# request学习

# 学习目标

1.能够使用工具查看HTTP协议内容

2.能够理解HTTP协议请求内容

3.能够使用Request对象获取HTTP协议请求内容

4.能够处理HTTP请求参数的乱码问题

5.能够使用Request域对象

6.能够使用Request对象做请求转发

7.能够完成登录案例

**反馈**

|  |
| --- |
| 需要扩展更多知识。又没有作业。内容少了  后面就有作业. |
| 还有中文的API也好像没法  放到今天的资料里面 |
| 问题一：在idea配置tomcat点击运行的时候报错C:\Users\admin\.IntelliJIdea2017.3\config\options\javaeeAppServers.xml前言不能有内容，这个怎么解决，而文件删除过后既然可以运行  百度不到答案,这个文件是配置了idea绑定的tomcat相关信息(本机,C:\Users\songyu\.IntelliJIdea2017.3\config\options)  问题二： java.lang.UnsatisfiedLinkError: F:\install\apache-tomcat-8.5.27-windows-x64\apache-tomcat-8.5.27\bin\tcnative-1.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform什么意思，为什么不影响程序的运行  注意,操作系统,idea,jdk都使用64位,若系统是32位,其他也都选择32位 |
| 使用@WebServlet(urlPatterns = "XXX"),当浏览器进入该路径后,是如何自动加载class文件的?还是要通过反射的?老师可以具体讲下吗?  Servlet的运行原理  <url:http://localhost:8080/day35/DemoServlet>  服务器会截取到/day35/DemoServlet  服务器扫描web.xml方式:服务器首先会进入day35资源目录,会根据/DemoServlet的映射路径去查找web.xml中的配置查找出来一个类全名com.itheima.servlet.DemoServlet,服务器根据类全名使用反射实例servlet对象.  服务器扫描注解方式:服务器在启动的时候获取所有webSerlvlet注解放到内存中,当前用户访问的就会扫描对应注解并获取对应的类全名,使用反射实例servlet对象.  服务器就会封装所有请求数据到request,所有响应数据到response,调用servlet对象的service方法传递request和response处理请求. |
| 能说一下doget和dopost这两个方法吗?  HttpServlet的处理请求方法,疑问是处理请求不是service方法,怎么是doget和dopost  看httpServlet源码,httpServlet重写service方法,判断请求类型是Get类型调用子类重写的doGet,是post类型,调用子类doPost(); |
| 有时间给我们补一补nginx.  整理资料,后面项目实战第4天讲解nignx. |

# http协议

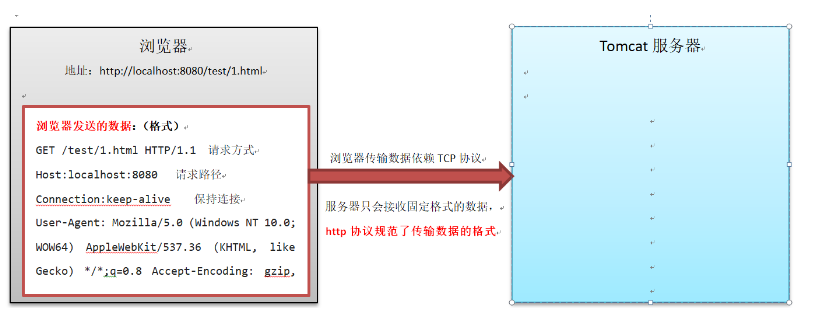
## http协议的基本概念

### http协议的简介

​ 超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。1960年美国人Ted Nelson构思了一种通过计算机处理文本信息的方法，并称之为超文本（hypertext）,这成为了HTTP超文本传输协议标准架构的发展根基。Ted Nelson组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force ）共同合作研究，最终发布了一系列的RFC，其中著名的RFC 2616定义了HTTP 1.1。​

### http协议的作用

​ 互联网的数据传输底层依赖tcp协议，http协议架构在tcp协议之上，**它规范了传输数据的内容和格式**。



### HTTP协议的应用场景

**刷屏软件**



刷屏软件的原理其实通过观察12306查阅剩余票数需要发出的参数，然后自己编程拼接服务器所需要的参数发给服务器，然后接受服务器的数据。如果需要查看12306发出的请求参数格式，那么就必须学习http协议。

**智能家居**



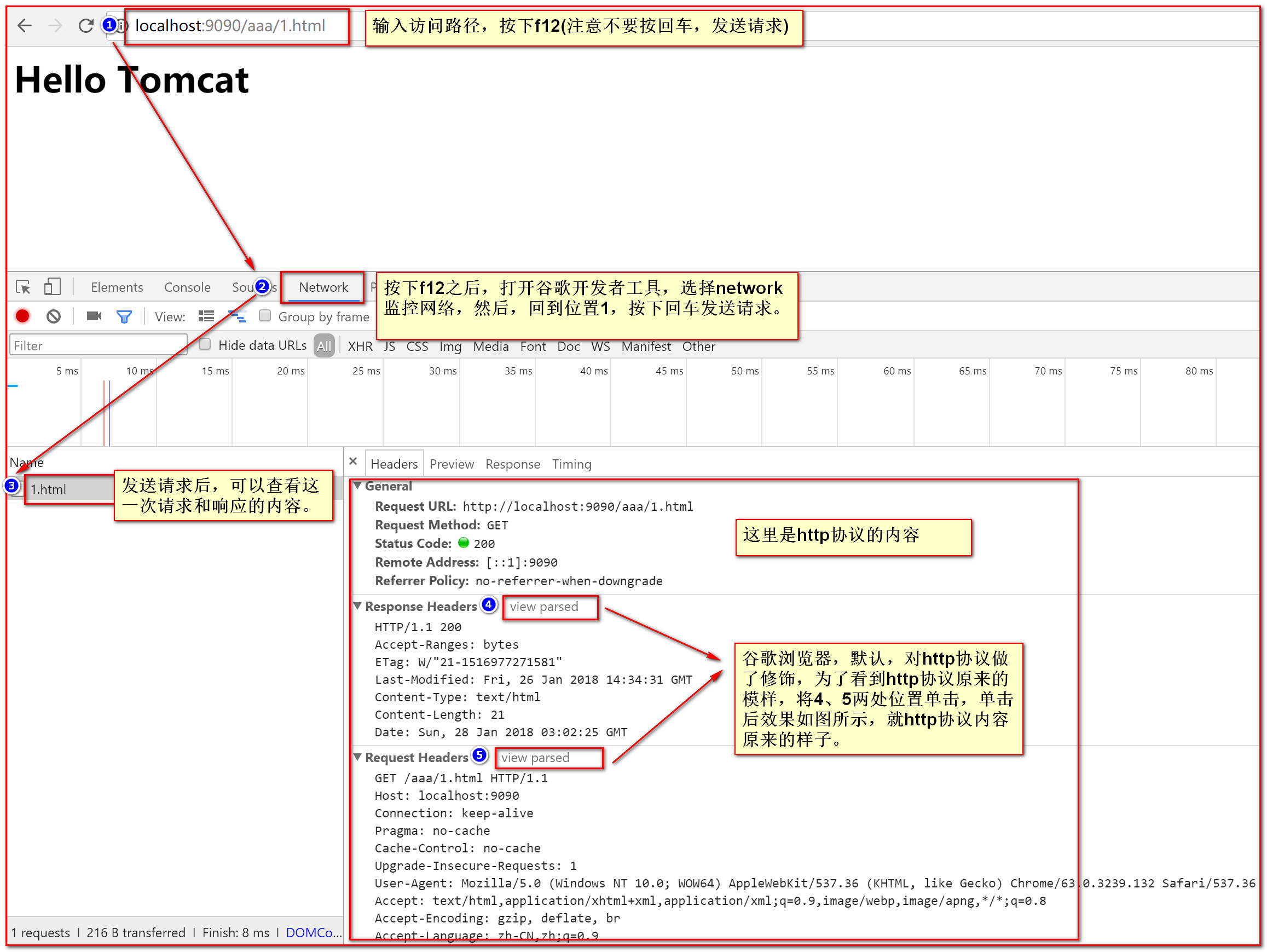
只能家居在我们的生活中的地位越来越重了，手机是如何操作我们的家庭设备的呢？其实也是手机软件通过拼接参数，然后发送给电话煲的，这个过程也是需要使用到http协议的。不管是刷屏软件还是智能家居都需要知道服务器要何种格式的数据，那么我们怎么才能知道呢？我们可以查看浏览器与服务器的通讯过程。

## http协议数据查看方式

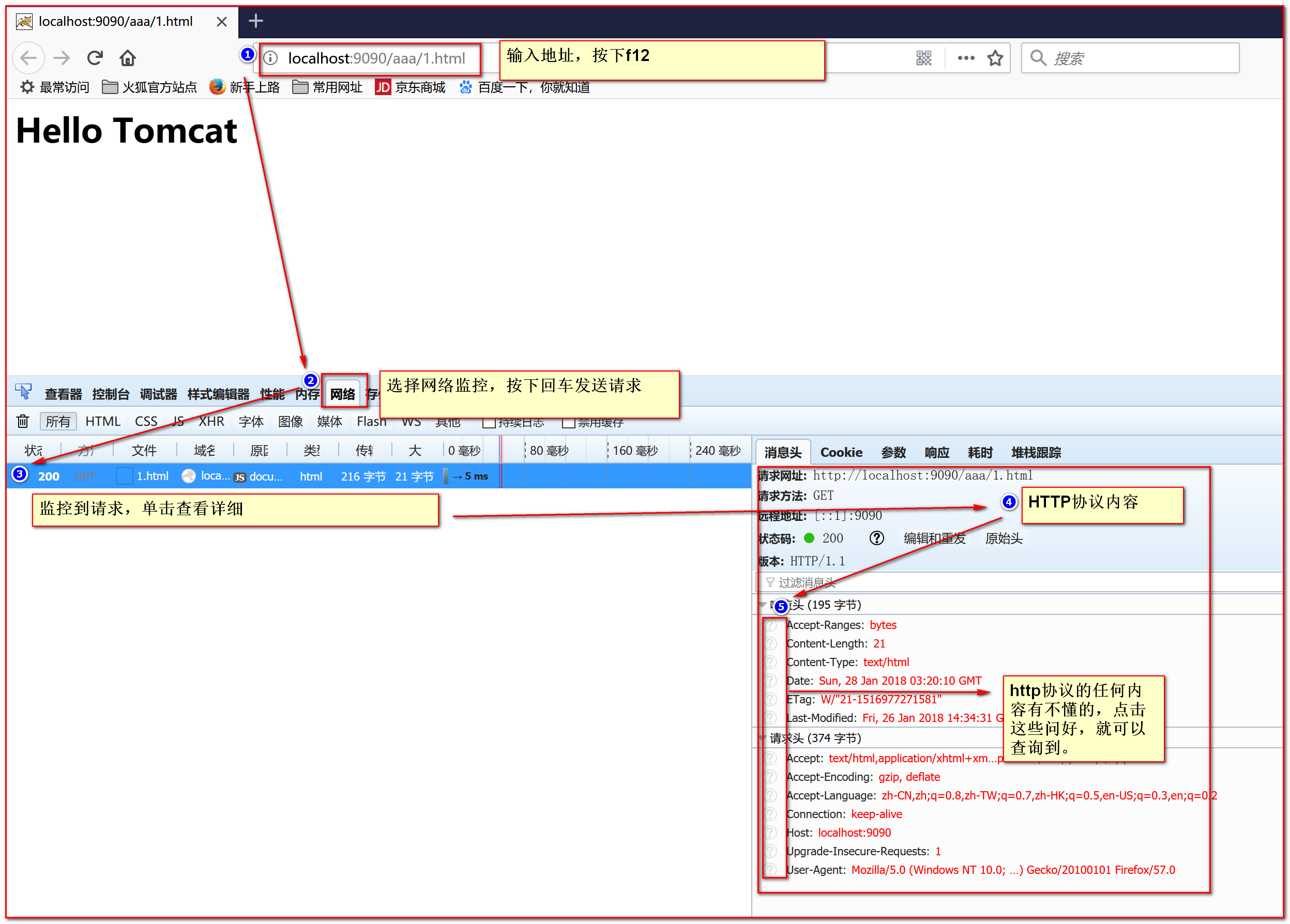
### 浏览器查看浏览器与服务器通讯过程

​ 接下来我们准备开始学习HTTP协议的具体内容，既然要学习内容，必然要先能看的具体的内容，这就需要借助查看http协议内容的工具了，工具一般有两个：

#### chrome(谷歌)——按键f12：



#### firefox(火狐)——按键f12：

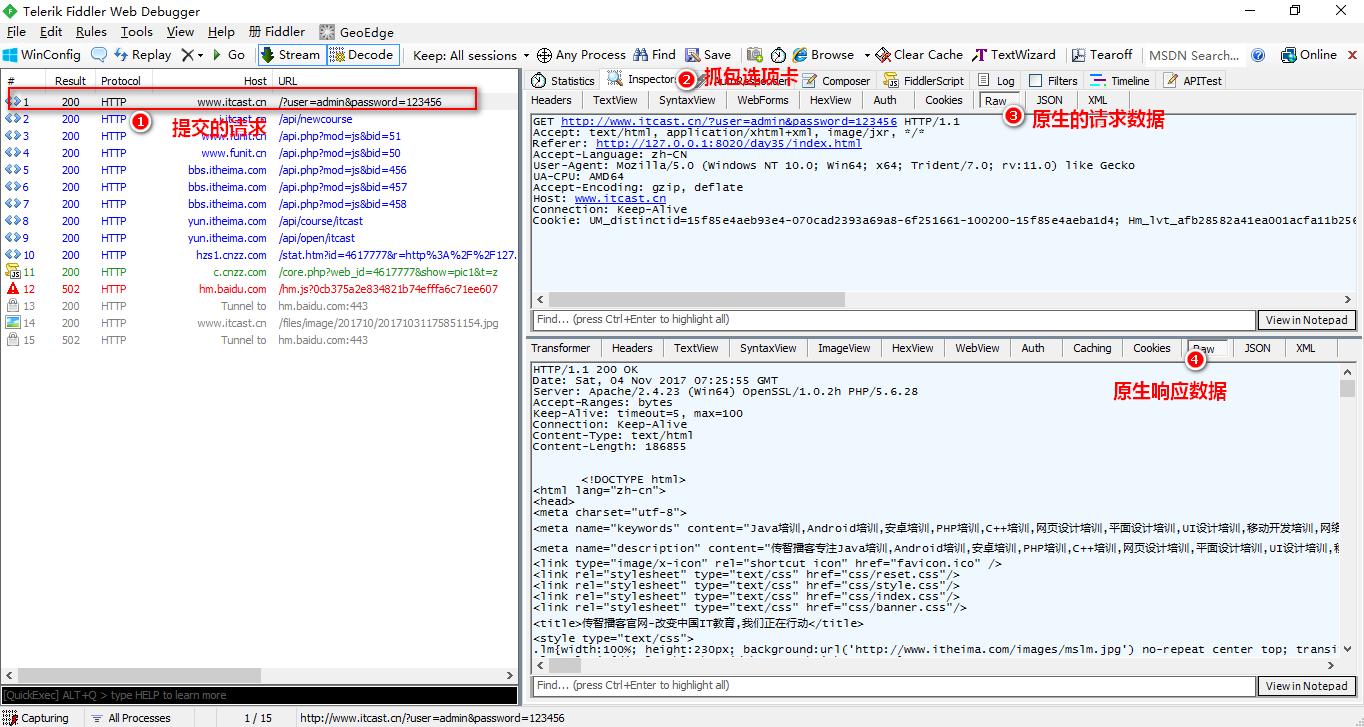


了解了如何使用工具查看http协议，那么接下来，我正式开始学习里面的内容：

### 专业Fiddler抓包工具软件查看

使用专业Fiddler抓包工具查看，是微软的开发。

优点：可以查看最原始的http协议数据。(就是浏览器与服务器真正通信的原始数据)



## 请求信息的组成部分

学习http请求信息的组成，我们的目的是，了解浏览器将那些数据，按照什么样的格式发送给服务器。​ http请求一共分为三个部分：请求行，请求头，请求体。接下来我们要看看具体的内容。

​注：演示的时候，我们以谷歌浏览器为例。

### http协议数据查看准备

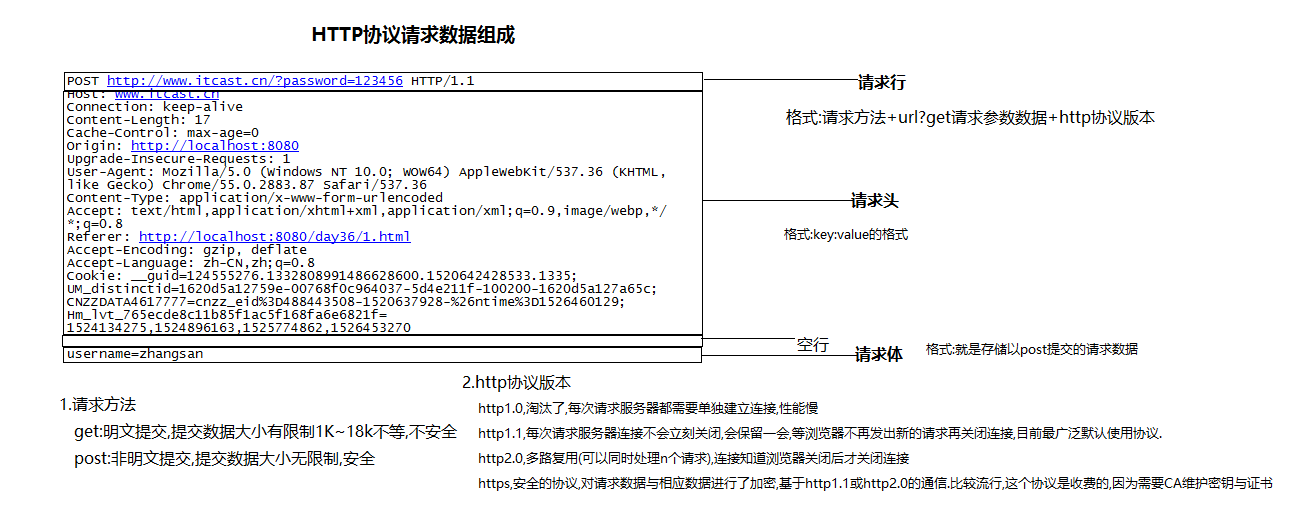
1. **创建一个工程day02，在web文件夹中准备一个1.html，内容为：**

<form action="1.html?name=admin" method="post">  
<input type="text" name="username"><br>  
<input type="submit" value="提交">  
</form>

1. **地址栏输入**<http://localhost:8080/day02/1.html>**测试效果（get请求）**
2. **启动服务器测试，访问1.html**

​

### Http请求数据的组成



### 请求行

1. **请求行介绍**

**请求行：POST /day02/1.html HTTP/1.1**

**POST：请求方式**

**/day02/1.html：访问路径**

**HTTP/1.1 ：http协议的版本**

1. **问：get和post请求方式的区别？**

答：

|  | **get请求方式** | **post请求方式** |
| --- | --- | --- |
| 数据长度是否有限制 | 是 | 否 |
| 是否包含请求体 | 否 | 是 |
| 数据是否暴露再地址栏 | 是 | 否 |

​ 对于请求行的学习，同学们可以读懂请求行内容，和了解get和post请求方式的区别即可。

​ 接下来我们再来学习第二个部分请求头的内容。

### 请求头

1. **请求头内容：**

Host: localhost:9090 （服务器的域名）  
Connection: keep-alive （网络连接是持久的）  
Content-Length: 12 （发送数据的大小，单位字节）  
Cache-Control: max-age=0 （设置缓存数据的存活时间，单位秒）  
Origin: http://localhost:9090 （指示了请求来自于哪个站点——服务器名称）  
Upgrade-Insecure-Requests: 1 （如果存在更安全的响应，客户端优先选择加密及带有身份验证的响应）  
Content-Type: application/x-www-form-urlencoded （发送数据的媒体类型——发挥作用类似后缀名：.mp3 .avi）  
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 （当前系统和浏览器版本）  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8  
Referer: http://localhost:9090/day01/1.html  
Accept-Encoding: gzip, deflate, br  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: JSESSIONID=98006861B5044ACC8D5C7840C93C17DA

想要知道这些请求头是干什么的，可以使用火狐浏览器的功能，详细可以回看——浏览器查看浏览器与服务器通讯过程。

在这所有的请求头中，比较重要的有两个：

Referer: http://localhost:9090/day01/1.html  
上一次请求的地址  
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36  
当前用户的系统和浏览器版本。

请求头中，同学们熟悉这两个即可。接下来最后就是请求体了。

### 请求体

1. **请求体内容：**

注意：get请求没有请求体，post请求有请求体

好到此为止，虽然我们了解了http协议请求相关的内容，但是，光了解不够，我们还需要可以在服务器获取这些数据，才能处理用户的请求，因此，接下来，我们需要学习和使用一个新的对象——Request。

## request对象作用

1. 获取请求信息数据
2. 获取工程名字
3. 进行请求转发页面跳转
4. 是一个域对象

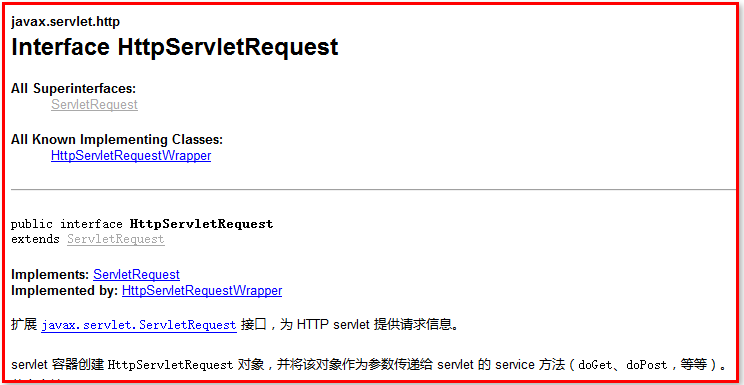
# request对象获取请求信息

## request对象的基本概念

1. **什么是HttpServletRequest？**

**HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求中的所有信息都封装在这个对象中，开发人员通过这个对象的方法，可以获得客户这些信息。**

以下是API文档中的介绍：



通过文档阅读可以注意到一个细节HttpServletReuqest不是相关请求的顶级接口，它继承自父接口——ServletRequest：



**通过Request对象进行的常用操作:**

**获取客户机信息**

**获取请求头信息**

**获取请求参数**

**利用请求域传递数据**

1. HttpServlet有许多的API我们从何学起？

答：我们按照学习http请求组成部分，按——请求行、请求头、请求体顺序学习。

## request获取请求行信息

### 请求行的组成元素（通过request获取请求行数据）

      在http协议中我已经看到了http协议中请求行的内容——分为请求方式、请求路径、协议版本。在HttpServletRequest概述中我们知道浏览器与请求相关的数据封装在request中，因此，接下来我们学习如何使用request对象获取请求行的数据。

### API介绍

1. String getMethod() 获取请求方式的类型
2. String getRequestURI()  获取请求行中的资源名部分
3. StringBuffer getRequestURL()  获取客户端发出请求完整URL  
       
   注：  
    uri: 统一资源标识符,用来标识一个资源,资源路径。(相当于身份证)  
    url: 统一资源定位符,是一种具体的URI,可以用来标识一个资源.并且指明了如何定位一个资源. （相当于身份证中的地址）
4. String getProtocol()  获取当前协议的名称和版本
5. String getRemoteAddr() 获取IP地址
6. int getLocalPort() 获取端口

### 使用步骤

1. 创建DemoServlet
2. 在DemoServlet中的doGet或者doPost方法的参数列表，已经包含了request对象，调用方法即可。
3. 将数据打印在控制台

### 演示代码

|  |
| --- |
| **package** com.itheima.requestdata;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"RowServlet"**, urlPatterns = **"/RowServlet"**) **public class** RowServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//使用request获取请求行数据  //格式 POST http://www.itcast.cn/?password=123456 HTTP/1.1  //1.获取请求方法类型* String method = request.getMethod();  *//2.获取请求的url* String url = request.getRequestURL().toString();  *//3.获取请求的uri* String uri = request.getRequestURI();  *//4.协议版本* String protocol = request.getProtocol();  *//5.客户端的ip* String ip = request.getRemoteAddr();  *//本地使用localhost访问,服务器会返回ip:0:0:0:0:0:0:0:1,ipV6,需要重新使用ipV4地址访问就可以   //打印出来* System.***out***.println(**"method:"**+method);  System.***out***.println(**"url:"**+url);  System.***out***.println(**"uri:"**+uri);  System.***out***.println(**"protocol:"**+protocol);  System.***out***.println(**"ip:"**+ip);   } } |

效果一：测试地址——<http://localhost:9090/day02/demo>

获取请求方式的类型:GET  
获取请求行中的资源名部分:/day02/demo  
获取客户端发出请求完整URL:http://localhost:9090/day02/demo  
获取当前协议的名称和版本:HTTP/1.1  
获取IP地址:0:0:0:0:0:0:0:1  
获取端口:9090

效果二：测试地址——<http://127.0.0.1:9090/day02/demo>

获取请求方式的类型:GET  
获取请求行中的资源名部分:/day02/demo  
获取客户端发出请求完整URL:http://127.0.0.1:9090/day02/demo  
获取当前协议的名称和版本:HTTP/1.1  
获取IP地址:127.0.0.1  
获取端口:9090

问：为什么要测试两次？

答：同学们观察下两次打印的IP地址会发现不一样，因此，注意，Localhost和127.0.0.1效果一致，但是localhost默认使用ipv6本机地址——0:0:0:0:0:0:0:1，而127.0.0.1是ipv4的本机地址。

## request获取请求头信息

### 获取请求头信息常用的方法

#### API介绍

1. String getHeader(String name)  以String 的形式返回指定请求头的值
2. Enumeration getHeaderNames() 返回此请求包含的所有头名称的枚举

#### 使用步骤

1. 创建DemoServlet2
2. 在DemoServlet2中的doGet或者doPost方法的参数列表，已经包含了request对象。因此，调用方法即可。
3. 将结果打印在控制台

#### 演示代码

package cn.itcast.web;  
​  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.util.Enumeration;  
​  
@WebServlet(name = "DemoServlet2",urlPatterns = "/demo2")  
public class DemoServlet2 extends HttpServlet {  
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
        doGet(request,response);  
   }  
​  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
​  
        /\*\*  
         \* 1.   String getHeader(String name)  
                以 String 的形式返回指定请求头的值  
         \* \*/  
        String header = request.getHeader("User-Agent");  
        System.out.println("getHeader:"+header);  
        System.out.println();  
        /\*\*  
         \* 2. Enumeration getHeaderNames()  
               返回此请求包含的所有头名称的枚举  
         \* \*/  
        Enumeration<String> headerNames = request.getHeaderNames();  
        while (headerNames.hasMoreElements()){  
            System.out.println("getHeaderNames:"+headerNames.nextElement());  
       }  
​  
   }  
}

效果：

getHeader:Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36  
​  
getHeaderNames:host  
getHeaderNames:connection  
getHeaderNames:cache-control  
getHeaderNames:user-agent  
getHeaderNames:upgrade-insecure-requests  
getHeaderNames:accept  
getHeaderNames:accept-encoding  
getHeaderNames:accept-language  
getHeaderNames:cookie  
​

### 请求头referer(重点)

#### referer作用

利用referer获取当前请求的上一个请求资源

代码

|  |
| --- |
| **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    //响应头-referer,作用获取当前请求资源的上一个资源地址  // 特点：如果没有上一个资源就返回null  //1.获取请求头referer数据  String preUrl = request.getHeader("referer");  System.*out*.println("上一个资源"+preUrl);    } |

test.html

|  |
| --- |
| <body>  <a href=*"/day38/referer"*>refererServlet</a>  </body> |

先浏览test.html再轻轻refererServlet，输出的效果

|  |
| --- |
| 上一个资源http://192.168.101.77:8080/day38/test.html |

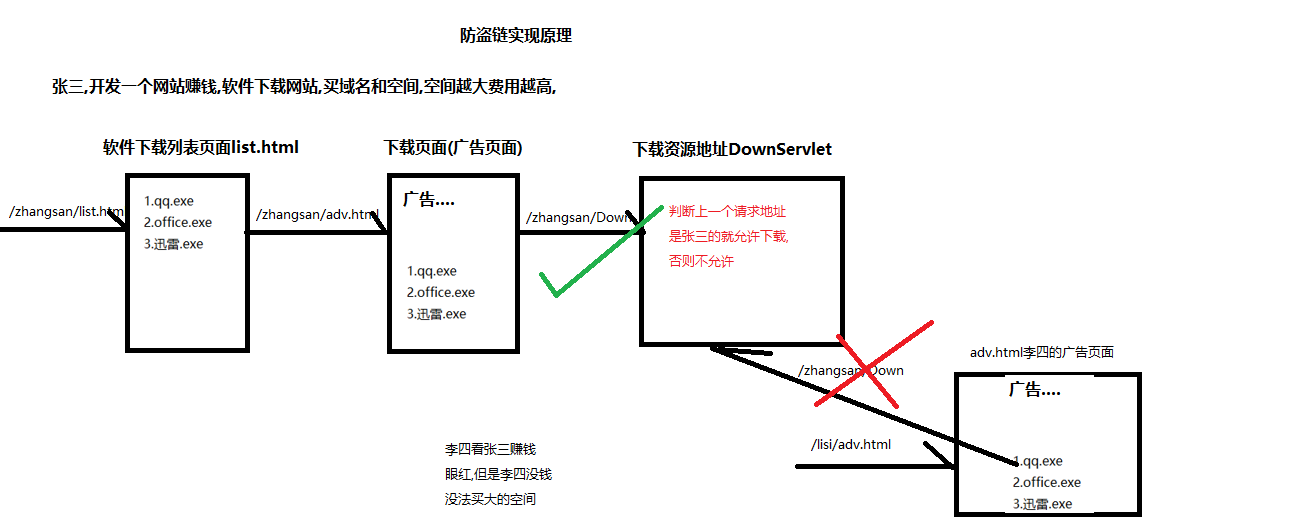
#### 案例:referer应用——防盗链功能

###### 防盗链案例情景

张三自己开发了软性下载的一个网站，然后通过购买域名和大空间将网站发布出去供互联网用户浏览访问，来到张三网站的用户先浏览广告再下载软件资源，这样张三通过广告就赚钱了，在这里张三购买的空间是比较贵的，空间越大费用越高。李四看到张三赚到了钱，但是自己不想花钱买大空间，于是自己买了域名和小空间，软件资源的下载盗取张三的下载地址，这样用户来到李四网站依然看到广告，但是真实下载软件的资源是从张三的空间下载的。这样李四就不用买大空间，就盗取了张三网站的资源。所以张三的网站要针对这样的情况进行防盗。

###### 防盗原理实现

张三的网站在请求资源下载的时候判断当前请求的上一个资源是否是来自张三自己网站的资源。如果上一个资源是自己网站的资源就允许下载，否则不允许。



zhangsan的资源（张三的网站资源）如下

list.html代码

|  |
| --- |
| <body>  <a href=*"adv.html"*>1.office.exe</a><br/>  <a href=*"adv.html"*>2.qq.exe</a><br/>  <a href=*"adv.html"*>3.迅雷.exe</a><br/>  </body> |

/zhangsan/adv.html（张三的广告页面）

|  |
| --- |
| <h1>广告。。。。</h1>  <a href=*"down"*>1.office.exe</a><br/>  <a href=*"down"*>2.qq.exe</a><br/>  <a href=*"down"*>3.迅雷.exe</a><br/>  </body> |

/zhangsan/download代码（张三下载请求时判断上一个资源）

|  |
| --- |
| **package** com.itheima.servlet;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"DownServlet"**, urlPatterns = **"/down"**) **public class** DownServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//处理下载请求  //1.获取上一个请求地址* String preUrl = request.getHeader(**"referer"**);  **if**(preUrl!=**null** && preUrl.equals(**"http://localhost:8080/zhangsan/adv.html"**)){  response.getOutputStream().write(**"下载成功"**.getBytes());  }**else**{  response.getOutputStream().write(**"小偷,别老偷我东西"**.getBytes());  }  } } |

/lisi/adv.html（李四网站的资源）

|  |
| --- |
| <body>  <h1>广告。。。。</h1>  <a href=*"http://localhost:8080/zhangsan/down"*>1.office.exe</a><br/>  <a href=*"http://localhost:8080/zhangsan/down"*>2.qq.exe</a><br/>  <a href=*"http://localhost:8080/zhangsan/down"*>3.迅雷.exe</a><br/>  </body> |

李四点击下载效果



### 请求头user-agent（了解）

#### 作用

利用user-agent获取客户端使用的是什么浏览器版本与浏览器内核

#### 案例:统计客户端使用的浏览器类型

|  |
| --- |
| **package** com.itheima.requestdata.head;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"UserAgentServlet"**, urlPatterns = **"/UserAgentServlet"**) **public class** UserAgentServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//利用user-agent获取客户端使用的是什么浏览器版本与浏览器内核* String userAgent = request.getHeader(**"user-agent"**);  System.***out***.println(userAgent);  String ip = request.getRemoteAddr();  *//应用:给网站统计用户都什么什么浏览器访问本网站  //使用user-agent可以统计客户端都使用什么浏览器* **if**(userAgent.contains(**"Chrome"**)){  System.***out***.println(ip+**":使用谷歌浏览器访问本网站"**);  }**else if**(userAgent.contains(**"Trident"**)){  System.***out***.println(ip+**":使用IE浏览器访问本网站"**);  }**else if**(userAgent.contains(**"Firefox"**)){  System.***out***.println(ip+**":使用火狐浏览器访问本网站"**);  }**else**{  System.***out***.println(ip+**":使用其他浏览器访问本网站"**);  }   } } |

## 获取请求体参数数据

​ 学习完了对请求行和请求头的内容，最后一部分就是请求体了，在请求体中，包含的是用户通过浏览器发送的请求参数，因此，我们主要学习的就是获取请求参数的方法。

### 获取请求参数使用方法

#### API介绍

1. String getParameter(String name)  根据表单的name属性 获取对应的值
2. String[] getParameterValues(String name)  获取name相同的所有value 例如复选框。
3. Map getParameterMap()  参数名作为key，参数值作为value，封装到map中。

#### 使用步骤

1. 准备html页面：getParam.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**h1**>提交get请求</**h1**> <**form action="BodyServlet" method="get"**>  用户名：<**input type="text" name="username"**><**br**>  爱 好：<**input type="checkbox" name="hobby" value="football"**>足球  <**input type="checkbox" name="hobby" value="basketball"**>篮球<**br**>  <**input type="submit" value="get提交"**> </**form**> <**h1**>提交post请求</**h1**> <**form action="BodyServlet" method="post"**>  用户名：<**input type="text" name="username"**><**br**>  爱 好：<**input type="checkbox" name="hobby" value="football"**>足球  <**input type="checkbox" name="hobby" value="basketball"**>篮球<**br**>  <**input type="submit" value="post提交"**> </**form**> </**body**> </**html**> |

1. 创建BodyServlet
2. 再BodyServlet中的doGet和doPost方法的参数列表，已经包含了request对象，调用相应方法即可。

#### 演示代码

|  |
| --- |
| **package** com.itheima.requestdata;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException; **import** java.util.Arrays;  @WebServlet(name = **"BodyServlet"**, urlPatterns = **"/BodyServlet"**) **public class** BodyServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//使用request获取请求体数据(就是获取以post提交数据)  //post提交的数据就是请求参数数据,有2种方式(get或post),获取请求参数核心方法  /\*  \* 1. String getParameter(String name) 根据表单的name属性 获取对应的值  2. String[] getParameterValues(String name) 获取name相同的所有value 例如复选框。  3. Map<String,String[]> getParameterMap() 参数名作为key，参数值作为value，封装到map中。   \* \*/  //获取get或post提交请求数据是一样的.*   *//获取用户名* String username = request.getParameter(**"username"**);  *//获取爱好(&hobby=football&hobby=basketball)* String[] hobbies = request.getParameterValues(**"hobby"**);  *//打印* System.***out***.println(**"username:"**+username);  System.***out***.println(**"hobbies:"**+ Arrays.*toString*(hobbies));   } } |

### 乱码解决

#### post请求参数乱码的原理

浏览器与服务器不会直接传递中文,首先会对中文进行编码

编码:从字符串🡺字节数组

解码:字节数组🡺字符串

只有tomcat8服务器get不会乱码,其他服务器get都会乱码,因为解码码表iso8859-1



#### API介绍

1. void setCharacterEncoding(String env)  
   设置request读取请求体的解码码表

#### 注意事项

1. 获取请求参数之后，调用setCharacterEncoding方法无效

#### 演示代码

准备提交数据页面

|  |
| --- |
|  |

处理请求Servlet

|  |
| --- |
| **package** com.itheima.requestdata;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException; **import** java.util.Arrays;  @WebServlet(name = **"BodyServlet"**, urlPatterns = **"/BodyServlet"**) **public class** BodyServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//使用request获取请求体数据(就是获取以post提交数据)  //post提交的数据就是请求参数数据,有2种方式(get或post),获取请求参数核心方法  /\*  \* 1. String getParameter(String name) 根据表单的name属性 获取对应的值  2. String[] getParameterValues(String name) 获取name相同的所有value 例如复选框。  3. Map<String,String[]> getParameterMap() 参数名作为key，参数值作为value，封装到map中。   \* \*/  //获取get或post提交请求数据是一样的.   //读取数据之前修改request读取post解码码表为utf-8* request.setCharacterEncoding(**"utf-8"**);   *//获取用户名* String username = request.getParameter(**"username"**);  *//获取爱好(&hobby=football&hobby=basketball)* String[] hobbies = request.getParameterValues(**"hobby"**);  *//打印* System.***out***.println(**"username:"**+username);  System.***out***.println(**"hobbies:"**+ Arrays.*toString*(hobbies));   } } |

### 案例：使用BeanUtils封装表单提交的数据到javaBean对象中

#### 案例需求

​ **现在我们已经可以使用request对象来获取请求参数，但是，如果参数过多，我们就需要将数据封装到对象。**​ **以前封装数据的时候，实体类有多少个字段，我们就需要手动编码调用多少次setXXX方法，因此，我们需要BeanUtils来解决这个问题。**

#### 案例效果

​ **使用BeanUtils，完成数据的封装到实体类。**

#### 案例分析

1. 设置一个登录页面准备提交表单数据（username、password）
2. 导入BeanUtils相关jar包
3. 创建Servlet获取请求参数
4. 调用BeanUtils.populate方法封装数据

#### 实现步骤

1. **准备登录页面：**

|  |
| --- |
| 1. <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**form action="LoginServlet" method="post"**>  <**table**>  <**tr**>  <**td**>用户名</**td**>  <**td**><**input type="text" name="username"**/></**td**>  </**tr**>  <**tr**>  <**td**>密码</**td**>  <**td**><**input type="password" name="password"**/></**td**>  </**tr**>  <**tr**>  <**td**></**td**>  <**td**><**input type="submit" value="登录"**/></**td**>  </**tr**>  </**table**> </**form**>   </**body**> </**html**> |

**在web目录下创建WEB-INF\lib文件夹，导入BeanUtils相关jar包**

commons-beanutils-1.8.3.jar  
commons-logging-1.1.1.jar

**注意:添加时选择添加到Module**

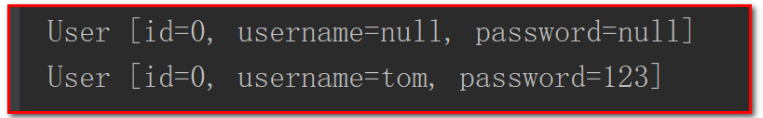
1. 导入实体类User

|  |
| --- |
| **package** com.itheima.model;  **public class** User {  **private int id**;  **private** String **username**;  **private** String **password**;   **public int** getId() {  **return id**;  }   **public void** setId(**int** id) {  **this**.**id** = id;  }   **public** String getUsername() {  **return username**;  }   **public void** setUsername(String username) {  **this**.**username** = username;  }   **public** String getPassword() {  **return password**;  }   **public void** setPassword(String password) {  **this**.**password** = password;  }   **public** User() {  }   **public** User(**int** id, String username, String password) {  **this**.**id** = id;  **this**.**username** = username;  **this**.**password** = password;  }   @Override  **public** String toString() {  **return "User{"** +  **"id="** + **id** +  **", username='"** + **username** + **'\''** +  **", password='"** + **password** + **'\''** +  **'}'**;  } } |

1. servlet代码：封装表单数据到User对象

package cn.itcast.web;  
​  
import cn.itcast.domain.User;  
import org.apache.commons.beanutils.BeanUtils;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.lang.reflect.InvocationTargetException;  
import java.util.Map;  
​  
@WebServlet(name = "LoginServlet",urlPatterns = "/login")  
public class LoginServlet extends HttpServlet {  
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    doGet(request,response);  
 }  
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    //获取请求参数  
    Map<String, String[]> map = request.getParameterMap();  
    //创建要封装数据的对象  
    User user = new User();  
    //封装前打印一次数据  
    System.out.println(user);  
    try {  
      //封装数据  
      BeanUtils.populate(user, map);  
   } catch (Exception e) {  
      e.printStackTrace();  
   }  
​  
    //封装后打印一次数据  
    System.out.println(user);  
 }  
}

效果：



## request的其他作用

### 获取工程名字

#### API介绍

1. String getContextPath()  获取工程名字

#### 演示代码

|  |
| --- |
| **package** com.itheima.getprojectname;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"GetProjectNameServlet"**, urlPatterns = **"/GetProjectNameServlet"**) **public class** GetProjectNameServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  *//使用request获取工程名字* System.***out***.println(request.getContextPath());  *//今天不用,以后在服务器端写全路径(绝对路径)使用,/工程名字/资源路径* } } |

### 请求转发页面跳转

#### 什么是请求转发页面跳转？

就是页面跳转(资源跳转),从一个资源跳转到另一个资源.

#### API介绍

1. RequestDispatcher getRequestDispatcher(String path) 获取请求转发器（request对象方法）
2. void forward(ServletRequest request, ServletResponse response) 将请求转发到另一个资源（servlet）上（RequestDispatcher对象的方法）

#### 使用步骤

1. 先通过请求对象获取转发器
2. 再调用转发器转发方法，转发请求​

#### 演示代码

OneServlet跳转到TwoServelt

OneServlet

|  |
| --- |
| **package** com.itheima.jump;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"OneServlet"**, urlPatterns = **"/OneServlet"**) **public class** OneServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  System.***out***.println(**"进入了OneServlet"**);     *//跳转到TwoServlet // 1. RequestDispatcher getRequestDispatcher(String path) 获取请求跳转转发器（request对象方法） // 2. void forward(ServletRequest request, ServletResponse response) // 将请求跳转转发到另一个资源（servlet）上（RequestDispatcher对象的方法）  //页面跳转(请求转发,明天讲这个概念,进行重点关注页面跳转)* request.getRequestDispatcher(**"TwoServlet"**).forward(request,response);   } } |

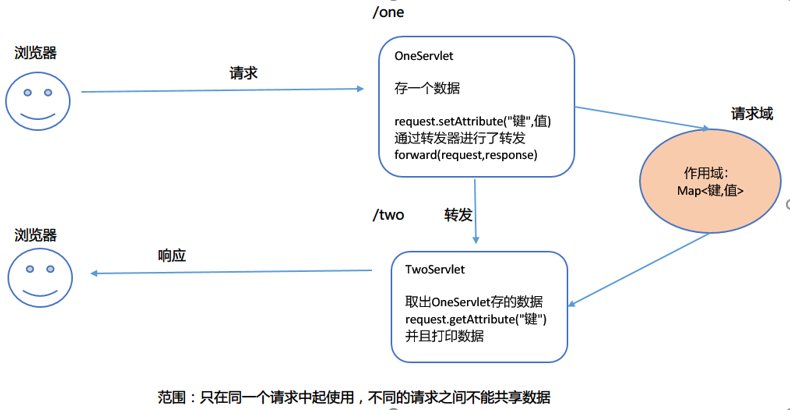
​TwoServlet

|  |
| --- |
| **package** com.itheima.jump;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"TwoServlet"**, urlPatterns = **"/TwoServlet"**) **public class** TwoServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  System.***out***.println(**"进入了TwoServlet"**);  request.getRequestDispatcher(**"login.html"**).forward(request,response);  } } |

### 域对象

问：什么是域对象？

答：域对象是一个存储数据的空间，主要用于servlet与servlet之间的数据传输使用的



request是一个请求域对象,就是每次请求的域对象是相互独立的,不同的请求不同共享.

#### API介绍(凡是域对象都有如下方法)

1. void setAttribute(String name, Object o)  设置数据到request域
2. Object getAttribute(String name)  从request域获取数据
3. void removeAttribute(String name)  从request域移除数据

#### 使用步骤

要求:OneServlet存储数据,跳转到TwoServlet读取数据并打印出来

1. 创建OneServlet和TwoServlet
2. 调用request对象存（setAttribute）取（getAttribute）删（removeAttribute）方法
3. 在保存和删除方法调用完成之后，都是的获取方法获取数据，打印在控制台上

OneServlet

|  |
| --- |
| **package** com.itheima.jump;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"OneServlet"**, urlPatterns = **"/OneServlet"**) **public class** OneServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  System.***out***.println(**"进入了OneServlet"**);   *//request是一个域对象,凡是域对象都有如下方法 // 1. void setAttribute(String name, Object o) 设置数据到request域 // 2. Object getAttribute(String name) 从request域获取数据 // 3. void removeAttribute(String name) 从request域移除数据    //使用request域对象存储数据* request.setAttribute(**"loginUser"**,**"admin"**);   *//跳转到TwoServlet // 1. RequestDispatcher getRequestDispatcher(String path) 获取请求跳转转发器（request对象方法） // 2. void forward(ServletRequest request, ServletResponse response) // 将请求跳转转发到另一个资源（servlet）上（RequestDispatcher对象的方法）  //页面跳转(请求转发,明天讲这个概念,进行重点关注页面跳转)* request.getRequestDispatcher(**"TwoServlet"**).forward(request,response);   } } |

TwoServlet

|  |
| --- |
| **package** com.itheima.jump;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"TwoServlet"**, urlPatterns = **"/TwoServlet"**) **public class** TwoServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  System.***out***.println(**"进入了TwoServlet"**);   *//从域里面获取loginUser* Object object = request.getAttribute(**"loginUser"**);  **if**(object!=**null**){  System.***out***.println(**"域里面的数据loginUser为:"**+object.toString());  }   request.getRequestDispatcher(**"login.html"**).forward(request,response);  } } |

#### 注意事项

​ **以上三个方法都是操作request中域对象的数据，与请求参数无关。**

# 案例： 实现登录功能

#### 案例需求

​ 实现用户登录功能。

#### 案例效果

​ 用户名和密码正确，显示登录成功

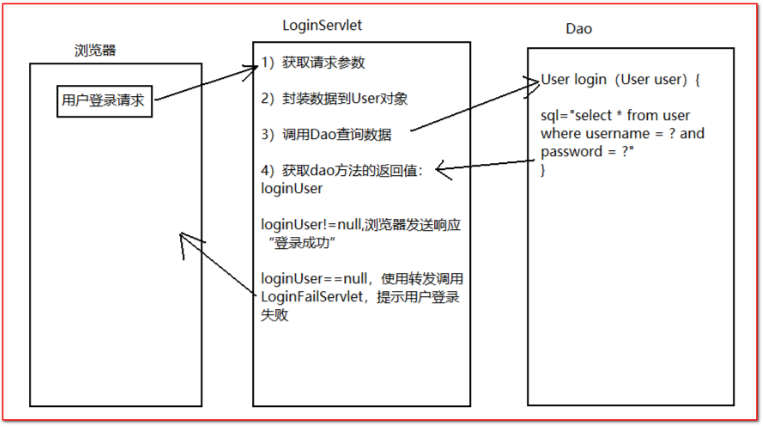
​ 用户名和密码错误，显示登录失败

#### 案例分析

1.设置一个登录页面准备提交表单数据（username、password）

2.导入相关jar包（BeanUtils、druid、mysql驱动、JDBCTempldate）

3.登录案例流程图：



#### 实现步骤

1. 准备登录页面：

|  |
| --- |
| 1. <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**form action="LoginServlet" method="post"**>  <**table**>  <**tr**>  <**td**>用户名</**td**>  <**td**><**input type="text" name="username"**/></**td**>  </**tr**>  <**tr**>  <**td**>密码</**td**>  <**td**><**input type="password" name="password"**/></**td**>  </**tr**>  <**tr**>  <**td**></**td**>  <**td**><**input type="submit" value="登录"**/></**td**>  </**tr**>  </**table**> </**form**>   </**body**> </**html**> |

1. 在web目录下创建WEB-INF\lib文件夹，导入相关jar包（今天资料文件夹中有）

     commons-beanutils-1.8.3.jar  
      commons-logging-1.1.1.jar  
      druid-1.0.9.jar  
      mysql-connector-java-5.1.18-bin.jar  
      spring-beans-4.2.4.RELEASE.jar  
      spring-core-4.2.4.RELEASE.jar  
      spring-jdbc-4.2.4.RELEASE.jar  
      spring-tx-4.2.4.RELEASE.jar

（注意：导入完成还要关联jar包到项目）

1. 导入实体类User（今天资料文件夹中有）

|  |
| --- |
| 1. **package** com.itheima.model;  **public class** User {  **private int id**;  **private** String **username**;  **private** String **password**;   **public int** getId() {  **return id**;  }   **public void** setId(**int** id) {  **this**.**id** = id;  }   **public** String getUsername() {  **return username**;  }   **public void** setUsername(String username) {  **this**.**username** = username;  }   **public** String getPassword() {  **return password**;  }   **public void** setPassword(String password) {  **this**.**password** = password;  }   **public** User() {  }   **public** User(**int** id, String username, String password) {  **this**.**id** = id;  **this**.**username** = username;  **this**.**password** = password;  }   @Override  **public** String toString() {  **return "User{"** +  **"id="** + **id** +  **", username='"** + **username** + **'\''** +  **", password='"** + **password** + **'\''** +  **'}'**;  } } |

1. 导入配置文件（druid-config.properties）和工具类（JDBCUtils）（今天资料文件夹中有）：
2. 导入JDBCUtils:
3. servlet代码：

LoginServlet

|  |
| --- |
| **package** com.itheima.servlet;  **import** com.itheima.dao.UserDao; **import** com.itheima.model.User; **import** org.apache.commons.beanutils.BeanUtils;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException; **import** java.lang.reflect.InvocationTargetException; **import** java.util.Map;  @WebServlet(name = **"LoginServlet"**, urlPatterns = **"/LoginServlet"**) **public class** LoginServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   *//实例UserDao* **private** UserDao **userDao** = **new** UserDao();   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {   **try** {  *//解决post中文乱码* request.setCharacterEncoding(**"utf-8"**);   *//获取登录请求数据封装到User里面  //原始方法:一个一个获取,再封装到User对象  // String username = request.getParameter("username");  // String password = request.getParameter("password");  // User user = new User(0,username,password);  //最新方法:使用BeanUtils工具类一次性将所有数据封装到User对象里面  //BeanUtil使用步骤  // 1.导入BeanUtils的jar包  // 2.使用BeanUtils工具类,BeanUtils.populate(java对象,map);  // java对象,就是要封装数据的对象  // map,是封装了所有java对象属性数据的集合* Map<String,String[]> map = request.getParameterMap();  User user = **new** User();  System.***out***.println(**"封装前user数据:"**+user.toString());   BeanUtils.*populate*(user,map);   System.***out***.println(**"封装后user数据:"**+user.toString());   *//访问dao层数据库查找对应用户对象* User loginUser = **userDao**.login(user);  *//判断用户是否为null* **if**(loginUser!=**null**) {  *//存在,输出登录成功* response.getOutputStream().write(**"登录成功"**.getBytes());  }**else** {  *//不存在,将用户名存储到请求域,跳转到LoginFailServlet,获取请求域数据,输出XX用户登录失败  //将用户名存储到域对象里面* request.setAttribute(**"username"**,user.getUsername());  request.getRequestDispatcher(**"LoginFailServlet"**).forward(request,response);  }   } **catch** (Exception e) {  e.printStackTrace();  }  } } |

LoginFailServlet

|  |
| --- |
| **package** com.itheima.servlet;  **import** javax.servlet.ServletException; **import** javax.servlet.annotation.WebServlet; **import** javax.servlet.http.HttpServlet; **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** java.io.IOException;  @WebServlet(name = **"LoginFailServlet"**, urlPatterns = **"/LoginFailServlet"**) **public class** LoginFailServlet **extends** HttpServlet {  **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {   *//获取域里面的username* String username = (String) request.getAttribute(**"username"**);  *//输出xx用户登录失败* response.getOutputStream().write((username+**"用户,登录失败"**).getBytes());  } } |