## 自我介绍

您好：

我叫王丽媛，来自河南漯河。从15年开始从事Java开发，目前在神州飞腾工作，公司的主要是做IT项目外包。最近一年多的时间里被外派万宠商贸有限公司进行项目的开发。然后最近做的项目是万宠牧场的社区网站。

## 项目描叙

**项目描述：**万宠牧场是集宠物论坛、宠物百科、宠物领养为一体的社区网站。该总体上分为了首页、论坛、领养、百科等多个栏目。实现了社区美图分享，站内信息搜、论坛发帖、宠物领养等多种功能。为用户提供了专家答疑、社区分享、资讯获取、养宠工具等服务。

项目在服务器的整体结构使用分布式+集群形式实现。在软件架构上，使用了SOA架构，实现了服务层和控制层的分类。后端框架采用Spring +SpringMVC+mybatis +Dubbox 。前端采用angularJS + Bootstrap。使用了SpringSecurity控制用户的登录和权限，防止sql注入等系统安全问题。

社区美图分享是宠物社区模块的主要构成。用户可以在社区中展示自己和宠物的生活图片，也可以上传社区中与他用户分享。

#### 美图分享

用户在社区页面点击我要分享按钮跳转当分享页面，用户在分享页面添加本信息（图片标题、关键词、图片分类），图片描述信息。并上传美图

#### 图片上传

用户上传的图片是通过专门的文件服务器FastDFS来管理的。FastDFS是专门放置图片和文件的服务器，用户上传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。然后在社区后台添加配置文件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。

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。最后在在用户提交的描述对象的时候，把图片地址插入到图片描述表中保存到我们的数据库。

用户在提交分享之后，分享的内容会在记录在个人中心中，用户可以在自己的主页中查看。如果用户想在社区中查看自己上传的图片可以使用搜索，通过关键词查询自己的图贴和他人的分享。

这里的搜索功能我们是同过solr。Solr是一个实现搜索的开源服务。它本身是一个war包，可以放到tomcat中。

首先配置中文分词器IK Analyzer，添加分词器的jar包，然后把扩展字典、停用字典、分词器的配置IKAnalyzer.cfg.xml复制到WEB—INF/class目录下。最后，修改 Solrhome下的 schema.xml 文件，配置一个 FieldType，使用 IKAnalyzer。

然后我们根据搜索的业务需要，指定了业务域，比如标题，图片。业务域，常用属性有域的名字（索引库的字段名），域的类型type，是否索引index，是否储存store，是否必须require，是否多值。

定义域分为一般域<field name="item\_goodsid" type="long" indexed="true" stored="true"/>，

复制域（通过复制域可以实现搜索多个域中的关键字）

动态域（把类似域名的域，设置为一个独立的域）

#### Solr实现数据导入

使用solr的索引库目的是为了减轻数据库的压力。这是需要实时的将数据库数据同步导入所引库中。在导入数据时首先，去指定pojo类中属性和索引库域的对应关系。然后查询数据库中所有需要导入的数据。最终通过solrTemplate对象的saveBeans（）把数据导入到索引库中。

#### 更新索引库

更新索引库主要在用户分享的时候，把图片信息提交到索引库。当删除图片分享的时候，从索引库删除图片信息。

当分享的时候，就是修改照片的状态audit\_status字段。然后根据照片表，查询对应的图片信息导入到索引库中。

当删除数据的时候，根据图片的id删除这个索引记录。

#### 搜索实现的原理

网络爬虫（网络蜘蛛）去各个网站上抓取网页，然后保存到服务器上，通过分词器,分析网页中的出现最多的关键词，按照关键词出现的次数进行排名。当我们搜索，实际上就是关键词出现的次数最多对应的网页的排名。

#### 实现搜索结果高亮显示功能

在搜索时我们给搜索的关键词设置了高亮显示。要实现后端实现高亮显示，首先创建高亮查询对象HighLightQuery，然后创建高亮选项对象HighlightOptions，高亮选项对象中设置了高亮”域和高亮前后缀的html代码，并把高亮选项设置给高亮查询对象。然后根据关键字进行查询，返回一个HighlightPage对象。对这个对象进行遍历，获取每一个对象，如果有高亮选项，并且有高亮选项片段，给标题中的关键字设置高亮显示，将查询的内容放入map集合。

最后使用map.putAll()将高亮显示的结果集合添加到原集合中，返回到页面中进行循环展示。这时候发现高亮显示的html代码原样输出，这是因为angularJS为了防止html攻击采取的安全机制，不允许自己插入不信任的HTML。要想将搜索的结果高亮的显示在页面上，首先在base.js定义一个过滤器trustHtml，使用$sce服务的trustAsHtml方法来实现字符串到html代码的转换，静态页面中使用ng-bind-html指令来显示高亮选项，通过标题加竖线加过滤器名，完成标题中的html的转换。

#### 多关键字搜索

Sorl不支持包含空格的多关键字查询，我们需要做的就是去掉其中的空格。使用String.replace()

#### 实现分页

用户在搜索会搜到多条内容为了界面的美观和用户浏览方便我们数据分页显示

~~从本质上来说，实现分页就是实现select \* from tablename limit 开始位置，每页显示的条数。~~

首先在前端定义开始位置和每页显示的条数的变量

设置分页显示的页码。当我们页码大于5条的时候显示部分页码。当前页小于5页，就显示1-5页。当前页+2大于总页数，就显示最后5页。页面在中间位置，前面显示两页，后面显示两页。

当我们点击上一页，下一页，或某一具体页的时候，把开始位置和每页显示的条数提交给后端。

在后端，我们获得提交的开始位置和每页显示的条数，使用sorl的query对象，设置开始位置和每页显示的条数。最后，把执行的结果放到map。跳转到前端进行展示。

显示省略号，就是判断如果在显示前5页的时候，后面显示省略号。当前页+2大于总页数，前面显示省略号。

页面样式不可用，在第一页的时候，上一页设置为不可用。在最后一页的时候，下一页设置不可用。

#### **发帖管理模块**

用户除了可以在社区分享美图还可以在论坛发帖其他用户分享资讯。用户在论坛中选择发表新帖就可以创建新帖。这里要说的是用户发帖所有除标题以外的内容都是通过富文本编辑器获得的。因为在论坛的主页面上不显示帖子的全部内容，只显示文章的标题，用户需要点击标题连接才能查看完整的贴子内容。另外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。用户在写完贴子后需要先提交审核，就是修改发帖表中的状态state字段，如果状态值为1，代表通过审核。如果状态值为2，代表驳回。审核和过后用户的帖子会被放入缓存和solr的索引库中。用户提交发表后就可以将帖子发表到论坛上。

#### 领养平台

用户在网站的领养平台上可以通过提交申请领养自己中意的宠物。在领养的主页面 上会有可以被领养的宠物的基本信息，用户点击宠物图片进入宠物详情介绍的页面。宠物的展示我们使用的是网页静态化FreeMarker技术。根据网页静态化FreeMarker原理将需要的数据填充到模板中。在后台宠物信息提交审核的时候，生成静态页。这个静态页生成在nginx的html目录之下，就可以直接访问。最后，把搜索的地址，设置为这个访问地址就可以了。

模板主要有四个元素：文本，注释（#），插值（$），FTL指令

创建一个 Configuration 对象,设置模板文件所在的路径。设置模板的字符集。一般就是 utf-8.

创建一个模板对象,创建一个模板使用的数据集,一般是 Map。创建一个 Writer 对象，一般创建一 FileWriter 对象，指定生成的文件名。调用模板对象的 process 方法输出文件。

关闭流

### 工厂模式

工厂模式是我们最常用的实例化对象模式了，是用工厂方法代替new操作的一种模式。著名的Jive论坛 ,就大量使用了工厂模式，工厂模式在Java程序系统可以说是随处可见。因为工厂模式就相当于创建实例对象的new，我们经常要根据类Class生成实例对象，如A a=new A() 工厂模式也是用来创建实例对象的，所以以后new时就要多个心眼，是否可以考虑使用工厂模式，虽然这样做，可能多做一些工作，但会给你系统带来更大的可扩展性和尽量少的修改量。

#### 使用SpringDataSorl实现简单查询

首先创建一个搜索对象Query，然后创建一个条件对象Criteria，设定查询的域以及查询的关键字。并添加查询对象中。最终调用sorlTemplate的queryForPage（）执行查询。

#### 规格选择

首先定义一个对象，当点击规格按钮的时候，保存我们选择的规格。

当前的规格是我们选中的规格，就给这个按钮添加选中样式。

#### 页面根据规格显示选择的SKU信息

后台根据goodsId查询需要的SKU信息，在页面中生成对应的JS对象。

页面中默认显示第一个SKU对象的信息。

根据选择的SKU的规格对象，与所有的SKU对象进行比较，获得选中规格的SKU对象，并显示这个对象的标题和价格。

#### 购买数量加减操作

在将商品添加到购物车前，在详情页面加入购物车的左侧可以对加入购物车数量进行加减操作。首先定义一个变量，这个变量是商品数量的初始值。我们定义一个函数，对这个变量进行加减。这里需要设定一个下限，变量值不能小于1。

#### 系统模块对接

当商品审核的时候，生成静态页。这个静态页生成在nginx的html目录之下，就可以直接访问。最后，把搜索的地址，设置为这个访问地址就可以了。

# 1 bean配置

bean配置有三种方法：

基于xml配置Bean

使用注解定义Bean

基于java类提供Bean定义信息

## 1.1 基于xml配置Bean

 对于基于XML的配置，Spring 2.0以后使用Schema的格式，使得不同类型的配置拥有了自己的命名空间，是配置文件更具扩展性。

①默认命名空间：它没有空间名，用于Spring Bean的定义；

②xsi命名空间：这个命名空间用于为每个文档中命名空间指定相应的Schema样式文件，是标准组织定义的标准命名空间；

③aop命名空间：这个命名空间是Spring配置AOP的命名空间，是用户自定义的命名空间。

Bean的基本配置

一般情况下，Spring IOC容器中的一个Bean即对应配置文件中的一个<bean>,这种镜像对应关系应该容易理解。其中id为这个Bean的名称，通过容器的getBean("foo")即可获取对应的Bean，在容器中起到定位查找的作用，是外部程序和Spring IOC容器进行交互的桥梁。class属性指定了Bean对应的实现类

# 数据库的几种索引

(1)按照索引列值的唯一性，索引可分为唯一索引和非唯一索引

①非唯一索引：B树索引

create index 索引名 on 表名（列名） tablespace 表空间名;

②唯一索引：建立主键或者唯一约束时会自动在对应的列上建立唯一索引

(2)索引列的个数：单列索引和复合索引

(3)按照索引列的物理组织方式

①B树索引

create index 索引名 on 表名（列名） tablespace 表空间名;

②位图索引

create bitmap index 索引名 on 表名（列名） tablespace 表空间名;

③反向键索引

create index 索引名 on 表名（列名） reverse tablespace 表空间名;

④函数索引

create index 索引名 on 表名（函数名(列名)） tablespace 表空间名;

删除索引

drop index 索引名

重建索引

alter index 索引名 rebuild

-

Dubbox原理

Dubbox实现了分布式服务的提供者和消费者。通过zookeeper的注册中心，

实现提供者和消费者的映射。消费者调用提供者提供的服务。

### AngularJS在项目中实现的原理：

我们在前端使用AngularJS框架，主要使用了AngularJS的MVC模式和模块化的思想，实现了前端JS代码分层、分模块，形成了和后端一样的，服务层和控制层。然后使用依赖注入把Controller层需要的服务注入进去。最后，使用双向数据绑定来异步加载数据。

### FTL指令

assign指令 定义一个变量

include指令 模板文件的嵌套

if指令

list指令

### 内建函数

内建函数语法格式： 变量+?+函数名称

size函数 数组长度

Eval json字符串转换为对象

Date，time 日期格式化

C 数字转换为字符串

variable??,如果该变量存在,返回true,否则返回false

缺失变量默认值:“!”

### BootStrap：

Bootstrap是一个支持响应式的Css框架它提供了很多组件，如导航条，面板，菜单，form表单，还有栅格，而且他们这些都是支持响应式的，可以在各种设备上进行完美的展现。这里面我感觉最有价值的就是bootstrap提供的栅格系统，这个栅格系统将整个页面分为12列，而且可以根据屏幕的宽窄进行自动调节，这也是响应式的关键所在。在使用栅格时候注意最外层样式是Container,里面是row,row里面包的是列，列里面可以用来填充各种各样的组件。

我在项目中使用bootstrap完成的情况大概是这个样子，首先我使用了bootstrap的导航条，并将其固定在顶部，使其在拖拉滚动条的时候不至于看不到之后在导航条的下面采用了bootstrap的栅格系统将其分为左右两部分，左边使用bootstrap的Treeview组件，菜单展现出来，当点击treeview上的菜单节点时结合一个第三方的tab组件，将需要展示的内容放到tab页内，并进行上下切分，上面使用了bootstrap的form组件，下面使用了它的响应式表格以及分页组件，在进行增加修改时，使用了第三方的bootbox弹出框。

ActiveMQ“

这个简单，ActiveMQ 是一个 MOM，具体来说是一个实现了 JMS 规范的系统间远程通信的消息代理。好。MOM 就是面向消息中间件（Message-oriented middleware），是用于以分布式应用或系统中的异步、松耦合、可靠、可扩展和安全通信的一类软件。MOM 的总体思想是它作为消息发送器和消息接收器之间的消息中介,这种中介提供了一个全新水平的松耦合。

Active

* 获取连接工厂
* 使用连接工厂创建连接
* 启动连接
* 从连接创建会话
* 获取 Destination
* 创建 Producer，或
  + 创建 Producer
  + 创建 message
* 创建 Consumer，或发送或接收message发送或接收 message
  + 创建 Consumer
  + 注册消息监听器（可选）
* 发送或接收 message
* 关闭资源（connection, session, producer, consumer 等)

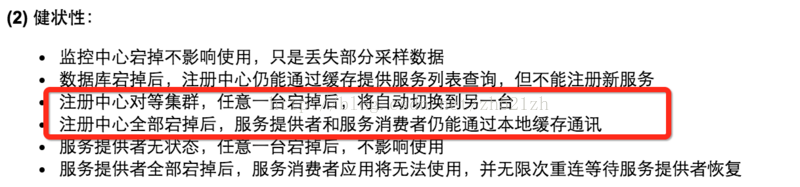
# 面试题：Dubbo中zookeeper做注册中心，如果注册中心集群全都挂掉，发布者和订阅者之间还能通信么？

可以的，消费者在启动时，消费者会从zk拉取注册的生产者的地址接口等数据，缓存在本地。

每次调用时，按照本地存储的地址进行调用

可以的，消费者本地有一个生产者的列表，他会按照列表继续工作，倒是无法从注册中心去同步最新的服务列表，短期的注册中心挂掉是不要紧的，但一定要尽快修复

挂掉是不要紧的，但前提是你没有增加新的服务，如果你要调用新的服务，则是不能办到的



Redis的雪崩和击穿

缓存穿透是指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，造成缓存穿透。

* 解决办法：

1. 对所有可能查询的参数以hash形式存储，在控制层先进行校验，不符合则丢弃。还有最常见的则是采用布隆过滤器，将所有可能存在的数据哈希到一个足够大的bitmap中，一个一定不存在的数据会被这个bitmap拦截掉，从而避免了对底层存储系统的查询压力。
2. 也可以采用一个更为简单粗暴的方法，如果一个查询返回的数据为空（不管是数 据不存在，还是系统故障），我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

### 缓存雪崩

* 认识

如果缓存集中在一段时间内失效，发生大量的缓存穿透，所有的查询都落在数据库上，造成了缓存雪崩。

这个没有完美解决办法，但可以分析用户行为，尽量让失效时间点均匀分布。大多数系统设计者考虑用加锁或者队列的方式保证缓存的单线程（进程）写，从而避免失效时大量的并发请求落到底层存储系统上。

* 解决方法

1. 在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。
2. 可以通过缓存reload机制，预先去更新缓存，再即将发生大并发访问前手动触发加载缓存
3. 不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀
4. 做二级缓存，或者双缓存策略。A1为原始缓存，A2为拷贝缓存，A1失效时，可以访问A2，A1缓存失效时间设置为短期，A2设置为长期。

#### Hibernate和mybatis的区别

**hibernate:**是一个标准的ORM框架（对象关系映射）。入门门槛较高，不需要写sql，sql语句自动生成，对sql语句进行优化、修改比较困难。

**mybatis：**专注sql本身，需要程序员自己编写sql语句，sql修改、优化比较方便。mybatis是一个不完全的ORM框架，虽然程序员自己写sql，mybatis也可以实现映射（输入映射，输出映射）

## 1.1 Hibernate 简介

Hibernate对数据库结构提供了较为完整的封装，Hibernate的O/R Mapping实现了POJO 和数据库表之间的映射，以及SQL 的自动生成和执行。程序员往往只需定义好了POJO 到数据库表的映射关系，即可通过Hibernate 提供的方法完成持久层操作。程序员甚至不需要对SQL 的熟练掌握， Hibernate/OJB 会根据制定的存储逻辑，自动生成对应的SQL 并调用JDBC 接口加以执行。

## 1.2 MyBatis简介

iBATIS 的着力点，则在于POJO 与SQL之间的映射关系。然后通过映射配置文件，将SQL所需的参数，以及返回的结果字段映射到指定POJO。 相对Hibernate“O/R”而言，iBATIS 是一种“Sql Mapping”的ORM实现。

## Hibernate的调优方案

1. 制定合理的缓存策略；
2. 尽量使用延迟加载特性；
3. 采用合理的Session管理机制；
4. 使用批量抓取，设定合理的批处理参数（batch\_size）;
5. 进行合理的O/R映射设计

## Mybatis调优方案

MyBatis在Session方面和Hibernate的Session生命周期是一致的，同样需要合理的Session管理机制。MyBatis同样具有二级缓存机制。 MyBatis可以进行详细的SQL优化设计。

## SQL优化方面

Hibernate的查询会将表中的所有字段查询出来，这一点会有性能消耗。Hibernate也可以自己写SQL来指定需要查询的字段，但这样就破坏了Hibernate开发的简洁性。而Mybatis的SQL是手动编写的，所以可以按需求指定查询的字段。

Hibernate HQL语句的调优需要将SQL打印出来，而Hibernate的SQL被很多人嫌弃因为太丑了。MyBatis的SQL是自己手动写的所以调整方便。但Hibernate具有自己的日志统计。Mybatis本身不带日志统计，使用Log4j进行日志记录。

#### hibernate的缓存机制

Hibernate缓存包括两大类：Hibernate一级缓存和Hibernate二级缓存Hibernate一级缓存又称为“Session的缓存”，它是内置的，不能被卸载（不能被卸载的意思就是这种缓存不具有可选性，必须有的功能，不可以取消session缓存）。由于Session对象的生命周期通常对应一个数据库事务或者一个应用事务，因此它的缓存是事务范围的缓存。第一级缓存是必需的，不允许而且事实上也无法卸除。在第一级缓存中，持久化类的每个实例都具有唯一的OID。 Hibernate二级缓存又称为“SessionFactory的缓存”，由于SessionFactory对象的生命周期和应用程序的整个过程对应，因此Hibernate二级缓存是进程范围或者集群范围的缓存，有可能出现并发问题，因此需要采用适当的并发访问策略，该策略为被缓存的数据提供了事务隔离级别。第二级缓存是可选的，是一个可配置的插件，在默认情况下，SessionFactory不会启用这个插件。

#### cookie和session的区别

　　1 session保存在服务器，客户端不知道其中的信息；cookie保存在客户端，服务端可以知道其中的信息

　　2 session中保存的是对象，cookie中保存的是字符串

　　3 session不能区分路径，同一个用户在访问一个网站期间，所有的session在任何一个地方都可以访问到；而cookie中如果设置了路径参数，那么同一个网站中不同路径下的cookie互相是访问不道德

### session的原理：

　　http是无状态的协议，客户每次读取web页面时，服务器都打开新的会话，而且服务器也不会自动维护客户的上下文信息。session就是一种保存上下文信息的机制，她是针对每一个用户的，session的内容在服务器端，通过sessionId来区分不同的客户，session是以cookie或url重写为基础的，默认用cookie来实现，系统会创造一个JSESSIONID的输出cookie，我们成为session cookie，以区分persistent coookies，注意session cookie是存储于浏览器内存中的，并不是写到硬盘上的；我们通常是看不见JSESSIONID的，但是当我们禁用浏览器的cookie后，web服务器会采用url重写的方式传递sessionid，我们就可以在浏览器看到sessionid=HJHADKSFHKAJSHFJ之类的字符串；session cookie针对某一次会话而言，会话结束session cookie也就消失了

### session与cookie的联系：

　　session是需要借助cookie才能正常工作的，如果客户端完全禁止cookie，session将失效

### session和cookie的应用：

如何利用实现自动登录  
　　当用户在某个网站注册后，就会收到一个惟一用户ID的cookie。客户后来重新连接时，这个用户ID会自动返回，服务器对它进行检查，确定它是否为注册用户且选择了自动登录，从而使用户无需给出明确的用户名和密码，就可以访问服务器上的资源

会话跟踪：

通常session cookie是不能跨窗口使用的，当你新开了一个浏览器窗口进入相同页面时，系统会赋予你一个新的sessionid，这样我们信息共享的目的就达不到了，此时我们可以先把sessionid保存在persistent cookie中，然后在新窗口中读出来，就可以得到上一个窗口SessionID了，这样通过session cookie和persistent cookie的结合我们就实现了跨窗口的session tracking

**1、内联接**（典型的联接运算，使用像 =  或 <> 之类的比较运算符）。包括相等联接和自然联接。       
内联接使用比较运算符根据每个表共有的列的值匹配两个表中的行。例如，检索 students和courses表中学生标识号相同的所有行。     
**2、外联接。**外联接可以是左向外联接、右向外联接或完整外部联接。       
在 FROM子句中指定外联接时，可以由下列几组关键字中的一组指定：       
1）LEFT  JOIN或LEFT OUTER JOIN       
左向外联接的结果集包括  LEFT OUTER子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值。         
2）RIGHT  JOIN 或 RIGHT  OUTER  JOIN       
右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。         
3）FULL  JOIN 或 FULL OUTER JOIN  
完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。     
  
**3、交叉联接**交叉联接返回左表中的所有行，左表中的每一行与右表中的所有行组合。交叉联接也称作笛卡尔积。      
FROM 子句中的表或视图可通过内联接或完整外部联接按任意顺序指定；但是，用左或右向外联接指定表或视图时，表或视图的顺序很重要。有关使用左或右向外联接排列表的更多信息，请参见使用外联接。

#### Spring的AOP面向切面编程

1、AOP概念：

面向切面编程，指扩展功能不修改源代码，将功能代码从业务逻辑代码中分离出来。

       主要功能：日志记录，性能统计，安全控制，事务处理，异常处理等等。

       主要意图：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑的方法中，进而改变这些行为的时候不影响业务逻辑的代码。

2、AOP特点：

采用横向抽取机制，取代了传统纵向继承体系重复性代码。

3、AOP底层实现：

AOP底层使用动态代理实现。包括两种方式：

      （1）使用JDK动态代理实现。

      （2）使用cglib来实现

#### Servlet的生命周期

Servlet 加载—>实例化—>服务—>销毁。

**init（）：**

在Servlet的生命周期中，仅执行一次init()方法。它是在服务器装入Servlet时执行的，负责初始化Servlet对象。可以配置服务器，以在启动服务器或客户机首次访问Servlet时装入Servlet。无论有多少客户机访问Servlet，都不会重复执行init（）。

**service（）：**

它是Servlet的核心，负责响应客户的请求。每当一个客户请求一个HttpServlet对象，该对象的Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对象和一个“响应”（ServletResponse）对象作为参数。在HttpServlet中已存在Service()方法。默认的服务功能是调用与HTTP请求的方法相应的do功能。

**destroy（）：**

仅执行一次，在服务器端停止且卸载Servlet时执行该方法。当Servlet对象退出生命周期时，负责释放占用的资源。一个Servlet在运行service()方法时可能会产生其他的线程，因此需要确认在调用destroy()方法时，这些线程已经终止或完成。

#### CAS

【1】开源的企业级单点登录解决方案。

【2】CAS Server 为需要独立部署的 Web 应用。

【3】CAS Client 支持非常多的客户端(这里指单点登录系统中的各个 Web 应用)，包括 Java, .Net, PHP, Perl, Apache, uPortal, Ruby 等。

SSO单点登录访问流程主要有以下步骤：

1. 访问服务：SSO客户端发送请求访问应用系统提供的服务资源。

2. 定向认证：SSO客户端会重定向用户请求到SSO服务器。

3. 用户认证：用户身份认证。

4. 发放票据：SSO服务器会产生一个随机的Service Ticket。

5. 验证票据：SSO服务器验证票据Service Ticket的合法性，验证通过后，允许客户端访问服务。

6. 传输用户信息：SSO服务器验证票据通过后，传输用户认证结果信息给客户端。

#### 程序访问百度是重定向还是请求转发

答：重定向。转发只能在程序内部进行页面的跳转，重定向重新获得地址，显示在浏览器的地址栏。