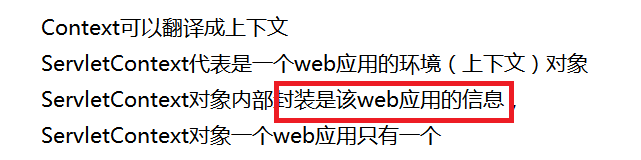
ALT SHIFT S + O

ALT SHIFT S + R

ALT SHIFT S + C

Alt + / 提示生成



ServletContext生命周期：

创建：服务期开启的时候创建（web被加载）

销毁：服务器关闭（web别移除）

获取：

1. 在init方法里面使用，通过ServletConfig对象获取（一般不使用）
2. 在servlet里面使用，this.getServletContext

域对象的通用方法：

SetAtrribute(String name, Object obj)

getAtrribute(String name);

removeAttribute(String name);

ServletContext的作用

1. 获取web应用全局的初始化参数
2. 获取web应用中任何资源的绝对路劲（重要重要重要）

注意：直接把文件放到项目里面，这个文件是无法获取的，因为发布是将webContext里面的资源整个copy过去，所以我们是没有办法获取的。

ServletContext context = getServletContext();

//获取a.txt文件绝对路径

String realPath\_A = context.getRealPath("a.txt");

//获取b.txt文件绝对路径

String realPath\_B = context.getRealPath("WEB\_INF/b.txt");

//获取c.txt文件绝对路径

String realPath\_C = context.getRealPath("WEB\_INF/classes/c.txt");

//获取d.txt文件绝对路径(无法获取！！)

String realPath\_D = context.getRealPath("WEB\_INF/classes/d.txt");

//类加载器获取src（classes）里面的资源--专门用来加载classes下面的文件的。

//getResource的相对路径相对于classes

String path = ContextServlet.**class**.getClassLoader().getResource("").getPath();

（3）servletContext是一个域对象（重要重要重要）

存储数据的区域

servletContext域对象作用范围：整个web应用（所有的web资源都可以随意向servletContext域存取数据，实现数据共享）

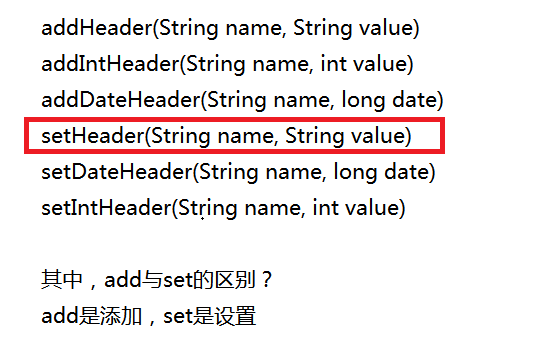
hTTP响应：

1. 响应行：
2. http协议
3. 状态码：200 302（请求重定向）304（访问本地缓存）404（找不到，路径错误，资源删除）500（服务器内部出错，通常程序出错）

手动设置http响应行中的状态码：respons.setStatus(404);

1. 状态信息：对应状态吗
2. 响应头：

服务器将信息以键值对的形式返回给客户端，理解为服务器提要求，客户端根据这个需求做出对应的响应



定时刷新：refresh

1. 响应体：

服务器写给客户端的页面正文，浏览器会去加载，特殊的文件，字节信息、

响应体设置文本

注意：地址写法

1. 服务器内部跳转，不带项目名
2. 客户端请求服务器地址，带项目名

Request：代表请求，所以我们可以通过对象分别获取http请求的信息（请求行，请求头，请求体）

1. 请求行：

请求方式：get post

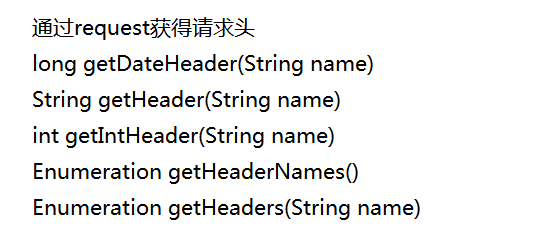
请求的资源：

协议版本：Http/1.1 Http/1.0类比发短信



2.请求头

请求头是客户端发送给服务器端的一些信息，告诉服务器我是一个怎样的客户端，使用键值对，也会存在特殊属性对应有特殊的功能



3.请求体：放post发送的内容

当我们请求方式是Post的时候，请求体才会有参数

Username = 张三&password=999

当我们请求方式是get，请求参数拼接出现在资源路径地址上

通过request去获取：

获取指定的单个表单之

String getParameter(String name);

获取多个表单值，返回数组

String getParameterValues(String name);

获取form表单里所有参数的名称

Enumeration getParameterNames();

获得所有的参数把这些参数封装到MAP集合里面

Map getParameterMap();

设置编码格式：响应是网页的，response来设置，服务器的话就是request来设置。

request请求转发是服务器内部

特点：地址栏不变，一次请求

重定向是发起多次的请求

特点：地址栏改变，访问服务器两次，有两个请求和响应

总：根据是否让地址栏改变来决定选择哪一种请求方式，频繁的刷新改变地址栏的话选择重定向，访问特点，需要传递request选择请求转发。

Request域对象：reques对象也是一个储存数据的区域对象

SetAttribute（String name,Object o）;

getAttribute(String name);

removeAttribute(String name);

ServletContext域与Request域的生命周期比较：

ServletContext：

创建：服务器启动

销毁：服务器关闭

作用范围：整个web项目

Request：

创建：访问时创建request

销毁：响应结束request销毁

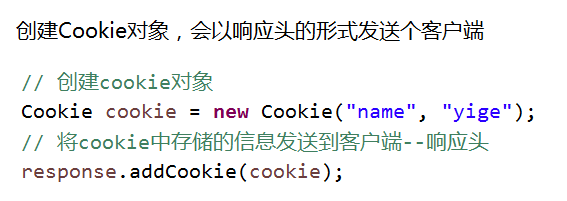
作用范围：一次请求中

会话技术cookie：帮助服务器记住客户端的状态，区分客户端

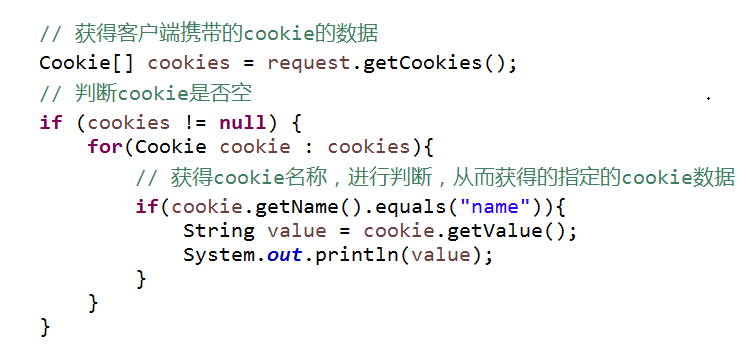
1. 把数据存储在客户端本地
2. 区分谁的数据，区分客户端

Cookie：将用户的数据存储到客户端本地技术

1. 服务器怎么将一个Cookie发送到客户端



1. 服务器怎么接受客户端所携带的cookie



第一次访问cookie会在响应头中，发送cookie到客户端

第二次访问可以在请求中看到cookie，这就是客户端通过请求将cookie发送到服务器

注意：cookie不能储存中文

会话技术的生命周期：

从打开一个浏览器访问某个站点，到关闭这个浏览器的整个过程，成为一次会话

创建：某个站点

销毁：关闭这个浏览器

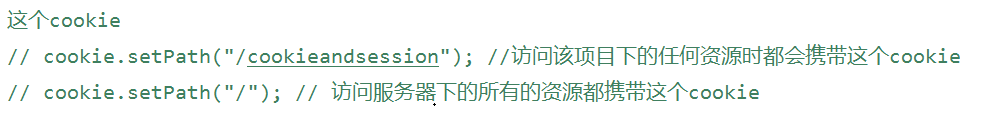
设置cookie在客户端的持久化时间

如果不设置持久化时间，cookie会储存在浏览器的内存中，浏览器关闭，cookie信息销毁（会话级别的cookie）

Cookie.setMaxAge(秒)；

设置cookie的携带路径，那么该cookie信息会在访问产生该cookie的web资源所在的路径都携带cookie信息

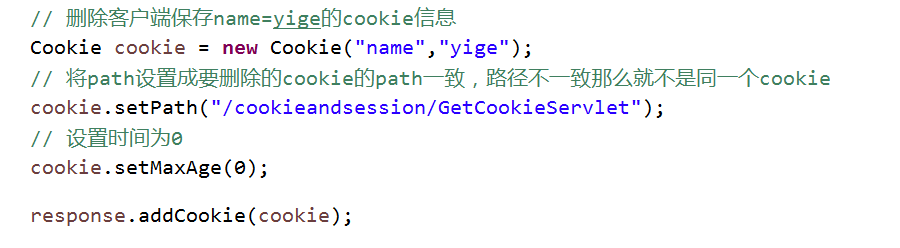
Cookie.setPath(“路径”);



如果设置持久化时间，cookie信息会被持久化到浏览器的磁盘文件中

删除客户端的cookie

如果要删除客户端已经存在的cookie信息，那么使用同名同路径的持久化时间为0的cookie进行覆盖即可



Session技术：将数据储存到服务器，会为每一个客户端创建一个内存空间，存储客户端数据，但是客户端没次都携带一个带有标识的ID去服务器内部寻找属于自己的内存空间。

Session的实现是基于cookie的，session需要借助cookie存储客户端的唯一标识（JSESSIONID）

获取session对象

HTTPSession session = request.getSession();

该方法会获取的专属于当前会话的session对象，判断服务器端有没有该回话的session对象如果服务器有了属于该会话session直接将已有的session返回

如果没有则创建

Session也是储存数据的区域对象（Session域对象）

Session对象的生命周期

创建：第一执行request.getSession();时创建；

销毁：服务器非正常关闭时，session过期失效，手动销毁：session.invalidate();

作用范围：默认再一次会话中

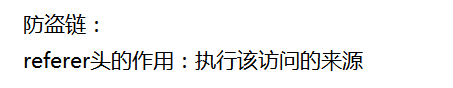
注意：

浏览器关闭，Session就销毁吗？

不对，销毁的cookie，session对象还会存在

Session的持久化：

因为Session的标识JSESSIONID是存储在cookie上的，而我们关闭浏览器的时候，cookie信息便销毁，但是我们可以通过给cookie设置持久化时间来让session保持持久化





URLDecoder类包含一个decode(String s,String charcter)静态方法,它可以将看上去乱码的特殊字符串转换成普通字符串

URLEncoder类包含一个encode(String s,String charcter)静态方法,它可以将普通字符串转换成application/x-www-form-urlencoded MIME字符串