Day11

java课程-李彦伯

- Day11
  - HttpServletRequest
    - 获取请求行内容
      - 获取请求方式
      - 获取uri和url路径
      - 获取web应用的名称
      - 获取请求的ip地址(了解)
      - 获取get请求表单数据(了解)
    - 获取请求头的内容
      - 获取指定头的数据
      - 获取所有头信息(了解)
    - 获取请求体的内容
      - 获取提交表单数据中某个具体的name的value
      - 获取提交表单数据中一个name多个value数据
      - 获取提交表单所有数据
      - 请求乱码问题
    - 内部转发
    - 内部转发和重定向的区别
    - 域对象
    - 数据的写入,读取,删除
      - request域生命周期
      - request和servletContext区别
  - 作业
  - Cookie&Session

HttpServletRequest

> 客户端发送过来的请求通过tomcat内核进行提取,然后将数据封装到HttpServletRequest对象中,在调用servlet的service方法的时候将此对象作为参数传递过来,HttpServlet在service内部调用doGet和doPost方法将此对象作为参数传递过来.

获取请求行内容

获取请求方式

```
    //获取请求方式
    String method =  request.getMethod();
    System.out.println(method);//GET
```

获取uri和url路径

- uri:统一资源标识符,通过request获取uri获取的是请求路径中web应用后的字符串不包含QueryString
- url:同一资源定位符,通过request获取的url是带有协议名称应用名称端口号的完全路径

```
    //获取uri,就是请求行中的地址
    String uri =  request.getRequestURI();
    //获取url,获取全路径
    StringBuffer url = request.getRequestU
RL();
    System.out.println(uri);///WebTest/tt
  System.out.println(url);//http://192.16
8.6.154:8080/WebTest/tt
```

获取web应用的名称

> 我们以后再写相对服务器的绝对路径就使用这种方式

```
    //获取当前web应用名称
    String webName = request.getContextPat
h();
    System.out.println(webName);// /WebTes
t
```

获取请求的ip地址(了解)

```
    //获取请求的客户端的ip地址
    String ipAddress =  request.getRemoteA
ddr();
    System.out.println(ipAddress);/192.16
8.6.171
```

获取get请求表单数据(了解)

```
    //获取get请求网址?后的内容,如果是post获取内
容为null
  String queryString = request.getQuerySt
ring();
  System.out.println(queryString);//nam
e=123
```

获取请求头的内容

获取指定头的数据

```
 //获取指定header的内容
 String header =  request.getHeader("User-Agent");
 System.out.println(header);
 //Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
```

获取所有头信息(了解)

```
//获取所有header的名字
Enumeration<String> names = request.getHeaderNames();
while(names.hasMoreElements()){
 String str = names.nextElement();
 String value = request.getHeader(str);
 System.out.println(str+"="+value);
}
```

> 练习:如何使用referer头信息设置防盗链

获取请求体的内容

> 获取请求体中的提交的表单数据,尽管get请求数据在url后,用此方法同样可以获取

获取提交表单数据中某个具体的name的value

```
//对于get请求虽然表单数据在请求行中,也是通过此方法获取提交数据的
String name = request.getParameter("username");
```

获取提交表单数据中一个name多个value数据

```
//对于表单中一个name有多个值的可以使用此方法,如checkbox,返回值是数组
String [] hobbys = request.getParameterValues("hobby");
```

获取提交表单所有数据

> 数据格式是map格式,key指的是表单中的name属性,value是字符串数组,数组中存放的是对应的value属性

```
//获取所有的表单数据返回值是map形式,key是字符串
表单的name属性,值是String数组,存放的是
Map<String, String[]> map = request.getPa
rameterMap();
for(String key : map.keySet()){
    String[] values = map.get(key);
    System.out.println(key+"="+Arrays.toS
tring(values));
}
```

请求乱码问题

> 对于表单提交,由于页面的编码格式utf-8,但在服务器端,将数据提取封装到request对象的时候采用的是iso8859-1,所以出现乱码

- 对于post提交

```
request.setCharacterEncoding("UTF-8");
```

- 对于get提交,如果出现乱码

```
String parameter = request.getParamete
r("username");
parameter = new String(parameter.getbyte
s("iso8859-1"),"utf-8");
```

内部转发

> 客户端向服务器发送请求,服务器收到这个请求以后,可以在内部进行一个转发功能,也就是在不告知客户端的情况下进行请求的转发

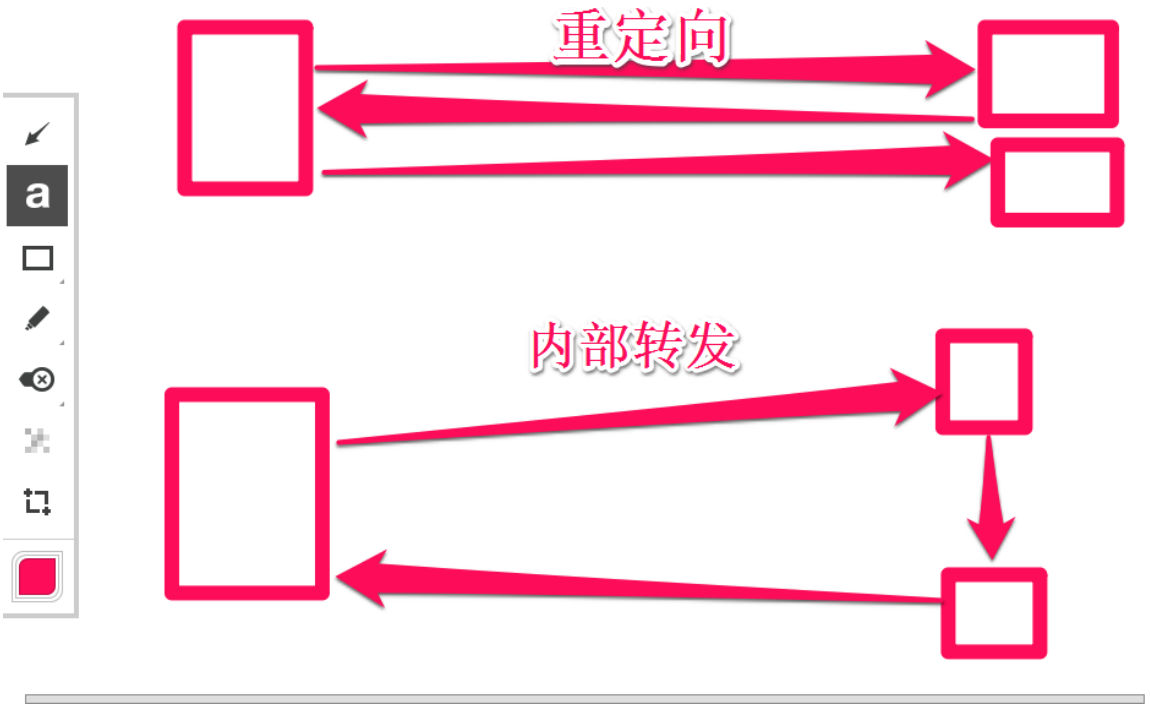
```
//获取内部转发器
RequestDispatcher dispatcher = request.ge
tRequestDispatcher("/index.jsp");
//进行转发
dispatcher.forward(request, response);
//通常是连写
//request.getRequestDispatcher("/index.js
p").forward(request, response);
```

- 内部转发客户端浏览器的地址不会发生改变,因为自始至终客户端只发送一个请求
- 通过内部转发路径一定要注意,因为操作过程是在服务器内部进行操作的,所以地址就不能够是客户端地址,而应该是服务器端地址,路径不能再写web应用的名称,/就

web应用的根目录

- 通过内部转发,两个servlet或者两个界面的接收到的request内的内容是相同的
- 因为内部转发是在服务器内部做的处理,所以内部转发是可以直接访问到WEB-INF下的内容的

内部转发和重定向的区别



域对象

> request对象也是一个域对象,所以也是可以存放数据的

数据的写入,读取,删除

- 向request域中放入数据`setAtrribute(String name,Object obj);`
- 从request域中获取数据`getAttribute(String name);`
- 从request域中删除数据`removeAttribute(String name);`

request域生命周期

- 创建:一次请求访问的时候创建request
- 销毁:一次请求结束后
- 作用范围:只在一次请求中

request和servletContext区别

- 相同点
  - 都是域对象,都能存放,读取,删除数据

- 不同点
  - servletContext随着服务器的开启而开启,服务器的关闭而关闭,作用范围在整个服务器服务期间
  - request每一次请求tomcat都会重新创建一个新的request,所以request的作用范围只在一个请求中,如果使用内部转发机制建议使用request做为存储

作业

> 注册登录

Cookie&Session

会话

> 思考:在用户登录后,如何保证这个用户进入我们web应用的每个界面都能正确的显示用户的名字.在购物网站上将购物的数据添加到购物车功能如何实现?
>
> HTTP协议是无状态的协议,也就是说每次客户端访问服务器,对于服务器而言,其实并不知道是谁在访问,如果要保证每次客户端访问时候,让我们的服务器能够区分出来是谁在访问,就需要运用会话技术

- 什么是会话

  > **从用户打开客户端访问我们的web应用,到用户关闭客户端这个过程称作为一次会话.**因为客户端访问我们的web应用,就会在操作系统的内存中开辟一块内存空间存储访问过程中的数据,直到用户关闭浏览器,内存中的数据才会清空.

- 会话技术的分类

- Cookie:数据存储在客户端本地，减少服务器端的存储的压力，安全性不好，客户端 可以清除cookie
- Session:将数据存储到服务器端，安全性相对好，增加服务器的压力

Cookie

> 将数据存入客户端,特点是安全性不好,但是减少了服务器端的压力,**注意cookie不能存中文**

cookie存储过程

> 服务器端将需要存储的数据放在cookie中,通过响应头的形式发送给客户端,客户端收到了这个头信息,就会将数据进行保存,客户端再次请求服务的时候就会将此信息放在请求头中,那么服务器端通过获取请求头就能得到不同用户的cookie,从而区分出了不同的用户,做出不同的处理

- 创建cookie

```
Cookie cookie = new Cookie("name", "lis
i");
```

- 发送cookie

```
response.addCookie(cookie);
```

- 设置的cookie默认是会话级别的,也就是说浏览器关闭后,cookie就没有了(因为会话级别的cookie存储在本机的内存当中),我们可以设置cookie的存取时间(设置后就会将cookie存储到本地磁盘当中)

```
//单位是秒
cookie.setMaxAge(10*60);
```

- 设置cookie默认会在请求和servlet同路径的情况下才会携带cookie中存储的数据，包含同级目录和下级目录,我们也可以设置cookie携带路径

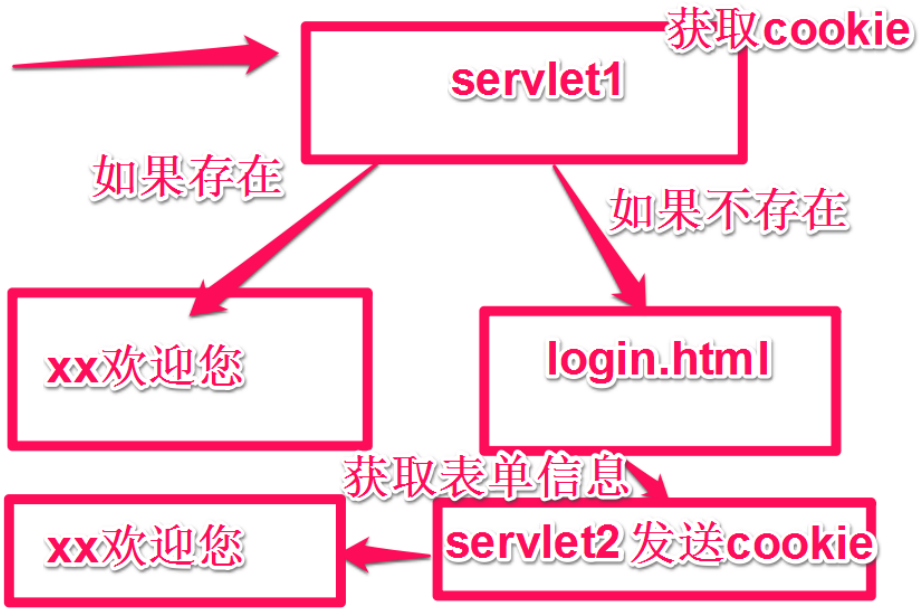
```
//代表访问WebTest应用中的任何资源都携带cookie
cookie.setPath("/WebTest");
//代表访问WebTest中的cookieServlet时才携带cookie信息
cookie.setPath("/WebTest/cookieServlet");
```

- 如果设置了时间那么客户端会自动删除cookie,如果未满时间,想要删除cookie,那么要使用**同名同路径的持久化时 间为0的cookie进行覆盖即可**

```
Cookie cookie = new Cookie("name", "");
cookie.setPath("/WebTest");
cookie.setMaxAge(0);
response.addCookie(cookie);
```

- 客户端接收cookie

```
Cookie[] cookies = request.getCookies();
for(Cookie cookie : cookies){
if(cookie.getName().equal("name")){
    String cookieValue = cookie.getValu
e();
    }
}
```



Session

> 将用户的数据存储到服务器,会为每一块客户端创建一块内存空间,在用户第一请求的时候通过响应头的形式将这块内存空间的编号发送给客户端,客户端每次请求将JSESSIONID放在请求头中访问服务器,服务器会找到客户端对应的那块内存空间

获得Session对象

> **此方法会做两件事**
> 1.得专属于当前会话的Session对象，如果服务器端没有该会话的Session对象会创建一个新的Session对象，如果已经有了属于该会话的Session直接获取已有的Session对象
> 2.并将session的JSESSIONID放入cookie,并设置cookie的携带路径为当前的web应用,自动通过response返回.下次再通过浏览器访问该web应用就会携带这个JSESSIONID,找到客户端对应的那块内存中的session

```
HttpSession session = request.getSessio
n();
//获取JSESSIONID
String sessionID = session.getId();
System.out.println(sessionID);
```

域对象

> session和request,servletContext相同都是一个域对象,可以存放读取删除数据

写入,读取,删除数据

- 向request域中放入数据`setAtrribute(String name,Object obj);`
- 从request域中获取数据`getAttribute(String name);`
- 从request域中删除数据`removeAttribute(String name);`

Session的生命周期

- 创建:第一次调用`request.getSession();`
- 销毁有3中方式:
  - 1.服务器关闭
  - 2.session过期,默认30分钟(是从不操作服务器端的

资源开始计算),可以在web.xml中进行修改,我们可以在web应用的web.xml进行同样的配置

```
<session-config>
<session-timeout>30</session-timeout>
</session-config>
```

- 3.调用`session.invalidate();`

> 注意,存放JSESSIONID的cookie默认是会话级别,所以关闭了浏览器,再次访问JSESSIONID就不存在了,所以会重新创建新的session,但是存放JSESSIONID的cookie是系统帮助我们创建,所以怎么设置这个cookie保存到磁盘呢?我们可以按照系统的格式创建一个同名称的cookie就行了,添加一个存取时间

```
HttpSession session = request.getSessio
n();
String sessionID = session.getId();
Cookie cookie = new Cookie("JSESSIONID",
sessionID);
cookie.setPath(request.getContextPath());
cookie.setMaxAge(10*60);
response.addCookie(cookie);
```