# 一.Maven基础

## 0.jar包

jar 是 Java ARchive 的缩写，是一种基本 zip 格式的文件格式。目标是将 java 生成的类文件、资源文件、管理文件等按照特定的结构打包成一个独立的文件，方便程序的发布或网络的下载。

## 1.什么是Maven

Maven:项目构建工具，打包工具

### （1）.Maven项目文件结构：

**src\**

**main\**

**java\ ----** 项目开发的java代码

**resources\ --**项目开发的资源文件夹

**test\**

**java\ ---**单元测试代码（开发人员自测）

**pom.xml --** 当前项目的配置文件

基于pom.xml配置，使用maven命令，完成项目构建工作

Maven常用配置

（1）.依赖配置：指定使用哪个依赖包

（2）.指定一些通用配置：jdk版本、编码等

Maven的作用

项目构建：项目依赖，项目打包

## 2.Maven的配置文件——pom.xml

**注意：更改完xml文件以后，一定要刷新一下** 

Xml简单的内容：标签，标签属性

标签的定义方式有以下两种：



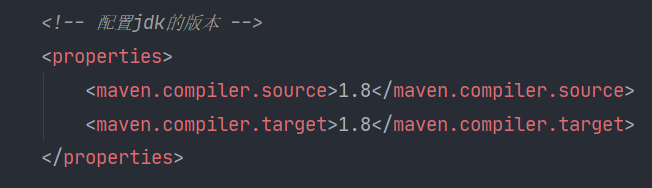
modelversion（pom的版本）之前的不用变



然后下面的groupId,artifactId,version对应的就是当前项目的组织号，产品号和版本号，我们自己修改



接下来就可以配置一些别的东西了，比如jdk的版本，依赖包等



## 3.依赖管理

（1）.maven仓库

maven 仓库是一个类似手机上的 App Store 的东西，上面会有全世界的人上传的各种第三方的 jar 包供我们使用，当我们的项目需要用到其中的内容，可以像手机安装 app 一样方便的进行查找和安装。

[maven 仓库](https://search.maven.org/)

其中我们的项目用到了另一个项目，叫做依赖关系。

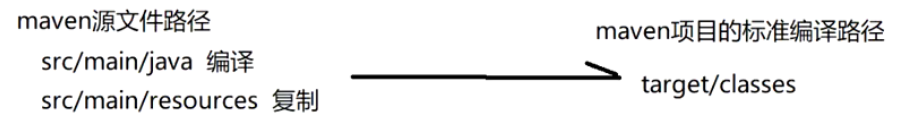
而一个项目中往往需要很多的依赖，所以诞生了依赖管理的概念。

（2）.在pom.xml文件中配置依赖



添加依赖的时候也要写明groupId,artifactId,version，如果在本地找不到，idea会自动开始下载

## 4.生命周期及其相关命令

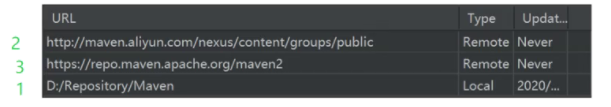


* clean:清楚target目录
* compile:编译当前文件，会在target目录下生成相关编译文件
* package:将当前项目中的类打成jar包供他人使用
* install:将target里面的jar包安装到本地仓库（了解）
* deploy:将target里面的jar包提交到公司的远程仓库给团队其他人员使用（了解）

## 4.Maven加载依赖包的顺序

Maven加载依赖包的顺序为：

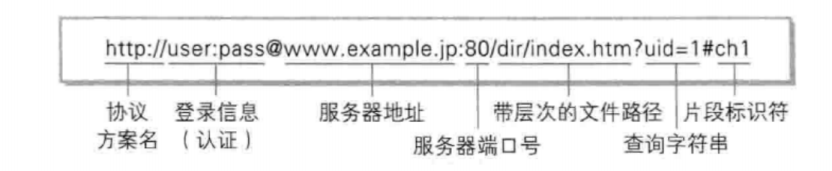
本地仓库——配置的远程仓库（阿里云）——官方远程仓库

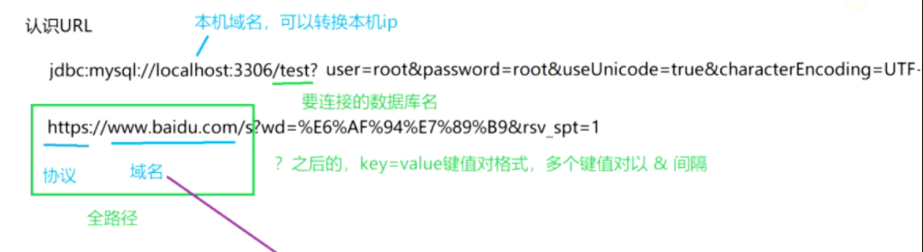


# 二.网络原理之Http

## 1.Http原理

### （1）.URL及协议





* url中的请求数据：请求路径?key1=value&key2=value2

1）.网络数据传输，都需要使用同样的协议，双方约定好的统一规范（封装和解析数据格式规范）

协议：数据格式的约定

目标：学习http协议里面的格式，结合理论实操，进行程序调试

http协议格式本身+http数据（数据中包含自己的数据格式）

2）.再次回顾URL

* 域名：基于DNS解析为IP
* IP:网络中定位主机的地址（逻辑地址）
* PORT:端口号，定位某个主机中唯一的进程（应用程序）
* http请求的访问路径：

服务路径定位资源

* + - url是全路径（绝对路径）
    - uri包含了url这种全路径，还包含相对路径

特殊的注意事项：输入域名进行访问，**其实是访问/这个路径的资源，/也有资源？**

浏览器中，不输入端口号，是因为http协议的默认端口是80

### （2）.urlencode和urldecode

像 / ? : 中文等这样的字符, 已经被url当做特殊意义理解了. 因此这些字符不能随意出现。

比如, 某个参数中需要带有这些特殊字符, 就必须先对特殊字符进行转义.

转义的规则如下:

将需要转码的字符转为16进制，然后从右到左，取4位(不足4位直接处理)，每2位做一位，前面加上%，编码成%XY格式

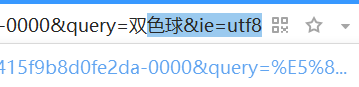


"+" 被转义成了 "%2B"

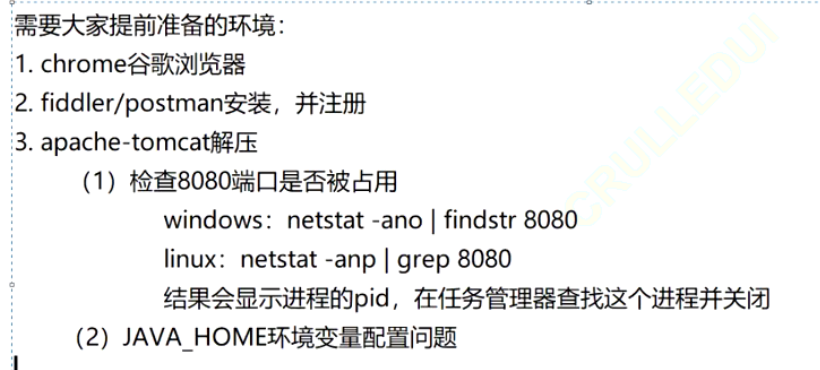
urldecode就是urlencode的逆过程

[urlencode工具](http://tool.chinaz.com/Tools/urlencode.aspx)

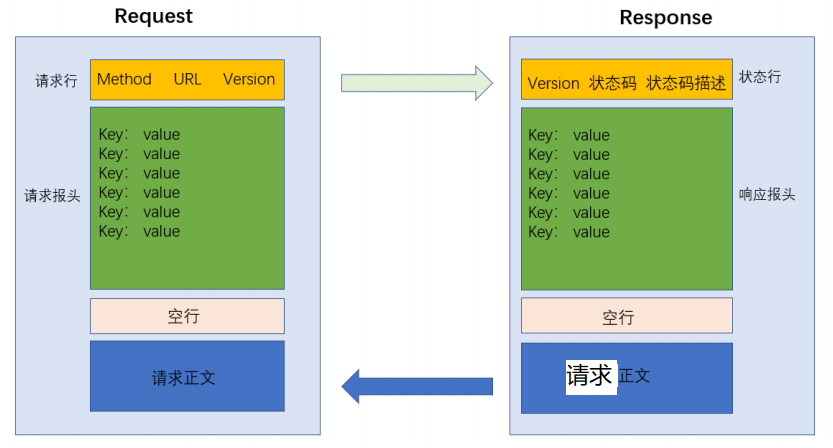
**注：有时候显示的是中文，但是实际上都是%XY的形式的**



* 所以，项目路径尽量避免中文



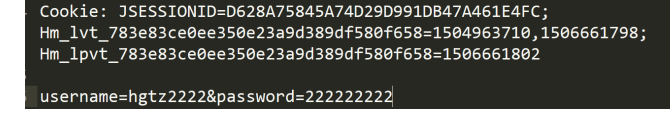
### （3）.HTTP协议格式



HTTP请求

* 首行: [方法] + [url] + [版本]
* Header: 请求的属性, 冒号分割的键值对;每组属性之间使用\n分隔;遇到空行表示Header部分结束
* Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个ContentLength属性来标识Body的长度;

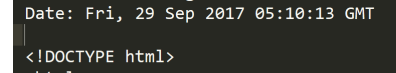




HTTP响应

* 首行: [版本号] + [状态码] + [状态码解释]
* Header: 请求的属性, 冒号分割的键值对;每组属性之间使用\n分隔;遇到空行表示Header部分结束
* Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个ContentLength属性来标识Body的长度; 如果服务器返回了一个html页面, 那么html页面内容就是在body中.





常见的Http工具

* 抓包工具：fiddler,charles,浏览器自带的开发者工具
* http请求模拟工具：postman

### （4）.HTTP的请求方法（get和post）

### \*\*\*\*Q:Get和Post的区别？

* 1.请求形式

get的请求数据只能放在url中，post的数据可以放在url和请求体中

* 2.传输数据的大小

get方法传输的数据量会受到限制，虽然在HTTP协议中没有硬性限制，但是特定的服务器和浏览器对url的长度有限制，而get是通过url传输数据的，所以会因此受到限制，一般传输数据量不超过2KB

post方法理论上没有限制，实际上post能传输的数据量取决于服务器的设置和内存大小

总的来说，get的传输数据量会受到限制，post的数据量则可以很大

* 3.传输数据的类型

get方法只允许传输ascii码字符（就是想要传中文要转码），post方法则无限制，也允许二进制数据

* 4.安全性

get传输的参数安全性低，因为数据会显示在url中，所有人可见，而且会被缓存和保留在浏览器历史中；post的数据则不会显示在url中，并且参数不会被缓存和保存在浏览器历史或者web服务日志中。

**什么时候用get，什么时候用post呢？**

1.有安全性要求的话，用post

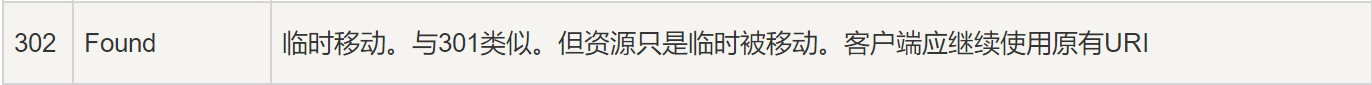
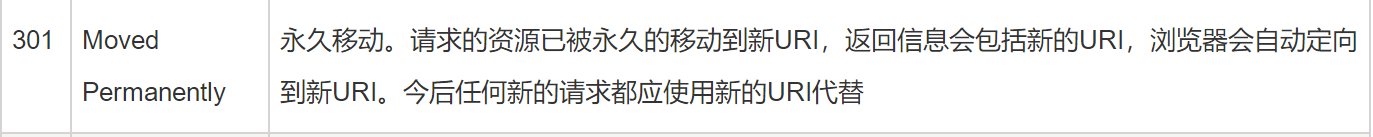
2.如果有中文，用post好一点，用get由于只能传输ascii码，还要进行编码和解码的操作

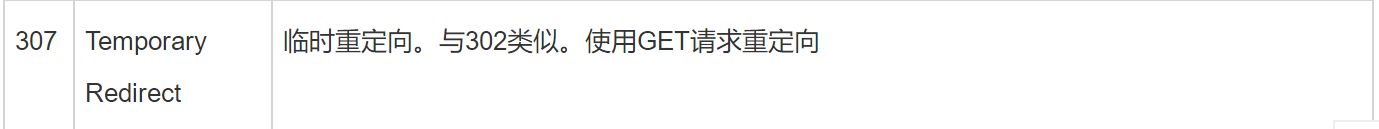
### （5）.HTTP的状态码

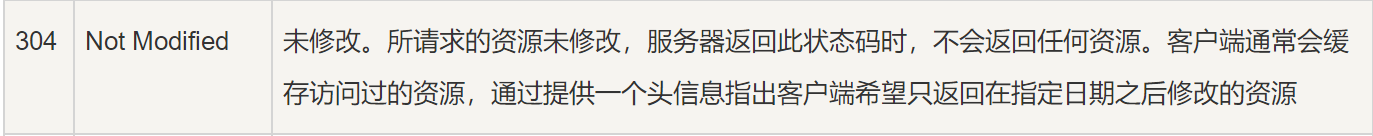
* 这些状态码是服务端返回给用户端状态码，是服务端自己设置的，这些状态码的含义都是站在服务端的角色上看的，在用户端的角色上就不一定。比如200在服务端是正确的，但是客户端受到的数据不一定是他们想要的正确的数据

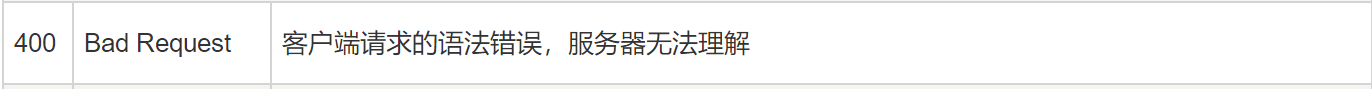




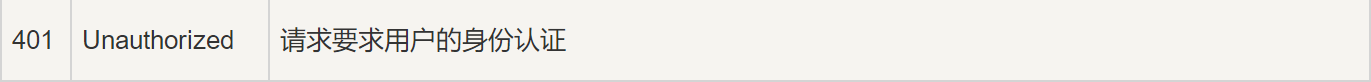


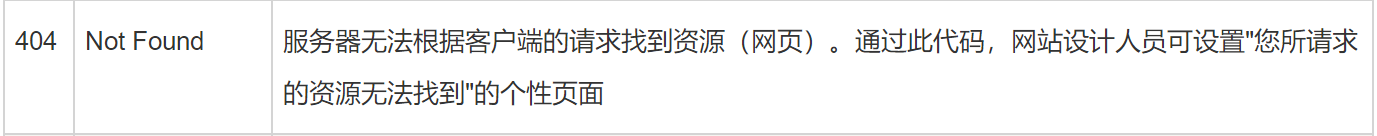






* 400错误包括http协议格式，请求数据的格式，数据类型等错误

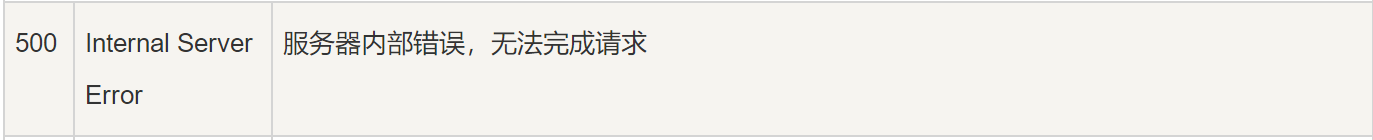




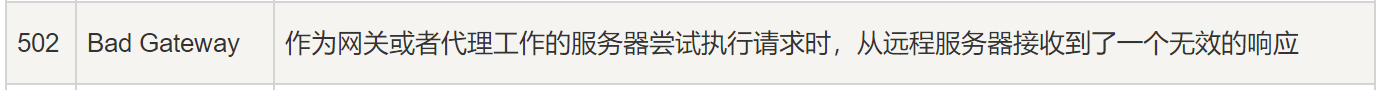
* 服务端通过uri和（资源/服务）建立映射关系**（大小写敏感）**，根据uri去找，找不到



* uri已经映射到了，但是服务端提供的请求方法（如get,post方法），不包含客户端的请求的方法（如来了一个poooo方法）



* 一般是服务端抛异常的时候



### （6）.HTTP的头信息（Header）

* Content-Type: 数据类型(text/html等)——（消息体body的格式，告诉对方该用什么方式解析）
* Content-Length: Body的长度——（消息体的长度）
* Host: 客户端告知服务器, 所请求的资源是在哪个主机的哪个端口上;
* User-Agent: 声明用户的浏览器版本信息——（一般在请求体中，告诉服务端我客户端用的什么浏览器）
* referer: 当前页面是从哪个页面跳转过来的;（在请求体中，告诉服务器当前页面是从哪里跳转过来的）
* location: 搭配3xx状态码使用, 告诉客户端接下来要去哪里访问;
* Cookie: 用于在客户端存储少量信息. 通常用于实现会话(session)的功能;、

## 2.前端开发

### （1）.前端三剑客：

（1）.html前端展示控件（页面内容）的资源文件

（2）.javascript：前端的脚本语言，写代码完成功能

（3）.css:前端样式文件

* javascript和css可以是单独的资源文件，也可以在html里边直接写

### （2）.tomcat简单使用

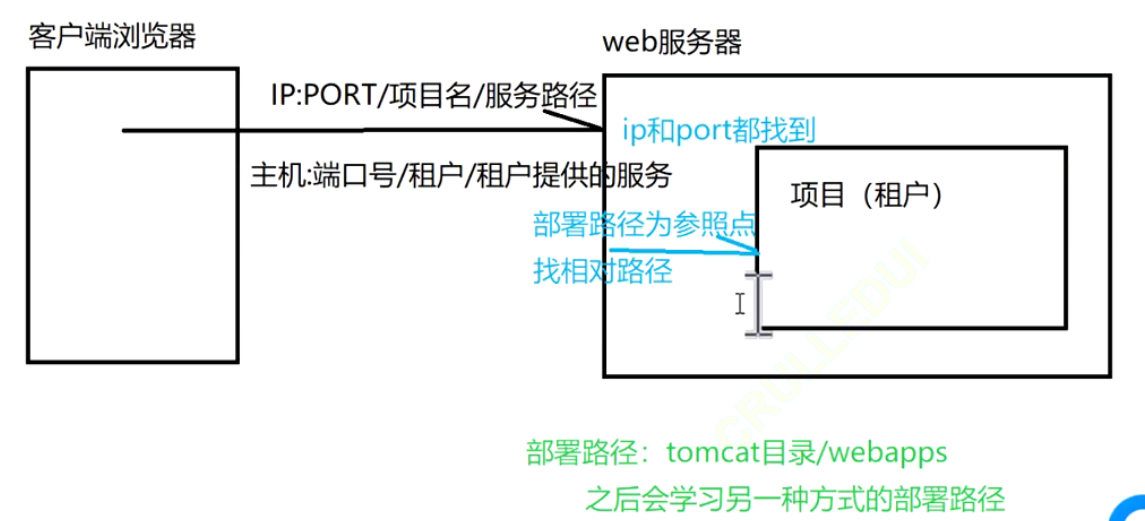
|---bin

|--startup.bat/shutdown.bat（启动、关闭的脚本）

|--webapps 部署的路径（有一栋房子，要招租提供服务）

|--项目（租户的名称，提供的各种服务）

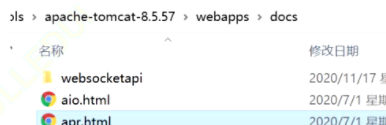
* 注意如果在localhost:8080后面不加上webapps里面的项目文件名的话，默认访问的是webapps下的ROOT文件中的内容。



这种情况下，访问对应的项目服务方法为

[http://localhost:8080/docs/apr.html](http://localhost:8080/docs/apr.html（）)

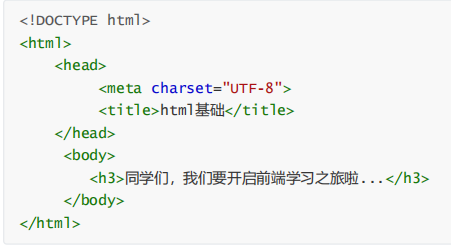
因为8080后面的内容为以部署路径（webapps）开始的相对路径



### （3）.html介绍

1.HTML代表超文本标记语言。HTML是一种标记语言，它是标记标签的集合。

2.HTML文档结构至少要包括head, body两部分.如:



3.HTML标签是由尖括号括起来的词，如 <html>， <body> 。

标签通常成对出现，例如 <html>和</html> 。也可以单个出现，如<br/>（两种写法）

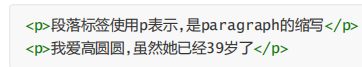


### （4）.html常见标签

#### 文本标签（段落+标题）

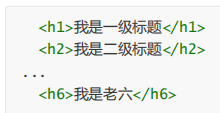
1）.段落标签

段落标签使用p标签，是paragraph的缩写，自带换行效果。如：



2）.标题标签

标题标签h1 ~ h6 标题标签从名字就能看出,是用来定义文字标题的,包括h1-h6,数字越小对应的字体越大，如下:



#### 表单标签（文本，密码，按钮，表单）

1）.文本框

单行的文本使用input标签，input标签有很多属性，如下：

* type ，表示文本的类型；
* name，表示文本的名称,后端使用name来获取框中的属性值；
* value，默认属性框的填充值，用户输入后显示输入的内容
* placeholder，文本框内容为空显示的内容（类似水印“请输入密码”）；
* size，文本框的长度



2）.密码框

密码框与文本框类似 , 区别在于type取值不同, 取值为password, 输入时候自动显示为星号,如:



3）. 普通按钮，一般按钮，如下：



4）. 提交按钮

当需要提交表单的时候，需要使用提交按钮。提交按钮需要配合form表单才能向服务器提交数据。如下：



5）. 表单

使用form来表示,表示提交到的服务器的信息,一般在注册或者登陆两个应用场景中使用 如下:

<form method="post" **action="http://how2j.cn/study/login.jsp"**>

账号：<input type="text" name="name">

**<br/>**

密码：<input type="password" name="password" >

**<br/>**

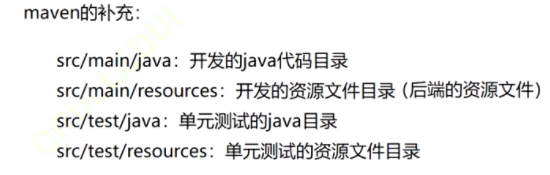
<input type="submit" value="登陆">

</form>

* 这两个<br/表示换行>
* action=请求路径（相对路径和绝对路径都可以，相对路径的话以当前html文件所在的文件夹为根目录的路径）

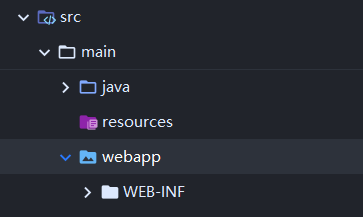
### （5）.Maven配置web项目

1）.如何进行配置



现在我们要配置一个web项目，那就会有前端的一些文件，但是maven的src/main/resources里面存放的文件是**后端的资源文件**，为此我们要建立一个文件夹存放**前端的资源文件**。

参考开发环境配置中的web项目配置，配置完成后，在src/main/下会多一个webapp的文件夹，用于存放前端的资源文件



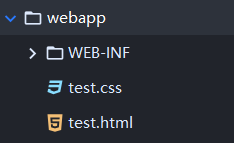
### （6）.写前端界面并运行（html,css,javascript）

##### 0）.所有代码在idea里面都有（maven\_test项目）

[菜鸟教程nb坏了](https://www.runoob.com/)，可以在上面查看一些前端和后端的一些知识

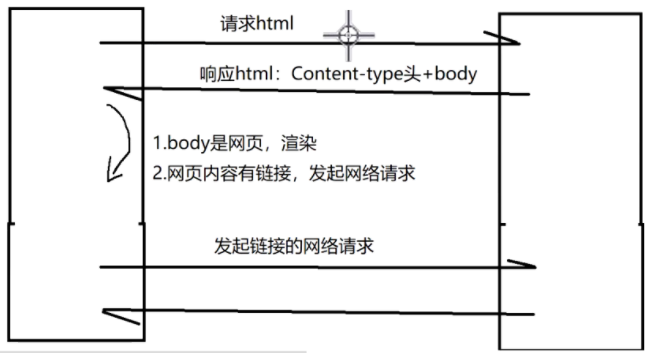
##### 1）.html+css

前端的文件都要放在webapp目录下，与WEB-INF目录同级，不能放在WEB-INF目录里面





##### 2）.理解网络传输

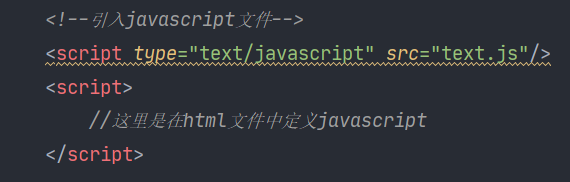


* 1.客户端（浏览器）发起对应的html文件请求，
* 2.服务器响应请求，返回content-type头和body（网页内容）；
* 3.客户端收到响应后，发现是网页开始渲染，发现网页内容里面有链接，于是再次发起该链接的网络请求；
* 4.服务端收到请求后，再次响应，将链接内的内容返回给客户端

##### 3）.javascript（简称js）

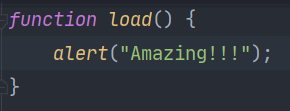
1.JavaScript是一种基于对象和事件驱动的客户端脚本语言

* 可以引入javascript文件
* 也可以直接在html中写javascript

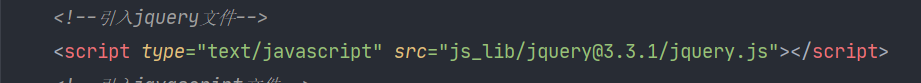


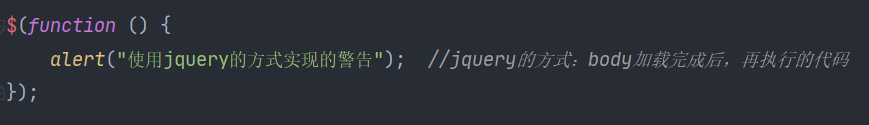
2.在body处使用了load函数，该函数在js文件中实现，能弹出提示框提示相应内容，这个是在body内容加载前执行





3.还可以引入jquery的js文件，这样在body加载完的时候，会自动执行function中的函数，不需要像前面那个那样在body处调用load函数了





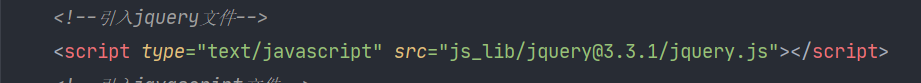
##### 4）（重要）jquery和ajax

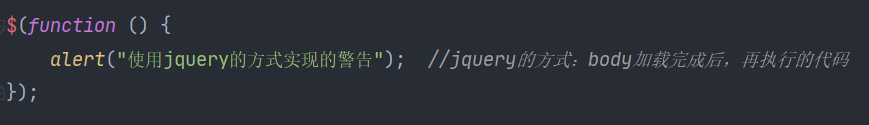
###### 1.三者联系

* JS是一门 前端语言。
* Ajax是一门 技术，它提供了异步更新的机制，使用客户端与服务器间交换数据而非整个页面文档，实现页面的局部更新。
* jQuery是一个 框架，它对JS进行了封装，使其更方便使用。jQuery使得JS与Ajax的使用更方便

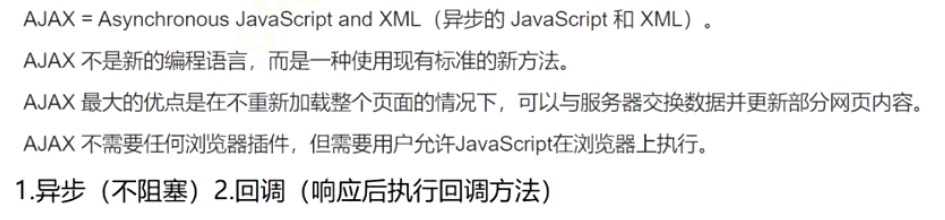
###### 2.[jquery的中文文档](https://jquery.cuishifeng.cn/)

可以引入jquery的js文件，这样在body加载完的时候，会自动执行function中的函数，不需要像前面那个那样在body处调用load函数了



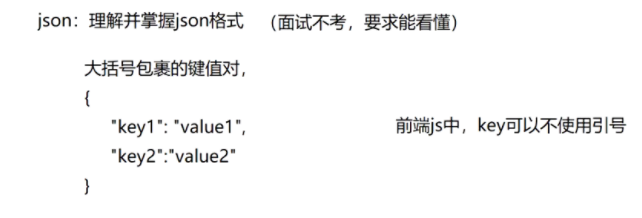


###### \*\*\*\*Q3.ajax：理解ajax异步请求（面试可能考）



###### 4.理解掌握json格式

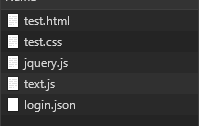
在$.ajax方法中是一个json对象，对象的内容用键值对的方式去写



后端的json对象里面的key还是要加上双引号的

###### 5.（重要）数据传输整个流程分析：

具体代码见idea

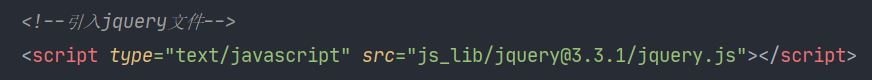


1.客户端请求html文件，服务端响应，客户端得到test.html

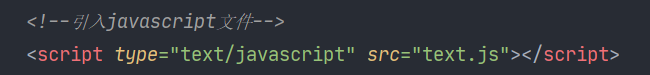
2.客户端得知是html网页开始渲染，发现有链接，固再次发起请求，服务端响应，客户端得到test.css



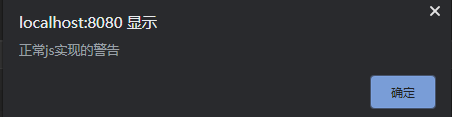
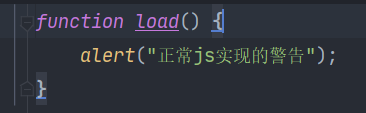
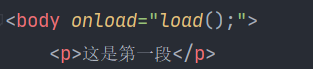
3.客户端发现有链接，继续请求，服务端响应，客户端得到jquery.js



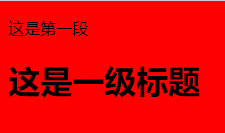
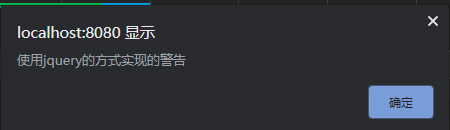
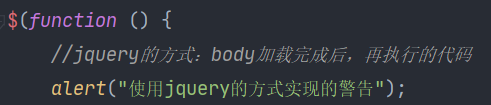
4.客户端发现有链接，继续请求，服务端响应，客户端得到test.js



5.html文件的head部分加载完成，开始加载body部分，遇到onload函数，于是在test.js里面调用函数执行，弹出第一个提示框，注意此时还没有加载出body的内容



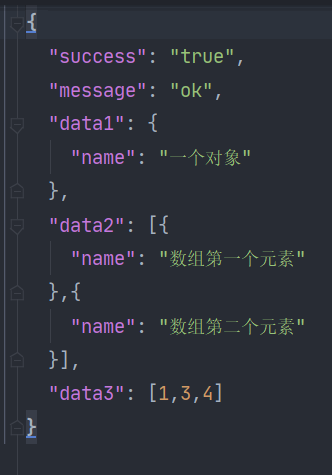
6.body加载完成，开始执行test.js里面的剩余部分（$();里面的内容），遇到alert弹出第二个提示框，注意此时body内容已经加载出来了



7.继续像下执行执行ajax里面的内容，发现链接，客户端发起请求，服务端响应，客户端得到login.json文件



8（重要）.响应成功，继续执行ajax方法中的内容，**响应成功后，会将中的login.json作为一个json对象回调给function里面的r，可以使用r.对象名来调用**,然后弹出第三个提示框，结束



json文件里面可以定义各种对象，数组等，{}表示对象，[]表示数组

###### 6.需要掌握的技能

1.抓包：会抓包查看网络数据传输（请求，响应数据包，包括任何内容）

2.部署：项目完成后部署到tomcat目录/webapps（ROOT不带项目名，非ROOT都带项目名）

专业部署方式：使用maven package命令，打一个war包，复制剪切到tomcat目录/webapps下，tomcat启动的时候，它会自动解压

3.访问：一定要通过http请求来访问，不要使用本地文件方式打开

4.要理解网络数据传输的流程（见前面第5点）

## 3.初识Servlet

### （1）.tomcat补充

##### 1）.tomcat启动报错的时候，检查

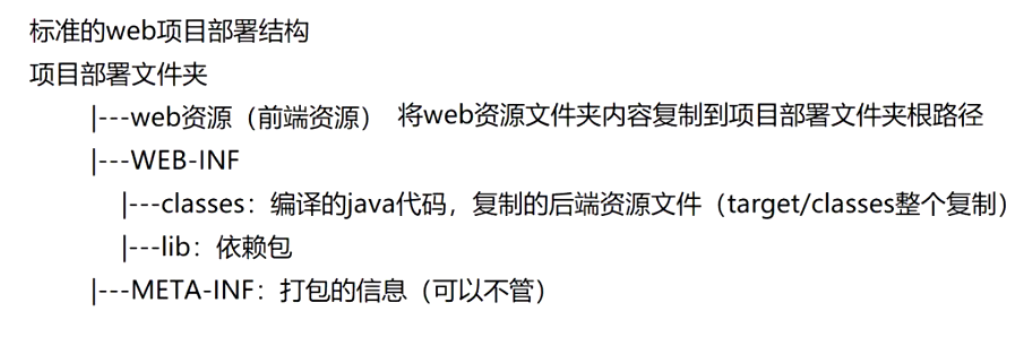
* 1.8080端口号是否被占用（netstat -ano|findstr 8080）
* 2.JAVA\_HOME是否配置出错（显然基本上不会）
* 3.tomcat目录/log/catalina.out，有时候是带日期的

##### 2）.tomcat部署web项目：

（1）.tomcat目录/webapps下，将项目复制进去，项目名就是文件夹名称（如果把war包放进去，tomcat启动后会自动解压）

（2）.docBase指定外部目录作为项目部署路径

* 标准的web项目标准结构（和开发的时候不一样）:



再次理解，项目构建工具，如maven的使用（从开发环境到部署环境的构建可配置化，所以如果发现前端界面出问题了，优先去target目录（部署环境）下找问题）

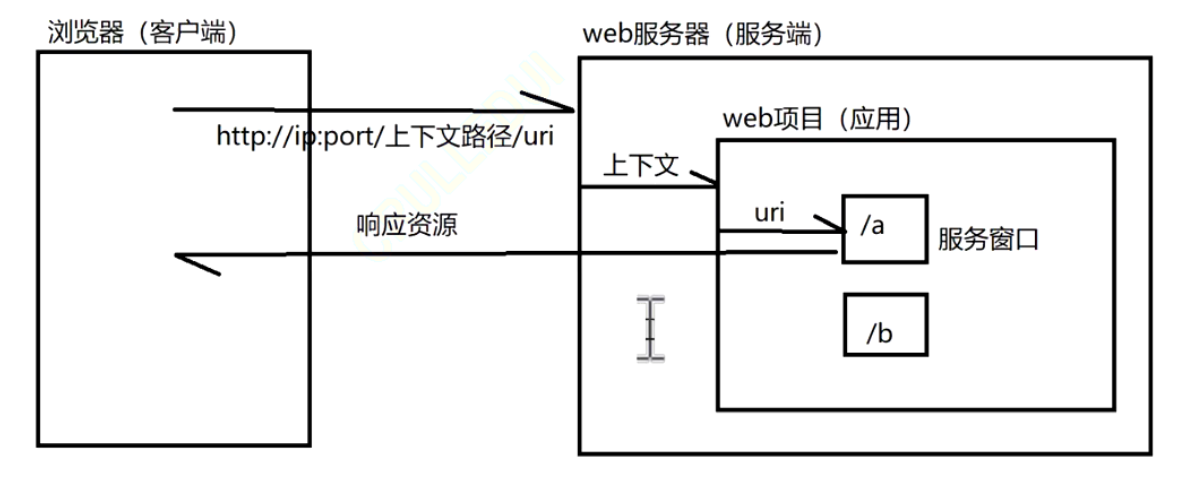
* idea中配置tomcat并运行（docBase外部目录部署的方式）

1.配置时注意是war explored，表示使用文件夹来部署

2.Deployment中Application Context表示应用上下文/项目部署名

3.访问也要使用Application Context

### （2）.（流程分析）uri查找的过程和映射的资源理解



* URL查找的过程：

ip对应主机得地址

port对应进程

应用文路径对应项目

之后的uri，对应项目中的资源

* uri映射资源：

（1）.资源包括静态资源文件

（2）.还包括程序提供的资源（Servlet的路径及http请求相应资源）

### （3）.写一个servlet

##### 1）.注解的使用：

小括号包括对个属性，属性名=属性值，多个之间逗号间隔，属性名为value时可以缺省

原来应该是这样



因为我们这里只需要用value,所以可以缺省

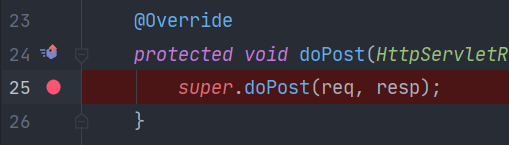


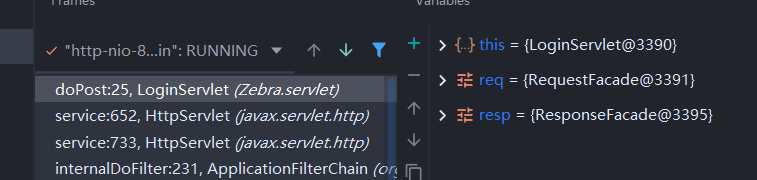
##### 2）.Servlet注意事项

1.Servlet定义服务：注意服务路径必须是/开始，否则tomcat启动就会报错

2.浏览器输入url的访问请求默认是get请求，想要对指定url发起post请求可以自己在文件中指定，也可以使用postman来发起

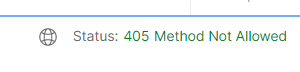
**3.servlet代码同样可以debug（要多用）**，打上断点，用debug的方式运行（平时也推荐一直用debug的方式运行），然后postman发起了post请求（即使是post,还是会出错）





可以看到方法栈里面有doPost方法，说明是成功进入这个方法了

4.为什么使用了post方法请求还是返回405呢？

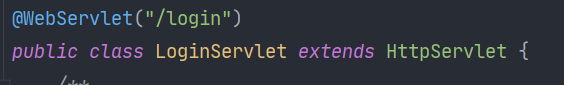


因为我们调用的是super.doPost方法，而父类的doPost方法是直接抛出一个405错误。（其实是内部的tomcat在里面在用的Servlet和官方的Servlet不统一导致的）

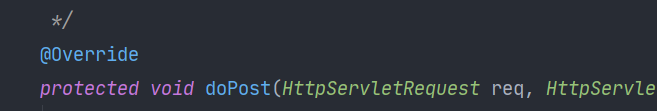
同样的，如果没有重写，如果发生了对应的请求，会自动调用父类的方法，抛出405错误。

##### 3）怎么写servlet

1.写上WebServlet注解的value值**（这个是服务路径，通过这个来定位到该Servlet对象）**，然后让Servlet类继承HttpServlet类



2.重写相应的请求方法

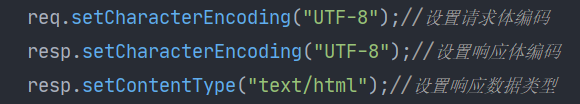


\* 每次http请求映射到某个Servlet的资源路径，都会调用service生命周期方法

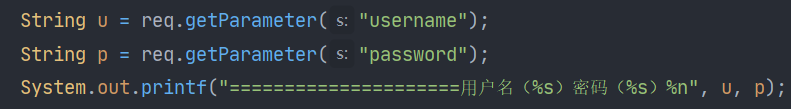
\* 如果请求方法没有重写，就调用父类的doXXX（对应请求方法），返回405

\* 如果重写，会将请求数据包装为一个Request请求对象，这时候虽然还没有响应，但是也包装了一个Response响应对象

3.设置请求，响应编码，及响应数据类型（为响应体设置Content-Type数据类型）

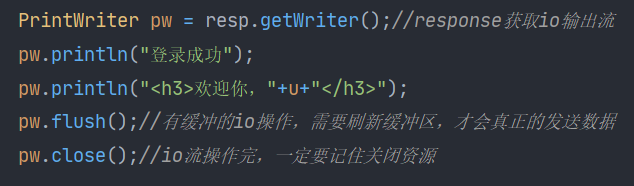


4.解析数据，request.getParameter方法获取请求数据：在url和请求体（body）内获取数据，数据格式为k1=v1&k2=v2（这种键值对的数据都可以获取）

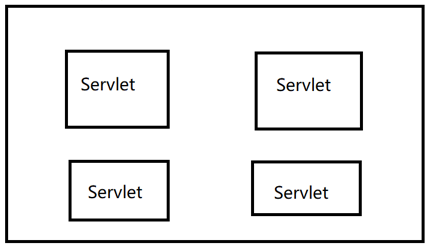
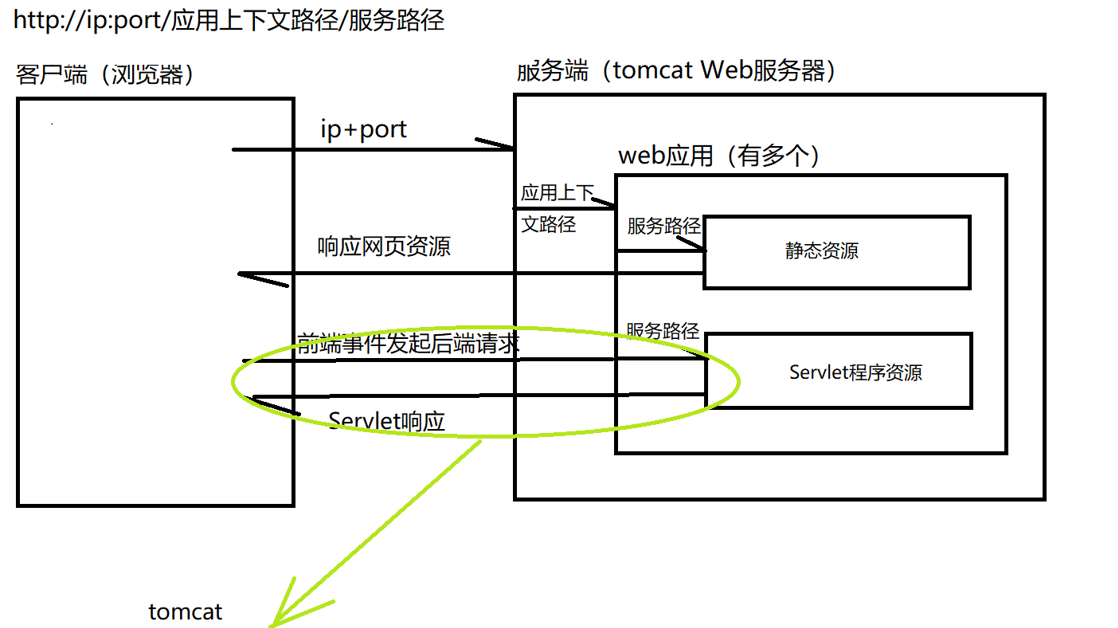


5.返回响应数据

获取io输出流，输出内容，刷新缓冲区，关闭资源



##### 4）（重要）servlet服务执行流程分析



* **总体流程分析**

1.客户端通过url请求响应服务，通过ip定位主机，port定位端口号，应用上下文路径定位web应用，然后通过剩下的uri（服务路径）定位静态网页资源。

2.服务端响应网页资源

3.客户端受到网页资源，网页初始化的时候，或者点击某个按钮，**发生了某个事件之后**（网页中我们可以自己写代码来调用Servlet服务）要调用后端的servlet服务，于是向后端发起Servlet请求（也是通过ip port 上下文路径 服务路径的方式进行请求）

4.服务端响应Servlet请求

* Servlet请求响应分析

tomcat启动：

通过一些机制，实例化Servlet对象：

1.产生服务路径（资源路径）和Servlet对象之间的映射关系

2.每次http请求，通过上下文路径+服务路径，查找到唯一的Servlet对象

3.调用Servlet对象的service方法

4.service方法中，将doXXX方法组织为一个模板方法，我们可以重写对应的方法，如果不重写，默认执行父类的方法，抛出405异常



### （4）.写一个网页\_Servlet处理前端数据，登录，页面跳转

##### 1）.注意点

1.页面跳转逻辑

index-首页，可以点击登录跳转到login，也可以点击主页跳转到index

2.

<hr>分割线

<br>换行

这两个标签都可以不加/结束

3.页面跳转使用A标签,<a herf=””>跳转<a/>

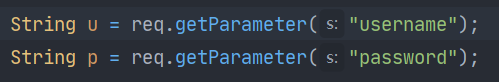
##### 2）.html界面（/\_前端action和后端Web注解）

**1.这里前端html文件里面action后面的login前面不用加/，但是在后端servlet文件中的注解后面必须加/**





2.前端的name=username表示：在当前文本框输入的值value和名为username的这键相对应，（username,Zebra）这一键值对放在请求体里，传送到后端；后端从请求体里面取出相应的数据



##### 3）.Servlet实现重定向跳转页面

1.这里核实用户民和密码时，字符串在前，如果变量在前可能会出现空指针异常（没有输入用户名或者密码的情况）

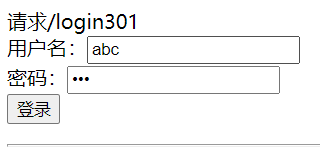
2.重定向使用resp.sendRedirect方法

3.要是密码错误，打印出错误信息

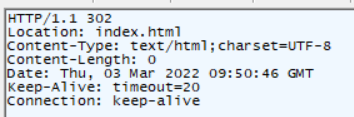


##### 4）.（重要）流程分析\_重定向

1. 当我们输入了正确的用户名和密码以后，客户端向服务端发起请求，请求login的servlet服务。



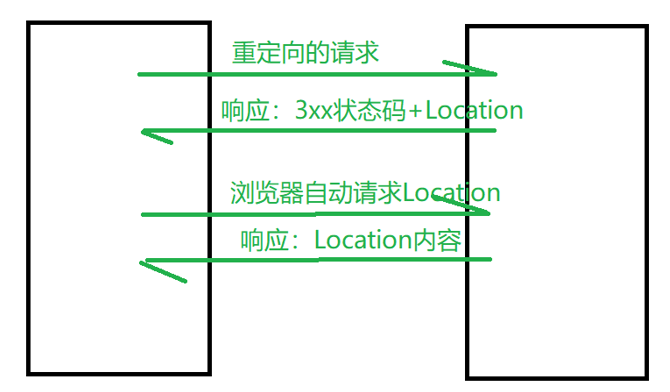
2. 服务端响应客户端，给了一个302状态码告诉浏览器要重定向，然后给了一个location让浏览器去跳转（所以resp.sendRedirect的实质作用是将location作为请求头的内容，没有响应体，然后返回给客户端了）

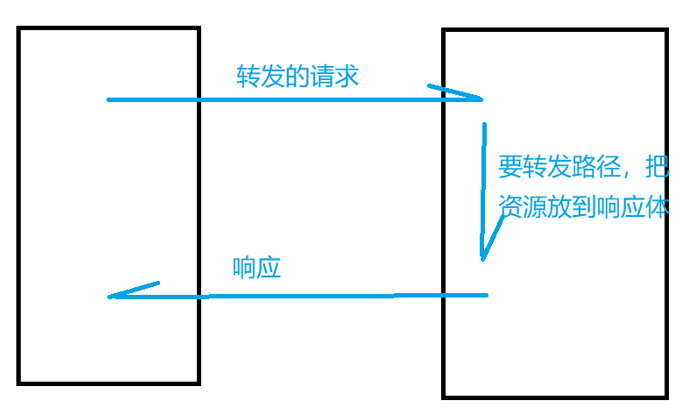




3. 客户端收到了响应，302状态码和跳转的地址location（index.html），固再次像后端发起请求，请求index.html文件

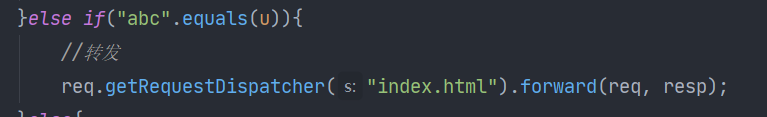
4. 服务端响应客户端的请求，状态码为200





##### 5）.Servlet实现转发跳转界面

转发：把转发的路径的资源作为响应体返回，注意转发是req.重定向是resp.



##### \*\*\*\*Q转发和重定向的区别

* 先说一下概念：

1. 重定向：返回3xx状态码+Location响应头，表示要跳转的路径，浏览器接收到响应数据后自动发起请求然后跳转

2. 转发：一次请求，后端直接把转发的路径的资源作为响应体返回

* 然后说一下区别:  
  （1）.转发不会改变url，重定向会

（2）.转发只需要发起一次请求，重定向要发起两次请求

（3）.转发是在服务器端发生的跳转，重定向是在客户端发生的跳转

//MY：转发是**req**.getRequestDispatcher("index.html").forward(req, resp)

// 重定向是**resp**.sendRedirect()