Servlet概述

星期五,2019年7月12日 9:00

1. servlet概述

servlet技术是一门由sun公司提供的动态资源开发的技术。servlet本质就是一个java文件。存放在servlet容器中。

servlet容器：

存储并运行servlet的环境，tomcat。

web容器：

存储并运行web资源的环境，tomcat。

2. servlet开发

a. 编写servlet文件，放入servlet容器

i. 编写一个.java文件，并实现 Servlet接口即可成为一个servlet文件。

ii. 具体实现：继承GenericServlet抽象类，重写service方法。

iii. 编写FirstServlet类，内部添加：

```
package cn.tedu.servlet;
import java.io.*;
import javax.servlet.*;
public class FirstServlet extends GenericServlet{
      public void service(ServletRequest req,
                  ServletResponse res)
            throws ServletException,
                  IOException{

            res.getWriter().write("first Servlet!!!");
   }
}
```

iv. 临时修改classpath，添加javax扩展包：

```
C:\>set classpath=%classpath%;C:\software\apache-tomcat-7.0.62\lib\servlet-api.j
ar

C:\>
```

v. 带包编译：

javac -d . FirstServlet.java

vi. 将编译后的目录放入一个web应用的classes目录中

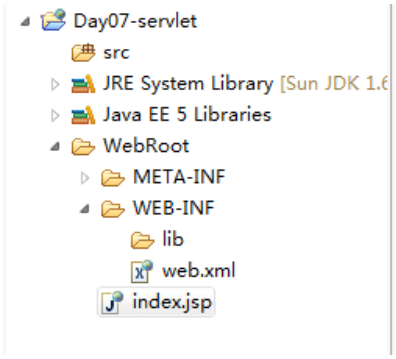
b. 配置servlet映射

```
<servlet>
    <servlet-name>FirstServlet</servlet-name>
    <servlet-class>cn.tedu.servlet.FirstServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FirstServlet</servlet-name>
    <url-pattern>/servlet/FirstServlet</url-pattern>
  </servlet-mapping>
```

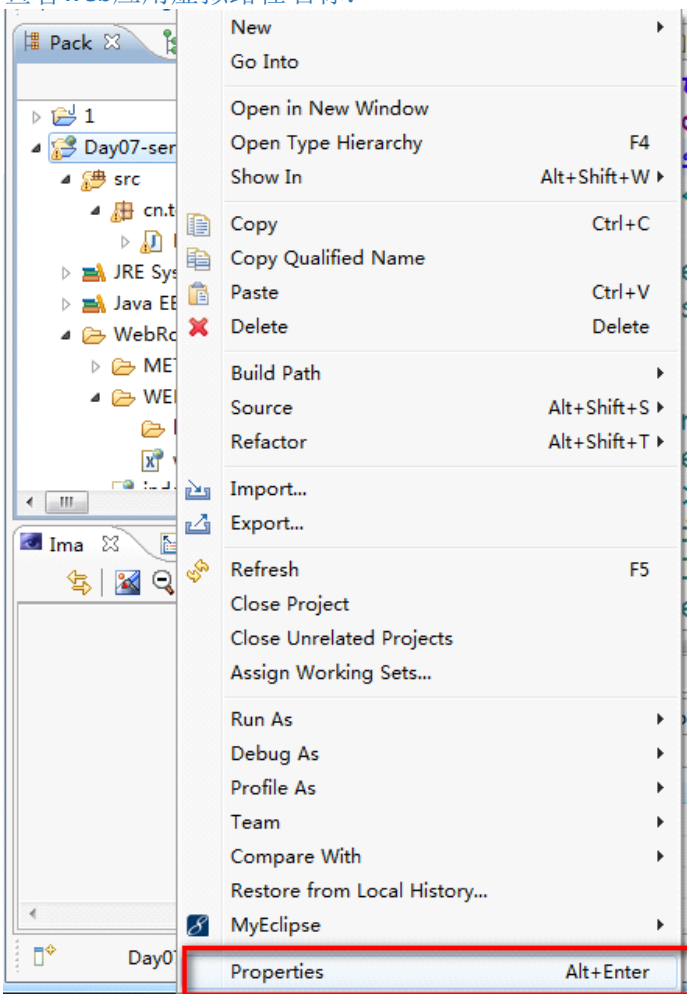
Myeclipse开发Servlet

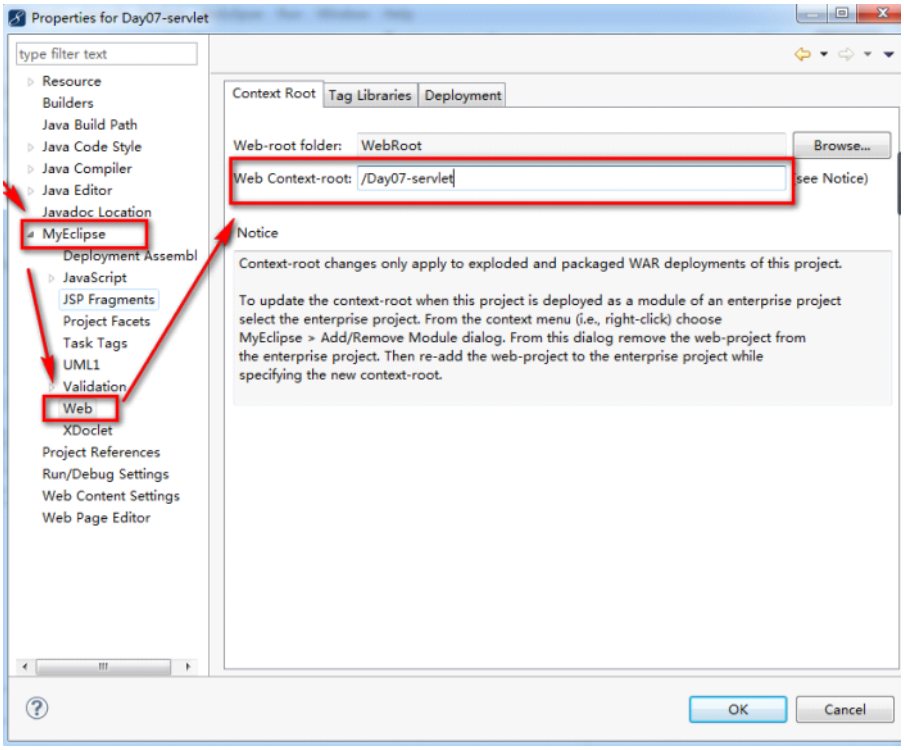
星期五,2019年7月12日　　10:56

1. 安装Myeclipse
2. Myeclipse创建web工程
   a. 创建一个web工程

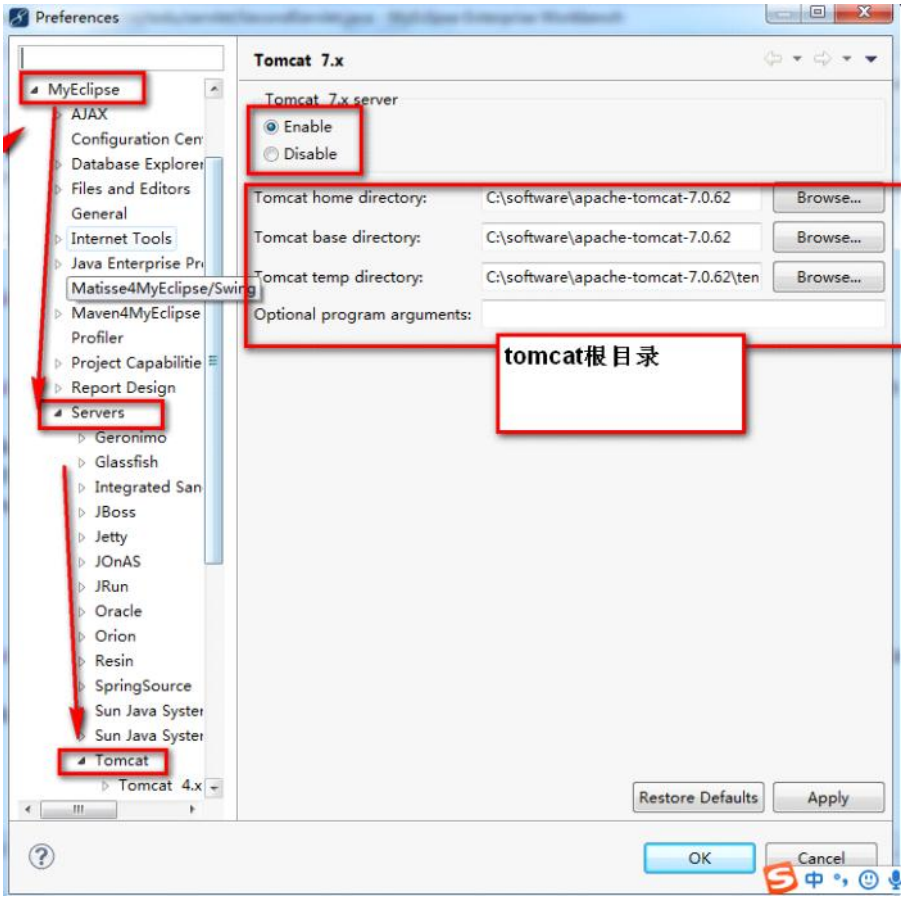


   b. 查看web应用虚拟路径名称：

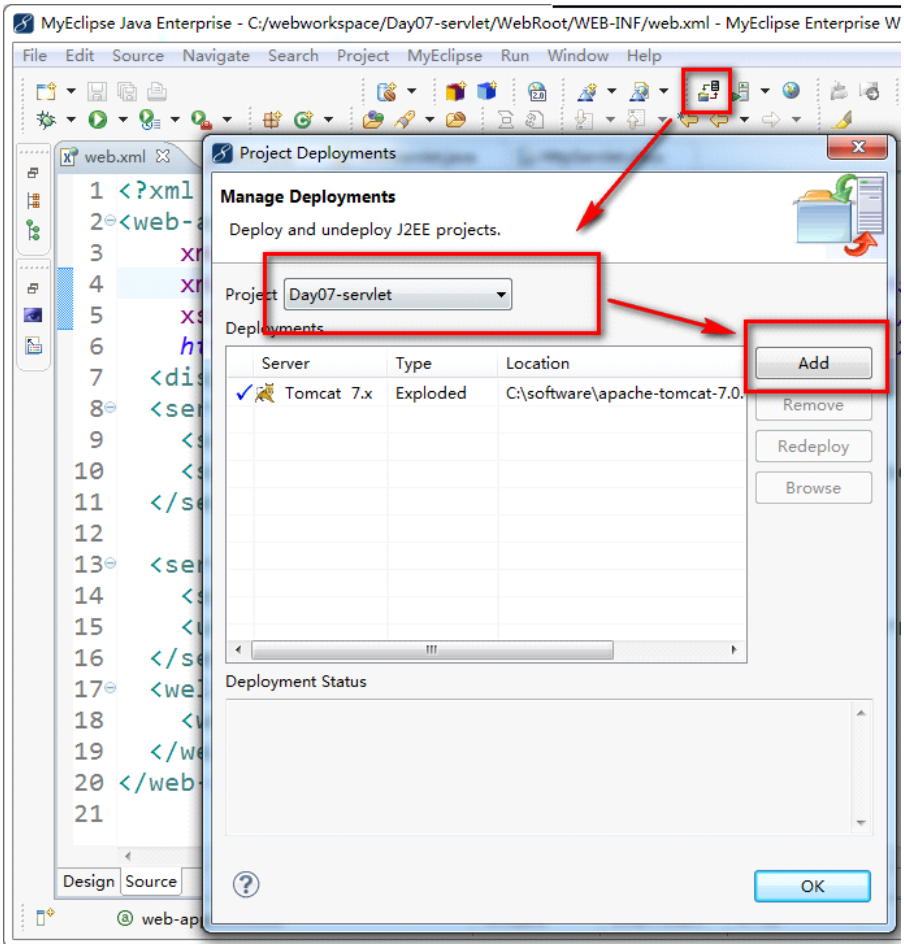


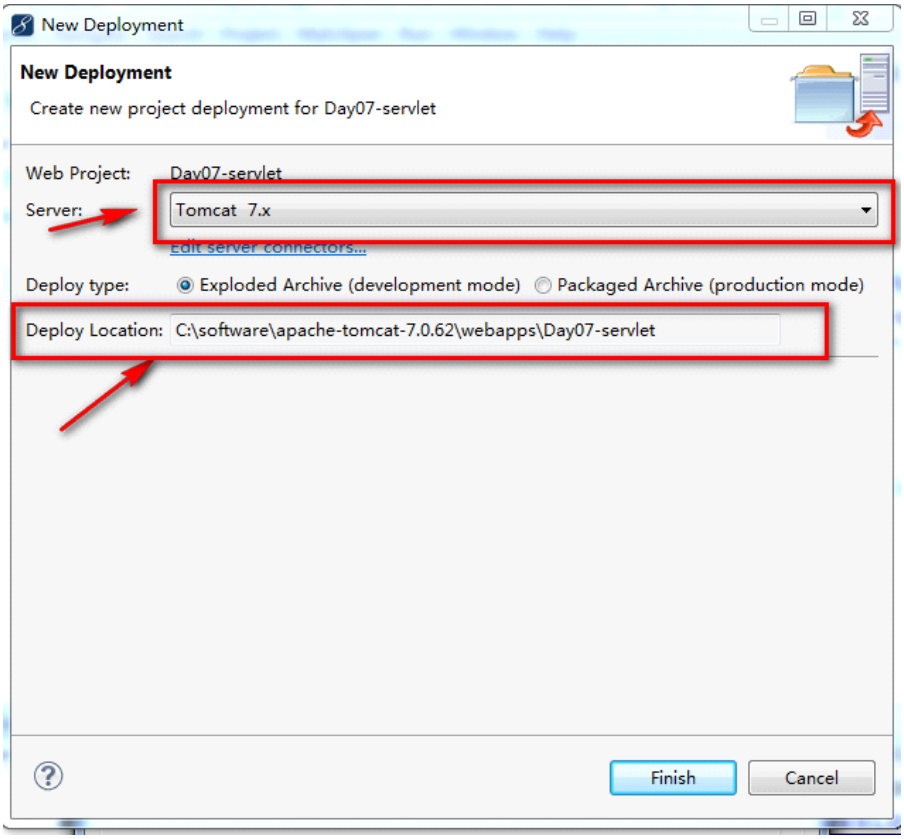


c. 部署tomcat



d. 发布web应用：





e. 在浏览器中访问：

localhost/Day07-servlet/servlet/FirstServlet

这样就可以访问到Day07-servlet的web应用中的FirstServlet文件。

在页面中输出response.getWriter().write("输出内容");中的数据。

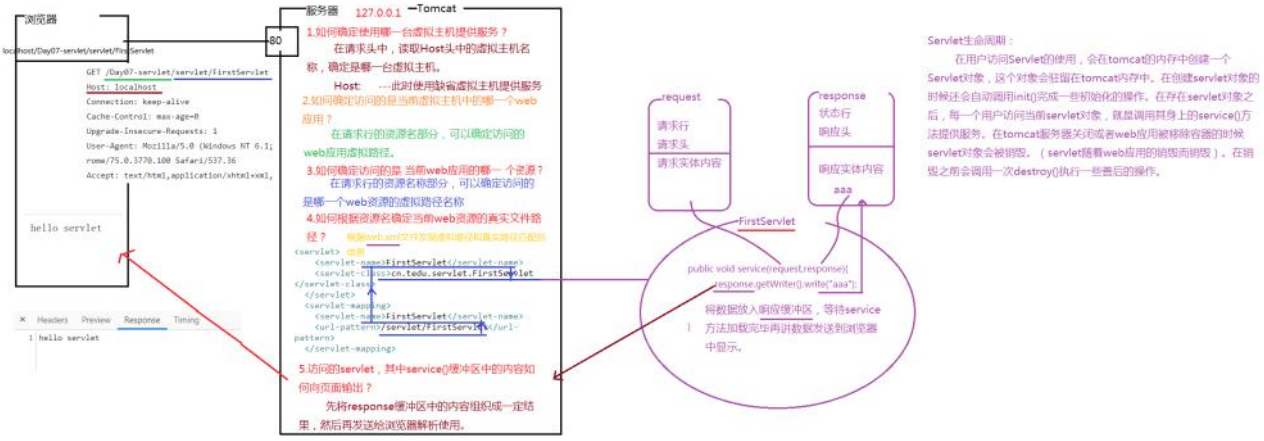
Servlet继承关系及运行过程

星期五,2019年7月12日　　11:46

1. Servlet继承关系

Servlet ---　基础servlet接口
|
|---GenericServlet　　通用servlet。(抽象类)，重写service()
　|
　|---HttpServlet　　是和HTTP协议相关的servlet。HttpServlet其中包含和HTTP协议相关的api操作，更善于HTTP相关开发。

2. Servlet运行过程



3. 拓展：request和response对象是何时产生的？

在servlet对象创建之后，request和response对象产生。

4. 问题：HttpServletRequest和HttpServletResponse都是接口，如何创建request和response对象？

两个接口并不能创建对象，而由各自接口的子实现类创建的对象，这两个子实现类分别为HttpServletRequestWrapper、HttpServletResponseWrapper

5. 问题：为什么静态资源也可以加载？

a. 分析：在地址栏中输入的index.html也会作为虚拟路径使用，它和servlet运行过程一致，可以在web.xml中没有与index.html匹配的路径，所以这个虚拟路径会交给DefaultServlet处理，这个servlet称之为缺省Servlet。缺省Servlet会将index.html与所有的静态资源相比较，一旦有匹配项则在浏览器中显示，如果没有匹配项则在页面中展示404页面。

b. 结论：DefaultServlet是为了展示静态资源和不存在的资源而使用的一个Servlet文件。

6. 修改Servlet模板

参考课前资料。

Servlet细节

星期五,2019年7月12日 15:35

1. Servlet细节--标签组成
   a. servlet组成是由两个标签servlet及servlet-mapping组成， 其中servlet称之为注册servlet标签，servlet-mapping称之为servlet映射标签。

```
<servlet>
    <servlet-name>FirstServlet</servlet-name>
    <servlet-class>cn.tedu.FirstServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>FirstServlet</servlet-name>
    <url-parttern>/servlet/FirstServlet</url-parttern>
</servlet-mapping>
```

   b. 如果有多个路径要映射在同一个servlet身上，那么需要重复书写映射servlet标签。修改其中的url-parttern标签。
2. servlet细节--通配映射

   如果有多个虚拟路径同时映射在一个servlet身上，这些路径还十分相似，这是就可以将他们写成一个通配的映射方式。

   通配映射的方式：

   i. 以"/"开头，以"/*"结尾。

   ii. 以"*.后缀"的形式使用。 *.do

```
<servlet-mapping>
    <servlet-name>
        AnyName
    </servlet-name>
    <url-pattern>
        *.do
    </url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
    <servlet-name>
        AnyName
    </servlet-name>
    <url-pattern>
        /action/*
    </url-pattern>
</servlet-mapping>
```

   /*表示将全部地址栏中的路径做拦截，所有的路径都会访问这个虚拟路径对应的Servlet。
3. servlet细节--映射顺序
   - 对于如下的一些映射关系：
     - Servlet1 映射到 /abc/*
     - Servlet2 映射到 /*
     - Servlet3 映射到 /abc
     - Servlet4 映射到 *.do （永远匹配级最低）

- 问题：
  - 当请求URL为“/abc/a.html”，“/abc/*”和“/*”都匹配，哪个servlet响应
    - Servlet引擎将调用Servlet1。
  - 当请求URL为“/abc”时，“/abc/*”和“/abc”都匹配，哪个servlet响应
    - Servlet引擎将调用Servlet3。
  - 当请求URL为“/abc/a.do”时，“/abc/*”和“*.do”都匹配，哪个servlet响应
    - Servlet引擎将调用Servlet1。
  - 当请求URL为“/a.do”时，“/*”和“*.do”都匹配，哪个servlet响应
    - Servlet引擎将调用Servlet2。
  - 当请求URL为“/xxx/yyy/a.do”时，“/*”和“*.do”都匹配，哪个servlet响应
    - Servlet引擎将调用Servlet2。
- 总结：Servlet映射路径与URL匹配程度越高，越优先使用。

  *.do匹配级别永远最低。

4. servlet细节--DefaultServlet (缺省Servlet)

   在浏览器中访问的静态资源，会经过web.xml文件，如果没有任何一个虚拟路径和这个资源名所匹配，则会交给DefaultServlet处理，DefaultServlet会使用这个资源名和所有的静态资源去匹配，如果可以匹配则使用静态资源展示，如果没有匹配的静态资源，则展示404页面。

5. Servlet的一些细节(5)
   - **如果在<servlet>元素中配置了一个<load-on-startup>元素，若标签中的数字为0或者大于0，那么WEB应用程序在启动时就会装载并创建Servlet的实例对象、以及调用Servlet实例对象的init()方法。**

     举例：

     ```
     <servlet>
         <servlet-name>invoker</servlet-name>
         <servlet-class>
             org.apache.catalina.servlets.InvokerServlet
         </servlet-class>
         <load-on-startup>2</load-on-startup>
     </servlet>
     ```

   - **用途：为web应用写一个InitServlet，这个servlet配置为启动时装载，为整个web应用创建必要的数据库表和数据。**

扩展：

在servlet的配置当中，<load-on-startup>2</load-on-startup>的含义是：

标记容器是否在启动的时候就加载这个servlet。

当值为0或者大于0时，表示容器在应用启动时就加载这个servlet；

当是一个负数时或者没有指定时，则指示容器在该servlet被选择时才加载。

正数的值越小，启动该servlet的优先级越高。

如果我们在web.xml中设置了多个servlet的时候，可以使用load-on-startup来指定servlet的加载顺序，服务器会根据load-on-startup的大小依次对servlet进行初始化。不过即使我们将load-on-startup设置重复也不会出现异常，服务器会自己决定初始化顺序。

6. Servlet的一些细节(7)—线程安全
   - 当多个客户端并发访问同一个Servlet时，web服务器会为每一个客户端的访问请求创建一个线程，并在这个线程上调用Servlet的service方法，因此service方法内如果访问了同一个资源的话，就有可能引发线程安全问题。
   - 如果某个Servlet实现了SingleThreadModel接口，那么Servlet引擎将以单线程模式来调用其service方法。
   - SingleThreadModel接口中没有定义任何方法，只要在Servlet类的定义中增加实现SingleThreadModel接口的声明即可。
   - 对于实现了SingleThreadModel接口的Servlet，Servlet引擎仍然支持对该Servlet的多线程并发访问，其采用的方式是产生多个Servlet实例对象，并发的每个线程分别调用一个独立的Servlet实例对象。
   - 实现SingleThreadModel接口并不能真正解决Servlet的线程安全问题，因为Servlet引擎会创建多个Servlet实例对象，而**真正意义上解决多线程安全问题是指一个Servlet实例对象被多个线程同时调用的问题**。事实上，在Servlet API 2.4中，已经将SingleThreadModel标记为Deprecated（过时的）。

   a. 解决线程安全问题的方式：

      i. 尽量少的使用全局变量，多使用局部变量。

      ii. 合理加锁。利用锁机制，锁住关键部分代码。因为锁会降低程序的执行效率，所以不能锁住过多的代码。

7. servlet细节--生命周期运行过程

   a. 生命周期

   在servlet被初次访问的时候，会在tomcat内存中创建一个对象，这个对象会驻留在tomcat内存中，后续访问的用户都会使用同一个servlet对象。在servlet对象创建的时候会调用init()执行初始化操作。每一次访问servlet都是希望调用其中的service()方法提供服务。在服务器关闭或者web应用移除容器的时候servlet对象会被销毁。在销毁之前会调用destroy()方法执行一些善后的操作。

   b. 运行过程

      i. 浏览器发出请求

      ii. 将域名解析为ip地址，访问对应ip地址的服务器，再根据端口号确定访问的是tomcat服务器。

      iii. 再根据请求头host确定访问哪一台虚拟主机。

      iv. 再根据请求行确定请求的web应用虚拟路径。

v. 再根据请求行确定请求的哪一个web资源虚拟路径。
vi. 再根据web.xml文件与虚拟路径相比较，如果有对应的servlet映射路径，则将创建一个servlet对象。
vii. 在执行service方法后，会将响应结果放入response缓冲区中。在服务器中组织成一定结构。
viii. 由服务器发送到浏览器中，浏览器解析执行。

request对象

星期五,2019年7月12日　　16:59

1. Request概述

   代表HTTP请求的对象。

   a. 请求的组成：

      一个请求行　GET /Day07-servlet/servlet/RequestDemo1　HTTP/1.1

      多个请求头

      一个空行

      请求实体内容

   b. 继承结构

      ServletRequest　---　request请求的最高级别的接口

      |

      |---HttpServletRequest　　在原有的接口之上实现了和HTTP协议相关的方法。这个接口更善于HTTP相关开。

2. HttpServletRequest的API操作：

   - 功能一：获取浏览器相关的信息

     getRequestURL方法 -- 返回客户端发出请求完整URL

     getRequestURI方法 -- 返回请求行中的资源名部分

     getQueryString方法 -- 返回请求行中的参数部分

     getRemoteAddr方法 -- 返回发出请求的客户机的IP地址

     getMethod -- 得到客户机请求方式

     !!getContextPath -- 获得当前web应用虚拟目录名称 -- 在写路径时不要将web应用的虚拟路径的名称写死, 应该在需要写web应用的名称的地方通过getContextPath方法动态获取

   - 功能二：获取请求头信息

     getHeader(name)方法 --- String

     getHeaders(String name)方法 --- Enumeration<String>

     getHeaderNames方法 --- Enumeration<String>

     getIntHeader(name)方法　--- int

     getDateHeader(name)方法 --- long(日期对应毫秒)