Head First JSP&Servlet 杂项笔记

1. javax.servlet.ServletRequest的getParameter方法和getParameterValues方法

HTTPServletRequest接口继承了ServletRequest接口

1. getParameter方法只用于获取单值参数

2）返回多值请使用getParameterValues方法，将返回一个String[]

2.Servlet真正的任务是**处理请求**

3.谁来实现HttpServletRequest和HttpServletResponse的接口？

A：容器，而且我们在API中是看不到源码的，只能看到接口

5.接口解耦成

GenericServlet和ServletRequest等。。。是为了提供servlet模型的灵活性，比如**把http协议和servlet解耦**，这样还可以用于实现其他协议的servlet。

6.Service方法处理请求，

如果HTTP方法是GET，service就会调用doGet，POST就调用doPost。Service还有其他的do方法，但servlet程序员一般只用这两个。

7.除了参数，还能从请求对象中获得什么东西

1）客户的平台和浏览器信息

String client = request.getHeader(“User-Agent”);

2）与请求相关的cookie

Cookie[] cookies = request.getCookies();

3）与客户相关的会话

HttpSession session = request.getSession();

4）请求的HTTP方法

String theMethod = request.getMethod();

5）请求的输入流

InputStream input = request.getInputStream();

6）etc.

8.得到请求的InputStream有什么用？

如果只是GET请求，那么没什么用，只是获取首部信息，因为参数信息在url给出了，

如果是POST请求，一般我们只用request.getParameter()抽出想要的参数值，如果想得到原始字节，就使用getInputStream。得到流后我们**还**可以**改变流中的首部信息**，**处理消息体的原始字节**，可能还会立即把它**写到服务器上的一个文件**中

9. getServerPort()、getLocalPort()、getRemotePort()的作用的辨析

1）getRemotePort获得远程的端口，相对于服务器，客户就是远程的，所以这是获得发出请求的客户端的端口号

2）请求会发送到服务器的一个指定端口（服务器的监听端口），但**服务器会为每个线程再找一个不同的本地端口**，这样，一个应用就能同时处理多个客户了。

所以，getServerPort()说请求**原来发送到**哪个端口

getLocalPort()说请求**最后发送到**哪个端口

10.GET请求**幂等**、POST请求**不幂等**，这是HTTP规范，实现时当然可以违背规范，但并不建议！

11.HTTP响应

大多数情况下，使用响应只是为了向客户发回数据

会对响应response调用两个方法：setContentType和getWriter

也可以使用响应**设置其他首部**，**发送错误**，以及**增加cookie**

12.在业务逻辑中发送一个文件（比如下载）

doGet(……){

response.setContentType(“application/jar”); //这是对浏览器的告知

ServletContext ctx = getServletContext();

InputStream is = ctx.getResourceAsStream(“/bookcode.jar”);

int read = 0;

byte[] bytes = new byte[1024];

OutputStream os = response.getOutputStream();

while( (read = is.read(bytes)) != -1){

//输出流，bytes是数据源，0和read是每次读取的长度

os.write(bytes, 0, read);

}

os.flush();

os.close();

}

当然，下载文件我们也可以直接在html中指定一个绝对路径的超链接，但是这不够灵活！因为我们可能还要动态生成文件（采用实时的字节），比如使用机器学习技术，将用户传入的图片数据生成合成的名画

13.对于输出，有两个选择

1）PrintWriter

2）OutputStream