今日内容

1. 会话技术
    1. Cookie
    2. Session
2. JSP：入门学习

会话技术

1. 会话：一次会话中包含多次请求和响应。
    - 一次会话：浏览器第一次给服务器资源发送请求，会话建立，直到有一方断开为止
2. 功能：在一次会话的范围内的多次请求间，共享数据
3. 方式：
    1. 客户端会话技术：Cookie
    2. 服务器端会话技术：Session

Cookie：翻译是-->饼干，甜点

概念：客户端会话技术，将数据保存到客户端

快速入门：

- 使用步骤：
    1. 创建Cookie对象，绑定数据
        - new Cookie(String name, String value)
    2. 发送Cookie对象
        - response.addCookie(Cookie cookie)
    3. 获取Cookie，拿到数据
        - Cookie[] request.getCookies()、
    4. 获得单个Cookie之后获取名字和值：
        - cookie.getName()
        - cookie.getValue()

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
```

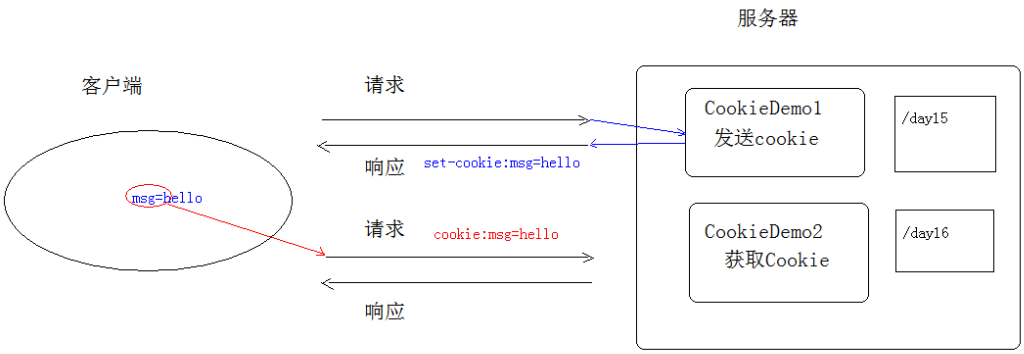
```java
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*创建cookie对象*/
        Cookie cookie = new Cookie("message", "hello");

        /*发送cookie*/
        response.addCookie(cookie);
    }

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

```java
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo2")
public class CookieDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*获取cookie数组*/
        Cookie[] cookies = request.getCookies();

        /*遍历cookies*/
        if (cookies!=null){
            for (Cookie cookie : cookies) {
                String name = cookie.getName();
                String value = cookie.getValue();
                System.out.println(name+"-->"+value);
            }
        }

    }
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doPost(request, response);
    }
}
/*
JSESSIONID-->A62C320CD77F8A73D9973A7A738B958F
message-->hello
Idea-eef7716b-->dd12e287-a58b-4a7b-8ed9-91ed4334fe90
*/
```

实现原理

- 基于响应头set-cookie和请求头cookie实现



cookie的细节

一次可不可以发送多个cookie?

- 可以
- 可以创建多个Cookie对象，使用response调用多次addCookie方法发送cookie即可。

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*
        * 发送多个Cookie
        * */
        Cookie cookie = new Cookie("message", "hello");
        Cookie cookie2 = new Cookie("name", "zhangsan");

        response.addCookie(cookie);
        response.addCookie(cookie2);
    }

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

cookie在浏览器中保存多长时间?

1. 默认情况下，当浏览器关闭后，Cookie数据被销毁
2. 持久化存储：
    - setMaxAge(int seconds)
        1. 正数：将Cookie数据写到硬盘的文件中。持久化存储。并指定cookie存活时间，时间到后，cookie文件自动失效，seconds是秒为计数单位。
        2. 负数：默认值
        3. 零：删除cookie信息

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        Cookie cookie = new Cookie("message", "hello");


        /*设置cookie的存活时间，存储到硬盘空间中
        * 设置为正数，设置30，那么30秒后会自动删除
```

```
        * 设置为负数，默认值，浏览器关闭后Cookie销毁
        * 设置为0，立刻销毁Cookie
        * */
        cookie.setMaxAge(30);

        response.addCookie(cookie);
    }

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

cookie能不能存中文?

- 在tomcat 8 之前 cookie中不能直接存储中文数据。
- 假如使用tomcat8之前的版本，那么需要将中文数据转码转为非中文的数据---一般采用URL编码(%E3)

```
String encode = URLEncoder.encode(format, "utf-8");
```

- 在tomcat 8 之后，cookie支持中文数据。特殊字符还是不支持，建议使用URL编码存储，URL解码解析

```
String decode = URLDecoder.decode(value, "utf-8");
```

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        Cookie cookie = new Cookie("message", "你好");

        response.addCookie(cookie);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
```

```
        this.doPost(request, response);
    }
}
```

cookie共享问题?

1. 假设在一个tomcat服务器中，部署了多个web项目，那么在这些web项目中cookie能不能共享?
    - 默认情况下cookie不能共享
    - setPath(String path):设置cookie的获取范围。默认情况下，设置当前的虚拟目录
        - 默认情况下是 cookie.setPath("/当前的虚拟目录")，这样保证的是只有在当前项目下才能访问
    - 如果要共享，则可以将path设置为"/"：cookie.setPath("/")

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        Cookie cookie = new Cookie("message", "你好");

        /*服务器上的所有项目都可以共享*/
        cookie.setPath("/");

        response.addCookie(cookie);
    }

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

不同的tomcat服务器间cookie共享问题?

这是需要域名来共享的，首先要了解一下什么是域名等级

- 百度知道：https://zhidao.baidu.com/

- 百度贴吧：https://tieba.baidu.com/index.html?traceid=
- 百度新闻：http://news.baidu.com/

那么在这两个都是属于百度的，从这里看，.baidu.com是一级域名，而zhidao，news和tieba是二级域名

- setDomain(String path):如果设置一级域名相同，那么多个服务器之间cookie可以共享

  > *setDomain(".baidu.com"),那么tieba.baidu.com和news.baidu.com中cookie可以共享*

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/cookieDemo1")
public class CookieDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        Cookie cookie = new Cookie("message", "你好");

        /*只要一级域名相同，那么不同服务器上的Cookie都可以共享*/
        cookie.setDomain(".baidu");

        response.addCookie(cookie);
    }

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

Cookie的特点和作用

1. cookie存储数据在客户端浏览器
2. 浏览器对于单个cookie 的大小有限制(4kb) 以及 对同一个域名下的总cookie数量也有限制(20个)

- 作用：
    1. cookie一般用于存出少量的不太敏感的数据

2. 在不登录的情况下，完成服务器对客户端的身份识别
    - 比如在浏览器的搜索引擎上可以设置是否联想，是否弹出广告，是否显示图片等等

案例：记住上一次访问时间

1. 需求：
    1. 访问一个Servlet，如果是第一次访问，则提示：您好，欢迎您首次访问。
    2. 如果不是第一次访问，则提示：欢迎回来，您上次访问时间为:显示时间字符串
2. 分析：
    1. 可以采用Cookie来完成
    2. 在服务器中的Servlet判断是否有一个名为lastTime的cookie
        1. 有：不是第一次访问
            1. 响应数据：欢迎回来，您上次访问时间为:2018年6月10日11:50:20
            2. 写回Cookie：lastTime=2018年6月10日11:50:01
        2. 没有：是第一次访问
            1. 响应数据：您好，欢迎您首次访问
            2. 写回Cookie：lastTime=2018年6月10日11:50:01

> *注意，非首次创建Cookie的时候即使设置了值也不要忘记发送response.addCookie(cookie);*

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.Date;

@WebServlet("/cookieDemo2")
public class CookieDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

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


        Cookie[] cookies = request.getCookies();

        boolean flag = false;
        Cookie cookie1 = null;
```

```
        for (Cookie cookie : cookies) {
            if ("time".equals(cookie.getName())){
                flag=true;
                cookie1=cookie;
            }

        }
        if (flag){

            String decode = decode(cookie1.getValue());

            cookie1.setValue(encode());

            cookie1.setMaxAge(60*60*24*30);

            response.addCookie(cookie1);
            response.getWriter().write("欢迎再次登录，您上次登陆的时间为："+decode);

        }else {
            Cookie time = new Cookie("time", encode());
            time.setMaxAge(60*60*24*30);
            response.addCookie(time);
            response.getWriter().write("欢迎您首次登陆");
        }

    }

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


    private String encode() throws UnsupportedEncodingException {
        Date date = new Date();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy年MM月dd日
HH:mm:ss");
        String format = simpleDateFormat.format(date);

        String encode = URLEncoder.encode(format, "utf-8");
        return encode;
    }

    private String decode(String encode) throws UnsupportedEncodingException {
        String decode = URLDecoder.decode(encode, "utf-8");
        return decode;
    }
}
```

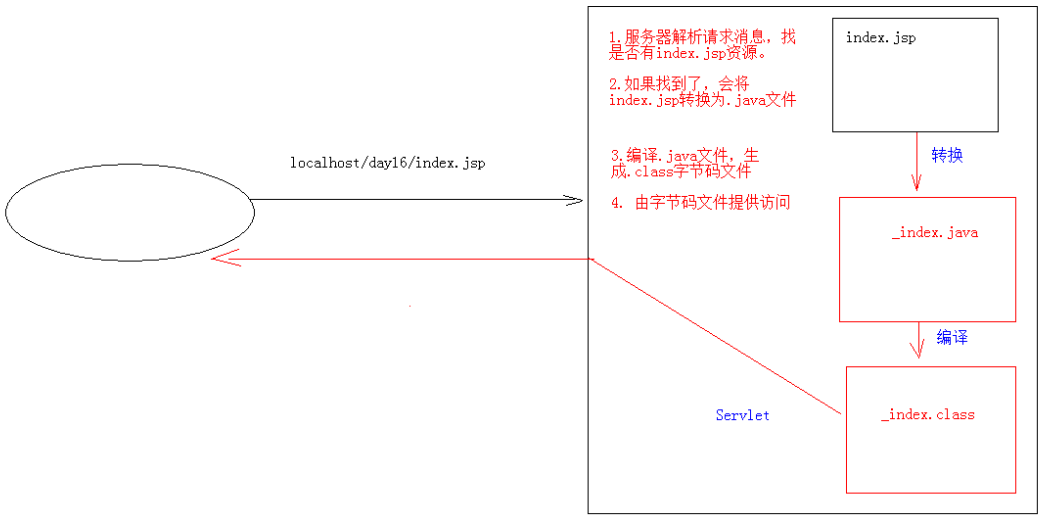
JSP：入门学习

概念：

- Java Server Pages： java服务器端页面
  - 可以理解为：一个特殊的页面，其中既可以指定定义html标签，又可以定义java代码
  - 用于简化书写！！！

原理

- JSP本质上就是一个Servlet

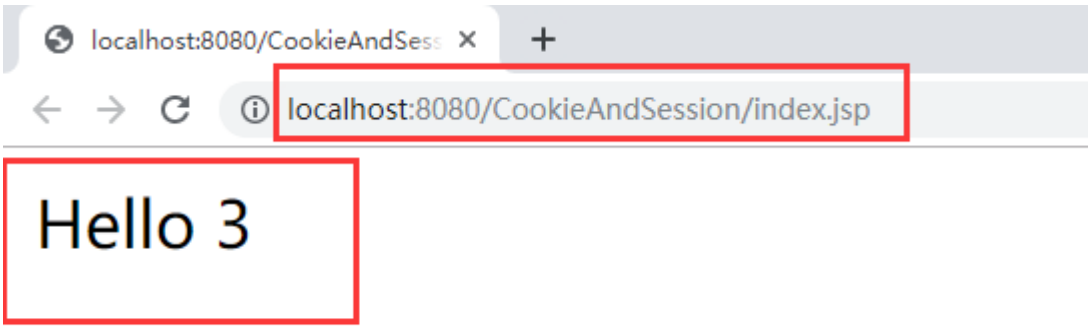


JSP的脚本：JSP定义Java代码的方式

1. <% 代码 %>：定义的java代码，在service方法中。service方法中可以定义什么，该脚本中就可以定义什么。
2. <%! 代码 %>：定义的java代码，在jsp转换后的java类的成员位置。
3. <%= 代码 %>：定义的java代码，会输出到页面上。输出语句中可以定义什么，该脚本中就可以定义什么。

```
<%--
  Created by IntelliJ IDEA.
  User: wanghongzhao
  Date: 2019/5/15
  Time: 10:13
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title></title>
  </head>
  <body>
```

```
        <%--定义的脚本里面的代码在service方法中，service中可以些什么这里就可以些什么--%>
        <%
          response.getWriter().write("Hello");
        %>

        <%--定义的是成员变量或者成员方法，反正就是定义成员xxx，但是在servlet尽量不要定义成员xxx，可能会有安全问题--%>
        <%!
          int i = 3;
        %>

        <%--相当于是一个输出语句，在这里会把里面的值输出到界面上--%>
        <%=
          i
        %>

    </body>
</html>
```



JSP的内置对象：

- 在jsp页面中不需要获取和创建，可以直接使用的对象
- jsp一共有9个内置对象。
- 今天学习3个：
    - request
    - response
    - out：字符输出流对象。可以将数据输出到页面上。和response.getWriter()类似
        - response.getWriter()和out.write()的区别：
            - 在tomcat服务器真正给客户端做出响应之前，会先找response缓冲区数据，再找out缓冲区数据。
            - response.getWriter()数据输出永远在out.write()之前
            - 也就是说，不论response.getWriter()在jsp哪里定义的，都会先于out.write()输出。
        - 但是response.getWriter()写在JSP界面中会优先输出，所以会影响布局，所以没啥要求就用out输出

案例:改造Cookie案例

```
<%@ page import="java.io.UnsupportedEncodingException" %>
<%@ page import="java.util.Date" %>
<%@ page import="java.text.SimpleDateFormat" %>
<%@ page import="java.net.URLEncoder" %>
<%@ page import="java.net.URLDecoder" %>
<%--
  Created by IntelliJ IDEA.
  User: wanghongzhao
  Date: 2019/5/15
  Time: 10:13
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title></title>
  </head>
  <body>


      <%!
        private String encode() throws UnsupportedEncodingException {
          Date date = new Date();
          SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy年MM月dd
日 HH:mm:ss");
          String format = simpleDateFormat.format(date);

          String encode = URLEncoder.encode(format, "utf-8");
          return encode;
        }

        private String decode(String encode) throws UnsupportedEncodingException
{
          String decode = URLDecoder.decode(encode, "utf-8");
          return decode;
        }
      %>

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


        Cookie[] cookies = request.getCookies();

        boolean flag = false;
        Cookie cookie1 = null;

        for (Cookie cookie : cookies) {
```

```
            if ("time".equals(cookie.getName())){
              flag=true;
              cookie1=cookie;
            }

          }
          if (flag){

            String decode = decode(cookie1.getValue());

            cookie1.setValue(encode());

            cookie1.setMaxAge(60*60*24*30);

            response.addCookie(cookie1);
            out.write("欢迎再次登录，您上次登陆的时间为："+decode);

          }else {
            Cookie time = new Cookie("time", encode());
            time.setMaxAge(60*60*24*30);
            response.addCookie(time);
            out.write("欢迎您首次登陆");
          }
        %>

    </body>
</html>
```

Session：翻译是-->主菜

概念

- 服务器端会话技术，在一次会话的多次请求间共享数据，将数据保存在服务器端的对象中。HttpSession

快速入门：

获取HttpSession对象：

- HttpSession session = request.getSession();

使用HttpSession对象：

- Object getAttribute(String name)
- void setAttribute(String name, Object value)
- void removeAttribute(String name)

> *注意，这个是一次性对话，虽然可以再一次会话的多次请求间共享数据吗，但是也仅限于浏览器存活的这段时间*

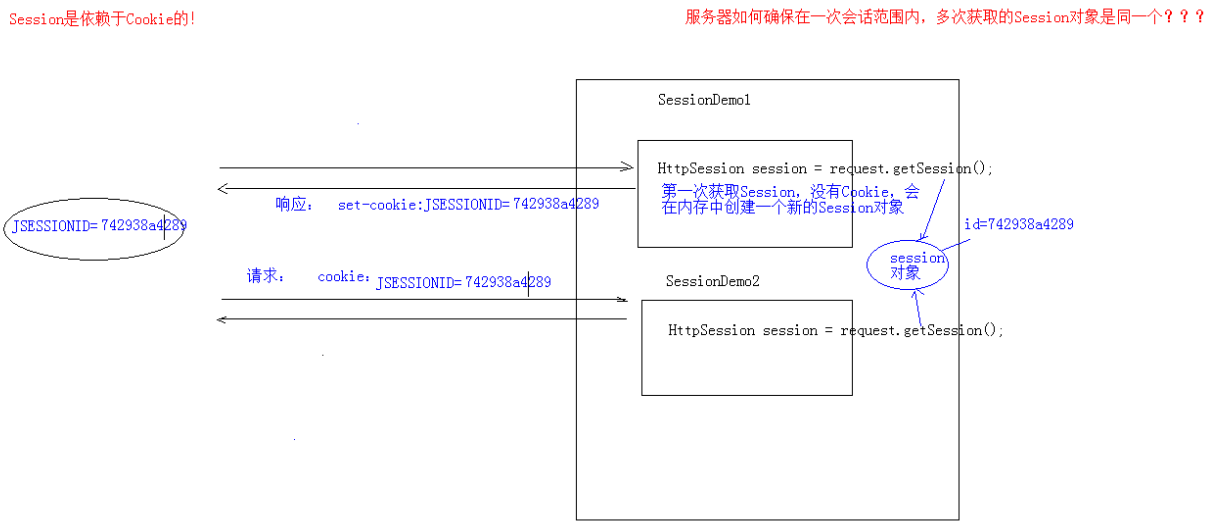
```
package cn.web;

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 javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/sessionDemo1")
public class SessionDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*获取session*/
        HttpSession session = request.getSession();
        /*存储数据*/
        session.setAttribute("message","hello");
    }

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

```
package cn.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 javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/sessionDemo2")
public class SessionDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*获取session*/
        HttpSession session = request.getSession();
        /*取得数据*/
        Object message = session.getAttribute("message");

        System.out.println(message);
```

```
    }

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

原理

- Session的实现是依赖于Cookie的。
    1. 客户端-->服务器
    2. 服务器响应，设置一个cookie返回：set-cookie:JESSIONID=session的ID，浏览器获取到这个cookie（JESSIONID=XXX）
    3. 客户端再次向服务器访问会带着这个cookie，以此来确保获取到的Session是一个



细节：

当客户端关闭后，服务器不关闭，两次获取session是否为同一个?

- 默认情况下。不是。

  因为session是基于cookie头来获取的，浏览器关闭后，cookie都没了，session自然也获取不是同一个了
- 如果需要相同，则可以创建Cookie,键为JSESSIONID，设置最大存活时间，让cookie持久化保存。

```
package cn.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.*;
import java.io.IOException;

@WebServlet("/sessionDemo1")
public class SessionDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        HttpSession session = request.getSession();

        Cookie sessionid = new Cookie("JSESSIONID", session.getId());

        System.out.println(session.getId());

        sessionid.setMaxAge(60*60);

        response.addCookie(sessionid);
    }

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

/*
7B6FFF79A17F15B570641DBB7B40BC4E
7B6FFF79A17F15B570641DBB7B40BC4E
*/
```

那么JSESSIONID=7B6FFF79A17F15B570641DBB7B40BC4E

客户端不关闭，服务器关闭后，两次获取的session是同一个吗?

- 不是同一个，但是要确保数据不丢失。tomcat自动完成以下工作
  - session的钝化：
    - 在服务器正常关闭之前，将session对象序列化到硬盘上成为一个文件
  - session的活化：
    - 在服务器启动后，将session文件转化为内存中的session对象即可。

注意，session的钝化和活化都是tomcat自动做了的，但是有一个点就是intellj idea只能钝化，不能活化，因为intellj idea在服务器重启之后会将那个目录删除然后再创建一个新的，所以原来的文件就读取不了了，所以将来部署项目的时候别用idea

session什么时候被销毁?

1. 服务器关闭
2. session对象调用invalidate()，自杀。

3. session默认失效时间 30分钟 选择性配置修改，在tomcat-->conf-->web.xml-->session-config

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

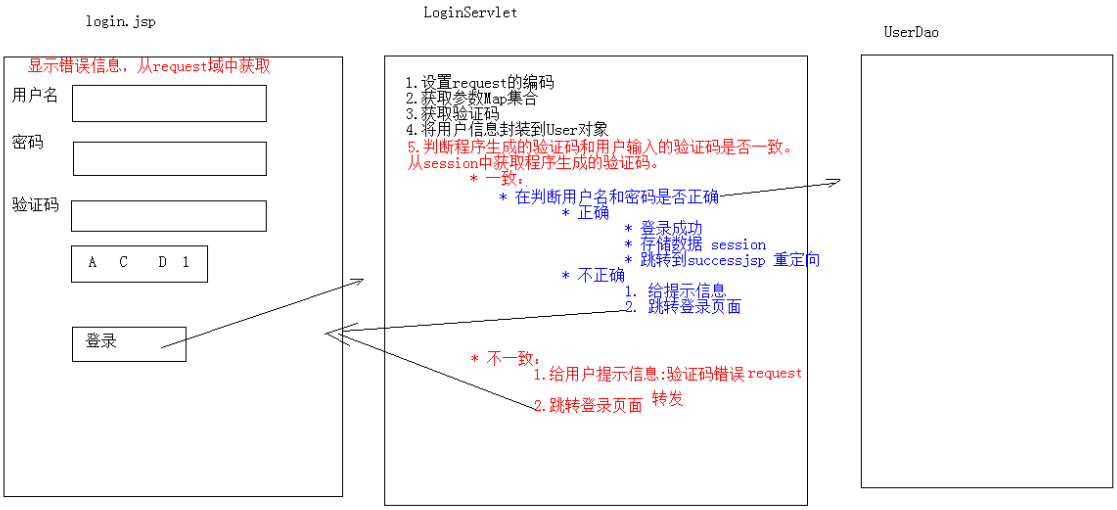
session的特点

1. session用于存储一次会话的多次请求的数据，存在服务器端
2. session可以存储任意类型，任意大小的数据

- session与Cookie的区别：
    1. session存储数据在服务器端，Cookie在客户端
    2. session没有数据大小限制，Cookie有
    3. session数据安全，Cookie相对于不安全

案例：验证码

1. 案例需求：
    1. 访问带有验证码的登录页面login.jsp
    2. 用户输入用户名，密码以及验证码。
        - 如果验证码输入有误，跳转登录页面，提示：验证码错误(先判断验证码，如果错了那么对数据库就没有开销，注意验证码忽略大小写)
            - 从servlet显示到客户端和客户端输入验证码提交这是两次请求
            - 把验证码存到session里面，然后和用户输入的验证码进行比对
        - 如果用户名和密码输入有误，跳转登录页面，提示:用户名或密码错误
        - 如果全部输入正确，则跳转到主页success.jsp，显示：用户名,欢迎您
2. 分析：




```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title>login</title>
  </head>
  <body>

  <form action="/checkCode/checkCodeServletText" method="post">
    <table>
      <tr>
        <td><input id="username" name="username" type="text" placeholder="请输入用户名"></td>
        <td><input id="password" name="password" type="password" placeholder="请输入密码"></td>
      </tr>
      <tr>
        <td><image id="image" src="/checkCode/checkCodeServlet"></image></td>
      </tr>
      <tr>
        <td><input id="checkCodeText" name="checkCodeText" type="text" placeholder="验证码"></td>
      </tr>
    </table>

    <input type="submit">
  </form>

  <script>

    document.getElementById("image").onclick = function (ev) {

        this.src="/checkCode/checkCodeServlet?"+new Date();
    }
```

```
  </script>

  </body>
</html>
```

```
package cn.web.servlet;

import javax.imageio.ImageIO;
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 javax.servlet.http.HttpSession;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.Random;

@WebServlet("/checkCodeServlet")
public class checkCodeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        checkCode(request,response);
    }

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


    private void checkCode(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        String str =
"QAZWSXEDCRFVTGBYHNUJMIKOLPqazwsxedcrfvtgbyhnujmikolp0123456789";
        Random random = new Random();
        StringBuffer stringBuffer = new StringBuffer();


        int width = 100,
                height = 50;

        BufferedImage bufferedImage = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);

        Graphics graphics = bufferedImage.getGraphics();

        graphics.setColor(Color.pink);
```

```
        graphics.fillRect(0,0,width,height);

        graphics.setColor(Color.GRAY);
        graphics.drawRect(0,0,width-1,height-1);

        graphics.setColor(Color.BLACK);
        for (int i = 1; i <= 4; i++) {
            char c = str.charAt(random.nextInt(str.length()));
            stringBuffer.append(c);
            graphics.drawString(c+"",width/5*i,height/2);
        }


        String checkCodeText = stringBuffer.toString();

        HttpSession session = request.getSession();

        session.setMaxInactiveInterval(60*60);

        session.setAttribute("checkCodeText",checkCodeText);

        ImageIO.write(bufferedImage,"jpg",response.getOutputStream());
    }
}
```

```
package cn.web.servlet;

import cn.table.User;
import cn.util.JDBCUtils;
import org.springframework.jdbc.core.JdbcTemplate;

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 javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/checkCodeServletText")
public class checkCodeServletText extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

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

        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String checkCodeText = request.getParameter("checkCodeText");
```

```
        if (checkCodeText!=null && checkCodeText!=""){
            HttpSession session = request.getSession();

            String str = (String)session.getAttribute("checkCodeText");

            if (str.equalsIgnoreCase(checkCodeText)){
                if ("zhangsan".equals(username) && "123".equals(password) ){

                    response.getWriter().write("登陆成功");
                }else{
                    response.getWriter().write("用户名或密码不正确");
                }
            }else {
                response.getWriter().write("验证码输入不正确");
            }
        }

    }

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

}
```