HttpServlet

HttpServlet首先必须读取Http请求的内容。Servlet容器负责创建HttpServlet对象，并把Http请求直接封装到HttpServlet对象中，HttpServlet容器响应Web客户请求流程如下：

1）Web客户向Servlet容器发出Http请求；

2）Servlet容器解析Web客户的Http请求；

3）Servlet容器创建一个HttpRequest对象，在这个对象中封装Http请求信息；

4）Servlet容器创建一个HttpResponse对象；

5）Servlet容器调用HttpServlet的service方法，把HttpRequest和HttpResponse对象作为service方法的参数传给HttpServlet对象；

6）HttpServlet调用HttpRequest的有关方法，获取HTTP请求信息；

7）HttpServlet调用HttpResponse的有关方法，生成响应数据；

8）Servlet容器把HttpServlet的响应结果传给Web客户。

创建HttpServlet的步骤——“四部曲”

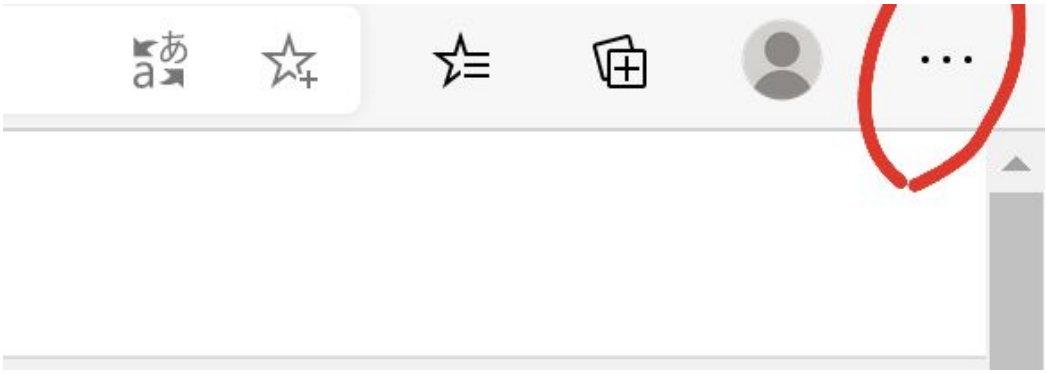
1）扩展HttpServlet抽象类；

2）覆盖HttpServlet的部分方法，如覆盖doGet()或doPost()方法；

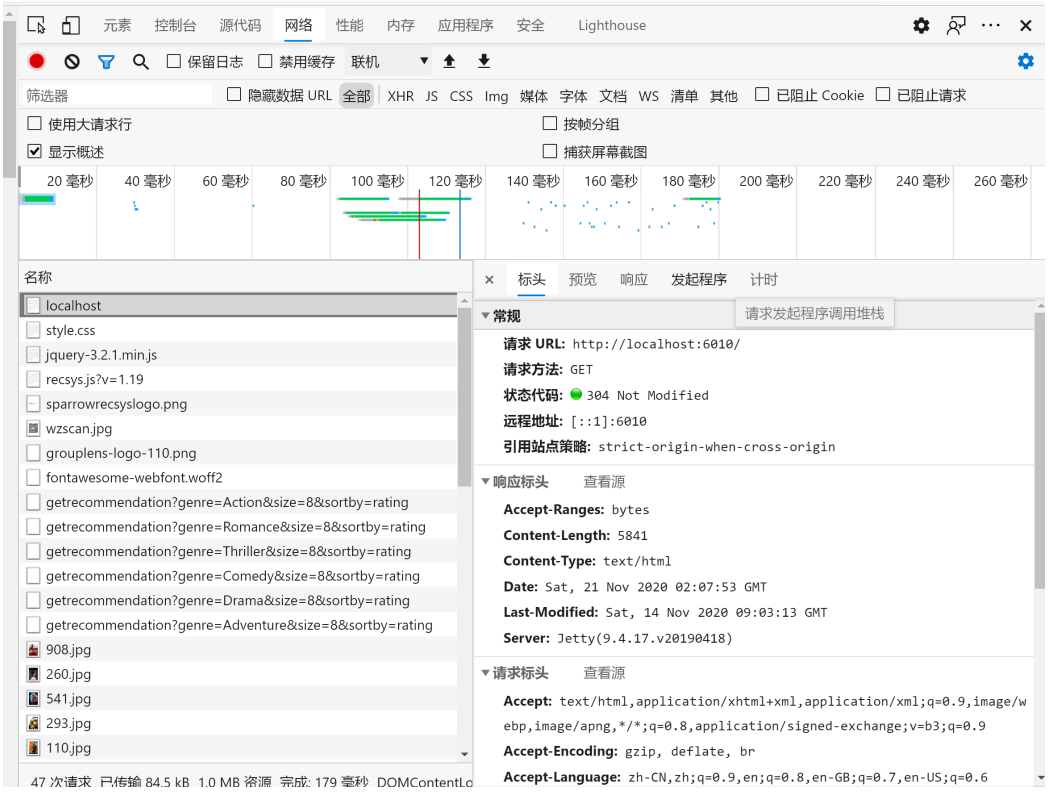
3）获取HTTP请求信息。通过HttpServletRequest对象来检索HTML表单所提交的数据或URL上的查询字符串；

4）生成HTTP响应结果。通过HttpServletResponse对象生成响应结果，它有一个getWriter()方法，该方法返回一个PrintWriter对象。

查看http请求头的方法



更多工具 - 开发人员工具 - 网络



HttpServletRequest：获取HTTP请求消息

获取请求行信息的常用方法

方法声明	功能描述
String getMethod()	该方法用于获取 HTTP 请求消息中的请求方式（如 GET、POST 等）
String getRequestURI()	该方法用于获取请求行中的资源名称部分即位于 URL 的主机和端门之后、参数部分之前的部分
String getQueryString()	该方法用于获取请求行中的参数部分，也就是资源路径后问号（？）以后的所有内容
String getContextPath()	该方法用于获取请求 URL 中属于 Web 应用程序的路径，这个路径以 / 开头，表示相对于整个 Web 站点的根目录，路径结尾不含 /。如果请求 URL 属于 Web 站点的根目录，那么返回结果为空字符串（""）
String getServletPath()	该方法用于获取 Servlet 的名称或 Servlet 所映射的路径
String getRemoteAddr()	该方法用于获取请求客户端的 IP 地址，其格式类似于 192.168.0.3
String getRemoteHost()	该方法用于获取请求客户端的完整主机名，其格式类似于 pcl.mengma.com。需要注意的是，如果无法解析出客户机的完整主机名，那么该方法将会返回客户端的 IP 地址
int getRemotePort()	该方法用于获取请求客户端网络连接的端口号
String getLocalAddr()	该方法用于获取 Web 服务器上接收当前请求网络连接的 IP 地址
String getLocalName()	该方法用于获取 Web 服务器上接收当前网络连接 IP 所对应的主机名

int getLocalPort()	该方法用于获取 Web 服务器上接收当前网络连接的端口号
String getServerName()	该方法用于获取当前请求所指向的主机名，即 HTTP 请求消息中 Host 头字段所对应的主机名部分
int gctServcrPort()	该方法用于获取当前请求所连接的服务器端口号，即 HTTP 请求消息中 Host 头字段所对应的端口号部分
StringBuffcr getRequestURL()	该方法用于获取客户端发出请求时的完整 URL，包括协议、服务器名、端口号、 资源路径等信息，但不包括后面的查询参数部分。注意，getRequcstURL() 方法返回的结果是 StringBuffer 类型，而不是 String 类型，这样更便于对结果进行修改

获取请求消息头的方法

方法声明	功能描述
String getHeader(String name)	该方法用于获取一个指定头字段的值，如果请求消息中没有包含指定的头字段，则 getHeader() 方法返回 null；如果请求消息中包含多个指定名称的头字段，则 getHeader() 方法返回其中第一个头字段的值
Enumeration getHeaders(String name)	该方法返回一个 Enumeration 集合对象，该集合对象由请求消息中出现的某个指定名称的所有头字段值组成。在多数情况下，一个头字段名在请求消息中只出现一次，但有时可能会出现多次
Enumeration getHeaderNames()	该方法用于获取一个包含所有请求头字段的 Enumeration 对象
int getIntHeader(String name)	该方法用于获取指定名称的头字段，并且将其值转为 int 类型。需要注意的是，如果指定名称的头字段不存在，则返回值为 -1；如果获取到的头字段的值不能转为 int 类型，则将发生 NumberFormatException 异常
long getDateHeader(String name)	该方法用于获取指定头字段的值，并将其按 GMT 时间格式转换为一个代表日期/时间的长整数，该长整数是自 1970 年 1 月 1 日 0 时 0 分 0 秒算起的以毫秒为单位的时间值
String getContentType()	该方法用于获取 Content-Type 头字段的值，结果为 String 类型
int getContentLength()	该方法用于获取 Content-Length 头字段的值，结果为 int 类型
String getCharacterEncoding()	该方法用于返回请求消息的实体部分的字符集编码，通常是从 Content-Type 头字段中进行提取，结果为 String 类型

HttpServletResponse：封装HTTP响应消息

设置响应消息头字段的方法

发送状态码相关的方法

当 Servlet 向客户端回送响应消息时，需要在响应消息中设置状态码。因此，HttpServletResponse 接口定义了两个发送状态码的方法。

1）setStatus（int status）方法

该方法用于设置 HTTP 响应消息的状态码，并生成响应状态行。由于响应状态行中的状态描述信息直接与状态码相关，而 HTTP 版本由服务器确定，因此，只要通过 setStatus（int status）方法设置了状态码，即可实现状态行的发送。需要注意的是，在正常情况下，Web 服务器会默认产生一个状态码为 200 的状态行。

2）sendError（int sc）方法

该方法用于发送表示错误信息的状态码。例如，404 状态码表示找不到客户端请求的资源。response 对象提供了两个重载的 sendError（int sc）方法，具体如下：

```
public void sendError(int code) throws java.io.IOException
public void sendError(int code,String message)throws java.io.IOException
```

在上面重载的两个方法中，第一个方法只发送错误信息的状态码，而第二个方法除了发送状态码以外，还可以增加一条用于提示说明的文本信息，该文本信息将出现在发送给客户端的正文内容中。

方法声明	功能描述
void addHeader(String name,String value)	这两个方法都是用于设置 HTTP 协议的响应头字段。其中，参数 name 用于指定响应头字段的名称，参数 value 用于指定响 应头字段的值。不同的是，addHeader() 方法可以增加同名的响应头字段，而 setHeader() 方法则会覆盖同名的头字段
void setHeader (String name,String value)	
void addIntHeader(String name,int value)	这两个方法专门用于设置包含整数值的响应头，避免了使用 addHeader() 与 setHeader() 方法时需要将 int 类型的设置值转换为 String 类型的麻烦
void setIntHeader(String name, int value)	
void setContentType(String type)	该方法用于设置 Servlet 输出内容的 MIME 类型，对于 HTTP 协议来说，就是设置 Content-Type 响应头字段的值。例如，如果发送到客户端的内容是 jpeg 格式的图像数据,就需要将响应头字段的类型设置为 image/jpeg。需要注意的是，如果响应的内容为文本，setContentType() 方法还可以设置字符编码，如 text/html;charset = UTF-8
void setLocale (Locale loc)	该方法用于设置响应消息的本地化信息。对 HTTP 来说，就是设置 Content-Language 响应头字段和 Content-Type 头字段中的字符集编码部分。需要注意的是，如果 HTTP 消息没有设置 Content-Type 头字段，则 setLocale() 方法设置的字符集编码不会出现在 HTTP 消息的响应头中，如果调用 setCharacterEncoding() 或 setContentType() 方法指定了响应内 容的字符集编码，则 setLocale() 方法将不再具有指定字符集编码的功能
void setCharacterEncoding(String charset)	该方法用于设置输出内容使用的字符编码，对 HTTP 协议来说，就是设置 Content-Type 头字段中的字符集编码部分。如果没有设置 Content-Type 头字段，则 setCharacterEncoding 方法设 置的字符集编码不会出现在 HTTP 消息的响应头中。setCharacterEncoding() 方法比 setContentType() 和 setLocale() 方法的优先权高，它的设置结果将覆盖 setContentType() 和 setLocale() 方法所设置的字符码表

发送响应消息体相关的方法

由于在 HTTP 响应消息中，大量的数据都是通过响应消息体传递的，因此，ServletResponse 遵循以 I/O 流传递大量数据的设计理念。在发送响应消息体时，定义了两个与输出流相关的方法。

1）getOutputStream() 方法

该方法所获取的字节输出流对象为 ServletOutputStream 类型。由于 ServletOutputStream是OutputStream 的子类，它可以直接输出字节数组中的二进制数据。因此，要想输出二进制格式的响应正文，就需要使用 getOutputStream() 方法。

2）getWriter() 方法

该方法所获取的字符输出流对象为 PrintWriter 类型。由于 PrintWriter 类型的对象可以直接输出字符文本内容，因此，要想输出内容全部为字符文本的网页文档，则需要使用 getWriter() 方法。

注意：虽然 response 对象的 getOutputStream() 和 getWriter() 方法都可以发送响应消息体，但是，它们之间互相排斥，不可同时使用，否则会发生 IllegalStateException 异常。