

2017

## SmartAs框架培训

烽火云创 - 平台组

陈涛

# 目录

CATALOG



框架简介



环境篇



配置篇



开发篇



进阶篇



框架简介

# 第一部分 框架简介

※ 简介

※ 特性

※ 架构图

## SmartAs简介

SmartAs是集成业界流行成熟的技术和良好的用户体验的**快速应用开发框架**



### 解放框架选型

集成业界流行的技术，从纷繁复杂的框架选型中解放出来



### 解放重复劳动力

集成基础功能，例如登录、用户角色管理、权限、上传下载等



### 聚焦业务本身

封装重复功能代码，只需关注业务本身

## SmartAs特性

### 特性

01

#### 轻量

技术选型及架构上避免选择重型的框架和架构

02

#### 前后端分离

前端完成UI相关的动作，服务端只负责数据处理

03

#### Rest风格

标准的restful风格架构

04

#### 约定俗成

规则优先（接口定义，命名等等）；统一代码风格

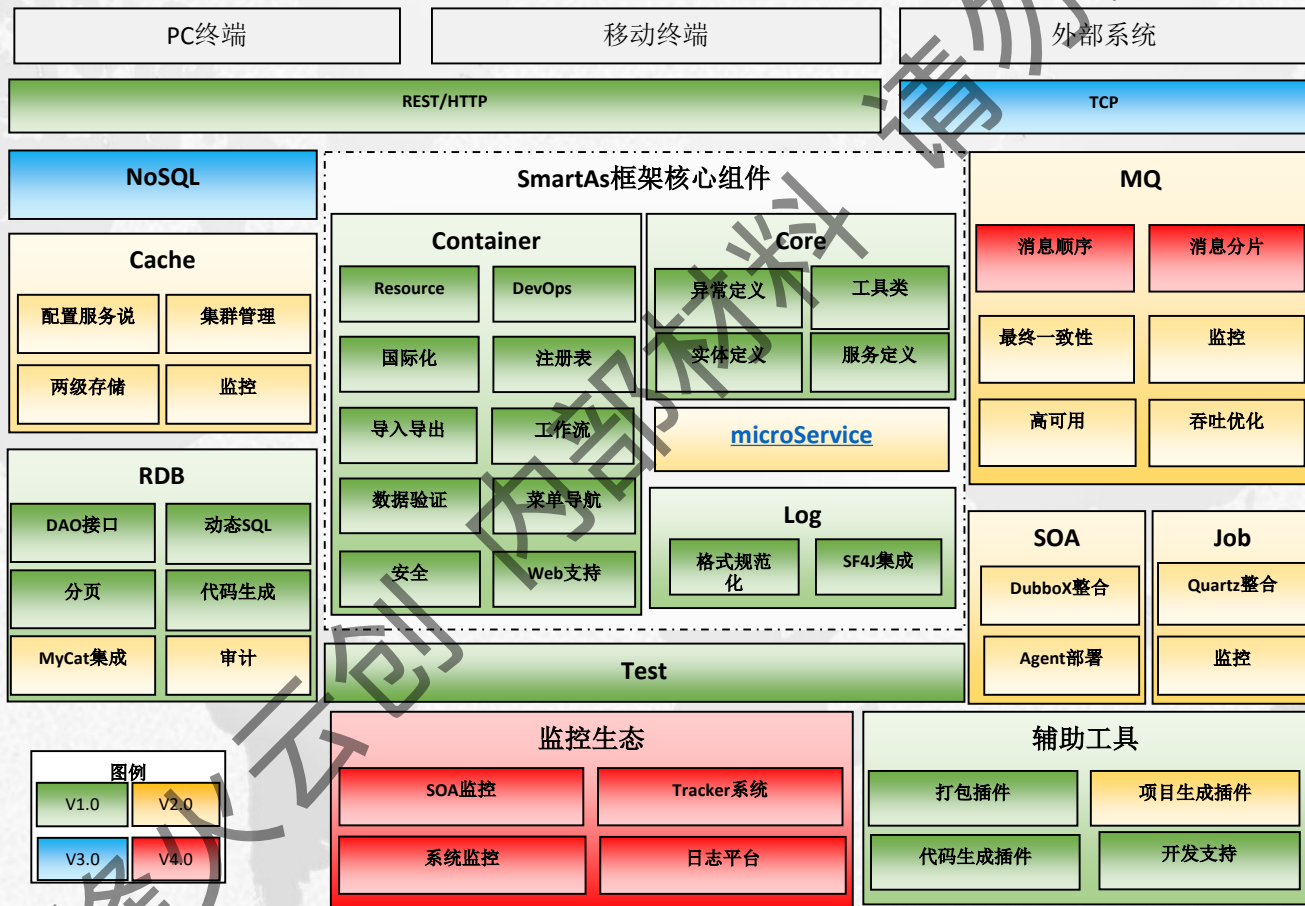
05

#### 高内聚低耦合









服务模块之间功能分离，耦合度很低
























# SmartAs架构图



## 完整Demo工程

 resource	2017/7/24 14:06	文件夹	
 xxx-business	2017/2/15 15:51	文件夹	
 xxx-doc	2017/2/15 15:51	文件夹	
 xxx-lib	2017/2/15 15:51	文件夹	
 xxx-web	2017/2/15 15:51	文件夹	
 .project	2017/2/15 15:51	PROJECT 文件	1 KB
 build.xml	2017/2/15 15:51	XML 文件	6 KB
 package.json	2017/2/15 15:51	JSON 文件	1 KB

## Jar包

 smartas-activiti-2.1.0.jar Executable Jar File 68.9 KB	 smartas-attachment-2.1.0.jar Executable Jar File 13.6 KB	 smartas-cloud-2.1.0.jar Executable Jar File 43.0 KB	 smartas-core-2.1.0.jar Executable Jar File 242 KB
 smartas-demo-2.1.0.jar Executable Jar File 26.1 KB	 smartas-devops-2.1.0.jar Executable Jar File 209 KB	 smartas-download-2.1.0.jar Executable Jar File 6.75 KB	 smartas-env-2.1.0.jar Executable Jar File 3.87 KB
 smartas-excel-2.1.0.jar Executable Jar File 42.4 KB	 smartas-i18n-2.1.0.jar Executable Jar File 9.42 KB	 smartas-job-2.1.0.jar Executable Jar File 22.9 KB	 smartas-lookup-2.1.0.jar Executable Jar File 16.1 KB
 smartas-message-2.1.0.jar Executable Jar File 10.3 KB	 smartas-registry-2.1.0.jar Executable Jar File 11.0 KB	 smartas-security-2.1.0.jar Executable Jar File 105 KB	 smartas-static-resources.jar Executable Jar File 10.9 MB
 smartas-upload-2.1.0.jar Executable Jar File 15.1 KB	 smartas-validator-2.1.0.jar Executable Jar File 61.1 KB	 smartas-web-resources-2.1.0.jar Executable Jar File 63.4 KB	 smartas-web-support-2.1.0.jar Executable Jar File 23.3 KB
 smartas-workflow-2.1.0.jar Executable Jar File 102 KB			



## 前端页面展示

SmartMs

Dashboard 开发 陈浩

首页

我的工作空间

系统管理

用户和权限

用户管理

角色管理

群组管理

组织机构管理

系统设置

定时任务

国际化信息管理

服务器管理

开发

系统管理 / 用户和权限 / 用户管理

用户列表

账户: 名称 组织机构: 组织机构 状态: 状态 查询 重置

提示: 新建用户密码初始值为账号+as

新增用户

	操作	账户	Email	姓名	组织机构	语言	状态
1		root	18@qq.com	超级用户	组织机构->办公室3	简体中文	有效
2		chenjpu	chent5780@fiberhome.com	陈兵	组织机构->办公室3	简体中文	有效
3		test1	sjzhang5782@fiberhome.com	test1	组织机构->党委工作室	简体中文	有效
4		test2	chent5780@fiberhome.com	test2	组织机构->工会工作部	简体中文	有效
5		test3	snli@fiberhome.com	test3	公司组织机构->test	简体中文	无效
6		test4		test4	组织机构->办公室3	简体中文	有效
7		test5		test5	组织机构->办公室3	简体中文	有效
8		test6_update		test6	组织机构->办公室3	英文	有效
9		test7	123@qq.com	test7	组织机构->办公室3	简体中文	有效
10		test8	1231254@qq.com	test8	组织机构->办公室3	繁体中文	有效

共 30 条 1 2 3 > 10 条/页

FiberHome



环境篇

## 第二部分 环境篇

※ Java

※ Tomcat

※ Eclipse

※ Git

## 软件安装



开发语言

Java

Tomcat



Web应用服务器



开发工具

Eclipse

Git



版本控制工具

## tomcat启动文件

D:\software\apache-tomcat-7.0.68\conf\Catalina\localhost

```
<Context reloadable="true" docBase="${project_dir}\emp\emp-web\src\main\webapp" workDir="${project_dir}\emp\emp-web\work" >
  <Loader className="org.apache.catalina.loader.DevLoader" reloadable="true" debug="1" useSystemClassLoaderAsParent="false" />

  <Resource
    name="jdbc/SmartAsDS"
    factory="com.alibaba.druid.pool.DruidDataSourceFactory"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://10.110.200.78:3306/smart_train?useUnicode=true&characterEncoding=UTF-8&
      useFastDateParsing=false&allowMultiQueries"
    username="admin"
    password="admin123"
    maxActive="50"
    maxWait="10000"
    removeabandoned="true"
    removeabandonedtimeout="60"
    logabandoned="false"
    poolPreparedStatements="false"
    maxPoolPreparedStatementPerConnectionSize="20"
    filters="wall,stat,log4j"/>
</Context>
```



配置篇

## 第三部分 配置篇

※ web工程



## web项目结构

- > emp-business [emp master]
- > emp-lib [emp master]
- > emp-web [emp master]

**app.properties** : 全局变量、（数据库）、redis缓存、session等

**log4j.xml** : 日志相关配置

**mybatis-config** : mybatis、数据库相关配置

**root-context.xml** : 配置缓存方式（本地 or redis）

**web-context.xml** : 定义不同开发模式下加载的资源（css、js）

**index.ftl** : 框架页面模版

**web** : 定义的一些 js、css 文件（主题、登录）

- emp-web [emp master]
  - src/main/resources
    - config
      - app.properties
      - log4j.xml
      - mybatis-config.xml
      - root-context.xml
      - web-context.xml
    - ftl.smartas.web
      - index.ftl
    - web
      - config
      - error
      - signin
      - theme

## app.properties

```
1 #env\u914d\u7f6e
2 env.appName=emp
3 env.dbName=smart_train
4 env.tenantId=smartas2
5 env.scope=Smart2
6 env.tablePrefix=emp
7
```

```
3 #\u5de5\u4f5c\u6d41\u90ae\u4ef6\u670d\u52a1\u566
4 workflow.mail.server.host=smtp.fiberhome.com
5 workflow.mail.server.port=25
6 workflow.mail.send.user=xxx
7 workflow.mail.send.password=xxx
8 workflow.mail.send.protocol=smtp
9 workflow.mail.auth=true
10 workflow.mail.timeout=25000
```

```
38 #session\u5171\u4eab\u914d\u7f6e\u5177\u4f53\u8bbe\u7f6e
39 session.useSecureCookie=false
40 session.useHttpOnlyCookie=true
41 session.cookiePath=/
42 #session.domainName=smartas.com
43 #\u9488\u5b9dIP\u5730\u5740\u8bbe\u7f6e\u5177\u4f53\u8bbe\u7f6e
44 session.domainName=
45 session.cookieMaxAge=-1
46
47 #\u914d\u7f6edomain\u6a21\u5f0f\u5339\u914d,\u5c5e\u4e8
48 session.domainNamePattern=
49
50 #\u6709\u6548\u671f\u5177\u4f53\u8bbe\u7f6e
51 session.maxInactiveIntervalInSeconds=3600
52
```

```
1 #redis configuration
2 redis.host=127.0.0.1
3 redis.port=6379
4 redis.pool=true
5 redis.password=
6 redis.timeout=2000
7
```

## root-context.xml

```
<beans>

  <!-- Root Context: defines shared resources visible to all other web components -->
  <context:property-placeholder location="classpath:config/app.properties"
    ignore-unresolvable="true" null-value="" trim-values="true" />

  <!-- JNDI方式的数据源 -->
  <jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/SmartAsDS"/>

  <import resource="classpath*/META-INF/spring/root-context.xml" />

  <!-- 本地缓存配置 -->
  <import resource="classpath:/META-INF/spring/local-cache.xml" />

  <!-- redis缓存配置 -->
  <import resource="classpath:/META-INF/spring/redis-cache.xml" />
  -->

</beans>
```

## web-context.xml

```
<beans profile="dev">
  <bean id="indexHandler" parent="baseDevIndexHandler">
    <property name="url" value="smartas/web/index" />
    <property name="css">
      <list merge="true">
        <value>web/theme/index</value>
        <value>web/theme/theme</value>
      </list>
    </property>
    <property name="commonJs">
      <list merge="true">
        <value>web/common/react/browser.min</value>
      </list>
    </property>
    <property name="uiJs">
      <list merge="true">
        <value>web/theme/index</value>
        <value>web/ui/react-debug</value>
      </list>
    </property>
  </bean>
</beans>
```

## web-context.xml

```
<beans>
  <bean id="loginHandler" class="com.fiberhome.smartas.web.support.handler.ResourceHandler">
    <property name="resource" value="classpath:web/signin/index.html" />
  </bean>

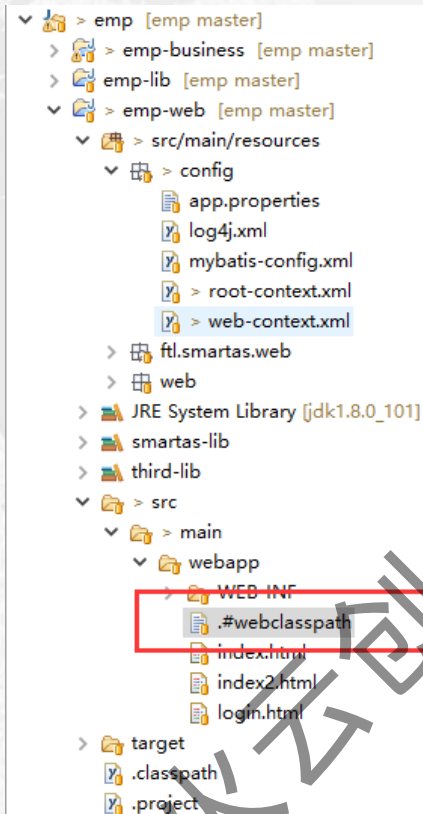
  <!-- 覆盖框架配置 -->
</beans>
```



## index.ftl

```
1<!DOCTYPE html>
2<html lang="en">
3<head>
4<meta charset="utf-8">
5<meta http-equiv="X-UA-Compatible" content="IE=edge">
6<meta name="viewport" content="width=device-width, initial-scale=1">
7<meta name="description" content="${name}">
8<meta name="author" content="chenbing@fiberhome.com">
9<link rel="icon" href="web/images/smartas.png">
10<title>${name}</title>
11<#list css as c<#lt>
12    <link href="${c}.css?v=${version}" rel="stylesheet">
13</#list><#lt>
14<script src="web/common/hack.min.js"></script>
15</head>
16<body>
17    <div id="app" class="theme-${theme}">
18        <div>
19            <div id="main-wrapper">
20                <div id="framework-wrapper">
21                    <nav id="header-wrapper" >
22                        <div class="header">
23                            <a class="brand" id="sidebar-toggle"><i class="fa fa-outdent"></i></a>
24                            <a class="brand logo" href="#!">${name}</a>
25                        </div>
```

## ..#webclasspath



```
1 #项目资源
2 emp-business/target/classes
3 emp-web/target/classes
4
5 #引用的jar包
6 emp-lib/smartas-lib/*.jar
7 emp-lib/third-lib/*.jar
8
```



开发篇

## 第四部分 开发篇

※ 143层架构

※ R-RSM

※ Resource

※ Service

※ Dao

## 1 + 3 层架构



为了更好的降低各层间的耦合度，在三层架构程序设计中，采用面向抽象编程。即上层对下层的调用，是通过接口实现的。而下层对上层的真正服务提供者，是下层接口的实现类。服务标准（接口）是相同的，服务提供者（实现类）可以更换。这就实现了层间的低耦合。

## R-RSM

01

React

作为View层的实现者，完成用户界面操作，通过AJAX完成资源的请求和响应。

02

Rest

作为Resource层的实现者，完成用户请求的接收功能。JAX-RS标准定义资源和方法的映射关系，完成用户请求的转发及对用户的响应。

03

Spring

以整个应用大管家的身份出现。管理整个应用中所有的Bean的生命周期行为，事务，还有拦截器。

04

Mybatis

作为 Dao层的实现者，完成对数据库的增、删、改、查功能。



## Resource层

### Resource层提供rest风格的Web服务

支持权限验证，登录验证

参数传入

请求和响应的数据处理

## Service层

### Service层是SmartAs框架的服务层

在该接口中定义服务相关的方法名。Service层提供事物支持，对外提供原子操作的服务。

框架提供父接口 `BaseService<T>`，服务层接口如果继承该接口通用功能就不必再重写基础方法。只需根据业务需求对服务层接口方法进行扩展。

## Dao层

### 数据库持久层

数据库持久层使用的是开源 Mybatis + 标签定制和扩展方式实现，Dao层只需要提供API，通过mybatis-spring的扫描机制，自动组装实现类。

SmartAs框架提供Dao父接口 BaseDao<T>，自定义的Dao接口继承该接口，则继承了 BaseDao<T>定义的全部方法。

## 添加标题内容

<http://localhost:8080/emp/services/emp/book/bookInfo/{java}>

Book.jsx

BookResource.java

BookService.java

BookServiceImpl.java

BookDao.java

BookDao.xml

## 业务demo

```

v emp-business [emp master]
v src/main/java
  v com.fiberhome.smartas.emp
    v dao
      > BookDao.java
    v service
      v impl
        > BookServiceImpl.java
        > BookService.java
      v ui
        > BookResource.java
        > Book.java
  v src/main/resources
    v com.fiberhome.smartas.emp.dao
      BookDao.xml
    v web.book
      BookAdd.jsx
      BookList.jsx
      BookUpdate.jsx
      BookView.jsx
```



## BookResource.java

```
@Path("emp/book")
@Resource(code = 30001, model = "Smart2", desc = "Book Resource")
public class BookResource extends BaseResource<Book> {
    @Autowired
    private BookService service;

    protected BookService getService() {
        return service;
    }

    @GET
    @Path(value = "/bookInfo/{name}")
    public Book getByName(@PathParam("name") String name) {
        return service.getByName(name);
    }
}
```

## BookService.java

```
4  */
5  public interface BookService extends BaseService<Book> {
6
7      @Transactional
8      Book getByName(String name);
9
10 }
11
```

## BookServiceImpl.java

```
@Service
public class BookServiceImpl extends BaseServiceImpl<Book> implements BookService {

    @Autowired
    private BookDao dao;

    protected BookDao getDao() {
        return dao;
    }

    @Override
    public Book getByName(String name) {

        return dao.getByName(name);
    }

}
```

## BookDao.java

```
15 //
16 @Repository
17 public interface BookDao extends BaseDao<Book> {
18
19     Book getByName(@Param("name") String name);
20
21 }
22
```

## BookDao.xml

```
<mapper namespace="com.fiberhome.smartas.emp.dao.BookDao">

    <resultMap id="BookResultMap" type="com.fiberhome.smartas.emp.Book">

        <id property="id" column="ID" jdbcType="BIGINT" />
        <result property="name" column="NAME" jdbcType="VARCHAR" />
        <result property="author" column="AUTHOR" jdbcType="VARCHAR" />
        <result property="publisher" column="PUBLISHER" jdbcType="VARCHAR" />
        <result property="price" column="PRICE" jdbcType="DOUBLE" />
        <result property="revision" column="REVISION" jdbcType="INTEGER" />
        <result property="createUserId" column="CREATE_USER_ID" jdbcType="NUMERIC" />
        <result property="lastUpdateUserId" column="LAST_UPDATE_USER_ID" jdbcType="NUMERIC" />
        <result property="createDate" column="CREATE_DATE" jdbcType="DATE" />
        <result property="lastUpdateDate" column="LAST_UPDATE_DATE" jdbcType="DATE" />
    </resultMap>

    <select id="getByName" resultMap="BookResultMap">
        SELECT
        T.*
        FROM emp_book_t T
        WHERE T.NAME = #{name}
    </select>
</mapper>
```



## 框架默认提供12种方法

Service<T extends POJO, PK extends Serializable>

- save(List<T>) : List<Serializable>
- update(List<T>) : void
- get(PK) : T
- get(PK[]) : List<T>
- find(PK) : T
- getAll(int, int) : Pageable<T>
- getAll(QueryFilter, int, int) : Pageable<T>
- getAll() : List<T>
- getAll(QueryFilter) : List<T>
- save(T) : T
- update(T) : T
- remove(PK) : void
- remove(PK[]) : void
- remove(T) : void



## 进阶篇

# 第五部分 进阶篇

※ 注解

※ 权限

※ 数据字典

※ 注册表

※ Lookup

※ 动态SQL查询

※ 用户、角色、群组

※ AuthInfo对象

※ 菜单

※ 骨架代码自动生成

### @Lookup

- 01 Lookup模块为用户提供了查找表的前后端实现，可以快速实现系统中的一些类型，分类，业务字典等信息的配置。
- 02 通过提供的UI控件，不需要编写任何代码就可以实现查找表的功能。

## 注解

### @Lookup如何使用？

- 1、在lookup管理中 新增数据类型
- 2、在javaBean对应字段上加上 `@Lookup(type= "PUBLISHER" )`
- 3、在前端页面中，对应属性字段加上 `"Desp"` 后缀
- 4、Lookup组件用法。 `<Lookup groupCode = "PUBLISHER " />`

## 注解

### @Transactional

- 01 Spring提供的事务一致性的注解
- 02 用在service层的类上或者方法上



## 注解

### @Param

@Repository

```
public interface BookDao extends BaseDao<Book> {
```

```
    Book getByName( @Param("name") String name);
```

```
}
```

MyBatis提供，作用是给参数命名,参数命名后就能根据名字得到参数值,正确的将参数传入sql语句中

## 注解

### @Repository

#### @Repository

```
public interface BookDao extends BaseDao<Book> {  
  
    Book getByName( @Param("name") String name);  
  
}
```

Spring提供，将数据访问层（Dao层）标识为 Spring Bean

## 注解

```
runWorker(ThreadPoolExecutor.java:1142)
Worker.run(ThreadPoolExecutor.java:617)

BeanDefinitionException: No qualifying bean of type [com.fiberhome.smartas.emp.dao.BookDao] found for dependency [co
..DefaultListableBeanFactory.raiseNoSuchBeanDefinitionException(DefaultListableBeanFactory.java:1398)
..DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1051)
..DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1018)
tion.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:570)

standardContext listenerStart
listener instance of class org.springframework.web.context.ContextLoaderListener
ancyException Error creating bean with name 'com.fiberhome.smartas.emp.service.impl.BookServiceImpl': Unsatisfied d
tion.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:573)
tion.InjectionMetadata.inject(InjectionMetadata.java:88)
tion.AutowiredAnnotationBeanPostProcessor.postProcessPropertyValues(AutowiredAnnotationBeanPostProcessor.java:350)
..AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1214)
..AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:543)
..AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:482)
..AbstractBeanFactory$1.getObject(AbstractBeanFactory.java:306)
..DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:230)
..AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:302)
..AbstractBeanFactory.getBean(AbstractBeanFactory.java:197)
..DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:775)
actApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:861)
actApplicationContext.refresh(AbstractApplicationContext.java:541)
```

## 注解

@Operation.Login

该注解代表的某个方法（操作）是需要登录后才能访问的。

## AuthInfo对象

### 当前登录用户及授权

#### 1、后台调用

```
AuthInfo subject = AuthInfoUtils.getAuthInfo();
```

#### 2、前台调用

```
const subject = Env.getUser();
```

```
▼ Object ⓘ  
  account: "admin"  
  currentRole: null  
  email: "admin@fiberhome.com"  
  firstname: "admin"  
  id: "1"  
  language: "cn"  
  lastname: "系统管理员"  
  orginfo: ""  
  ► permissions: Array[171]  
  roleMerged: true  
  ► roles: Array[2]  
  ► __proto__: Object
```



## 动态SQL查询

### QueryFilter过滤的查询条件

格式必须为：Q\_field\_T\_OP

Q：表示该参数为查询的参数

Field：查询的字段名称，必须驼峰格式命名

T：代表该参数的类型

OP：执行的操作

## 动态SQL查询

### Q\_field\_T\_OP

T

Z boolean  
B byte  
I int  
L long  
F float  
J double  
S String  
D Date "yyyy-MM-dd" or  
"yyyy-MM-dd HH:mm:ss"  
T Time "HH:mm:ss"

OP

LT <  
LE <=  
GT >  
GE >=  
EQ =  
NE !=  
IN in  
NI not in  
LK like %aa%  
ST like aa%  
ED like %aa  
NL null  
NN not null

## 动态SQL查询

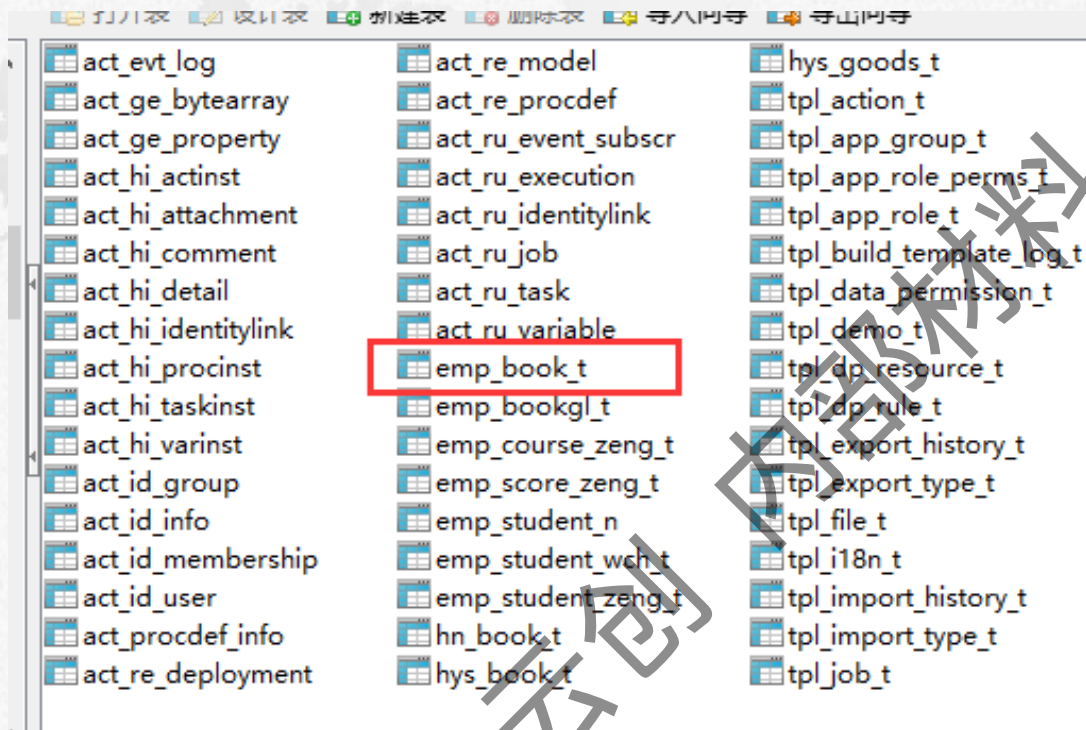
```
const App = React.createClass({
  render: function() {
    var nowDate = new Date().format("yyyy-MM-dd hh:mm:ss")
    var qs={Q_orderState_S_LK:'2', Q_askEndDate_D_GE:nowDate};

    return <Grid rowKey='id'
      QForm={QForm}
      qs={qs}
      columns={columns}
      service={service}
      title='服务询价单报价列表' />
  }
});
```

## 动态SQL查询

```
<select id="selectAll" resultMap="ServiceTradeResultMap">
  SELECT
    T4.*
  FROM(
    SELECT
      T1.*, T2.ENT_NAME, T3.SERVICE_NAME, T3.ADDRESS, T3.CLASS_CODE, T3.UNIT,
      T3.PROFESS_CODE, T3.PRICE
    FROM oic_service_trade_info T1, oic_enterprise_info T2, oic_service_apply T3
    WHERE T1.SERVICE_CODE = T3.SERVICE_CODE AND T1.TENANT_ID = T2.ENT_ID
  ) T4
  where T4.ENTERPRISE_CODE = #{tenantId}
  <filter open="and (" close=")" name="T4" tenant="false"/>
</select>
```

## 骨架代码自动生成



app.properties

```
1 #env\u914d\u7f6e
2 env.appName=emp
3 env.dbName=smart_train
4 env.tenantId=smartas2
5 env.scope=Smart2
6 env.tablePrefix=emp
7
```



## 骨架代码自动生成

SmartAs Web2.0

开发 / Generator

操作	schema	name
1 创建   更多	smart_train	emp_book_t
2 创建   更多	smart_train	emp_bookgl_t
3 创建   更多	smart_train	emp_course_zeng_t
4 创建   更多	smart_train	emp_score_zeng_t
5 创建   更多	smart_train	emp_student_n
6 创建   更多	smart_train	emp_student_wch_t
7 创建   更多	smart_train	emp_student_zeng_t

共 7 条 1 10 条/页

Generator

异常测试

## 骨架代码自动生成

### 参数

**Project :** 项目模块名称

**Package :** 包路径

**URL :** 前端文件路径 ( url )

**Code :** 权限码

**Name :** 文件名称

**生成首页 :** 生成前端文件

New

\* Project: emp-business

\* Package: com.fiberhome.smartas.emp

\* URL: web.book

\* Code: 30002

\* Name: Book

☒ 生成首页

### 生成文件

Book.jsx

Book.java

BookResource.java

BookService.java

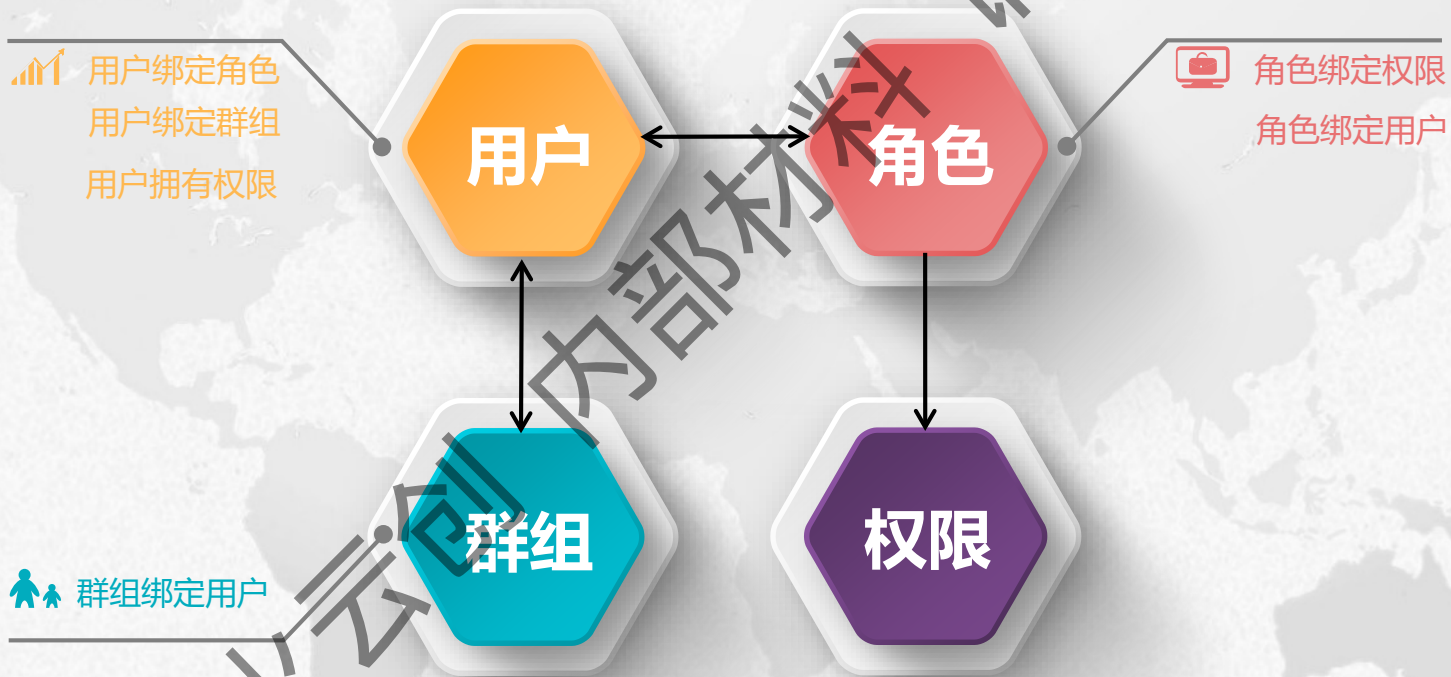
BookServiceImpl.java

BookDao.java

BookDao.xml

# RBAC模型

Role Based Access Control , 基于角色的访问控制



## 权限

### 资源码（5位）+ 操作码组成（3位）

- 01 资源码是用户是否对模块具有权限
- 02 操作码是用户是否对该模块下的某些操作有权限
- 03 模块的代码不能用20000以下，20000以下为系统层使用  
操作码不能使用200以下的，200以下为平台系统权限

```
permissions: Array[168]  
▼ [0 ... 99]
```

```
0: "10510.000"  
1: "12002.105"  
2: "10510.004"  
3: "10510.003"  
4: "10510.002"  
5: "10510.001"  
6: "19004.001"  
7: "10512.007"  
8: "19004.004"  
9: "19004.005"  
10: "99002.004"  
11: "19004.002"  
12: "19004.003"  
13: "10103.004"  
14: "19004.800"  
15: "10103.003"  
16: "10103.002"  
17: "19004.006"  
18: "10103.001"  
19: "10103.000"  
20: "99002.001"  
21: "99002.000"  
22: "99002.003"  
23: "10111.003"
```

## 权限

### 权限验证

- 01 用户登录时，权限数据将会写入到session缓存
- 02 前端发起Rest请求时，security中的SecurityBinder将会检查该用户是否拥有该权限
- 03 如权限满足，则执行后续操作;如未包含该权限，则写入403状态到Response



## 权限

```
@Path("emp/book")
@Resource(code = 30001, model = "Smart2", desc = "Book Resource")
public class BookResource extends BaseResource<Book> {
    @Autowired
    private BookService service;

    protected BookService getService() {
        return service;
    }

    @GET
    @Path(value = "/bookInfo/{name}")
    @Operation(code = Operation.READ, desc = Operation.READ_DESC)
    public Book getByName(@PathParam("name") String name) {
        return service.getName(name);
    }
}
```

### 参数

#### @Resource

code : 资源码

model : 显示模块

desc : 资源描述

#### @Operation

code : 操作码

desc : 操作描述

## 菜单

\* 名称: 用户管理

URL: #!web/smartas/security/user/userList2.jsx

权限码: 10100.000

Class: 样式

Icon: 图标

锚点: userList2

排序号: 1

☒ 是否发布

确定

参数

名称: 菜单显示名称

URL: 前端页面链接

权限码: 菜单是否显示对应的权限码

Class: 自定义菜单样式名

Icon: <http://fontawesome.dashgame.com>

锚点: 控制页面与菜单的对应关系

排序号: 菜单显示排序

感谢您的聆听