javax.servlet.http.\*”及其子包中的类绑定到服务层。如：某服务的一个接口方法：getRoleTreeDTO(HttpSession session)， 禁止这样使用。

# 开发参考资源

1. MyBatis Core Framework

<http://www.mybatis.org/core/>

1. MyBatis Generator

<http://mybatis.org/generator/>

1. FreeMarker

<http://freemarker.org/>

1. Spring Framework

<http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/>

1. JSR 303 - Bean Validation 介绍及最佳实践

<http://www.ibm.com/developerworks/cn/java/j-lo-jsr303/index.html>