HTTP 与 Servlet 介绍

一、HTTP 协议介绍

HTTP 协议（Hypertext Transfer Protocol，超文本传输协议），是一个客户端请求和回应的标准协议，这个协议详细规定了浏览器和万维网服务器之间互相通信的规则。用户输入地址和端口号之后就可以从服务器上取得所需要的网页信息。

通信规则规定了客户端发送给服务器的内容格式，也规定了服务器发送给客户端的内容格式。客户端发送给服务器的格式叫“**请求协议**”；服务器发送给客户端的格式叫“**响应协议**”。

1、HTTP 协议的主要特点

1.）支持客户/服务器模式。

2.）简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、POST。每种方法规定了客户与服务器联系的类型不同。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。

3.）灵活：HTTP 允许传输任意类型的数据对象。正在传输的类型由 **Content-Type** 加以标记。

4.）无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。**（http1.1已经支持长连接）**

5.）无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

2、HTTP 之 URL

http（超文本传输协议）是一个基于请求与响应模式的、应用层的协议，常基于 TCP 的连接方式，绝大多数的 Web 开发，都是构建在 HTTP 协议之上的 Web 应用。

HTTP URL (URL是一种特殊类型的URI，包含了用于查找某个资源的足够的信息)的格式如下：

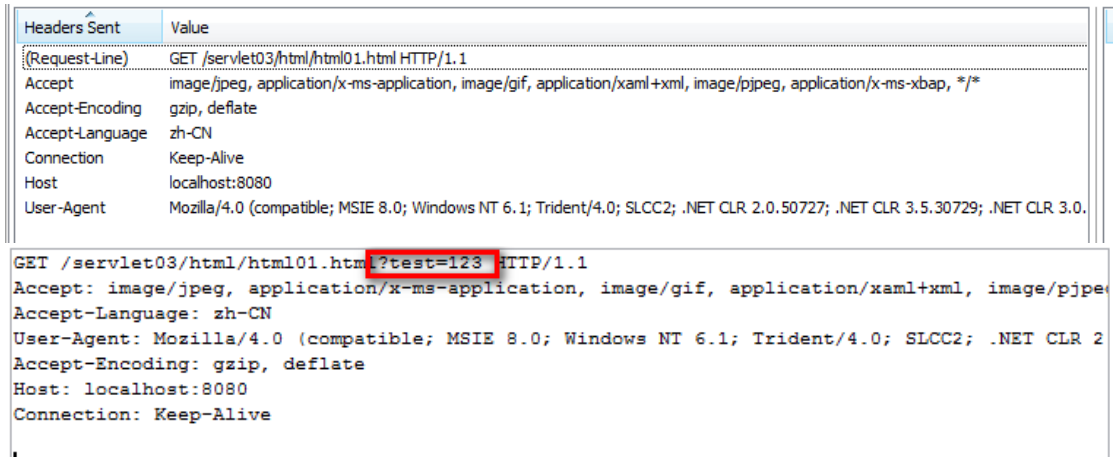
http://host[":"port][abs_path]

http表示要通过HTTP协议来定位网络资源；**host**表示合法的Internet主机域名或者IP地址；**port**指定一个端口号，为空则使用缺省端口80；**abs_path**指定请求资源的URI；如果URL中没有给出abs_path，那么当它作为请求URI时，必须以“/”的形式给出，通常这个工作浏览器自动帮我们完成。

二、HTTP 协议详解之请求篇

http请求由三部分组成，分别是：**请求行、请求头、请求正文**

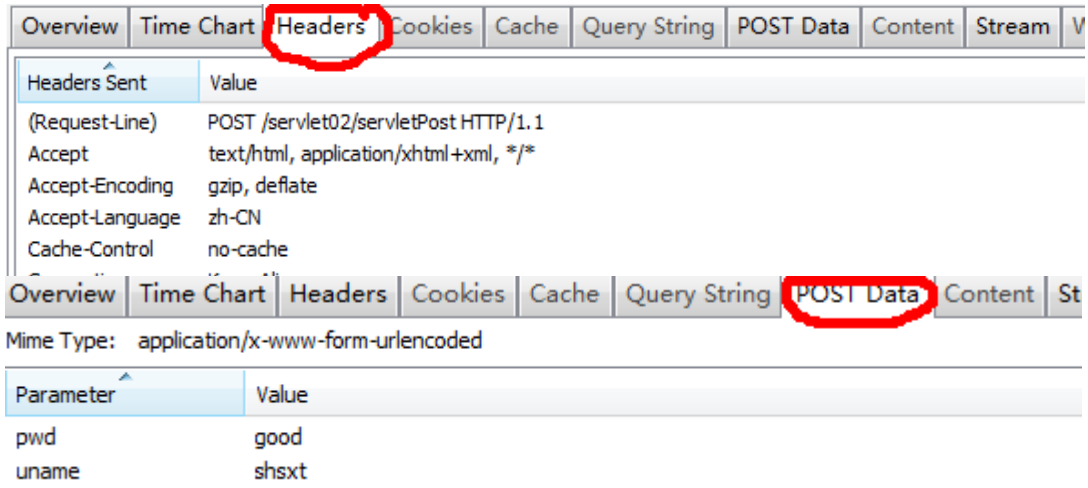
get



Headers Sent	Value
(Request-Line)	GET /servlet03/html/html01.html HTTP/1.1
Accept	image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, */*
Accept-Encoding	gzip, deflate
Accept-Language	zh-CN
Connection	Keep-Alive
Host	localhost:8080
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.

```
GET /servlet03/html/html01.html?test=123 HTTP/1.1
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpe
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2
Accept-Encoding: gzip, deflate
Host: localhost:8080
Connection: Keep-Alive
```

post



Overview | Time Chart | Headers | Cookies | Cache | Query String | POST Data | Content | Stream

Headers Sent	Value
(Request-Line)	POST /servlet02/servletPost HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Accept-Encoding	gzip, deflate
Accept-Language	zh-CN
Cache-Control	no-cache

Overview | Time Chart | Headers | Cookies | Cache | Query String | POST Data | Content | St

Mime Type: application/x-www-form-urlencoded

Parameter	Value
pwd	good
uname	shsxt

1、请求行

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如下：

```
Method Request-URI HTTP-Version CRLF
```

其中 Method 表示请求方法；Request-URI 是一个统一资源标识符；HTTP-Version 表示请求的 HTTP 协议版本；CRLF 表示回车和换行

请求方法（所有方法全为大写）有多种，各个方法的解释如下：

方法	解释
GET	**请求获取Request-URI所标识的资源**
POST	**在Request-URI所标识的资源后附加新的数据**
HEAD	请求获取由Request-URI所标识的资源的响应消息报头
PUT	请求服务器存储一个资源，并用Request-URI作为其标识

DELETE	请求服务器删除Request-URI所标识的资源
TRACE	请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT	保留将来使用
OPTIONS	请求查询服务器的性能，或者查询与资源相关的选项和需求

应用举例：

GET方法：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用GET方法向服务器获取资源，eg:GET /form.html HTTP/1.1 (CRLF)

POST方法：要求被请求服务器接受附在请求后面的数据，常用于提交表单。

```
POST /reg.jsp HTTP/ (CRLF)
Accept:image/gif,image/x-xbit,... (CRLF)
...
HOST:www.guet.edu.cn (CRLF)
Content-Length:22 (CRLF)
Connection:Keep-Alive (CRLF)
Cache-Control:no-cache (CRLF)
(CRLF)          //该CRLF表示消息报头已经结束，在此之前为消息报头
user=jeffrey&pwd=1234  //此行以下为提交的数据
```

2、请求头

3、请求体

三、HTTP 协议详解之响应篇

在接收和解释请求消息后，服务器返回一个HTTP响应消息。

HTTP响应也是由三个部分组成，分别是：**状态行、消息报头、响应正文**

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"273-1479693310000"
Last-Modified: Mon, 21 Nov 2016 01:55:10 GMT
Content-Type: text/html
Content-Length: 273
Date: Mon, 21 Nov 2016 01:55:19 GMT

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>hello html</h1>
<form action="../myservlet01" method="POST">
    <input type="text" name="test">
    <input type="submit" value="submit">
</form>
</body>
</html>
```

1、状态行

HTTP-Version Status-Code Reason-Phrase CRLF

其中，HTTP-Version 表示服务器 HTTP 协议的版本；Status-Code 表示服务器发回的响应状态代码；Reason-Phrase 表示状态代码的文本描述。

状态代码由三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息--表示请求已接收，继续处理
2xx：成功--表示请求已被成功接收、理解、接受
3xx：重定向--要完成请求必须进行更进一步的操作
4xx：客户端错误--请求有语法错误或请求无法实现
5xx：服务器端错误--服务器未能实现合法的请求

2、响应头

3、响应正文

响应正文即服务器返回的资源内容

常见响应码

200：请求成功，浏览器会把响应体内容（通常是 html）显示在浏览器中；

404：请求的资源没有找到，说明客户端错误的请求了不存在的资源；

500：请求资源找到了，但服务器内部出现了错误；

302：重定向，当响应码为 302 时，表示服务器要求浏览器重新再发一个请求，服务器会发送一个响应头 Location，它指定了新请求的 URL 地址；

四、消息头

HTTP消息由客户端到服务器的请求和服务器到客户端的响应组成。请求消息和响应消息都是由开始行（对于请求消息，开始行就是请求行，对于响应消息，开始行就是状态行），消息报头（可选），空行（只有CRLF的行），消息正文（可选）组成。
每一个报头域都是由名字+“：”+空格+值组成，消息报头域的名字是大小写无关的。

1、请求头

请求报头允许客户端向服务器端传递请求的附加信息以及客户端自身的信息。

常见的请求报头

Accept：Accept请求报头域用于指定客户端接受哪些类型的信息。
eg：Accept：image/gif，表明客户端希望接受GIF图象格式的资源；
Accept：text/html，表明客户端希望接受html文本。

Accept-Charset：Accept-Charset请求报头域用于指定客户端接受的字符集。
eg：Accept-Charset:iso-8859-1,gb2312.如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。

Accept-Encoding：Accept-Encoding请求报头域类似于Accept，但是它是用于指定可接受的内容编码。

eg：Accept-Encoding:gzip.deflate.如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。

Accept-Language：Accept-Language 请求报头域类似于 Accept，但是它是用于指定一种自然语言。

eg：Accept-Language:zh-cn.如果请求消息中没有设置这个报头域，服务器假定客户端对各种语言都可以接受

Host（发送请求时，该报头域是必需的）:Host请求报头域主要用于指定被请求资源的Internet主机和端口号，它通常从HTTP URL中提取出来的

eg:

我们在浏览器中输入：http://www.guet.edu.cn/index.html

浏览器发送的请求消息中，就会包含Host请求报头域，如下：

Host：www.guet.edu.cn

此处使用缺省端口号80，若指定了端口号，则变成：Host：www.guet.edu.cn:指定端口号

User-Agent：我们上网登陆论坛的时候，往往会看到一些欢迎信息，其中列出了你的操作系统的名称和版本，你所使用的浏览器的名称和版本，这往往让很多人感到很神奇，实际上，服务器应用程序就是从User-Agent这个请求报头域中获取到这些信息。User-Agent请求报头域允许客户端将它的操作系统、浏览器和其它属性告诉服务器。不过，这个报头域不是必需的，如果我们自己编写一个浏览器，不使用User-Agent请求报头域，那么服务器端就无法得知我们的信息了。

Refere：指明请求 从哪里来

如果是地址栏中输入地址访问的都没有该请求头

地址栏输入地址

(Request-Line)	GET /helloworld/myservlet HTTP/1.1
Accept-Encoding	gzip, deflate
Accept	image/jpeg, application/x-ms-application, image/gif, applic
Connection	Keep-Alive
Host	localhost:8080
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trider
Accept-Language	zh-CN

通过请求

Content-Length	11
Content-Type	application/x-www-form-urlencoded
Accept-Encoding	gzip, deflate
Referer	http://localhost:8080/helloworld/index.html
Accept	image/jpeg, application/x-ms-application, image/gif, application/xaml+xml,
Connection	Keep-Alive
Host	localhost:8080
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .N
Cache-Control	no-cache
(Request-Line)	POST /helloworld/myservlet HTTP/1.1
Accept-Language	zh-CN

可以看到，此时多了一个 Referer 的请求头，并且后面的值为该请求从哪里发出,百度竞价，只能从百度来的才有效果，否则不算;通常用来做统计工作、防盗链

2、响应头

响应报头允许服务器传递不能放在状态行中的附加响应信息，以及关于服务器的信息和对Request-URI所标识的资源进行下一步访问的信息。

常见的响应报头

Location：Location响应报头域用于重定向接受者到一个新的位置。Location响应报头域常用在更换域名的时候。<response.sendRedirect("http://www.baidu.com");>

Refresh：自动跳转（单位是秒）

```
<metahttp-equiv="Refresh"content="3;url=http://www.baidu.com">
```

五、Servlet 之 HelloWorld 体验

Servlet 是 Server 与 Applet 的缩写，是服务端小程序的意思。使用 Java 语言编写的服务器端程序，可以像生成动态的 WEB 页，Servlet 主要运行在服务器端，并由服务器调用执行，是一种按照 Servlet 标准来开发的类。 是 SUN 公司提供的一门用于开发动态 Web 资源的技术。（言外之意：要实现 web 开发，需要实现 Servlet 标准）

Servlet 本质上也是 Java 类，但要遵循 Servlet 规范进行编写，没有 main()方法，它的创建、使用、销毁都由 Servlet 容器进行管理(如 Tomcat)。（言外之意：写自己的类，不用写 main 方法，别人自动调用）

Servlet 是和 HTTP 协议是紧密联系的，其可以处理 HTTP 协议相关的所有内容。这也是 Servlet 应用广泛的原因之一。

提供了 Servlet 功能的服务器，叫做 Servlet 容器，其常见容器有很多，如 Tomcat, Jetty, WebLogic Server, Websphere, JBoss 等等。

作用：接收请求数据， 处理请求， 完成响应

实现步骤：

1、 new Dynamic web Project （新建动态 web 项目）

2、 new class（新建自己的类）

3、 extends HttpServlet （满足 Servlet 规范）

4、 重写 service()（增加处理程序）

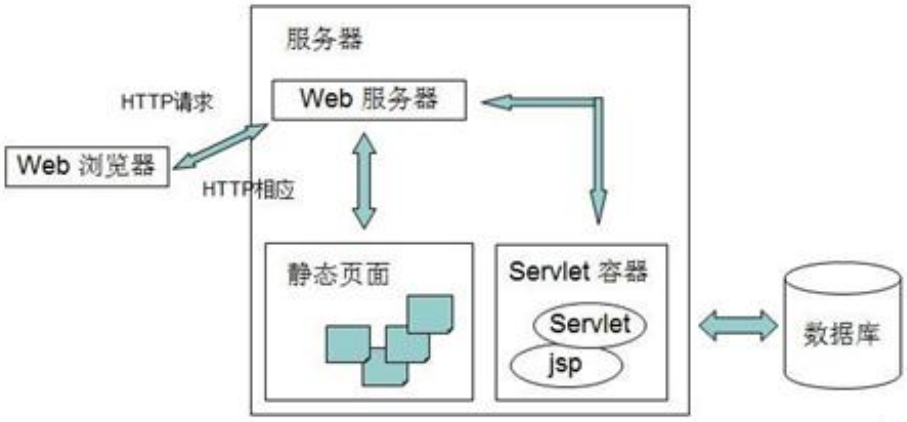
5、 配置 web.xml(配置映射文件)

6、 发布项目

7、 启动服务器

8、 访问

1、请求处理工作原理



执行过程

- 客户端发出请求
- 根据 Web.xml 文件的配置，找到<url-pattern>对应的<servlet-mapping>
- 读取<servlet-mapping>中<servlet-name>的值
- 找到<servlet-name>对应的<servlet-class>
- 找到该 class 并加载执行该 class，返回结果
- 由 Web 服务器将结果响应给客户端

2、Servlet 的实现方式

SUN 公司定义了一个接口和两个实现类，Servlet 接口，GenericServlet 和 HttpServlet 实现类，其中后者是前者的子类，子类在原有基础上添加了一些 HTTP 协议处理方法，HttpServlet 比 GenerricServlet 功能更强大，所以我们一般将自己的类继承自 HttpServlet，并重写 doGet 方法和 doPost 方法，或重写 Service 方法。以下三种方式都可以

1、 实现 Servlet 接口

2、 继承 GenericServlet 类

3、 继承 HttpServlet 类

由于我们是针对 HTTP 协议的操作，所以直接继承 **HttpServlet** 类即可

六、Form 提交请求 servlet

form_get.html

```
<formaction="servletGet"method="get">
        <h1>登录页面</h1>
      <table>
                        <tr>
                                <td>用户名</td>
                                <td><inputtype="text"name="uname"></td>
                        </tr>
                        <tr>
                                <td>密码</td>
                                <td><inputtype="password"name="pwd"></td>
                        </tr>
                        <tr>
                                <tdcolspan="2">
                                        <inputtype="submit"value="登录">
                                        <inputtype="reset"value="重置">

                                </td>
                        </tr>
                </table>
</form>
```

FormGet.java

```
publicclassFormGetextendsHttpServlet{
        @Override
        protectedvoiddoGet(HttpServletRequestreq, HttpServletResponseresp)
throwsServletException, IOException {
                // TODO Auto-generated method stub
                String uname=req.getParameter("uname");
                String pwd=req.getParameter("pwd");
                if(uname.equals("shsxt")&&pwd.equals("good")){
                        System.out.println(uname+"用户,登录成功");
                }else{
                        System.out.println("用户名或密码错误!!!");
                }
        }
}
```

七、Servlet 的生命周期

1、Servlet 的生命周期

Servlet 没有 main()方法，不能独立运行，它的运行完全由 Servlet 引擎来控制和调度。所谓生命周期，指的是 servlet 容器如何创建 servlet 实例、分配其资源、调用其方法、并销毁其实例的整个过程

在如下两种情况下会进行对象实例化。

阶段一: 实例化（就是创建 servlet 对象,调用构造器）

第一种情况：

当请求到达容器时，容器查找该 servlet 对象是否存在，如果不存在，才会创建实例。

第二种情况：

容器在启动时，或者新部署了某个应用时，会检查 web.xml 当中，servlet 是否有 load-on-starup 配置。如果有，则会创建该 servlet 实例(仅仅被创建一次 默认为单例)。

load-on-starup 参数值越小，优先级越高（最小值为 0，优先级最高）。

配置方式如下:

```
<servlet>
<servlet-name>login</servlet-name>
<servlet-class>com.shsxt.controller.LoginController</servlet-class>
<!--容器启动时实例化当前 servlet -->
<load-on-startup>1</load-on-startup>
</servlet>
```

阶段二： 初始化

为 servlet 分配资源，调用 init(ServletConfig config);方法 config 对象可以用来访问 servlet 的初始化参数,父类方法 init 存在两个,**如果重写父类所有 init 方法，初始化时仅仅会执行带参 init 方法 不会执行无参 init 方法**

```
<servlet>
<servlet-name>login</servlet-name>
<servlet-class>com.shsxt.controller.LoginController</servlet-class>
<!-- servlet 初始化参数配置 -->
<init-param>
<param-name>encode</param-name>
<param-value>utf-8</param-value>
```

```
</init-param>
<!--容器启动时实例化当前 servlet -->
<load-on-startup>1</load-on-startup>
</servlet>
```

带参 init 方法获取初始化参数值

```
@Override
publicvoidinit(ServletConfig config)throwsServletException{
System.out.println("编码:"+config.getInitParameter("encode"));
}
```

阶段三： 就绪/调用(服务)

有请求到达容器，容器调用 servlet 对象的 service()方法,**处理请求的方法在整个声明周期中可以被多次调用；**

HttpServlet 的 service()方法，会依据请求方式来调用 doGet()或者 doPost()方法。但是，这两个 do 方法默认情况下，会抛出异常，需要子类去 override。

阶段四： 销毁

容器依据自身的算法，将不再需要的 servlet 对象删除掉。

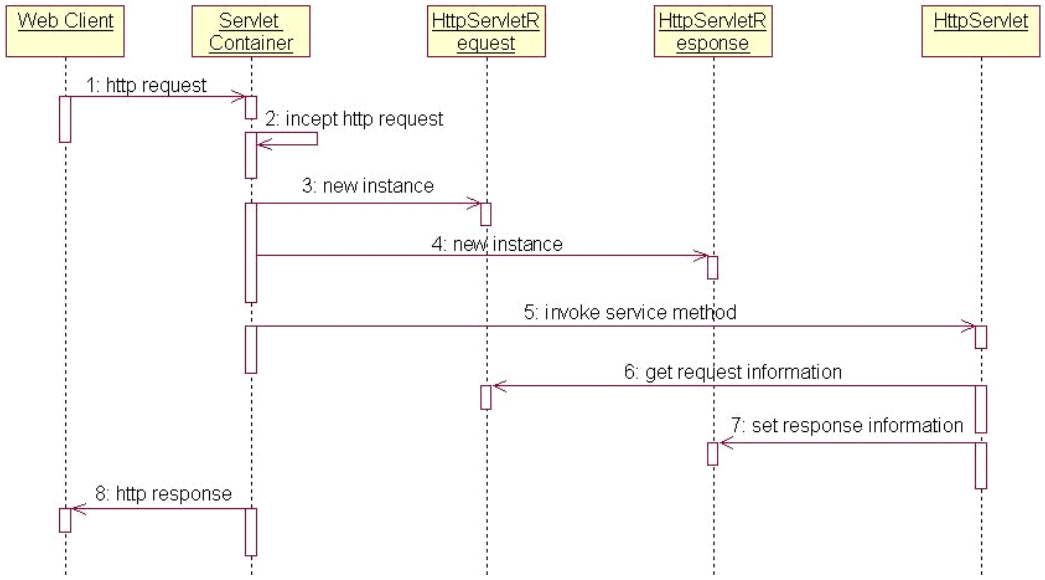
在删除之前，会调用 servlet 对象的 destroy()方法。

destroy()方法用于释放资源。

在 servlet 的整个生命周期当中，init,destroy 只会执行一次，而 service 方法会执行多次。

```
@Override
publicvoiddestroy(){
System.out.println("LoginController 被销毁...");
}
```

Servlet的生命周期，简单的概括这就分为四步：servlet类加载--->实例化--->服务--->销毁。下面我们描述一下Tomcat与Servlet是如何工作的,看看下面的时序图.



1、Web Client 向Servlet容器（Tomcat）发出Http请求

2、Servlet容器接收Web Client的请求

3、Servlet容器创建一个HttpRequest对象，将Web Client请求的信息封装到这个对象中

4、Servlet容器创建一个HttpResponse对象

5、Servlet容器调用HttpServlet对象的service方法，把HttpRequest对象与HttpResponse对象作为参数，传给HttpServlet对象

6、HttpServlet调用HttpRequest对象的有关方法，获取Http请求信息

7、HttpServlet调用HttpResponse对象的有关方法，生成响应数据

8、Servlet 容器把 HttpServlet 的响应结果传给 Web Client

2、Servlet 配置

基本配置

```
   <servlet>
        <servlet-name>first<servlet-name>
        <servlet-class>com.shsxt.web.HelloWorld</servlet-class>

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

<init-param> 配置初始化参数

```
<servlet>
      <servlet-name>first<servlet-name>
      <servlet-class>com.shsxt.web.HelloWorld</servlet-class>
      <init-param>
              <param-name>param1</param-name>
              <param-value>1</param-value>
      </init-param>
      <init-param>
              <param-name>param2</param-name>
              <param-value>2</param-value>
      </init-param>
      </servlet>
<servlet-mapping>
      <servlet-name>first</servlet-name>
      <url-pattern>/servlet.do</url-pattern>
</servlet-mapping>
```

<load-on-startup> 自启动

```
<servlet>
      <servlet-name>first<servlet-name>
      <servlet-class>com.shsxt.web.HelloWorld</servlet-class>
      <init-param>
              <param-name>param1</param-name>
              <param-value>1</param-value>
      </init-param>
      <init-param>
              <param-name>param2</param-name>
              <param-value>2</param-value>
      </init-param>
      <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>first</servlet-name>
      <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

servlet-name：Servlet对象的名称

servlet-class：创建Servlet对象所要调用的类

param-name：参数名称

param-value：参数值

load-on-startup：Servlet容器启动时加载Servlet对象的顺序

servlet-mapping/servlet-name：要与servlet中的servlet-name配置节内容对应

url-pattern：客户访问的Servlet的相对URL路径

说明：

1、url-pattern　可以配多个（一个servlet可以通过多个url-pattern访问）

2、当多个 servlet 配置成了同一个 url-pattern，报错

java.lang.reflect.InvocationTargetExceptionion

3、通配符　“*”

- 只能放在最前面或最后面，　* 不能单独存在
- 越精确越优先

可以在<url-pattern>中使用通配符，所谓通配符就是星号“*”，星号可以匹配任何 URL 前缀或后缀，使用通配符可以命名一个 Servlet 绑定一组 URL，例如：

```
<url-pattern>/servlet/*<url-patter>：/servlet/a、/servlet/b，都可匹配
/servlet/*；
<url-pattern>*.do</url-pattern>：/abc/def/ghi.do、/a.do，都可匹配*.do；
<url-pattern>/*<url-pattern>：匹配所有 URL；
```