# 富文本编辑器

## 入门案例

### 引入js

<%@ page language="java" contentType="text/html; charset=utf-8"

pageEncoding="utf-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<link href="/js/kindeditor-4.1.10/themes/default/default.css" type="text/css" rel="stylesheet">

<script type="text/javascript" charset="utf-8" src="/js/kindeditor-4.1.10/kindeditor-all-min.js"></script>

<script type="text/javascript" charset="utf-8" src="/js/kindeditor-4.1.10/lang/zh\_CN.js"></script>

<script type="text/javascript" charset="utf-8" src="/js/jquery-easyui-1.4.1/jquery.min.js"></script>

<script type="text/javascript">

$(**function**(){

KindEditor.ready(**function**(){

KindEditor.create("#editor")

})

})

</script>

</head>

<body>

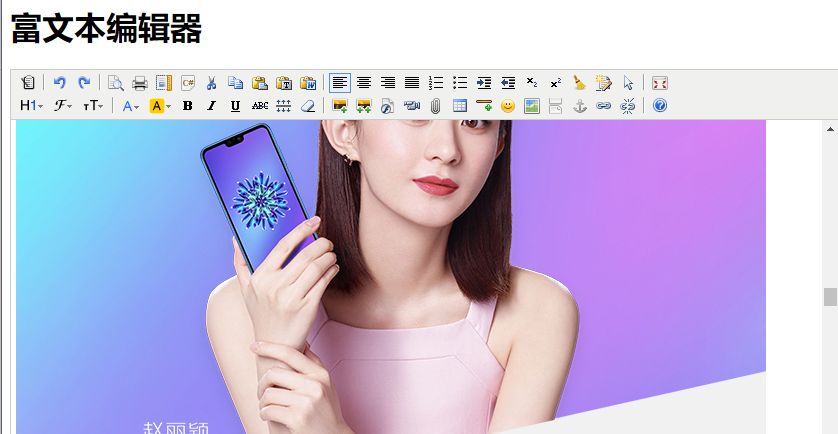
<h1>富文本编辑器</h1>

<textarea style="width:700px;height:350px" id="editor"></textarea>

</body>

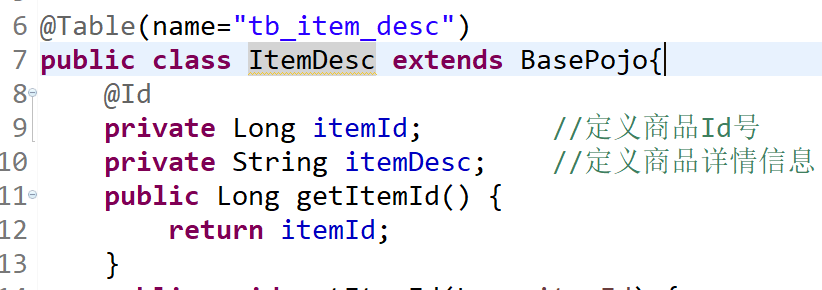
</html>

### 页面展现



## 实现商品描述信息新增

### 编辑POJO对象



### 编辑Controller

//实现商品的新增

@RequestMapping("/save")

@ResponseBody

**public** SysResult saveItem(Item item,String desc){

**try** {

itemService.saveItem(item,desc);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"商品新增失败");

}

### 编辑Service

/\*\*

\* 问题描述:

\* 因为item入库是主键自增,只有入库操作执行后,才能获取

\* id值.但是itemDesc入库时,必须有id主键才行.并且ItemDesc

\* 中的Id值必须和Item中的Id值一致.如何解决??

\*

\* 解决方案:

\* 对应操作数据库的线程而言,每次入库都可以获取当前线程的

\* 最后插入的Id值.只要获取该Id即可.

\* INSERT INTO USER VALUES(NULL,"马化腾",18,"男");

SELECT LAST\_INSERT\_ID();

\*/

@Override

**public** **void** saveItem(Item item,String desc) {

item.setStatus(1); //表示商品正常

item.setCreated(**new** Date());

item.setUpdated(item.getCreated());

itemMapper.insert(item);

//实现商品描述信息入库操作

ItemDesc itemDesc = **new** ItemDesc();

itemDesc.setItemId(item.getId());

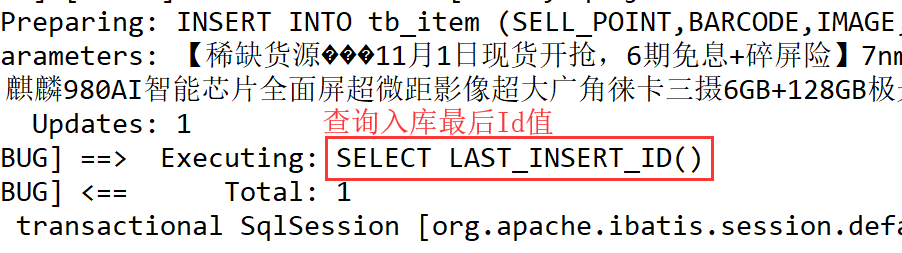
itemDesc.setItemDesc(desc);

itemDesc.setCreated(item.getCreated());

itemDesc.setUpdated(item.getCreated());

itemDescMapper.insert(itemDesc);

}



## 商品详情回显

### 页面分析

1.页面JS

$.getJSON('/item/query/item/desc/'+data.id,**function**(\_data){

**if**(\_data.status == 200){

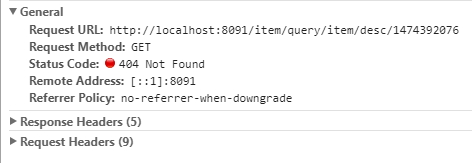
//UM.getEditor('itemeEditDescEditor').setContent(\_data.data.itemDesc, false);

itemEditEditor.html(\_data.data.itemDesc);

}

});

2.用户请求路径



### 编辑Controller

//根据商品Id查询商品详情信息

@RequestMapping("/query/item/desc/{itemId}")

@ResponseBody

**public** SysResult findItemDescById(@PathVariable Long itemId){

**try** {

ItemDesc itemDesc = itemService.findItemDescById(itemId);

**return** SysResult.*oK*(itemDesc);

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"商品详情查询失败");

}

### 编辑Service

@Override

**public** ItemDesc findItemDescById(Long itemId) {

**return** itemDescMapper.selectByPrimaryKey(itemId);

}

### 页面效果



## 商品详情更新

### 编辑Controller

说明:在原有方法中进行扩展

//实现商品信息的修改

@RequestMapping("/update")

@ResponseBody

**public** SysResult updateItem(Item item,String desc){

**try** {

itemService.updateItem(item,desc);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"商品更新失败");

}

### 编辑Service

@Override

**public** **void** updateItem(Item item,String desc) {

item.setUpdated(**new** Date());

//动态更新,更新其中不为空的数据

itemMapper.updateByPrimaryKeySelective(item);

ItemDesc itemDesc = **new** ItemDesc();

itemDesc.setItemDesc(desc);

itemDesc.setItemId(item.getId());

itemDesc.setUpdated(item.getUpdated());

itemDescMapper.updateByPrimaryKeySelective(itemDesc);

}

## 商品删除

### 编辑Controller

//实现商品信息删除

@RequestMapping("/delete")

@ResponseBody

**public** SysResult deleteItems(String[] ids){

**try** {

itemService.deleteItems(ids);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"商品删除失败");

}

### 编辑Service

/\*

\* 规范:一般删除时,先删除关联表中的数据,

\* 之后再删除主表中的数据

\* 发展:数据库中的关联关系,会影响程序的执行性能,

\* 所以后期时,数据库中的全部的关联关系,都通过

\* 业务层代码维护.主键除外.

\*/

@Override

**public** **void** deleteItems(String[] ids) {

itemDescMapper.deleteByIDS(ids);

itemMapper.deleteByIDS(ids);

}

## 文件上传入门

### 如何实现文件上传

1. 页面需要开放多媒体标签
2. 接收图片信息,通过IO流写入磁盘(调用解析器中的方法即可)
3. 配置文件上传视图解析器.

### 编辑页面

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>实现文件长传</h1>

<form action="http://localhost:8091/file" method="post"

**enctype="multipart/form-data"**>

<input name="fileImage" type="file" />

<input type="submit" value="提交"/>

</form>

</body>

</html>

### 编辑Controller

@Controller

**public** **class** FileController {

//实现文件上传入门案例

@RequestMapping("/file")

**public** String file(MultipartFile **fileImage**) **throws** IllegalStateException, IOException{

//1.判断文件夹是否存在

File fileDir = **new** File("E:/jt-upload");

**if**(!fileDir.exists()){

fileDir.mkdirs();//创建文件夹

}

//abc.jpg

String fileName = fileImage.getOriginalFilename();

//实现文件上传

fileImage.**transferTo**(**new** File("E:/jt-upload/"+fileName));

//使用重定向技术

**return** "redirect:/file.jsp";

//转发

//return "forword:/file.jsp";

}

}

### 配置文件上传视图解析器

<!--配置文件上传视图解析器

id的值必须为multipartResolver

-->

<bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<property name=*"maxUploadSize"* value=*"10485760"*/>

<property name=*"defaultEncoding"* value=*"utf-8"*/>

</bean>

## 富文本编辑器实现文件上传

### 正则表达式

|  |  |
| --- | --- |
| 元字符 | 描述 |
| \ | 将下一个字符标记符、或一个向后引用、或一个八进制转义符。例如，“\\n”匹配\n。“\n”匹配换行符。序列“\\”匹配“\”而“\(”则匹配“(”。即相当于多种编程语言中都有的“转义字符”的概念。 |
| ^ | 匹配输入字符串的开始位置。如果设置了RegExp对象的Multiline属性，^也匹配“\n”或“\r”之后的位置。 |
| $ | 匹配输入字符串的结束位置。如果设置了RegExp对象的Multiline属性，$也匹配“\n”或“\r”之前的位置。 |
| \* | 匹配前面的子表达式任意次。例如，zo\*能匹配“z”，也能匹配“zo”以及“zoo”。\*等价于o{0,} |
| + | 匹配前面的子表达式一次或多次(大于等于1次）。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。 |
| ? | 匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”中的“do”。?等价于{0,1}。 |
| {n} | n是一个非负整数。匹配确定的n次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个o。 |
| {n,} | n是一个非负整数。至少匹配n次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o\*”。 |
| {n,m} | m和n均为非负整数，其中n<=m。最少匹配n次且最多匹配m次。例如，“o{1,3}”将匹配“fooooood”中的前三个o为一组，后三个o为一组。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格。 |
| ? | 当该字符紧跟在任何一个其他限制符（\*,+,?，{n}，{n,}，{n,m}）后面时，匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如，对于字符串“oooo”，“o+”将尽可能多的匹配“o”，得到结果[“oooo”]，而“o+?”将尽可能少的匹配“o”，得到结果 ['o', 'o', 'o', 'o'] |
| .点 | 匹配除“\r\n”之外的任何单个字符。要匹配包括“\r\n”在内的任何字符，请使用像“[\s\S]”的模式。 |
| (pattern) | 匹配pattern并获取这一匹配。所获取的匹配可以从产生的Matches集合得到，在VBScript中使用SubMatches集合，在JScript中则使用$0…$9属性。要匹配圆括号字符，请使用“\(”或“\)”。 |
| (?:pattern) | 非获取匹配，匹配pattern但不获取匹配结果，不进行存储供以后使用。这在使用或字符“(|)”来组合一个模式的各个部分时很有用。例如“industr(?:y|ies)”就是一个比“industry|industries”更简略的表达式。 |
| (?=pattern) | 非获取匹配，正向肯定预查，在任何匹配pattern的字符串开始处匹配查找字符串，该匹配不需要获取供以后使用。例如，“Windows(?=95|98|NT|2000)”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。 |
| (?!pattern) | 非获取匹配，正向否定预查，在任何不匹配pattern的字符串开始处匹配查找字符串，该匹配不需要获取供以后使用。例如“Windows(?!95|98|NT|2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。 |
| (?<=pattern) | 非获取匹配，反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“(?<=95|98|NT|2000)Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”。 |
| (?<!pattern) | 非获取匹配，反向否定预查，与正向否定预查类似，只是方向相反。例如“(?<!95|98|NT|2000)Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”。这个地方不正确，有问题  此处用或任意一项都不能超过2位，如“(?<!95|98|NT|20)Windows正确，“(?<!95|980|NT|20)Windows 报错，若是单独使用则无限制，如(?<!2000)Windows 正确匹配 |
| x|y | 匹配x或y。例如，“z|food”能匹配“z”或“food”(此处请谨慎)。“[zf]ood”则匹配“zood”或“food”。 |
| [xyz] | 字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”。 |
| [^xyz] | 负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“plin”。 |
| [a-z] | 字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。  注意:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围; 如果出字符组的开头,则只能表示连字符本身. |
| [^a-z] | 负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。 |
| \b | 匹配一个单词边界，也就是指单词和空格间的位置（即正则表达式的“匹配”有两种概念，一种是匹配字符，一种是匹配位置，这里的\b就是匹配位置的）。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”。 |
| \B | 匹配非单词边界。“er\B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”。 |
| \cx | 匹配由x指明的控制字符。例如，\cM匹配一个Control-M或回车符。x的值必须为A-Z或a-z之一。否则，将c视为一个原义的“c”字符。 |
| \d | 匹配一个数字字符。等价于[0-9]。grep 要加上-P，perl正则支持 |
| \D | 匹配一个非数字字符。等价于[^0-9]。grep要加上-P，perl正则支持 |
| \f | 匹配一个换页符。等价于\x0c和\cL。 |
| \n | 匹配一个换行符。等价于\x0a和\cJ。 |
| \r | 匹配一个回车符。等价于\x0d和\cM。 |
| \s | 匹配任何不可见字符，包括空格、制表符、换页符等等。等价于[ \f\n\r\t\v]。 |
| \S | 匹配任何可见字符。等价于[^ \f\n\r\t\v]。 |
| \t | 匹配一个制表符。等价于\x09和\cI。 |
| \v | 匹配一个垂直制表符。等价于\x0b和\cK。 |
| \w | 匹配包括下划线的任何单词字符。类似但不等价于“[A-Za-z0-9\_]”，这里的"单词"字符使用Unicode字符集。 |
| \W | 匹配任何非单词字符。等价于“[^A-Za-z0-9\_]”。 |
| \xn | 匹配n，其中n为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，“\x41”匹配“A”。“\x041”则等价于“\x04&1”。正则表达式中可以使用ASCII编码。 |
| \num | 匹配num，其中num是一个正整数。对所获取的匹配的引用。例如，“(.)\1”匹配两个连续的相同字符。 |
| \n | 标识一个八进制转义值或一个向后引用。如果\n之前至少n个获取的子表达式，则n为向后引用。否则，如果n为八进制数字（0-7），则n为一个八进制转义值。 |
| \nm | 标识一个八进制转义值或一个向后引用。如果\nm之前至少有nm个获得子表达式，则nm为向后引用。如果\nm之前至少有n个获取，则n为一个后跟文字m的向后引用。如果前面的条件都不满足，若n和m均为八进制数字（0-7），则\nm将匹配八进制转义值nm。 |
| \nml | 如果n为八进制数字（0-7），且m和l均为八进制数字（0-7），则匹配八进制转义值nml。 |
| \un | 匹配n，其中n是一个用四个十六进制数字表示的Unicode字符。例如，\u00A9匹配版权符号（&copy;）。 |
| \p{P} | 小写 p 是 property 的意思，表示 Unicode 属性，用于 Unicode 正表达式的前缀。中括号内的“P”表示Unicode 字符集七个字符属性之一：标点字符。  其他六个属性：  L：字母；  M：标记符号（一般不会单独出现）；  Z：分隔符（比如空格、换行等）；  S：符号（比如数学符号、货币符号等）；  N：数字（比如阿拉伯数字、罗马数字等）；  C：其他字符。  *\*注：此语法部分语言不支持，例：javascript。* |
| \<  \> | 匹配词（word）的开始（\<）和结束（\>）。例如正则表达式\<the\>能够匹配字符串"for the wise"中的"the"，但是不能匹配字符串"otherwise"中的"the"。注意：这个元字符不是所有的软件都支持的。 |
| ( ) | 将( 和 ) 之间的表达式定义为“组”（group），并且将匹配这个表达式的字符保存到一个临时区域（一个正则表达式中最多可以保存9个），它们可以用 \1 到\9 的符号来引用。 |
| | | 将两个匹配条件进行逻辑“或”（Or）运算。例如正则表达式(him|her) 匹配"it belongs to him"和"it belongs to her"，但是不能匹配"it belongs to them."。注意：这个元字符不是所有的软件都支持的。 |

### 页面分析



### Vo对象的封装

{"error":0,"url":"图片的保存路径","width":图片的宽度,"height":图片的高度}

参数说明： 0代表是一张图片，如果是0，前台才可以解析并显示。1代表不是图片，

不显示如果不设置宽度和高度，则默认用图片原来的大小，所以不用设置

### 编辑Controller

//实现文件上传

@RequestMapping("/pic/upload")

@ResponseBody

**public** PicUploadResult uploadFile(MultipartFile uploadFile){

**return** fileService.uploadFile(uploadFile);

}

### 编辑Service

@Service

**public** **class** FileServiceImpl **implements** FileService {

**private** String dirPath = "E:/jt-upload/";

//定义虚拟路径

**private** String urlPath = "http://image.jt.com/";

/\*\*

\*思路:

\* 1.校验上传的文件是否为图片 jpg|png|gif

\* 2.判断文件是否为恶意程序 exe/rpm

\* 3.为了提高检索效率,一般采用分文件存储 层级最好不要超过5级.

\* 4.为了防止文件名称重复,采用动态的方式获取文件名称.

\* ^开始 $结束 .除了/r/n的任意字符 +一个或者多个 \* 没有或者多个 () 组or 或者

\*/

@Override

**public** PicUploadResult uploadFile(MultipartFile uploadFile) {

PicUploadResult result = **new** PicUploadResult();

//1.校验文件的名称 abc.jpg

String fileName = uploadFile.getOriginalFilename();

fileName = fileName.toLowerCase(); //将数据转化为小写

**if**(!fileName.matches("^.\*\\.(jpg|png|gif)$")){

result.setError(1);

**return** result; //提前返回

}

//2.判断是否为恶意程序

**try** {

BufferedImage bufferedImage =

ImageIO.*read*(uploadFile.getInputStream());

**int** width = bufferedImage.getWidth();

**int** height = bufferedImage.getHeight();

**if**( width == 0 || height == 0){

result.setError(1);//表示不是图片

**return** result;

}

/\*\*

\* 3.为了提高检索效率,采用分文件存储格式

\* yyyy/MM/dd

\*/

String dateDir = **new** SimpleDateFormat("yyyy/MM/dd").format(**new** Date());

//4.保证文件不重名 UUID + 随机数0-999

String uuid = UUID.*randomUUID*().toString()

.replace("-", "");

**int** random = **new** Random().nextInt(1000);

String fileType =

fileName.substring(fileName.lastIndexOf("."));

String uuidFileName = uuid + random + fileType;

//定义文件夹路径 E:/jt-upload/2018/11/11

String fileDirPath = dirPath + dateDir;

File fileDir = **new** File(fileDirPath);

**if**(!fileDir.exists()){

fileDir.mkdirs();

}

//实现文件上传 E:/jt-upload/2018/11/11/abc.jpg

File realFilePath = **new** File(fileDirPath + "/" +uuidFileName);

uploadFile.transferTo(realFilePath);

//设定宽度和高度

result.setHeight(height+"");

result.setWidth(width+"");

/\*

\* 为了实现图片页面访问,拼接虚拟路径

\* http://image.jt.com/2018/11/11/abc.jpg

\*/

String realUrlPath = urlPath + dateDir + "/" + uuidFileName;

result.setUrl(realUrlPath);

} **catch** (Exception e) {

e.printStackTrace();

result.setError(1);

}

**return** result;

}

}

## Nginx实现图片回显

### 反向代理机制

业务需求:

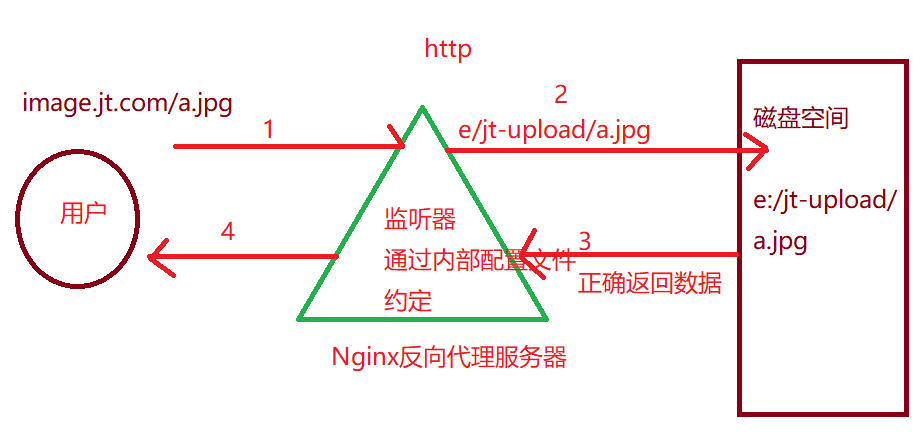
用户上传的图片和用户请求图片的网址有差别.

磁盘路径: **E:\jt-upload**\2018\11\02\abc.jpg

虚拟路径: [**http://image.jt.com**\2018\11\02\abc.jpg](http://image.jt.com\2018\11\02\abc.jpg)

如何将虚拟路径地址正确的映射到磁盘路径中???

### 反向代理说明



步骤:

1. 用户发起请求,被nginx中的监听器所拦截.
2. Nginx通过内部的配置文件,将用户的请求的路径进行中转.转化为数据真实的磁盘路径.根据路径发起http请求.
3. 从服务器中正确的获取数据后,返回给nginx
4. Nginx将返回的数据最终返回给用户.

**总结:**

**反向代理服务器代替用户发起请求,最终将请求结果返回给用户的过程.**

### Nginx介绍

*Nginx* (engine x) 是一个高性能的[HTTP](https://baike.baidu.com/item/HTTP" \t "_blank)和[反向代理](https://baike.baidu.com/item/%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86/7793488" \t "_blank)服务，也是一个IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为[俄罗斯](https://baike.baidu.com/item/%E4%BF%84%E7%BD%97%E6%96%AF/125568)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](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81)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](https://baike.baidu.com/item/%E9%97%BB%E5%90%8D/2303308)。2011年6月1日，nginx 1.0.4发布。

**Nginx是一款**[**轻量级**](https://baike.baidu.com/item/%E8%BD%BB%E9%87%8F%E7%BA%A7/10002835)**的**[**Web**](https://baike.baidu.com/item/Web/150564)**服务器/**[**反向代理**](https://baike.baidu.com/item/%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86/7793488)**服务器及**[**电子邮件**](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6/111106)**（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**](https://baike.baidu.com/item/%E5%B9%B6%E5%8F%91/11024806)**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](https://baike.baidu.com/item/%E4%BA%AC%E4%B8%9C/210931)、[新浪](https://baike.baidu.com/item/%E6%96%B0%E6%B5%AA/125692)、[网易](https://baike.baidu.com/item/%E7%BD%91%E6%98%93/185754)、[腾讯](https://baike.baidu.com/item/%E8%85%BE%E8%AE%AF/112204)、[淘宝](https://baike.baidu.com/item/%E6%B7%98%E5%AE%9D/145661)等。

占用内存:1-2M

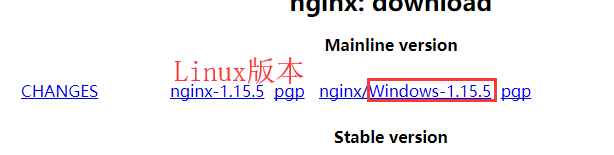
并发能力:5万/秒(C语言开发), 3万/秒

免费:不花钱

特点:nginx是软件负载均衡.

### Nginx下载

网址介绍: <http://nginx.org/>

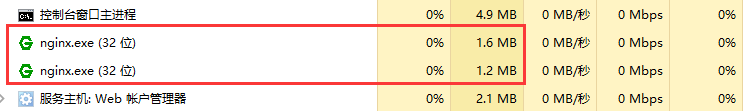


### Nginx安装

1.解压nginx文件

注意事项:1.不要放C盘 2.不要有中文路径

2.以管理员身份运行,检测进程项



主进程:主要提供反向代理和负载均衡服务的.

守护进程:防止主进程意外关闭.

所以关闭时,应该先关闭守护进程.

### Nginx命令

说明:执行nginx命令,必须在nginx.exe文件所在目录中执行.

1. 启动命令 start nginx
2. 停止命令 nginx -s stop
3. 重启命令 nginx -s reload

### 实现方向代理-入门案例

说明:当访问<http://localhost:80>时,默认跳转到nginx系统首页.

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

#root代表文件夹

root html;

#index 默认访问页面

index index.html index.htm;

}

}

### Nginx实现图片回显

#配置图片服务器

server {

listen 80;

server\_name image.jt.com;

location / {

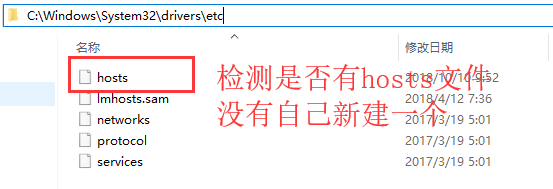
root E:\jt-upload;

}

}

修改完配置文件后,重启nginx

### 修改hosts文件



编辑hosts文件



将环境配置导入switch文件中.保存即可.

# 京淘电商环境

127.0.0.1 image.jt.com

127.0.0.1 manage.jt.com

127.0.0.1 www.jt.com

127.0.0.1 sso.jt.com

127.0.0.1 cart.jt.com

127.0.0.1 order.jt.com

127.0.0.1 solr.jt.com

192.168.126.148 [www.easymall.com](http://www.easymall.com)

### 实现反向代理-效果

