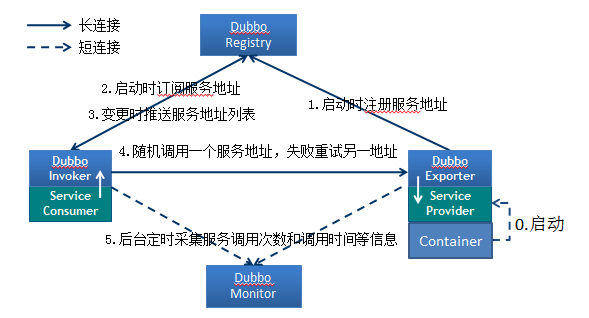
# dubbox服务中间件

## 原理



**节点角色说明：**

 Provider: 暴露服务的服务提供方。

 Consumer: 调用远程服务的服务消费方。

 Registry: 服务注册与发现的注册中心。

 Monitor: 统计服务的调用次调和调用时间的监控中心。

 Container: 服务运行容器。

**调用关系说明：**

 0. 服务容器负责启动，加载，运行服务提供者。

 1. 服务提供者在启动时，向注册中心注册自己提供的服务。

 2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

 3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推

送变更数据给消费者。

 4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，

如果调用失败，再选另一台调用。

 5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计

数据到监控中心。

## RPC（远程过程调用）

是一项广泛用于支持分布式应用程序，RPC 的主要目的是为组件提供一种相互通信的方式，使这些组件之间能够相互发出请求并传递这些请求的结果。

# zookeeper注册中心

## Zookeeper 在Linux系统的安装

## 安装步骤：

第一步：安装 jdk（此步省略，我给大家提供的镜像已经安装好JDK）

第二步：把 zookeeper 的压缩包（资源\配套软件\dubbox\zookeeper-3.4.6.tar.gz）上传到 linux 系统。

Alt+P 进入SFTP ，输入put d:\zookeeper-3.4.6.tar.gz 上传

第三步：解压缩压缩包

|  |
| --- |
| tar -zxvf zookeeper-3.4.6.tar.gz |

第四步：进入 zookeeper-3.4.6 目录，创建 data 文件夹。

|  |
| --- |
| mkdir data |

第五步：进入conf目录 ，把 zoo\_sample.cfg 改名为 zoo.cfg

|  |
| --- |
| cd conf  mv zoo\_sample.cfg zoo.cfg |

第六步：打开zoo.cfg , 修改 data 属性：dataDir=/root/zookeeper-3.4.6/data

## Zookeeper 服务启动

进入bin目录，启动服务输入命令

|  |
| --- |
| ./zkServer.sh start |

输出以下内容表示启动成功



关闭服务输入命令

|  |
| --- |
| ./zkServer.sh stop |

输出以下提示信息



查看状态：

|  |
| --- |
| ./zkServer.sh status |

如果启动状态，提示



如果未启动状态，提示：



## Dubbox本地 JAR包部署与安装（了解）

Dubbox的jar包并没有部署到Maven的中央仓库中，大家在Maven的中央仓库中可以查找到Dubbo的最终版本是2.5.3 , 阿里巴巴解散了Dubbo团队后由当当网继续维护此项目，并改名为 Dubbox ,坐标不变，版本变更了，但是并没有提交到中央仓库。

我们现在需要手动将Dubbox的jar包安装到我的本地仓库中。

先将dubbo-2.8.4.jar包放到d:\setup, 然后输入命令

|  |
| --- |
| mvn install:install-file -Dfile=d:\setup\dubbo-2.8.4.jar -DgroupId=com.alibaba -DartifactId=dubbo -Dversion=2.8.4 -Dpackaging=jar |
|  |

## 工程搭建

**创建Maven工程（WAR）dubboxdemo-service ，在pom.xml中引入依赖spring** dubbo zookeeper的依赖

Service层

import com.alibaba.dubbo.config.annotation.Service;

注意：Service注解与原来不同，需要引入com.alibaba包下的

## 否则报在注册中心找不到对应的服务

另外无法连接到注册中心的错误：请检查IP与端口是否填写正确，检查注册中心是否正常启动

配置四个属性：application计算依赖关系，registry address=“”配置zookeeper的端口2181， dubbo的端口是20880，dubbo:annotation用于扫描@Service注解

## 项目配置dubbox

在src/main/resources下创建applicationContext-service.xml ,内容如下：

|  |
| --- |
| <!-- 使用dubbox发布服务 -->  <!-- 提供方应用信息，用于计算依赖关系 -->  <dubbo:application name=*"mall-sellergoods-service"*/>  <!-- zookeeper的端口2181，集群的时候，使用不同的端口-->  <dubbo:registry address=*"zookeeper://192.168.0.137:2181"*/>  <!-- dubbo的端口是20880，暴露服务,一个服务只能使用一个端口 -->  <dubbo:protocol name=*"dubbo"* port=*"20881"*></dubbo:protocol>  <!-- dubbo:annotation用于扫描@Service注解 -->  <dubbo:annotation package=*"com.cblue.sellergoods.service.impl"* /> |

applicationContext-service.xml

<dubbo:application name=*"dubboxdemo-service"*/>

<dubbo:registry address=*"zookeeper://192.168.25.132:2181"*/>

<dubbo:annotation package="com.cblue.dubboxdemo.service" />

注意：dubbo:annotation用于扫描@Service注解。

applicationContext-web.xml

<!-- 引用dubbo 服务 -->

<dubbo:application name=*"dubboxdemo-web"* />

<dubbo:registry address=*"zookeeper://192.168.25.132:2181"*/>

<dubbo:annotation package=*"cn.itcast.dubboxdemo.controller"* />

# SOA开发模式

SOA是一个面向服务的架构随着项目的进展

多个子系统直接相互交互，相互调用非常凌乱，所以我们就用到了我们的SOA架构，

SOA又叫服务治理就是帮助我们把服务之间调用的乱七八糟的关系给治理起来，然后提供一个统一的标准，以前我们的服务是互相交互，现在是只对数据总线（数据总线是起到调度服务的作用，数据总线里面一个key对于一个value，key指的是服务名，value则是服务的调度方式，还有一点需要说明的是，数据总线只是指路人，服务是不经过数据总线的，）进行交互，这样系统就变得统一起来。

# SpringSecurity（权限管理）

## （1）

我们实现商家登录使用spring Security，我们都是使用spring-security.xml来配置的。

依赖spring-security-core，spring-security-web，spring-security-config，

首先，运营商登录的账户在authentication-manager标签中user标签，定义了登录的账户信息和角色 ，登录页面地址form-login 标签的login-page属性，登录成功地址default-target-url，登录失败跳转地址authentication-failure-url。

如果要页面能够加载框架页，需要设置框架页的策略为SAMEORIGIN sameorigin

为了让我们登录之后，访问默认的页面form-login 标签的always-use-default-target的属性

在登录页面中，action=”/login” method=”post”, 用户名是username，密码是password都是固定的。

我们要用数据库数据来实现登录，首先我们需要创建一个实现了UserDetailsService接口的实现类，并复写loadUserByUsername。在该方法中，首先获取登录的用户名，然后根据用户名去查询用户的密码，最后，把用户名和密码交给SpringSecurity与我们输入的用户名和密码进行匹配。如果匹配成功，跳转到login-page指定的页面。

## （2）

配置文件：（1）pom.xml中添加spring-security-core，spring-security-web，spring-security-config三个依赖，

（2）然后在web.xml中配置项目启动时加载的配置文件，并设置监听器ContextLoaderListener去监听，之后设置filer-mapping

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping> 进行拦截

1. 然后配置spring-security.xml

先配置不被拦截的资源<http pattern=*"/\*.html"* security=*"none"*></http>

如(.html,*/css/\*\*,/img/\*\*,/js/\*\*,/plugins/\*\*)*

*然后配置*页面拦截规则

在<from-login>标签中设置跳转路径

login-page 指定登陆的页面

default-target-url 登陆成功跳转的页面

authentication-failure-url失败跳转

always-use-default-target：指定了是否在身份验证通过后总是跳转到default-target-url属性指定的URL。

为了安全还有配置关掉跨站<csrf disabled=*"true"*/>

工作流程：我们在login.jsp中的from表单设置 id属性，action属性，和method属性

当用户在页面填写name和password之后，点击登陆触发onclick="document:loginform.submit()"事件，

之后去后台spring-security.xml文件中的认证管理器<authentication-manager>中进行验证：

name和login.html中的name="username"；password和name="password"关联

之后根据验证<http>的结果进行页面跳转，在<from-login>标签中设置跳转路径

login-page 指定登陆的页面

default-target-url 登陆成功跳转的页面

authentication-failure-url失败跳转

always-use-default-target：指定了是否在身份验证通过后总是跳转到default-target-url属性指定的URL。

# Cas（单点登录）

# BCrypt（密码加密）

我们使用BCrypt算法实现密码加密。当我们添加用户信息的时候，创建BCrypt密码加密的对象，然后调用加密encode方法，实现加密，放到数据库。最后，还需要在springsecurity的配置文件中指定加密算法。

//密码加密

BCryptPasswordEncoder passwordEncoder = **new** BCryptPasswordEncoder();

String password = passwordEncoder.encode(seller.getPassword());

seller.setPassword(password);

配置

<password-encoder ref=*"bcryptEncoder"*></password-encoder>

# KindEditor（富文本）

这里需要注意的是，商品介绍的内容是通过富文本编辑器获得的。要使用富文本编辑器，在页面中导入富文本编辑器的css和js，然后在页面中添加js的代码，初始化富文本编辑器。然后在页面中通过js的edit.Html()用来获取编辑器的内容。

# Fast DFS（图片存储）

我们的图片是通过专门的文件服务器FastDFS来管理的。FastDFS是可以放置图片和文件的服务器，我们录入的所有的图片信息全部放在这个文件服务器上。图片上传的第一步，我们要在工程中添加文件上传的依赖fastdfs和fileupload。并添加工具类FastDFSClient。然后在商家后台添加配置文件application.properties（服务器地址），fdfs\_client.conf(图片上传路径)。并在springMVC的配置文件中配置图片上传解析器。

第二步、在前端的js页面，我们通过AngularJS实现上传，这里面需要特别设置两个属性：Content-Type header属性，这个属性的默认值是application/json。

我们如果要上传图片，需要把这个属性设置为null，这样就会使用 multipart/form-data值。另外需要

设置transformRequest: angular.identity，这个属性主要是上传对象序列化。

当点击上传按钮的时候，我们在后端通过FastDFS的客户端上传图片，并获得上传地址。最后在添加商品描述对象的时候，把图片地址插入到tb\_goods\_desc表中的item\_images字段中。

# Freemarker（网页静态化）

1. Freemarker是一种用Java编写的模板引擎，可以根据模板以各种形式向前台输出后台数据，html文本，电子邮件，配置文件，源代码等等。它支持的数据类型有数字，日期，字符串，布尔值几种数据类型。

2. 是一种网页展示技术，简单的来说就是网页模板和数据类型结合体，生成的就是静态化页面。

3. 他的好处就是能把网页设计人员和编程人员的工作彻底分离，提高效率，使用JSP 开发过程中在页面中大量的存在业务逻辑的代码，使得页面内容凌乱，不利益后期维护。

4. 缺点是可能看到过期的数据，就是没有及时更新模板，每个变量都需要进行非空判断，太过繁琐，不判断就会抛异常

注意：freemarker与web应用程序无关，他是web框架里面的一个组件，FreeMarker最初的设计，是被用来在MVC模式的Web开发框架中生成HTML页面的，没有被绑定到Servlet或HTML或任意Web相关的东西上，它也可以用于非Web应用环境中。

Freemarker需要用到map，而且他的key必须是String类型

两大功能：1.作为模型引擎，生成静态化页面2.替代jsp作为前台视图层使用的

模板文件中四种元素

1. 文本，直接输出的部分  
     2、注释，即<#--...-->格式不会输出  
     3、插值（Interpolation）：即${..}部分,将使用数据模型中的部分替代输出  
     4、FTL指令：FreeMarker指令，和HTML标记类似，名字前加#予以区分，不会输出。

## assign指令

此指令用于在页面上定义一个变量 （1）定义简单类型：

<#assign linkman="周先生">

联系人：${linkman}

（2）定义对象类型：

<#assign info={"mobile":"13301231212",'address':'北京市昌平区王府街'} >

电话：${info.mobile} 地址：${info.address}

## include指令

此指令用于模板文件的嵌套

if指令

list指令

内建函数

获取集合大小

转换JSON字符串为对象

空值处理运算符

运算符

# Redis

# Solr（搜索）

# 阿里大于

# Activem（消息中间件）

# 事务（声明式事务）

# Linux