# 京淘项目第六天

## 文件上传

### 入门案例

#### 页面编辑

<body>

<h1>实现文件长传</h1>

<!--enctype="开启多媒体标签" -->

<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>

#### 入门案例

@Controller

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

/\*\*

\* 当用户上传完成时重定向到上传页面

\* 思路:

\* 1.获取用户文件信息 包含文件名称

\* 2.指定文件上传路径 if

\* 3.实现文件上传

\* **@param** fileImage

\* **@return**

\* **@throws** IOException

\* **@throws** IllegalStateException

\*/

@RequestMapping("/file")

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

//1.获取input标签中的name属性

String inputName = fileImage.getName();

System.***out***.println("1:"+inputName);

//2.获取文件名称

String fileName = fileImage.getOriginalFilename();

//3.定义文件夹路径

File fileDir = **new** File("D:/1-jt/image");

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

//创建文件夹

fileDir.mkdirs();

}

//4.实现文件上传

fileImage.transferTo(**new** File("D:/1-jt/image/"+fileName));

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

}

}

### 关于文件上传小结

1. 开启多媒体标签

enctype="multipart/form-data"

1. 使用工具API进行接收

MultipartFile 参数名称 必须与input标签中的name属性一致.

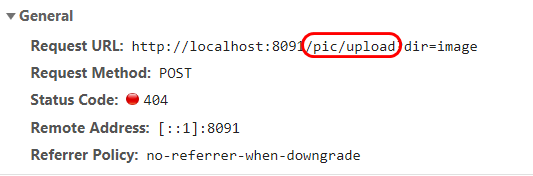
1. 实现文件上传

fileImage.transferTo(**new** File("D:/1-jt/image/"+fileName));

## 商品图片上传

### 页面分析

#### 页面url



Dir=image 表示上传为图片.

#### 页面JS

kingEditorParams : {

filePostName : "uploadFile",

uploadJson : '/pic/upload',

dir : "image"

},

#### 文件上传回显数据要求

{"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：

1.error 表示文件上传是否正确 0表示正常 1表示失败

2.url 该路径为虚拟的url路径.

3.width/height 宽和高

#### 定义VO对象

@Data

@Accessors(chain = **true**)

@NoArgsConstructor

@AllArgsConstructor

**public** **class** ImageVO **implements** Serializable{

**private** Integer error;

**private** String url;

**private** Integer width;

**private** Integer height;

}

### 编辑Controller

//实现文件上传

@RequestMapping("/pic/upload")

@ResponseBody

**public** ImageVO uploadFile(MultipartFile uploadFile) {

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

}

### 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。 |
| ^ | 匹配输入字行首。如果设置了RegExp对象的Multiline属性，^也匹配“\n”或“\r”之后的位置。 |
| $ | 匹配输入行尾。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{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”。?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地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+”将尽可能多地匹配“o”，得到结果[“oooo”]，而“o+?”将尽可能少地匹配“o”，得到结果 ['o', 'o', 'o', 'o'] |
| .点 | 匹配除“\n”和"\r"之外的任何单个字符。要匹配包括“\n”和"\r"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”。  \*python的正则表达式没有完全按照正则表达式规范实现，所以一些高级特性建议使用其他语言如java、scala等 |
| (?<!patte\_n) | 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”。  \*python的正则表达式没有完全按照正则表达式规范实现，所以一些高级特性建议使用其他语言如java、scala等 |
| x|y | 匹配x或y。例如，“z|food”能匹配“z”或“food”(此处请谨慎)。“[z|f]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”任一字符。 |
| [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 | 匹配一个单词的边界，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”；“\b1\_”可以匹配“1\_23”中的“1\_”，但不能匹配“21\_3”中的“1\_”。 |
| \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\_]”。 |
| \x*n* | 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*。 |
| \u*n* | 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。 |

### 编辑Service

@Service

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

//定义本地磁盘路径

**private** String localDirPath ="D:/1-jt/image/";

/\*\*

\* 1.获取图片名称

\* 2.校验是否为图片类型 jpg|png|gif

\* 3. 校验是否为恶意程序 木马.exe.jpg

\* 4.分文件保存 按照时间存储 yyyy/MM/dd

\* 5.防止文件重名. UUID 32位16进制数+毫秒数

\*

\* 正则常用字符:

\* 1.^ 标识..开始

\* 2.$ 标识以...结束

\* 3.点. 任意单个字符

\* 4.\* 表示任意个 0~~~~无穷

\* 5.+ 表示 1~~~~无穷

\* 6.\. 标识特殊字符.

\* 7.(xxx|xx|xxx) 代表分组 满足其中一个条件即可

\*/

@Override

**public** ImageVO updateFile(MultipartFile uploadFile) {

ImageVO imageVO = **new** ImageVO();

//1.获取图片名称 a.jpg A.JPG

String fileName = uploadFile.getOriginalFilename();

//将字符统一转化为小写

fileName = fileName.toLowerCase();

//2.校验图片类型 使用正则表达式判断字符串.

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

imageVO.setError(1); //表示上传有无

**return** imageVO;

}

//3.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) {

imageVO.setError(1);

**return** imageVO;

}

//4.时间转化为字符串 2019/5/31

String dateDir =

**new** SimpleDateFormat("yyyy/MM/dd")

.format(**new** Date());

//5.准备文件夹 D:/1-jt/image/yyyy/MM/dd

String localDir = localDirPath + dateDir;

File dirFile = **new** File(localDir);

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

//如果文件不存在,则创建文件夹

dirFile.mkdirs();

}

//b8a7ff05-8356-11e9-9997-e0d55e0fcfd8

//6.使用UUID定义文件名称 uuid.jpg

String uuid =

UUID.*randomUUID*().toString().replace("-","");

//图片类型 a.jpg 动态获取 ".jpg"

String fileType =

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

//拼接新的文件名称

//D:/1-jt/image/yyyy/MM/dd/文件名称.类型

String realLocalPath = localDir+"/"+uuid+fileType;

//7.2完成文件上传

uploadFile.transferTo(**new** File(realLocalPath));

//将文件文件信息回传给页面

imageVO.setError(0)

.setHeight(height)

.setWidth(width)

.setUrl("https://img14.360buyimg.com/n0/jfs/t1/32386/3/9341/287224/5ca5a2afEb0d96c09/b5315f160ca38265.jpg");

//暂时写死.后期维护.

} **catch** (Exception e) {

e.printStackTrace();

imageVO.setError(1);

**return** imageVO;

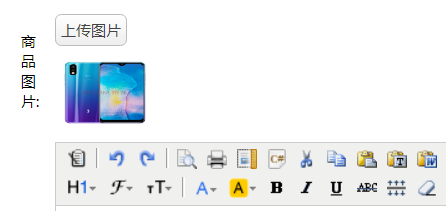
}

**return** imageVO;

}

}

### 页面效果



## 实现商品数据回显

### 需求说明

文件上传后,将数据保存到本地磁盘中.但是该图片只允许本机自己访问.如果需要通过浏览器访问.则必须设定虚拟访问地址.

### 拼接url路径

根据URL地址拼接虚拟路径.之后测试文件上传是否成功

//定义虚拟路径名称

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

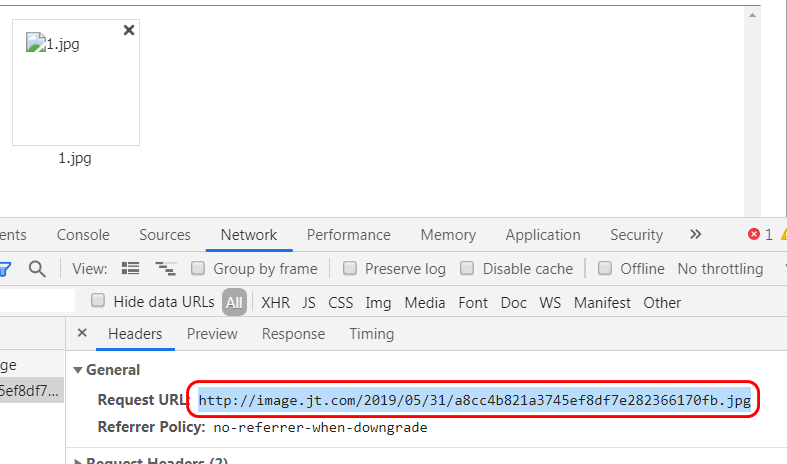
//8.拼接url路径 http://image.jt.com/yyyy/MM/dd/uuid.jpg

String realUrlPath =

urlPath+dateDir+"/"+uuid+fileType;

### 上传测试

1.上传文件之后显示路径.



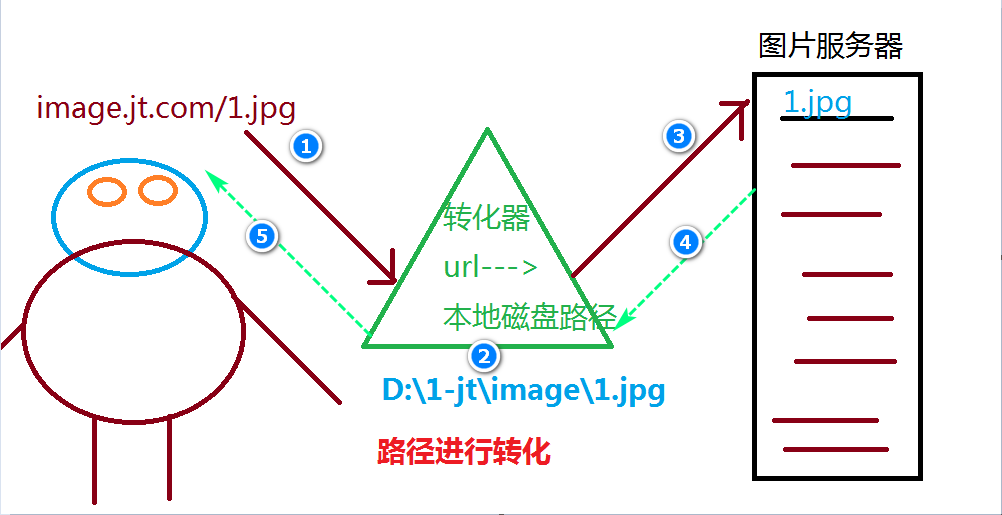
2.通过浏览器检查文件是否上传成功!!



### 图片回显策略

说明:

用户通过虚拟路径访问.之后利用转化的机制,将虚拟路径转化为本地磁盘路径.这时用户可以访问真实的图片信息.



### 反向代理

概念:

当用户发起请求时根据规则拦截用户的请求之后,**代替用户访问真实资源的过程**称之为反向代理.

## Nginx

### Nginx介绍

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，在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。

特点:

1.并发能力强 5万/秒---3万/秒 tomcat:220/秒—150/秒

并发能力受限于 网络/硬件条件.

2.轻量级 占用磁盘空间 不到2M

### Nginx官网介绍

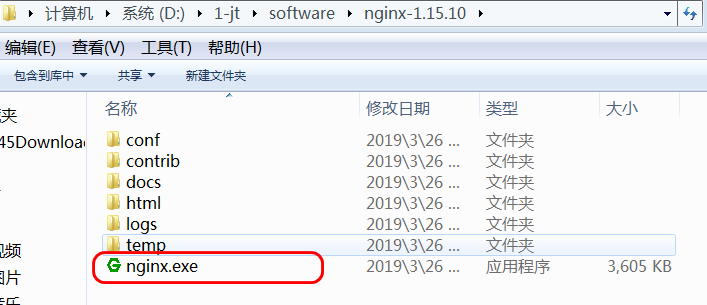
1. url地址: <http://nginx.org/>



### Nginx安装和使用

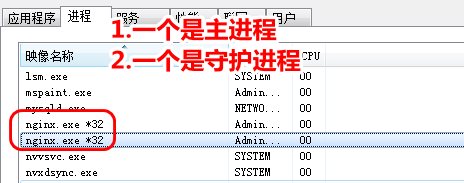
#### 安装Nginx

说明:将课前资料中的nginx文件导入本地.不要有中文/空格.之后解压.



#### Nginx启动

1. 使用超级管理员权限启动nginx.exe
2. 检查服务是否启动成功



守护进程作用:防止主进程意外关闭.

1. 关闭nginx

先关闭守护进程/再关闭主进程.

### Nginx启动异常问题

Nginx启动时会自动的占用80端口,如果计算机中80已经被其他进程使用.则nginx启动异常.

思路:关闭80端口服务 重启Nginx.

例外:如果占用端口的服务好PID是2,该问题由于驱动问题.建议根据网上教程自行恢复.

### Nginx命令

**前提:命令的执行必须在nginx.exe所在路径中执行**

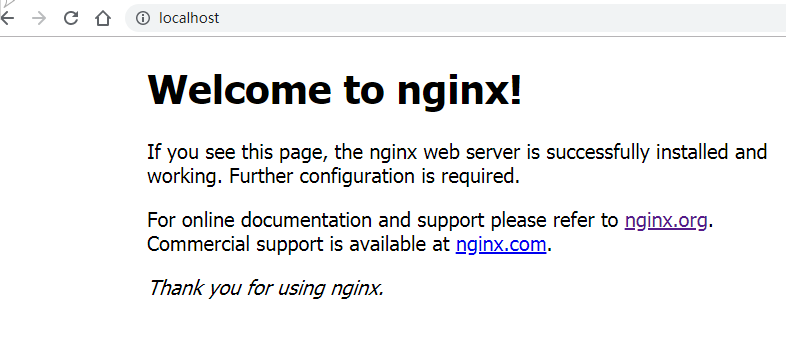
1.启动命令 start nginx

2.重启命令 nginx –s reload

3.关闭命令 nginx –s stop

### Nginx实现反向代理

Nginx:默认占用端口80 通过localhost:80访问跳转首页



#### Nginx配置文件介绍

#server代表一个服务项

server {

listen 80; #监听端口

#server\_name 表示拦截域名

server\_name localhost;

#拦截请求之后处理方式

location / {

#root 是关键字 代表反向代理的文件夹名称

root html;

#index 默认跳转页面

index index.html index.htm;

}

}

## 实现商品信息回显

### 业务说明

用户访问图片网址为:

<http://image.jt.com/2019/05/31/a8cc4b821a3745ef8df7e282366170fb.jpg> 需要进行反向代理.访问真实的磁盘路径

D:\1-jt\image\2019\05\31\d4625e93883b4c31a1389da0b06f66a5.jpg

### Nginx配置

#配置图片服务器

server {

listen 80;

server\_name image.jt.com;

location / {

root D:/1-jt/image;

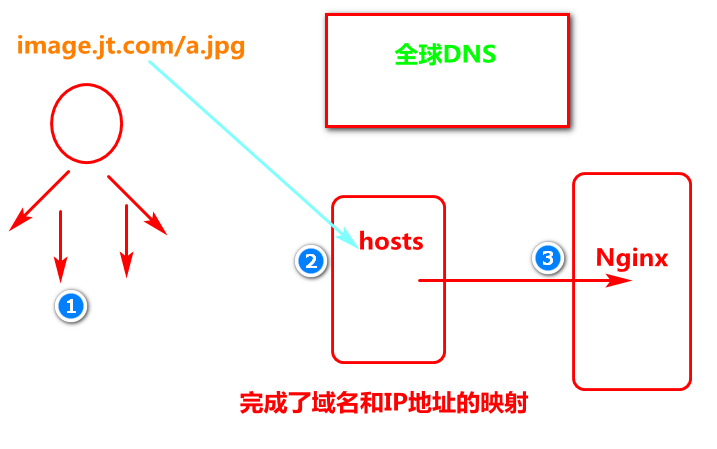
}

}

编辑完成后,重启nginx服务器

### 编辑HOSTS文件

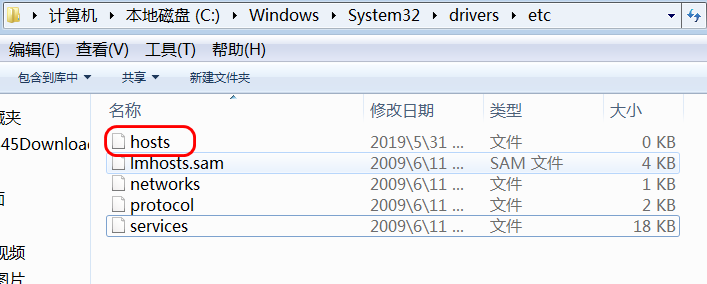
#### 图片访问流程



调用过程:

1. 用户发起http请求image.jt.com;
2. 修改hosts文件 将用户请求地址映射到本机中.
3. Nginx接收到用户请求之后进行反向代理最终实现数据访问.

#### 修改Hosts文件



修改如下:

127.0.0.1 localhost

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

#### 效果展现

当用户通过域名访问时.检查程序是否正确



## 实现域名代理

### 业务需求

说明:需要通过manage.jt.com访问localhost:8091服务器.

业务实现:

利用Nginx的反向代理技术实现.

### 配置nginx

#后台管理服务器 用户访问manage.jt.com时访问localhost:8091

server {

listen 80;

server\_name manage.jt.com;

location / {

#代理路径

proxy\_pass http://localhost:8091;

}

}

修改完成之后重启nginx

### 页面效果

