http get 和 post 有什么区别

1、get 用于获取数据，post 用于提交数据

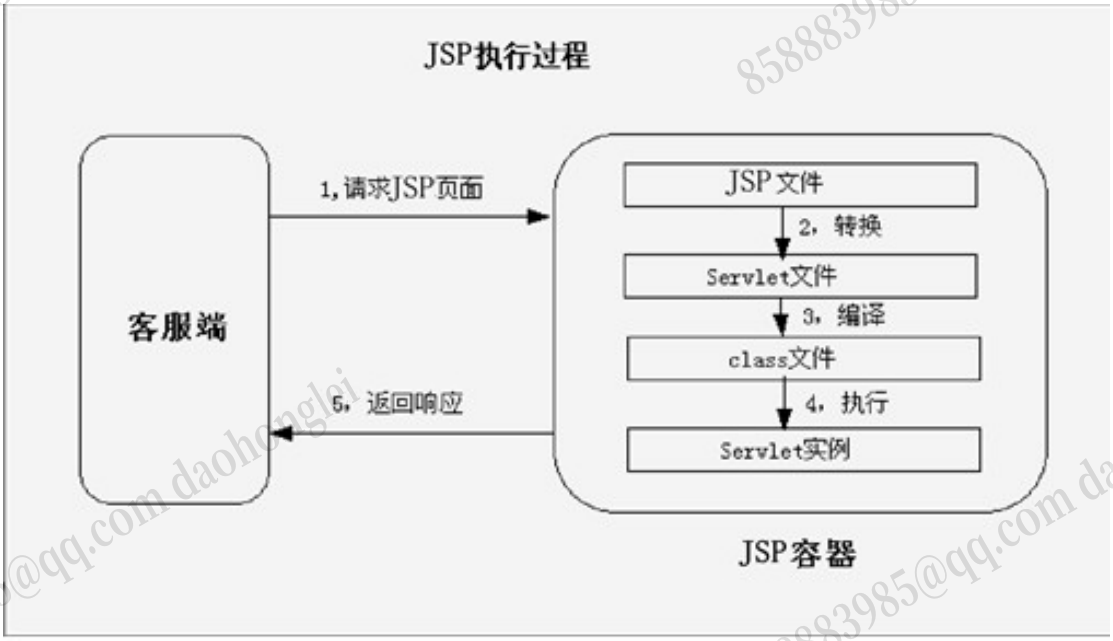
2、get 提交参数追加在 url 后面，post 参数可以通过 http body 提交

3、get 的 url 会有长度上的限制，则 post 的数据则可以非常大

4、get 提交信息明文显示在 url 上，不够安全，post 提交的信息不会在 url 上显示

5、get 提交可以被浏览器缓存，post 不会被浏览器缓存

JSP 的执行过程



1.当一个 JSP 文件第一次被请求的时候，JSP 引擎(本身也是一个 Servlet)首先会把这个 JSP 文件转换成一个 Java 源文件。在转换过程中如果发现 JSP 文件有语法错误，转换过程将中断，并向服务端和客户端输出出错信息；如果转换成功，JSP 引擎用 javac 把该 Java 源文件编译成相应的.class 文件并将该.class 文件加载到内存中。

2.其次创建一个该 Servlet 的实例，并执行该实例的 jspInit()方法(jspInit()方法在 Servlet 的生命周期中只被执行一次)。

3.然后创建并启动一个新的线程，新线程调用实例的 jspService()方法。(对于每一个请求，JSP 引擎会创建一个新的线程来处理该请求。如果有多个客户端同时请求该 JSP 文件，则 JSP 引擎会创建多个线程，每个客户端请求对应一个线程)。

4.浏览器在调用 JSP 文件时，Servlet 容器会把浏览器的请求和对浏览器的回应封装成 HttpServletRequest 和 HttpServletResponse 对象，同时调用对应的 Servlet 实例中的 jspService()方法，把这两个对象作为参数传递到 jspService()方法中。 jspService()方法执行后会将 HTML 内容返回给客户端。

5.如果 JSP 文件被修改了，服务器将根据设置决定是否对该文件进行重新编译。如果需要重新编译，则将编译结果取代内存中的 Servlet，并继续上述处理过程。 如果在任何时候由于系统资源不足，JSP 引擎将以某种不确定的方式将 Servlet 从内存中移去。当这种情况发生时，jspDestroy()方法首先被调用, 然后 Servlet 实例便被标记加入“垃圾收集”处理。

jsp 有哪些内置对象？作用分别是什么?

JSP 有 9 个内置对象：

- request：封装客户端的请求，其中包含来自 GET 或 POST 请求的参数；
- response：封装服务器对客户端的响应；
- pageContext：通过该对象可以获取其他对象；
- session：封装用户会话的对象；
- application：封装服务器运行环境的对象；
- out：输出服务器响应的输出流对象；

- config：Web 应用的配置对象；
- page：JSP 页面本身（相当于 Java 程序中的 this）；
- exception：封装页面抛出异常的对象。

说一下 jsp 的 4 种作用域?

1. JSP 中的四种作用域包括 page、request、session 和 application，具体来说：
2. page 代表与一个页面相关的对象和属性。
3. request 代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件；需要在页面显示的临时数据可以置于此作用域。
4. session 代表与某个用户与服务器建立的一次会话相关的对象和属性。跟某个用户相关的数据应该放在用户自己的 session 中。
5. application 代表与整个 Web 应用程序相关的对象和属性，它实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域。

Servlet 的运行过程

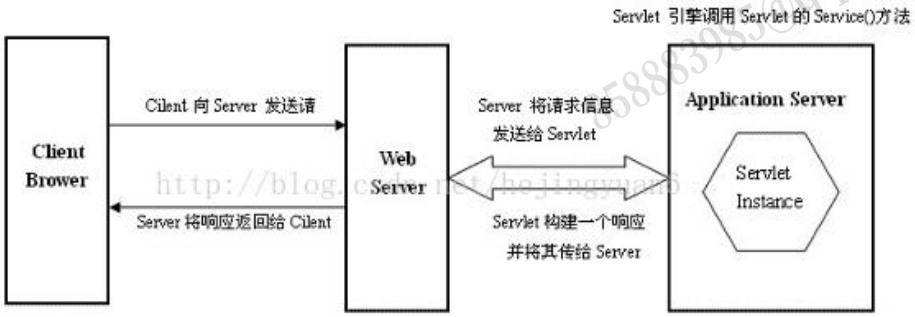
1 客户端发送请求至服务器端；

2 服务器端根据 web.xml 文件中的 Servlet 相关配置信息，将客户端请求转发到相应的 Servlet

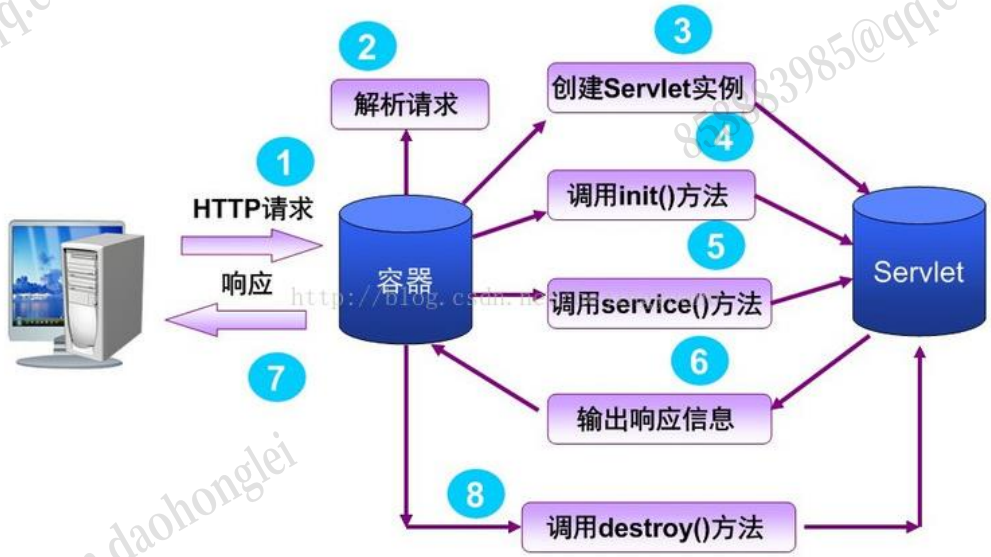
3 Servlet 引擎调用 Service()方法，根据 request 对象中封装的用户请求与数据库进行交互，返回数据之后，Servlet 会将返回的数据封装到 response 对象中；

4 Servlet 生成响应内容并将其传给服务器。响应内容动态生成，通常取决于客户端的请求

5 服务器将响应返回给客户端



Servlet 生命周期



1） 加载和实例化；在第一次请求 Servlet 时，Servlet 容器将会创建 Servlet 实例；

2） 初始化；Servlet 容器加载完成 Servlet 之后，必须进行初始化，此时，init 方法将被调用；

3） Servlet 初始化之后，就处于响应请求的就绪状态，此时如有客户端请求发送，就会调用 Servlet 实例的 service()方法，并且根据用户的请求方式，调用 doPost 或者 doGet 方法；

4） 最后，Servlet 容器负责将 Servlet 实例进行销毁，调用 destroy 方法实现；

对于更多的客户端请求，Server 创建新的请求和响应对象，仍然激活此 Servlet 的 service()方法，将这两个对象作为参数传递给它。如此重复以上的循环，但无需再次调用 init()方法。

一般 Servlet 只初始化一次(只有一个对象),当 Server 不再需要 Servlet 时(一般当 Server 关闭时)，Server 调用 Servlet 的 Destroy()方法。

session 和 cookie 有什么区别?

- 由于 HTTP 协议是无状态的协议，所以服务端需要记录用户的状态时，就需要用某种机制来识具体的用户，这个机制就是 Session.典型的场景比如购物车，当你点击下单按钮时，由于 HTTP 协议无状态，所以并不知道是哪个用户操作的，所以服务端要为特定的用户创建了特定的 Session，用用于标识这个用户，并且跟踪用户，这样才知道购物车里面有几本书。这个 Session 是保存在服务端的，有一个唯一标识。在服务端保存 Session 的方法很多，内存、数据库、文件都有。集群的时候也要考虑 Session 的转移，在大型的网站，一般会有专门的 Session 服务器集群，用来保存用户会话，这个时候 Session 信息都是放在内存的，使用一些缓存服务比如 Memcached 之类的来放 Session。
- 思考一下服务端如何识别特定的客户？这个时候 Cookie 就登场了。每次 HTTP 请求的时候，客户端都会发送相应的 Cookie 信息到服务端。实际上大多数的应用都是用 Cookie 来实现 Session 跟踪的，第一次创建 Session 的时候，服务端会在 HTTP 协议中告诉客户端，需要在 Cookie 里面记录一个 Session ID，以后每次请求把这个会

话 ID 发送到服务器，我就知道你是谁了。有人问，如果客户端的浏览器禁用了 Cookie 怎么办？一般这种情况下，会使用一种叫做 URL 重写的技术来进行会话跟踪，即每次 HTTP 交互，URL 后面都会被附加上一个诸如 sid=xxxxx 这样的参数，服务端据此来识别用户。

- Cookie 其实还可以用在一些方便用户的场景下，设想你某次登陆过一个网站，下次登录的时候不想再次输入账号了，怎么办？这个信息可以写到 Cookie 里面，访问网站的时候，网站页面的脚本可以读取这个信息，就自动帮你把用户名给填了，能够方便一下用户。这也是 Cookie 名称的由来，给用户的一点甜头。所以，总结一下：Session 是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中；Cookie 是客户端保存用户信息的一种机制，用来记录用户的一些信息，也是实现 Session 的一种方式。

说一下 session 的工作原理?

其实 session 是一个存在服务器上的类似于一个散列表格的文件。里面存有我们需要的信息，在我们需要用的时候可以从里面取出来。类似于一个大号的 map 吧，里面的键存储的是用户的 sessionid，用户向服务器发送请求的时候会带上这个 sessionid。这时就可以从中取出对应的值了。

如果客户端禁止 cookie 能实现 session 还能用吗?

Cookie 与 Session，一般认为是两个独立的东西，Session 采用的是在服务器端保持状态的方案，而 Cookie 采用的是在客户端保持状态的方案。但为什么禁用 Cookie 就不能得到 Session 呢？因为 Session 是用 Session ID 来确定当前对话所对应的服务器

Session，而 Session ID 是通过 Cookie 来传递的，禁用 Cookie 相当于失去了 Session ID，也就得不到 Session 了。

假定用户关闭 Cookie 的情况下使用 Session，其实现途径有以下几种：

1. 手动通过 URL 传值、隐藏表单传递 Session ID。
2. 用文件、数据库等形式保存 Session ID，在跨页过程中手动调用。

https://github.com/daohonglei/javaStereotypedWriting

https://gitee.com/daohonglei/javaStereotypedWriting