## 1.[HTTP协议状态码详解（HTTP Status Code）](http://www.cnblogs.com/shanyou/archive/2012/05/06/2486134.html)

1xx（临时响应）   
表示临时响应并需要请求者继续执行操作的状态代码。

代码   说明   
100   （继续） 请求者应当继续提出请求。 服务器返回此代码表示已收到请求的第一部分，正在等待其余部分。    
101   （切换协议） 请求者已要求服务器切换协议，服务器已确认并准备切换。

2xx （成功）   
表示成功处理了请求的状态代码。

代码   说明   
200   （成功）  服务器已成功处理了请求。 通常，这表示服务器提供了请求的网页。   
201   （已创建）  请求成功并且服务器创建了新的资源。   
202   （已接受）  服务器已接受请求，但尚未处理。   
203   （非授权信息）  服务器已成功处理了请求，但返回的信息可能来自另一来源。   
204   （无内容）  服务器成功处理了请求，但没有返回任何内容。   
205   （重置内容） 服务器成功处理了请求，但没有返回任何内容。   
206   （部分内容）  服务器成功处理了部分 GET 请求。

3xx （重定向）   
表示要完成请求，需要进一步操作。 通常，这些状态代码用来重定向。

代码   说明   
300   （多种选择）  针对请求，服务器可执行多种操作。 服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。   
301   （永久移动）  请求的网页已永久移动到新位置。 服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。   
302   （临时移动）  服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。   
303   （查看其他位置） 请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。   
304   （未修改） 自从上次请求后，请求的网页未修改过。 服务器返回此响应时，不会返回网页内容。   
305   （使用代理） 请求者只能使用代理访问请求的网页。 如果服务器返回此响应，还表示请求者应使用代理。   
307   （临时重定向）  服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

4xx（请求错误）   
这些状态代码表示请求可能出错，妨碍了服务器的处理。

代码   说明   
400   （错误请求） 服务器不理解请求的语法。   
401   （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。   
403   （禁止） 服务器拒绝请求。   
404   （未找到） 服务器找不到请求的网页。   
405   （方法禁用） 禁用请求中指定的方法。   
406   （不接受） 无法使用请求的内容特性响应请求的网页。   
407   （需要代理授权） 此状态代码与 401（未授权）类似，但指定请求者应当授权使用代理。   
408   （请求超时）  服务器等候请求时发生超时。   
409   （冲突）  服务器在完成请求时发生冲突。 服务器必须在响应中包含有关冲突的信息。   
410   （已删除）  如果请求的资源已永久删除，服务器就会返回此响应。   
411   （需要有效长度） 服务器不接受不含有效内容长度标头字段的请求。   
412   （未满足前提条件） 服务器未满足请求者在请求中设置的其中一个前提条件。   
413   （请求实体过大） 服务器无法处理请求，因为请求实体过大，超出服务器的处理能力。   
414   （请求的 URI 过长） 请求的 URI（通常为网址）过长，服务器无法处理。   
415   （不支持的媒体类型） 请求的格式不受请求页面的支持。   
416   （请求范围不符合要求） 如果页面无法提供请求的范围，则服务器会返回此状态代码。   
417   （未满足期望值） 服务器未满足”期望”请求标头字段的要求。

5xx（服务器错误）   
这些状态代码表示服务器在尝试处理请求时发生内部错误。 这些错误可能是服务器本身的错误，而不是请求出错。

代码   说明   
500   （服务器内部错误）  服务器遇到错误，无法完成请求。   
501   （尚未实施） 服务器不具备完成请求的功能。 例如，服务器无法识别请求方法时可能会返回此代码。   
502   （错误网关） 服务器作为网关或代理，从上游服务器收到无效响应。   
503   （服务不可用） 服务器目前无法使用（由于超载或停机维护）。 通常，这只是暂时状态。   
504   （网关超时）  服务器作为网关或代理，但是没有及时从上游服务器收到请求。   
505   （HTTP 版本不受支持） 服务器不支持请求中所用的 HTTP 协议版本。

[RFC 6585](http://tools.ietf.org/html/rfc6585) 最近刚刚发布，该文档描述了 4 个新的 HTTP 状态码。

HTTP 协议还在变化？是的，HTTP 协议一直在演变，新的状态码对于开发 REST 服务或者说是基于 HTTP 的服务非常有用，下面我们为你详细介绍这四个新的状态码以及是否应该使用。

##### 428 Precondition Required (要求先决条件)

先决条件是客户端发送 HTTP 请求时，如果想要请求能成功必须满足一些预设的条件。

一个好的例子就是 If-None-Match 头，经常在 GET 请求中使用，如果指定了 If-None-Match ，那么客户端只在响应中的 ETag 改变后才会重新接收回应。

先决条件的另外一个例子就是 If-Match 头，这个一般用在 PUT 请求上用于指示只更新没被改变的资源，这在多个客户端使用 HTTP 服务时用来防止彼此间不会覆盖相同内容。

当服务器端使用 [428 Precondition Required](http://tools.ietf.org/html/rfc6585#section-3) 状态码时，表示客户端必须发送上述的请求头才能执行请求，这个方法为服务器提供一种有效的方法来阻止 'lost update' 问题。

##### 429 Too Many Requests (太多请求)

当你需要限制客户端请求某个服务数量时，该状态码就很有用，也就是请求速度限制。

在此之前，有一些类似的状态码，例如 '509 Bandwidth Limit Exceeded'. [Twitter 使用 420](https://dev.twitter.com/docs/error-codes-responses) （这不是HTTP定义的状态码）

如果你希望限制客户端对服务的请求数，可使用 429 状态码，同时包含一个 Retry-After 响应头用于告诉客户端多长时间后可以再次请求服务。

##### 431 Request Header Fields Too Large (请求头字段太大)

某些情况下，客户端发送 HTTP 请求头会变得很大，那么服务器可发送 [431 Request Header Fields Too Large](http://tools.ietf.org/html/rfc6585#section-5) 来指明该问题。

我不太清楚为什么没有 430 状态码，而是直接从 429 跳到 431，我尝试搜索但没有结果。唯一的猜测是 430 Forbidden 跟 403 Forbidden 太像了，为了避免混淆才这么做的，天知道！

##### 511 Network Authentication Required (要求网络认证)

对我来说这个状态码很有趣，如果你在开发一个 HTTP 服务器，你不一定需要处理该状态码，但如果你在编写 HTTP 客户端，那这个状态码就非常重要。

如果你频繁使用笔记本和智能手机，你可能会注意到大量的公用 WIFI 服务要求你必须接受一些协议或者必须登录后才能使用。

这是通过拦截HTTP流量，当用户试图访问网络返回一个重定向和登录，这很讨厌，但是实际情况就是这样的。

使用这些“拦截”客户端，会有一些讨厌的副作用。在 RFC 中有提到这两个的例子：

* 如果你在登录WIFI前访问某个网站，网络设备将会拦截首个请求，这些设备往往也有自己的网站图标 ‘favicon.ico'。登录后您会发现，有一段时间内你访问的网站图标一直是WIFI登录网站的图标。
* 如果客户端使用HTTP请求来查找文档（可能是JSON），网络将会响应一个登录页，这样你的客户端就会解析错误并导致客户端运行异常，在现实中这种问题非常常见。

因此 511 状态码的提出就是为了解决这个问题。

如果你正在编写 HTTP 的客户端，你最好还是检查 511 状态码以确认是否需要认证后才能访问。

## 2.Forward与Redirct区别

**1.从地址栏显示来说**   
forward是服务器请求资源,服务器直接访问目标地址的URL,把那个URL的响应内容读取过来,然后把这些内容再发给浏览器.浏览器根本不知道服务器发送的内容从哪里来的,所以它的地址栏还是原来的地址.  
redirect是服务端根据逻辑,发送一个状态码（301、302）,告诉浏览器重新去请求那个地址.所以地址栏显示的是新的URL.

**2.从数据共享来说**   
forward:转发页面和转发到的页面可以共享request里面的数据.  
redirect:不能共享数据.

**3.从运用地方来说**   
forward:一般用于用户登陆的时候,根据角色转发到相应的模块.  
redirect:一般用于用户注销登陆时返回主页面和跳转到其它的网站等.

**4.从效率来说**   
forward:高.  
redirect:低.

**本质区别**

* **转发是服务器行为，重定向是客户端行为**

转发过程：客户浏览器发送http请求----》web服务器接受此请求--》调用内部的一个方法在容器内部完成请求处理和转发动作----》将目标资源 发送给客户；转发的路径必须是同一个web容器下的url，其不能转向到其他的web路径上去，中间传递的是自己的容器内的request。在客 户浏览器路径栏显示的仍然是其第一次访问的路径，也就是说客户是感觉不到服务器做了转发的。转发行为是浏览器只做了一次访问请求。

重定向过程：客户浏览器发送http请求----》web服务器接受后发送302状态码响应及对应新的location给客户浏览器--》客户浏览器发现 是302响应，则自动再发送一个新的http请求，请求url是新的location地址----》服务器根据此请求寻找资源并发送给客户。在这里 location可以重定向到任意URL，既然是浏览器重新发出了请求，则就没有什么request传递的概念了。在客户浏览器路径栏显示的是其重定向的 路径，客户可以观察到地址的变化的。重定向行为是浏览器做了至少两次的访问请求的。

**举例**

假设你去办理某个执照，

重定向：你先去了A局，A局的人说：“这个事情不归我们管，去B局”，然后，你就从A退了出来，自己乘车去了B局。

转发：你先去了A局，A局看了以后，知道这个事情其实应该B局来管，但是他没有把你退回来，而是让你坐一会儿，自己到后面办公室联系了B的人，让他们办好后，送了过来。

**问题**

* springmvc中controller内方法跳转forward？redirect？场景：华阳广告推送

forward：

1. 例如：return "forward:/rest/queryData"，实际的效果是在浏览器中的url地址还是原地址，存在重复提交的问题，所以forward就不推荐使用了
2. 如果是需要携带参数，直接拼接传递的参数，例如：return "forward:/rest/queryShopAlisName?phone=xxxxxxx"; 在跳转的controller中使用参数【@RequestParam("phone") String phone】获得传递的参数值，显然这样的方式也是不推荐的。

redirect：

1. 使用redirect在controller方法的参数中使用RedirectAttributes来

不带参数：直接使用 return "redirect:/rest/queryShopAlisName";浏览器的地址变成跳转的新地址，避免了重复提交的问题。

b、带参数的时候：

第一种选择：直接在url后面拼接参数，使用@RequestParam来取值，不推荐使用

第二种选择：在controller方法的参数中使用RedirectAttributes来传递参数

@RequestMapping(value = "/checkMember")

public String checkMember(HttpServletRequest request, RedirectAttributes attr) {

Member member = null;

try {

String phone = request.getParameter("phone");

\*\*\*attr.addAttribute("phone", "xxxx");\*\*\*

member = cashierService.checkIsMember(phone);

} catch (Exception e) {

logger.error("query member is error happen : " + e);

}

return "redirect:/rest/queryShopAlisName";

}