SOA 分布式个人博客系统

（开发手册）

项目名称：SOA 分布式个人博客系统

作　　者：宸　凯

日　　期：2017 年 10 月 24 日

地　　址：https://github.com/zkaif/MyBlog

内容摘要

此网站使用前后端分离架构，后端为 SOA 分布式架构,主要由承担入口、静态页面和负载均衡服务的 Nginx 服务器，应用服务器 Tomcat，数据库服务器 MySQL，缓存服务器 Memcached 服务器，以及阿里云提供的 OSS 对象存储服务器，这几部分组成。

演示网站地址：https://www.zhoukaifan.com

源码下载地址：https://github.com/zkaif/MyBlog

目　录

SOA 分布式网站的设计与制作

一、 系统实现软件

(一)IntelliJ IDEA

IDEA 全称 IntelliJ IDEA，是 java 语言开发的集成环境，IntelliJ 在业界被公认为最好的 java 开发工具之一，尤其在智能代码助手、代码自动提示、重构、J2EE 支持、各类版本工具(git、svn、github 等)、JUnit、CVS 整合、代码分析、 创新的 GUI 设计等方面的功能可以说是超常的。IDEA 是 JetBrains 公司的产品，这家公司总部位于捷克共和国的首都布拉格，开发人员以严谨著称的东欧程序员为主。它的旗舰版本还支持 HTML，CSS，PHP，MySQL，Python 等。（摘自：百度百科）

(二)Nginx

Nginx (engine x) 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 服务器。Nginx 是由伊戈尔•赛索耶夫为俄罗斯访问量第二的 Rambler.ru 站点（俄文：Рамблер）开发的，第一个公开版本 0.1.0 发布于 2004 年 10 月 4 日。

其将源代码以类 BSD 许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011 年 6 月 1 日，nginx 1.0.4 发布。

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个 BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上 nginx 的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用 nginx 网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。（摘自：百度百科）

(三)Tomcat

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现，Tomcat 5 支持最新的 Servlet 2.4 和 JSP 2.0 规范。因为 Tomcat 技术先进、性能稳定，而且免费，因而深受 Java 爱好者的喜爱并得到了部分软件开发

商的认可，成为目前比较流行的 Web 应用服务器。（摘自：百度百科）

(四)Memcached

Memcached 是一个高性能的分布式内存对象缓存系统，用于动态 Web 应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高动态、数据库驱动网站的速度。Memcached 基于一个存储键/值对的 hashmap。其守护进程（daemon ）是用 C 写的，但是客户端可以用任何语言来编写，并通过 memcached 协议与守护进程通信。（摘自：百度百科）

(五)MySQL

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL 是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。（摘自：百度百科）

(六)阿里云 OSS 对象存储

阿里云对象存储服务（Object Storage Service，简称 OSS），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。您可以通过调用 API，在任何应用、任何时间、任何地点上传和下载数据，也可以通过 Web 控制台对数据进行简单的管理。OSS 适合存放任意类型的文件，适合各种网站、开发企业及开发者使用。按实际容量付费真正使您专注于核心业务。（摘自：阿里云官方文档）

(七)JAVA

Java 是一门面向对象编程语言，不仅吸收了 C++语言的各种优点，还摒弃了 C++里难以理解的多继承、指针等概念，因此 Java 语言具有功能强大和简单易用两个特征。Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程 。（摘自：百度百科）

(八)HTML/CSS/JS

HTML：超文本标记语言，标准通用标记语言下的一个应用。“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。（摘自：百度百科）

CSS：层叠样式表(英文全称：Cascading Style Sheets)是一种用来表现 HTML（标准通用标记语言的一个应用）或 XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS 不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化（摘自：百度百科）

JS：JavaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网页增加动态功能。（摘自：百度百科）

系统分析

(九)用户群体

本系统的用户群体均为计算机行业的技术人员，所以对于页面的设计可以考虑引入更多的专业性元素

(十)网站目标

架设一个以计算机技术为中心的个人博客系统，以向他人展示自己为主，使网站作为一个他人了解自己的一个窗口。网站将分为前台和后台供他人浏览即自己管理网站。

(十一)功能划分

本系统功能分为两大块，一是提供给用户的展示互动，二是用于自己管理的后台功能。对于用户开放的功能主要有浏览已发布的博文、评论博文、查看他人评论、在线留言、分类浏览、最热博文推荐。对于管理员开放的有发表删除文章、发表删除评论、查看删除留言、发布文章。

(十二)功能描述

1.关于我：一个静态页面有我自己的自我介绍以及联系方式，所有人可见。

2.联系我：用于用户向我留言，反馈问题，留言后将可以通过后台查看。

3.博文浏览：用户可以查看管理员已发布的博文。

4.评论博文：用户可以对于谋篇文章发表自己的看法，评论内容将会公开。

5.最热文章：将根据文章的浏览次数排列出最热的几篇文章在首页展示。

6.分类浏览：用户可以浏览某个分类下或者某个时间段的文章。

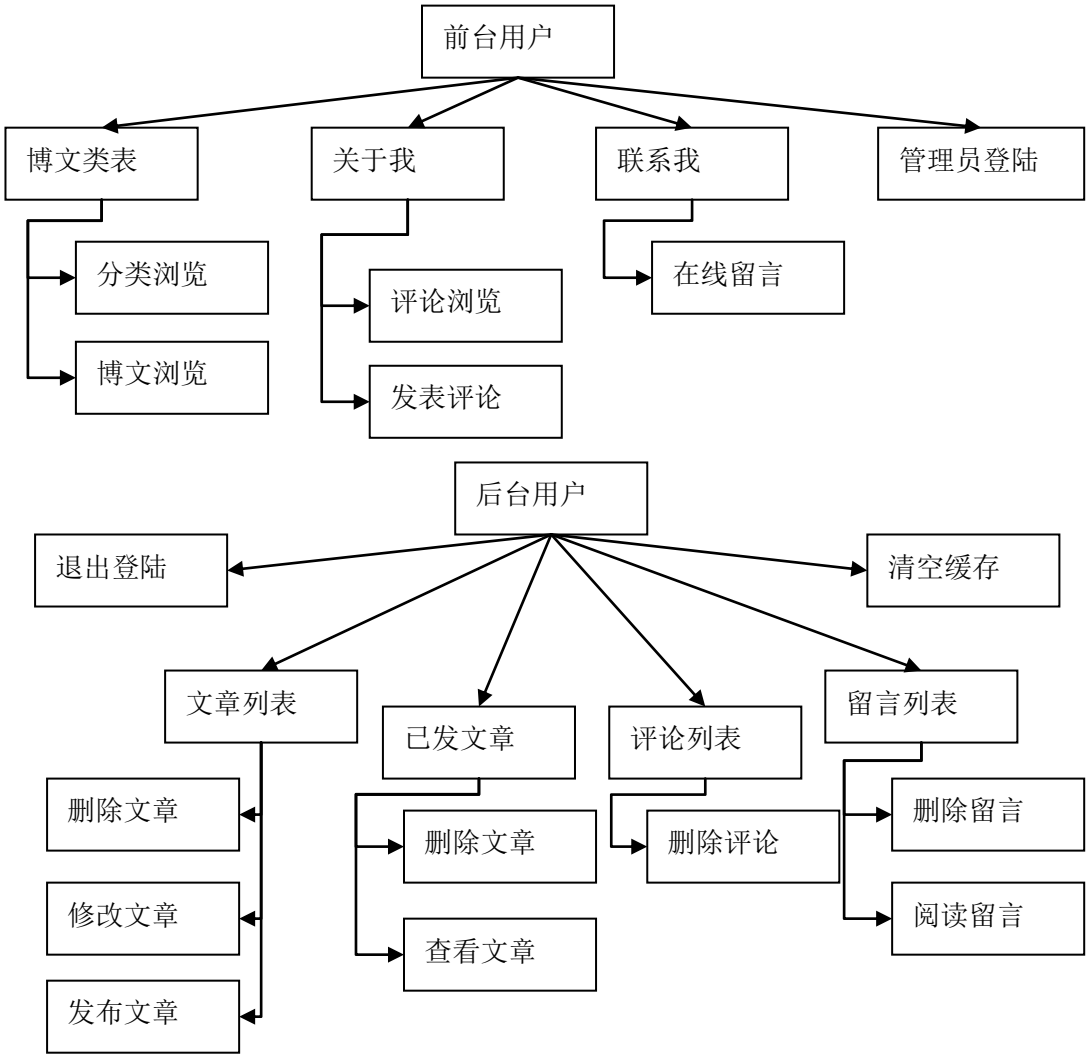
7.发布文章：管理员可以发布新文章。

8.删除文章：管理员可以删除文章。

9.删除评论：管理员可以删除评论。

10. 查看/删除留言：管理员可以查看/删除留言。

(十三)网站导航图



二、 系统架构

(一)整体架构

使用前后端分离架构，前后端使用 AJAX 技术进行通信，如图所示

架构优点：

1.前后端分离后可以降低系统开发过程中的复杂性，使得后端人员能更专注与后端的业务逻辑，而前端人员也能更专注与页面的效果实现。

2.前后端分离后页面响应无需等待后端的业务，异步加载的方式可以极大的提升用户体验。

3.使得前端页面与后端的耦合性降低，同一个系统可以拥有多套前端页面，我们只需匹配前后端之间的接口规范即可。

(二)后端架构

本系统使用了 SOA 分布式的架构，即面向服务。架构图如下（下下一页）

1.客户端：从 Nginx 服务器获得静态页面，然后使用 AJAX 技术与系统进行通信，客户端仅知道 Nginx 服务器的地址，仅能与 Nginx 服务器通信。

2.应用服务器 Tomcat：应用服务器是用于处理用户请求的服务器，所有业务逻辑均在此处理，通常情况下应用服务器可以有多台服务器同时部署，再利用 Nginx 的负载均衡对用户请求进行分发，以此来增大系统整体的并发量。另外应用服务器是不接受用户的直接访问的，除应用服务器间调用外，一切请求都需要通过 Nginx 服务器进行反向代理，应用服务器与用于处于不同的网络，Nginx 服务器是他们唯一的桥梁。应用服务器本身无存储数据的能力，数据存储需要依赖 OSS 和数据库。

3.应用服务器 Tomcat：应用服务器是用于处理用户请求的服务器，所有业务逻辑均在此处理，通常情况下应用服务器可以有多台服务器同时部署，再利用 Nginx 的负载均衡对用户请求进行分发，以此来增大系统整体的并发量。另外应用服务器是不接受用户的直接访问的，除应用服务器间调用外，一切请求都需要通过 Nginx 服务器进行反向代理，应用服务器与用于处于不同的网络，Nginx 服务器是他们唯一的桥梁。

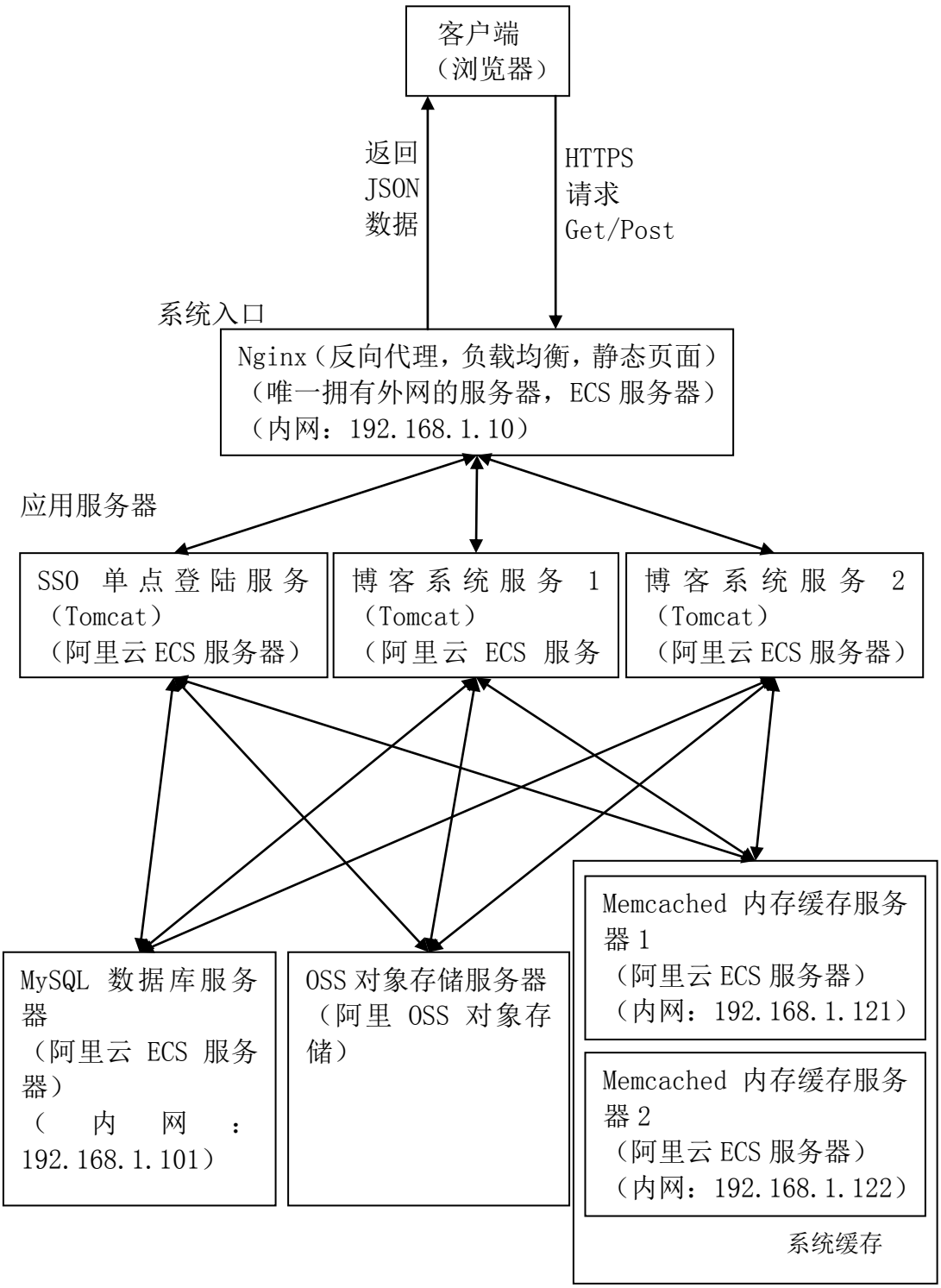
应用服务器本身无存储数据的能力，数据存储需要依赖 OSS 和数据库。

4.数据库服务器 MySQL：数据库服务器是对于系统数据进行存储的服务器，在系统中能长期保存数据的服务器有 OSS 和 MySQL 数据库，不同的是 MySQL 数据库主要用于保存关系型数据。

5.对象存储服务器 OSS：本系统中的 OSS 由阿里云提供的 OSS 服务其实现并不属于本文范畴，但这里还是简单介绍一下自己如何实现 OSS。通常情况下我们使用 Mongodb 数据库作为 OSS 的存储仓库，用 MySQL 数据建立索引。

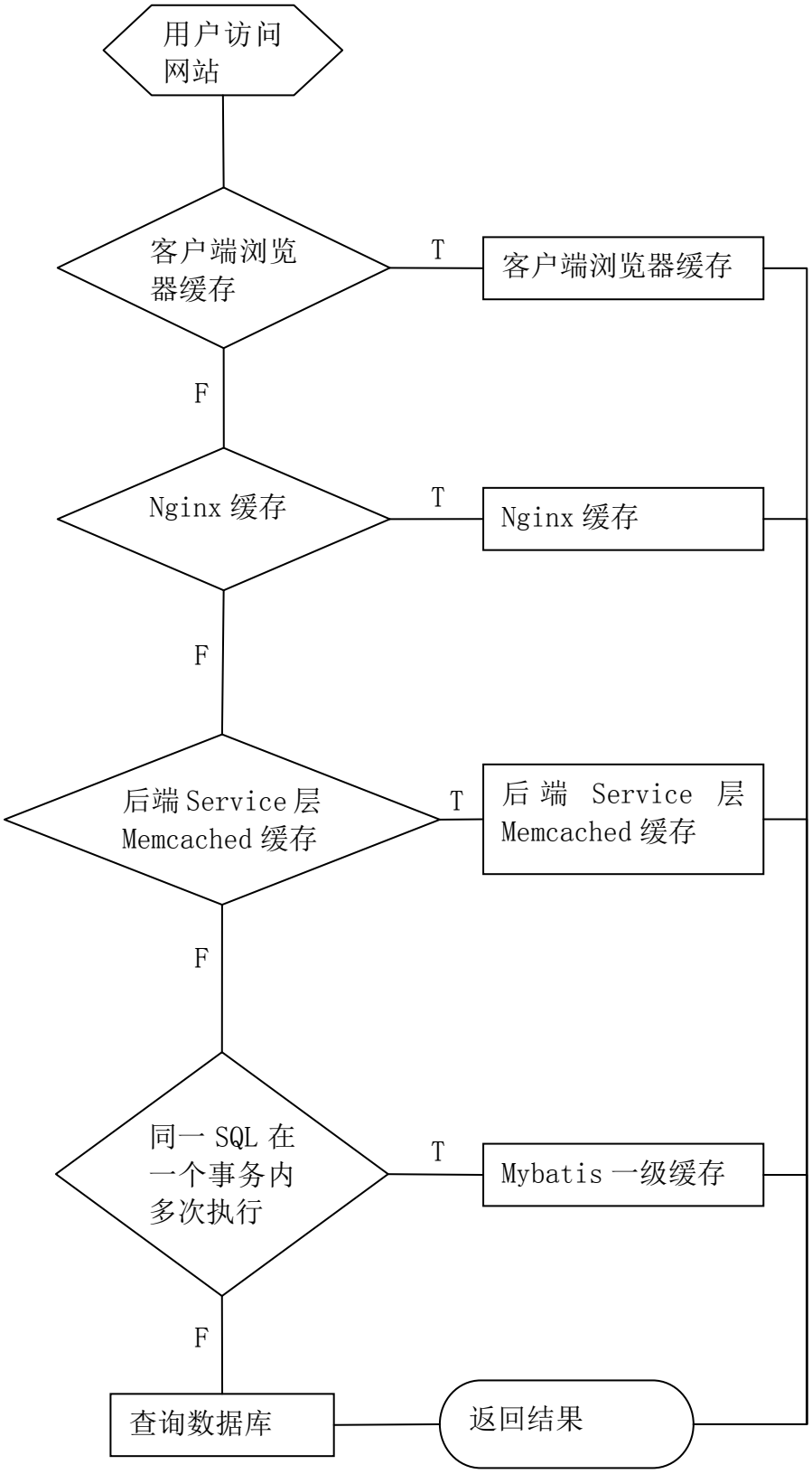
6.缓存服务器 Memcached：缓存服务器在系统中是一个数据临时仓库的服务器，它的作用是用于缓存 MySQL 中的数据减少数据库查询次数，并且 SSO 单点登陆中我们利用了缓存服务器来存放 Session 数据，以达到多系统间共享 Session 的目的。它所存储的数据都是临时的，或是数据副本。在系统重启后该服务器中数据将会清空，另外该服务器中的数据均存在超时时间。缓存服务器是一个 Key-Value 存储系统。

(三)后端架构图


客户端
（浏览器）
返回
JSON
数据
HTTPS
请求
Get/Post
系统入口
Nginx（反向代理，负载均衡，静态页面）
（唯一拥有外网的服务器，ECS 服务器）
（内网：192.168.1.10）
应用服务器
SSO 单点登陆服务
（Tomcat）
（阿里云 ECS 服务器）
博客系统服务 1
（Tomcat）
（阿里云 ECS 服务
博客系统服务 2
（Tomcat）
（阿里云 ECS 服务器）
MySQL 数据库服务器
（阿里云 ECS 服务器）
（内网：192.168.1.101）
OSS 对象存储服务器
（阿里 OSS 对象存储）
Memcached 内存缓存服务器 1
（阿里云 ECS 服务器）
（内网：192.168.1.121）
Memcached 内存缓存服务器 2
（阿里云 ECS 服务器）
（内网：192.168.1.122）
系统缓存

三、 缓存机制

(一)缓存流程


用户访问网站
客户端浏览器缓存
T
客户端浏览器缓存
F
Nginx 缓存
T
Nginx 缓存
F
后端 Service 层 Memcached 缓存
T
后端 Service 层 Memcached 缓存
F
同一 SQL 在一个事务内多次执行
T
Mybatis 一级缓存
F
查询数据库
返回结果

(二)客户端浏览器缓存

浏览器缓存（Browser Caching）是为了节约网络的资源加速浏览，浏览器在用户磁盘上对最近请求过的文档进行存储，当访问者再次请求这个页面时，浏览器就可以从本地磁盘显示文档，这样就可以加速页面的阅览。

浏览器缓存主要有两类：缓存协商：Last-modified ，Etag 和彻底缓存：cache-control，Expires。（摘自：百度百科）

该缓存用于在客户端缓存整个页面，缓存对象是页面。

(三)Nginx 缓存

Nginx 缓存是为了降低应用服务器的负载而设置的通常情况下 Nginx 的并发量是 Tomcat 的 10 倍，所以在 Nginx 服务器上缓存一些静态或是长期不变的资源能大大降低应用服务器的负载。

Nginx 缓存可减少请求在服务器内的转发次数。降低响应等待时间。提高整个集群的性能。

该缓存通常缓存一个请求的结构，缓存对象为静态资源或是长期不变的资源。

(四)Memcached 缓存

Memcached 缓存是应用服务器内部的缓存该缓存位于应用服务器的 Service 层上，该缓存的作用是减少数据库服务器的负载，通常用于缓存一个业务逻辑的结果。

该缓存通常缓存一个业务逻辑的结果，缓存对象是经过业务逻辑处理的数据库数据。

(五)Mybatis 一级缓存

Mybatis 一级缓存是一个 Dao 层框架提供的缓存，它用于缓存一个数据库事务内执行过的 SQL 查询语句结果，对于多次执行同样的 SQL 查询语句实际只执行一次。该缓存用于减少对数据的访问，故在设计业务逻辑时应该尽可能的使用原有的 API 而不是从新编写 SQL。

该缓存通常缓存数据库事务内的查询结果，缓存对象是数据库查询语句结果。

四、 系统设计与实现

(一)前端实现

1.主要技术

使用以 HTML/CSS/JS 为基础的前端开发技术。

使用 Bootstrap、jquery、highlight、LayUI 等框架实现。

2. 主要页面展示

首页

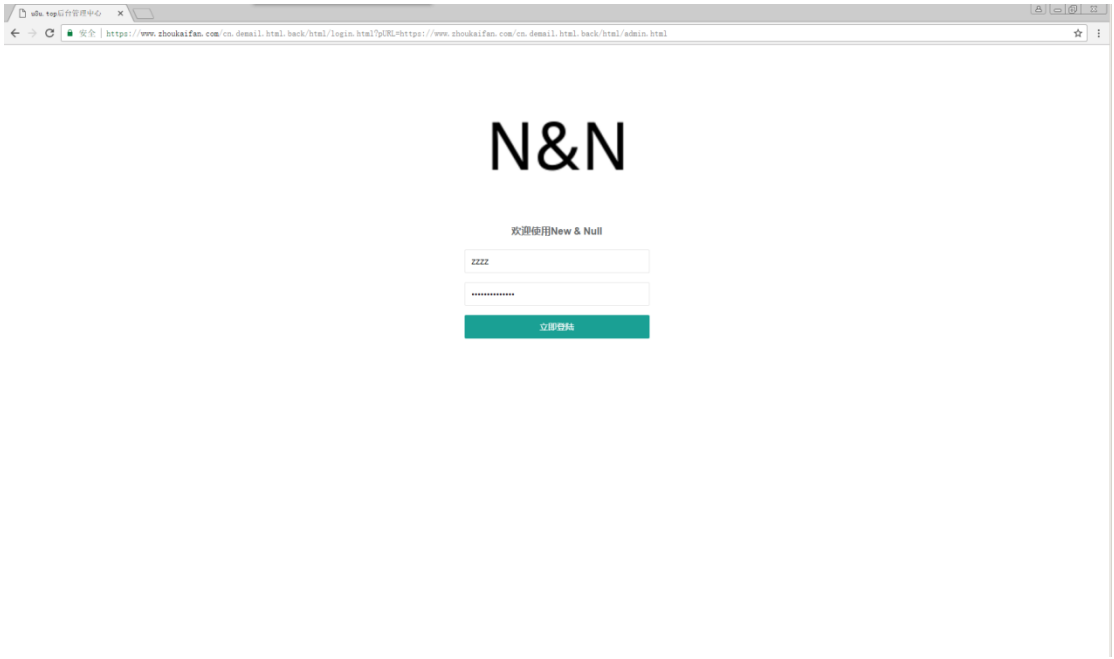
文章页面


New & Null
Home About Contact Login
网络请求与JSON解析类库
个人作品 • 2017-10-22 • 原创 • 0人阅读
JSONInterpret：一个自己编写用于处理不确定JSON数据格式情况下对JSON数据处理的类库； NetHttpPost ：针对Android的简化版的Http请求，针对Android中UI线程不能发起网络请求的特点提供requestThread方法，与JSONInterpretInt配合使用。
阅读全文
web-app_2_3.dtd不能使用EL表达式
WEB开发 • 2017-10-22 • 原创 • 1人阅读
今天搭建SSM环境测试时jsp居然拿不到ModelAndView的值,后来发现是在web-app_2_3.dtd的情况下El表达式不能用.....
阅读全文
JAVA数组Arrays类的常用方法
JAVA • 2017-10-22 • 原创 • 1人阅读
本文讲述了Arrays类中最常用的几个方法，并一一举例说明，本文主要用于复习笔记，
最热文章
JAVA的LinkedList类的简述 (7人阅读)
shiro身份验证 (4人阅读)
web-app_2_3.dtd不能使用EL表达式 (1人阅读)
JAVA数组Arrays类的常用方法 (1人阅读)
代理设计模式 (1人阅读)
网络请求与JSON解析类库 (0人阅读)
JAVA strictfp关键字 (0人阅读)
重写父类方法 (0人阅读)
文章分类
JAVA (共9篇)
WEB开发 (共2篇)
个人作品 (共2篇)
其他 (共1篇)
Linux (共1篇)
生活 (共0篇)
前端开发 (共0篇)
算法&数学 (共0篇)
历史文章
2017年10月 (共15篇)

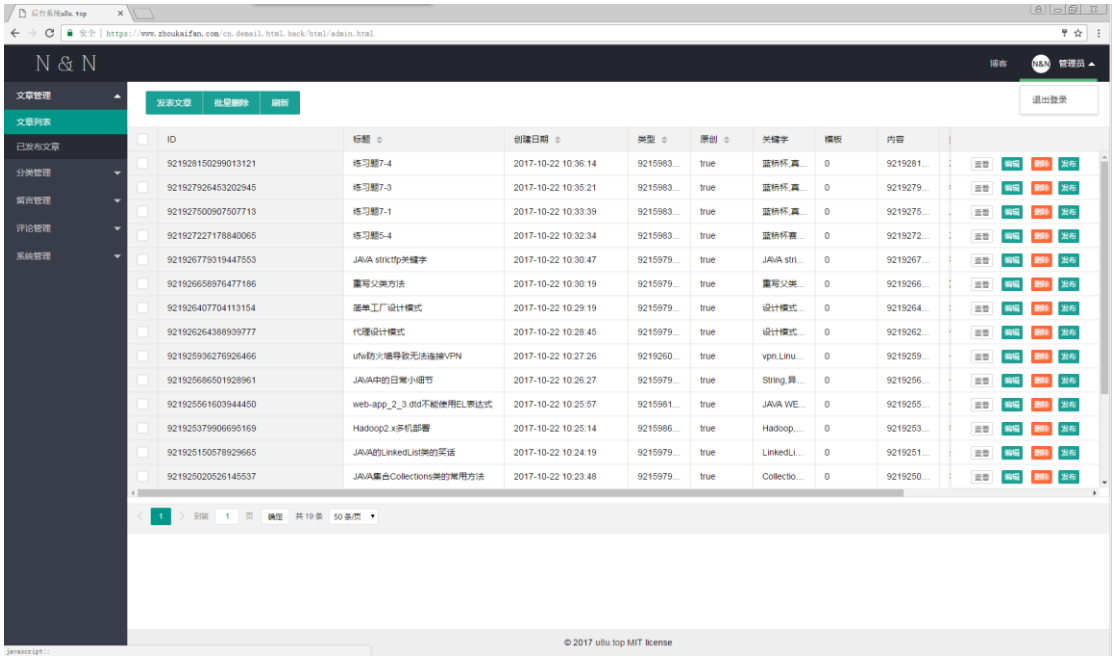
留言页面


New & Null
Home About Contact Login
在线留言
dsa
dsa
d
请在电子邮件地址中包括“@”。“dsa”中缺少“@”。
a
© 2017 zhoukaifan.com - New & Null

后台登陆页面



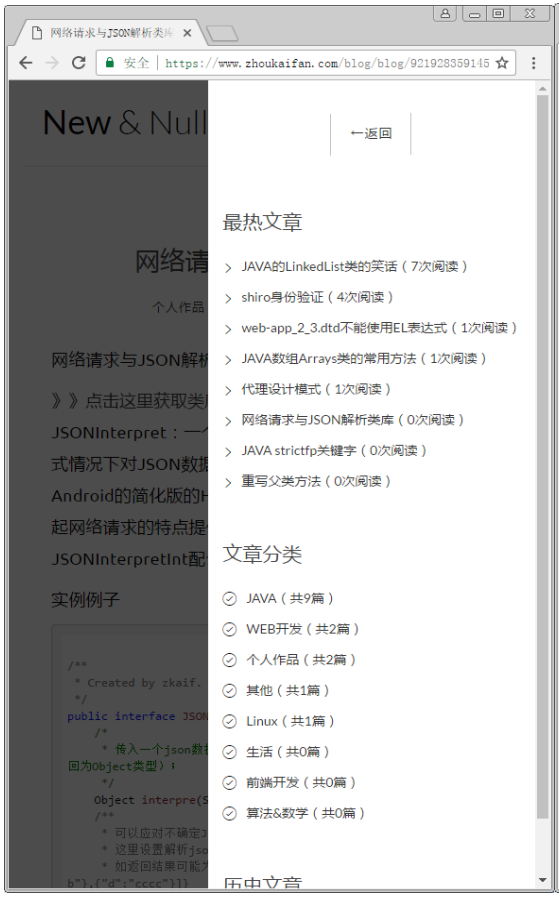
后台管理页面



手机端页面


New & Null
网络请求与JSON解析类库
个人作品 • 2017-10-22 • 原创 •0人阅读
JSONInterpret：一个自己编写用于处理不确定JSON数据格式情况下对JSON数据处理的类库； NetHttpPost ：针对Android的简化版的Http请求，针对Android中UI线程不能发起网络请求的特点提供requestThread方法，与JSONInterpretInt配合使用。
阅读全文 →
web-app_2_3.dtd不能使用EL表达式
WEB开发 • 2017-10-22 • 原创 •1人阅读
今天搭建SSM环境测试时jsp居然拿不到ModelAndView的值,后来发现是在web-app_2_3.dtd的情况下El表达式不能用.....
阅读全文 →


New & Null
网络请求与JSON解析类库
个人作品 • 2017-10-22 • 原创 •0人阅读
JSONInterpret：一个自己编写用于处理不确定JSON数据格式情况下对JSON数据处理的类库； NetHttpPost ：针对Android的简化版的Http请求，针对Android中UI线程不能发起网络请求的特点提供requestThread方法，与JSONInterpretInt配合使用。
阅读全文 →
web-app_2_3.dtd不能使用EL表达式
WEB开发 • 2017-10-22 • 原创 •1人阅读
今天搭建SSM环境测试时jsp居然拿不到ModelAndView的值,后来发现是在web-app_2_3.dtd的情况下El表达式不能用.....
阅读全文 →


←返回
最热文章
JAVA的LinkedList类的笑话（7次阅读）
shiro身份验证（4次阅读）
web-app_2_3.dtd不能使用EL表达式（1次阅读）
JAVA数组Arrays类的常用方法（1次阅读）
代理设计模式（1次阅读）
网络请求与JSON解析类库（0次阅读）
JAVA strictfp关键字（0次阅读）
重写父类方法（0次阅读）
文章分类
JAVA（共9篇）
WEB开发（共2篇）
个人作品（共2篇）
其他（共1篇）
Linux（共1篇）
生活（共0篇）
前端开发（共0篇）
算法&数学（共0篇）
历史文章


New & Null
在线留言
怎么称呼您？
您的E-mail是？
内容的主题是？
您要说点什么？
© 2017 zhoukaifan.com - New & Null

3. 主要页面展示

生成列表

```
$.ajax({
type: "get",
url: urlHot,
data: {
    count: 8,
    readerCount: -1
},
dataType: 'json',
success: function (msg) {
    //错误处理
    if (msg.code != 0) {
        return;
    }
    //成功处理
    var data = msg.body[0].data;
    var html = '';
    for (var i = 0; i < data.length; ++i) {
        html += '<li><a href="'+blogServer+data[i].uri+'">' +
            data[i].title + '（' + data[i].readerCount + '
            人阅读）</a></li>';
    }
    document.getElementById("hot").innerHTML = html;
},
error: function (data) {
}
});
$.ajax({
type: "get",
url: urlType,
dataType: 'json',
success: function (msg) {
    //错误处理
```

```
            if (msg.code != 0) {
                return;
            }
            //成功处理
            var data = msg.body[0].data;
            var html = '';
            for (var i = 0; i < data.length; ++i) {
                html += '<li><a href="javascript:showBlogContent(\" +
                    data[i].id + '\',undefined,undefined,undefined);
                    ">' + data[i].name + '（共' + data[i].blogCount
                     + '篇）</a></li>';
            }
            document.getElementById("type").innerHTML = html;
        },
        error: function (data) {
        }
});
$.ajax({
        type: "get",
        url: urlDate,
        dataType: 'json',
        success: function (msg) {
            //错误处理
            if (msg.code != 0) {
                return;
            }
            //成功处理
            var data = msg.body[0];
            var html = '';
            for (var i = 0; i < data.length; ++i) {
                var dates = data[i].date.split('-');
                html+='<li><a href="javascript:showBlogContent
                    (undefined,\" + data[i].id + '\',undefined,
                    undefined);">' +    dates[0]+'年'+dates[1]+'月' + '
```

```
                --（共' + data[i].blogCount + '篇）</a></li>';
            }
            document.getElementById("date").innerHTML = html;
        },
        error: function (data) {
        }
});
```

登陆页面 JS

```
    //监听提交
    layui.form.on('submit(login)', function (data) {
        var userName = layui.jquery("#userName").val();
        var password = layui.jquery("#password").val();
        password = hex_md5(password);
        password = hex_md5(password);
        password = hex_md5(password);//密码加密 3 次
        var url = ssoServer + 'user/login';
        layui.jquery.ajax({
            type: "post",
            url: url,
            data: {
                loginName: userName,
                password: password
            },
            success: function (msg) {
                if (msg.code != 0) {
                    layer.msg("登录失败！ ");
                    $("#userName").val("");
                    $("#password").val("");
                    return;
                }
                if (isEmpty(msg.body[0])){
                    layer.msg("登录失败！ ");
                    $("#userName").val("");
                    $("#password").val("");
```

```
                return;
            }
            setSID(msg.body[0]);
            window.location = pURL;
        },
        error: function (data) {
            layer.msg("登录失败，网络错误！");
        }
    });
    return false;
});
```

后台列表页

在对于框架进行封装后可以很方便的开发一个列表页

```
var modelList = [];
before = function (initData) {
initData.deleteURL=blogServer+'
    blogContentRelease/deletesById/';
initData.listURL = blogServer + 'blogContentRelease/list';
initData.barType = 'rdBar';
initData.barHtml = '';
initData.tableHead.push(
    {field: 'title', title: '标题', width: 260, sort: true},
    {field: 'cTime', title: '创建日期', width: 200, sort: true},
    {field: 'typeId', title: '类型', width: 100, sort: true},
    {field: 'original', title: '原创', width: 100, sort: true},
    {field: 'readerCount', title: '阅读量', width: 100,
        sort: true},
    {field: 'keyWord', title: '关键字', width: 100},
    {field: 'blogContentId', title: '文章 ID', width: 100},
    {field: 'htmlContentId', title: '页面 ID', width: 100},
    {field: 'digest', title: '描述', width: 400},
    {field: 'uri', title: '页面地址', width: 500}
);
```

```
};
after = function () {
};
barDetail = function (obj) {
    var layer = layui.layer;
    //sessionStorage.setItem("id",obj.data.id);

    if (obj.event === 'detail') {
        layer.open({
            type: 2,
            title: '博客查看',
            shadeClose: true,
            shade: 0.8,
            area: ['70%', '80%'],
            content: blogServer + obj.data.uri //iframe 的 url
        });
    }
}
```

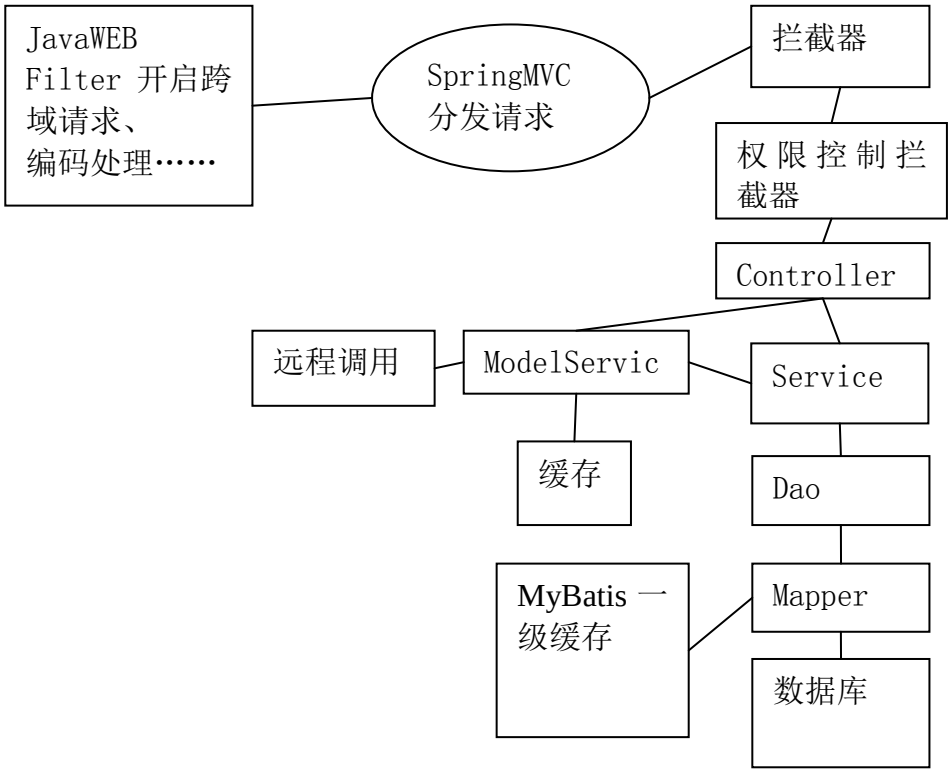
(二)后端实现

1.主要技术

以 JAVA WEB 为基础的技术进行开发。

主要使用了 Spring、Spring MVC、Mybatis、Mybatis Puls、fastJson、OSS、XMemcached 等开源框架。

2.技术架构



3.接口举例

请求文章列表接口”/blogContent/list”

参数：

pageD：分页后的第 N 页面

countD：按照此大小分页

typeId：过滤出该类型数据

dateId：过滤出这个时间段的数据

original：过滤出是否原创的数据

SID：用户登陆后获取到的 Token

返回：

{"body":[{"count":"3","data":[{"blogType":{"blogCount":"0","countD":"0","delFlag":"false","id":"92159833196247041","name":" 算 法 & 数学","pageD":"0"},"content":"","contentId":"921928150152212481","countD":"0","date":"2017-10-22 10:36:14","dateId":"921615819241226242","delFlag":"false","digest":"X 星球的机器人表演拉拉队有两种服装，A 和 B。\n 他们这次表演的是搭机器人塔。","id":"921928150299013121","keyWord":"蓝桥

杯,真题","original":"true","pageD":"0","templateId":"0","title":"练习题 7-4","typeId":"92159833196247041"},{"blogType":{"blogCount":"0","countD":"0","delFlag":"false","id":"921598333196247041","name":"算法&数学","pageD":"0"},"content":"","contentId":"921927926289625089","countD":"0","date":"2017-10-22 10:35:21","dateId":"921615819241226242","delFlag":"false","digest":"棋子换位\n 有 n 个棋子 A，n 个棋子 B，在棋盘上排成一行。\n 它们中间隔着一个空位，用“.”表示，比如：AAA.BBB","id":"921927926453202945","keyWord":"蓝桥杯,真题","original":"true","pageD":"0","templateId":"0","title":"练习题 7-3","typeId":"92159833196247041"}],"page":"1","total":"19","totalPage":"7"}],"code":"0","des":"","mgs":{}}

删除评论接口”/comment/listByBlogId/{blogId}”

{ blogId }为评论 ID，多个 ID 可如下填写

/comment/listByBlogId/1,2,3,4,5

成功返回:

{"body":[],"code":"0","des":"","mgs":{}}

请求接口需要登陆时而用户为登陆

返回：

{"code":"1","des":"no login!!!","mgs":{},"body":[""]}

4.部分代码示例

Controller 层：

```
@RequestMapping("list")
public Message list(BlogContent blogContent){
    Message message = new Message();
    Page<BlogContent> blogContentPage =
        myBlogService.getBlogContentList(blogContent);
    message.add(blogContentPage);
    return message;
}
```

```
@RequestMapping("getById/{id}")
public Message getById(@PathVariable("id") long id){
        Message message = new Message();
        BlogContent blogContent =
                moduleService.getBlogContentById(id);
        message.add(blogContent);
        return message;
}
```

Service 层：

```
@Override
public Page<BlogContent> getBlogContentList
      (BlogContent blogContent) {
      Page<BlogContent> blogContentPage =
            blogContentDao.getAllPage(blogContent);
      List<BlogContent> blogContents = blogContentPage.getData();
      for (BlogContent blogContent1 : blogContents) {
            BlogType blogType =
                  blogTypeDao.getById(blogContent1.getTypeId());
            blogContent1.setBlogType(blogType);
       }
      return blogContentPage;
}

//ModelService
@Override
 public BlogContent getBlogContentById(long id) {
       BlogContent blogContent =
                        myBlogService.getBlogContentById(id);
      Content content = stroageService.
            getContentById(blogContent.getContentId());
       if (content == null) {
             return blogContent;
       }
       String contentStr =
```

```
                stroageService.getObjectStrByContent(content);
        blogContent.setContent(contentStr);
        return blogContent;
}
```

Dao 层：

```
@Override
public Page<BlogContent> getAllPage(BlogContent blogContent){
        Wrapper<BlogContent> wrapper = new EntityWrapper<>();
        if(blogContent.getOriginal()!=null)
                wrapper.eq("ORIGINAL",blogContent.getOriginal());
        if(blogContent.getDateId()>0)
                wrapper.eq("DATE_ID",blogContent.getDateId());
        if(blogContent.getTypeId()>0)
                wrapper.eq("TYPE_ID",blogContent.getTypeId());
        wrapper.orderBy("DATE",false);
        return super.getAllPage(blogContent,wrapper);
}
```

Mapper：

```
public interface BlogContentReleaseMapper extends
                BaseMapper<BlogContentRelease> {
        BlogContentRelease getByBlogContenId(
                @Param("id") long id);
        void addReaderCount(@Param("id") long id,
                @Param("count") int count);
        boolean updateByIdPhysics(BlogContentRelease
                blogContentRelease);
}
```

领域对象：

```
 @TableName("DATE_TYPE")
 public class DateType extends BaseDomain {
        private static final long serialVersionUID = -
                        617899884376724083L;
        @TableField("DATE")
        private Date date;
```

```
    @TableField(exist = false)
    private long blogCount;
    public long getBlogCount() {return blogCount;}
    public void setBlogCount(long blogCount) {
        this.blogCount = blogCount;
    }
    public Date getDate() {return date;}
    public void setDate(Date date) {this.date = date;}
}
```

5.权限控制

权限控制系统也是自己开发的使用了 Shiro 框架，但是权限控制系统是一个独立的系统并不是文将要介绍，但是由于本文使用了该系统故简单的介绍一下如何进入该系统。

权限控制系统采用了非入侵式的设计它可以兼容所有使用 SpringMVC 搭建的系统。我们可以这样引用该系统

1.引入依赖 Jar 包 sso-client，如果是 Maven 可以这样引入

```
<dependency>
    <groupId>top.u8u</groupId>
    <artifactId>sso-client</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

2.在 Spring 配置文件中加入一句

```
<import resource=
    "classpath:config/spring-sso-client-module.xml"/>
```

3.在在 Spring 设定一个产量 sso.anon.uri 表示可以不经过授权访问的接口。

```
sso.anon.uri=/blogContentRelease/list;/blogContentRelease/addReaderCo
unt/*;/blogContentRelease/getReaderCount/*;\
/blog/*.html;/blogType/list;/comment/create;/comment/listByBlogId/*;\
/contactMessage/create;/dateType/list;
```

多条 URL 使用”;“分割。

(三)数据库设计

1.数据库表结构

BLOG_CONTENT

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
DATE	DATETIME	NULL	发表日期
DIGEST	VARCHAR	NULL	文章描述
ORIGINAL	TINYINT	NULL	是否原创
TEMPLATE_ID	BIGINT	NULL	模板 ID
CONTENT_ID	BIGINT	NULL	OSS 中的 ID
TITLE	VARCHAR	NULL	标题
DATE_ID	BIGINT	NULL	日期索引 ID
TYPE_ID	BIGINT	NULL	类型 ID
KEY_WORD	VARCHAR	NULL	搜索关键字

BLOG_CONTENT_RELEASE

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
C_TIME	DATETIME	NULL	发布日期
DIGEST	VARCHAR	NULL	副本用于搜索
ORIGINAL	TINYINT	NULL	副本用于搜索
BLOG_CONTENT_ID	BIGINT	NULL	文章 ID
TITLE	VARCHAR	NULL	副本用于搜索
DATE_ID	BIGINT	NULL	副本用于搜索
TYPE_ID	BIGINT	NULL	副本用于搜索
KEY_WORD	VARCHAR	NULL	副本用于搜索
HTML_CONTENT_ID	BIGINT	NULL	OSS 中的页面 ID
URI	VARCHAR	NULL	OSS 中的页面 URI

READER_COUNT	INT	NULL	文章的阅读次数

BLOG_TYPE

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
NAME	VARCHAR	NULL	分类名称

DATE_TYPE

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
DATE	DATETIME	NULL	时间

COMMENT

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
P_ID	BIGINT	NULL	记录树形结构
NAME	VARCHAR	NULL	分类名称
BLOG_CONTENT_ID	BIGINT	NULL	评论的文章
CONTENT	VARCHAR	NULL	评论内容
DATE	DATETIME	NULL	评论时间

CONTACT_MESSAGE

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
NAME	VARCHAR	NULL	留言人
TITLE	VARCHAR	NULL	留言标题
EMAIL	VARCHAR	NULL	邮箱
CONTENT	VARCHAR	NULL	留言内容
READ	TINYINT	NULL	是否已阅读

CONTENT

名称	数据类型	是否可为空	说明
ID	BIGINT	NOT NULL(主键)	主键 ID
DEL_FLAG	TINYINT	NOT NULL	逻辑删除
C_TIME	DATETIME	NULL	创建日期
URI	VARCHAR	NULL	文件 URI
OSS_KEY	VARCHAR	NULL	OSS 中的 KEY
URI_FLAG	TINYINT	NULL	是否可下载
FILE_NAME	VARCHAR	NULL	文件名称

2.数据库 ER 关系图