

# day30-cookie&session&jsp入门

## 今日任务

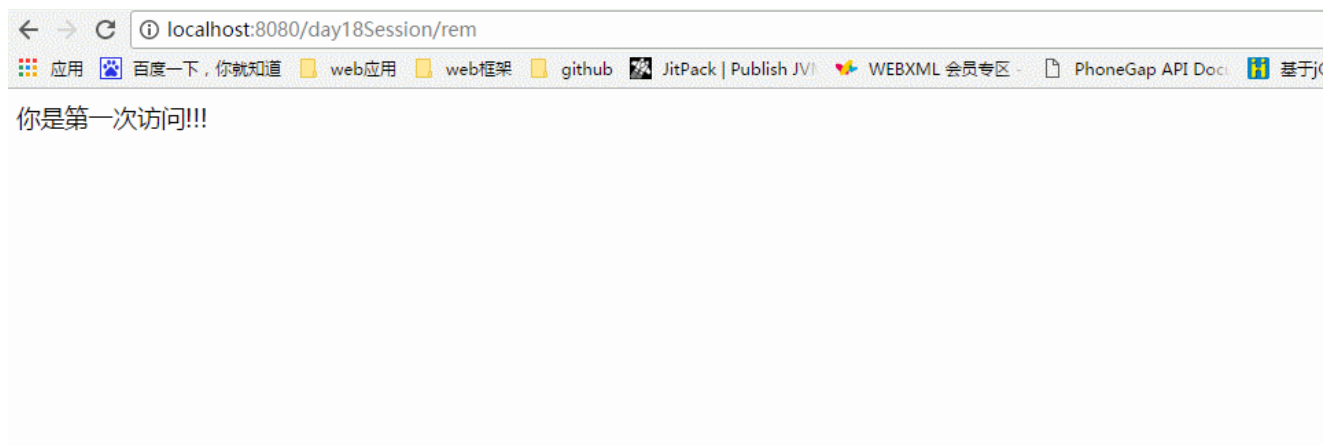
- 案例一：记录上次访问时间
- 案例二：一次性验证码校验

## 教学目标

1. 能够说出会话的概念
2. 能够创建、发送、接收、删除cookie
3. 能够获取session对象、添加、删除、获取session中的数据
4. 能够完成登录验证码案例
5. 能够说出jsp的优势
6. 能够编写jsp代码片段、声明、脚本表达式

## 案例一：记录上次访问时间

### 一，需求分析



在访问一个资源的时候,展示上次访问的时间

若是第一次访问则展示:你是第一次访问,若不是第一次则展示:你上次访问的时间是:xxxx

### 二，技术分析

#### 1.会话的概念

用户打开浏览器，浏览不同的网页，发出多个请求，直到关闭浏览器的过程，称为一次会话。

如同打电话。

我们在会话的过程(多次请求)之中,用户可能会产生一些数据,这些数据有的需要保存起来的,我们就可以通过会话技术来保存用户各自的数据

#### 2.为什么要使用会话技术

保存用户各自的数据。

\*私有的数据,购物信息数据保存在会话技术中.

### 3.常用的会话技术

#### 3.1cookie

cookie是客户端（浏览器）端的技术，用户浏览的信息以键值对(key=value)的形式保存在浏览器上。如果没有关闭浏览器，再次访问服务器，会把cookie带到服务端，服务端就可以做响应的处理。

#### 3.2session

session是服务器端的技术。服务器为每一个浏览器开辟一块内存空间，即session。由于内存空间是每一个浏览器独享的，所有用户在访问的时候，可以把信息保存在session对象中。同时，每一个session对象都对应一个sessionId，服务器把sessionId写到cookie中，再次访问的时候，浏览器会把cookie（sessionId）带过来，找到对应的session对象。

### 4.cookie的使用

#### 4.1API概述

创建一个 cookie，cookie 是 servlet 发送到 Web 浏览器的少量信息，这些信息由浏览器保存，然后发送回服务器。cookie 的值可以唯一地标识客户端，因此 cookie 常用于会话管理。

一个 cookie 拥有一个名称、一个值和一些可选属性，比如注释、路径和域限定符、最大生存时间和版本号。一些 Web 浏览器在处理可选属性方面存在 bug，因此有节制地使用这些属性可提高 servlet 的互操作性。

servlet 通过使用 `HttpServletResponse#addCookie` 方法将 cookie 发送到浏览器，该方法将字段添加到 HTTP 响应头，以便一次一个地将 cookie 发送到浏览器。浏览器应该支持每台 Web 服务器有 20 个 cookie，总共有 300 个 cookie，并且可能将每个 cookie 的大小限定为 4 KB。

浏览器通过向 HTTP 请求头添加字段将 cookie 返回给 servlet。可使用 `HttpServletRequest#getCookies` 方法从请求中获取 cookie。一些 cookie 可能有相同的名称，但却有不同的路径属性。

cookie 影响使用它们的 Web 页面的缓存。HTTP 1.0 不会缓存那些使用通过此类创建的 cookie 的页面。此类不支持 HTTP 1.1 中定义的缓存控件。

#### 4.2 创建一个Cookie对象

```
new Cookie(String name,String value); //cookie只能保存字符串数据。且不能保存中文
```

#### 4.3 把cookie写回浏览器:

HttpServletResponse的一个方法

```
void addCookie(Cookie cookie);
```

#### 4.4 获得浏览器带过来的所有Cookie:

```
HttpServletRequest  
Cookie[] getCookies() ;得到所有的cookie对象。是一个数组，开发中根据key得到目标cookie
```

#### 4.5cookie的 API

```
cookie.getName() ; 返回cookie中设置的key  
cookie.getValue(); 返回cookie中设置的value
```

### 三，思路分析

【步骤一】：创建RememberServlet.

【步骤二】：RememberServlet中，获得上次访问时间.判断是否是第一次访问,进行展示

## 四，代码实现

```
@WebServlet("/rem")
public class RememberServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        //0.处理响应乱码
        response.setContentType("text/html;charset=utf-8");

        //1.获得所有的cookie对象
        Cookie[] cookies = request.getCookies();
        //2. 获得目标cookie,进行判断
        Cookie targetCookie = CookieUtils.getTargetCookie("lastTime", cookies);
        if(targetCookie == null){
            //3. 第一次访问
            //a. 展示您是第一次访问
            response.getWriter().print("您是第一次访问...");
            //b. 把当前的访问时间保存到cookie,写给浏览器
            Cookie cookie = new Cookie("lastTime", System.currentTimeMillis() + "");
            response.addCookie(cookie);
        }else{
            //4. 不是第一次访问
            //a.从目标cookie里面获得值(上次访问时间)
            String timeStr = targetCookie.getValue();//19998989898998
            Date date = new Date(Long.parseLong(timeStr));
            response.getWriter().print("您上次访问的时间是"+date.toLocaleString());

            //b. 把当前的访问时间保存到cookie,写给浏览器
            Cookie cookie = new Cookie("lastTime", System.currentTimeMillis() + "");
            response.addCookie(cookie);
        }
    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        doGet(request,response);
    }
}
```

## 五，总结

### 1,对cookie的基本操作

- Cookie[] getCookies();得到所有的cookie。是一个数组，开发中根据key得到目标cookie
- Cookie(String name, String value); 创建一个cookie

- `response.addCookie(Cookie cookie)` ; 把cookie写回浏览器

## 2.cookie的分类

### 2.1会话级别cookie

在默认的情况下, 当浏览器进程结束(浏览器关闭,会话结束)的时候, cookie就会消失。

### 2.2持久性cookie

给cookie设置有效期,`setMaxAge(int expiry)` :时间是秒

-1: 默认。代表Cookie数据存到浏览器关闭 (保存在浏览器内存中)。

正整数: 以秒为单位保存数据有效时间 (把缓存数据保存到磁盘中)

0: 代表删除Cookie.如果要删除Cookie要确保路径一致。

## 3.cookie设置有效路径

`setPath(String url)` ;设置路径

浏览器可以存储多个cookie, 每一个cookie都有各自的路径(没有设置, 有一个默认的路径)。一个路径不存在重名的cookie, 如果路径和名字一样,后面的会把之前的给覆盖掉,不同路径下可以有重名的cookie

- 默认路径

例如:

访问[http://localhost:8080/web18A\\_Cookie/demo01](http://localhost:8080/web18A_Cookie/demo01); cookie默认路径 /web18A\_Cookie

访问[http://localhost:8080/web18A\\_Cookie/aaa/demo01](http://localhost:8080/web18A_Cookie/aaa/demo01); cookie默认路径 /web18A\_Cookie/aaa

访问[http://localhost:8080/web18A\\_Cookie/aaa/bbb/demo01](http://localhost:8080/web18A_Cookie/aaa/bbb/demo01); cookie默认路径 /web18A\_Cookie/aaa/bbb

- 随带Cookie需要的条件

只有当访问的url包含此cookie的path的时候,才会携带这个cookie;反之不会。

例如:

设置cookie的路径 /web18A\_Cookie/aaa,

下次访问路径:[http://localhost:8080/web18A\\_Cookie/aaa/demo01](http://localhost:8080/web18A_Cookie/aaa/demo01); cookie是可以带过来

下次访问路径:[http://localhost:8080/web18A\\_Cookie/bbb/demo01](http://localhost:8080/web18A_Cookie/bbb/demo01); cookie带不过来

- cookie的路径通常设置 / 或者 /发布项目名

## 4.cookie的弊端

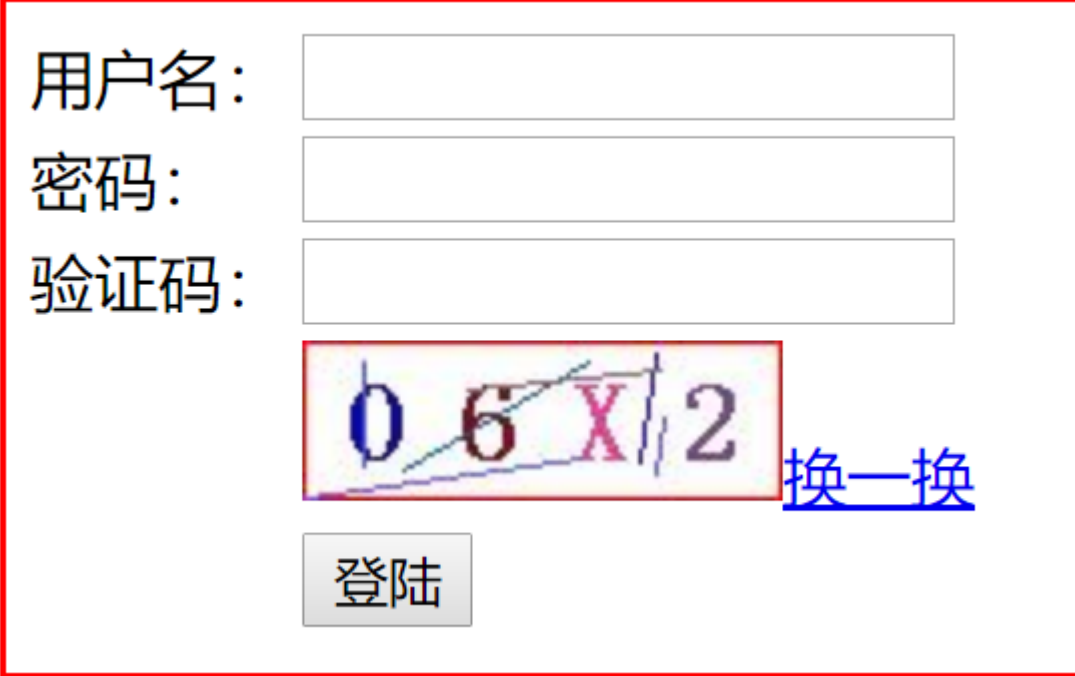
cookie的大小(个数和自身大小)和格式(只能存字符串)有限制

不支持中文,解决中文办法

```
//存入的时候(先通过utf-8编码)
URLEncoder.encode(value,"utf-8");
//取出 (通过utf-8解码)
URLDecode.decode (value, "utf-8")
```

## 案例二: 一次性验证码校验


### 一，需求分析



用户名:

密码:

验证码:

 [换一换](#)

在网站登录的时候,生成一个验证码.登录的时候对验证码进行校验.

### 二，技术分析

#### 1.session概述

session是服务器端的技术。服务器为每一个浏览器开辟一块内存空间，即session对象。由于session对象是每一个浏览器特有的，所有用户的记录可以存放在session对象中。同时，每一个session对象都对应一个sessionId，服务器把sessionId写到cookie中，再次访问的时候，浏览器把sessionId带过来，找到对应的session对象

#### 2.cookie和Session的不同

- cookie是保存在浏览器端的，大小和个数都有限制。session是保存在服务器端的，安全一些。
- cookie不支持中文，并且只能存储字符串；session可以存储基本数据类型，集合,对象等

#### 3.Session的基本用法（作为域对象存数据）

范围: 会话(多次请求)

- request.getSession(); 获得session
- Object getAttribute(String name); 获取值
- void setAttribute(String name, Object value); 存储值

- void removeAttribute(String name) ;移除

#### 4.Session的执行原理：基于Cookie的

#### 5.getSession()的执行原理(了解)

- 1、获得cookie中传递过来的SessionId
- 2、如果Cookie中没有sessionid,则创建session对象
- 3、如果Cookie中有sessionid,找指定的session对象

如果有sessionid并且session对象存在，则直接使用

如果有sessionid，但session对象销毁了，则执行第二步

### 三，思路分析

【步骤一】生成验证码的时候,将随机产生的4个字母或数字存入到session中。

【步骤二】在页面中输入一个验证码点击登录.提交到LoginServlet

【步骤三】在LoginServlet中获得页面提交的验证码和session中验证码比较

【步骤四】如果不一致,给用户一个错误的提示.

【步骤五】如果一致,再去比较用户名和密码

### 四，代码实现

- 登录页面

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body >
  <center>
    <h1>用户登录</h1>
    <form action="http://localhost:8080/userServlet" method="post">
      姓名: <input type="text" name="username" /><br/>
      密码: <input type="password" name="password"/><br/>
      验证码:<input type="text" name="code"/><br/>
      <br/>
      <input type="submit" value="登录"/>
    </form>
  </center>

</body>
<script>
  //没点击图片一下，图片就换一下，说白了就是把img的src值改变
  function changeImg(obj) {
    obj.src ="http://localhost:8080/codeServlet?a="+new Date().getMilliseconds();
  }

</script>
</html>
```

- CodeServlet

```

@WebServlet("/codeServlet")
public class CodeServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        //1. 创建验证码对象
        ValidateCode validateCode = new ValidateCode(100, 40, 4, 10);

        //获得验证码的值(ABC3)
        String code = validateCode.getCode();
        System.out.println("生成的code=" + code);
        //存到session里面
        request.getSession().setAttribute("code", code);

        //2. 通过response响应给页面(通过响应的字节码把验证码响应给前端)
        validateCode.write(response.getOutputStream());

    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        doGet(request, response);
    }
}

```

- UserServlet



```

@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        try {

            response.setContentType("text/html;charset=utf-8");

            //*****验证码的校验*****
            //a. 获得用户输入的验证码
            String userCode = request.getParameter("code");
            //b. 获得程序生成的(在session里面)
            String formCode = (String) request.getSession().getAttribute("code");
            //c. 比较这两个验证码是否一致
            if(!formCode.equalsIgnoreCase(userCode)){
                response.getWriter().print("验证码不一致,请重新输入...");
                return;
            }

            //*****
            //1. 获得请求参数(用户名和密码)
            String username = request.getParameter("username");
            String password = request.getParameter("password");

            //2. 调用业务, 进行登录逻辑
            UserService userService = new UserService();
            User user = userService.login(username,password);
            //3. 判断是否为null
            if(user != null){
                response.getWriter().print("Login Success");
            }else{
                response.getWriter().print("Login Failed");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().print("Login Failed");
        }

    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        doGet(request,response);
    }

}

```

|                    |                                     |  |              |  |
|--------------------|-------------------------------------|--|--------------|--|
| 五，总结               |                                     |  |              |  |
| 1.三个域对象比较          | 创建                                  | 销毁   | 作用范围         | 应用场景   |
| 29分钟操作了一下 30分钟     |                                     |  |              |  |
| 域对象                | 创建                                  | 销毁   | 作用范围         | 应用场景   |
| ServletContext     | 服务器启动                               | 服务器正常关闭/项目从服务器移除                           | 整个应用         | 记录访问次数,聊天室                                     |
| HttpSession        | 第一次调用<br>request.getSession()<br>方法 | session过期（默认30分钟）/调用invalidate（）方法/服务器异常关闭 | 会话<br>(多次请求) | 购物车,验证码校验,保存用户登录状态等                            |
| HttpServletRequest | 来了请求                                | 响应这个请求(或者请求已经接收了)                          | 一次请求         | servletA和jsp（servletB）之间<br>数据传递(转发的时候<br>存数据) |

C:\Users\yp\.IntelliJ IDEA2017.2\system\tomcat\tomcat43\_sz43\_4\work\Catalina\localhost 目录查看

如果是正常关闭服务器,

把session(内存)序列化到服务器磁盘上,再次启动,把磁盘上的文件反序列化到内存里面

序列化:对象变成字节序列(以文件形式存在的)的一个过程      内存-->磁盘

反序列化: 字节序列(以文件形式存在的)变成对象的一个过程      磁盘--->内存

三个域对象怎么选择?

一般情况下,最小的可以解决就用最小的.但是需要根据情况(eg: 重定向, 多次请求, 会话范围, 用session; 如果是转发,一般选择request)

## 2. session里面存的数据, 重定向有效吗?

request里面存的数据, 重定向是无效的;

session里面存的数据, 重定向有效的;

## 3.cookie和session的选择?

- 如果保存的数据量不大, 如果保存的数据不是很重要, 如果保存的数据是一般的字符串,通常用cookie
- 如果保存的数据量很大, 如果保存的是对象(登录信息), 如果保存的数据特别重要, 通常用session

## 第三章\_JSP入门

### 一,JSP概述

#### 1.什么是JSP

Java server page(java服务器页面). JSP本质就是Servlet

它和servle技术一样，都是SUN公司定义的一种用于开发动态web资源的技术。

JSP=html+java+jsp特有的内容

#### 2.JSP产生的原因

需求: 我们要向页面动态输出一个表格. 发现特别的繁琐

servlet在展示页面的时候，相当的繁琐。sun公司为了解决这个问题，参照asp开发了一套动态网页技术jsp。

#### 3.JSP执行原理

JSP会翻译(通过默认的JspServlet,JSP引擎)成Servlet(.java),Servlet编译成class文件

JSP执行流程

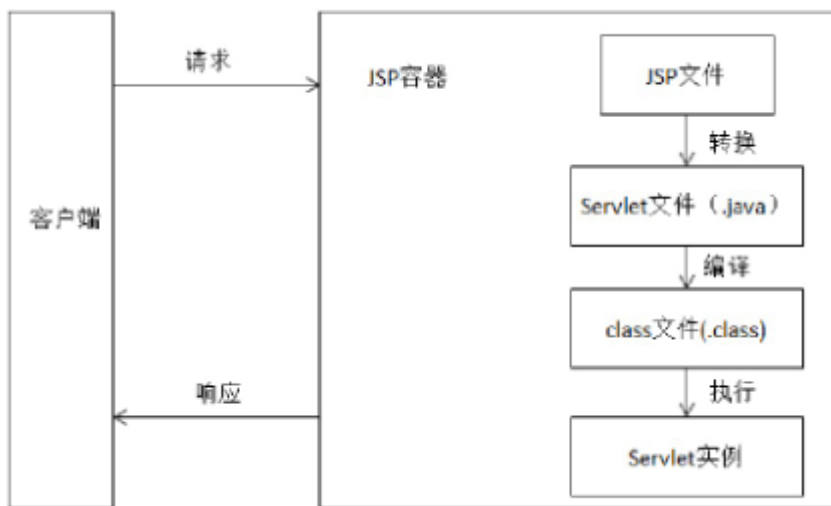
第一次访问的xxx.jsp时候,服务器收到请求,JspServlet会去查找对应的jsp文件

找到之后,服务器会将这个jsp文件转换成java文件(Servlet)

服务器编译java文件,生成class文件

服务器运行class文件,生成动态的内容

服务器收到内容之后,返回给浏览器



### 二,JSP基本语法

#### 1.JSP脚本

我们可以通过JSP脚本在JSP页面上编写Java代码. 一共有三种方式:

| 类型               | 翻译成Servlet和HTML源码中(浏览器查看)            | 翻译的Servlet代码中 |
|------------------|--------------------------------------|---------------|
| <%...%>:Java程序片段 | 翻译成Service()方法里面的内容                  |               |
| <%=...%>:输出表达式   | 翻译成Service()方法里面的内容,相当于调用out.print() | 输出表达式不能以;结尾   |
| <%!...%>:声明成员变量  | 翻译成Servlet类里面的内容                     |               |

- eg

```
<%
    for(int i = 0; i < 10;i++){
        out.print("i="+i);
        %>
        <hr/>
        <%
            }
        %>
```

2.JSP注释

| 注释类型                   | HTML源码中(浏览器查看) | 翻译的Servlet代码中 |
|------------------------|----------------|---------------|
| HTML注释                 | 存在             | 存在            |
| JAVA注释    //; /* */    | 不存在            | 存在            |
| JSP注释;    <%--注释内容--%> | 不存在            | 不存在           |

注释快捷键:Ctrl+Shift+/,